RESEARCH ON COMPUTING SCIENCE

# RECONOCIMIENTO DE PATRONES AVANCES Y PERSPECTIVAS

**Juan Luis Díaz** de **León** antiago
**Cornelio Yáñez** Márquez {Eds.)

Centro de Investigación en Computación
Instituto Politécnico Nacional

# Reconocimiento de Patrones
## Avances y Perspectivas

# Research on computing science

Editorial Board

LUIS HERNÁNDEZ LARA, CIC
EDUARDO GÓMEZ RAMÍREZ, ULSA
JOSÉ RUIZ SHULCLOPER, ICIMAF
ALEJANDRO FLORES MÉNDEZ, ULSA
JESÚS FIGUEROA NAZUNO, CIC
MARCO MORENO ARMENDÁRIZ, ULSA
MARIO FARÍAS ELINOS, ULSA

Edition:
Flavio A. Sánchez Garfias

Cover design:
Ignacio García Araoz

Volume 1

# Reconocimiento de Patrones
## Avances y Perspectivas

*Editado por*

**Juan Luis Díaz de León Santiago**
*Director del Centro de Investigación en Computación*
*Instituto Politécnico Nacional, México*

*y*

**Cornelio Yáñez Márquez**
*Investigador del Centro de Investigación en Computación*
*Instituto Politécnico Nacional, México*

Centro de Investigación en Computación
Instituto Politécnico Nacional
**México**

ISBN ISBN 970189476-6

Copyright © by Instituto Politécnico Nacional

Published by Instituto Politécnico Nacional
Centro de Investigación en Computación
Av. Juan de Dios Bátiz s/n esq. M. Othón de Mendizábal
Unidad Profesional "Adolfo López Mateos", Zacatenco
07738, México, D.F., MÉXICO

http://www.cic.ipn.mx

Coordination: Blanca Gómez Galindo
Luis Hernández Lara
Carmen Rodríguez Aparicio

The editors and publisher of this book have made their best effort in preparing this book, but make no warranty of any kind, expressed or implied, with regard to the documentation contained in this book

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior permission of Instituto Politécnico Nacional.

Printing: 500
Printed in México

# Contenido

**Introducción** ix

**Algoritmos Genéticos**

Cellular Neural Networks Learning using Genetic Alorithm
*E. Gómez-Ramírez & F. Mazzanti* 1

**Hardware**

Modelo Simplificado de CNN para el Rápido Procesamiento de Imágenes Binarias en Estructuras Programables
*J. E. Molinar-Solís, F. Gómez-Castañeda & J. A. Moreno-Cadenas* 15

**Morfología Matemática**

Minimal Representations and Morphological Associative Memories for Pattern Recall
*Gerhard X. Ritter* 27

Aplicación de la Morfología Matemática a la Telemedicina y la Biotecnología: Caracterización Morfológica de Células de la Sangre y Análisis de *cDNA Microarrays*
*Jean Serra* 39

Filtros Morfológicos por Pendiente basados en Criterios de Gradiente Ponderado
*J. D. Mediola-Santibáñez & I. R. Terol-Villalobos* 51

Reconocimiento de Símbolos Musicales usando una Combinación de Técnicas de PDI y RNA
*Raúl Cruz-Barbosa* 63

## Procesamiento y Análisis de Imágenes

Normalized Similarity Measure of 2D Objects and their Morphing
*Hermilo Sánchez*

Clasificación, Segmentación y Reconstrucción de Líneas en Mapas Temáticos a Color
*Joaquín Decelis-Burguete*

Estrategias Evolutivas Aplicadas a Registro de Imágenes
*H. F. Gómez-G., A. Hernández-A., J. L. Marroquín-Z. & S. Botello-R*

Mejoramiento Selectivo Condicional: Una Estrategia para el Mejoramiento de Imágenes Digitales en Colores Impresas en Semitonos
*E. M. Felipe-Riverón & M. E. Ramos-García*

Segmentation of Numerous Tiny Objects in Very Noisy Gray-level Images
*I. Castellanos-Bisset, E. M. Felipe-Riverón & Leudis Sánchez*

Intelligent Segmentation in the Task of Raster-to-vector Conversion of Color Cartographic Images
*S. Levachkine, A. Velázquez, M. Torres & M. Moreno*

## Procesamiento y Análisis de Señales

Filtrado en Tiempo Real basado en Estimadores Robustos
*F. J. Gallegos-Funes, V. I. Ponomaryov, O. B. Pogrebnyak & L. N. de Rivera*

## Redes Neuronales

Arquitectura Neuronal Convolucional para la Detección de Defectos en Productos en Continuo
*J. A. Calderón-Martínez & P. Campoy-Cervera*

Incremental Construction of LSTM Recurrent Neural Network
*Sabrine Ribeiro & René Alquézar*

Agrupador de Complejos QRS basado en Aprendizaje Competitivo Selectivo
*Pablo Aguirre, Juan Cardelino, Nicolás Loeff & Franco Simini*

Identificación de Células Cervicales a través de Redes Neuronales Pulsantes
*A. Barreto-Flores, L. Altamirano-Robles & O. Pérez-Chalini*

Using Neural Networks for Pattern Recognition in Robotic Tasks
*I. López-Juárez & J. Corona*

Cellular Neural Networks and Mobile Robot Vision
*Marco Balsi & X. Vilasís-Cardona* 229

**Reconocimiento de Patrones**

Formación Integral de Especialistas en Reconocimiento de Patrones
*José Ruiz-Shulcloper* 245
Seismic Pattern Recognition using Wavelet Transform and Seismic Attributes
*Margarita Fernández, Adriana Mavilio, J. R. Jiménez & Nieves Henríquez* 257
Un Algoritmo Incremental para la Obtención de Particiones con Datos Mezclados
*A. Pons-Porrata, J. Ruiz-Shulcloper, R. Berlanga-Llavori & Y. Santiesteban-Alganza* 265
Combining Evolution Techniques to Estimate Features'weights and the Support Sets System for ALVOT
*J. A. Carrasco-Ochoa & J. F. Martínez-Trinidad* 277
LCARS un Lenguaje de Especificación de Problemas en Reconocimiento de Patrones bajo el Enfoque Lógico-Combinatorio
*S. Godoy-Calderón, J. F. Martínez-Trinidad & M. Lazo-Cortés* 289
*Chunking* Suave: Un Método de Descomposición para el Entrenamiento de Máquinas de Soporte Vectorial
*R. del Ángel Pérez-Flores* 301
Teoría para Reconocimiento de Patrones Colaborativo
*A. V. de la Cruz-Rivera & M. de los A. Alonso-Lavernia* 313
Reconocimiento de Patrones Auto-Supervisado. Método de las Clases Cerradas
*A. V. de la Cruz-Rivera & M. de los A. Alonso-Lavernia* 325
Matriz de Conocimiento sobre la Complejidad de Predicción en Series de Tiempo
*E. Bautista-Thompson & J. Figueroa-Nazuno* 337
Análisis de Patrones aplicado a Problemas de Teoría de Números: El Problema de Collatz
*E. Bautista-Thompson, A. Villanueva-Becerril & J. Figueroa-Nazuno* 349
Time Series Modeling using Recurrence Plots and Face Recognition Techniques
*J. M. Medina-Apodaca, J. Figueroa-Nazuno & S. García-Benítez* 359
Distributed Bayesian Networks for Patterns Representation
*J. M. Medina-Apodaca & J. Figueroa-Nazuno* 371
Algoritmo de Agrupamiento GLC Paralelo
*R. Gil-García & J. M. Badía-Contelles* 383
Análisis Comparativo de Tres Sistemas de la Programación Lógica Inductiva
*Hilda Escobedo-Hernández* 395

Un Algoritmo Incremental para la Obtención de Cubrimientos con Datos Mezclados
*A. Pons-Porrata, J. Ruiz-Shulcloper, R. Berlanga-Llavori & Y. Santiesteban-Alganza* 405

Desarrollo de un *Framework* de Clasificación de Patrones
*I. Ramírez, A. Alcarraz, A. Fernández & A. F. de Oliveira* 41

Lernmatrix de Steinbuch: Condiciones Necesarias y Suficientes para Recuperación Perfecta
*F. A. Sánchez-Garfias, J. L. Díaz-de-León-S. & C. Yáñez-Márquez* 437

Clasificador Híbrido de Patrones basado en la Lernmatrix de Steinbuch y el Linear Associator de Anderson-Kohonen
*R. Santiago-Montero, J. L. Díaz-de-León-S. & C. Yáñez-Márquez* 449

**Sistemas Híbridos**

Automated Quality Control in Sound Speaker Manufacturing using Soft Computing Techniques and Fractal Theory for Pattern Recognition
*Patricia Melin & Oscar Castillo* 461

A Hybrid Intelligent Approach for Automated Quality Control Combinig Learning Vector Quantization Neural Networks and Fuzzy Logic
*O. Castillo, R. Sepúlveda, O. Montiel & P. Melin* 473

**Índice de Autores** 485

# Introducción

Este libro contiene una selección, con arbitraje internacional, de los artículos que fueron presentados en el *VII Congreso Iberoamericano de Reconocimiento de Patrones*, CIARP 2002, llevado a cabo en las instalaciones del Centro de Investigación en Computación del Instituto Politécnico Nacional de México, del 19 al 22 de noviembre de 2002.

Uno de los propósitos fundamentales del CIARP es brindar a la comunidad internacional involucrada en Reconocimiento de Patrones, un espacio de alto nivel para la discusión de los recientes avances en las áreas de interés. En cada una de las versiones del congreso, los participantes han tenido la invaluable oportunidad de escuchar, ver y realizar presentaciones especializadas, al mismo tiempo que se relacionan con prestigiados colegas de diferentes latitudes del orbe.

libro contiene 39 trabajos distribuidos en las siguientes grandes áreas: algoritmos genéticos, hardware, morfología matemática, procesamiento y análisis de imágenes, procesamiento y análisis de señales, redes neuronales, reconocimiento de patrones y sistemas híbridos. El espectro geográfico de los autores incluye varios países de América Europa: Cuba, España, Estados Unidos, Francia, Grecia, Italia, Uruguay y, por supuesto, los Estados Unidos Mexicanos. Nuestro agradecimiento a todos los autores y miembros de los comités organizadores, nacional e internacional, y técnico.

Esperamos que el material presentado en este libro cumpla las expectativas de investigadores y estudiantes de maestría y doctorado cuyo trabajo involucre alguna o algunas de las áreas afines al reconocimiento de patrones. Pretendemos que los lectores tengan a la mano un diagnóstico preciso y bien documentado de los avances recientes en Reconocimiento de Patrones y de las perspectivas de desarrollo a corto y mediano plazo en los temas afines.

Deseamos manifestar nuestro agradecimiento por el apoyo brindado, para el desarrollo de este trabajo, a las siguientes instituciones: Instituto Politécnico Nacional, Universidad La Salle de México, COFAA y Secretaría Académica del IPN, CONACyT y Sistema Nacional de Investigadores.

Juan Luis Díaz de León Santiago
Cornelio Yáñez Márquez

In this work a genetic algorithm approach is developed to compute all the parameters of a Cellular Neural Network (weights and current). The results show that it is possible to find many different matrixes that fulfill the objective condition. Different examples of image processing are presented to illustrate the procedure developed

The structure of the papers is the following: Section 2 describes the theory of Cellular Neural Networks, section 3 the genetic algorithm proposed and in section 4 the way that the algorithm was applied to find the optimal parameters for image processing.

## 2. CNN

From its origins in 1987 the CNN became a very powerful algorithm, and right now there are many research groups in the area. As it was defined on its origins, a CNN is an identical array of non-linear dynamic circuits. This array could be of different types depending on the way they are connected or the class of the neighborhood.

### 2.1 Mathematical Model

The simplified mathematical model that defines this array [7] is the following:

$$\frac{dx^c}{dt} = -x^c(t) + \sum_{d \in N_r(c)} a_d^c y^d(t) + \sum_{d \in N_r(c)} b_d^c u^d + i^c \qquad (1)$$

$$y^c(t) = \frac{1}{2}\left(\left|x^c(t)+1\right| - \left|x^c(t)-1\right|\right) \qquad (2)$$

where $x^c$ represents the state of the cell c, $y^c$ the output and $u^c$ its input. The state of each cell is controlled by the inputs and outputs of the adjacent cells inside the r-neighborhood N(c). The outputs are feedback and multiplied by $a^c_d$ and the inputs are multiplied by control parameters $b^c_d$. The value $i^c$ is constant and is used for adjusting the threshold[1]. These coefficients are invariants to translations and it would be called cloning template.

The output $y_c$ is obtained applying the equation (2). This equation limits the value of the output on the range between [-1,1], in the same way the activation function does in the classical ANN models (Fig. 1).

[1] This coefficient is equivalent to the input bias on other nets.

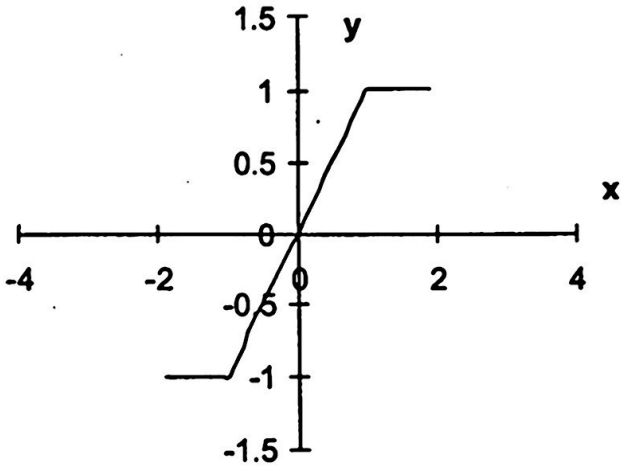



*Figure 1 Output function*

## 2.2 Discrete Mathematical Model

The mathematical model illustrated before could be represented by the following expression:

$$\frac{dx^c}{dt} = -x^c(t) + k^c \tag{3}$$

where:

$$k^c = \sum_{d \in N_r(c)} a_d^c y^d(t) + \sum_{d \in N_r(c)} b_d^c u^d + i^c$$

Because the final state is obtained evaluating x(∞), the model could be simplified as:

$$x^c(k) = \sum_{d \in N_r(c)} a_d^c y^d(k) + \sum_{d \in N_r(c)} b_d^c u^d + i^c \tag{4}$$

Some authors also considered a modification on the output (eq. 2), as follows [4]:

$$y^c(k) = f(x^c(k-1))$$

$$y^c(k) = \begin{cases} 1 & si \;\; x^c(k-1) > 0 \\ -1 & si \; x^c(k-1) < 0 \end{cases} \tag{5}$$

However, the simplification made (eq. 4) could be used with the function of the eq. 2 [8]. Equation 4 can be represented using the convolution operator like:

$$X_k = A * Y_k + B * U_k + I \tag{6}$$

where matrix A and B correspond to the rotation of 180° of the corresponding coefficients $a^c_d$ and $b^c_d$. The previous step it is not necessary if the matrix has the property of $A=A^T$.

The cloning template in some cases, is a set of different combinations that fulfill the same conditions, e. g., there are different matrix that can be solve the same problem. This can be a problem with the use of gradient error techniques, because in these cases, it is not possible to find an optimal solution of the learning problem. One alternative is the use of evolutionary computation techniques that can be applied to find the different solutions or alternatives to the same problem.

The next section describes the methodology of Genetic Algorithm used in this work.

## 3. Genetic Algorithm

Genetic Algorithm (GA) is a knowledge model inspired on some of the evolution mechanisms that are observed in nature. GA usually follows the next cycle [9][10]:

- Generation of an initial population in a random way.
- Evaluation of the fitness or some objective function of every individual that belongs to the population.
- Creation of a new population by the execution of operations like crossover and mutation over individuals which fitness or profit value has been measured.
- Elimination of the former population and iterate using the new one until the termination criteria is fulfilled or it's reached a certain number of generations.

In this work the crossover or sexual recombination, the mutation and other special process that we call add parents [11] and add random parents are used. Next Sections describe these processes.

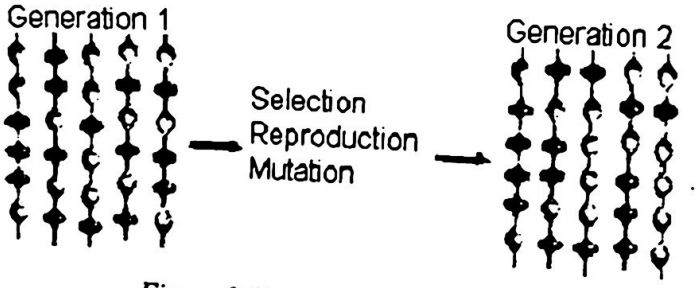



*Figure 2 Operators of Genetic Algorithm*

### 3.2 Crossover

The crossover operator is characterized by the combination of the genetic material of the individuals that are selected in function of the good performance (objective function).

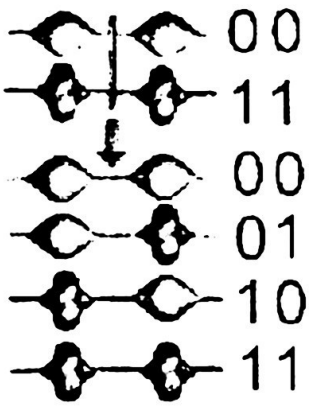



*Figure 3 Multipoint Crossover Operator*

To explain the multipoint crossover for each fixed number $g=1,...,n_g$, where $n_g$ the number of total generations, let introduce the matrix $F_g$ which is the set of parents of a given population. This matrix is boolean of dimension $F_g : n_p x n_b$, where $n_p$ is the number of parents of the population at the generation $g$ and $n_b$ is the size of every array (chromosomes). Let $C(F_g, n_t)$ be the crossover operator which can be defined as the combination between the information of the parents set considering the number of intervals $n_t$ of each individual and the number of sons $n_s$ such that:

$$n_s = n_p^{n_t} \tag{7}$$

then $C(F_g, n_t): n_p x n_b \rightarrow n_s x n_b$. To show how the crossover operator can applied the following example is introduced. Let $F_g$ has $n_p=2$ and $n_t=2$. This means that the array (the information of one father) is divided in 3 sections and every section is determined with $a_i$ and $b_i$ respectively for $i=1,...,n_t$. It's important appoint that with this operator the parents $F_g$ of the population $g$ are included the result of the crossover:

**Example:**

$$F_g = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix}, M_1 = [a \quad b], M_2 = [c \quad d]$$

$$\Rightarrow C(F_g) = \begin{bmatrix} [a & b] \leftarrow \\ [a & d] \\ [c & b] \\ [c & d] \leftarrow \end{bmatrix}$$

Note that with this operator the parents $F_g$ of the population $g$ are included in the result of the crossover.

For the problem of CNN learning every section of one father can be represented by one column or one row of one matrix of the cloning template.

### 3.3 Mutation

The mutation operator just changes some bits that were selected in a random way from a fixed probability factor $P_m$; in other words, we just vary the components of some *genes*. This operator is extremely important, because assures the maintenance of the diversity inside the population, which is basic for the evolution. This operator $M : n_s x n_b \rightarrow n_s x n_b$ changes with probability $P_m$ a specific population in the following way:

$$M(F_{ij}, P_m) = \begin{cases} \overline{F}_{ij} & r(\omega) \leq P_m \\ F_{ij} & r(\omega) > P_m \end{cases} \tag{8}$$

### 3.4 Add Parents

In this part the parents $F_g$ are added to the result of mutation process, then the population $A_g$ at the generation $g$ can be obtained like:

$$A_g = \begin{bmatrix} M(C(F_g)) \\ F_g \end{bmatrix} \tag{9}$$

This step and the previous one ensure the convergence of the algorithm, because every generation at least hast the best individual obtained in the process.

With this step the original and mutated parents are included in the generation $g$ and with some probability the individuals tend to the optimal one.

### 3.5 Selection Process

The Selection Process $S_g$ computes the objective function $O_g$ that represents a specific criterion to maximize or minimize and selects the best $n_p$ individuals of $A_g$ as:

$$S_g(A_g, n_p) = \min^{n_p} O_g(A_g) \tag{10}$$

Then, the parents of the next generation can be calculated by:

$$F_{g+1} = S_g(A_g, n_p) \tag{11}$$

In resume the Genetic Algorithm can be describe with the following steps:

1. For the initial condition $g=0$ select the $A_0$ in random way, such that $A_0 : n_s x n_b$
2. Compute $F_1 = S_0(A_0)$
3. Obtain $A_g$
4. Calculate $S_g$
5. Return to step 3 until the maximal number of generations is reached or one of the individuals of $S_g$ obtain the minimal desire value of $O_g$

### 3.6 Add Random Parents

To avoid local minima a new scheme is introduced and it is called add random parents. If the best individual of one generation is the same than the previous one, a new random individual is included like parent of the next generation. This step increases the population because when crossover is applied with the new random parent, the number of sons increases. This step is tantamount to have a very big mutation probability and to search in new one points of the solution space.

# 4. Application in image processing

The procedure used to train the network was:

1. To generate the input patterns using a known matrix in image processing to find the different kind of lines (horizontal, vertical, diagonal) in different images.
2. To train the network using the methodology above explained using GA
3. Test the results using a new pattern

The results were repeated to obtain different templates for the same problem. The parameters used are: 4000 individuals in the initial population and 5 parents. The change in the values for every coefficient was discretized using an increment of 0.125 to reduce the space solution.

### 4.2 Example 1 Horizontal Line Detector

Figures 4 and 5 show the input and output patterns used. This patterns was obtained using a well known matrix:

$$M = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 2 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

The parameters obtained are:

$$A = \begin{bmatrix} -0.125 & 0 & 0 \\ 0 & 2.5 & 0.125 \\ -0.75 & -1.375 & 0.125 \end{bmatrix}, B = \begin{bmatrix} -0.125 & -2 & 0 \\ 0.375 & 2.75 & 0 \\ 0.375 & -0.875 & -0.25 \end{bmatrix}$$

$I=-2.875$

$$A = \begin{bmatrix} 4.5 & -2.25 & -0.5 \\ -1.625 & 0.75 & 3.375 \\ 2.125 & -0.875 & -0.875 \end{bmatrix}, B = \begin{bmatrix} 0.75 & -7.125 & -0.625 \\ 2.625 & -0.875 & 2.25 \\ -3.125 & -0.875 & 2.375 \end{bmatrix}$$

$I=0.625$

### 4.3 Example 2 Vertical Line Detector

Figures 6 and 7 show the input and output patterns used. This patterns was obtained using a well known matrix:

$$M = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 2 & -1 \\ 0 & 0 & 0 \end{bmatrix}$$

In this case we put constraints to select only the matrix B. The parameters obtained are:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & -0.5 & 0 \\ 0 & 0.625 & 0 \\ 0 & -0.125 & 0 \end{bmatrix}, I=0$$

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & -0.75 & 0 \\ 0 & 1.625 & 0 \\ 0 & -0.875 & 0 \end{bmatrix}, I=0$$

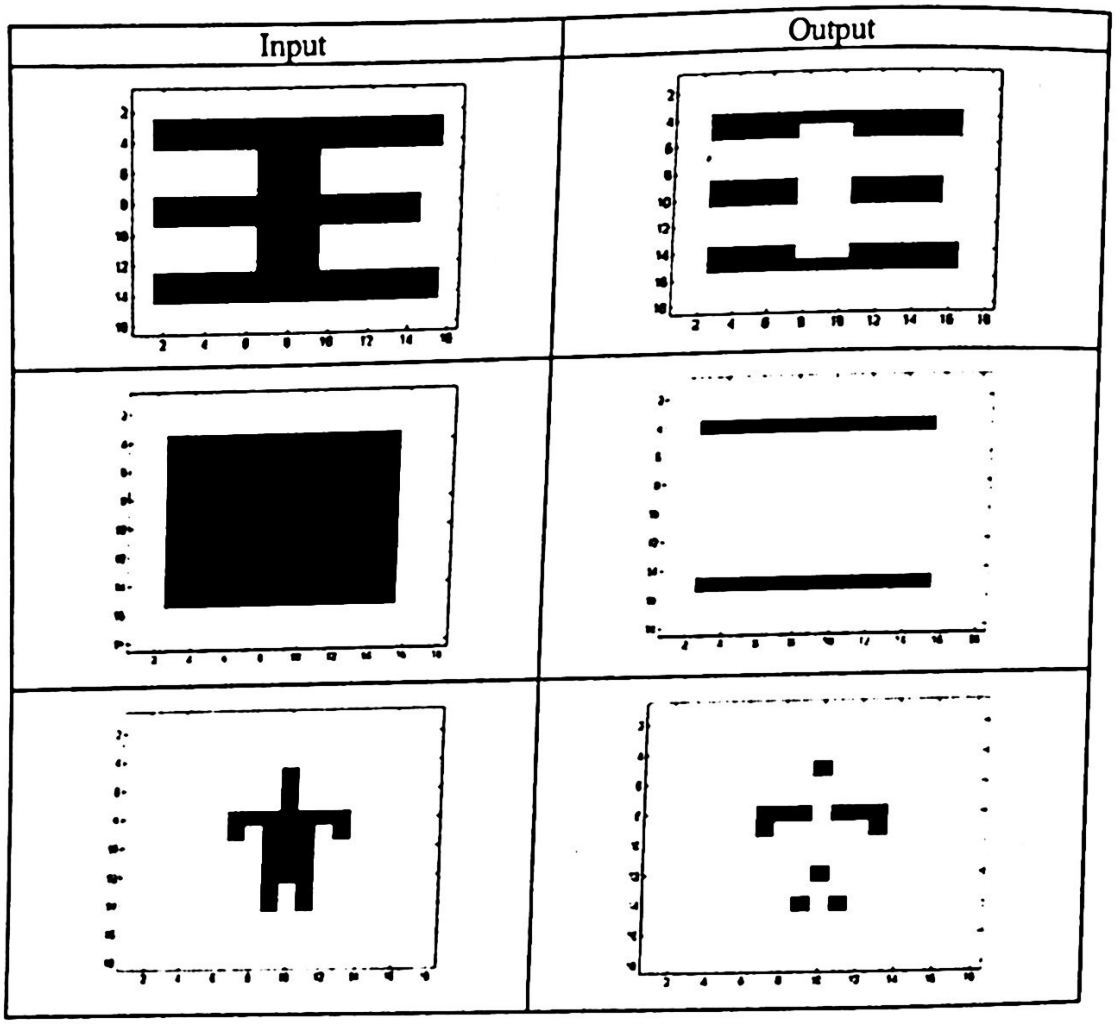



*Figure 4 Input and Output patterns for CNN Learning*

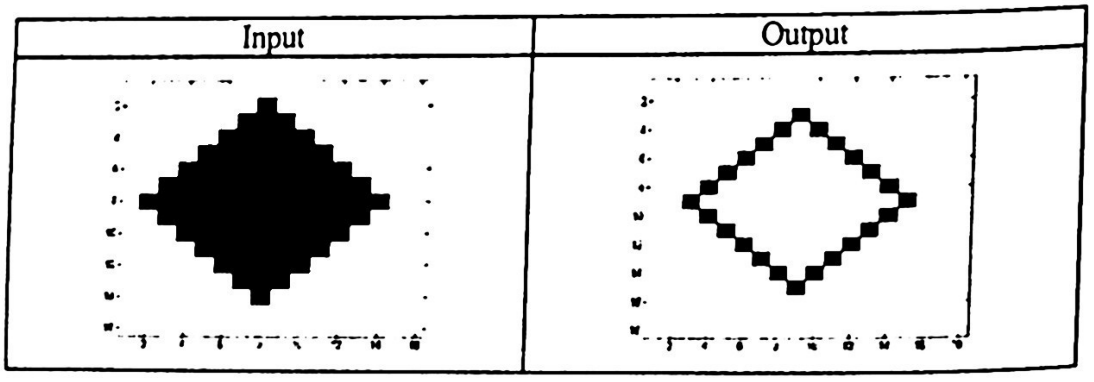



*Figure 5 Input and Output Patterns for CNN Test*

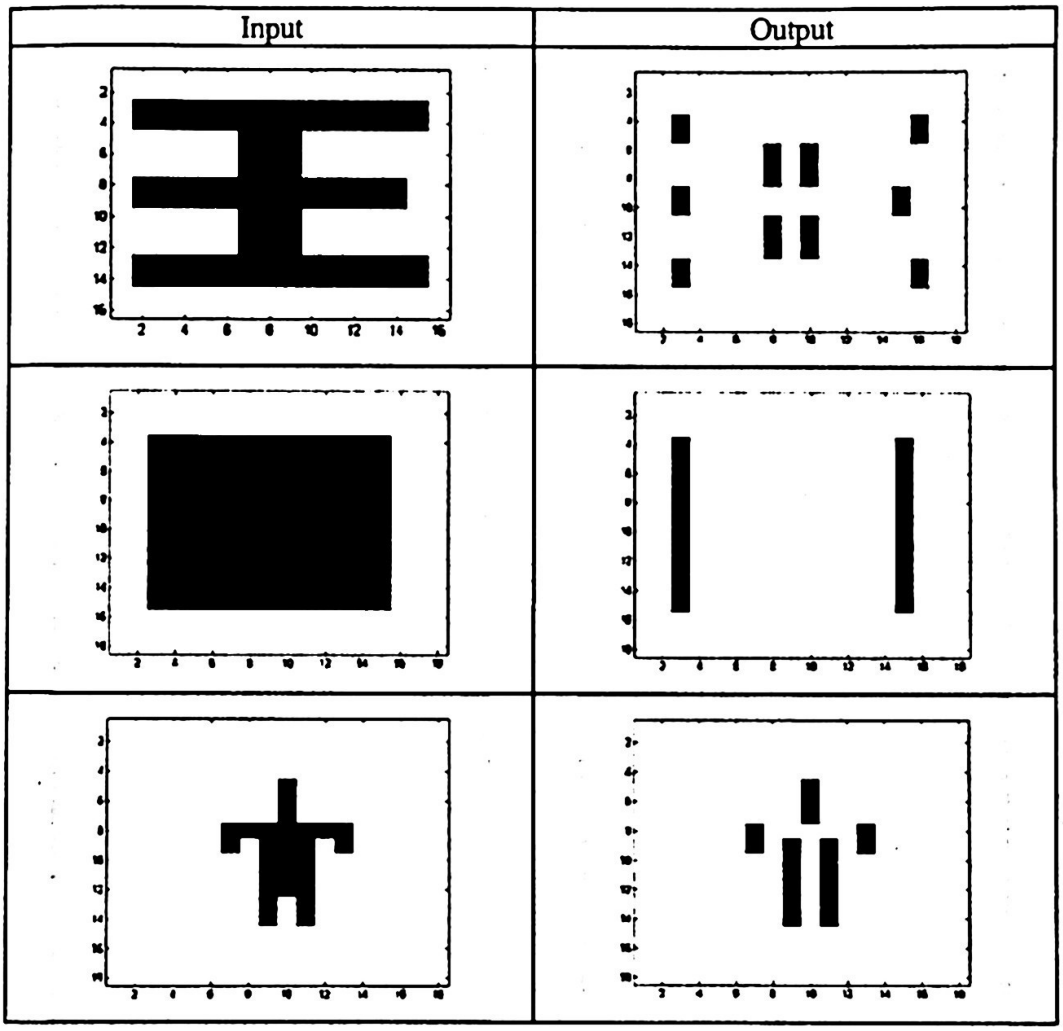



*Figure 6 Input and Output patterns for CNN Learning*

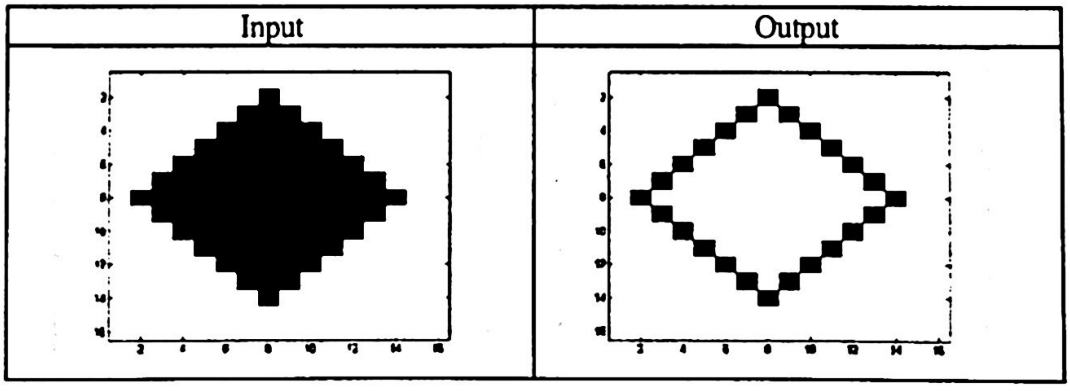



*Figure 7 Input and Output Patterns for CNN Test*

## 5. Conclusions

Genetic Algorithm is one alternative to the Learning of CNN that can be used to solve the problem of techniques that use the gradient error in the algorithm. In this work the use of add random parent avoid to reach local minima and improves the convergence of the algorithm. This is very important when the space solution includes all the elements of the cloning template.

## 6. References


[1] O. Chua & L. Yang (1988), "Cellular neural networks: Theory", *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1257-1272.

[2] O. Chua & L. Yang (1988),"Cellular neural networks: Applications", *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1273-1290.

[3] Mayol W, Gómez E. (1994), "2D Sparse Distruted Memory-Optical Neural Network for Pattern Recognition". *IEEE International Conference on Neural Networks*. Orlando, Florida, June28-July 2.

[4] Taraglio S. & Zanela A. (1996) , Cellular Neural Networks: a genetic algorithm for parameters optimization in artificial vision applications., *CNNA-96., Proceedings., 1996 Fourth IEEE International Workshop on*, 24-26 June 1996 , Seville, Spain, pp. 315-320.

[5] Destri, G., (1996), Discrete-time cellular neural network construction through evolution programs *Cellular Neural Networks and their Applications, CNNA-96., Proceedings, 1996 Fourth IEEE International Workshop on*, 24-26 June 1996 , Seville, Spain, pp. 473 – 478.

[6] Doan, M. D., Halgamuge, S., Glesner M. & Braunsforth, (1996), Application of Fuzzy, GA and Hybrid Methods to CNN Template Learning, *CNNA-96., Proceedings., 1996 Fourth IEEE International Workshop on*, 24-26 June 1996 , Seville, Spain pp. 327-332.

[7] Harrer H. & Nossek J., (1993), "Discrete-Time Cellular Neural Networks". *Cellular Neural Networks*. Edited by T. Roska and J. Vandewalle. John Wiley & Sons.

[8] M. Alencastre-Miranda, A. Flores-Méndez & E. Gómez Ramírez:, (1995) "Comparación entre Métodos Clásicos y Redes neuronales celulares para el análisis de imágenes". *XXXVIII Congreso Nacional de Física*. Zacatecas, Zacatecas, Mexico del 16 al 20 de octubre.

[9] Alander J. T., *An Indexed Bibliography of Genetic Algorithms*: Years 1957--1993, 1994, Art of CAD ltd.

[10] Bedner I., (1997), *Genetic Algorithms and Genetic Programming at Stanford* , Stanford Bookstore.

[11] E. Gómez-Ramírez, A. Poznyak, A. González Yunes & M. Avila-Alvarez. Adaptive Architecture of Polynomial Artificial Neural Network to Forecast Nonlinear Time Series. *CEC99 Special Session on Time Series Prediction.* Mayflower Hotel, Washington D.C., USA, July 6-9, 1999.



En este artículo, se presenta un nuevo modelo de CNN capaz ejecutar diferentes tareas de procesamiento en imágenes binarias, el cual explota las características que estas tareas presentan de separabilidad lineal. modelo permite realizar una neurona CNN con aproximadamente 50 transistores (casi la mitad que la reportada en [4]), en donde a diferencia de [5], absolutamente todas las señales son unipolares. El artículo está organizado de la manera siguiente: En la sección 2 se presenta el fundamento teórico del modelo. En la sección 3, el diseño eléctrico y geométrico (*layout*) de una neurona, y finalmente en la sección 4 se muestran algunos resultados de las simulaciones, así como en la sección 5 las conclusiones.

# 2. Fundamento teórico del modelo.

## 2.1 Tareas de procesamiento para imágenes binarias linealmente separables.

Desde su invención en 1988, muchos métodos de diseño y aprendizaje han sido propuestos para la CNN, de los cuales, la mayoría se enfoca a imágenes binarias. Particularmente en el trabajo presentado en [6], la técnica de diseño es reducida a resolver, mediante el método Simplex, un sistema de desigualdades lineales derivadas de unas consideraciones analíticas que involucran la condición inicial y de estado estable que estarán presentes en una neurona para cada tarea en particular. Si se toma en cuenta lo establecido como ec. (10) en [6]:

$$v_{y,kl} = \begin{cases} v_{xij}(0) = v_{uij} & k = i, l = j \\ v_{y,kl}(\infty) & k \neq i, l \neq j \end{cases} \tag{1}$$

el análisis de la dinámica de la neurona *C(i,j)* puede llevarse a cabo considerando que las salidas $v_{y,kl}$ y las entradas $v_{ukl}$ de todas las demás neuronas $C(k \bullet i, l \bullet j)$ del vecindario $N_r(i,j)$ se encuentran en estado estable. Esta consideración puede es válida si las salidas de las neuronas poseen una evolución monotónica en el tiempo, esta característica la tienen la mayoría de las CNN para imágenes binarias a excepción de la tarea del detector de componentes conectados (CCD). Por lo que, considerando (1), puede hacerse un análisis sencillo de la variable de estado $v_{xij}$ de la neurona *C(i,j)* (partiendo de la ecuación de estado normalizada )como se muestra a continuación:

$$\frac{dv_{xij}(t)}{dt} = -v_{xij}(t) + \sum_{C(k,l) \in N_r(i,j)} A(i,j:k,l) v_{y,kl} + \sum_{C(k,l) \in N_r(i,j)} B(i,j:k,l) v_{ukl} + I_{ij} \tag{2}$$

esta ecuación puede ser reescrita como:

En este artículo, se presenta un nuevo modelo de CNN capaz ejecutar diferentes tareas de procesamiento en imágenes binarias, el cual explota las características que estas tareas presentan de separabilidad lineal. modelo permite realizar una neurona CNN con aproximadamente 50 transistores (casi la mitad que la reportada en [4]), en donde a diferencia de [5], absolutamente todas las señales son unipolares. El artículo está organizado de la manera siguiente: En la sección 2 se presenta el fundamento teórico del modelo. En la sección 3, el diseño eléctrico y geométrico (*layout*) de una neurona, y finalmente en la sección 4 se muestran algunos resultados de las simulaciones, así como en la sección 5 las conclusiones.

# 2. Fundamento teórico del modelo.

## 2.1 Tareas de procesamiento para imágenes binarias linealmente separables.

Desde su invención en 1988, muchos métodos de diseño y aprendizaje han sido propuestos para la CNN, de los cuales, la mayoría se enfoca a imágenes binarias. Particularmente en el trabajo presentado en [6], la técnica de diseño es reducida a resolver, mediante el método Simplex, un sistema de desigualdades lineales derivadas de unas consideraciones analíticas que involucran la condición inicial y de estado estable que estarán presentes en una neurona para cada tarea en particular. Si se toma en cuenta lo establecido como ec. (10) en [6]:

$$v_{y,kl} = \begin{cases} v_{xij}(0) = v_{uij} & k = i, l = j \\ v_{y,kl}(\infty) & k \neq i, l \neq j \end{cases} \tag{1}$$

el análisis de la dinámica de la neurona $C(i,j)$ puede llevarse a cabo considerando que las salidas $v_{y,kl}$ y las entradas $v_{ukl}$ de todas las demás neuronas $C(k \bullet i, l \bullet j)$ del vecindario $N_r(i,j)$ se encuentran en estado estable. Esta consideración puede es válida si las salidas de las neuronas poseen una evolución monotónica en el tiempo, esta característica la tienen la mayoría de las CNN para imágenes binarias a excepción de la tarea del detector de componentes conectados (CCD). Por lo que, considerando (1), puede hacerse un análisis sencillo de la variable de estado $v_{xij}$ de la neurona $C(i,j)$ (partiendo de la ecuación de estado normalizada )como se muestra a continuación:

$$\frac{dv_{xij}(t)}{dt} = -v_{xij}(t) + \sum_{C(k,l) \in N_r(i,j)} A(i,j:k,l) v_{y,kl} + \sum_{C(k,l) \in N_r(i,j)} B(i,j:k,l) v_{ukl} + I_{ij} \tag{2}$$

esta ecuación puede ser reescrita como:

$$\frac{dv_{xij}}{dt} = -v_{xij} + a_o v_{yij} + k_1 \tag{3}$$

donde $k_1$ está en la forma:

$$k_1 = \sum_{C(k,l)\in N_r(i,j)/C_{ij}} A(k,l)v_{ykl} + \sum_{C(k,l)\in N_r(i,j)} B(i,j;k,l)v_{ukl} + I_{ij} \tag{4}$$

y por lo tanto siguiendo lo establecido en (1), sólo toma valores discretos.

La notación $C(k,l) \in N_r(i,j)/C_{ij}$ en (4), denota que se incluyen todas las neuronas del vecindario de $C_{ij}$ con excepción de ella misma, de tal forma que $a_o$ en (3) es el elemento de la matriz de retroalimentación $A(i,j;i,j)$ que corresponde a la salida de la neurona $C_{ij}$. De ésta manera, las soluciones de la ecuación (3) para las dos regiones que componen la función de salida (*piecewise linear*) establecida para la CNN en [1], (figura 1 a)), esta dada por la siguiente tabla.

Tabla 1. Ecuación de estado y sus soluciones

| Región | Ecuación de estado | Solución |
|---|---|---|
| $v_{yij} = v_{xij}$ | $\frac{dv_{xij}}{dt} = -v_{xij} + a_o v_{xij} + k_1$ | $v_{xij}(t) = \frac{k_1}{1-a_o} + Ce^{(a_o-1)t}$ (5) |
| $v_{yij} = +1$<br>$k_2 = k_1 + a_o$ | $\frac{dv_{xij}}{dt} = -v_{xij} + k_2$ | $v_{xij}(t) = k_2 + Ce^{-t}$ (6) |

De tal forma que las consideraciones analíticas del método de diseño en [6], se basan en restringir la ecuacion de estado para cada tarea en particular a través de un sistema de desigualdades, donde la forma posible de cada desigualdad se muestra en la tabla 2.

Donde (A) y (B) son desigualdades que determinan el valor de $k_2$ en (6) y por consiguiente la salida en estado estable.(C) y (D), son desigualdades necesarias para determinar el signo de la constante $C$ en (5) asegurando la evolución de la neurona hacia un valor determinado.

Así que, tomando en cuenta (1) y la tabla 2, un problema de procesamiento para imágenes binarias puede establecerse como una lista de estados. En la tabla 3, se muestra un ejemplo de este planteamiento para la

tarea del removedor de ruido, el cual debe ser resuelto posteriormente mediante un método de optimización, en este caso el método Simplex.

Tabla 2. Desigualdades para restringir a la ecuación de estado dada una salida deseada.

| Salida deseada | Estado | Desigualdades para el Estado Estable ($dv_{xij}/dt = 0$) |
|---|---|---|
| $v_{yij} = +1$ | $v_{xij} \bullet +1$ | $1 \leq \sum_{C(k,l)\in N_r(i,j)} A(i,j:k,l)v_{ykl} + \sum_{C(k,l)\in N_r(i,j)} B(i,j:k,l)v_{ukl} + I_{ij}$ (A) |
| $v_{yij} = -1$ | $v_{xij} \bullet -1$ | $-1 \geq \sum_{C(k,l)\in N_r(i,j)} A(i,j:k,l)v_{ykl} + \sum_{C(k,l)\in N_r(i,j)} B(i,j:k,l)v_{ukl} + I_{ij}$ (B) |
| Salida deseada | Condición Inicial | Desigualdades para las Condiciones Iniciales |
| $v_{yij}(\bullet) = +1$ | $v_{xij}(0) = -1$ | $0 > -v_{xij}(0) + \sum_{C(k,l)\in N_r(i,j)} A(i,j:k,l)v_{ykl} + \sum_{C(k,l)\in N_r(i,j)} B(i,j:k,l)v_{ukl} + I_{ij}$ (C) |
| $v_{yij}(\bullet) = -1$ | $v_{xij}(0) = +1$ | $0 < -v_{xij}(0) + \sum_{C(k,l)\in N_r(i,j)} A(i,j:k,l)v_{ykl} + \sum_{C(k,l)\in N_r(i,j)} B(i,j:k,l)v_{ukl} + I_{ij}$ (D) |

Como puede observarse en la tabla 3, todas las desigualdades se encuentran en una de las formas: ($\bullet$ 1), ($\bullet$ -1), (>-1) y (<1). Matemáticamente es posible reducir la región solución de aquellas para la condición inicial (>-1 y (<1), en dos unidades, de manera que todas las desigualdades se lleven a las dos únicas formas ($\bullet$ 1) y ($\bullet$ -1) sin modificar el objetivo de restricción de cada desigualdad más que en su región solución. Es necesario mencionar que para este nuevo planteamiento, cualquier solución matemática implica que el original establecido en ($\bullet$ 1), ($\bullet$ -1), (>-1) y (<1) se cumple también, pero que por el contrario de éste, las salidas pueden agruparse en dos grupos diferentes: en ($\bullet$ 1) para todos los casos de salida negra y ($\bullet$ -1) para todos aquellos de salida blanca. Las soluciones matemáticas obtenidas para este nuevo planteamiento, sugieren la existencia de un hiperplano mediante el cual es posible separar los casos de salidas opuestas. Esta característica de separabilidad lineal fue comprobada para los cuatro casos de procesamiento que abarca este trabajo: removedor de ruido, extractor de bordes, extractor de sombras y detector de conectividad global.

Tabla 3. Planteamiento para la obtención de las desigualdades del removedor de ruido.

| CASO | Pixel de Entrada $v_{uij}$ | Pixel de Salida $v_{yij}$ | Neuronas del vecindario $C_{kl}$ $k \neq i, l \neq j$ | | Desigualdades |
|---|---|---|---|---|---|
| | | | No. de Negros | No. de Blancos | |
| (1) | Negro (+1) | Blanco(-1) | 0 | 4 | -ao - 4a + bo+ I ≤ -1 B) |
| | | | | | ao - 4a + bo + I < 1 D) |
| | | | 0 | 3 | -ao - 3a + bo + I ≤ -1 B) |
| | | | | | ao - 3a + bo + I < 1 D) |
| (2) | Negro (+1) | Negro (+1) | 1 | 3 | ao – 2a + bo + I ≥ 1 A) |
| | | | 1 | 2 | ao – a + bo + I ≥ 1 A) |
| | | | 2 | 2 | ao + bo + I ≥ 1 A) |
| | | | 2 | 1 | ao + a + bo + I ≥ 1 A) |
| | | | 3 | 1 | ao + 2a + bo + I ≥ 1 A) |
| | | | 3 | 0 | ao + 3a + bo + I ≥ 1 A) |
| | | | 4 | 0 | ao + 4a + bo + I ≥ 1 A) |
| (3) | Blanco(-1) | Blanco(-1) | 0 | 4 | -ao – 4a - bo + I ≤ -1 B) |
| | | | 0 | 3 | -ao – 3a - bo + I ≤ -1 B) |
| | | | 1 | 3 | -ao – 2a - bo + I ≤ -1 B) |
| | | | 1 | 2 | -ao – a - bo + I ≤ -1 B) |
| | | | 2 | 2 | -ao - bo + I ≤ -1 B) |
| | | | 2 | 1 | -ao + a - bo + I ≤ -1 B) |
| | | | 3 | 1 | -ao +2a - bo + I ≤ -1 B) |
| (4) | Blanco(-1) | Negro (+1) | 3 | 0 | ao + 3a - bo + I ≥ 1 A) |
| | | | | | -ao + 3a - bo + I > -1 C) |
| | | | 4 | 0 | ao + 4a - bo + I ≥ 1 A) |
| | | | | | -ao + 4a - bo + I > -1 C) |
| Condición para Estabilidad | | | | | ao > 1 |

## 2.2. Función de salida escalón y la omisión del elemento de retroalimentación $a_o$.

Dada la separabilidad lineal, la función de salida "piecewise linear"(figura 1 a)), propuesta en [1], puede ser modificada por una salida del tipo escalón (figura 1 b)). De manera que todas las desigualdades antes planteadas corresponderán ahora sólo al elemento $k_2$ de (6) y el comportamiento de la variable de estado para todas las neuronas estará determinado por esta misma ecuación hacia los valores correspondientes para un píxel negro o uno blanco. Además, si se toma en cuenta el nuevo planteamiento propuesto con las desigualdades en la forma (• 1) y (• •1) para el caso específico en negrillas de la tabla 3, se tiene ahora lo siguiente.

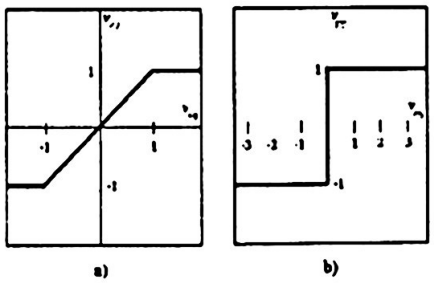



Figura 1. Funcion de salida

Caso particular de la tabla 3.

| | | | | | |
|---|---|---|---|---|---|
| (4) | Blanco(-1) | Negro (+1) | | | |
| | | | 4 | 0 | ao + 4a - bo + I ≥ 1 A) |
| | | | | | -ao + 4a - bo + I ≥ 1 C) |

Puede observarse que ahora la desigualdad establecida para el estado estable (A) y la de la condición inicial (C) son iguales salvo por el signo del elemento central de la mascarilla (*template*) de retroalimentación $a_o$. Esta característica propia del nuevo planteamiento establece que independientemente de la propia salida de la neurona *C(i,j)*, las desigualdades ( $k_2$ ) tienen que satisfacer condición • 1 propia para un píxel negro. Esto conduce a que para en esta tarea, así como en las otras tres que abarca este trabajo, el elemento de retroalimentación $a_o$ pueda ser omitido en todas las desigualdades. diferencia de la teoría original de Chua-Yang, donde dicho elemento siempre debe ser considerado e incluso debe cumplir con $a_o > 1$ para garantizar salida binaria [1], la solución matemática de este nuevo planteamiento con la función de salida escalón, comprueba que $a_o$ puede ser omitido en algunas tareas de procesamiento incluyendo las cuatro estudiadas en este trabajo. Este hecho, es una parte fundamental del nuevo modelo de CNN propuesto, ya que permite reducir la cantidad de coeficientes por neurona.

## 2.3. La multiplicación puede reducirse a un caso de selección.

Una de las ventajas de sustituir la función de salida "piecewise linear" por una de tipo escalón, radica en que esta última puede ser realizada en hardware mediante estructuras más compactas, además, la ausencia de la región lineal en la función de salida permite sustituir los multiplicadores necesarios para las mascarillas en (2), por multiplexores analógicos como se muestra en la figura2.

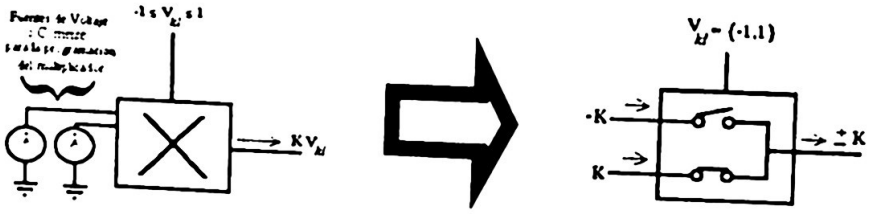



Figura 2. Los multiplicadores de cuatro cuadrantes pueden reducirse a un multiplexor analógico.

Dado que las variables $v_{xij}$ y $v_{yij}$ en el modelo Chua-Yang pueden cambiar en el intervalo [-1,1], es necesario el empleo de multiplicadores de cuatro cuadrantes para la realización de los productos de la ecuación de estado. En el nuevo modelo, debido a que estas mismas variables sólo podrán tomar dos valores diferentes {-1,1}, es posible reducir la utilización de complejos multiplicadores por interruptores MOS, reduciendo notablemente el número de transistores.

## 2.4. Unipolaridad en las señales de corriente.

La característica propia de las CNN programables radica en que los valores de las mascarillas pueden ser cambiados mediante la modificación externa de parámetros eléctricos y así poder realizar diferentes tareas de procesamiento. En el propio caso en el que los coeficientes de dichas mascarillas son llevados a cabo mediante multiplexores analógicos (figura 2), dicha programación consiste en la posibilidad de cambiar la variable K en diferentes magnitudes y sentidos. Normalmente, en circuitos integrados con arquitectura CNN, las operaciones de las mascarillas son llevadas a cabo en modo de corriente ya que esto permite efectuar de manera sencilla la operación de suma mediante la Ley de Corrientes de Kirchoff (LCK), es decir, haciendo incidir los resultados de toda la mascarilla en modo corriente a un solo nodo.

Para poder llevar a cabo la programación de una CNN en modo de corriente mediante un multiplexor analógico, sería necesario la utilización de espejos de corriente bilaterales o fuentes de corriente ajustables que permitan cambiar la variable K en magnitud y sentido. Como es conocido, sería necesario la utilización de por lo menos cuatro transistores por cada coeficiente, además de los interruptores MOS del multiplexor. De esta manera, sería deseable un modelo que permitiera que todos los resultados de las operaciones de las mascarillas fueran siempre positivas, de tal forma que podria reducirse el número de transistores empleados por cada coeficiente mediante el uso de espejos de corriente sencillos.

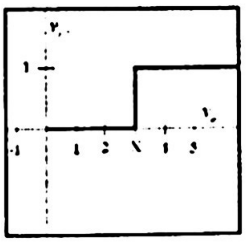


Figura 3. Función con desplazamiento.

Para lograr este objetivo en el nuevo modelo propuesto, la función escalón debe desplazarse al semiplano positivo, como se muestra en la figura 3 (cambiando la convención de valores extremos, ahora un "0" para píxeles blancos y "+1" para negros), de manera que se da lugar a que valores positivos de $v_{xij}$ menores al punto de transición X correspondan a una salida en blanco y valores positivos mayores a X correspondan a una salida en negro.

De esta manera, los coeficientes de las mascarillas

también tienen que ser modificados, esto es, descomponer cada coeficiente dos variables de las cuales una representara el valor del coeficiente para entrada o salida negra y la otra para una blanca. El nuevo planteamiento el mismo caso del removedor de ruido con todas las consideraciones para nuevo modelo se muestran en la tabla 4.

Donde $b_{OB}$ y $n_{OB}$ son las dos nuevas variables que corresponden al coeficiente "$b_o$" de la mascarilla de control para el caso de una entrada blanca y una negra respectivamente. $n_A$ y $b_A$ son las variables correspondientes a los coeficientes "a" de la mascarilla de retroalimentación, y "*f*" es una nueva variable que introducida y sustituye al umbral "*I*" del modelo Chua-Yang y solo afecta aquellas neuronas de frontera (*frontera Dirichlet*).

Tabla 4. Removedor de Ruido para X=45

$$A=\begin{bmatrix}0 & a & 0\\ a & 0 & a\\ 0 & a & 0\end{bmatrix},\quad B=\begin{bmatrix}0 & 0 & 0\\ 0 & b_o & 0\\ 0 & 0 & 0\end{bmatrix},\quad f$$

| Desigualdades | Resultado para $n_{OB}$=28.1 $b_{OB}$ =1.8 $n_A$ =12.5 $b_A$=2.2 $f$= 7.8 |
|---|---|
| $n_{OB} + 4b_A < X$ | 37.0140 |
| $n_{OB} + 3\ b_A + f < X$ | 42.6389 |
| $n_{OB} + n_A + 3\ b_A > X$ | 47.3611 |
| $n_{OB} + n_A + 2\ b_A + f > X$ | 52.9860 |
| $n_{OB} + 2\ n_A + 2\ b_A > X$ | 57.7082 |
| $n_{OB} + 2\ n_A + b_A + f > X$ | 63.3331 |
| $n_{OB} + 3\ n_A + b_A > X$ | 68.0553 |
| $n_{OB} + n_A + f > X$ | 73.6802 |
| $n_{OB} + 3\ n_A + f > X$ | 78.4023 |
| $b_{OB} + 4\ b_A < X$ | 10.6949 |
| $b_{OB} + 3\ b_A + f < X$ | 16.3198 |
| $b_{OB} + n_A + 3\ b_A < X$ | 21.0420 |
| $b_{OB} + n_A + 2\ b_A + f < X$ | 26.6669 |
| $b_{OB} + 2\ n_A + 2\ b_A < X$ | 31.3891 |
| $b_{OB} + 2\ n_A + b_A + f < X$ | 37.0140 |
| $b_{OB} + 3\ n_A + b_A < X$ | 41.7362 |
| $b_{OB} + 3\ n_A + f > X$ | 47.3611 |
| $b_{OB} + 4\ n_A > X$ | 52.0833 |

## 2.5. Modelo CNN simplificado.

Este nuevo planteamiento ha encontrado solución matemática satisfactoria en las cuatro tareas de procesamiento tratadas en este trabajo, de las cuales todas utilizan los elementos no centrales de la mascarilla de retroalimentación (*A*) y el elemento central de la mascarilla de control ($b_o$), a

excepción de la tarea del extractor de bordes que utiliza tanto los elementos no centrales como el central de la mascarilla de control (*B*). Esta característica se debe a que esencialmente existen 2 tipos de tareas: Aquellas donde es necesario la propagación de información entre neuronas y aquellas donde sólo se procesa la informacion inicial. Aunque no existe un método o regla que defina cual mascarilla a utilizar, mediante la práctica es conocido que en la mayoría de las tareas binarias, siempre se utiliza una sola mascarilla, ya sea la de control o la de retroalimentación [7]. Esta observación da pie al hecho de que en lugar de implementar los 19 coeficientes necesarios para las dos mascarillas (de la cual sólo una de ellas se utiliza en realidad) y el umbral, pueda crearse una sola matriz que pueda ser utilizada de manera multiplexada mediante una señal de selección que determine su funcionamiento, ya sea como mascarilla de control o de retroalimentación, lo cual permitiría reducir aún más el número de transistores por neurona. Así que basándose en las modificaciones expuestas para el modelo original CNN, el modelo simplificado puede ser expresado mediante las siguientes ecuaciones:

***ecuación de estado normalizada:***
$$\frac{dv_{xij}}{dt} = -v_{xij} + k_3 \tag{7}$$

donde

$$k_3 = b_O \cdot \bar{v}_{uij} + n_O \cdot v_{uij} + \sum_{C(k,l)\in Nr(i,j)/C_{ij}} \left(b_{kl} \cdot \bar{v}_{kl} + n_{kl} \cdot v_{kl}\right) + f \tag{8}$$

En esta expresión, $b_o$, $n_o$, $b_{kl}$ y $n_{kl}$ represesntan los coeficientes de las mascarillas (*A* o *B*), la variable $v_{kl}$ estará constituida por las salidas de la demás neuronas $v_{ykl}$ en el caso de que se requieran utilizar elementos no centrales de la mascarilla *A* o por $v_{ukl}$ en el caso de elementos no centrales de la mascarilla *B*. Este posible cambio de la señal como será visto en el modelo eléctrico, se lleva a cabo mediante la utilización de un multiplexor digital de dos entradas. Puesto que ahora sólo existirán dos valores posibles para las entradas $v_{uij}$ y las salidas $v_{yij}$ correspondiendo a los valores lógicos {0,1} (figura 4), los elementos $\bar{v}_{ij}$ y $\bar{v}_{kl}$ de la expresión (8) representan los complementos lógicos de dichas variables respectivamente.

La solución de la ecuación (7) esta dada en la forma:

$$v_{xij} = k_3 + Ce^{-t} \tag{9}$$

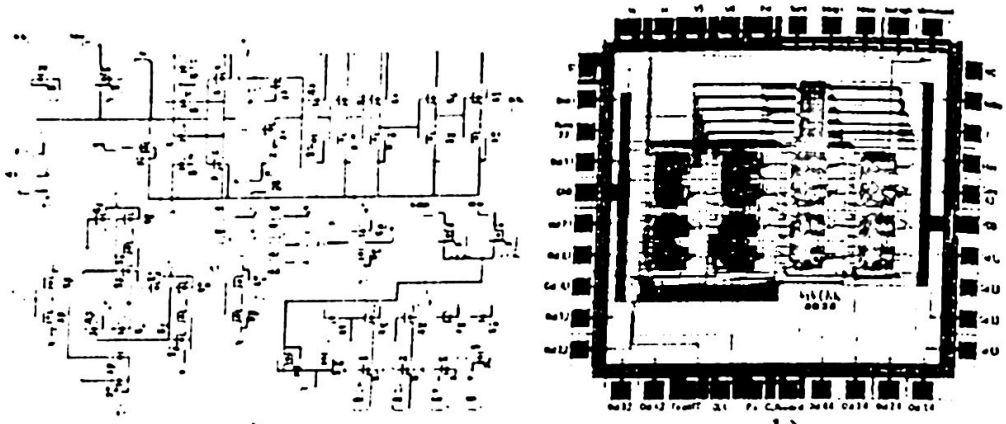
La **función de salida** escalón con desplazamiento:

$$v_{yij} = F(v_{xij}) = \begin{cases} 0 & v_{xij} < X \\ 1 & v_{xij} \geq X \end{cases} \tag{10}$$

Y finalmente, las **condiciones iniciales** que se deben satisfacer:

$$v_{ykl} = \begin{cases} v_{uij} & k = i, l = j \\ v_{ykl}(\infty) & k \neq i, l \neq j \end{cases}$$

# 3. Diseño eléctrico y geométrico.



a) b)

Figura 4. Una neurona del modelo simplificado. a) Diseño eléctrico. b) Diseñ geométrico.

El diseño eléctrico del arreglo de 4x4 se llevó a cabo mediante arquitectura de la neurona CNN de la figura 4 a). Se consideró una tecnologí CMOS de 1.2• m con un voltaje de alimentación de 5V. El diseño consta de transistores por neurona, la cual esta constituida básicamente de interruptore MOS, un multiplexor digital en base a compuertas *NAND*, espejos de corrient sencillos (coeficientes) y el comparador de corriente, el cual es llevado a cab mediante una estructura con entrada en configuración seguidor-fuente. Com etapa de salida se considera un buffer de voltaje compuesto de una serie inversores en cascada para la rápida carga y descarga de capacitancia externas. El diseño geométrico del circuito integrado se muestra en la figura b), éste se realizó considerando dos niveles de polisilico, dos metales y pozo N, propios del proceso AMI ABN 1.2micron disponible a través de MOSIS. área de una neurona individual es de 204• m X 166• m, con lo cual permitiría una densidad de 29.5 neuronas por milímetro cuadrado.

# 4. Simulaciones.

El arreglo programable de CNN 4x4 para imágenes binarias simulado a través de PSpice en la ejecución de las cuatro tareas procesamiento. En la figura 5 se muestran algunos de los resultados de la que prueban la funcionalidad del diseño, gran número de simulaciones llevaron a cabo con el propósito de verificar el cumplimiento de la lista

estados de cada tarea en particular. En los oscilogramas correspondientes, se observa un primer pulso en el cual las condiciones iniciales son introducidas a la red, un segundo pulso en el cual la red es activada y las correspondientes transiciones de las salidas de las neuronas.

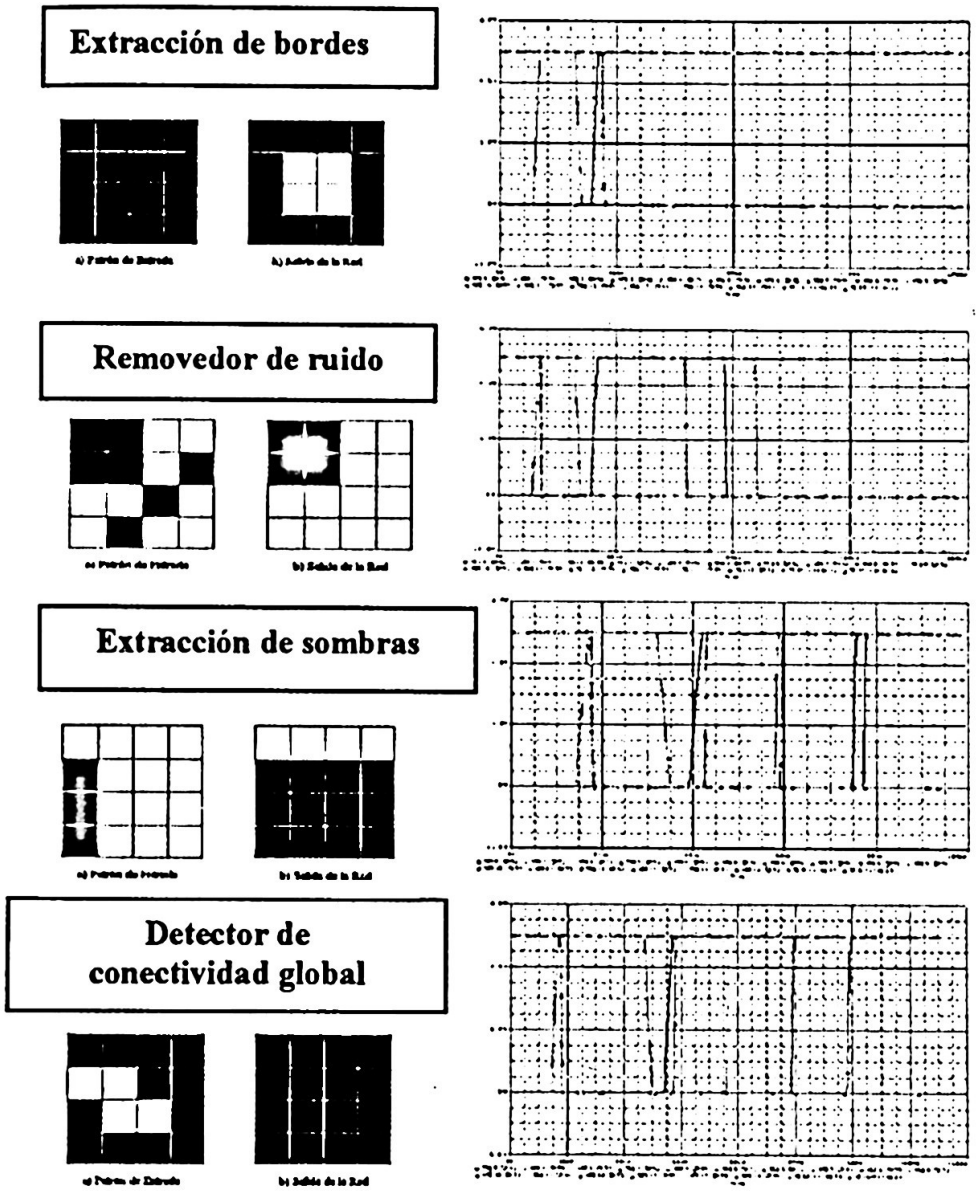



Figura 5. Resultados de las simulaciones para las diferentes tareas.

# 5. Conclusiones.

La CNN es una estructura que hoy en día representa ser una de las mejores alternativas para el procesamiento en tiempo real en imágenes, en este

trabajo se presenta un modelo simplificado que permite integrar un número mayor de neuronas en silicio (comparado con otros diseños similares en 1.2• ) y por lo tanto, arreglos mas densos de neuronas en futuras implementaciones. La salida en niveles de voltaje lógico CMOS representa una ventaja respecto a otras propuestas en modo corriente [8], ya que permite acoplamiento de este sistema con otros dispositivos lógicos convencionales la necesidad de interfaces A/D. El diseño muestra una velocidad de respuesta en el rango de 30 a 100ns, mejor a la obtenida en otras estructuras binarias programables [9-11], y podría mejorarse notablemente sobre tecnologías submicrométricas en los cuales se cuenta con un mayor número de niveles metalización.

# Referencias


[1] Leon O. Chua and L. Yang, "Cellular Neural Networks: Theory" IEEE Trans. Circuits and Systems, vol. 35, pp 1257-1272, Oct. 1988.

[2] Angel Rodriguez-Vazquez, Servando Espejo, Rafael Dominguez-Castro, Huertas and E. Sánchez-Sinencio, "Current Mode Techniques for Implementation of Continuous and Discrete-Time Cellular Neural Networks" IEEE Trans On Circuits and Systems II, vol. 40, no. 3, March 1993.

[3] A. Paasio and K. Halonen, " A New Cell Output Nonlinearity for Dense Cellular Nonlinear Network Integration" IEEE Trans. On Circuits and Systems I, vol. no. 3 March 2001.

[4] H. Harrer, J. A. Nossek, R. Stelz, "An Analog Implementation on Discrete-Time Cellular Neural Network" IEEE Trans. On Circuits and Systems, vol.3,no. 3, May 1992.

[5] M. Anguita, F.J. Pelayo, E. Ros, D. Palomar and A. Prieto, " Focal-Plane and Multiple Chip VLSI Approaches to CNNs" Analog Integrated Circuits and Signal Processing, 15, 263-275, 1998.

[6] K. Nakai and A. Ushida, "Design Technique on Cellular Neural Network" Electronics and Comunications in Japan, Part 3, vol. 78, no. 3, 1995.

[7] ACE4K: System-Level Measurement versión 2 Instituto Microelectrónica de Sevilla –Centro Nacional de Microelectrónica.

[8] Mancia Anguita, Francisco J. Pelayo, Alberto Prieto, and Julio Ortega. "Analog CMOS implementation of a discrete time CNN with programmable cloning templates" IEEE Trans. on circuits and systems II, vol 40. no. 3, march 1993.

[9] P. Kinget and M. S. J. Steyaert, " A Programmable Analog Cellular Neural Network CMOS Chip for High Speed Image Processing" IEEE Journal of Solid State Circuits, vol. 30, no. 3, March 1995.

[10] M. Anguita, F.J. Pelayo, E. Ros, D. Palomar and A. Prieto, " Focal-Plane and Multiple Chip VLSI Approaches to CNNs" Analog Integrated Circuits and Signal Processing, 15, 263-275, 1998.

[11] G. F. Dalla Betta, S. Graffi, Zs. M. Kovacs and G. Masetti, "CMOS Implementation of an Analogically Programmable Cellular Neural Network" IEEE Trans. on Circuits and Systems II, vol. 40, no. 3, March 1993

# Minimal Representations and Morphological Associative Memories for Pattern Recall

**Gerhard X. Ritter**
CISE Department, University of Florida, Gainesville, FL 32611, USA.
Email: ritter@cise.ufl.edu
**Gonzalo Urcid S.**
Optics Department, INAOE, Tonantzintla, Pue. 72000, Mexico.
Email: gurcid@inaoep.mx

**Abstract**

Morphological associative memories are a recent development in the field of artificial neural feedforward networks. The underlying computation of morphological associative memories is based on lattice algebra. Lattice algebra uses the operations of minimum or maximum together with addition, instead of addition and multiplication commonly employed in traditional models of associative memories. In comparison with associative memories based in the Hopfield recurrent network model, morphological associative memories have bigger storage capacity for exemplar patterns and possess better recall capabilities of non-boolean patterns degraded by erosive and dilative noise. Key notions, such as, kernels, morphological independence and minimal representantions are introduced in order to recall stored patterns distorted by random noise.

## 1 Introduction

The purpose of the present paper is to introduce novel concepts in the theory and applications of *morphological associative memories* (MAM's) as a new mathematical technique for storing image patterns and recalling degraded input versions of the stored patterns. In Section 2, we discuss some concepts and operations of matrix lattice algebra that form the basis for the morphological associative memories. MAM's are defined in a similar way to the classical method of correlation encoding. Section 3 describes the fundamentals of morphological associative memories and their application to the restoration of degraded *non-boolean* patterns by erosive and dilative noise. In Section 4 the key concepts of morphological strong independence and minimal representantions are defined and used for dealing with non-boolean patterns corrupted with random noise. Section 5 gives the conclusions to the present work.

## 2 Matrix lattice algebra

The basic numerical operations of taking the maximum or minimum of two numbers usually denoted as functions $\max(x, y)$ and $\min(x, y)$ will be written as binary operators using the "join" and "meet" symbols employed in lattice theory [1], i.e., $x \vee y = \max(x, y)$ and $x \wedge y = \min(x, y)$; since both operators are associative, we will use the following symbols to represent the maximum and minimum of a finite set of real numbers $\{x_1, \ldots, x_n\}$

$$\bigvee_{i=1}^{n} x_i = \max_{1 \le i \le n} \{x_i\} \text{ and } \bigwedge_{i=1}^{n} x_i = \min_{1 \le i \le n} \{x_i\}. \tag{1}$$

It was shown in [2, 11] that the algebraic systems $(\mathbb{R}_{-\infty}, \vee, +)$ and $(\mathbb{R}_{+\infty}, \wedge, +')$ are *semirings* for corresponding max-$\vee$, min-$\wedge$ operations and their respective additions, $+$ and $+'$ over the sets $\mathbb{R}_{-\infty}$ $\mathbb{R} \cup \{-\infty\}$ and $\mathbb{R}_{\infty} = \mathbb{R} \cup \{+\infty\}$, and that $(\mathbb{R}_{\pm\infty}, \vee, \wedge, +, +')$ is a *bounded lattice-group* or bounded $l$-gro where $\mathbb{R}_{\pm\infty} = \mathbb{R} \cup \{-\infty, +\infty\}$ is the set of extended real numbers.

Many popular models of associative memories allow for a formulation using matrices [6, 7, 4, 9]. model of associative memories described in this paper can also be expressed using matrix products. $\mathbb{R}$ denote the set of real numbers. For an $m \times p$ matrix $A$ and a $p \times n$ matrix $B$ with entries from $\mathbb{R}$, matrix *product* $C = A \boxtimes B$, also called the *max product* of $A$ and $B$, is defined by

$$c_{ij} = \bigvee_{k=1}^{p} (a_{ik} + b_{kj}).$$

The *min product* of $A$ and $B$ is defined in a similar fashion. Specifically, the $i,j$th entry of $C = A \boxtimes B$ given by

$$c_{ij} = \bigwedge_{k=1}^{p} (a_{ik} + b_{kj}).$$

Note that these matrix products are of the same form as regular matrix products with the role of addition and multiplication replaced by the operations of maximum (or minimum) and addition, respectively. The addition of two matrices is replaced by the operation of maximum (or minimum) in this matrix algebra. Specifically, the maximum of two matrices replaces the usual matrix addition of linear algebra. Here the $i,j$th entry of the matrix $C = A \vee B$ is given by $c_{ij} = a_{ij} \vee b_{ij}$. Similarly, the minimum two matrices $C = A \wedge B$ is defined by $c_{ij} = a_{ij} \wedge b_{ij}$. The *morphological outer product* of two vectors $\mathbf{x} = (x_1, \ldots, x_n)' \in \mathbb{R}^n$ and $\mathbf{y} = (y_1, \ldots, y_m)' \in \mathbb{R}^m$ is defined as

$$\mathbf{y} \times \mathbf{x}' = \begin{pmatrix} y_1 + x_1 & \cdots & y_1 + x_n \\ \vdots & \ddots & \vdots \\ y_m + x_1 & \cdots & y_m + x_n \end{pmatrix}.$$

It is worthwhile to note that $\mathbf{y} \times \mathbf{x}' = \mathbf{y} \boxtimes \mathbf{x}' = \mathbf{y} \boxtimes \mathbf{x}'$. Finally, we say that $A$ is *less or equal than* denoted by $A \leq B$, and $A$ is *strictly less than* $B$, denoted by $A < B$, if and only if for each corresponding entry of these matrices we have that $a_{ij} \leq b_{ij}$ and $a_{ij} < b_{ij}$, respectively.

# 3 Morphological associative memories

## 3.1 Background

One of the first goals achieved in the development of morphological neural networks was the establishment of a morphological associative memory network (MAM). In its basic form, this model of an associative memory resembles the well-known correlation memory or linear associative memory [6]. As in correlation encoding, the morphological associative memory provides a simple method to add new associations. A weakness in correlation encoding is the requirement of orthogonality of the key vectors in order to exhibit perfect recall of the fundamental associations. The morphological auto-associative memory does not restrict the domain of they key vectors in any way. Thus, as many associations as desired can be encoded into the memory [12].

In the real number case, the capacity for a memory of length $n$ can be as large as desired. That is, if $k$ denotes the number of distinct patterns of length $n$ to be encoded, then $k$ is allowed to be any integer, no matter how large. Of course, in the binary case, the limit is $k = 2^n$ as this is the maximum number of distinct patterns of length $n$. In comparison, McEliece et al. showed that the asymptotic limit capacity of the Hopfield associative memory is $n/2 \log n$ if with high probability the unique fundamental memory is to

be recovered, except for a vanishingly small fraction of fundamental memories [8]. Likewise, the information storage capacity (number of bits which can be stored and recalled associatively) of the morphological auto-associative memory also exceeds the respective number of certain linear matrix associative memories which was calculated by Palm [10]. Among the various auto-associative networks the Hopfield network is the most widely known today [4, 5]. Unlike the Hopfield network, which is a recurrent neural network, the morphological model provides the final result in one pass through the network without any significant amount of training. Ritter, Sussner, and Diaz de Leon used a number of experiments to demonstrate the MAM's efficiency to deal with either dilative (additive) or erosive (subtractive) changes to input (binary) patterns, including incomplete patterns [12]. Thus, the morphological associative memory model provides almost all the characteristics of an ideal associative memory with one notable exception: the network's inability to deal with patterns which contain both erosive as well as dilative noise.

Henceforth, let $(\mathbf{x}^1,\mathbf{y}^1),\ldots,(\mathbf{x}^k,\mathbf{y}^k)$ be $k$ vector pairs with $\mathbf{x}^\xi = (x_1^\xi,\ldots,x_n^\xi)' \in \mathbb{R}^n$ and $\mathbf{y}^\xi = (y_1^\xi,\ldots,y_m^\xi)' \in \mathbb{R}^m$ for $\xi = 1,\ldots,k$. For a given set of pattern associations $\{(\mathbf{x}^\xi,\mathbf{y}^\xi) : \xi = 1,\ldots,k\}$ we define a pair of associated pattern matrices $(X,Y)$, where $X = (\mathbf{x}^1,\ldots,\mathbf{x}^k)$ and $Y = (\mathbf{y}^1,\ldots,\mathbf{y}^k)$. Thus, $X$ is of dimension $n \times k$ with $i,j$th entry $x_i^j$ and $Y$ is of dimension $m \times k$ with $i,j$th entry $y_i^j$.

The earliest neural network approach to associative memories was the linear associative memory or correlation memory [6]. In this approach the goal is to store $k$ vector pairs $(\mathbf{x}^1,\mathbf{y}^1),\ldots,(\mathbf{x}^k,\mathbf{y}^k)$ in an $m \times n$ *associative memory* $W$ such that for any given input vector $\mathbf{x}^\xi$, the associative memory $W$ recalls the output vector $\mathbf{y}^\xi = W\mathbf{x}^\xi, \forall \xi = 1,\ldots,k$. The simplest solution for this goal is to set

$$W = \sum_{\xi=1}^{k} \mathbf{y}^\xi (\mathbf{x}^\xi)'. \tag{5}$$

In this case, the $i,j$th entry of $W$ is given by $w_{ij} = \sum_{\xi=1}^{k} y_i^\xi x_j^\xi$. If the input patterns $\mathbf{x}^1,\ldots,\mathbf{x}^k$ are orthonormal, then

$$W\mathbf{x}^\xi = \mathbf{y}^\xi((\mathbf{x}^\xi)' \bullet \mathbf{x}^\xi) + \sum_{\gamma \neq \xi} \mathbf{y}^\gamma((\mathbf{x}^\gamma)' \bullet \mathbf{x}^\xi) = \mathbf{y}^\xi. \tag{6}$$

Thus, we have *perfect recall* of the output patterns $\mathbf{y}^1,\ldots,\mathbf{y}^k$. If $\mathbf{x}^1,\ldots,\mathbf{x}^k$ are not orthonormal (as in most realistic cases), then filtering processes using activation functions become necessary in order to retrieve the desired output pattern.

## 3.2 Definitions and properties

Morphological associative memories are surprisingly similar to these classical correlation memories. With each pair of pattern associations $(X,Y)$ we associate two natural morphological $m \times n$ memories $W_{XY}$ and $M_{XY}$ defined by

$$W_{XY} = \bigwedge_{\xi=1}^{k} [\mathbf{y}^\xi \times (-\mathbf{x}^\xi)'] \;;\; M_{XY} = \bigvee_{\xi=1}^{k} [\mathbf{y}^\xi \times (-\mathbf{x}^\xi)']. \tag{7}$$

Note the similarities between the definition of the memory given by Eq.(5) and those defined by Eq.(7). Also, a consequence of Eq.(7) is that

$$W_{XY} \boxtimes X \leq Y \leq M_{XY} \boxtimes X. \tag{8}$$

A fundamental relationship between the *canonical* MAM's and other morphological associative memories is given by the next theorem which was proved in [12].

**Theorem 3.1.** Let (X,Y) denote the associate sets of pattern vector pairs. Whenever there exists pe recall memories A and B such that $A \boxtimes \mathbf{x}^\xi = \mathbf{y}^\xi$ and $B \boxtimes \mathbf{x}^\xi = \mathbf{y}^\xi$ for $\xi = 1, \ldots, k$, then

$$A \leq W_{XY} \leq M_{XY} \leq B \text{ and}$$
$$\forall \xi, \; W_{XY} \boxtimes \mathbf{x}^\xi = \mathbf{y}^\xi = M_{XY} \boxtimes \mathbf{x}^\xi.$$

Hence, $W_{XY}$ is the least upper bound of all perfect recall memories involving the $\boxtimes$ operation $M_{XY}$ is the greatest lower bound of all perfect memories involving the $\boxtimes$ operation. Furthermore, if the exists perfect recall memories, then the canonical memories are also perfect recall memories. If $X =$ (i.e., $\forall \xi, \mathbf{x}^\xi = \mathbf{y}^\xi$), then we obtain the morphological auto-associative memories $W_{XX}$ and $M_{XX}$. In [12 we proved that

$$W_{XX} \boxtimes X = X = M_{XX} \boxtimes X, \tag{10}$$

where $X$ can consist of any arbitrarily large number of pattern vectors. For example, consider the sev



Figure 1: The seven patterns in the top row were used in constructing the memories $W_{XX}$ and $M_{XX}$ (of size $2500 \times 2500$). The bottom row shows the output for these morphological autoassociative memories when presented with the respective patterns in the top row.

pattern images $\mathbf{p}^1, \ldots, \mathbf{p}^7$ shown in Fig.(1). Each $\mathbf{p}^\xi$ is a $50 \times 50$ pixels 256-gray scale image. For uncorrupted input, perfect recall is guaranteed if we use the memory $W_{XX}$ or $M_{XX}$. Using the standard row-scan method, each pattern image $\mathbf{p}^\xi$ can be converted into a pattern vector $\mathbf{x}^\xi = (x_1^\xi, \ldots, x_{2500}^\xi)$ by defining

$$x_{50(r-1)+c}^\xi = p^\xi(r,c) \text{ for } r, c = 1, \ldots, 50. \tag{11}$$

## 3.3 Erosive and dilative noise

Morphological associative memories are extremely robust in the presence of certain types of noise, missing data, or occlusions. We say that a distorted version $\tilde{\mathbf{x}}^\gamma$ of the pattern $\mathbf{x}^\gamma$ has undergone an *erosive change* whenever $\tilde{\mathbf{x}}^\gamma \leq \mathbf{x}^\gamma$ and a *dilative change* whenever $\tilde{\mathbf{x}}^\gamma \geq \mathbf{x}^\gamma$. Corrupting the patterns $\mathbf{x}^\xi$ with 30% randomly generated erosive and dilative noise with an intensity level of 128 results in almost perfect recall (NMSE $< 10^{-3}$)[1] when using the memory $W_{XX}$ and $M_{XX}$, respectively. Figs.(2)–(3) provide for a visual example of this experiment. The reason for the robustness of associative memories in the presence of erosive or dilative noise is a consequence of the following theorem which was validated in [12].

**Theorem 3.2.** Let $\tilde{\mathbf{x}}^\gamma$ denote the distorted version of the pattern $\mathbf{x}^\gamma$. Then $W_{XY} \boxtimes \tilde{\mathbf{x}}^\gamma = \mathbf{y}^\gamma$ if and only if $\forall j = 1, \ldots, n$

$$\tilde{x}_j^\gamma \leq x_j^\gamma \vee \bigwedge_{i=1}^{m} \left( \bigvee_{\xi \neq \gamma} \left[ y_i^\gamma - y_i^\xi + x_j^\xi \right] \right) \tag{12}$$

[1] Normalized mean square error, computed for each $\xi$ as $\sum_j (\tilde{x}_j^\xi - x_j^\xi)^2 \big/ \sum_j (x_j^\xi)^2$



Figure 2: Top row: corrupted input patterns (erosive change); bottom row, the corresponding recalled patterns using the morphological memory $W_{XX}$.



Figure 3: Top row: corrupted input patterns (dilative change); bottom row, the corresponding recalled patterns using the morphological memory $M_{XX}$.

and for each row index $i \in \{1, \ldots, m\}$ there exists a column index $j_i \in \{1, \ldots, n\}$ such that

$$\tilde{x}_{j_i}^{\gamma} = x_{j_i}^{\gamma} \vee \left( \bigvee_{\xi \neq \gamma} \left[ y_i^{\gamma} - y_i^{\xi} + x_{j_i}^{\xi} \right] \right). \tag{13}$$

A similar result holds for the memory $M_{XY}$ by changing the inequality sign and the operations of maximum and minimum in Eqs. (12)–(13). The next corollary is an easy consequence of this theorem [14].

**Corollary 3.3.** Suppose that $\tilde{\mathbf{x}}^{\gamma}$ denotes an eroded version of $\mathbf{x}^{\gamma}$. The equation $W_{XX} \boxtimes \tilde{\mathbf{x}}^{\gamma} = \mathbf{x}^{\gamma}$ holds if and only if for each row index $i \in \{1, \ldots, n\}$ there exists a column index $j_i \in \{1, \ldots, n\}$ such that

$$\tilde{x}_{j_i}^{\gamma} = x_{j_i}^{\gamma} \vee \left( \bigvee_{\xi \neq \gamma} \left[ x_i^{\gamma} - x_i^{\xi} + x_{j_i}^{\xi} \right] \right). \tag{14}$$

A similar result holds for the memory $M_{XX}$ by changing the maximum operator by the minimum operator in Eq. (14).

Although Theorem 3.2 and its Corollary 3.3 provides necessary and sufficient conditions for the bounds of the corruption of the pattern $\mathbf{x}^{\gamma}$ that guarantees perfect recall, it also implies that $W_{XX}$ will fail if dilative noise not satisfying these bounds is present. Insertion of only minute amounts of dilative noise, often in only one vector component, can result in complete recall failure. Similar comments hold for the memory $M_{XX}$ and erosive noise. Hence, neither memory $W_{XX}$ or $M_{XX}$ is useful in the presence of random noise which, generally, consists of both erosive as well as dilative noise.

The kernel method proposed in [14, 12] suggests a solution to this dilemma. However, it became clear that finding an algorithmic method for selecting an optimal set of proper kernels was not going to be an easy task. Part of the difficulty is due to the fact that the existence of proper kernels for a given set of pattern vectors remains an unsolved problem if the definition of kernels proposed in [12] is used.

More recently, Ritter, Urcid, and Iancu [13] introduced the notion of minimal representations of exemplar *non-boolean* patterns and by redefining the concept of kernels they illustrate that kernels can be effectively used for building morphological associative memories that are robust in the presence of both erosive and dilative noise. In order to deal with the problem of random noise, we introduce, in the next section, the notions of morphological independence and of minimal representations.

# 4 Morphological independence and minimal representations

## 4.1 Background

Since $W_{XX}$ is suitable for recognizing patterns corrupted by erosive noise and $M_{XX}$ is suitable for recognizing patterns corrupted by dilative noise, an intuitive idea is to process a noisy version $\tilde{\mathbf{x}}^\gamma$ of $\mathbf{x}^\gamma$ containing both erosive and dilative noise through a combination of $W_{XX}$ and $M_{XX}$. Sussner proved that passing the output of $M_{XX} \boxtimes \tilde{\mathbf{x}}^\gamma$ through the memory $W_{XX}$ or, dually, the output of $W_{XX} \boxtimes \tilde{\mathbf{x}}^\gamma$ through $M_{XX}$ will, generally, not result in $\mathbf{x}^\gamma$ [14]. Nevertheless, the modified kernel approach proposed by Ritter et al. is based on this intuitive idea using the memories $M_{XX}$ and $W_{XX}$ in sequence in order to create a morphological memory that is robust in the presence of random noise, even in the general situation where $X \neq Y$ and $X$ and $Y$ are not boolean [12]. The underlying idea is to define a memory $M$ which associates with each input pattern $\mathbf{x}^\gamma$ an intermediate pattern $\mathbf{z}^\gamma$. Another associative memory $W$ is defined which associates each pattern $\mathbf{z}^\gamma$ with the desired output pattern $\mathbf{y}^\gamma$. In terms of min-max products, one obtains the equation:

$$W \boxtimes (M \boxtimes \mathbf{x}^\gamma) = \mathbf{y}^\gamma. \tag{15}$$

Obviously, if $Z$ is a kernel for $(X, Y)$ and $\mathbf{z}^\gamma \leq \tilde{\mathbf{x}}^\gamma \leq \mathbf{x}^\gamma$, then

$$\mathbf{z}^\gamma = M_{ZZ} \boxtimes \mathbf{z}^\gamma \leq M_{ZZ} \boxtimes \tilde{\mathbf{x}}^\gamma \leq M_{ZZ} \boxtimes \mathbf{x}^\gamma = \mathbf{z}^\gamma \tag{16}$$

and, hence, $M_{ZZ} \boxtimes \tilde{\mathbf{x}}^\gamma = \mathbf{z}^\gamma$. Thus, for eroded versions of $\mathbf{x}^\gamma$ that are bounded below by $\mathbf{z}^\gamma \leq \tilde{\mathbf{x}}^\gamma$ we are guaranteed that

$$W_{ZY} \boxtimes (M_{ZZ} \boxtimes \tilde{\mathbf{x}}^\gamma) = \mathbf{y}^\gamma. \tag{17}$$

Sussner showed the following fundamental result regarding kernels for binary patterns [14].

**Theorem 4.1.** Let $X, Y$ and $Z$ be sets of binary patterns with $Z \leq X$. If

$$\forall \xi \neq \gamma, \ \mathbf{z}^\gamma \wedge \mathbf{z}^\xi = 0 \text{ and } \mathbf{z}^\gamma \not\leq \mathbf{x}^\xi, \tag{18}$$

then $Z$ is a kernel for $(X, Y)$.

## 4.2 Morphological independence

Various attempts at generalizing this result to the non-boolean case have been unsuccessful. The failure has been due to the fact that in the boolean case the condition specified by Eq.(18) in Theorem 4.1 is implied by the notion of *morphological independence* and results in a kernel. As it turns out, the same is not true in the non-boolean case. A consequence of Sussner's theorem is that for a morphologically independent set of binary patterns $X$, a kernel set $Z$ for $(X, Y)$ can now be easily chosen. The requirement of morphological independence is not nearly as restrictive as linear independence. Formally, we have:

**Definition 4.1.** A set of pattern vectors $X = (\mathbf{x}^1, \ldots, \mathbf{x}^k)$ is said to be *morphologically independent* if and only if for $\gamma = 1, \ldots, k$

$$\mathbf{x}^\gamma \not\leq \bigvee_{\xi \neq \gamma} \mathbf{x}^\xi. \tag{19}$$

It is worthwhile to note the resemblance between morphological independence and linear independence. In linear independence, no vector of a set can be a linear sum of the remaining vectors, while in morphological independence, no vector can be less than the maximum of the remaining vectors. This resemblance is due to the algebraic similarity between linear algebra and lattice algebra, where the operation of summation replaced by the operation of maximum and equalities are replaced by appropriate inequalities. We need to mention that morphological independence is a completely different concept than the notion of linear independence in minimax algebra proposed by Cuninghame-Green [3]. A vector can be morphologically independent but not linearly independent as defined by Cuninghame-Green, and vice-versa.

Thus, for example, the set of pattern images shown in Fig.(1) is morphologically independent. It follows from the definition that if $X$ is morphologically independent, then there must be an index $j_\gamma \in \{1,\dots,n\}$ such that $x^\xi_{j_\gamma} < x^\gamma_{j_\gamma}$ $\forall \xi \neq \gamma$. Hence, $\mathbf{x}^\gamma \not\leq \mathbf{x}^\xi$ $\forall \xi \neq \gamma$. However, the converse does not hold. As mentioned earlier, if $X$ is morphologically independent, then it becomes easy to define a set of patterns $Z \leq X$ that satisfies Eq.(18) of Theorem 4.1. For each $\gamma \in \{1,\dots,k\}$ we simply pick the index $j_\gamma \in \{1,\dots,n\}$ for which $x^\xi_{j_\gamma} < x^\gamma_{j_\gamma}$ $\forall \xi \neq \gamma$ and define $\mathbf{z}^\gamma$ by setting

$$z_i^\gamma = \begin{cases} x^\gamma_{j_\gamma} \text{ if } i = j_\gamma \\ 0 \text{ if } i \neq j_\gamma \end{cases} \tag{20}$$

for $i = 1,\dots,n$. It follows that $\mathbf{z}^\gamma \wedge \mathbf{z}^\xi = \mathbf{0}$ and $\mathbf{z}^\gamma \not\leq \mathbf{x}^\xi$ $\forall\, \xi \neq \gamma$. If $X$ is also boolean, then it follows from Theorem 4.1 that $Z$ is a kernel for $(X, Y)$. However, in our construction of $Z$ we did not assume that $X$ was boolean.

Consider again the pattern images given in Fig.(1), which are morphologically independent. Table 1 gives the row indexes $j_\gamma$ for $\gamma = 1,\dots,7$ such that $x^\gamma_{j_\gamma} > x^\xi_{j_\gamma}$ $\forall \xi \neq \gamma$. Using these values of $j_\gamma$, the matrix $Z$ was generated using Eq.(20) to test if it resulted in a kernel; Fig.(4) shows in visual form the pixel position corresponding to each pattern row index $j = j_\gamma$ However, $Z$ is not a kernel since the min product

Table 1: Row index $j$, pixel position $(r, c)$, and value (underscored) $x^\gamma_j$ of each pattern $\gamma$ used to build matrix $Z$.

| $\gamma$ | $j$ | $(r,c)$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1759 | (36,9) | <u>255</u> | 141 | 165 | 94 | 142 | 159 | 196 |
| 2 | 453 | (10,3) | 25 | <u>255</u> | 71 | 164 | 136 | 101 | 184 |
| 3 | 2358 | (48,8) | 233 | 205 | <u>237</u> | 163 | 192 | 116 | 107 |
| 4 | 260 | (6,10) | 20 | 56 | 62 | <u>255</u> | 134 | 105 | 44 |
| 5 | 2186 | (44,36) | 175 | 112 | 102 | 89 | <u>255</u> | 173 | 200 |
| 6 | 737 | (15,37) | 195 | 70 | 96 | 116 | 136 | <u>255</u> | 164 |
| 7 | 1276 | (26,26) | 208 | 46 | 199 | 153 | 159 | 176 | <u>251</u> |

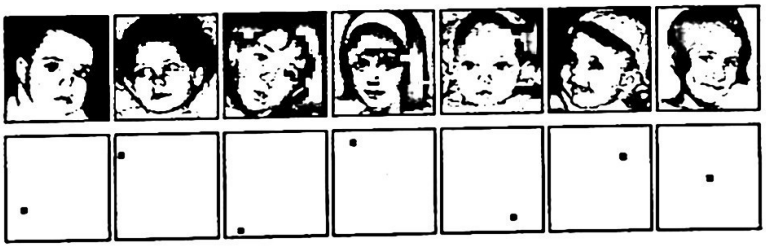


Figure 4: Top row: morphologically independent patterns; bottom row, the corresponding non-zero entries in matrix $Z$ used as a candidate kernel.

$M_{ZZ} \boxtimes X \neq Z$; also, the max product $W_{XX} \boxtimes (M_{ZZ} \boxtimes X) \neq X$. Partial reconstruction and the effect of crosstalk noise between patterns for perfect input is shown in Fig.(5); direct computation of $W_{ZX} \boxtimes Z$ is also different from $X$ and visually the output is similar to the bottom row of the same figure.

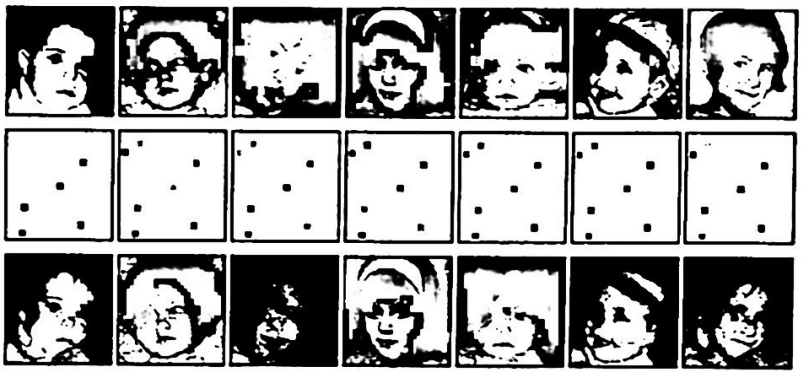


Figure 5: Middle row: output of memory $M_{ZZ}$ when presented with the morphologically independent set shown in the top row; bottom row, output of combined memory scheme $M_{ZZ} \rightarrow W_{XX}$.

**Definition 4.2.** A set of pattern vectors $X = (\mathbf{x}^1, \ldots, \mathbf{x}^k)$ is said to be *morphologically strongly independent* if and only if the following two conditions are satisfied for all $\xi \neq \gamma$:

1. For each $\gamma \in \{1, \ldots, k\}$, $\mathbf{x}^\gamma \not\leq \mathbf{x}^\xi$,
2. For each $\gamma \in \{1, \ldots, k\}$ there exists an index $j_\gamma \in \{1, \ldots, n\}$ such that

$$x^\xi_{j_\gamma} - x^\xi_i \leq x^\gamma_{j_\gamma} - x^\gamma_i, \ \forall i = 1, \ldots, n. \tag{21}$$

The notions of morphological independence and strong independence are, generally not equivalent. However, if $X$ is morphologically independent, then $X$ satisfies Condition 1 of strong morphological independence. Also, we have shown previously, that if $X$ is morphologically independent, then $\forall \gamma = 1, \ldots, k$ there exists an index $j_\gamma \in \{1, \ldots, n\}$ such that $x^\xi_{j_\gamma} < x^\gamma_{j_\gamma}$ $\forall \xi \neq \gamma$. As it turns out, the same property also holds for morphologically strongly independent sets. The proof of the following theorems appears in [13],

**Theorem 4.2.** Morphological strong independence implies morphological independence.

**Theorem 4.3.** Suppose $X$ is boolean. Then $X$ is morphologically independent if and only if $X$ is morphologically strongly independent.

## Minimal Representations and Kernels

roadblock in obtaining meaningful kernels in the non-boolean case is the overly restrictive requirement that $M_{ZZ} \boxtimes \mathbf{x}^\gamma = \mathbf{z}^\gamma$. However, if we simply require that there exists a memory $W$ such that

$$W \boxtimes (M_{ZZ} \boxtimes \mathbf{x}^\gamma) = \mathbf{y}^\gamma, \tag{22}$$

which agrees with our original intuitive idea expressed by Eq.(15), namely that $M_{ZZ} \boxtimes \mathbf{x}^\gamma$ need only be some intermediate pattern, then several results that mirror Sussner's theorem can be obtained for non-boolean patterns. We therefore suggest the following less restrictive definition of a kernel together with the concept of a minimal representation:

**Definition 5.1.** Let $Z = (\mathbf{z}^1, \ldots, \mathbf{z}^k)$ be an $n \times k$ matrix. We say that $Z$ is a *kernel* for $(X, Y)$ if and only if $Z \neq X$ and there exists a memory $W$ such that

$$W \boxtimes (M_{ZZ} \boxtimes \mathbf{x}^\gamma) = \mathbf{y}^\gamma. \tag{23}$$

If $Y = X$, then we say that $Z$ is a *kernel* for $X$.

**Definition 5.2.** A set of patterns $Z \leq X$ is said to be a *minimal representation* of $X$ if and only if for $\gamma = 1, \ldots, k$

1. $\mathbf{z}^\gamma \wedge \mathbf{z}^\xi = 0 \; \forall \xi \neq \gamma$,
2. $\mathbf{z}^\gamma$ contains at most one non-zero entry, and
3. $W_{ZX} \boxtimes \mathbf{z}^\gamma = \mathbf{x}^\gamma$.

Condition 1 of this definition satisfies part of Eq.(18) of Sussner's theorem while Condition 2 assures sparsity. Condition 3 simply says that $X$ can be reconstructed from $Z$. In this sense $Z$ acts as an orthogonal basis within the lattice algebra underlying the morphological operations.

There is an obvious close connection between kernels and minimal representations. If $Z$ is a kernel for $X$ in the sense of Definition 4.1, then $Z \leq X$, $M_{ZZ} \boxtimes \mathbf{x}^\gamma = \mathbf{z}^\gamma$, and $W_{ZX} \boxtimes \mathbf{z}^\gamma = \mathbf{x}^\gamma$. Thus, kernels satisfy Condition 3 of minimal representations. But from examples given in [12], kernels need not satisfy Conditions 1 and 2 of minimal representations. We now consider the converse, namely for what pattern sets do there exist minimal representations that may also serve as kernels. In the remainder of this section we assume that pattern features are *non-negative*, i.e., $x_i^\gamma \geq 0 \; \forall \gamma$ and $\forall i$.

**Theorem 5.1.** If $X$ is morphologically strongly independent, then there exists a set of patterns $Z \leq X$ with the property that for $\gamma = 1, \ldots, k$

1. $\mathbf{z}^\gamma \wedge \mathbf{z}^\xi = 0 \; \forall \xi \neq \gamma$,
2. $\mathbf{z}^\gamma$ contains at most one non-zero entry, and
3. $W_{XX} \boxtimes \mathbf{z}^\gamma = \mathbf{x}^\gamma$.

The next corollary is an easy consequence of this theorem.

**Corollary 5.2.** If $X$ and $Z$ are as in Theorem 5.1, then $Z$ is a minimal representation of $X$.

Suppose $X$ and $Z$ are as in Theorem 5.1 and $\mathbf{u}^\gamma = M_{ZZ} \boxtimes \mathbf{x}^\gamma$. Then for each $i = 1, \ldots, n$ we have that

$$u_i^\gamma = (M_{ZZ} \boxtimes \mathbf{x}^\gamma)_i = \bigwedge_{j=1}^{n} (m_{ij} + x_j^\gamma) \leq m_{ii} + x_i^\gamma = x_i^\gamma \tag{24}$$

since $m_{ii} = 0$. Hence $\mathbf{u}^\gamma \leq \mathbf{x}^\gamma$ for each $\gamma = 1, \ldots, k$. Since $\mathbf{z}^\gamma \leq \mathbf{x}^\gamma$, it now follows that

$$\mathbf{z}^\gamma = M_{ZZ} \boxtimes \mathbf{z}^\gamma \leq M_{ZZ} \boxtimes \mathbf{x}^\gamma \leq \mathbf{x}^\gamma. \tag{25}$$

In view of Theorem 5.1 and Eq.(25) we have

$$\begin{aligned} \mathbf{x}^\gamma &= W_{XX} \boxtimes \mathbf{z}^\gamma = W_{XX} \boxtimes (M_{ZZ} \boxtimes \mathbf{z}^\gamma) \\ &\leq W_{XX} \boxtimes (M_{ZZ} \boxtimes \mathbf{x}^\gamma) \leq W_{XX} \boxtimes \mathbf{x}^\gamma = \mathbf{x}^\gamma. \end{aligned} \tag{26}$$

Therefore,

$$W_{XX} \boxtimes (M_{ZZ} \boxtimes \mathbf{x}^\gamma) = \mathbf{x}^\gamma \; \forall \gamma = 1, \ldots, k. \tag{27}$$

By letting $W = W_{XX}$ the preceding argument verifies the following corollary:

**Corollary 5.3.** If $X$ and $Z$ are as in Theorem 5.1, then $Z$ is a kernel for $X$.

According to Corollary 5.2, a minimal representation is also a kernel. Hence, for a set of patterns $X$ to be reducible to a kernel, it is sufficient that $X$ is strongly independent. Furthermore, if $X$ is strongly independent, then in order to obtain a kernel one simply selects a minimal representation $Z$ of $X$ using the method given in the proof of Theorem 5.1. Given a minimal representation $Z$ which is also a kernel for $X$ and a noisy version $\tilde{\mathbf{x}}^\gamma$ of the pattern $\mathbf{x}^\gamma$ having the property that $\mathbf{z}^\gamma \leq \tilde{\mathbf{x}}^\gamma$ and $M_{ZZ} \boxtimes \tilde{\mathbf{x}}^\gamma \leq \mathbf{x}^\gamma$, then it must follow that

$$W_{XX} \boxtimes (M_{ZZ} \boxtimes \tilde{\mathbf{x}}^\gamma) = \mathbf{x}^\gamma. \tag{28}$$

The pattern images $\mathbf{p}^1, \ldots, \mathbf{p}^7$ of Fig.(1) were slightly modified so that the new set of patterns is morphologically strongly independent; Table 2 gives the list of the corresponding indexes where the pixel value for each pattern $\gamma$ in row $j = j_\gamma$ was taken to be the maximum (white) and for $\xi \neq \gamma$ the minimum (black) was assigned. Matrix $Z$ was again defined according to Eq.(18) and applying the results established in Theorem 5.1 and Corollaries 5.2 and 5.3, $Z$ is a minimal representation as well as a kernel respectively; Fig.(6) shows the morphologically strongly independent set of patterns and the associated minimal representation given by $(\mathbf{z}^1, \ldots, \mathbf{z}^7)$. Randomly corrupting the patterns shown in Fig.(6) with 30% of noise

Table 2: Row index $j$, pixel position $(r, c)$, and value (underscored) $x_j^\gamma$ of each pattern $\gamma$ used to build matrix $Z$.

| $\gamma$ | $j$ | $(r,c)$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2463 | (50,13) | 255 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1845 | (37,45) | 0 | 255 | 0 | 0 | 0 | 0 | 0 |
| 3 | 2430 | (49,30) | 0 | 0 | 255 | 0 | 0 | 0 | 0 |
| 4 | 65 | (2,15) | 0 | 0 | 0 | 255 | 0 | 0 | 0 |
| 5 | 112 | (3,12) | 0 | 0 | 0 | 0 | 255 | 0 | 0 |
| 6 | 2466 | (50,16) | 0 | 0 | 0 | 0 | 0 | 255 | 0 |
| 7 | 14 | (1,14) | 0 | 0 | 0 | 0 | 0 | 0 | 255 |



Figure 6: Top row: morphologically strongly independent patterns: bottom row, the corresponding nonzero entries in matrix $Z$ used as a minimal representation or kernel.

with an intensity level of 128, and using the minimal representation $Z$ as our kernel set, we obtained the perfect recall $W_{XX} \boxtimes (M_{ZZ} \boxtimes \tilde{\mathbf{x}}^\gamma) = \mathbf{x}^\gamma$ for $\gamma = 1, \ldots, 7$ shown in Fig.(7).

As we have observed earlier, for a set of patterns $X$ to be reducible to a kernel, it is sufficient that $X$ is strongly independent. Strong independence, however, is not a necessary condition. The question of necessary and sufficient conditions for the existence of kernels remains open. The condition that $W_{XX} \boxtimes \mathbf{z}^\gamma = \mathbf{x}^\gamma$ is crucial in our proof of the kernel scheme

$$input \rightarrow M_{ZZ} \rightarrow W_{XX} \rightarrow output. \tag{29}$$



Figure 7: Top row: corrupted input patterns (random noise); bottom row, perfect recall using the kernel $Z$ for the memory scheme $M_{ZZ} \rightarrow W_{XX}$ (also the output of memory $W_{ZX}$).

In order to prove Condition 3 of Theorem 5.1, we had to use the fact that $X$ is strongly independent. Thus far we have been unable to weaken the hypothesis of strong independence. There is a good reason why minimal representations are the preferred kernels for the recovery of patterns from noisy inputs. Recall that Eq.(28) will be satisfied whenever

$$z_i^\gamma \leq \tilde{x}_i^\gamma \text{ and } (M_{ZZ} \boxtimes \tilde{\mathbf{x}}^\gamma)_i \leq x_i^\gamma \tag{30}$$

$\forall i = 1, \ldots, n$. Now, if for some $i$ the $i,j$th entry of $M_{ZZ}$ is zero for every $j = 1, \ldots, n$, then Eq.(30) is satisfied for this particular index $i$ as long as

$$\bigwedge_{j=1}^{n} \tilde{x}_j^\gamma \leq x_i^\gamma. \tag{31}$$

This claim follows from the fact that since $m_{ij} = \bigvee_{\xi=1}^{k}(z_i^\xi - z_j^\xi) = 0$ for every $j = 1, \ldots, n$ and $Z$ is a minimal representation, we must have that $z_i^\xi = 0$ for every $\xi = 1, \ldots, k$. Hence, $z_i^\gamma = 0 \leq \tilde{x}_i^\gamma$ and

$$(M_{ZZ} \boxtimes \tilde{\mathbf{x}}^\gamma)_i = \bigwedge_{j=1}^{n}(m_{ij} + \tilde{x}_j^\gamma) = \bigwedge_{j=1}^{n} \tilde{x}_j^\gamma. \tag{32}$$

Due to the lower bound given by Eq.(31), components of $\mathbf{x}^\gamma$ can be arbitrarily corrupted and still satisfy Eq.(30) for the given index $i$ as long as there exists at least one index $j \in \{1, \ldots, n\}$ such that $\tilde{x}_j^\gamma \leq x_i^\gamma$. In many cases, Eq.(30) is automatically satisfied for a large number of indices $i$ whenever $Z$ is a minimal representation. These cases occur when $k \ll n$ as, for example, in the case of the seven image patterns shown in Figs.(1) or (6) where $k = 7 \ll 2500 = n$. If $n$ is large and $k \ll n$, then $n - k$, which is the cardinality of the set $I = \{i : z_i^\xi = 0 \; \forall \xi = 1, \ldots, k\}$, is also large. Since $m_{ij} = \bigvee_{\xi=1}^{k}(z_i^\xi - z_j^\xi)$ and $z_j^\xi > 0$ for at most one $\xi$, we have that $m_{ij} = 0$ for every $j = 1, \ldots, n$ whenever $i \in I$. This means that $M_{ZZ}$ contains $n - k$ rows having only zero entries. Hence Eq.(30) is satisfied for at least $n - k$ indices $i$.

Although Eq.(30) is guaranteed to be satisfied for at least $n - k$ indices $i$, the likelihood that it is satisfied for the remaining $k$ indices is also very high. Since $Z$ is a minimal representation, the inequality $z_i^\gamma \doteq 0 \leq \tilde{x}_i^\gamma$ is guaranteed for all $i$ except one. The only time the inequality may not hold is in the event that for one single index $j_\gamma$, $\tilde{x}_{j_\gamma}^\gamma < x_{j_\gamma}^\gamma = z_{j_\gamma}^\gamma$. The probability of this event occurring becomes small as $n$ increases. Also, since $M_{ZZ}$ acts as an erosive memory in that $M_{ZZ} \boxtimes \tilde{\mathbf{x}}^\gamma \leq \mathbf{x}^\gamma$, the expectation that $(M_{ZZ} \boxtimes \tilde{\mathbf{x}}^\gamma)_i \leq x_i^\gamma$ is dramatically enhanced for large $n$.

# 6 Conclusions

In this paper we describe a new technique for recalling stored patterns from noisy input patterns using morphological associative memories. We define the notions of morphological independence, strong independence, and minimal representations of patterns sets. We also refined the notion of kernels. Our new

notion of kernels generalizes the original concept of kernels and provides the major tool for dealing with noisy input pattern when using morphological memories. We established theorems that provide for the existence of minimal representations for strongly independent sets of patterns. We proved that these minimal representations are also kernels and provided the rationale for the preference of these type of kernels over general kernels for the recovery of patterns from noisy inputs. The proof given for the existence of a minimal representation is constructive and provides a method for the construction of kernels. Although we established sufficient conditions for the existence of kernels, we have been unable to establish necessary and sufficient conditions. It is our hope that fellow researchers in this new paradigm of neural computing will be able to solve this problem in the near future.

# References

[1] Birkhoff, G. *Lattice Theory.* American Mathematical Society, Providence, RI, 1984.

[2] Cuninghame-Green, R. *Minimax algebra. Lectures Notes in Economics and Mathematical Systems,* Vol. 166. New York, Springer, 1979.

[3] Cuninghame-Green, R. "Minimax algebra and applications," in *Advances in Imaging and Electron Physics,* P. Hawkes, (Ed.), vol. 90, pp.1–121, Academic Press, San Diego, CA, 1995.

[4] Hopfield, J. J. "Neural networks and physical systems with emergent collective computational abilities," in *Proceedings of the National Academy of Sciences,* Vol. 79, pp. 2554–2558, U.S.A., April 1982.

[5] Hopfield, J. J. "Neurons with graded response have collective computational properties like those of two state neurons," in *Proceedings of the National Academy of Sciences,* Vol. 81, pp. 3088–3092, U.S.A., May 1984.

[6] Kohonen, T. "Correlation matrix memory," *IEEE Transactions on Computers,* vol. C-21, pp. 353–359, 1972.

[7] T. Kohonen and M. Ruohonen. "Representation of associative data by computers," *IEEE Transactions on Computers,* vol. C-22, pp. 701–702, 1973.

[8] McEliece, R. et al. "The capacity of Hopfield associative memory," *Trans. Information Theory,* Vol. 1, pp. 33–45, 1987.

[9] Okajima, K., S. Tanaka, and S. Fujiwara. "A heteroassociative memory with feedback connection," in *Proc. IEEE 1st International Conf. on Neural Networks.* Vol. II, pp. 711–718, San Diego, CA, 1987.

[10] Palm, G. "On associative memory," *Biological Cybernetics.* Vol. 36. pp. 19–31, 1980.

[11] Ritter, G. X. *Image Algebra.* Unpublished manuscript available via anonymous ftp from `ftp://ftp.cise.ufl.edu/pub/scr/ia/documents`. University of Florida, Gainesville, FL. 1994.

[12] Ritter, G. X., P. Sussner, and J. L. Diaz de Leon. "Morphological associative memories," *IEEE Transactions on Neural Networks,* Vol. 9(2), pp. 281–293. March 1998.

[13] Ritter, G. X., G. Urcid, and L. Iancu. Reconstruction of noisy patterns using morphological associative memories. Submitted to the *Journal of Mathematical Imaging and Vision.* December 2001.

[14] Sussner, P. "Observations on morphological associative memories and the kernel method," Elsevier *Neurocomputing,* Vol. 31, pp. 167–183. 2000.

# Aplicación de la morfología matemática a la telemedicina y la biotecnología: caracterización morfológica de células de la sangre y análisis de *cDNA microarrays*

Jean Serra and Jesús Angulo
Centre de Morphologie Mathématique - Ecole des Mines de Paris,
35, rue Saint-Honoré, 77305 Fontainebleau, France
{serra,angulo}@cmm.ensmp.fr

**Resumen**

La morfología matemática es una metodología no-lineal de tratamiento de la imagen digital, basada en la aplicación de la teoría de retículos a estructuras espaciales. En este artículo, presentamos dos recientes aplicaciones del análisis morfológico a la imagen en biomedicina. La primera de ellas consiste en el desarrollo de una plataforma integrada para la cuantificación morfológica, el reconocimiento de patrones y la indexación basada en el contenido de células, a partir de imágenes en color de sangre periférica; considerando además el problema desde el punto de vista de la telemedicina. La segunda aplicación consiste en la automatización de la lectura de *cDNA micoraarays*. Esta nueva herramienta experimental de la biotecnología comienza a ser utilizada de manera masiva en los laboratorios y el análisis de imagen es un aspecto fundamental. Nuestro método permite una cuantificación completamente automática y precisa de la intensidad asociada a los spots, siendo además robusto frente a artefactos y variaciones indeseadas.

**Palabras-clave:** tratamiento digital de la imagen, indexación por el contenido, morfología matemática, segmentación morfológica, extracción de características, microscopía cuantitativa, telemedicina, biotecnología, citología hematológica, cDNA microarrays.

## 1 Introducción

En 35 años de desarrollo, la morfología matemática se ha convertido en una de las herramientas más utilizadas para el tratamiento de la imagen. La morfología matemática es una metodología no-lineal de tratamiento de la imagen digital, basada en la aplicación de la teoría de retículos a estructuras espaciales; y cuyos operadores son capaces de realizar sofisticadas tareas de procesamiento y análisis sobre imágenes 2D y 3D, a nivel de gris y en color. El lector interesado en profundizar en esta teoría puede consultar las obras de referencia de Serra [20]; o en el caso de querer conocer los últimos avances en teoría y aplicaciones, las tesis doctorales del *Centre de Morphologie Mathématique* de la *Ecole des Mines de Paris* constituyen una fuente indispensable. La morfología matemática ha demostrado en numerosas aplicaciones constituir una herramienta muy potente para el análisis de las imágenes de microscopía cuantitativa [18] [19].

En este artículo, describimos en detalle dos recientes aplicaciones, muy diferentes pero complementarias, del análisis morfológico sobre la imagen en biomedicina. La primera de ellas consiste en el desarrollo de una plataforma integrada para la cuantificación morfológica, el reconocimiento de patrones y la indexación basada en el

contenido de células a partir de imágenes en color de sangre periférica; considerando además el problema desde el punto de vista de la telemedicina. La segunda aplicación consiste en la automatización de la lectura de *cDNA microarrays*.

# 2 Caracterización morfológica de células de la sangre

## 2.1 Diagnostico morfológico en citología hematológica

Históricamente, el examen bajo microscopio de frotis sanguíneos de calidad ha sido la mejor manera de estimar el número y el tipo de los leucocitos o glóbulos blancos, para estudiar la morfología de los eritrocitos o glóbulos rojos (en el caso del paludismo, para detectar sus inclusiones parasitarias), para caracterizar la morfología de los linfocitos, para el estudio de las plaquetas, etc. En los últimos 30 años, se han desarrollado sistemas automáticos para el reconocimiento de los leucocitos, siendo ampliamente utilizados en rutina de laboratorio. Estos sistemas se basan en la citometría de flujo, pero están limitados a la identificación de células normales o a lo sumo, a la detección de las anormales. Sin embargo, el examen de frotis sanguíneos de sangre periférica continua siendo necesario para el análisis visual y la clasificación de células anormales, particularmente para el estudio de la morfología de los linfocitos, principio básico para la identificación y discriminación de los síndromes linfoproliferativos crónicos [23]. La clasificación de células de la sangre es un problema bien establecido en tratamiento de la imagen y en reconocimiento de formas, sobre el que se continúa trabajando [14] [10] [21].

## 2.2 Telepatología e interés de las láminas virtuales

Desde un punto de vista metodológico, hay dos cualidades necesarias para un frotis: un área suficiente de lectura sobre la lámina; y una calidad adecuada y reproducible del procedimiento de coloración. Ambas cualidades han sido previamente estudiadas [7] [8]; además, los aparatos para la preparación automática y coloración de los frotis, disponibles comercialmente, producen preparaciones relativamente buenas.

Diseñando una estrategia de automatización del estudio de los frotis y dentro de un contexto de telepatología hematológica [12] [13], el uso de estaciones de microscopía robotizadas permite la producción masiva de imágenes digitales microscópicas. La adquisición de estas imágenes permite su posterior almacenamiento, transmisión y tratamiento. La solución clásica pasaba por el concepto de dossier de imágenes (incluyendo una selección de imágenes de la lámina) pero que tiene el inconveniente de la falta de representatividad semántica del contenido de la lámina y que limita algunas aplicaciones. La solución alternativa, actualmente propuesta, es la lámina virtual o *teleslide* [16]: un *teleslide* es una imagen de alta resolución composición espacial de cientos o miles de imágenes del microscopio (al aumento usado para el diagnostico). Este tipo de macro-imágenes aseguran la representatividad del contenido de la lámina aunque, evidentemente, su gran tamaño implica la necesidad del desarrollo de herramientas especiales para su manipulación y análisis.

## 2.3 Plataforma para la hematología morfológica

En esta sección, presentamos la arquitectura de nuestro sistema *software* bajo la forma de una plataforma integrada para el tratamiento de la imagen, reconocimiento de patrones e indexación por el contenido de *teleslides* de sangre periférica. Este sistema de análisis de imagen, basado en operadores de la morfología matemática, proporciona una descripción objetiva de células de la sangre y más específicamente, el sistema ha sido diseñado para asistir a los hematólogos en el diagnostico de los linfoproliferativos. En la figura 1 se muestra el de la

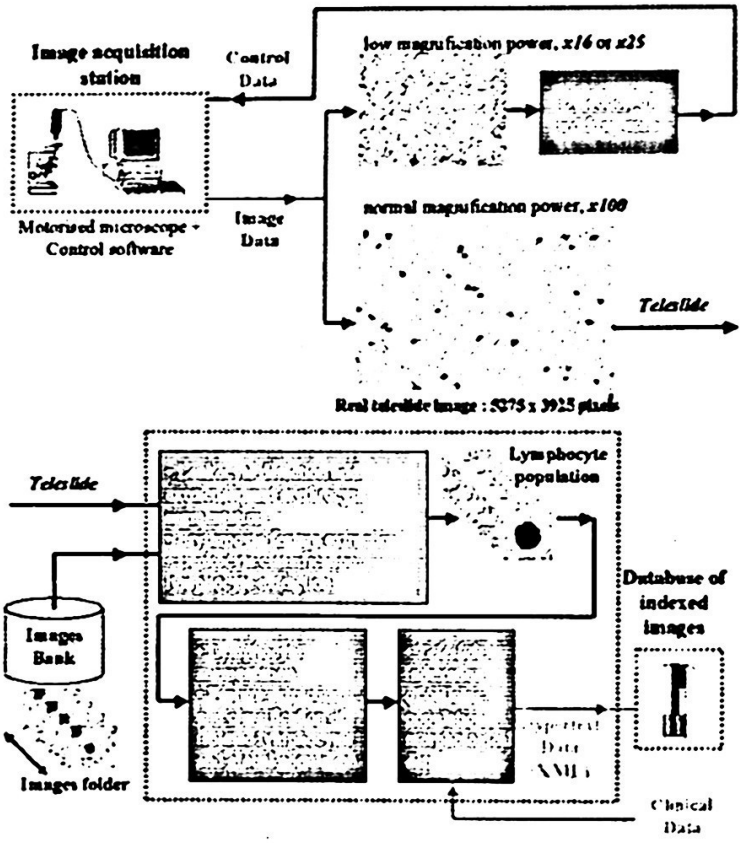



Figura 1: Análisis morfológico automático de células en frotis de sangre periférica: arriba, estación para la producción de los *teleslides*; y abajo, sistema integrado para el tratamiento de la imagen, reconocimiento de patrones e indexación basada en el contenido.

propuesta. Hemos separado claramente la producción de las imágenes *teleslides* (figura 1, arriba), del análisis de éstas (figura 1, abajo). De esta forma ambos subsistemas pueden estar separados físicamente y además, el análisis podrá realizarse sobre un dossier clásico de imágenes.

### 2.3.1 Automatización en la producción de los *teleslides* de frotis

Para facilitar la producción automática de las imágenes mediante un microscopio robotizado, hemos desarrollado un módulo de pretratamiento que implementa la detección automática de la zona de buena lectura del fotis a bajo aumento (×16 o ×25). Los detalles de la técnica pueden encontrarse en [4]. Cuando la zona óptima es detectada, el objetivo del microscopio se sube al aumento para diagnostico (×100), adquiriendo toda la serie de imágenes campo que serán compuestas para obtener la imagen *teleslide*, $f_{\times 100}$.

### 2.3.2 Segmentación de las imágenes de células de la sangre

Motivados por la necesidad de una técnica de segmentación óptima y robusta para las aplicaciones de citología hematológica, hemos desarrollado una estrategia jerárquica para la extracción de las entidades visuales de los *teleslides*. El esquema de segmentación consiste en tres módulos independientes (detalles en [1]).

El objetivo del primer módulo es la segmentación de $f_{\times 100}$, ver figura 2 izquierda, obteniendo de una parte la capa de los eritrocitos $f_{\times 100}^{eryth}$ (que puede ser utilizada para el control de calidad de la técnica de coloración así como para el estudio, en aplicaciones posteriores, de la morfología de los eritrocitos) y de otra, la posición (centroide) de cada uno de los $n$ leucocitos presentes, lo cual permite su posterior extracción en



Figura 2: Segmentación de las imágenes: izquierda, segmentación del *teleslide* para obtener la capa de los eritrocitos y para localizar y extraer los leucocitos; y derecha, algoritmo morfológico para segmentar el linfocito, extrayendo su núcleo y su citoplasma.

subimágenes que constituyen la serie de imágenes leucocitos $\{f_{\times 100}^{leuko}(i)\}_{i=1}^{n}$. El algoritmo de esta etapa de segmentación es relativamente simple: sobre la componente verde (la más contrastada) de la imagen color se realizan dos umbralizaciones para obtener todas las células en una imagen, y los núcleos de los leucocitos en otra imagen. Los umbrales óptimos se obtienen a partir de un modelo predefinido para el histograma. Sobre estas imágenes binarias se realizan una serie de filtrados para eliminar artefactos.

En un segundo nivel, el objetivo buscado es la segmentación de los linfocitos $f_{\times 100}^{lymph}$, obteniendo el núcleo $f_{\times 100}^{nucl\_ly}$ y el citoplasma $f_{\times 100}^{cyto\_ly}$ (ver en la sección siguiente la forma de detectar los linfocitos entre todos los leucocitos). El método para segmentar estas imágenes es la línea divisoria de aguas (*watershed*) aplicada sobre un gradiente suma del gradiente morfológico de la componente verde y del de la saturación (ambas imágenes previamente simplificadas con un filtrado por reconstrucción); y los marcadores del *watershed* son obtenidos utilizando una técnica de umbralizacion similar a la descrita anteriormente. Ver ejemplo con imágenes intermedias en figura 2 derecha.

El último módulo de segmentación consiste en la partición de la cromatina del núcleo en regiones con densidad similar. La organización de la textura de la cromatina es una característica importante a cuantificar. Debido al tamaño reducido del núcleo y de las partes a caracterizar, la utilización de la técnica del *watershed* no garantiza que las regiones sean correctamente separadas, y por eso hemos preferido elegir una técnica alternativa denominada conexión por saltos (*jump connection*), basada en la extensión de la noción de zona plana. La conexión por salto de tamaño $k$ desde los mínimos, se define como una conexión compuesta de todos los conjuntos conexos alrededor de cada mínimo, y donde el valor de la imagen es menor que $k$ encima del mínimo; de forma dual se define para los máximos, o se puede tomar la intersección de ambas conexiones, tal y como nosotros hacemos para la cromatina. Para mejorar esta segmentación hemos implementado a continuación un algoritmo de crecimiento de regiones sobre el retículo de particiones que se obtiene en la conexión por saltos. Todo este tratamiento lo realizamos sobre una imagen a nivel de gris, $f_{\times 100}^{nucl\_ly*}$, obtenida como la primera componente (aquella que maximiza la varianza) de un análisis por componentes principales sobre la imagen color $f_{\times 100}^{nucl\_ly}$. A partir de la partición final obtenida, cada región podrá ser caracterizada, constituyendo el descriptor de la organización de la cromatina.

### 2.3.3 Clasificación de los leucocitos

Los leucocitos pueden ser clasificados en cinco categorías: (1) monocitos, (2) neutrófilos, (3) basófilos, (4) eosinófilos y (5) linfocitos. Sobre las imágenes $\{f_{\times 100}^{leuko}(i)\}$, las cinco clases de leucocitos pueden ser diferenciadas de acuerdo con las carac-

terísticas morfológicas (forma y textura) y de color. Para este propósito, hemos introducido un algoritmo de clasificación usando técnicas estadísticas (*template matching*, utilizando diferentes distancias sobre los histogramas como medida de similitud entre las plantillas y las entradas a clasificar). El color es caracterizado mediante los histogramas en los espacios HLS y Lab (con una cierta normalización basada en el color del plasma, es decir, en el fondo de la imagen) y la forma-textura es descrita mediante granulometrías morfológicas clásicas (familias de aperturas/cierres de tamaño creciente) extendidas al color. Después de una etapa de aprendizaje, donde diferentes clasificadores has sido evaluados, se ha propuesto un clasificador que utiliza básicamente la información de las granulometrías combinada con las características de color y cuyos resultados son bastante satisfactorios, para más detalles ver [5]. Mediante este módulo, a cada imagen $f_{\times 100}^{leuko}(i)$ se le asignará la categoría de leucocito correspondiente y sobre todo, podremos detectar los linfocitos, cuyas imágenes pasarán a denominarse $\{f_{\times 100}^{lymph}(j)\}_{j=1}^{m}$, y que constituyen la población de $m$ células a caracterizan de forma detallada.

### 2.3.4 Extracción de características de los linfocitos y su clasificación

Tomando las imágenes color segmentadas del núcleo y del citoplasma, $f_{\times 100}^{nucl\_ly}(j)$ y $f_{\times 100}^{cyto\_ly}(j)$ respectivamente, como punto de partida, todo un conjunto de parámetros cuantitativos deben ser calculados.

Hemos definido alrededor de 30 parámetros de forma, textura y color; incluyendo medidas complejas de textura como las curvas granulométricas para un análisis complementario de la cromatina, la detección y extracción de nucleolos, etc., ver en la figura 3 algunos ejemplos. Estas medidas proporcionan una primera clasificación de cada una de las características morfológicas retenidas (1- tamaño del núcleo, 2- tamaño de la célula, 3- ratio núcleo/célula, 4- densidad de la cromatina, 5- forma del núcleo, 6- nucleolo, 7- basofilia del citoplasma, 8- granulaciones del citoplasma, 9- forma del citoplasma, 10- excentricidad del núcleo) en una categoría: por ejemplo, el tamaño del núcleo podrá ser muy pequeño, pequeño, medio o grande. El clasificador asociado a cada característica morfológica ha sido obtenido mediante técnicas clásicas de aprendizaje a partir de una base de datos de referencia, en un contexto de lógica difusa. Para cada célula, el conjunto de valores de todas las características morfológicas constituyen el descriptor del linfocito.

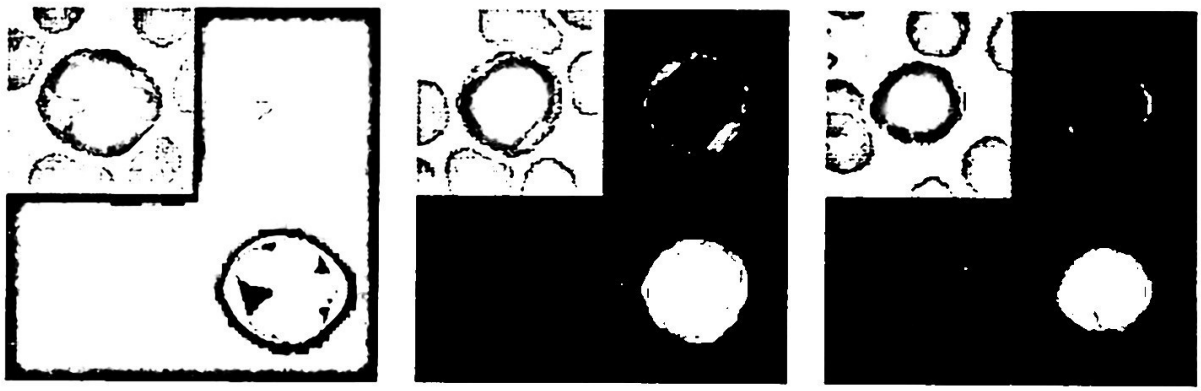


Figura 3: Ejemplos de la extracción de características: izquierda, cuantificación de la forma de un núcleo multilobular; centro, detección y extracción del nucleolo; derecha, detección de hendiduras en el núcleo.

En la etapa siguiente, este descriptor morfológico del linfocito (que ya puede ser incluso interpretado por un utilizador humano) es la entrada para un algoritmo de clasificación del linfocito en una de las 20 categorías celulares que se han definido, en

base al consenso relativo histórico de la hematología. Este clasificador es básicamente un árbol de decisión difuso.

Ya que una categoría celular será asociada a cada uno de los $m$ linfocitos de la población, después de forma estadística deberán definirse las categorías o subpoblaciones presentes así como su presencia relativa.

### 2.3.5 Indexación de la información

Una vez que el conjunto de los descriptores morfológicos y de la información de la clasificación han sido obtenidos, mediante el modulo de indexado toda esta descripción resultante (imágenes y datos alfanuméricos: *metadata*) toma la forma de un documento en formato XML (utilizado en la norma MPEG-7, ya que es muy eficiente para la edición, búsqueda, filtrado y procesado de la información en bases de datos). Esta descripción morfológica proporciona una información objetiva que podrá ser indexada junto con el *teleslide*, así como con otros datos complementarios (características clínicas, inmunofenotipado, genética, etc) en una base de datos.

# 3 Análisis de *cDNA microarrays*

## 3.1 *cDNA microarrays* y análisis de imagen

Los *cDNA microarrays* [1] constituyen una tecnología experimental para identificar y cuantificar niveles de expresión en genes, y que consisten en matrices de miles de secuencias discretas de ADN depositadas sobre laminas para microscopio u otro soporte similar. Para comparar la abundancia relativa de los genes a estudiar en dos muestras de ADN, las dos muestras son etiquetadas usando dos tintes fluorescentes (Cy5 y Cy3), entonces son mezcladas con los spots de ADN depositados, interactuando biológicamente. Después de la hibridación, se realizan separadamente medidas de fluorescencia sobre cada tinte. Estas medidas son utilizadas para determinar el ratio, es decir, la abundancia relativa, de cada secuencia de gen en las dos muestras [9].



(a) (b)

Figura 4: (a) Imagen típica de una matriz (tamaño 2200 × 3000 pixels) y zoom, conteniendo miles de spots. (b) Variacion de la intensidad en tres spots (3D y corte horizontal).

Típicamente, las matrices varían entre algunos cientos hasta muchos miles o más de spots (cada pixel de la imagen es equivalente a $10\mu m$), figura 4(a). El tratamiento de la imagen es un aspecto importante en los experimentos con *cDNA microarrays* ya que las intensidades extraídas pueden tener una impacto potencialmente importante en los subsiguientes pasos de análisis de datos. El objetivo básico es reducir la imagen de los spots de diferentes intensidades a una tabla con una medida de intensidad para cada spot. El principal inconveniente está en el hecho que los spots

[1] En este artículo hemos decidido conservar el termino inglés *cDNA microarray* (en la literatura anglosajona también se designan como *biochips* o *genomic arrays*) debido a la falta de consenso sobre un nombre en español para esta tecnología, causado principalmente por la falta de publicaciones científicas en español.

son construidos utilizando un robot que dispone de una serie de pins que transfieren pequeñas cantidades de ADN. Por consiguiente, el desplazamiento mecánico del robot puede generar una distorsión geométrica y la imposibilidad del alineamiento de una plantilla. Otras técnicas intentan una segmentación automática de los spots usando un algoritmo de umbralización de intensidad adaptativo. Sin embargo, importantes errores son causados por la difícil elección del umbral óptimo (ver figura 4(b)): la frontera entre spot-fondo no es abrupta; el contraste (altura) entre las regiones spot-fondo y el volumen (integral de la intensidad) son muy diferentes de un spot al otro; y además la no homogeneidad del proceso de hibridación hace que las regiones de los spots estén "rotas".

Varias técnicas y herramientas software han sido desarrolladas para ser utilizadas en el procesamiento de las imágenes de las matrices [11] [15] [22] [24]. En es sta sección, presentamos nuestra técnica automática y no-supervisada basada en la morfología matemática para detectar las regiones de la imagen correspondientes a los spots y cuantificar sus intensidades asociadas.

## 3.2 Perspectiva general del algoritmo

La técnica morfológica para el tratamiento de las imágenes de microarrays se divide en seis sub-algoritmos. Los detalles completos de estos pasos son dados en [2]. La entrada es el par de imágenes escaneadas y la salida es la intensidad asociada a cada uno de los spots de cada imagen. Los principales pasos pueden ser resumidos cómo sigue (ver figura 5).

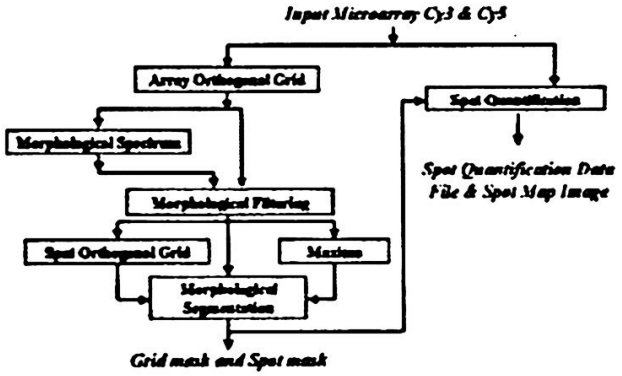



Figura 5: Diagrama de flujo de los algoritmos morfológicos para el procesamiento de las imágenes de microarrays.

### 3.2.1 Rejilla ortogonal del array

Inicialmente, un algoritmo de mallado debe proporcionar la segmentación automática, mediante una rejilla, de la imagen de la matriz completa en sub-matrices, definiendo cada grupo de spots o sub-imágenes, las cuales serán analizadas individualmente. Sean $f_{Cy3}$ y $f_{Cy5}$ las imágenes de 16-bits escaneadas de las fluorescencias Cy3 y Cy5. Nuestro algoritmo para el tratamiento de las imágenes requiere un imagen simple y además, computacionalmente es conveniente que sea de 8-bits. La imagen propuesta $f$ es obtenida por medio de una *combinación lineal sopesada con los valores de mediana* $\nu$, después de una transformación raíz cuadrada; i.e. $f = (\nu_{Cy3}\sqrt{f_{Cy3}} + \nu_{Cy5}\sqrt{f_{Cy5}})/(\nu_{Cy3} + \nu_{Cy5})$. Para segmentar los grupos de spots, la imagen $f$ es reducida en tamaño por medio de un *diezmado con promediado* (tamaño $K = 4$); i.e., $\widetilde{f} = f \div 4$. Entonces, los grupos de spots son realzados mediante el supremo de un cierre vertical $\varphi_n^\pi$ y un cierre horizontal $\varphi_n^0$; i.e, $\widetilde{f}^\bullet = \varphi_n^\pi(\widetilde{f}) \vee \varphi_n^0(\widetilde{f})$. El tamaño $n$ del cierre así como otros parámetros de tamaño dependen de cada microar-

ray; y en la sección siguiente será presentada una técnica espectral morfológica para calcular el tamaño aproximado del spot y por consiguiente, el valor de los parámetros de tamaño. A partir de la imagen realzada, son calculadas las proyecciones horizontales y verticales. El objetivo del subsiguiente filtrado morfológico unidimensional es simplificar la señal proyección, eliminando la contribución de los spots y del ruido de fondo. Sea $P^{\tilde{f}^\bullet}(i)$ la proyección horizontal o vertical, el tratamiento de la señal para obtener la rejilla se ejecuta en tres pasos: *filtrado intra-bloque* usando una apertura de tamaño $n_{ib}$; i.e., $P_{ib}^{\tilde{f}^\bullet} = \gamma_{n_{ib}}(P^{\tilde{f}^\bullet}(i))$; *extracción de los bloques* por medio de una transformación de chistera (*top-hat*) de tamaño $n_b$, que extrae los bloques; i.e., $P_b^{\tilde{f}^\bullet} = P_{ib}^{\tilde{f}^\bullet} - \gamma_{n_b}(P_{ib}^{\tilde{f}^\bullet})$; y *umbralización*, el valor óptimo para el umbral $u_P$ es definido como el 20% del valor medio de $P_b^{\tilde{f}^\bullet}$. Después de la umbralización, una señal binaria unidimensional (bloques y fondo) es utilizada para definir el origen $(x_0(j), y_0(j))$ y las dimensiones $[xsize(j), ysize(j)]$ para cada bloque $j$ que son extraídos a partir de la imagen $f$. Entonces, el análisis de las imágenes des los grupos de spots $\{f^j\}$ se lleva a cabo en cinco pasos.

### 3.2.2 Ley de distribución de tamaño de spot

En [3], ha sido presentado el espectro morfológico de extinción (histograma de los extremos) caracterizado por tres parámetros o medidas (contraste, área o volumen) así como su aplicación para el análisis de imágenes de matrices genómicas. Una nueva herramienta: *la ley de distribución de tamaño de spot* ha sido introducida, basada en el espectro de extinción por área en escala logarítmica, $SS[\lambda, f^j]$, donde $\lambda$ es el tamaño del spot (área) y $SS[\lambda, f^j] = n_\lambda$ denota el número normalizado de ocurrencias del valor de extinción $\lambda$ (se trata de una función densidad de probabilidad). El $SS[\lambda, f^j]$ está usualmente constituido de varios modos (el modo más interesante es el primero y los otros pueden considerarse como armónicos de la frecuencia morfológica fundamental). Este modo fundamental, o tamaño medio de spot, proporciona el umbral de área para el subsiguiente filtrado.

### 3.2.3 Filtrado morfológico por área

La contribución del ruido de fondo puede ser importante e introducir errores en la construcción de la rejilla y en la definición de la frontera de los spots. La extracción de este ruido de fondo se obtiene usando un *filtrado morfológico por área* $\gamma_n^a$. Usando esta técnica de filtrado sobre la imagen grupo de spots $f^j$; i.e., $f_s^j = \gamma_{\lambda_s}^a(f^j)$, las estructuras con un área mayor o igual que un umbral dado $\lambda_s$ son preservadas en $f_s^j$. Así pues, el valor del umbral $\lambda_s$ óptimo para cada imagen es una elección importante que depende del tamaño de los spots, siendo esta la justificación del calculo de $SS[\lambda]$. Otra ventaja importante del filtrado morfológico por área es la selección implícita de los máximos: después de este filtrado por reconstrucción hipotéticamente hay un y solamente un máximo asociado a cada spot.

### 3.2.4 Rejilla ortogonal de los spots

Dentro de un grupo de spots, éstos están situados de acuerdo con un alineamiento ortogonal y de nuevo, utilizando las proyecciones horizontal y vertical, se debe obtiener una rejilla correspondiente a los spots. El algoritmo es como sigue. Sea $P(i)$ la proyección horizontal y vertical: (1) calcular el valor medio de los elementos de $P(i)$; i.e, $\overline{P} = \frac{1}{N}\sum_{i=1}^{N} P(i)$; (2) sustraer la media de la proyección; i.e, $P_\eta(i) = P(i) - \overline{P}$; (3) reconstrucción morfológica de $P(i)$ utilizando $P_\eta(i)$ cómo marcador; i.e, $P^{rec}(i) = \gamma^{rec}(P(i); P_\eta(i))$; (4) tomar el residuo de la proyección inicial $P(i)$ y la reconstrucción $P^{rec}(i)$; i.e, $P_{TH}(i) = P(i) - P^{rec}(i)$; (5) estimar el

valor del umbral óptimo $u_P$, definido como $\alpha\%$ de la media del residuo $P_{TH}(i)$; i.e., $u_P = \frac{\alpha}{100}\frac{1}{N}\sum_{i=1}^{N} P_{TH}(i)$; (6) encontrar la señal binaria referencia $P_u(i)$ por medio de la umbralización en $u_P$ sobre la señal residuo $P_{TH}(i)$ y usar el $i$ medio de cada intervalo igual a 1 en $P_u(i)$, trazando la líneas rectas que corresponderán a la rejilla ortogonal. Después de diferentes tests en nuestra base de datos, la elección de $\alpha = 50\%$ aparece como un valor relativamente bueno.

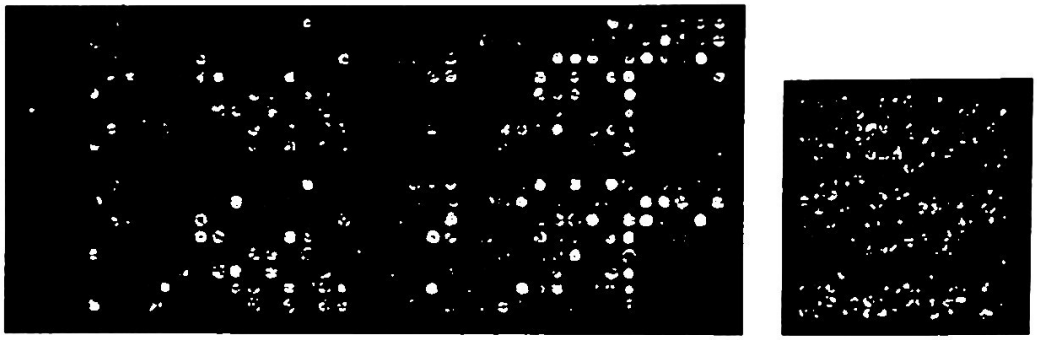


Figura 6: Dos ejemplos de segmentación de los spots.

### 3.2.5 Segmentación morfológica de los spots

La segmentación de los spots o definición de sus fronteras, se lleva a cabo utilizando la transformación de la línea divisoria de aguas (*watershed*): (1) la *función a inundar* es el gradiente externo $g^+$, definido cómo la diferencia entre la imagen dilatada y la imagen original; i.e, $g^+(f_{ss}^j) = \delta_B(f_{ss}^j) - f_{ss}^j$ (antes, la imagen filtrada $f_s^j$ es simplificada de nuevo por medio de un filtro de nivelado (*leveling*); i.e., $f_{ss}^j = \Lambda(f_s^j)$); (2) los *marcadores externos* $g_r$ serán los bordes de la imagen rellenados hasta el principio de la rejilla (marcadores del fondo) junto con la rejilla misma (marcadores de las regiones de influencia de los spots) y para los *marcadores internos* $mki$ se ha desarrollado un algoritmo específico que se basa en un análisis individual de la región de influencia rectangular de cada spot definida por la rejilla, y de los máximos después del filtrado por área; los marcadores globales a imponer vendrán dados por $mk = g_r \vee mki$; (3) construcción de la *línea divisoria de aguas* de cada spot $sm = Wshed(g^+(f_{ss}^j, mk)$, donde $sm$ son los contornos cerrados de cada spot. En la figura 6 son mostrados dos ejemplos de segmentación de los spots.

### 3.2.6 Cuantificación de los spots y extracción de ruido

La capa de segmentación $sm$, obtenida anteriormente se utiliza sobre las imágenes iniciales de 16-bits $f_{Cy3}$ y $f_{Cy5}$ para la cuantificación de los spots y la extracción del ruido. La intensidad calculada (integral en la región del spot) puede ser expresada como la suma de un *valor de intensidad de señal*, $s_i$, y un *valor de intensidad de ruido*, $n_i$, tal que $\hat{s}_i = s_i + n_i = s_i + \bar{N}_i A_i$, dónde $\bar{N}_i$ es el *ruido medio* del spot $i$ y $A_i$ es el *área* (número de pixels) del spot $i$. Brevemente, hay dos alternativas a estudiar. Primero, considerar que el ruido de fondo es uniforme sobre toda la matriz (más precisamente sobre todo el bloque $j$); o segundo, considerar que el ruido de fondo no es uniforme, y por consiguiente es necesario una estimación local por spot. Obviamente, la técnica de *ruido de fondo global* es más simple: hay que cuantificar solamente un valor medio, mientras que para el modelo de *ruido de fondo local* es necesario cuantificar el ruido en una región para cada spot. Basándose en la teoría de la geoestadística de Matheron [17], si suponemos que la imagen $f(x,y)$ es una variable regionalizada, para estimar la media de esta variable en el área $S$ cómo $\mu = 1/S\sum\sum f(x,y)$, la varianza estimada debe seguir la ley $\sigma^2(0/S) = 1/S^2C$ (la varianza deber variar inversamente con respecto al área al cuadrado). Para poder

verificar esta hipótesis, se han realizado algunos tests que implican el muestreo del fondo utilizando diferentes áreas de muestra y el cálculo de la varianza asociada a estas áreas (ver ejemplo numérico en la figura 7). Como conclusión, podemos afirmar que las variaciones locales del ruido de fondo a la escala del tamaño del spot no permiten una estimación global del ruido medio. La rejilla ortogonal de los spots proporciona una segmentación adicional: las regiones-cajas rectangulares $BS_i$ que son consideradas las regiones de influencia de los spots $S_i$. Estas regiones pueden ser usadas para cuantificar el ruido local asociado a cada spot. De forma práctica, una zona envolvente de protección (para evitar el sesgo introducido en la vecindad de la región del spot) es obtenida mediante el residuo de una dilatación de la región del spot; y el ruido $\check{n}_i$ se estima en la región $BS_i - \delta_n(S_i)$ que tiene un área de $\check{A}_i$ pixels (el tamaño típico de la dilatación es $n = 3$). La expresión global para la señal de intensidad viene dada por $s_i = \hat{s}_i - n_i = \hat{s}_i - \check{N}_i A_i = \hat{s}_i - \frac{\check{n}_i}{\check{A}_i} A_i$.

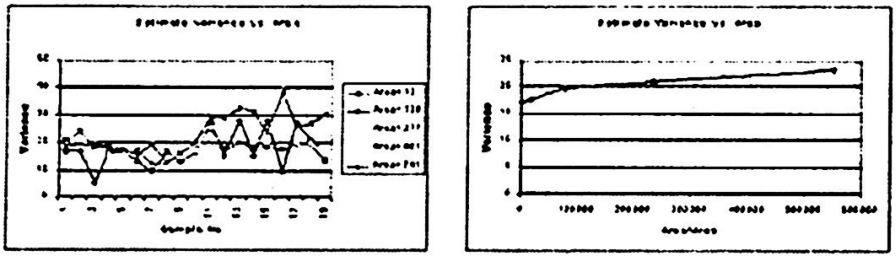


Figura 7: Muestreo del ruido de fondo utilizando diferentes áreas de muestra y estimación de la varianza: izquierda, valores estimados de la varianza y derecha, área al cuadrado con respecto a la varianza.

## 4 Conclusiones

En este artículo hemos presentado la utilización satisfactoria de operadores morfológicos para el análisis cuantitativo de la imagen en microscópica aplicado a la citología hematológica y a los *cDNA microarrays*.

El conjunto de algoritmos de la plataforma de hematología constituyen una herramienta global para el análisis e indexación de imágenes *teleslides* de frotis sanguíneos, la cual proporciona una descripción-clasificación objetiva y automática de células de la sangre; habiendo sido diseñada para asistir a los hematopatólogos en el diagnóstico de afecciones hematológicas. En una fase posterior, la extensión de nuestra plataforma al estudio de la malaria (paludismo) sería desarrollada; por ejemplo, la morfología de los eritrocitos (forma y color) y las inclusiones eritrocitarias arrojan un conocimiento fundamental para la detección del estadio y gravedad de la malaria. Los algoritmos se han basado en tests metodológicos en una amplia base de imágenes (más de 3000). Añadiendo capacidades telemáticas al sistema, es decir, integrándola en red, ver figura 8, pueden desarrollarse nuevas aplicaciones tal como: bases epidemiológicas, diagnostico colaborativo, tele-hematología, etc. Además, actualmente se está llevando a cabo una experiencia de tele-consenso, implicando la opinión de cuatro expertos, con el objetivo de validar los descriptores y los resultados de la clasificación [6].

La metodología descrita para el estudio de los *cDNA microarrays* permite resolver los problemas principales de estas imágenes: forma de los spots, fuerte ruido global de fondo, señales localmente débiles, etc. Como ya hemos dicho, nuestra propuesta es completamente automática y la segmentación de los spots consiste en una técnica adaptativa en posición, forma y tamaño. La evaluación experimental del resultado de la segmentación y cuantificación de los spots muestra que el uso de estos algo-

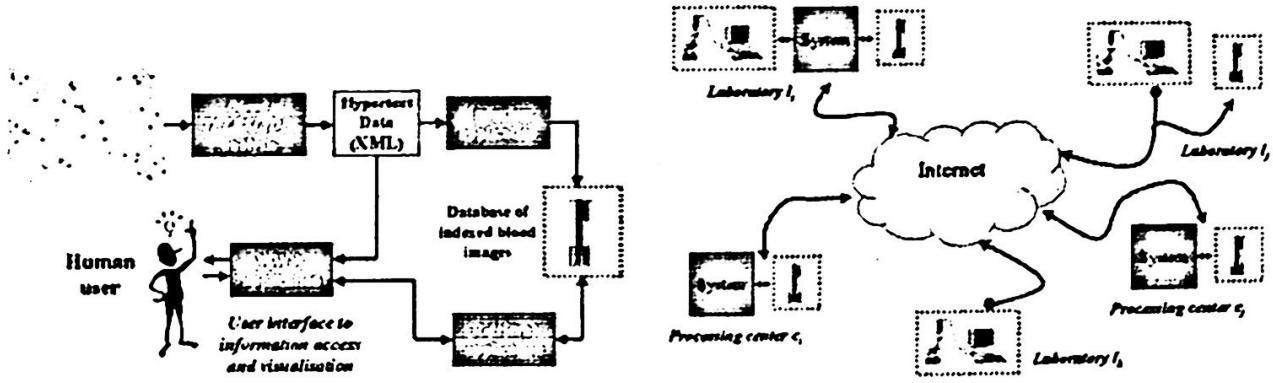



Figura 8: Aplicaciones de la plataforma de hematología: izquierda, Asistente para el diagnóstico y la consulta basada en el contenido morfológico de bases de datos epidemiológicas; y derecha, escenario en red (laboratios con o sin capacidad de tratamiento, bases de datos centralizadas y locales, capacidad de procesamiento distribuida, etc.)

ritmos son generalmente iguales o mejores que aquellos obtenidos mediante técnicas convencionales manuales, ver estudio comparativo en [2].

## References

[1] Angulo, J. and Serra, J. (2001) Semantic-based hierarchical segmentation of color blood smear images using mathematical morphology. *Internal Note N-50/01/MM, CMM-Ecole des Mines de Paris*, August 2001, 28 pp.

[2] Angulo, J. and Serra, J. (2002a) Automatic analysis of DNA microarray images using mathematical morphology. *Submitted to Bioinformatics*, January 14th 2002 (revised version on August 26th), 50 pp.

[3] Angulo, J. and Serra, J. (2002b) Morphological spectrum applied to genome arrays quantification. *Submitted to Journal of Visual Communication and Image Representation*, January 14th 2002, 20 pp.

[4] Angulo, J. and Flandrin, G. (2002c) Automated detection of working area of peripheral blood smears using mathematical morphology. *To appear in Analytical Cellular Pathology*, July 1st 2002, 35 pp.

[5] Angulo, J. and Serra, J. (2002d) Morphological color size distributions for image classification and retrieval. *Proc. of ACIVS'2002 (Advanced Concepts for Intelligent Vision Systems)*, Ghent, Belgium, pp. 46–53.

[6] Angulo, J., Benattar, L., Lusina, D., Martelli, J.M. and Flandrin, G. (2002e) Inter-observer consensus on lymphoid cell morphology, compared to obective measurement by mathematical morphology procedure. *In preparation.*

[7] Benattar, L. and Flandrin, G. (1999a) Morphometry and Quality Control for a May-Grünwald Giemsa stained preparation. A 40 centers cooperative study. *Leukemia and Lymphoma*, Vol. 33, pp. 587-589.

[8] Benattar, L. and Flandrin, G. (1999b) Comparison of the classical manual pushed wedge films, with an improved automated method for making blood smears. *Hematology and Cell Therapy*, Vol. 41, pp. 1-5.

[9] Brown, P. O. and Botstein, D. (1999) Exploring the New World of the genome with DNA microarrays. *Nature Genetics*, Vol. 21 (supplement), pp. 33–37.

[10] Comaniciu, D., Meer, P. and Foran, D. J. (1999) Image-guided decision support system for pathology. *Machine Vision and Applications*, Vol. 11, pp. 213–224.

[11] Chen, Y., Dougherty, E. R. and Bittner, M. L. (1997) Ratio-based decisions and the quantitative analysis of cDNA microarray images. *Journal of Biomedical*

*Optics*, Vol. 2, pp. 364–374.

[12] Flandrin, G. (1995) Haematological cytology image bank and teletransmision for microscopic diagnosis. *Arch. Ana. Cytol. Path.*, Vol. 43, pp. 257-261.

[13] Flandrin, G. (1997) Image Bank, diagnostic codification and telediagnosis in hematology. *Leukemia and Lymphoma*, Vol. 25, pp. 97-109.

[14] Haus, H. M., Harms, H., Ter Meulen, V. and Gunzer, U. (1987) Statistical evaluation of computer extracted blood cell features for screening populations to detect leukemias. Pattern Recognition Theory and Applications. *Edited by P. A. Devjver and J. Kittler, Spring-Verlag Berling,* Heidelberg.

[15] Hirata JR, R., Barrera, J., Hashimoto, R. F. and Dantas, D. O. (2001) Microarray Gridding by Mathematical Morphology. *Proc. SIBGRAPI, International Symposium on Computer Graphics, Image Processing and Vision 2001*, Florianopolis, IEEE Computer Society, pp. 112–119.

[16] Klossa, J., Flandrin, G. and Hémet, J. (2002) Teleslide: Better slide representativeness for digital diagnostic microscopy applications. *6th Congress of European Group of Telepathology*, Crete, Greece, 14-21 September.

[17] Matheron, G. (1975) Random Sets and Integral Geometry. *Wiley*, New York.

[18] Meyer, F. (1986) Automatic Screening of Cytological Specimens. *Computer Vision, Graphics, and Image Processing*, Vol. 35, pp. 356-369.

[19] Seron, D., Moreso, F., Gratin, G. *et al.* (1996) Automated Classification of Renal Interstitium and Tubules by Local Texture Analysis and a Neural Network. *Analytical and Quantitative Cytology and Histology*, Vol. 18, pp. 410-419.

[20] Serra, J. (1982,1988) Image Analysis and Mathematical Morphology. Vol I and II. *Academic Press*, London.

[21] Theera-Umpon, N. and Gader, P. D. (2000) Counting white blood cells using morphological granulometries. *Journal of Electronic Imaging*, Vol. 9, pp. 170–177.

[22] Vesanen, P., Tiainen, M. and Yli-Harja, Olli (2002) On Calibration-Free Methods in Segmentation of cDNA Microarray Images. *IS&T/SPIE Symposium Image Processing: Algorithms and Systems. Proceedings of SPIE Vol. 4667*, 12 pp.

[23] World Health Organization (2001) Pathology and Genetics of Tumours of Haematopoietic and Lymphoid Tissues. *Edited by E. S. Jaffe, N. L. Harris, H. Stein and J. W. Vardiman, IARC Press*, Lyon, France, 2001.

[24] Yang, Y. H., Buckley, M. J., Dudoit, S. and Speed, T. P. (2002) Comparison of methods for image analysis on cDNA microarray data. *Journal of Computational and Graphical Statistics*, Vol. 11, pp. 108–136.

# Filtros Morfológicos por Pendiente basados en Criterios de Gradiente Ponderado

Jorge D. Mendiola- Santibañez[1], Iván R. Terol-Villalobos[2]
[1]Doctorado en Instrumentación y Mecatrónica de la Facultad de Ingeniería de la Universidad Autónoma de Querétaro, Centro Universitario, Cerro de las Campanas S/N, APDO 76000, Qro., México.
[2]CIDETEQ, SC, Parque Tecnológico Querétaro S/N, Sanfandilia-Pedro Escobedo, C.P. 76700-APDO 064, Qro. México. E-mail: famter@ciateq.net.mx

**Resumen**

El presente artículo, tiene como objetivo principal la modificación del criterio de gradiente en los filtros morfológicos por pendiente mediante el uso de funciones ponderadoras. La finalidad de estas funciones es ponderar al gradiente para modificar su comportamiento y asi detectar varias zonas dentro de la imagen como de alto o bajo contraste de acuerdo a las características de la función propuesta. De esta manera, la ponderación del gradiente en los filtros morfológicos por pendiente surge como una posibilidad importante para mejorar el contraste en las imágenes de salida. Finalmente se plantea el uso de transformaciones conexas basadas en la noción de partición y zona plana.

**Palabras clave.** Gradiente morfológico, gradiente ponderado, filtros morfológicos por pendiente, morfología sobre particiones, contraste morfológico.

## 1. Introducción

Dentro de la morfología matemática, uno de los tópicos de interés es el estudio del contraste morfológico. El contraste morfológico fue inicialmente estudiado por Meyer y Serra [1] mediante los mapeos de contraste morfológicos. Estos mapeos, basan su funcionamiento en la modificación de la imagen de entrada mediante primitivas bajo un criterio de proximidad. Por otra parte; surge una clase de filtros no crecientes denominados filtros morfológicos por pendiente (FMP) [2,3,4]. Este tipo de filtros usan el criterio de pendiente para modificar a las imágenes y obtener un buen

contraste. Al aplicar estos filtros, las zonas de la imagen con pendientes débiles son eliminados y zonas con pendientes fuertes son mantenidas como regiones de alto contraste. La detección de contraste en las diferentes regiones de la imagen, se lleva a cabo al aplicar los conceptos de gradientes morfológicos [5][6]. Los FMP se definen como mapeos de contraste, al igual que los propuestos por Serra y Meyer; pero con diferencias significativas. Por ejemplo, los FMP al ser iterados, no comparan una función f con diferentes patrones de la imagen original, sino que retienen zonas de alto contraste y eliminan aquellas donde el gradiente es débil. También los FMP involucran dos parámetros importantes durante su iteración; la talla del elemento estructural del gradiente y una constante $\phi$ como el valor de la pendiente; por lo tanto, es posible encontrar diferentes descriptores de la imagen de entrada al variar dichos parámetros. Por otra parte, Terol [7] propone la ponderación del gradiente en los FMP; esto, con la finalidad de mostrar que al tener un control sobre el mismo, se puede influir de manera sustancial en el resultado final. En el presente trabajo, la propuesta gira alrededor de la ponderación del gradiente. Básicamente, se manipula al gradiente por medio de una función o composición de funciones; las cuales refuerzan o atenúan los contornos detectados. Entre las funciones propuestas, se presenta una función en particular de sumo interés y que corresponde a la ley de Weber [8]. Esta es una ley física de contraste y se encuentra descrita por un logaritmo. Otra parte fundamental de este trabajo, esta basado en el formalismo de las diferentes expresiones morfológicas ya que se definirán sobre el concepto de partición y zona plana [9]. Al trabajar con estos conceptos, se asegura que las diferentes transformaciones propuestas son conexas; es decir, no hay generación de nuevos contornos en la imagen de salida, solo se extienden o desaparecen las zonas planas. Finalmente, este artículo esta organizado de la siguiente manera: en la sección 2 se presenta la definición de zona plana, así como las transformaciones morfológicas sobre partición, mientras que en la sección 3, se proponen las diferentes funciones que ponderan al gradiente y se comparan los resultados de las imágenes de salida a nivel partición y a nivel pixel.

# 2. Conectividad y Morfología Sobre Partición

## 2.1. Nociones Básicas de Conectividad y Zona Plana

La noción de conectividad es un concepto definido en topología que se establece también bajo otra forma en la teoría de grafos. Esta noción tiene importantes aplicaciones en procesamiento de imágenes por computadora: las componentes conexas de una imagen corresponden a sus diferentes

partes que la componen. En el caso puramente digital la noción de trayectoria arco-conexa permite definir una componente conexa. Una trayectoria arco-conexa P entre dos puntos $x$, $y$ es una secuencia de puntos, $p_1,p_2,...p_n$, tal que $p_1 = x$, y $p_n = y$, y para todo $i$ $(p_i,p_{i+1})$ son vecinos. De esta forma, una componente A es arco-conexa si para todo par de píxeles $x$, $y$ en A, existe una trayectoria completamente incluida en A uniendo $x$ e $y$. La existencia de una nueva definición de conectividad permitiría separar de otra manera un objeto en sus diferentes partes conexas. Esta es una de las razones que originó la definición de conectividad en morfología matemática. La conectividad fue establecida por Serra [10], por medio del concepto de clase conexa. Esta noción de conectividad generó el concepto de zona plana. El concepto de zona plana es muy importante porque permite definir las componentes conexas en imágenes binarias, así como en imágenes numéricas. Esta noción se expresa en la siguiente definición.

**Definición 1**. Las zonas planas de una función f: $Z^2 \rightarrow Z$ se definen como las componentes conexas (más grandes) de puntos con el mismo valor de la función.

En la figura 1(a), se muestra una imagen conteniendo 5 zonas planas, mientras que la Fig. 1(b) y 1(c) ilustra las componentes conexas obtenidas por el operador $F_x(f)$, $F_y(f)$. Observe que la noción de zona plana genera una partición de la imagen puesto que verifica las siguientes condiciones: (a) La unión de zonas planas recubre completamente el dominio de la imagen, (b) Si la intersección de dos zonas planas diferentes es el vacío; las zonas planas son disjuntas. En particular, todo punto $x$ pertenece exactamente a un elemento de la partición. De este modo, es posible usar ambos conceptos sobre particiones: noción de zona plana que genera la partición y las transformaciones morfológicas que trabajan a nivel de vecindades. Para definir la noción de vecindad sobre la partición generada por la noción de zona plana se requiere definir la adyacencia entre elementos de la partición.

**Definición 2**. Dos zonas planas $F_x(f)$ y $F_y(f)$ en $Z^2$ se dicen adyacentes si la unión de las dos zonas planas $F_x(f) \cup F_y(f)$ es una componente conexa.

En la figura 1(d) se ilustra la noción de zonas planas adyacentes. La unión de las dos zonas planas adyacentes $F_x(f)$, $F_y(f)$ forma una componente conexa. De esta forma, el conjunto de zonas planas adyacentes a una zona plana se define a continuación.

**Definición 3**. Sea x un punto de $Z^2$, el conjunto de zonas planas $A_x$ adyacentes a $F_x$ está dado por: $A_x = \{ F_{x'} : x' \in Z^2, F_x \cup F_{x'} \text{ es conexa} \}$.

Finalmente, dado que trabajaremos con operadores conexos, utilizaremos la siguiente definición.

**Definición 4.** Un operador $\psi$ es conexo si y sólo si extiende las zonas planas de la imagen de entrada.




(a) (b) (c) (d)

**Figura 1.** (a) Imagen f con 5 zonas planas, (b) Zona plana en el punto x, $F_x(f)$, (c) Zona plana en el punto y, $F_y(f)$, (d) zonas planas adyacentes.

## 2.2. Transformaciones Morfológicas Sobre Partición

Los operadores morfológicos descritos en esta sección van ha operar sobre el par $( f , P_f )$. Se define entonces el elemento $(f , P_f )(x)$ como el valor de nivel de gris de la componente conexa $F_x(f)$. La dilatación y erosión morfológicas aplicadas sobre las zonas planas están dadas por:

$$\delta((f , P_f))(x) = \max\{( f , P_f )(y), F_y \in A_x \cup \{F_x\} \} \quad (1)$$
$$\varepsilon((f , P_f))(x) = \min\{( f , P_f )(y), F_y \in A_x \cup \{F_x\} \}$$

donde max y min representan los valores máximo y mínimo. De este modo, el valor de la dilatación o de la erosión en la zona plana $F_x$ estará dado por el máximo o el mínimo nivel de gris, respectivamente, de los niveles de gris de las zonas planas adyacentes $A_x$ y $F_x$. Puesto que la noción de elemento estructurante no existe en los operadores morfológicos sobre particiones, este elemento es eliminado de las definiciones de la dilatación y de la erosión. La dilatación y la erosión de tamaño $\lambda$ se obtienen iterando $\lambda$ veces la dilatación y la erosión elemental:

$$\delta_\lambda((f , P_f))(x) = \underbrace{\delta\delta...\delta}_{\lambda \text{ veces}}((f , P_f))(x) \quad y \quad \varepsilon_\lambda((f , P_f))(x) = \underbrace{\varepsilon\,\varepsilon\,...\varepsilon}_{\lambda \text{ veces}}((f , P_f))(x) \quad (2)$$

Por medio de la erosión y de la dilatación sobre zona plana, el gradiente externo y el gradiente interno pueden ser expresados, respectivamente, por:

$$\text{grade}((f , P_f))(x) = \delta((f , P_f))(x) - ( f , P_f )(x) \quad (3)$$
$$\text{gradi}((f , P_f))(x) = ( f , P_f )(x) - \varepsilon((f , P_f))(x)$$

Igualmente, utilizando la dilatación y la erosión se define la apertura y la cerradura sobre la partición inducida por f como:

$$\gamma_\lambda((f , P_f))(x) = \delta_\lambda((\varepsilon_\lambda( f , P_f ), P_f ))(x)$$
$$y \quad \varphi_\lambda((f , P_f))(x) = \varepsilon_\lambda((\delta_\lambda( f , P_f ), P_f ))(x) \quad (4)$$

## 2.3. Breve Recapitulación de los Filtros Morfológicos por Pendiente (FMP)

Los FMP son filtros de contraste morfológico no crecientes. Se construyen con transformaciones morfológicas básicas (erosión y dilatación) y el gradiente morfológico. Las versiones de gradiente empleadas en estos filtros son el gradiente externo y el gradiente interno (ver ecuación 3). Se iteran en base de un criterio de pendiente ($\phi$) para modificar la imagen y obtener un buen contraste. La definición de estos filtros sobre partición [7] se muestra a continuación:

$$\xi_{\phi}^{\varepsilon 1}(f)(x)=\begin{cases}\varepsilon\ ((f,P_f))(x) & \text{si } \text{gradi}((f,P_f))(x)\le\phi\\ ((f,P_f))(x) & \text{si } \text{gradi}((f,P_f))(x)>\phi\ ,\end{cases}$$

$$\xi_{\phi}^{\delta 1}(f)(x)=\begin{cases}\delta\ ((f,P_f))(x) & \text{si } \text{grade}\ ((f,P_f))(x)\le\phi\\ ((f,P_f))(x) & \text{si } \text{grade}((f,P_f))(x)>\phi\ ,\end{cases} \tag{5}$$

En el enésimo paso cuando la estabilidad es alcanzada ($n\to\infty$):

$$\overset{\varepsilon}{\xi_{\phi}}^{n}((f,P_f))(x)=\overset{\varepsilon}{\xi_{\phi}}^{1}(\overset{\varepsilon}{\xi_{\phi}}^{n-1}((f,P_f))(x)) \quad \text{y} \quad \overset{\delta}{\xi_{\phi}}^{n}((f,P_f))(x))=\overset{\delta}{\xi_{\phi}}^{1}(\overset{\delta}{\xi_{\phi}}^{n-1}((f,P_f))(x))$$

Las primitivas de estos filtros son la erosión y la función original (ó dilatación y función original). El parámetro $\phi$ hace que estos filtros trabajen como mapeos de contraste [1][11]; sin embargo existen algunas diferencias a considerar; por ejemplo, la idea de los FMP no es comparar una función con diferentes patrones, mas bien, atenuar o retener zonas de acuerdo a la información proporcionada por el gradiente. En la Figura 2 se ilustra con imágenes de salida el comportamiento de los FMP a nivel píxel y sobre partición.



(a) (b) (c)

**Figura 2.** (a) Imagen original, (b) FMP a nivel píxel con $\phi = 10$, (c) FMP a nivel partición con $\phi = 10$.

Teniendo en consideración la forma de actuar de estos filtros, en la siguiente sección se presenta la propuesta de este trabajo, la cual consiste en ponderar al gradiente morfológico para tener un control sobre el y poder contrastar en las imágenes de salida.

# 3. Ponderación del Criterio de Gradiente en los FMP Sobre Partición

## 3.1. Definición de los FMP Ponderados

La idea de ponderar el gradiente en los FMP surge en [7]; sin embargo esta propuesta aun no ha sido estudiada adecuadamente. En este artículo se proponen los FMP con gradiente ponderado. La idea consiste en cambiar el criterio de gradiente con la finalidad de que el criterio de proximidad permita detectar y contrastar zonas con determinados niveles de intensidad. De esta manera, la modificación del criterio de gradiente en los FMP surge como una alternativa importante para controlar el contraste en las imágenes de salida. En la siguiente expresión se definen los FMP ponderados por una función $\chi(t)$ los cuales se definen de manera general como filtros morfológicos por pendiente ponderados (FMPP):

$$\xi_{\phi}^{\varepsilon 1}(f)(x) = \begin{cases} \varepsilon\ ((f, P_f))(x) & \text{si } \chi(t)\text{gradi}((f, P_f))(x) \le \phi \\ ((f, P_f))(x) & \text{si } \chi(t)\text{gradi}((f, P_f))(x) > \phi\ , \end{cases}$$

$$\xi_{\phi}^{\delta 1}(f)(x) = \begin{cases} \delta\ ((f, P_f))(x) & \text{si } \chi(t^c)\,\text{grade}((f, P_f))(x) \le \phi \\ ((f, P_f))(x) & \text{si } \chi(t^c)\text{grade}((f, P_f))(x) > \phi\ , \end{cases} \tag{6}$$

Donde $t$ toma los valores de intensidad de las diferentes zonas planas dentro de la imagen, esto es $t = (f, P_f)(x)$ y $t^c = (f^c, P_f)(x)$. En el enésimo paso cuando la estabilidad es alcanzada ($n \to \infty$):

$$\xi_{\phi}^{\varepsilon n}((f, P_f))(x) = \xi_{\phi}^{\varepsilon 1}(\xi_{\phi}^{\varepsilon n-1}((f, P_f))(x)) \quad \text{y} \quad \xi_{\phi}^{\delta n}((f, P_f))(x) = \xi_{\phi}^{\delta 1}(\xi_{\phi}^{\delta n-1}((f, P_f))(x))$$

Entonces la función $\chi(t)$ multiplica al gradiente y por lo tanto este efecto se notará en el resultado final. La función ponderadora $\chi(t)$ puede ser modificada si se considera un parámetro de corrimiento $\varsigma(\varsigma \in N)$; es decir

$\chi(t - \varsigma)$. El parámetro de corrimiento origina que la función $\chi(t)$ sea desplazada respecto al origen. Por lo tanto se mantiene la forma original de la función $\chi(t)$; pero el criterio de gradiente será ponderado de diferente forma al ser multiplicado por esta nueva función. Una vez definido el parámetro de corrimiento $\varsigma(\varsigma \in N)$ se introducen los FMP ponderados con corrimiento(FMPPC). Los filtros FMPP con corrimiento cumplen lo siguiente: i) el criterio de gradiente trabaja si $(t-\varsigma) \geq 0$ y ii) para cada valor de corrimiento, el gradiente es ponderado con funciones diferentes.
La expresión matemática de los filtros morfológicos por pendiente ponderados con corrimiento es la siguiente:

$$\xi_{\phi}^{\varepsilon 1}(f)(x) = \begin{cases} \varepsilon\ ((f,P_f))(x) & \text{si } \chi(t-\varsigma)\text{gradi}((f,P_f))(x) \leq \phi \\ ((f,P_f))(x) & \text{si } \chi(t-\varsigma)\text{gradi}((f,P_f))(x) > \phi\ , \end{cases} \tag{7}$$

$$\xi_{\phi}^{\delta 1}(f)(x) = \begin{cases} \delta\ ((f,P_f))(x) & \text{si } \chi(t^c-\varsigma)\text{grade}((f,P_f))(x) \leq \phi \\ ((f,P_f))(x) & \text{si } \chi(t^c-\varsigma)\text{grade}((f,P_f))(x) > \phi\ , \end{cases}$$

En el enésimo paso cuando la estabilidad es alcanzada ($n \to \infty$):

$$\overset{\varepsilon}{\xi_{\phi}}^{n}((f,P_f))(x) = \overset{\varepsilon}{\xi_{\phi}}^{1}(\overset{\varepsilon}{\xi_{\phi}}^{n-1}((f,P_f))(x)) \quad \text{y} \quad \overset{\delta}{\xi_{\phi}}^{n}((f,P_f))(x) = \overset{\delta}{\xi_{\phi}}^{1}(\overset{\delta}{\xi_{\phi}}^{n-1}((f,P_f))(x))$$

Cuando $\varsigma = 0$, los FMP ponderados por corrimiento se reducen a la forma de los FMP ponderados por la función $\chi(t)$.
En la siguiente sección se presentan algunos ejemplos de funciones para la ponderación del criterio de gradiente. Primero se propone la modificación del criterio de gradiente mediante una ley física de contraste conocida como ley de Weber[8]. Después se proponen algunas funciones que permitan el realce del contraste en obscuro y en claro para ciertas regiones con determinados niveles de intensidad.

## 3.2. Ejemplos de Funciones para la Ponderación del Gradiente en los FMP

La primera función propuesta consiste en obtener una ponderación de los FMP usando una ley física conocida como ley de Weber. Esta ley física permite utilizar un modelo logarítmico [8] mediante el cual se mejora el contraste en una imagen al cambiar la luminancia del objeto. El modelo

gráfico de contraste logarítmico se presenta en la Figura 3. La expresión matemática de esta ley logarítmica es la siguiente:

$$C = a_1 + a_2 \log f \qquad \text{donde } a_1, a_2 \text{ son constantes}$$

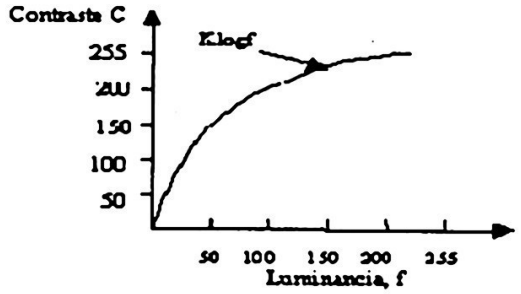



**Figura 3.** Modelo de contraste logarítmico.

La ley logarítmica puede escribirse como:

$$C = K\log_{10} f \quad 1 \leq f \leq m \quad \text{si } a_1 = 0 \text{ y } a_2 = K \tag{8}$$

Este concepto físico puede emplearse para mejorar el contraste en imágenes digitales. Considerando el modelo de contraste anterior y los FMP descritos en la sección 3.1, las siguientes funciones $\chi(t)$ y $\chi(t-\varsigma)$ se proponen como ponderadores. De esta manera se definen los FMPP por logaritmo y por corrimiento de logaritmo respectivamente:

$$\chi(t) = K\log_{10}(t) \quad \text{y} \quad \chi(t-\varsigma) = K\log_{10}(t-\varsigma) \quad \text{donde } t = (f, P_f)(x) \tag{9}$$

El valor de la constante K se calcula tomando en cuenta las siguientes consideraciones: si el contraste inicial es 0 entonces: $C = a_2 \log\chi$ con $a_1 = 0$. De esta manera, si C = 255 y suponiendo que el valor más grande tomado por el gradiente y la imagen original es 255, tenemos que:

$$255 = 255 \text{ X } K\log_{10}255 \qquad \text{y} \qquad K = 1/(\log_{10}255) \approx 0.41,$$

El corrimiento del logaritmo se ilustra en la Figura 4 donde el parámetro de corrimiento $\varsigma \in [1,255]$.

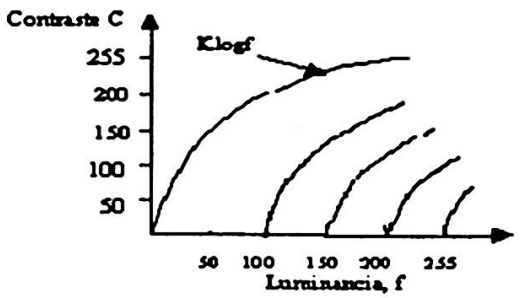



**Figura 4.** La aplicación del parámetro de corrimiento a la función logaritmo, permite el manejo de diferentes funciones para ponderar al gradiente.

Cuando el parámetro de corrimiento cambia, el gradiente es ponderado de diferente manera y de esta forma las primitivas de los FMPP por logarítmo actuarán de otro modo sobre las regiones de la imagen de entrada. Algunos ejemplos se muestran en la Figura 5. En las figuras de la 5(b)-5(c) se presentan dos imágenes de salida después de iterar los FMP ponderados por corrimiento de logaritmo. Además puede compararse el desempeño a nivel partición y a nivel pixel. La imagen de salida correspondiente a la Figura 5(d) muestran el resultado de iterar los FMP ponderados por corrimiento de logaritmo a nivel pixel. El incremento en el parámetro $\varsigma$ origina que regiones de nivel de intensidad mas claro sean iteradas, dando origen a que nuevas regiones sean ponderadas como zonas de alto contraste.



(a) (b) (c) (d)

**Figura 5.** (a) Imagen original, (b) Imágenes de salida después de iterar los FMPP por la ley logarítmica con corrimiento sobre partición, $\varsigma = 50, 80$ con $\phi=5$, (d) Imagen de salida después de iterar los FMPP por la ley logarítmica con corrimiento a nivel pixel, con $\varsigma = 80$ y $\phi=5$.

A continuación se propone otra función, la cual pondera al gradiente con la finalidad de contrastar regiones claras y obscuras.

$$f(x) = \sqrt{A - Bx} \tag{10}$$

La gráfica de esta fórmula se muestra en la Figura 6. Considerando la ecuación 11, se definen los FMPP por SQRT con y sin corrimiento respectivamente; las ecuaciones correspondientes son las siguientes:

$$\chi(t) = \sqrt{1 - \frac{t}{255}} \tag{11}$$

$$\chi(t - \varsigma) = \sqrt{1 - \frac{(t-\varsigma)}{255}} \qquad \text{con } t=(f, P_f)(x)$$

Con este criterio de gradiente, las zonas claras y obscuras serán ponderadas como zonas de alto contraste y por tanto, serán mantenidas cuando los FMPP por SQRT sean iterados. Algunos ejemplos se muestran en las imágenes de la Figura 7.

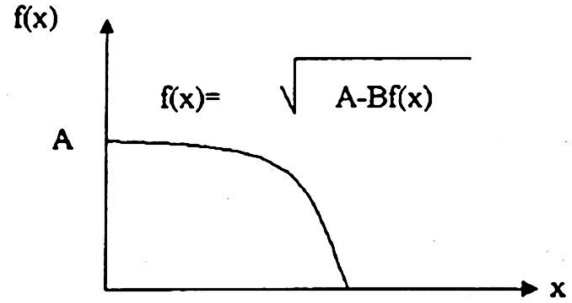



**Figura 6**. Función ponderadora para contrastar regiones claras y obscuras.



(a) (b) (c) (d)

**Figura 7**. (a) Imagen original, (b),(c) Imágenes de salida después de iterar FMPP por SQRT con corrimiento sobre partición, $\varsigma$ = 100,140 con parámetro $\phi$= 10, (d) Imagen de salida después de iterar FMPP por SQRT con corrimiento a nivel pixel, $\varsigma$ = 140 con parámetro $\phi$ =10.

En las Figuras 7(b)-(c) han sido tratadas al mismo tiempo regiones claras y obscuras a nivel partición. Al comparar la Fig. 7(c) con la Fig. 7(d), se observa un mejor comportamiento cuando las imágenes son tratadas a nivel partición. De la ecuación (11) nótese que la función ponderadora toma el valor de cero cuando *t* toma el máximo valor de intensidad dentro de la imagen. Lo interesante de esta transformación es el hecho de actuar sobre regiones claras y obscuras al mismo tiempo.

Hasta ahora no se ha propuesto ninguna función para contrastar regiones claras. A continuación se da una función como ejemplo. La formulación matemática de las funciones $\chi(t)$ y $\chi(t-\varsigma)$ se presenta en la ecuación 12. Estas funciones permiten definir los FMPP por función absoluto con y sin corrimiento.

$$\chi(t) = abs(1 - \log_{10}(t))\log_{10}(t) \tag{12}$$

$$\chi(t - \varsigma) = abs(1 - \log_{10}(t-\varsigma))\log_{10}(t-\varsigma) \qquad \text{con } t=(f, P_f)(x)$$

La gráfica de puntos que pondera al gradiente interno se muestra en la Figura 8. En la figura 9 se ilustra el comportamiento de este filtro.

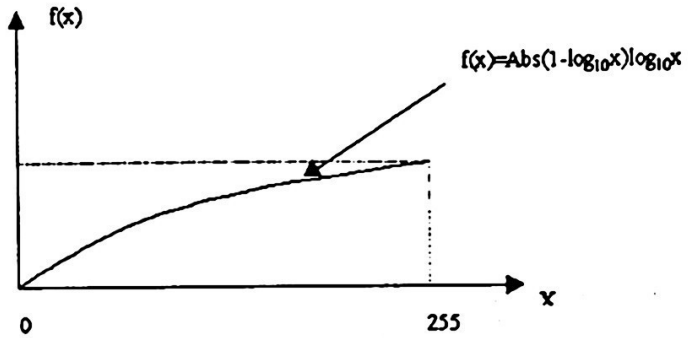



**Figure 8.** Gráfica propuesta para la ponderación del gradiente interno para contrastar regiones claras.

En la Figura 9 nótese el efecto de aplicar este filtro. Las regiones claras han sido retenidas y por lo tanto se modifica el contraste en las imágenes de salida. Cuando el parámetro de corrimiento aumenta, muchas otras regiones son ponderadas como regiones claras.




**Figura 9.** (a) Imagen original, (b) Imagen de salida después de iterar los FMPP por función absoluto con corrimiento sobre partición, $\varsigma = 50$ con $\phi=10$, (f) Imagen de salida después de iterar FMPP por función absoluto con corrimiento a nivel pixel, $\varsigma = 50$ con $\phi=10$.

# 4. Conclusión

En este trabajo se muestra el interés de ponderar el gradiente en los Filtros morfológicos por pendiente con la finalidad de tener control en el contraste de las imágenes de salida. La ponderación se realiza al multiplicar el gradiente por una función sugerida y de esta forma modificar el criterio de gradiente. La modificación de este criterio, permite obtener resultados interesantes; tal es el caso de los FMPP por logaritmo. Sin embargo la ponderación del gradiente debe ser acorde con el problema a resolver. Por otra parte las transformaciones que se definen en este trabajo son conexas.

**Agradecimientos.** El autor Jorge D. Mendiola Santibañez agradece a CONACyT por la beca de doctorado otorgada. El autor Terol-Villalobos agradece a Diego R. y a Darío T.G. por la motivación en el desarrollo de

este trabajo. Parte de este trabajo ha sido realizado dentro de la red VII de Telemedicina e Informática Médica financiada por el CyTED (España).

## Referencias

[1] Meyer, F. and Serra, J.. "Activity Mappings" *Signal Processing,* 16, 1989, pp.303-317.

[2] Terol-Villalobos, I. R., "Nonincreasing filtres using morphological gradient criteria", *Optical Engineering,* 35, 1996, pp. 3172-3182.

[3] Terol-Villalobos, I. R. y Cruz-Mandujano, J.A. "Contrast enhancement and image segmentation using a class of morphological nonincreasing filters", *Journal of Electronic Imaging,* 7, 1998 ,641-654.

[4] Terol-Villalobos, I. R., "Toggle mappings and some related transfortions: A study of contrast enhancement", in *Mathematical Morphology and Its Applications to Image and Signal Processing,* H.J.A.M. Heijmans and J.B.T.M. Roerdink , Eds., Kluwer Academic Publishers, The Netherlands, 1998, pp. 11-18.

[5] Beucher S., "Segmentation d'Images et Morphologie Mathématique", PhD Thesis , Centre de Morphologie Mathématique. ENSMP, Fontanebleau, France (1990).

[6] Rivest J.F., Soille P., and Beucher S., "Morphological gradients", *J.Electron Eng.* 2(4), 326-336(1993).

[7] Terol-Villalobos, I.R., "Morphological Image Enhancement and Segmentation", en Advances in Imaging and Electron Physics, Editor Peter W. Hawkes, Vol. 118, capítulo 4, Academic Press, septiembre, 2001, pp. 207-273.

[8] Jain, A., K., *Funfamentals of Digital image Processing*, Vol. I, Prentice Hall Information International, USA, 1989.

[9] Mendiola-Santibañez J.D y Terol-Villalobos, I. R., "Morphological Contrast Enhancement using Connected Transformations", en *Proceedings of SPIE,* Vol. 4667, pp. 365-376 (2002).

[10] Serra, J., *Image Analysis and Mathematical Morphology,* J. Serra, Ed., Vol. II, Academic, New York, 1988.

[11] Serra, J., "Toggle Mappings" *Technical report N-18/88/MM,* Centre de Morphologie Mathematique, ENSMP, Fontainebleau, France, 1988.

# Reconocimiento de Símbolos Musicales usando una Combinación de Técnicas de PDI y RNA

Raúl Cruz Barbosa
Universidad Tecnológica de la Mixteca
Carretera a Acatlima Km. 2.5. C.P. 69000
Huajuapan de León, Oax. México
Tel:(953) 53 20214 y 20399 Ext. 200
e-mail: rcruz@nuyoo.utm.mx

**Resumen**

En este trabajo se presenta una metodología para reconocer diez símbolos musicales dentro del pentagrama, en clave de Sol y clave de Fa, en una partitura musical estándar. Se combinan varias técnicas como el Procesamiento Digital de Imágenes y Morfología Matemática (para el preprocesamiento de las imágenes), y Redes Neuronales Arti...ciales (para el reconocimiento de los símbolos). Se reconoce tanto el símbolo (duración) como la nota del símbolo (sonido). Para este último reconocimiento se propone un algoritmo. Se muestran algunos ejemplos de los resultados obtenidos.

Palabras Clave: Reconocimiento de Símbolos Musicales, Red Neuronal Arti...cial, Morfología Matemática, Procesamiento Digital de Imágenes, Análisis de Documentos.

## 1 Introducción

En nuestro país, actualmente, existen pocas investigaciones acerca del reconocimiento de símbolos, si se habla de símbolos musicales este número se reduce. Una de las pocas investigaciones fue presentada en [1], la cual presenta algunas limitaciones, como la resolución de la imagen, excluye imágenes que contengan el respectivo pentagrama de una partitura, etc. Con respecto a la investigación del reconocimiento de símbolos musicales en el resto del mundo, existen algunos proyectos que han realizado tal objetivo, pero que requieren de equipo de cómputo especializado (máquinas paralelas) o emplean técnicas muy so...sticadas. Por ejemplo en [2], presentan un método para obtener un procedimiento de reconocimiento de notas musicales, mostrando una imagen original y una imagen objetivo, y para poder llegar a la imagen objetivo utilizan procedimientos de morfología matemática (operaciones de dilatación y erosión sucesivas).

Para decidir que procedimientos de morfología utilizar, se emplean los algor mos genéticos, es decir, el procedimiento de búsqueda lo realizan los algoritm genéticos. Como el procedimiento de búsqueda consume mucho tiempo, utiliz algoritmos genéticos paralelos. La investigación antes citada, sólo reconoce gunos símbolos básicos.

Otra investigación realizada en [3], presenta la solución del problema usan Razonamiento Probabilístico, donde se usa un modelo descriptivo de la de documento, para encontrar la interpretación más probable del documento imagen escaneado (partitura musical). También utiliza conceptos de Gram cas libres de contexto, así como de Modelos de Markov ocultos. Este artícu muestra sólo cierta parte del proyecto, ya que al momento de redactarlo había terminado.

En el presente trabajo se propone una metodología (combinación de técn cas) a seguir para reconocer diez símbolos musicales, 5 ...guras de nota: redond blanca, negra, corchea y semicorchea (unidad, mitad, cuarto, octavo y diec seisavo, respectivamente), y sus respectivos silencios, en clave de Sol y clave Fa. Además de reconocer diez ...guras (duración) de nota, se reconoce tambié el nombre (sonido: do, re, mi, fa, sol, la, si) de la nota.

En la sección 2 se presenta la metodología a seguir para reconocer los símb los musicales y los sonidos de estos. En la sección 3 se muestran los resultad obtenidos, y por último en la sección 4, las conclusiones y trabajo a futuro.

# 2 Metodología

La metodología propuesta en este trabajo lleva a cabo las siguientes etapa preprocesamiento, entrenamiento de la red neuronal, reconocimiento de símbol musicales y reconocimiento del sonido de los símbolos.

## 2.1 Preprocesamiento

El objetivo de la etapa de preprocesamiento es tener por separado (aislado) cada símbolo que se encuentre en un pentagrama. La etapa de preprocesamiento toma como entrada una partitura musical pianística estándar (hecha con editor de notas musicales Cakewalk) y realiza sobre esta las siguientes subetapas: digitalización, binarización, erosión, delimitación de los símbolos y centrado los símbolos. La digitalización consiste en escanear la imagen en tonos de La binarización de una imagen $f(x;y)$ se de...ne como en [4]

$$g(x;y) = \begin{matrix} ½ \\ \\ \end{matrix} \begin{matrix} 1; & si & f(x;y) > T \\ 0; & si & f(x;y) \ 6 \ T \end{matrix}$$

donde T se conoce como umbral, y es obtenido a partir del análisis del tograma correspondiente de la imagen digitalizada. Uno de los objetivos para

poder tener por separado a cada símbolo, es la eliminación de las líneas del pentagrama, para esto se utilizó la operación morfológica de erosión. La erosión binaria es la transformación morfológica, la cual combina dos conjuntos utilizando la substracción de vectores de elementos de conjunto. Si A y B son conjuntos en un N-espacio Euclideano, entonces la erosión de A por B es el conjunto de todos los elementos x, para los cuales x + b²A para todo b²B [5], [6]. La erosión de A por B es denotada por A Ä B y se de...ne por:

$$A \ddot{A} B = \overset{\bullet}{x^2 E^N} jx + b^2 A \text{ para todo } b^2 \overset{a}{B} : \qquad (2)$$

El conjunto A es la imagen binarizada y el conjunto B es un parámetro de forma (llamado elemento estructural) que sirve para erosionar al conjunto A. Partiendo de un análisis de varias imágenes se llegó a la conclusión de que los elementos estructurales que eliminan las líneas del pentagrama son una matriz de 3x1 y otra de 2x1, y dependiendo del grosor de las líneas se aplica una u otra. Después de aplicar la operación de erosión a la imagen de la partitura musical, se procede a delimitar cada símbolo de la partitura. La idea básica del algoritmo propuesto consiste en buscar en el rango de un pentagrama, de arriba hacia abajo y de izquierda a derecha, un pixel, y después de encontrarlo, se investiga el ancho de ese objeto, en el caso de que el ancho sea mayor que 4 pixels (esto signi...ca que es un símbolo musical y no una línea divisoria), entonces se busca la altura del símbolo y se guarda éste en una matriz. Y así sucesivamente, seguimos buscando símbolos en el pentagrama actual, hasta que terminamos de revisar todas las columnas de ese pentagrama. Una vez delimitados los símbolos se observa que las matrices donde se almacenan son de diferentes dimensiones, lo cual no es conveniente para la extracción de características, ni es congruente con las entradas a la red neuronal. Por lo anterior, el siguiente preprocesamiento que recibe cada uno de los símbolos delimitados es el centrado de estos en una imagen de 30x50 pixels. Se tomó la resolución de 30x50 por dos razones: primero porque el editor de notas musicales permite una impresión estándar, y segundo, porque de acuerdo a observaciones ninguno de los símbolos delimitados excede estos rangos. La ...gura 1 muestra el preprocesamiento de una partitura truncada.

## 2.2 Entrenamiento de la Red Neuronal

La etapa de entrenamiento de la red neuronal es la más importante en el proyecto, ya que de ésta dependen las siguientes etapas. La red que se utilizó es un Perceptrón multinivel, y se entrenó con el algoritmo de aprendizaje Backpropagation incluyendo el término momentum (veáse [7], [8] o [9]). Antes de entrenar la red neuronal se recolectaron 126 muestras de los símbolos a reconocer, los cuáles forman nuestro conjunto de entrenamiento. Dichos símbolos se tomaron de una partitura musical prede...nida y se les aplicó la etapa de preprocesamiento mencionada arriba. Cabe recordar que cada muestra es almacenada en una imagen de 30x50 pixels, por lo que 1500 pixels describen a

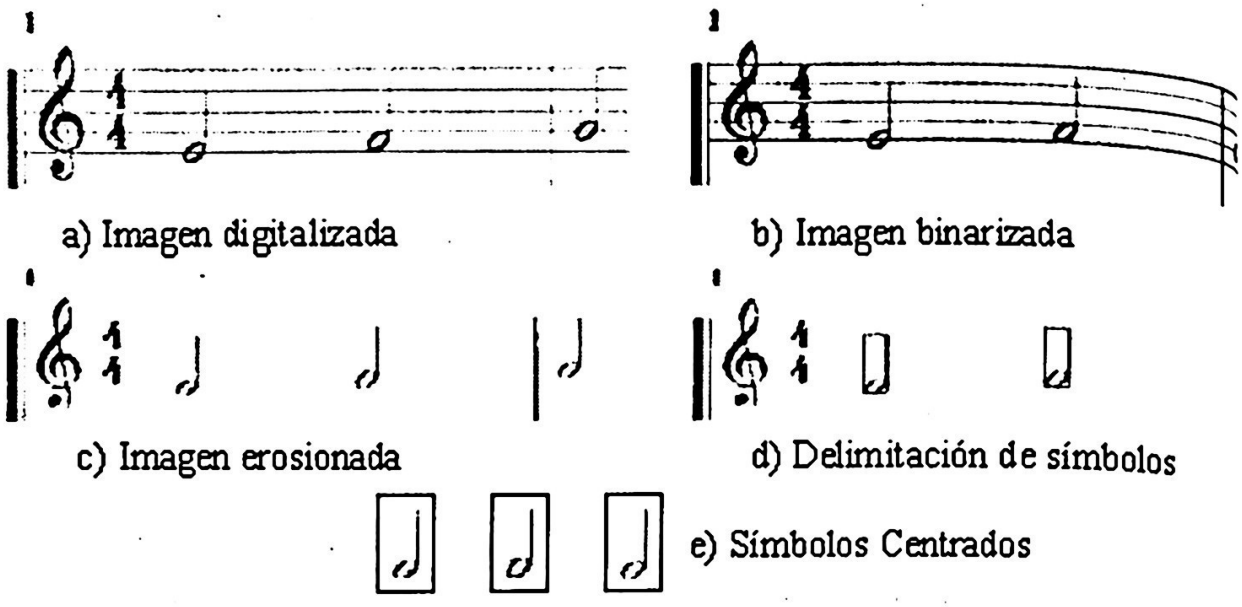



Figura 1: Preprocesamiento de una partitura pequeña.

cada una, y tomando en cuenta que se tienen 126 muestras, se decidió entonces representar a cada muestra con un vector característico de menos entradas. Las características o rasgos que se extraigan de cada imagen servirán como entradas a la red neuronal, la cual aprenderá con éstas. Para el entrenamiento de la red se proponen 3 conjuntos de entrenamiento. El primer conjunto se forma obteniendo los histogramas horizontales y verticales de las imágenes centradas. El histograma horizontal de una imagen binarizada, es la suma de los pixels marcados con 1 (correspondientes al objeto) en cada renglón. La cantidad anterior se divide entre el número de columnas del renglón. De manera análoga se de...ne el histograma vertical, es decir, es la suma de los pixels marcados con 1 en cada columna dividido por el número de renglones de la columna [10], [11]. La ...gura 2 muestra la obtención de los histogramas horizontales y verticales de una imagen. Como las imágenes centradas tienen una resolución de 30x50 pixels, entonces cada imagen se representará con un vector de 80 entradas (la concatenación del histograma horizontal y el histograma vertical). El segundo conjunto de entrenamiento se forma obteniendo los momentos invariantes (desarrollados por Hu, veáse [4]) de cada imagen centrada, es decir, cada imagen se representará con un vector de 7 entradas.

Por último, el tercer conjunto se forma con una combinación de los anteriores, es decir, de cada imagen se obtienen los histogramas horizontales y verticales, y además los 7 momentos invariantes, por lo tanto, cada imagen se representará con un vector de 87 entradas. Ya que se tienen los tres conjuntos de entrenamiento para la red, lo que sigue es proponer la arquitectura del perceptrón multinivel, es decir, su topología. Se propusieron 6 topologías de red diferentes. La tabla 1 muestra las topologías propuestas.

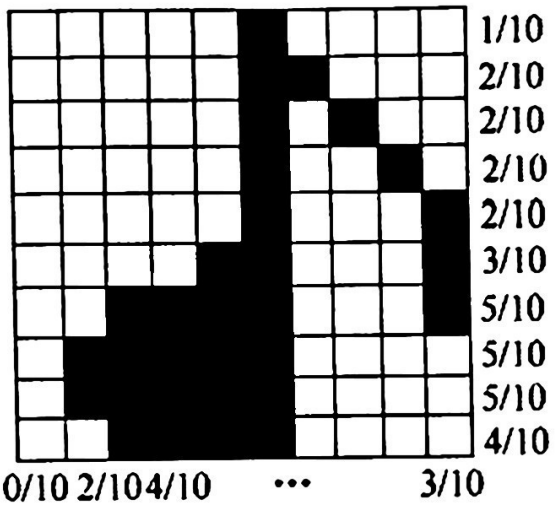



Figura 2: Obtención de histogramas horizontal y vertical de una imagen.

| Topología | Número de neuronas en cada capa | | | | | Capas ocultas |
|---|---|---|---|---|---|---|
| 1 | 80 | 20 | 15 | 9 | | 2 |
| 2 | 80 | 25 | 20 | 15 | 9 | 3 |
| 3 | 7 | 15 | 9 | 9 | | 2 |
| 4 | 7 | 20 | 18 | 15 | 9 | 3 |
| 5 | 87 | 25 | 15 | 9 | | 2 |
| 6 | 87 | 30 | 15 | 9 | 9 | 3 |

Tabla 1: Topologías de red para entrenamiento.

De la tabla 1 se observa que el número de neuronas en la capa de entrada, de cada topología, depende del conjunto de entrenamiento que se use. También, se observa que el número de neuronas en la capa de salida siempre es de 9, debido a que los silencios de redonda y blanca, no tienen diferencias signi...cativas, y al momento de extraer sus características, éstas serían semejantes, provocando un error al momento de reconocerlas. Por lo tanto, en la etapa de reconocimiento de los símbolos sólo se indicará que es un "silencio de redonda o blanca", dejando como tarea el nombre correcto a la etapa de reconocimiento del sonido de las notas, ya que el nombre correcto depende más de la posición (el silencio de redonda se coloca debajo de la cuarta línea, y el silencio de blanca se coloca sobre la tercera línea [12]), que de su forma. Ahora solo falta establecer los valores de los parámetros del algoritmo backpropagation (tasa de aprendizaje ´, constante de momentum ® y el error promedio). Los valores de ´ se establecieron a.0.75 y 0.25, y los de ® a 0.1 y 0.5. El error promedio se estableció a 0.0001 por considerarlo un buen margen de error. Debido a que se tienen 6 topologías diferentes, y 2 valores diferentes para ´ y ®, se realizaron 24 entrenamientos.

## 2.3 Reconocimiento de Símbolos Musicales

El procedimiento de reconocer símbolos es más sencillo, ya que utiliza las rutinas de la etapa de preprocesamiento y la primera fase del algoritmo backpropagation. Dicho procedimiento recibe como entrada la imagen de una partitura musical estándar en tonos de gris, y lleva a cabo lo siguiente:

- Seleccionar tipo de reconocimiento (con histogramas, momentos invariantes o combinación de los anteriores).
- Preprocesar la imagen (aplicar binarización, erosión, delimitación y centrado de los símbolos) de la partitura a reconocer.
- Obtener características (vector característico) de los símbolos a reconocer de acuerdo al tipo de reconocimiento elegido.
- Aplicar la primera fase del algoritmo backpropagation para cada vector característico obtenido.
- Guardar los resultados del reconocimiento, y coordenadas de los símbolos en un archivo.

## 2.4 Reconocimiento del sonido de las notas

El único preprocesamiento que se le hace a la imagen de la partitura original, antes de reconocer los sonidos, es la aplicación del algoritmo de binarización. El reconocimiento de los sonidos (nombres) de las notas, depende del reconocimiento de los símbolos musicales de la etapa anterior, debido a que hace uso de las posiciones (coordenadas) de los símbolos reconocidos. Entonces, es claro que el reconocimiento de las notas se basa en las posiciones de las líneas del pentagrama. Como se mencionó en la etapa de preprocesamiento, la imagen de la partitura es estándar (no presenta cambios de escala), por lo que existe una distancia determinada entre cada línea del pentagrama, así como entre cada pentagrama. Aprovechando estas características de la imagen de la partitura a reconocer, se presenta un algoritmo para reconocer los sonidos de las notas en la partitura. La idea del algoritmo es utilizar los resultados del reconocimiento de los símbolos y sus coordenadas, de la etapa anterior, y tomando de renglón en renglón (pentagrama), los símbolos reconocidos. Si el símbolo es menor a 6 (los silencios se etiquetaron de 6 a 8, ya que los silencios de redonda y blanca se fusionaron, formando la clase 5), signi...ca que no es un silencio, y se procede a buscar su posición en el pentagrama de acuerdo a las coordenadas del símbolo. Se busca un pixel partiendo de la mitad del ancho del símbolo, y de arriba hacia abajo. Una vez encontrado se busca continuidad de pixels en el eje y (para eliminar la posibilidad de que sea una línea del pentagrama), y si esta es mayor a c (constante de...nida), entonces se encuentra su posición y se le asigna una nota (número entero), y se guarda en el archivo de reconocimiento ...nal el símbolo y la nota correspondiente. En caso de que el símbolo sea mayor o igual a 6, signi...ca que es un silencio, por lo tanto, solo se guarda directamente en el archivo de reconocimiento ...nal

# 3 Resultados y Discusión

Primero se presentan los resultados obtenidos al reconocer imágenes con símbolos ocupados en el entrenamiento de las redes neuronales propuestas, después se presentan los resultados que se obtuvieron al reconocer una partitura musical cualquiera, es decir, con símbolos que la red neuronal no conoce. Al primer tipo de resultados mencionado, le llamaremos Reconocimiento con Imágenes de Prueba, y al segundo tipo como Reconocimiento Final.

- Reconocimiento con Imágenes de Prueba. Se tomaron 140 símbolos al azar, de las 126 muestras que se ocuparon en el entrenamiento de las redes neuronales. En esta fase lo que interesa es conocer que topologías (de las 24 combinaciones) arrojan los mejores resultados, es decir, las topologías que proporcionen el mayor porcentaje de reconocimiento, para después utilizar sólo éstas en la fase de Reconocimiento Final. Las topologías que obtuvieron mejores resultados se presentan en la tabla 2.

| Topología | ´ | ® | % rec. |
|---|---|---|---|
| 1 | 0.25 | 0.1 | 99.28 |
| 5 | 0.75 | 0.1 | 99.28 |
| 5 | 0.75 | 0.5 | 100 |
| 5 | 0.25 | 0.1 | 99.28 |
| 5 | 0.25 | 0.5 | 100 |
| 6 | 0.75 | 0.1 | 100 |
| 6 | 0.25 | 0.1 | 100 |

Tabla 2: Topologías con mayores porcentajes de reconocimiento.

- Reconocimiento Final. Para el reconocimiento ...nal se utilizaron 9 partituras, las cuales contienen un total de 1007 símbolos a reconocer. Los resultados de los reconocimientos de los símbolos, así como de los sonidos de los símbolos se presentan en la tabla 3.

| Topología | ´ | ® | % rec. simb. | % rec. nota |
|---|---|---|---|---|
| 1 | 0.25 | 0.1 | 95.63 | 100 |
| 5 | 0.75 | 0.1 | 95.03 | 100 |
| 5 | 0.75 | 0.5 | 93.44 | 100 |
| 5 | 0.25 | 0.1 | 89.7 | 100 |
| 5 | 0.25 | 0.5 | 93.25 | 100 |
| 6 | 0.75 | 0.1 | 95.13 | 100 |
| 6 | 0.25 | 0.1 | 93.44 | 100 |

Tabla 3: Reconocimiento Final usando 9 partituras con 1007 símbolos.

De la tabla 3 se observa que el mayor porcentaje de reconocimiento (95.63) lo obtuvo la topología 1 con ´=0.25 y ®=0.1, con...rmando la teoría que

nos dice que se obtienen buenos resultados con valores pequeños para o también combinando valores altos para ´ y pequeños para ®. Con topología, valores de ´ y ®, y con las 1007 muestras de imágenes, el conocimiento real se encuentra entre 95.51% y 95.75%. Los reconocimientos reales se calcularon con un nivel de con...anza del 95% [13].

Una red neuronal además de clasi...car un patrón, proporciona también un nivel de con...anza del reconocimiento. Otro resultado que se observó al momento del reconocimiento, es que para la mayoría de las topologías 5 y 6 el nivel de con...anza del reconocimiento es alto, es decir, la neuronal construyó una función que discrimina bien (está segura de clasi...cación, aunque se equivoque), y en cambio con la topología 1, nivel de con...anza es muy variable, por lo cual conduce a reconocimientos frágiles, es decir, puede ser que el reconocimiento sea correcto, pero con nivel de con...anza bajo. Lo anterior se pudo originar por las características que se extraen de las imágenes (histogramas y momentos).

También se observa de la tabla 3, que el algoritmo para el reconocimiento de los sonidos de los símbolos funciona muy bien, obteniendo el 100 ciento de reconocimiento de los sonidos de las notas en todas las partituras. Cabe aclarar que este algoritmo se aplica sólo a los símbolos reconocidos correctamente por la red neuronal.

Las ...guras 3 y 4 muestran dos pequeñas partituras antes de aplicar algoritmos de reconocimiento y sus respectivos reconocimientos ...nales. puede observar de la ...gura 3 que el formato del archivo de reconocimiento ...nal consiste de: la primera línea que indica el número total de símbolos reconocidos en la partitura, y a continuación se van alternando las cadenas de "Sol" y "Fa", lo cual signi...ca que los símbolos que se presentan continuación de dicha cadena, se encuentran en esta clave. Después la cadena "Sol" ó "Fa", se muestran pares de números enteros separados por un espacio. El primer número entero indica el símbolo reconocido, y el segundo número indica el sonido (nombre) del símbolo, de acuerdo a la tabla 4. Debido a que los símbolos que representan a los silencios siempre tienen la misma posición en el pentagrama, se optó por asignarles en el segundo número entero (correspondiente al sonido del símbolo en archivo), el mismo número que representa al símbolo.

De la tabla 4 se observa, que los sonidos de las notas presentes corresponden solamente a la clave de Sol, y en la introducción se especi...ca también se reconocerán las notas en clave de Fa. La solución a esto es siguiente. De acuerdo con [12], la clave da su nombre a la nota escrita la línea donde está ella, y las demás notas, ascendiendo o descendiendo, van tomando el suyo en el orden conocido: Do, Re, Mi, Fa, Sol, La, Si. Ahora, si se observa un pentagrama, por ejemplo, el de la ...gura 3, clave de Sol está en la segunda línea y la clave de Fa en la cuarta línea. Por lo tanto, haciendo un análisis de las posiciones de las notas, en las claves, y con base en la tabla 4, para encontrar el correspondiente nombre

de nota en clave de Fa, sólo hay que reconocer la nota en clave de Sol y a este número sumarle dos.

| Símbolo | Rep. | Sonidos de Nota | Rep. |
|---|---|---|---|
| Blanca | 0 | Mi_menor | 11 |
| Negra | 1 | Fa_menor | 12 |
| Corchea | 2 | Sol_menor | 13 |
| Semicorchea | 3 | La_menor | 14 |
| Redonda | 4 | Si | 15 |
| Silencio de redonda | 5 | Do_mayor | 16 |
| Silencio de blanca | 6 | Re_mayor | 17 |
| Silencio de negra | 7 | Mi_mayor | 18 |
| Silencio de corchea | 8 | Fa_mayor | 19 |
| Silencion de semicorchea | 9 | No_existe | 20 |

Tabla 4: Representación de los símbolos y sonidos de notas.

Por lo anterior, a la tabla 4, sólo hay que agregarle dos sonidos de nota: sol_mayor (representado por el número 20) y la_mayor (representado por el número 21). Además el símbolo de nota no_existe se representa con el número 22.

De la ...gura 4, se puede observar que sólo hubo un error en el reconocimiento, el segundo símbolo, el cuál es una "corchea" y no una "negra". Lo anterior se podría mejorar (es decir, mejorar los porcentajes de reconocimiento) con varias alternativas. Una de ellas es incluir nuevas características que describan a los símbolos (por ejemplo, las ...rmas [14]), en los vectores característicos (patrones de entrada de la red neuronal); otra alternativa sería mejorar el preprocesamiento de los símbolos, de tal manera que conserven sus rasgos descriptivos. También se podría cambiar las con...guraciones de las topologías y utilizar una técnica que acelere la convergencia del algoritmo de aprendizaje Backpropagation, con el objetivo de que éste alcance un error promedio muy pequeño.

# 4 Conclusiones

La metodología propuesta es muy sencilla, donde destaca el uso de una red perceptrón multinivel entrenada con el algoritmo backpropagation incluyendo el término momentum, para el reconocimiento de los símbolos musicales. La generalización lograda por la red neuronal (de la topología 1) es aceptable, ya que con un nivel de con...anza del 95%, se logra un reconocimiento real entre 95.51% y 95.75%. Además, los reconocimientos con niveles de con...anza más altos los proporcionan las topologías 5 y 6, donde se utilizó una combinación de los histogramas vertical y horizontal, y los momentos, como entradas a la red neuronal. Esto nos sugiere que para entrenamientos futuros, es mejor entrenar con combinaciones de otras características de la imagen.

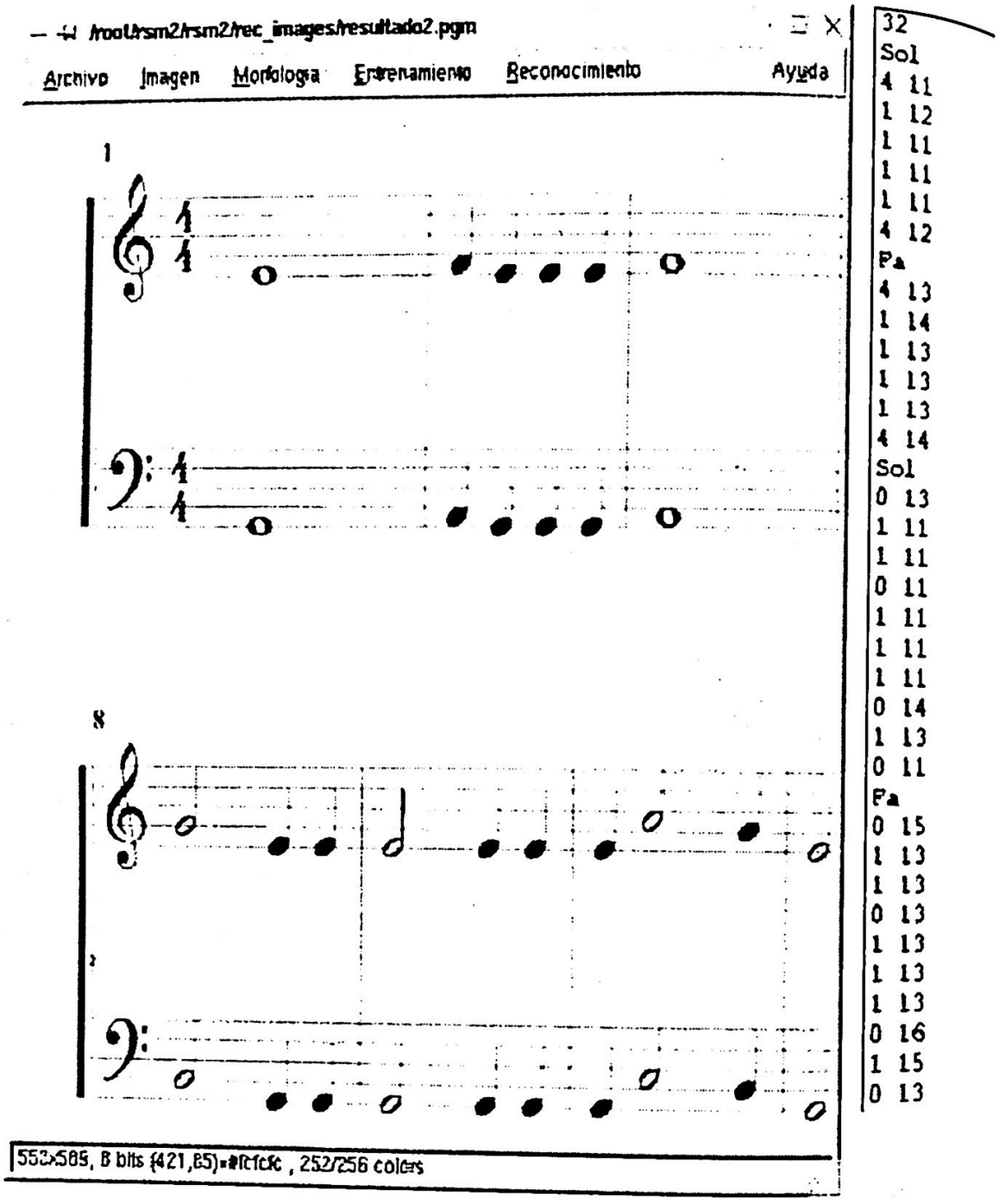



Figura 3: Ejemplo con reconocimiento de 100% de los símbolos. A la izquierda, imagen de partitura a reconocer; a la derecha, reconocimiento ...nal.

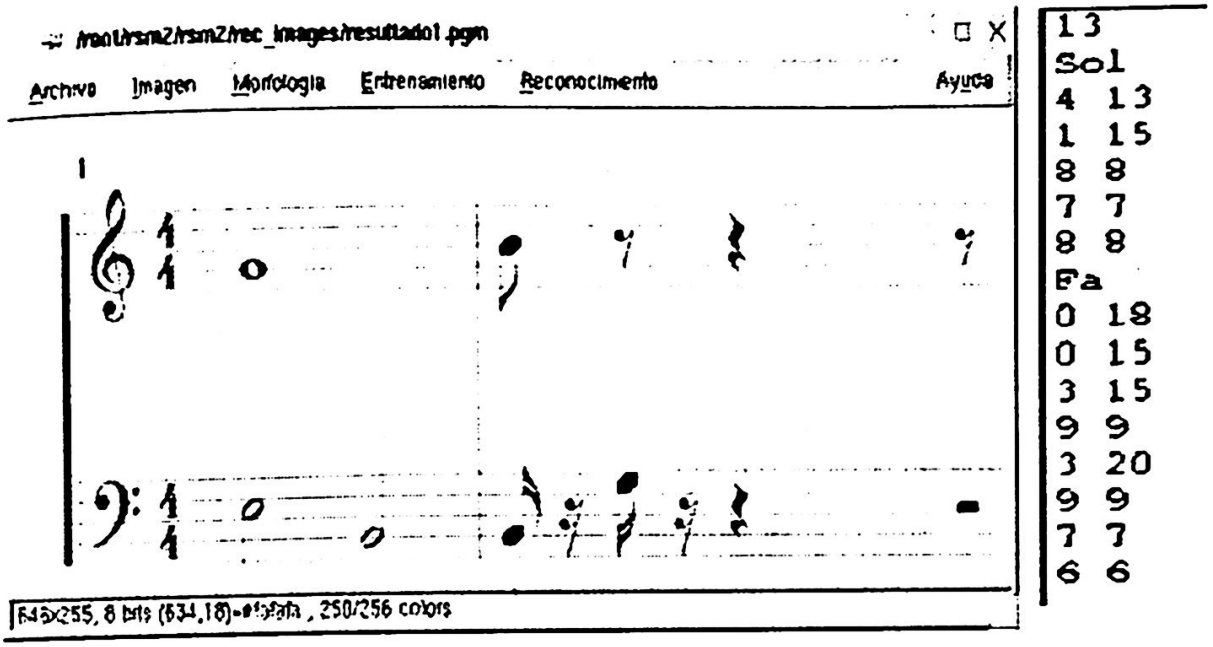



Figura 4: Ejemplo con reconocimiento del 92.3% de los símbolos.

La combinación de las técnicas: PDI, Morfología Matemática y Redes Neuronales, fué fundamental para poder construir el pequeño sistema híbrido de reconocimiento. Actualmente, esta es la tendencia (en reconocimiento de patrones), ya que las técnicas por separado, en algunas ocasiones, no llegan a ser su...cientes para poder resolver los problemas planteados.

Como trabajo a futuro se pretende extender el reconocimiento a notas manuscritas, usando la metodología planteada. Para esto, habría que obtener una base de datos de imágenes de símbolos manuscritos de una población grande, así como mejorar el porcentaje de reconocimiento, al incluir otras características los vectores de entrada de la red, las cuáles ayuden a distinguir mejor cada símbolo. También se debe de implementar una técnica de aceleración (mejor que la del término momentum) del algoritmo backpropagation, ya que esto nos conduce a alcanzar errores promedio muy pequeños, lo cual ayuda a mejorar el porcentaje de reconocimiento.

# Referencias

] Vargas Martínez, Héctor S.; "Visión Arti...cial del primer robot pianista mexicano Don Cuco el Guapo", Tesis de Licenciatura, BUAP, 1996.

[2] Yoda I., Yamamoto K., Yamada H.; "Automatic Construction of Recognition Procedures for Musical Notes by GA", Proceedings of DAS94, pp.203-209, IAPR Workshop on Document Analysis Systems, Germany, October 1994.

[3] Stückelberg M., Doermann D.; "On Musical Score Recognition using babilistic Reasoning", International Journal on Document Analysis Recognition, IJDAR'97, Springer 1997.

[4] Gonzalez, Rafael y Woods, Richard; "Tratamiento Digital de Imágenes Addison-Wesley, 1996.

[5] Haralick, Robert M., Shapiro G. Linda; "Computer and Robot Vision Addison-Wesley, 1987.

[6] Haralick, Robert M., Stenberg, Stanley R, y Zhuang, Xinhua; "Image Analysis using Mathematical Morphology", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAM-9, No. 4, Julio 1987.

[7] Haykin, Simon; "Neural Networks. A Comprehensive Foundation" Macmillan College Publishing Company, 1994.

[8] Rojas González, Raúl; "Neural Networks. A systematic introduction" Springer-Verlag, 1996.

[9] Fausset, Laurene; "Fundamentals of Neural Networks. Architectures, rithms, and applications", Prentice-Hall, 1994.

[10] Starner, Thad; "Notas del Handwritten Digit Recognition Project", gia Tech, 1999.

[11] Maldonado B. J., Cruz Barbosa R.,Vargas G. P.; "Aplicación del ceptrón Multinivel entrenado con el algoritmo Backpropagation y samiento de Imágenes para Reconocer Caracteres Manuscritos", Simposium Internacional de Computación CIC'98, 1998.

[12] Moncada García, Francisco; "Teoría de la Música", Ediciones Framong 1997.

[13] Spiegel, Murray R.; "Probabilidad y Estadística", McGraw-Hill, 1989.

[14] Sánchez Díaz, Guillermo, Díaz de León S., Juan Luis; "Método de para reconocimiento de símbolos musicales", Simposium Internacional Computación CIC'98, 1998.

# Normalized similarity measure of 2D objects and their morphing

Hermilo Sánchez*

Department of Computer Science
Institute of Research in Applied Mathematics and Systems
National Autonomous University of Mexico
Apdo. 20-726, México, D.F., 01000. Fax: 5622-3620
E-mail: herssan@uxmcc2.iimas.unam.mx

**Abstract**

The purpose of this paper is to show that in order to obtain a normalized similarity measure, the study of some properties like *compactness*, the *work* done in object transformations and *the number of pixels* to be moved help us to establish the way of normalize a similarity measure of an appropriate set of 2D objects, including irregular objects; at the same time, it is shown that pixel representation permits break the object to get the so called *morphing*. It is also shown that orientation (by means of principal axes) of objects to be compared is crucial in the performance of the work done in their transformation.

Key words: compactness; transforming; shapes; positive pixels, morphing.

## 1 Introduction

It has been written about comparing, registering and recognizing of 2D objects. Sometimes there are considered contour shapes and their geometrical

*Part of this work is a section of the Doctoral Dissertation presented to the Universidad Nacional Autónoma de México

properties [1], [2] and [3] to establish their similarity. Generally, in works are assumed that objects to recognize are invariant under affine formations like rotation, translation and scaling. Some authors employ tance transforms to compare objects. Arkin et. al [3] for example compar several polygons trough distance functions. One of the first formal works establish the relationship between shape and similarity degree, for 2D jects, was proposed by Bribiesca and Guzman [5]. They presented a resolution method through the so called *shape numbers*: greater resolution o 2D objects with the same shape numbers, more similar will be the compare objects.

One of the methods to represent 2D objects are through the so *picture elements*, or pixels for short. In this work, we are going to conside a 2D *object* as a binary spatial representation of a bi-dimensional scene, which every pixel takes the value of "0" or the value of "1"; as *irregula objects* we consider those objects that have no symmetry axes.

In 1997, Bribiesca and Wilson [6] intended, for the first time, to compar 2D objects through the movement of pixels, from the so called *positive* the *negative* sets of pixels, and this is what we will know as *transformatio of objects*. As *work done* in transforming objects, we consider the concep introduced by Bribiesca and Wilson [6] in the sense that while moving pixels there is a work done to translate every positive pixel to the negative Because the force involved in the translation is constant and equal to work done has a numerical value of distance. The grater distance in displacin one set into another, the more dissimilar are the objects considered.

Bribiesca and Wilson [6] did not normalize their similarity measure employed maximum correlation to orientate objects. In this paper propose the use of principal axes to orientate the objects, and provide garian algorithm to move every pixel to find the total smallest distance in transforming two objects. 2) Also we provide a method to normalize similarity measure proposed by Bribiesca and Wilson [6]; and 3) we provid a new method to make a *morphing*, through the movement of pixels, givin at the same time the degree of similarity measure.

# 2 Discrete compactness in 2D

Again, in 1997, Bribiesca [7] established a compactness measure of 2D objects composed of pixels. This measure related contact perimeter (total length

the sides that pixels get in touch) to contour perimeter. Given a 2D object composed of $n$ pixels, let $P_c$ be the contact perimeter. Discrete compactness, normalized in the interval [0,1], is given by the expression:

$$C_D = \frac{P_c - P_{cmin}}{P_{cmax} - P_{cmin}}, \tag{1}$$

where $P_{cmin} = n - 1$ and $P_{cmax}$ is defined by $P_{cmax} = 2(n - n^{1/2})$.

# 3 Relationship between discrete compactness, work done in transforming objects and number of positive pixels

Consider twelve different objects invariant under translation (their centroids coincide), scale (same number of pixels) and rotation (aligned through their principal axes) given in figure 1. All of them are composed of 674 pixels.

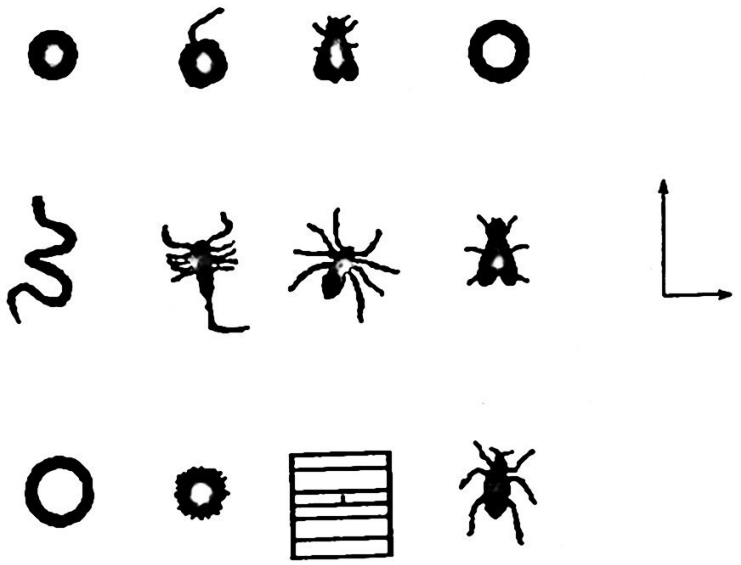


Figure 1: Twelve different 2D objects invariant under scale and rotation

To see the behaviour of compactness in terms of work done and positive pixels, let us choose the circle to be compared with the other eleven objects, because it has a high compactness (0.9954, calculated using the eq. 1) and

whatever perpendicular diameters can be used as principal axes. Table and figure 2 show the work done in transforming all the objects into circle, also contains the number of positive pixels and discrete compactness of each object. As we can see, the most similar object to the circle, is circle with noise (circur), following the circle with a hair (cirpel). The most different to the circle is the frame, following the snake. Note that the frame has the smallest compactness: 0.212. Despite it does not have the maximum number of pixels to move, the work increases to 12,903.10. But, for the ring2, whose compactness is much grater than the frame (0.854), hardly the work done in transforming it to the circle is increased: 5,541.25. Another example. Note the case of the scorpion and the ring. The first has 0.32 times smaller compactness than the second. Despite the scorpion has smaller number pixels to be moved than the second, the work done in transforming it into circle is grater. So, from this sample of objects, the least compact objects are less similar to the circle than those more compact, despite sometimes these have more pixels to be moved. It is also shown that when some pixels are moved (circle with a hair) its similitude does not change so much. Also, despite the noise, the similarity with the circur is so high, showing that measure is accurate respect to the noise.

# 4 How to normalize the similarity measure

## 4.1 The set of objects is complete

As we have said last section, the objects that have small compactness tend to be more rapidly different to the circle than those with higher compactness (of course, compactness is not the only determinant feature of objects computing the similarity, remember that the number of positive pixels plays an important roll too). What is the most compact object and what is least compact object of the whole universe of shapes? If it is considered set of $N$ objects composed of all combinations of $n$ pixels, it is not necessary to compute all shapes, and then the work done in transforming every pair of objects until find the two most dissimilar. Clearly, the complete set objects is so huge, in comparison with the 12 objects presented in this paper.

From the results of last section, it is deduced that a way to find the most dissimilar objects, from the whole universe of $N$ objects composed $n$ pixels, corresponds to transform the most compact object into the object

| OBJECTS | *Comm Pixels* | *Pixels (+)* | $C_D$ | $W_H$ |
|---|---|---|---|---|
| cirpel | 574 | 100 | 0,90358 | 1702 |
| bee | 516 | 158 | 0,876296 | 1785,16 |
| fly | 483 | 191 | 0,816914 | 2083,78 |
| beetle | 433 | 241 | 0,669259 | 2893,49 |
| spider | 392 | 282 | 0,539259 | 3359,12 |
| scorpion | 379 | 295 | 0,574568 | 4994,21 |
| ring | 291 | 383 | 0,893951 | 3531,62 |
| snake | 229 | 445 | 0,751111 | 6920,96 |
| frame | 85 | 589 | 0,211852 | 12903,1 |
| ring2 | 24 | 650 | 0,853827 | 5483,75 |
| circur | 622 | 52 | 0,873086 | 334,77 |

Table 1. Common pixels, pixels to move, discrete compactness and work done in transforming the eleven objects into the circle.

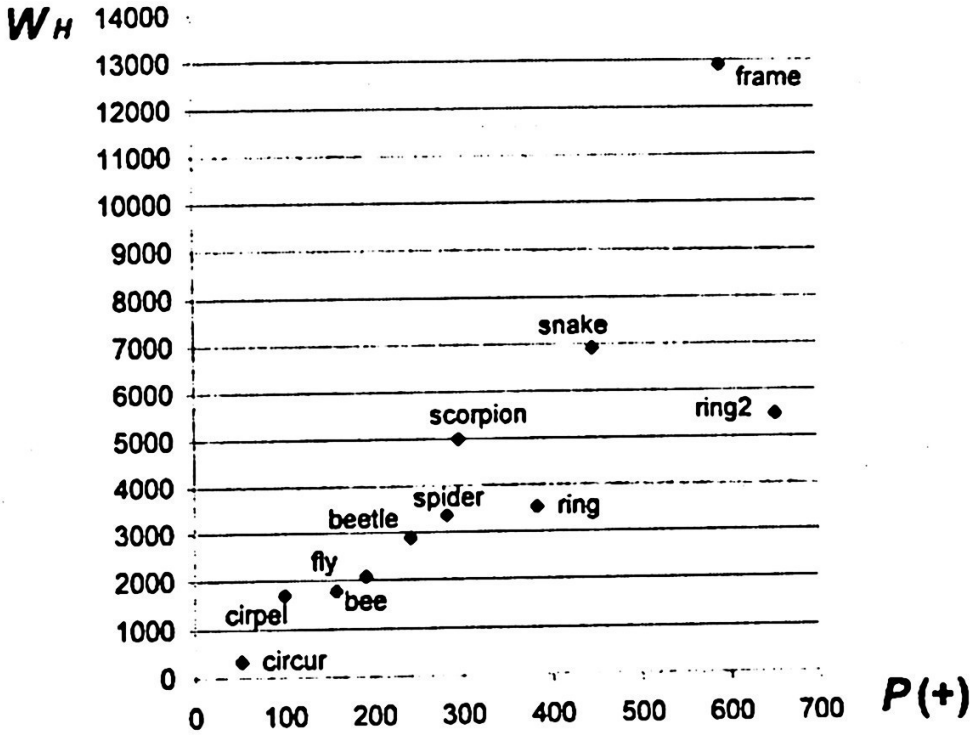



Figure 2. Work done to transform the objects as a function of positive pixels.

of smallest compactness. These objects correspond to a square of $n^{1/2}$ of side, that has the greatest compactness: 1, and a stick of length $n$ that has the smallest compactness: 0. To know what is the work in transforming the square into the stick composed of $n$ pixels, as big as we desire, we could obtain the work for small $n's$, and then to adjust a polynomial curve. Table shows the work done between the square and the stick, for five different values of $n$. Adjusting a second order polynomial extrapolation, for x = 674 pixels, the work done is approximately 454,276. What it means, is that if normalizing with this amount, the circle and the snake would be similar in 98.47% and circle and the frame in 97.16%.

| Pixels | P(+) | EMC | $W_H$ |
|---|---|---|---|
| 9 | 6 | 15,54 | 15,28 |
| 25 | 20 | 131,14 | 129,6 |
| 49 | 42 | 527,06 | 522,82 |
| 81 | 72 | 1479,6 | 1471,02 |
| 121 | 110 | 3257,46 | 3346,14 |

Table 2. Work done to transform a square into a stick of 676 *pixels* each one.

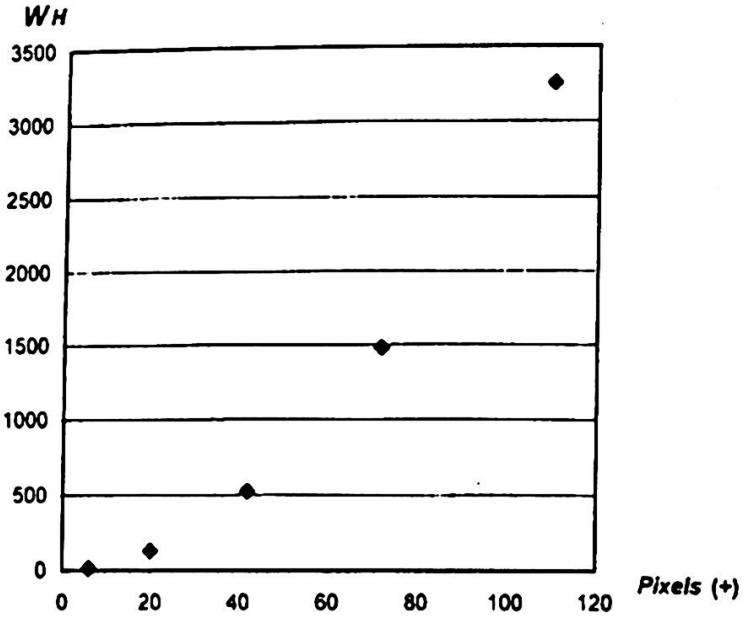



Figure 3. Work done with hungarian algorithm as a function of pixels to move, to transform a square into a stick at different resolutions.

Of course, those amounts do not correspond to the visual meaning. reason is that there are a lot of objects that have more capricious shapes that do not appear here, but there can be formed with the 674 pixels. it is more appropriate to compute the normalization in this way if having a so large representative sample of objects and with so different forms. normalize a sample like that given in section 3, is more convenient to consider another method that we propose in the next section.

## 4.2 The set of objects is not complete

Given a set of $N$ objects, suppose each one is composed of $n$ pixels. A method to normalize the similarity measure, is to look for, between the $N$ objects, what are the two most dissimilar objects. With this amount we have maximum work done in the transformation of this $N$ objects. Considering the 12 objects, it is observed that the two most different are the frame the circle, they correspond to the object with greatest compactness and object with the smallest compactness. Note that their compactness are near the extrems of the interval between 0 and 1. Considering the work done transforming the frame into de circle as a similarity measure of 0%, and

| Transformation | Common pixels | Pixels to move | Similarity with the circle(%) |
|---|---|---|---|
| cirpel | 574 | 100 | 86.81 |
| bee | 516 | 158 | 86.16 |
| fly | 483 | 191 | 83.85 |
| beetle | 433 | 241 | 77.58 |
| spider | 392 | 282 | 73.97 |
| scorpion | 379 | 295 | 61.29 |
| ring | 291 | 383 | 72.63 |
| snake | 229 | 445 | 46.36 |
| frame | 85 | 589 | 0 |
| ring2 | 24 | 650 | 57.5 |
| circur | 622 | 52 | 97.4 |

Table 3. Number of common pixels and positive pixels to move in each transformation.

work done in transforming the circle into itself as a 100%, we can derive a normalized expression for the similarity degree between two objects:

$$S_{1,2} = (1 - \frac{W_{1,2}}{W_{max}}) * 100\%, \qquad (2)$$

where $W_{1,2}$ is the work done in transforming one object into another, and $W_{max}$ is the maximum work done of the $N$ objects transformed. Then, we can be give a table with the work normalized in transforming all the objects into the circle (Table 3). Of course, we have shown some samples from a complete set composed of 674 pixels.

# 5 Orientation is crucial in performing the work

From last figures it can be seen what are the most similar objects to the circle, however, following one of the properties in pattern recognition, two near points in pattern space (in our case the space is composed of two features: the work done in the transformation of objects into the circle and the number of positive pixels), correspond to points very similar. For instance, it can be said that spider is more similar to beetle than to fly, because the two first are more near in pattern space. Circle with hair is more similar to beetle than to scorpion, and so forth. To give an accurate measure in the similitude

of every pair of objects, we have to transform every pair of them. Table 4 shows the values of the work, without normalizing, carried out to every pair of objects, while Table 5 shows values of normalized work.

| Objects | circle | circurr | circ_pel | bee | fly | beetle | spider | ring | scorpion | ring2 | sna |
|---|---|---|---|---|---|---|---|---|---|---|---|
| circle | 0 | | | | | | | | | | |
| circurr | 334,77 | 0 | | | | | | | | | |
| circ_pel | 1702 | 1818,89 | 0 | | | | | | | | |
| bee | 1785,16 | 1689,01 | 1607,14 | 0 | | | | | | | |
| fly | 2083,78 | 2042,55 | 1124,29 | 723,65 | 0 | | | | | | |
| beetle | 2893,49 | 2770,64 | 1351,85 | 1747,92 | 1762,42 | 0 | | | | | |
| spider | 3359,12 | 2195 | 3348,87 | 3212,7 | 3219,17 | 2570,85 | 0 | | | | |
| ring | 3531,62 | 3456,6 | 4061,87 | 3487,34 | 3535,87 | 3739,13 | 3763,97 | 0 | | | |
| scorpion | 4994,21 | 4913,96 | 4932,39 | 4627,46 | 4062,13 | 3421,89 | 3822,41 | 5181,84 | 0 | | |
| ring2 | 5483,75 | 5410,92 | 5649,88 | 5013,2 | 4885,95 | 4818,14 | 4633,84 | 1962,94 | 5897,01 | 0 | |
| snake | 6920,96 | 6823,44 | 6292,82 | 7401,57 | 7162,41 | 6755,82 | 5212,06 | 6047,34 | 4750,16 | 5816,78 | 0 |
| frame | 12903,1 | 12821,7 | 12538,4 | 12537,5 | 12226,8 | 11241,5 | 10004,2 | 9947,5 | 10452,5 | 8605,07 | 8672 |

a) Work done in the object transformation, acording to Hungarian algorithm.

| Objects | circle | circurr | circ_pel | bee | fly | beetle | spider | ring | scorpion | ring2 | sna |
|---|---|---|---|---|---|---|---|---|---|---|---|
| circle | 674 | | | | | | | | | | |
| circurr | 622 | 674 | | | | | | | | | |
| circ_pel | 574 | 553 | 674 | | | | | | | | |
| bee | 516 | 511 | 514 | 674 | | | | | | | |
| fly | 483 | 476 | 479 | 559 | 674 | | | | | | |
| beetle | 433 | 240 | 265 | 468 | 444 | 674 | | | | | |
| spider | 392 | 391 | 367 | 369 | 332 | 351 | 674 | | | | |
| ring | 291 | 291 | 235 | 268 | 239 | 213 | 204 | 674 | | | |
| scorpion | 379 | 371 | 357 | 353 | 318 | 352 | 276 | 204 | 674 | | |
| ring2 | 24 | 57 | 72 | 149 | 162 | 131 | 132 | 378 | 116 | 674 | |
| snake | 229 | 233 | 228 | 262 | 277 | 222 | 215 | 86 | 179 | 110 | 67 |
| frame | 85 | 78 | 81 | 65 | 52 | 69 | 111 | 52 | 58 | 46 | 8 |

b)Common pixels

Table 4. a) Work done in transformations without normalizing; b) common pixels.

Experimentally, it is found that in transforming spider into fly without aligning respect to their principal axes, making an angle of 28.6° between their greatest axes, and with 355 no common pixels, the work employed in the transformation is $W_H = 3,642.76$. On the other hand, if the objects are aligned as their principal axes, the work done is $W_H = 3,219.17$ reducing the work in a 11.6%. Making something similar with the spider and the snake, note three different cases shown in Table 6. In the first line it is presented the smallest number of pixels of the three cases. It is found that when they are superimposed, without aligned in terms of their principal axes (see Figure

4a) the work done is $W_H = 5,685.12$, with an amount of common pixels of 229. Meanwhile aligned in terms of their principal axes (Figure 4c)) the work done is $W_H = 5,212.06$. Thus, there is an error of 8.32%. An intermediate case is given in the second line (note how, despite to have small number of common pixels between two objects, the work done was smaller).

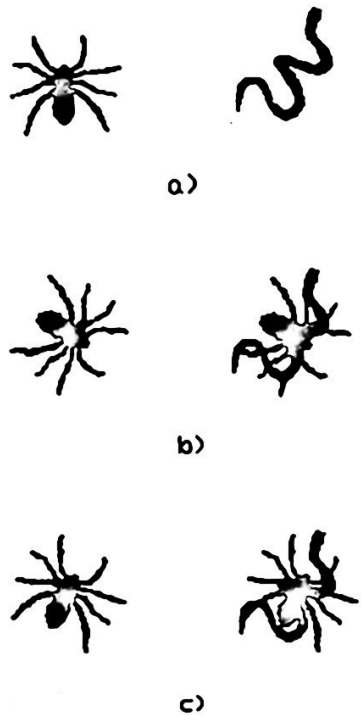



Figure 4. Transforming spider into snake; a)spider and snake b)their principal axes are at 90° between them, and the common pixels are 229; c) objects are rotation invariant with 215 common pixels

In last two lines, spider-fly, performance of transformation is better when grater axes are aligned (by 0.47° of error).

# 6 Morphing

According to Cohen-Or *et. al* [8] the term *morphing* is used for all methods that gradually and continuously deform one geometrical model into another. For Cheng et. al [9] morphing is a gradual change from one shape into another. Lee and Kim [10] employ Minkowski sum to transform one object into another by eliminating all redundant parts in convolution curves. Mathematically speaking, let $O^I = [O_0^I, ..., O_n^I]$ and $O^F = [O_0^F, ..., O_n^F]$ the point sets

| Objects | circle | circur | cirpel | bee | fly | beetle | spider | ring | scorpion | ring2 | snake |
|---|---|---|---|---|---|---|---|---|---|---|---|
| circle | 100 | | | | | | | | | | |
| circur | 97,41 | 100 | | | | | | | | | |
| cirpel | 86,81 | 85,90 | 100 | | | | | | | | |
| bee | 86,16 | 86,91 | 87,54 | 100 | | | | | | | |
| fly | 83,85 | 84,17 | 91,29 | 94,39 | 100 | | | | | | |
| beetle | 77,58 | 78,53 | 89,52 | 86,45 | 86,34 | 100 | | | | | |
| spider | 73,97 | 82,99 | 74,05 | 75,10 | 75,05 | 80,08 | 100 | | | | |
| ring | 72,63 | 73,21 | 68,52 | 72,97 | 72,60 | 71,02 | 70,83 | 100 | | | |
| scorpion | 61,29 | 61,92 | 61,77 | 64,14 | 68,52 | 73,48 | 70,38 | 59,84 | 100 | | |
| ring2 | 57,50 | 58,06 | 56,21 | 61,15 | 62,13 | 62,66 | 64,09 | 84,79 | 54,30 | 100 | |
| snake | 46,36 | 47,12 | 51,23 | 42,64 | 44,49 | 47,64 | 59,61 | 53,13 | 63,19 | 54,92 | 100 |
| frame | 0 | 0,63 | 2,83 | 2,83 | 5,24 | 12,88 | 22,47 | 22,91 | 18,99 | 33,31 | 32,78 |

Table 5. Normalized similarity measure, from 0% to 100% for every pair of objects compared.

| Object to Transform | Transformed Object Object | Angle between greatest axes | P(+) | $W_H$ |
|---|---|---|---|---|
| spider | snake | $169.15^o$ | 445 | 5,685.12 |
| spider | snake | $90^o$ | 459 | 5,598.48 |
| spider | snake | $1^o$ | 496 | 5,212.06 |
| spider | fly | $28.6^o$ | 355 | 3,270.25 |
| spider | fly | $0.47^o$ | 342 | 3,219.17 |

Table 6. Number of common pixels of both objects and positive pixels to be moved in every transformation.

representing the initial and final shapes of objects, respectively. Morphing from $O^I$ to $O^F$ is defined by a sequence of intermediate objects as:

$$\begin{aligned} O(t) &= uO^I + tO^F \\ &= [uO_0^I + tO_0^F, uO_1^I + tO_1^F, ..., uO_n^I + tO_n^F] \\ &= [O_0(t), O_1(t), ..., O_n(t)] \end{aligned}$$

where $u = 1 - t$. $O_i(t)$ is the *ith* point in the intermediate object, formed time $t$. The time parameter $t$ is normalized to the interval [0,1].

Binary representation of the objects showed here, permits transform them in order to compare them. But also, it permits break the objects to move positive pixels to conform intermediate objects. There are different orders move pixels. An order following here is by lines. We could obtain a morphing like that given by Figure 5, that shows the transformation of the fly into

beetle in steps of around 10 pixels, showing 22 different discrete times.



Figure 5: Morphing of fly into beetle

# Conclusions

this work, we have solved next problems: 1) similarity measure, like that given by Bribiescà and Wilson [6], has been normalized by studying compactness as an important characteristic of binary objects; 2) the measure was optimised by means of principal axes, that improves transformation of 2D objects; and 3) it has been shown that pixel representation can transform one object into another by moving positive pixels, giving a new technique to obtain a morphing. There were presented some irregular objects to apply the method of transforming 2D objects, giving good results in recognizing them.

The main disadvantage of this method is that it is slow, because every pair objects have to be compared, giving $N(N-1)/2$ comparisons. However is efficient and sound method to give the similarity degree between different 2D objects.

To improve the morphing, a future work could try to look for better functions, like that of interpolation or distance fields, giving at the same time, the similarity measure. Also, it should be better to improve the method of normalizing similarity measure without depending in the number of objects be compared.

# Acknowledgments

This work was in part supported by CONACyT and UNAM. I wish to express my gratitude to Dr. Ernesto Bribiesca for his help in reviewing this work.

# References

[1] Henk J.A.M. Heijmans, Alexander V. Tuzikov. Similarity and Symme Measures for Convex Shapes Using Minkowski Addition. IEEE actions on Pattern Analisys and Maching Intelligence. Vol. 20. Septiembre 1998.

[2] Van Otterloo J. Peter. A Contour-Oriented Approach to Shape Analy Prentice Hall, pp 368. 1991.

[3] Arkin, Esther M. Daniel; Chew L. Paul; Huttenlocher, Daniel P.; Klara; and Mitchel Joseph S. B. An Efficient Computable Metric Comparing Poligonal Shapes. IEEE Transactions on Pattern Analy and Machine Intelligence, vol. 13, no 3. 1991.

[4] Jain K., Fundamentals of Digital Image Processing. Prentice Hall mation and System Sciences Series. Thomas Kailath, series Editor. pp 569 .

[5] Bribiesca, Ernesto and Guzman, Adolfo. How to Describe pure form a how to Measure Differences in Shapes Using Shape Numbers. Patte Recognition 12,101-112 (1979).

[6] Bribiesca, E. and Wilson, R. G. A Measure of 2D Shape-of-Object similarity. Appl. Math. Lett. 10 6 (1997) 107-115.

[7] Bribiesca, E. Measuring 2-D shape compactness using the contact perim ter. Comput. Math. with Appl. 33: (11) 1-9 JUN 1997.

[8] D. Cohen-Or, D. Levin and A. Solomovici. Three-dimensional distan field methamorphosis, ACM Transactions on Graphics 17 (1998) 116-1

[9] Chen, Ho-Lun; Edelsbruner, Herbert and Fu, Ping. Shape space deformation. Computational Geometry. 19 (2001) 191-204.

[10] Lee, In-Kwon and Kim, Myung-Soo. Polynomial/rational approxim tion fo Minkowski sum boundary curves. Graphical models and processing. 60 (2)(1998) 136-165.

# Clasificación, segmentación y reconstrucción de líneas en mapas temáticos a color

Joaquín Decelis Burguete

Centro de Investigación en Computación,
Instituto Politécnico Nacional,
UPALM, Zacatenco, Edificio CIC
C. P. 07738, México D. F.
decelis@sagitario.cic.ipn.mx

**Resumen**

En este trabajo se presenta una metodología para extraer líneas de mapas temáticos de manera semiautomática, inmersas en imágenes compuestas en color de 24 bits en formato JPEG. La metodología propuesta se realiza en tres etapas: preprocesamiento, procesamiento y postprocesamiento. La primera de ellas inicia desde el momento de leer digitalmente el mapa en papel. En la segunda etapa, las líneas de interés son clasificadas y segmentadas por sus características de color y posición de cada uno de sus pixeles. También se introduce un factor discriminante que ayuda a la clasificación. Dentro de la tercera etapa, las líneas se convierten en vectores y son reconstruidas automáticamente.

**Palabras clave:** Interpretación automática de mapas cartográficos, segmentación a color, clasificación supervisada, operaciones morfológicas, capa vectorial.

## 1. Introducción

En vista de que actualmente los Sistemas de Información Geográfica (GIS - Geographical Information System) han adquirido gran importancia, existe la necesidad de producir nuevos mapas vectoriales y los que ahora están en papel se integren a los modernos sistemas automáticos.

El uso del escáner y los algoritmos de reconocimiento de patrones, establecen una alternativa prometedora para un análisis completo del mapa en forma automática o semiautomática. Estos modelos esencialmente deberán ser capaces de discriminar los elementos representados en el mapa y transformarlos a un formato estandarizado. Aunque esta idea no es nueva, las soluciones que minimicen la interacción humana aún se encuentran bajo investigación [9].

La importancia económica que tiene la conversión cartográfica automática, que muchos de los enfoques relacionados al tema sean comercializables, por cuya r no son publicados [10]. Hasta el momento, los trabajos publicados relacionados análisis de los mapas cartográficos pueden agruparse de la siguiente manera:

1) Según las herramientas de interpretación usadas: la morfología matemática d cional, la extracción de líneas rayadas [6], los procesos de conversión [5], etc.;
2) Sistemas para la segmentación de los mapas según el color [12];
3) Sistemas para la extracción de símbolos [2] y cadenas de caracteres media pareo de patrones y procesos de clasificación;
4) Sistemas para la extracción de redes [16] y curvas de nivel [7];
5) El uso de redes semánticas de información para dirigir la interpretación [14], o modelos orientados a objetos con capacidad de aprendizaje mediante mas interactivos;
6) Plataformas para convertir en vectores a los mapas que se presentan como automáticos o automáticos [1].

El problema principal para desarrollar la digitalización automática estriba cantidad de capas de información que contiene cada mapa, donde están parcial ocultas unas por otras [9]. Seguido aparecen varios tipos de problemas dentro de que se ha leído digitalmente, con las siguientes características:

1) El proceso de rastrear con el escáner es la principal fuente de errores, éste ce ruido Gausiano que distorsiona el color de la imagen capturada, provoc que el color de las líneas no sea constante en toda su trayectoria;
2) Elementos de otras capas que pueden tener el mismo color, éstos se tocan, zan y se traslapan;
3) Los colores cambian dependiendo del color del fondo o la región que atravies

Aunque los segmentos con rasgos lineales continuos, sin ramificaciones pueden ser detectados y ser convertidos en vector por procesos automáticos, los manuales son frecuentemente utilizados para este fin, además de unir segmentos las líneas son discontinuas dentro de una entidad continua.

La investigación enfocada a la interpretación de planos cartográficos tiene rios campos que tratar, dentro de éstos destaca la segmentación, la identificación separación de la capa de líneas [9].

Las dificultades principales de una interpretación totalmente automática son la densidad del mapa temático y a la variedad de leyendas. Se puede adoptar la de "pelar la cebolla" donde se eliminan del mapa las capas de información ya das para facilitar la interpretación de las restantes que son más complejas [4].

Para realizar este tipo de proyecto es recomendable desarrollar un sistema de bajo costo con poca intervención manual. No se considera la conversión en vector de forma interactiva o la interpretación automática con verificación humana exhaustiva como económicamente viable. Por el otro lado, un sistema completamente automático libre de error parece ser realista [10, 15].

En este trabajo se desarrollan métodos que funcionan de manera semiautomática y automática. El proceso es completo para minimizar las intervenciones del usuario que en mayoría de los casos es costoso.

# Metodología propuesta

metodología que aquí se propone comienza desde la lectura digital del mapa temático originalmente en papel y termina hasta convertir la línea de interés en vector para que un pueda manipularlo. Ésta es una alternativa que tiene la característica de su sencillez operación y casi nula intervención del usuario.

## 2.1. Descripción de la metodología

de las causas para que no se llegue a un buen resultado depende del proceso de su lectura digital, para ayudar a corregir en parte el problema, se realiza un preprocesado a la imagen para quitar el ruido obtenido en su adquisición y corregir en lo posible el cambio tonalidades que suceden al mezclarse los colores contenidos en el mismo. Se hace notar que en el proceso de promediado de los colores existentes no hay una función inverque nos pueda ayudar a corregir este evento.

Una vez que tengamos a la imagen en condiciones adecuadas, se aplicará alguno de cinco métodos diferentes de clasificación para obtener el conjunto de líneas deseadas continuar el proceso, la evaluación de cual de éstos es mejor para algún mapa digital determinado quedará delegada a la opinión humana [5], [6], [13].

El proceso siguiente será el de reducir la cantidad de pixeles que conforman a los segmentos de líneas encontradas sin perder información. En este proceso se aprovecha para quitar el excedente de ruido que se introdujo en la clasificación.

Posteriormente al convertirlos en vector se crean listas o tablas secuenciales con los puntos clasificados, para luego unir los diferentes extremos de las mismas y reconstruir posibles trayectorias.

### 1.1. Preprocesamiento

**Normalización de la imagen**. Para que el clasificador pueda trabajar de forma rápida y buenos resultados, la imagen debe estar con la menor cantidad de información indeseable proveniente del proceso de captura de la misma.

La normalización se realiza para cada uno de los tres colores primarios en RGB, en donde un color determinado será dependiente de los otros dos. Una vez expandida la imagen, se recorta de manera que todos los valores mayores o iguales a 255, se quedan en

del valor de ése pixel ±1.5 desviaciones estándar se guardan en un tercer plano. se agregan los pixeles del segundo plano. En el tercer paso, se repite el mismo con los valores del tercer plano, tomamos el valor de cada pixel ±1.0 desviaciones es dar. En los casos que cumplan con esta condición se guardan en un plano resultante al se agregan los del tercer plano mencionado.

**Búsqueda de colores [3]:** Aquí se considera la característica de que las líneas madas en los mapas con algunos colores específicos. De los $255^3$ colores diferentes puede contener una imagen en color de 24 bits, la reducimos a $3^3$, donde cada sub-cubo contendrá un conjunto limitado de niveles de intensidad. Tomamos al la línea de interés como centro del sub-cubo y clasificamos solamente a los que mismo orden RGB que el punto de la muestra.

En la figura 2 podemos observar la cantidad de información indeseable que adquieren diferentes clasificadores, el clasificador que más ruido capta es el de Búsqueda de y el que menos ruido capta es el de Expansión en tres pasos.

### 2.1.3. Postprocesamiento

**Eliminación de ruido y adelgazamiento:** Después de preparar a la imagen y realiz clasificación por alguno de los métodos anteriores (el que mejor resultado proporcion usuario experto dependiendo del mapa en cuestión), se prosigue a arreglar a las para conseguir mayor continuidad y reducir el grosor al mínimo evitando informa redundante para el proceso de reconstrucción.

**Filtrado:** Este filtro se aplica a imágenes en donde las líneas de interés se mezclan fondo del mismo color. Los valores de los pixeles a analizar son todos los contenidos la imagen y se comparan con el valor en RGB del punto seleccionado que se manti constante durante este proceso. De un punto a analizar se toman solamente sus 8-veci y se calcula para cada color RGB la desviación estándar. Se utiliza como centro analizado cuyo radio está dado por la desviación estándar $\sigma_{\text{Analizado}}$ calculada. Se si el punto seleccionado pertenece a este rango, de ser así permanece en la imagen da, el resto será eliminado.

**Binarizado:** Basándose en el hecho de que el color ya no nos proporciona más ción, realizamos el binarizado de la imagen que consiste en cambiar el valor de la dad de los pixeles que son diferentes del blanco a valores de intensidad negro.

**Cierre Morfológico:** Con este proceso vamos a suavizar los contornos de las fusionar separaciones estrechas, eliminar pequeños huecos y agujeros del contorno.

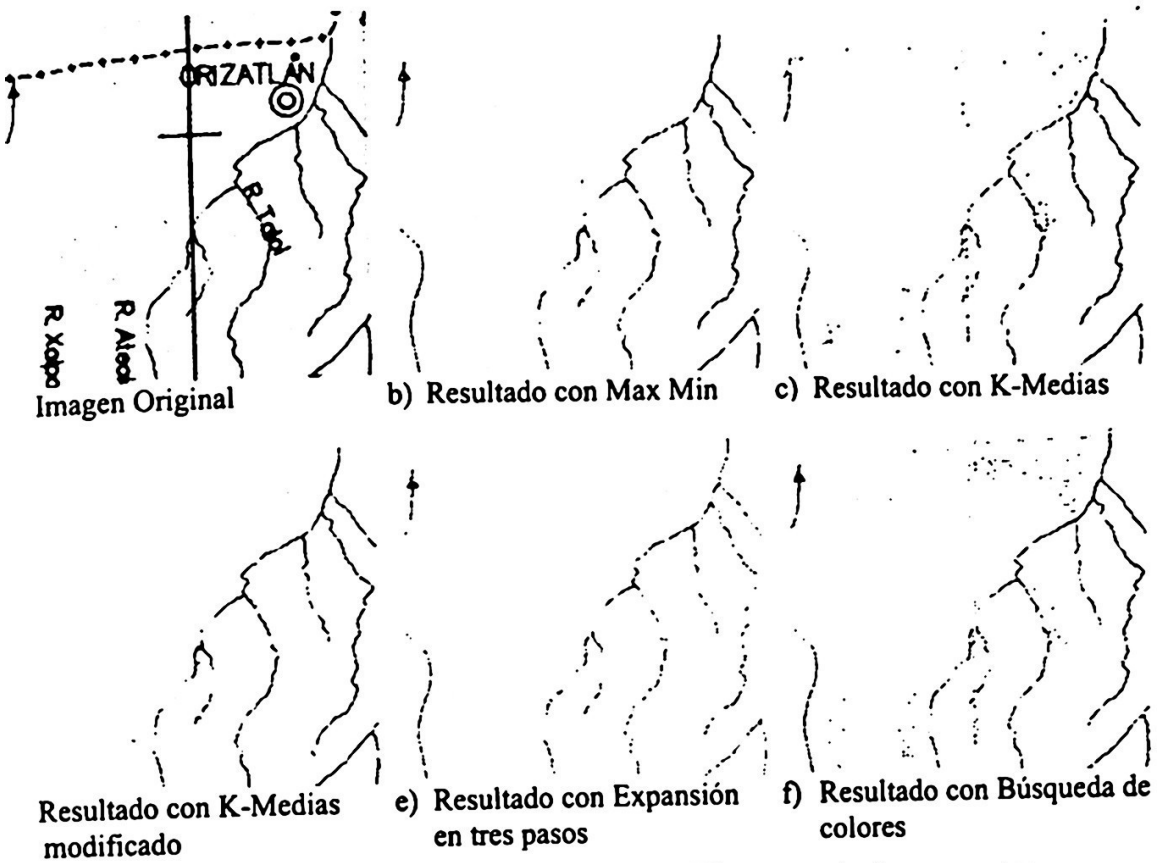



**Fig. 2.** Resultado de la clasificación con los diferentes métodos expuestos

**Eliminación de pixeles mal clasificados:** Existen pixeles que están en el límite del conunto clasificado, éstos pueden no pertenecer a las líneas de interés. Podemos eliminarlos verificamos que son 8-vecinos al conjunto de pixeles que no fueron seleccionados en el proceso de clasificación. Este tipo de filtro se utiliza cuando la imagen contiene objetos obscuros que la línea de interés y el fondo no es del mismo color que la línea buscapor ejemplo para eliminar el texto.

**Eliminación de pixeles aislados y en parejas:** Cuando quedan pixeles aislados podemos eliminarlos. Si pertenecen a la trayectoria de alguna línea posteriormente la recuperaremos. No es conveniente utilizarlo cuando han quedado las líneas muy discontinuas.

**Adelgazamiento:** La transformación morfológica al azar ("hit" or "miss") es una herramienta para la detección de formas. En esta ocasión la utilizaremos para encontrar el centro de la línea.

del valor de ése pixel ±1.5 desviaciones estándar se guardan en un tercer plano. se agregan los pixeles del segundo plano. En el tercer paso, se repite el mismo con los valores del tercer plano, tomamos el valor de cada pixel ±1.0 desviaciones es dar. En los casos que cumplan con esta condición se guardan en un plano resultante al se agregan los del tercer plano mencionado.

**Búsqueda de colores [3]:** Aquí se considera la característica de que las líneas madas en los mapas con algunos colores específicos. De los $255^3$ colores diferentes puede contener una imagen en color de 24 bits, la reducimos a $3^3$, donde cada sub-cubo contendrá un conjunto limitado de niveles de intensidad. Tomamos al la línea de interés como centro del sub-cubo y clasificamos solamente a los que mismo orden RGB que el punto de la muestra.

En la figura 2 podemos observar la cantidad de información indeseable que adquieren diferentes clasificadores, el clasificador que más ruido capta es el de Búsqueda de y el que menos ruido capta es el de Expansión en tres pasos.

### 2.1.3. Postprocesamiento

**Eliminación de ruido y adelgazamiento:** Después de preparar a la imagen y realiza clasificación por alguno de los métodos anteriores (el que mejor resultado proporcion usuario experto dependiendo del mapa en cuestión), se prosigue a arreglar a las para conseguir mayor continuidad y reducir el grosor al mínimo evitando informa redundante para el proceso de reconstrucción.

**Filtrado:** Este filtro se aplica a imágenes en donde las líneas de interés se mezclan fondo del mismo color. Los valores de los pixeles a analizar son todos los contenidos la imagen y se comparan con el valor en RGB del punto seleccionado que se manti constante durante este proceso. De un punto a analizar se toman solamente sus 8-veci y se calcula para cada color RGB la desviación estándar. Se utiliza como centro analizado cuyo radio está dado por la desviación estándar $\sigma_{\text{Analizado}}$ calculada. Se si el punto seleccionado pertenece a este rango, de ser así permanece en la imagen da, el resto será eliminado.

**Binarizado:** Basándose en el hecho de que el color ya no nos proporciona más ción, realizamos el binarizado de la imagen que consiste en cambiar el valor de la dad de los pixeles que son diferentes del blanco a valores de intensidad negro.

**Cierre Morfológico:** Con este proceso vamos a suavizar los contornos de las fusionar separaciones estrechas, eliminar pequeños huecos y agujeros del contorno.

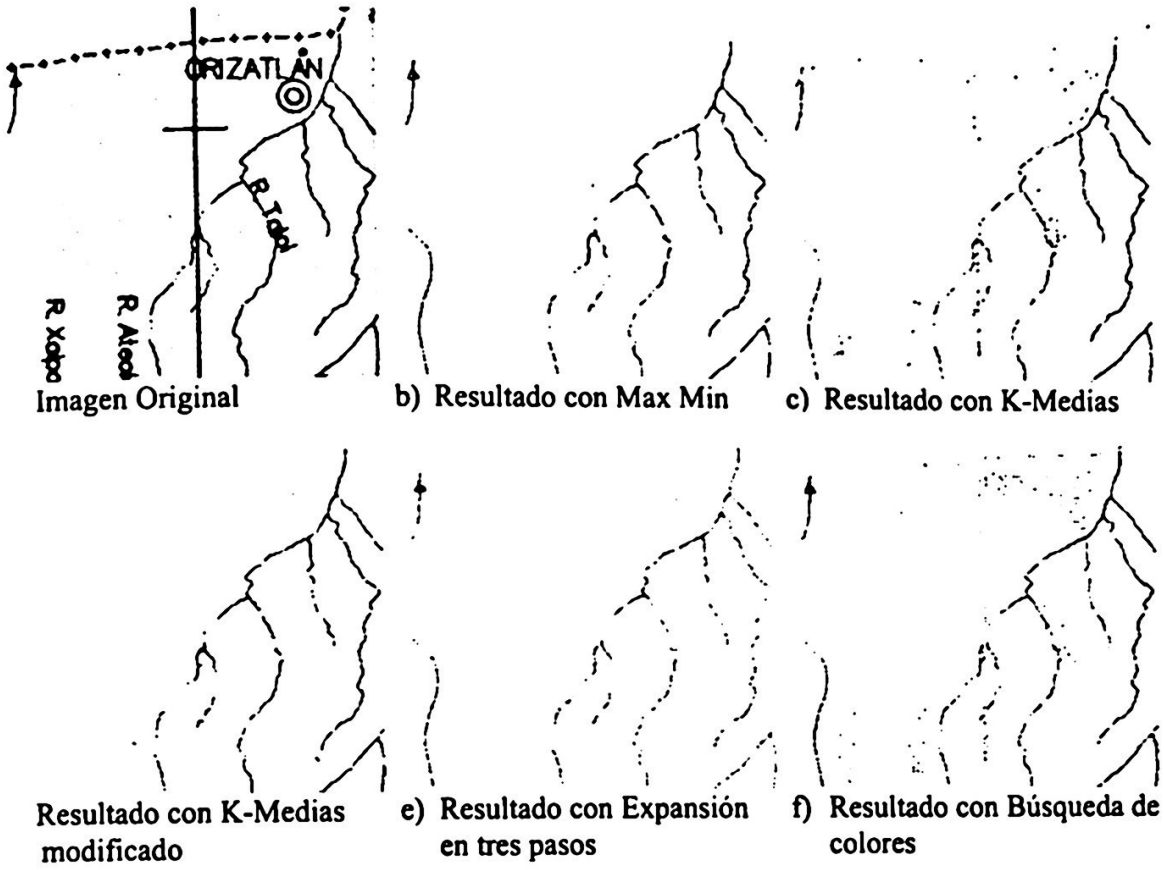



Imagen Original b) Resultado con Max Min c) Resultado con K-Medias

Resultado con K-Medias modificado e) Resultado con Expansión en tres pasos f) Resultado con Búsqueda de colores

**Fig. 2.** Resultado de la clasificación con los diferentes métodos expuestos

**Eliminación de pixeles mal clasificados:** Existen pixeles que están en el límite del conjunto clasificado, éstos pueden no pertenecer a las líneas de interés. Podemos eliminarlos verificamos que son 8-vecinos al conjunto de pixeles que no fueron seleccionados en el proceso de clasificación. Este tipo de filtro se utiliza cuando la imagen contiene objetos obscuros que la línea de interés y el fondo no es del mismo color que la línea busca- por ejemplo para eliminar el texto.

**Eliminación de pixeles aislados y en parejas:** Cuando quedan pixeles aislados podemos eliminarlos. Si pertenecen a la trayectoria de alguna línea posteriormente la recuperaremos. No es conveniente utilizarlo cuando han quedado las líneas muy discontinuas.

**Adelgazamiento:** La transformación morfológica al azar ("hit" or "miss") es una herramienta para la detección de formas. En esta ocasión la utilizaremos para encontrar el centro de la línea.

### 2.1.3.1. Reconstrucción

En los resultados anteriores, la mayoría de los casos contienen segmentos de líneas otros la ausencia de éstos, sin embargo toda esta información es necesaria para la trayectoria de las líneas.

**Búsqueda de extremos en líneas:** A partir de un extremo de la línea podremos otra que esté cercana. Para que un punto sea extremo debe cumplirse cualquiera condiciones:

1) El pixel de la línea que se está analizando debe tener un solo 8-vecino;
2) Si son dos 8-vecinos, entonces éstos deben ser 8-vecinos entre sí.

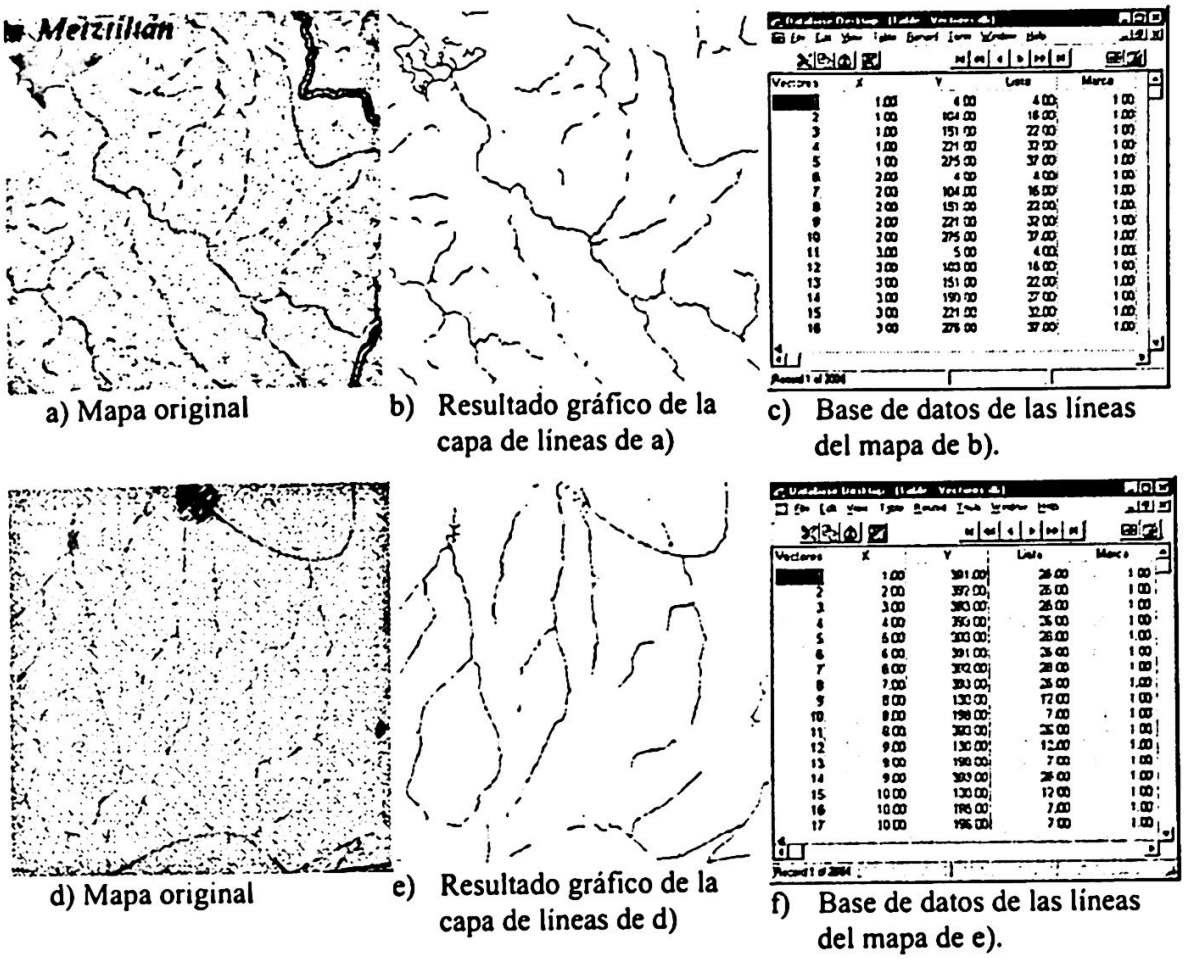
**Creación de listas:** Construiremos listas o segmentos de líneas con conjuntos que son 8-vecinos entre sí. Todos los pixeles se relacionarán a una base de datos respectivas listas.

**Búsqueda de candidatos:** Después de que se agregaron todos los puntos de la una tabla de la base de datos, comienza el proceso de buscar otro punto extremo; unirse a una lista determinada, con las siguientes características:

1) Debe ser el más próximo en caso de haber más de un punto extremo;
2) La distancia máxima entre puntos extremos debe definirse de tal manera unan líneas paralelas;
3) Las líneas de unión no deben atravesar o tocar ningún punto de las líneas tes en la imagen;
4) La línea recta nueva debe formar un ángulo entre 45° y 75° respecto a los puntos extremos de cada línea a unirse;
5) La relación ángulo-distancia es inversamente proporcional;
6) Cuando se unen dos listas éstas formarán una sola;
7) No se pueden unir puntos extremos de la misma lista;
8) Un punto extremo se puede unir una sola vez;
9) Las líneas nuevas se agregarán a la tabla de la base de datos.

De esta manera queda completo el proceso de convertir los centros de las líneas en vectores.

# Resultados

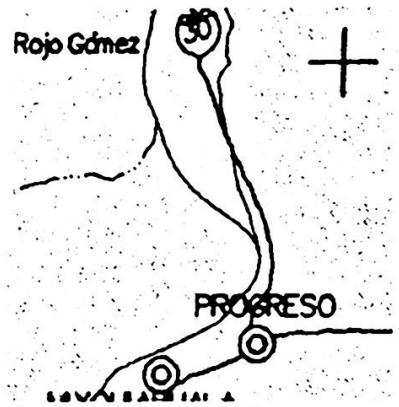



a) Mapa original b) Resultado gráfico de la capa de líneas de a) c) Base de datos de las líneas del mapa de b).

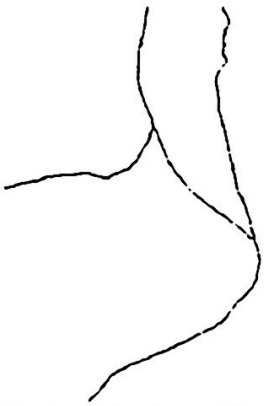
d) Mapa original e) Resultado gráfico de la capa de líneas de d) f) Base de datos de las líneas del mapa de e).

**Fig. 3.** Resultados del proceso de convertir en vector a las líneas inmersas en una región del mismo color.

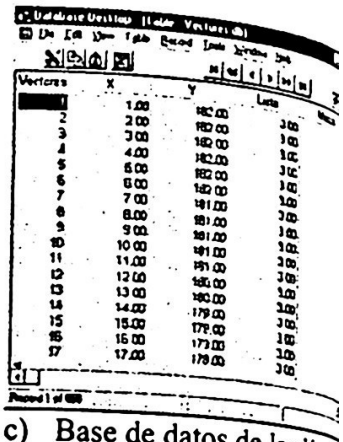
En la figura 3 podemos observar algunos resultados de procesamiento completo del sistema aplicado a dos imágenes complejas exploradas digitalmente a color. Las características de esta imagen son: formato JPEG, 500 ppp, color de 24 bits. La dificultad principal para procesar esta imagen es que las líneas están inmersas en el fondo del mismo color, y aún visualmente es difícil distinguirlas. Se observa también la presencia de otros objetos tales como texto, carreteras y rayas artificiales provocadas por las condiciones de papel. En las imagenes de la figura 4d) y 4g), el texto aparece cubriendo las líneas de interés, se observa la presencia de diferentes colores y tonos a lo largo de unas líneas. Además, los extremos de las líneas que se desean unir, están muy cercanos a otras líneas que son paralelas y que no deben ser unidas. Después de la clasificación y segmentación se reconstruyen las líneas discontinuas (Fig. 3b), 3e), 4b), 4e) y 4h)) y se genera la base de datos de las líneas al convertirlas en vector (Fig. 3c), 3f), 4c), 4f) y 4i)), lista para ser incluida en un GIS.




a) Mapa original, formato JPEG, 76 ppp, 24 bits.



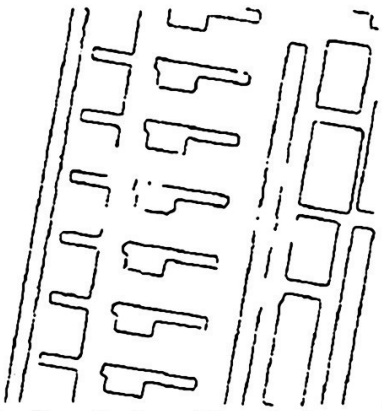
b) Resultado gráfico de la capa de líneas de a)



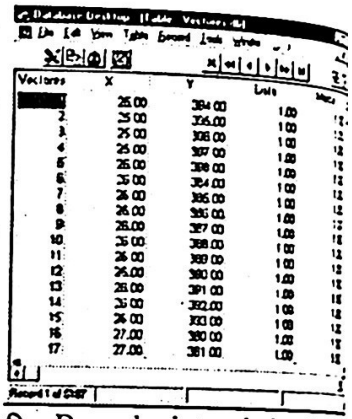
c) Base de datos de las líneas del mapa de b).




d) Mapa original, formato JPEG, 500 ppp, 24 bits.



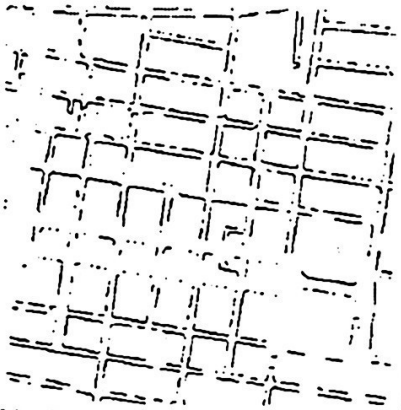
e) Resultado gráfico de la capa de líneas de d)



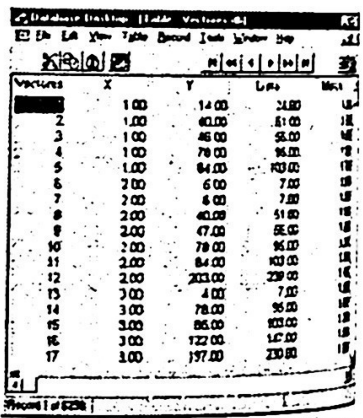
f) Base de datos de las líneas del mapa de e).




g) Mapa original, formato GIF, 96 ppp, 8 bits.



h) Resultado gráfico de la capa de líneas de g)



i) Base de datos de las líneas del mapa de h).

**Fig. 4.** Resultados del proceso de convertir en vector varios mapas con direfentes características.

## Conclusiones y trabajo a futuro

este trabajo se realizó un tratamiento completo para digitalizar líneas en mapas provenientes de papel que parte desde la etapa de la lectura digital hasta su conversión a vector, donde se incluyen las etapas de: 1) Preprocesamiento (Normalización), donde se preparó mapa temático para la transformación adecuada del papel a imagen digital. Se utilizaron mapas en color de 24 bits en formato JPEG, aunque también trabaja con otros formatos; 2) Procesamiento (Conjunto de clasificadores), donde se propuso un conjunto de clasificadores semiautomáticos eficientes que identifican a cada una de las partes de una línea; 3) Postprocesamiento (Adelgazamiento y conversión en vector de forma automática), donde se recuperó en el espacio correspondiente las líneas completas de los traslapes por otros objetos presentes en el mapa temático; 4) Generación automática de la base de datos de las líneas convertidas en vectores; 5) Reconstrucción de líneas interrumpidas por otros objetos presentes en el mismo mapa digital; 6) Generación de la capa vectorial de objetos lineales y optimización de la intervención de los expertos humanos.

Se incluirá el sistema propuesto al sistema integral A2R2V (from Analogic-to-Raster-Vector) para la interpretación automática de mapas cartográficos leídos digitalmente en color que se desarrolla en este instituto de investigación [10-12].

## Referencias

] Ablameyko, S., Bereishik, V., Kryuchkov, A.: Automated Map-Drawing Digitizing Technology Based on Scanner Input. IAPR Workshop on Machine Vision Applications, (1994), pp. 48-51.

Chhabra, A. K.: Graphic Symbol Recognition: an Overview. IAPR Lecture Notes in Computer Science, vol. 1389, Graphics Recognition Algorithms and Systems, August, (1998), pp. 68-79.

Decelis-Burguete, J.: Digitalización automatizada de líneas en mapas raster. Tesis de M. en C., Centro de Investigación en Computación – Instituto Politécnico Nacional (México, D. F.), noviembre, (2001).

Deseilligny, P., Mariani, R., Labiche, J., Mullot, R.: Topographic Maps Automatic Interpretation: Some Proposed Strategies. IAPR Lecture Notes in Computer Science, vol. 1389, Graphics Recognition Algorithms and Systems, August, (1998), pp. 175-193.

Doermann, D.S.: An Introduction to Vectorization and Segmentation. IAPR Lecture Notes in Computer Science, vol. 1389, Graphics Recognition Algorithms and Systems, (1998), pp. 1-8.

Dori, D., Wenyin, L., Peleg, M.: How to Win a Dashed Line Detection Contest. IAPR Lecture Notes in Computer Science, vol. 1072, Graphics Recognition Algorithms and Systems, (1996), pp. 286-300.

Dupont, F., Deseillingny, M. P., Gondran, M.: Automatic Interpretation of Scanned Maps: Reconstruction of Contour Lines. IAPR Lecture Notes in Computer Science, vol. 1389, Graphics Recognition Algorithms and Systems, (1998), pp. 194-206.

[8] Gómez-Allende, D.: Reconocimiento de formas y visión artificial. Ed. Wesley Iberoamericana, 1994.

[9] Levachkine, S. P., Polchkov, E. A.: Cartographic Pattern Recognition. Internacional MICAI 2000, 10-14 April Acapulco, México, (2000), pp. (ISBN 970-18-4465-3).

[10] Levachkine, S. P., Polchkov, E. A.: Integrated Technique for Automated Digitiz of Raster Maps. http://www.revista.unam.mx/vol.1/art4/index.html, Revista Universitaria, Vol. 1, No. 1, 30 de junio, (2000).

[11] Levachkine, S. P., Velázquez, A., Alexandrov, V.: Color Image Segmentation False Colors and Its Applications to Geo-Images Treatment: Alphanumeric ters Recognition. International Geoscience and Remote Sensing Symposi IGARSS 2001, Sydney, Australia, (IEEE Catalog Number (CD-RO 01CH37217C; Library of Congress Number: 01-087978; ISBN CD-ROM: 7033-3), 9-13 July, (2001).

[12] Levachkine, S. P., Velázquez, A., Alexandrov, V., Kharinov, M.: Semantic and Recognition of Raster-Scanned Color Cartographic Images. Proc 4th, IARP Workshop on Grapics Recognition – GREC 2001, 7-8 September Kingston, Canada, (2001), pp. 255-266.

[13] Myers, G. K., Mulgaonkar, P. G., Chen, C. H., DeCurtins, J. L., Chen, E.: tion-Based Approach for Automated Text and Feature Extraction from Scanned Maps. IAPR Lectures Notes in Computer Science, vol. 1072, Graphics R cognition (1996), pp. 190-203.

[14] Nishijima, M., Watanabe, T.: An Automatic Extraction Approach of Road Infor tion on the Basis of Recognition of Character Regions. Lecture Notes in Compu Sciences, vol. 1024 (1995), pp. 173-180.

[15] Ogier, J. M., Adam, S., Vessaid, A., Bechar, H.: Automatic Topographic Map sis System. Proc. 4th. IAPR Int. Workshop on Graphics Recognition-GREC 2001, September Kingston, Ontario, Canada, (2001), pp. 229-244.

[16] Schavemaker, J. G. M., Reinders, M. J. T.: Information Fusion for Conflict tion in Map Interpretation. IARP Lecture Notes in Computer Science, vol. Graphics Recognition Algorithms and Systems (1998), pp. 231-242.

# Estrategias Evolutivas Aplicadas a Registro de Imágenes

Héctor Fernando Gómez García, Arturo Hernández Aguirre,
José Luis Marroquín Zaleta, Salvador Botello Rionda.

Centro de Investigación en Matemáticas, CIMAT.
Departamento de Ciencias de la Computación. Guanajuato, México.

hector, artha, jlm, botello@cimat.mx

**Resumen**

Presentamos un algoritmo de registro de imágenes basado en estrategias evolutivas como técnica de optimización. Se propone el uso de una pirámide Gaussiana, es decir, las imágenes a registrar son multiescaladas y filtradas en un proceso descendente; posteriormente son registradas de manera ascendente. Al comparar con métodos de gradiente nuestra técnica mostró ser mas robusta e insensible al ruido.

## 1. Introducción

El registro de imágenes es una técnica de gran importancia dentro del procesamiento digital de imágenes. Es utilizado para alinear espacialmente una o más imágenes adquiridas en instantes de tiempo diferentes, desde perspectivas distintas, o mediante varias modalidades. Su aplicación ha resultado en una herramienta de gran utilidad en varias áreas, entre las que destacan la visión computacional, el procesamiento de datos adquiridos remotamente y la medicina, en esta última es utilizada con fines de diagnóstico de enfermedades principalmente [1].

La computación evolutiva está formada por un conjunto de técnicas de optimización estocástica entre las que encontramos a los algoritmos genéticos, la programación evolutiva y las estrategias evolutivas. Estas últimas se destacan del resto debido a la facilidad con que son implementadas y al gran porcentaje de éxito que se obtiene en sus aplicaciones [7]. En el presente trabajo exploramos la aplicación de las estrategias evolutivas al problema de registro de imágenes (una descripción mas completa desarrollada por los autores se encuentra en [4]).

## 2. Registro de Imágenes

Matemáticamente afirmamos que un par de imágenes $\mathbf{I}_1(\mathbf{x})$, $\mathbf{I}_2(\mathbf{x})$ se encuentran registradas, si podemos encontrar una transformación $T(\mathbf{x})$, tal que la siguiente igualdad se cumpla

$$\mathbf{I}_1[T(\mathbf{x})] = \mathbf{I}_2(\mathbf{x}) \tag{1}$$

A lo largo de este trabajo, llamaremos a $\mathbf{I}_1(\mathbf{x})$ *imagen de referencia*, $\mathbf{I}_2(\mathbf{x})$ *imagen objetivo*, $T(\mathbf{x})$ es una *transformación de coordenadas* que toma su argumento $\mathbf{x}$ de $Z^2$ en el caso de imágenes bidimensionales, o de $Z^3$ para arreglos de imágenes o imágenes tridimensionales. La imagen $\mathbf{I}_1[T(\mathbf{x})]$, se denomina *imagen de prueba* y representa el resultado de aplicar la transformación $T(\mathbf{x})$ a la imagen de referencia. Los valores de intensidad de la imagen de prueba se calculan mediante interpolación numérica sobre los puntos con coordenadas enteras de la imagen de referencia pudiendo utilizar para ello interpolación por splines o simplemente interpolación lineal, dependiendo del grado de exactitud que se quiera obtener.

La ecuación 1, representa una igualdad entre matrices, en la que cada entrada se hace corresponder con un punto de la imagen, y el valor numérico de dicha entrada mide la intensidad del punto de la imagen asociado a ella.

Si la transformación de coordenadas $T(\mathbf{x})$ puede ser definida mediante un conjunto pequeño de parámetros $\theta = [\theta_1, \ldots, \theta_m]$ (como supondremos en el resto de este trabajo), el registro se denomina *registro paramétrico* y el problema se reduce a encontrar los valores de los componentes del vector $\theta$ que aproximen de la mejor manera la ecuación 1. Entre tales transformaciones encontramos las *transformaciones afines* las cuales pueden expresarse como la multiplicación de una matriz cualquiera por el vector de coordenadas $\mathbf{x}$ más un vector de traslación

$$T(\theta, \mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{t} \tag{2}$$

La matriz $\mathbf{A}$ puede considerarse como la composición de transformaciones matriciales más simples [3]

$$\mathbf{A} = \mathbf{RSC}$$

$\mathbf{R}$ es una matriz de rotación, $\mathbf{S}$ es una matriz de escalamiento y $\mathbf{C}$ es una matriz de cizallamiento. La factorización de la matriz $\mathbf{A}$ no se realiza necesariamente en ese orden.

Las transformaciones afines son de gran utilidad en el registro de imágenes debido a la composición de operaciones que engloban.

A partir de la ecuación 1. podemos definir un valor residual sobre cada punto de la imagen objetivo de la siguiente manera

$$r(\mathbf{x}_i) = I_1[T(\theta, \mathbf{x}_i)] - I_2(\mathbf{x}_i)$$

$i = 1, \ldots, N$. Siendo $N$ el número de puntos de la imagen objetivo.

Los algoritmos de registro buscarán minimizar este valor para cada punto de la imagen objetivo.

# 3. Estrategias Evolutivas

Las *estrategias evolutivas* son técnicas de optimización estocásticas que imitan los procesos evolutivos de las especies en la naturaleza [5],[6]. Están basadas en una población de individuos (cada uno de los cuales representa una posible solución al problema de optimización) sobre la cual se aplican operadores genéticos probabilísticos como la recombinación y la mutación, para dirigir la búsqueda hacia el óptimo mediante la selección de los mejores individuos generados durante el proceso. Las estrategias evolutivas buscan encontrar el vector de parámetros $\theta^*$ que maximice la función objetivo $f(\theta) : \Re^m \rightarrow \Re$

Cada individuo es representado a través de dos vectores $m-dimensionales$

$$\mathbf{I} = \mathbf{a} = [\theta, \sigma] = [\theta_1, \ldots, \theta_m, \sigma_1, \ldots, \sigma_m]$$

El vector $\theta$ se denomina vector de *variables objetivo* y está compuesto por los parámetros sobre los cuales se desea optimizar. El vector $\sigma$ se denomina vector de *variables de control*, cada componente de este vector se asocia con un componente del vector objetivo y se utiliza como argumento para la aplicación del operador de mutación.

A partir de un conjunto de individuos, definimos las poblaciones de *padres* e *hijos* como

Población de Padres = $\mathbf{P}_p = \{\mathbf{a}_1, ..., \mathbf{a}_\mu\}$, $\mathbf{P}_p \in \mathbf{I}^\mu$.

Población de Hijos = $\mathbf{P}_h = \{\mathbf{a}'_1, ..., \mathbf{a}'_\lambda\}$, $\mathbf{P}_h \in \mathbf{I}^\lambda$.

Se asocia a cada individuo un valor real denominado *aptitud* a través de una función $\Phi$ que toma como argumento el vector de variables objetivo del individuo, y que debe representar la calidad del individuo como solución al problema de optimización, de manera que los mejores individuos tengan un valor de aptitud más alto que los peores. Generalmente $\Phi$ será igual a la función objetivo que se busca maximizar, cumpliéndose la siguiente igualdad

$$\Phi(\mathbf{a}) = f(\theta)$$

en donde $\theta$ es el vector de variables objetivo del individuo $\mathbf{a}$.

El operador de *recombinación* $r : \mathbf{I}^\mu \rightarrow \mathbf{I}^\lambda$, permite generar la población de hijos a partir de la de los padres mediante la mezcla de la información codificada en los individuos. El operador de recombinación se puede considerar como la aplicación repetida ($\lambda$ veces) de los operadores de selección, *se*, y cruza, *cr*. En cada aplicación se genera un nuevo hijo. El operador *se*, escoge de manera aleatoria a los individuos necesarios para la cruza entre la población de padres. El operador *cr* genera los valores de las variables del nuevo hijo como un promedio pesado entre los valores de los individuos que selecciona el operador *se*.

El operador de *mutación* $m : \mathbf{I}^\lambda \rightarrow \mathbf{I}^\lambda$, aplica pequeños cambios en los valores de los componentes de cada individuo de la población de hijos. Inicialmente se

mutan las variables de control y los valores obtenidos son utilizados para mutar los componentes del vector de variables objetivo, de la siguiente manera

$$\begin{aligned} \sigma_i' &= \sigma_i \exp\left[\tau' N(0,1) + \tau N_i(0,1)\right] \\ \theta_i' &= \theta_i + N(0,\sigma_i) \end{aligned}$$

$i = 1, \ldots, m$
$\tau' = \frac{1}{\sqrt{2\sqrt{m}}}$
$\tau = \frac{1}{\sqrt{2m}}$

$N(0,1)$ es un número aleatorio de una distribución normal con media cero, que se genera al inicio de la mutación del individuo y que permanece fijo al mutar cada componente.

$N_i(0,1)$ es un número aleatorio de una distribución normal con media cero, generado para cada componente.

Los valores $\sigma_i'$ y $\theta_i'$ sustituyen a los valores $\sigma_i$ y $\theta_i$ en el individuo sobre el que se aplica la mutación.

El proceso de *selección* permite generar la nueva población de padres, escogiendo para ello a los $\mu$ mejores individuos de acuerdo a su valor de aptitud. En este sentido, encontramos dos versiones del operador de selección, cada una de las cuales origina una estrategia evolutiva distinta. La primer versión selecciona a los $\mu$ mejores individuos de la población generada por la unión de padres e hijos, $s_{(\mu+\lambda)} : \mathbf{I}^{\mu+\lambda} \rightarrow \mathbf{I}^{\mu}$, su aplicación da lugar a la denominada *Estrategia Evolutiva* $(\mu + \lambda)$. La segunda versión opera sobre la población de hijos solamente, $s_{(\mu,\lambda)} : \mathbf{I}^{\lambda} \rightarrow \mathbf{I}^{\mu}$, escogiendo a los $\mu$ mejores individuos entre ellos, dando lugar a la denominada *Estrategia Evolutiva* $(\mu, \lambda)$, en la que necesariamente $\mu \leq \lambda$. En la Estrategia Evolutiva $(\mu + \lambda)$, un individuo de la población de padres sobrevive indefinidamente, hasta que se genera un número suficiente de hijos que lo superen, o por lo menos, lo igualen en aptitud, mientras que en la Estrategia Evolutiva $(\mu, \lambda)$ un padre vive solamente una generación.

El algoritmo general de una estrategia evolutiva se resume a continuación

1.– Generar $\mathbf{P}_{p,0} = \{\mathbf{a}_1, \ldots, \mathbf{a}_\mu\}$
2.– Evaluar Aptitud $\Phi(\mathbf{P}_{p,0}) = \{\Phi(\mathbf{a}_1), \ldots, \Phi(\mathbf{a}_\mu)\}$
3.– Iniciar contador de generaciones, $g = 0$.
4.– Mientras (*criterio de paro*$(\mathbf{P}_{p,g}) = falso$)
  Recombinar $r(\mathbf{P}_{p,g}) \rightarrow \mathbf{P}_{h,g} = \{\mathbf{a}_1, \ldots, \mathbf{a}_\lambda\}$
  Mutar $m(\mathbf{P}_{h,g}) \rightarrow \mathbf{P}'_{h,g} = \{\mathbf{a}'_1, \ldots, \mathbf{a}'_\lambda\}$
  Seleccionar
    $s_{(\mu+\lambda)}(\mathbf{P}_{p,g} \cup \mathbf{P}'_{h,g}) = \mathbf{P}_{p,g+1}$
    ó
    $s_{(\mu,\lambda)}(\mathbf{P}'_{h,g}) = \mathbf{P}_{p,g+1}$
  Incrementar generaciones, $g = g + 1$.
5.– Fin

El criterio de paro puede establecerse mediante un número fijo de generaciones, mediante una diferencia pequeña entre la aptitud del mejor y del peor

individuo o cuando algún individuo alcance un valor de aptitud apropiado al problema de optimización.

# 4. Registro de Imágenes Basado en Estrategias Evolutivas

En esta sección se presenta un algoritmo de registro, el cual utiliza las estrategias evolutivas como técnica de optimización sobre los residuos en los puntos de las imágenes a registrar.

La codificación de cada individuo contiene como variables objetivo los parámetros de la transformación utilizada para realizar el registro

$$\mathbf{I} = \mathbf{a} = \{\theta_1, \ldots, \theta_m, \sigma_1, \ldots, \sigma_m\}$$

de manera que cada individuo representa una transformación $T(\theta)$ distinta.

Tomando en cuenta que nuestra implementación de estrategias evolutivas trabaja bajo un esquema de maximización (una simple modificación convierte el algorítmo en minimizador), la función de aptitud se eligió de manera que su valor fuera inversamente proporcional al residual

$$\Phi(\mathbf{a}) = \frac{1}{1 + \frac{1}{N}\sum_{\mathbf{x}_i \in CI} |r(\mathbf{x}_i)|} = \frac{1}{1 + \frac{1}{N}\sum_{\mathbf{x}_i \in CI} |I_1[T(\theta, \mathbf{x}_i)] - I_2(\mathbf{x}_i)|}$$

$\theta$, vector de variables objetivo del individuo **a**.

$CI$, conjunto de puntos utilizado para realizar la comparación entre las imágenes.

$N$, cardinalidad del conjunto $CI$.

En lugar de utilizar todos los puntos de la imagen para realizar la comparación, lo que haría la evaluación de la función de aptitud muy cara, se utiliza un conjunto reducido de puntos de interés $CI$, que pueden escogerse equiespaciados sobre la imagen de referencia o aleatoriamente distribuidos sobre toda la imagen mediante una distribución uniforme en las coordenadas. En este último caso, el conjunto puede renovarse al transcurrir un número fijo de generaciones, dando buenos resultados al trabajar con imágenes ruidosas.

El número de puntos utilizados para evaluar la aptitud, dependerá de la variabilidad de las intensidades de las imágenes a registrar y del tamaño de las mismas. En la práctica hemos obtenido buenos resultados utilizando alrededor del 0.5 % de los puntos de la imagen.

La función de aptitud asigna a cada individuo un valor en el intervalo $(0, 1]$, variando de la *no correspondencia* hasta la *correspondencia perfecta*, evaluada sobre el conjunto $CI$.

Es posible relacionar un error promedio en el registro, en base a la función de aptitud

$$\textit{error promedio}(\mathbf{a}) = \frac{1}{\Phi(\mathbf{a})} - 1$$

lo cual puede utilizarse como un criterio de paro del algoritmo, al detener la búsqueda cuando algún individuo alcance un valor de aptitud que tenga asociado un error promedio menor a una tolerancia definida por el usuario. la valoración de este error promedio, se debe tomar en cuenta el rango de intensidades de las imágenes a registrar.

La aplicación de este algoritmo se beneficia notablemente de la utilización de una representación multiescala de las imágenes a registrar. Este esquema produce un arreglo piramidal de imágenes aplicando sucesivamente las operaciones de filtrado y submuestreo, en ese orden, sobre las imágenes originales. El filtrado puede realizarse mediante la convolución de la imagen con un kernel gaussiano, por ejemplo. En cada nivel del arreglo piramidal identificamos par de imágenes producto de las imágenes objetivo y de referencia cada una, las cuales aplicamos el algoritmo de registro, comenzando en el nivel más y transportando la solución de cada nivel al nivel inmediato inferior, terminando en el más bajo, en el cual las imágenes originales son registradas. En cada nivel, se genera una nueva población, en la que se siembra el individuo con mayor aptitud obtenido en el nivel inmediato inferior y el resto de la población se genera aleatoriamente sobre un espacio centrado en las variables objetivo del individuo sembrado, la amplitud de dicho espacio puede reducirse entre niveles ya que suponemos que el óptimo está siendo acotado. Deberá tenerse cuidado de reescalar los parámetros que son afectados por la escala al transportar soluciones entre niveles, como puede ser el caso de los parámetros relacionados con la traslación. El esquema multiescala permite tratar con problemas de do y de grandes transformaciones, ya que produce la eliminación de estructuras débiles y además es posible trabajar con un número reducido de puntos en niveles más altos [8].

# 5. Resultados

Para evaluar el desempeño del algoritmo propuesto, se comparó con el método de Gauss-Newton el cual se aplicó para realizar la minimización de la Suma de Diferencias al Cuadrado SSD. sobre todos los puntos de la imagen

$$f(\theta) = \sum_{i=0}^{N} [r(\mathbf{x}_i)]^2 = \sum_{i=0}^{N} \{I_1 [T(\theta, \mathbf{x}_i)] - I_2(\mathbf{x}_i)\}^2$$

La comparación, se realizó a través de transformaciones afines sintéticas aplicadas a diferentes imágenes bidimensionales. Para este caso la transformación de coordenadas puede escribirse

$$T(x, y) = \begin{bmatrix} \theta_1 & \theta_2 \\ \theta_3 & \theta_1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \theta_5 \\ \theta_6 \end{bmatrix}$$

y como se señaló en la Sección 2. esta transformación es la composición una rotación. un escalamiento. un cizallamiento y una traslación

$$T(x, y) = \mathbf{RECx} + \mathbf{t} \tag{3}$$

$\mathbf{R} = \begin{bmatrix} \cos\alpha & -\operatorname{sen}\alpha \\ \operatorname{sen}\alpha & \cos\alpha \end{bmatrix}$, matriz de rotación de un ángulo $\alpha$.

$\mathbf{E} = \begin{bmatrix} e_x & 0 \\ 0 & e_y \end{bmatrix}$, matriz de escalamiento con factores $e_x$, $e_y$.

$\mathbf{C} = \begin{bmatrix} 1 & c_x \\ 0 & 1 \end{bmatrix}$ ó $\begin{bmatrix} 1 & 0 \\ c_y & 1 \end{bmatrix}$, matriz de cizallamiento sobre eje $x$ o sobre eje $y$, respectivamente.

$\mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$, vector de traslación.

Las pruebas se realizaron generando valores aleatorios para ángulos, factores de escala, cizallamiento y traslación mediante una distribución uniforme sobre intervalos numéricos cuya amplitud se fue aumentando gradualmente, y construyendo las transformaciones afines determinadas por los valores generados. Para cada intervalo, se produjo un conjunto de 20 transformaciones. Se aplicaron las transformaciones a las imágenes seleccionadas y se probaron los algoritmos sobre ellas, obteniendo como resultado para cada problema los 6 parámetros de una transformación afín. De estos parámetros se recuperaron el ángulo de rotación, factores de escala. cizallamiento y traslación. Dichos valores se obtuvieron mediante las siguientes relaciones [3]

$$\begin{aligned} e_x &= \sqrt{\theta_1^2 + \theta_3^2}.\ e_y = \frac{\theta_1\theta_4 - \theta_2\theta_3}{\sqrt{\theta_1^2 + \theta_3^2}} \\ \operatorname{sen}\alpha &= \frac{\theta_3}{e_x},\ \cos\alpha = \frac{\theta_1}{e_x} \\ c_x &= \frac{\theta_1\theta_2 + \theta_2\theta_4}{\sqrt{\theta_1^2 + \theta_3^2}} \\ t_1 &= \theta_5,\ t_2 = \theta_6 \end{aligned}$$

Las relaciones anteriores son válidas bajo el supuesto de que la multiplicación de matrices se realizó en el orden que especifica la ecuación 3 y que no se realizó cizallamiento en el eje $y$.

Los valores encontrados se compararon con los reales.

Los intervalos numéricos para los valores aleatorios de ángulos. factores de escala. cizallamiento y traslación. para cada conjunto de transformaciones se muestran en la Tabla 1

Ambos métodos se ejecutaron bajo un esquema multiescala utilizando pirámides gaussianas de 4 niveles. Se utilizó una Estrategia Evolutiva (50 + 250). Se utilizó como punto inicial de ambos algoritmos la transformación identidad. Para el algoritmo Gauss-Newton. se permitió un máximo de 1000 iteraciones en cada nivel. pudiendo terminar en caso de que no se produjeran cambios significativos sobre el vector de parámetros. En el caso de estrategias evolutivas. se utilizaron 256 pixeles para evaluar la función de aptitud en cada nivel. Se

| Par | 1er. Con. | 2do. Con. | 3er. Con. | 4to. Con. | 5to. Con. |
|---|---|---|---|---|---|
| $\theta$ | $\pm 10°$ | $\pm 20°$ | $\pm 30°$ | $\pm 40°$ | $\pm 50°$ |
| $e_x$ | $1 \pm 0{,}1$ | $1 \pm 0{,}2$ | $1 \pm 0{,}3$ | $1 \pm 0{,}4$ | $1 \pm 0{,}5$ |
| $e_y$ | $1 \pm 0{,}1$ | $1 \pm 0{,}2$ | $1 \pm 0{,}3$ | $1 \pm 0{,}4$ | $1 \pm 0{,}5$ |
| $c_x$ | $\pm 0{,}1$ | $\pm 0{,}2$ | $\pm 0{,}3$ | $\pm 0{,}4$ | $\pm 0{,}5$ |
| $t_1$ | $\pm 10$ pixs | $\pm 20$ pixs | $\pm 30$ pixs | $\pm 40$ pixs | $\pm 50$ pixs |
| $t_2$ | $\pm 10$ pixs | $\pm 20$ pixs | $\pm 30$ pixs | $\pm 40$ pixs | $\pm 50$ pixs |

Cuadro 1: Conjuntos de transformaciones

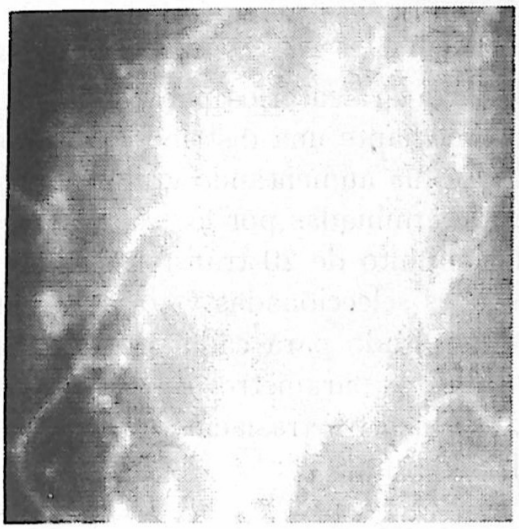


Figura 1: Angiograma

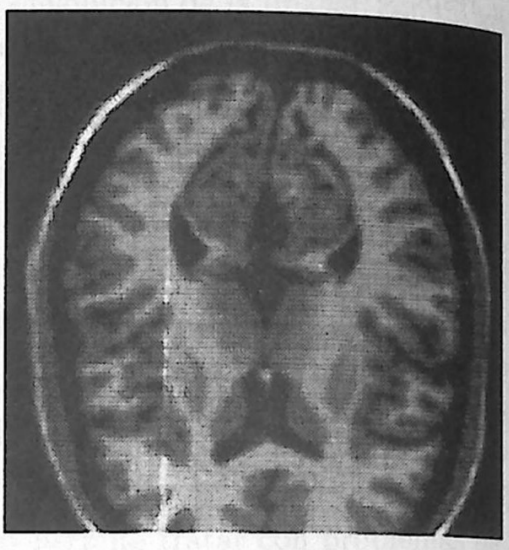


Figura 2: Cerebro (MR-T1)

permitió un máximo de 500 generaciones y el algoritmo terminaba al encontrar una diferencia de aptitud menor a 0.001 entre el mejor y el peor individuo de la población o al producir un individuo con aptitud mayor a 0.99.

Para realizar las comparaciones se utilizaron como imágenes de referencia. un angiograma con un tamaño de $\mathbf{128 \times 128}$ pixeles. y una imagen de resonancia magnética obtenida del simulador de resonancias magnéticas del Instituto Neurológico de la Universidad de Montreal [9], con un tamaño de $\mathbf{256 \times 256}$ pixeles. A partir de la aplicación de las transformaciones afines aleatorias sobre estas imágenes, se obtuvieron las imágenes objetivo.

Los resultados para ambas imágenes se muestran en las Tablas 2,3,4 y 5. En tales tablas el símbolo $\Delta$ representa el error promedio sobre alguno de los parámetros, medido sobre los problemas que fueron resueltos con éxito (se consideró un éxito cuando error para el ángulo de rotación fue menor a un grado y el error en cualquiera de las componentes de traslación fuera menor a un pixel).

Se repitieron las pruebas utilizando transformaciones de la imagen de resonancia magnética perturbada con 9 % de ruido como imagen objetivo y como imagen de referencia la misma resonancia sin ruido. El valor del ruido agregado es el máximo producido por el simulador. Los resultados se observan en las Tablas 6 y 7.

| Conj. | Algor. | $\Delta\theta$ | $\Delta e_x$ | $\Delta e_y$ | $\Delta c_x$ | $\Delta t_1$ | $\Delta t_2$ |
|---|---|---|---|---|---|---|---|
| 1ro. | G-N | 0,0021° | 0,0000 | 0,0000 | 0,0000 | 0,0022 | 0,0033 |
| | E-Ev. | 0,0412° | 0,0032 | 0,0017 | 0,0028 | 0,2163 | 0,0938 |
| 2do. | G-N | 0,0001° | 0,0000 | 0,0000 | 0,0000 | 0,0007 | 0,0001 |
| | E. Ev. | 0,0752° | 0,0020 | 0,0020 | 0,0021 | 0,1484 | 0,1006 |
| 3ro. | G-N | 0,0043° | 0,0001 | 0,0000 | 0,0002 | 0,0053 | 0,0083 |
| | E. Ev. | 0,0667° | 0,0036 | 0,0014 | 0,0045 | 0,1581 | 0,0741 |
| 4to. | G-N | 0,0004° | 0,0000 | 0,0000 | 0,0000 | 0,0009 | 0,0002 |
| | E. Ev. | 0,1732° | 0,0033 | 0,0020 | 0,0035 | 0,1264 | 0,0785 |
| 5to. | G-N | 0,0001° | 0,0000 | 0,0000 | 0,0000 | 0,0006 | 0,0001 |
| | E. Ev. | 0,0994° | 0,0030 | 0,0041 | 0,0037 | 0,1621 | 0,1828 |

Cuadro 2: Resultados obtenidos con la imagen del angiograma

| Conjunto | Algor. | Tiempo | %éxito |
|---|---|---|---|
| 1ro. Conj. | G-N | 5,9$s$ | 65 % |
| | E-Ev. | 15,2$s$ | 100 % |
| 2do. Conj. | G-N | 5,6$s$ | 75 % |
| | E-Ev. | 18,9$s$ | 90 % |
| 3ro. Conj. | G-N | 6,2$s$ | 65 % |
| | E-Ev. | 21,0$s$ | 100 % |
| 4to. Conj. | G-N | 3,8$s$ | 50 % |
| | E-Ev. | 16,0$s$ | 80 % |
| 5to. Conj. | G-N | 2,05$s$ | 25 % |
| | E-Ev. | 14,65$s$ | 70 % |

Cuadro 3: Comparación de velocidad y convergencia para la imagen del angiograma

| Conj. | Algor. | $\Delta\theta$ | $\Delta e_x$ | $\Delta e_y$ | $\Delta c_x$ | $\Delta t_1$ | $\Delta t_2$ |
|---|---|---|---|---|---|---|---|
| 1ero. | G-N | 0,0003° | 0,0000 | 0,0000 | 0,0000 | 0,0005 | 0,0005 |
| | E. Ev. | 0,0010° | 0,0000 | 0,0000 | 0,0000 | 0,0043 | 0,0035 |
| 2do. | G-N. | 0,0008° | 0,0000 | 0,0000 | 0,0000 | 0,0008 | 0,0009 |
| | E. Ev. | 0,0010° | 0,0000 | 0,0000 | 0,0000 | 0,0048 | 0,0037 |
| 3ero. | G-N | 0,0000° | 0,0000 | 0,0000 | 0,0000 | 0,0000 | 0,0000 |
| | E. Ev. | 0,0011° | 0,0000 | 0,0000 | 0,0000 | 0,0052 | 0,0041 |
| 4to. | G-N | 0,0000° | 0,0000 | 0,0000 | 0,0000 | 0,0000 | 0,0000 |
| | E. Ev. | 0,0657° | 0,0012 | 0,0010 | 0,0020 | 0,0369 | 0,0443 |
| 5to. | G-N. | 0,0000° | 0,0000 | 0,0000 | 0,0000 | 0,0000 | 0,0000 |
| | E. Ev. | 0,0018° | 0,0000 | 0,0000 | 0,0000 | 0,0080 | 0,0062 |

Cuadro 4: Resultados obtenidos con la imagen de resonancia

| Conjunto | Algor. | Tiempo | %éxito |
|---|---|---|---|
| 1ro. Conj. | G-N | 16,8$s$ | 100 % |
| | E-Ev. | 19,8$s$ | 100 % |
| 2do. Conj. | G-N | 13,75$s$ | 90 % |
| | E-Ev. | 17,9$s$ | 90 % |
| 3ro. Conj. | G-N | 11,75$s$ | 80 % |
| | E-Ev. | 16,3$s$ | 80 % |
| 4to. Conj. | G-N | 3,90$s$ | 55 % |
| | E-Ev. | 12,0$s$ | 80 % |
| 5to. Conj. | G-N | 7,4$s$ | 45 % |
| | E-Ev. | 18,6$s$ | 80 % |

Cuadro 5: Comparación de velocidad y convergencia para la imagen de resonancia

| Conj. | Algor. | $\Delta\theta$ | $\Delta e_x$ | $\Delta e_y$ | $\Delta c_x$ | $\Delta t_1$ | $\Delta t_2$ |
|---|---|---|---|---|---|---|---|
| 1ro. | G-N | 0,0095° | 0,0002 | 0,0002 | 0,0003 | 0,0360 | 0,0408 |
| | E. Ev. | 0,0794° | 0,0014 | 0,0010 | 0,0023 | 0,2250 | 0,2402 |
| 2do. | G-N. | 0,0142° | 0,0003 | 0,0003 | 0,0004 | 0,0543 | 0,0399 |
| | E. Ev. | 0,0618° | 0,0019 | 0,0010 | 0,0030 | 0,2428 | 0,1976 |
| 3ro. | G-N | 0,0095° | 0,0003 | 0,0003 | 0,0005 | 0,0562 | 0,0437 |
| | E. Ev. | 0,0588° | 0,0009 | 0,0008 | 0,0021 | 0,1279 | 0,1406 |
| 4to. | G-N | 0,0073° | 0,0001 | 0,0000 | 0,0003 | 0,0274 | 0,0191 |
| | E. Ev. | 0,0613° | 0,0007 | 0,0006 | 0,0015 | 0,0922 | 0,1252 |
| 5to. | G-N. | 0,0066° | 0,0002 | 0,0002 | 0,0008 | 0,0210 | 0,0245 |
| | E. Ev. | 0,0789° | 0,0015 | 0,0007 | 0,0026 | 0,1494 | 0,2006 |

Cuadro 6: Resultados obtenidos con la imagen de resonancia con ruido

| Conjunto | Algor. | Tiempo | %éxito |
|---|---|---|---|
| 1ro. Conj. | G-N | 49,1$s$ | 100 % |
| | E-Ev. | 16,55$s$ | 100 % |
| 2do. Conj. | G-N | 36,25$s$ | 90 % |
| | E-Ev. | 14,5$s$ | 90 % |
| 3ro. Conj. | G-N | 27,3$s$ | 75 % |
| | E-Ev. | 11,5$s$ | 70 % |
| 4to. Conj. | G-N | 21,85$s$ | 45 % |
| | E-Ev. | 10,7$s$ | 65 % |
| 5to. Conj. | G-N | 17,55$s$ | 45 % |
| | E-Ev. | 9,2$s$ | 55 % |

Cuadro 7: Comparación de velocidad y convergencia para la imagen de resonancia con ruido

# 6. Conclusiones y Trabajo Futuro

De las comparaciones realizadas podemos afirmar que el algoritmo propuesto presentó un mejor desempeño que el algoritmo tradicional; esto basado solamente en el porcentaje de éxito obtenido. Sin embargo, el algoritmo brinda una menor precisión y requiere generalmente de más tiempo para su ejecución que el algoritmo tradicional. Lo anterior nos lleva a pensar en la implementación de una metodología híbrida que combine la robustez de las estrategias evolutivas con la gran precisión y rapidez de los métodos tradicionales.

# Agradecimientos

El primer autor agradece al CONACyT el apoyo para realizar sus estudios de Maestria y continuar estudiando el Doctorado en Ciencias en el CIMAT. El segundo autor reconoce apoyo parcial del CONACyT a través del proyecto No. I-39324-A. Los dos últimos autores reconocen apoyo a través del proyecto No. 34575-A.

# Referencias

[1] Lisa Gottesfeld . Brown. A survey of image registration techniques. ACM Computing Surveys, 24(4):325–376, December 1992. http://citeseer.nj.nec.com/gottesfeldbrown92survey.html.

[2] J. B. A. Maintz and M. A. Viergever, "A survey of medical image registration," Medical Image Analysis 2(1), pp. 1–36, 1998. (Web version: http://www.oup.co.uk/mediaj/hdb/Volume 02/Issue 01/). http://citeseer.nj.nec.com/maintz98survey.html

[3] K. Voss and H. Suesse. Comunicaciones Técnicas del CIMAT, 05.06.2001, I-01-06 (CC).

[4] Héctor Fernando Gómez García. "Estratégias Evolutivas aplicadas al problema del registro de imagénes". Tésis de Maestría. Centro de Investigación en Matemáticas, Departamento de Computación. Guanajuato, Gto., Agosto de 2002.

[5] Ingo Rechenberg. Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen. Evolution. FrommannHolzboog, Stuttgart, 1973.

[6] Hans-Paul Schwefel. Evolutionsstrategie und numerische Optimierung. Dr.-Ing. Thesis, Technical. University of Berlin, Department of Process Engineering, 1975.

[7] Thomas Bäck. Evolutionary Algorithms in Theory and Practice. Oxford University Press. New York, 1996.

[8] Tony Lindeberg. Scale space theory in computer vision. Kluwer academic publishers. Netherlands, 1994.

[9] http://www.bic.mni.mcgill.ca/brainweb/

[10] C.A. Cocosco, V. Kollokian, R.K.-S. Kwan, A.C. Evans : "BrainWeb: line Interface to a 3D MRI Simulated Brain Database". NeuroImage, vol.5, no.4, part 2/4, S425, 1997. Proceedings of 3-rd International Conference on Functional Mapping of the Human Brain, Copenhagen, May 1997.

[11] R.K.-S. Kwan, A.C. Evans, G.B. Pike : "MRI simulation-based evaluation of image-processing and classification methods". IEEE Transactions on Medical Imaging. 18(11):1085-97. Nov1999.

[12] R.K.-S. Kwan, A.C. Evans, G.B. Pike : "An Extensible MRI Simulator for Post-Processing Evaluation". Visualization in Biomedical Computing (VBC'96). Lecture Notes in Computer Science. vol. 1131. Springer-Verlag, 1996. 135-140.

[13] D.L. Collins, A.P. Zijdenbos, V. Kollokian, J.G. Sled, N.J. Kabani, C.J. Holmes, A.C. Evans : "Design and Construction of a Realistic Digital Brain Phantom". IEEE Transactions on Medical Imaging, vol.17, No.3, p.463-468, June 1998.

# Mejoramiento selectivo condicional: una nueva estrategia para el mejoramiento de imágenes digitales en colores impresas en semitonos

**Edgardo Manuel Felipe Riverón* y Mario Eduardo Ramos García****

* Laboratorio de Procesamiento de Imágenes y Reconocimiento de Patrones,
Centro de Investigación en Computación, Instituto Politécnico Nacional, México, DF.
Instituto de Cibernética, Matemática y Física, CITMA, La Habana, Cuba.
** Instituto Mexicano del Petróleo.
* edgardo@cic.ipn.mx Teléfono: 5729 6000/56515
** mramos@imp.mx Teléfono: 5333 7668

## RESUMEN

En este trabajo se presenta una nueva estrategia de mejoramiento de las imágenes digitales en colores impresas en semitonos. Debido al carácter repetitivo del proceso de digitalización y de la estructura de los patrones usados para la creación para la impresión de los semitonos en imágenes en colores, se ha observado que la alteración de los colores de los pixeles, en general, es también repetitiva. Sobre la base de la frecuencia con que aparecen los colores de los pixeles en la imagen digitalizada y los valores adquiridos por sus componentes de color RGB, se han definido varias reglas heurísticas. Estas reglas, al considerar válidos los pixeles con una alta frecuencia de ocurrencia y como 'artefactos' a los que tienen pequeña frecuencia de ocurrencia, asocia los segundos a los primeros cambiándoles el color. La aplicación de esta estrategia genera imágenes notablemente mejoradas. A este método de mejoramiento le hemos llamado *mejoramiento selectivo condicional*, el cual se ha aplicado para mejorar imágenes digitales en colores de varios tipos, todas impresas en semitonos.

**Palabras clave:** Mejoramiento de Imágenes; Imágenes de Semitono; Digitalización de Imágenes en Colores; Eliminación de Artefactos.

## 1 Introducción

Durante años, las imágenes en colores han sido impresas en semitonos por capas mediante la mezcla sustractiva de colores CMY(K). Este proceso se caracteriza por tener un carácter eminentemente discreto en su composición. Para extraer automáticamente la información de interés de estas imágenes, es necesario transformarla a un formato digital. Cuando estos mapas discretos son digitalizados, por ejemplo, con un escáner de colores de cama plana convencional, aparecen numerosos 'artefactos' debidos a las alteraciones que sufren las

tres componentes de color RGB de cada píxel. El resultado neto es que los originales de muchos pixeles son apreciablemente alterados. Este fenómeno prácticamente imposible usar los colores para el reconocimiento e interpretación confia de los patrones gráficos presentes en las imágenes. Las características fundamentale imágenes digitales de mapas así obtenidas, aparecen en [1], [2], [3].

Las soluciones basadas en el uso de los filtros tradicionales para eliminar los uniforme o impulsivo en las imágenes digitalizadas, no son generalmente efectivas análisis sobre la base del color, debido a que por ser operaciones globales y "ciega aplicadas a todos los pixeles, alterados o no, provocan que los nuevos reasignados sean numerosos y muy diferentes a los colores reales del mapa original. indispensable proceso de digitalización para llevar a cabo esta transformación mostrado la Figura 1, potencialmente se originan innumerables problemas expuestos previamente [3], en imágenes de mapas impresos en semitonos digitalizados. Si los 'artefactos' eliminan, se hace muy difícil realizar la segmentación, descripción, reconocimiento interpretación confiables de los patrones gráficos presentes en las mismas, para posterior utilización, por ejemplo, en el caso de mapas, para integrarla a un Sistema Información Geográfica [4]. Un antecedente del presente trabajo aparece en [5].

Debido a la considerable magnitud que normalmente tienen las imágenes, digitalización crea también archivos de gran tamaño. Esto es realmente crítico si son colores y se les asigna color verdadero (24 bits / píxel) a cada uno de los pixeles. Con de reducir el número de colores, es común asignarle a los pixeles sólo colores de entre 216 posibles de la Paleta de Color Seguro - PCS (Ver epígrafe 4.3) [6].

El propósito que se persigue con este trabajo, es lograr una imagen final con el mínim de 'artefactos', reteniendo al máximo las estructuras básicas de los diversos componen gráficos de todo tipo presentes en ellas.

Para la comprobación de los resultados del método, las imágenes se han digitalizado un escáner de color HP PrecisionScan LTX [7] a 300 ppp (puntos por pulgada) y bits / píxel, a partir de imágenes impresas en papel en semitonos en colores obtenidas [8] y de un papel publicitario común, o simplemente con una imagen de la red.

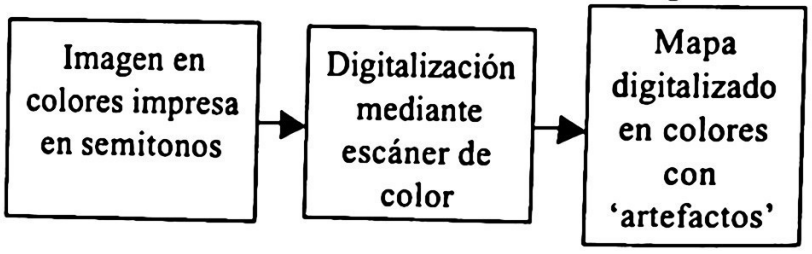
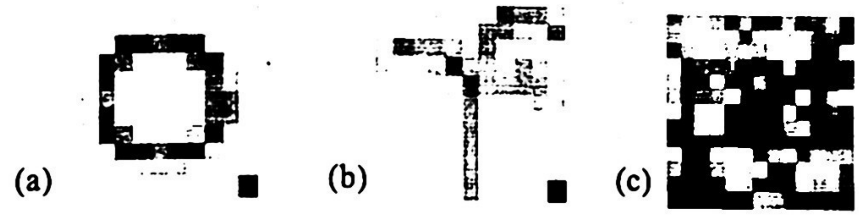



**Figura 1.** Etapas de la digitalización de imágenes en colores impresas en semitonos.

## 2 El problema

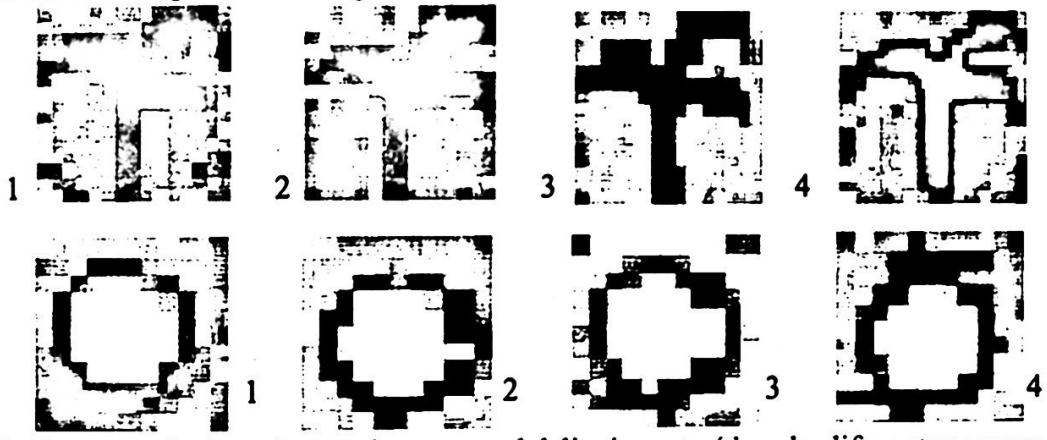
En la Figura 2 se muestran las subimágenes digitales de 11x11 pixeles ampliadas de dos patrones gráficos puntuales [1], que simbolizan una palmera de coco y un limón, tomados de la leyenda de un mapa cartográfico en colores impreso en semitonos digitalizado y una letra 'a' ubicada dentro de una subimagen de 13x13 pixeles del mismo mapa. En ellas

apreciar el efecto de los 'artefactos', que se manifiestan con los diferentes colores niveles de gris) adquiridos por los pixeles al digitalizarlo. La subimagen de la palmera de aparece con 33 colores y la del limón con 31 colores diferentes, contando el color anco del fondo. La subimagen de la letra 'a' aparece con 46 colores diferentes. El píxel de la esquina inferior derecha es de control.




**gura 2.** Subimágenes de 11x11 pixeles de dos patrones puntuales extraídos de la leyenda mapa y de un carácter alfanumérico con 13x13 pixeles. (a) Palmera de coco. (b) Limón. Letra 'a'.

Figura 3 se muestran las subimágenes de cuatro versiones de los dos tipos de mbolos gráficos puntuales anteriores, situados en diferentes zonas del mapa. Se puede ıreciar la disimilitud entre ellos y los extraídos de la leyenda mostrados en la Figura 2, ebido a los 'artefactos' que se presentan. El número total de colores diferentes de los xeles de las subimágenes de la Figura 3 se detalla en la Tabla 1.




**gura 3.** Patrones de la palmera de coco y del limón extraídos de diferentes zonas del

| Imagen | Tamaño (pixeles) | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| Palmera de coco | 15 x 15 | 74 | 73 | 79 | 70 |
| Limón | 11 x 11 | 54 | 39 | 41 | 43 |

**Tabla 1.** Número de colores diferentes de las subimágenes de la Figura 3.

## Mejoramiento de las imágenes por métodos convencionales

es de todos conocido, cualquier algoritmo común de umbralamiento, de limpieza de idos, suavizador de imágenes, o destinado para la eliminación de pulsos espúreos, es operadores con máscaras de promediación, filtros de mediana, de seudomediana y

otros, realizan su labor de una manera global y "ciega", es decir, sin tener en cuenta características contextuales y propias de aquellos pixeles particulares que no desaparecer ni transformarse en valores no deseados. Adicionalmente, alguna de soluciones comunes mencionadas provocan inconvenientes conocidos, tales pérdidas de bordes, sustitución de pixeles válidos por otros no adecuados, eliminación información de interés, ocultamiento de colores reales, empañamientos, etc., que aplicación como la nuestra, dificultarían el posterior reconocimiento e interpretación símbolos gráficos del mapa.

La solución propuesta tampoco es la de modificación de los histogramas de los planos de color sobre la base de un procesamiento local o de otro método adaptativo que tales soluciones brindarían resultados no siempre valederos en esta aplicación basarse en métodos que, aunque más asociados a la información digital, siguen siendo "ciegos", con pocas posibilidades de lograr una imagen final cercana a la originalmente fue.

La Figura 4 muestra las subimágenes de 13 x 13 pixeles de los dos símbolos ejemplificados (palmera de coco y limón) del mapa, junto con la de la letra 'a', después aplicarle un procesamiento "outlier" con máscara de 3 x 3 a la imagen digital original 'artefactos'. La palmera resultó con 116, el limón con 130 y la letra 'a' con 117 colores diferentes.

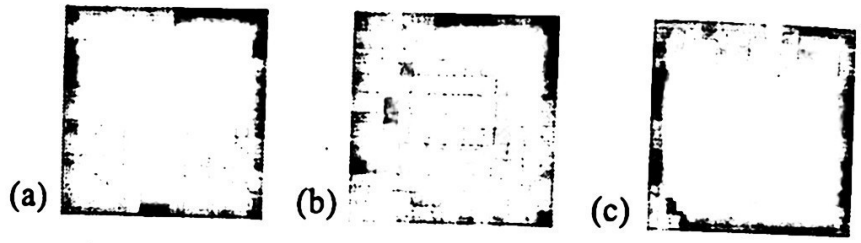



**Figura 4.** Imágenes obtenidas después de aplicar un procesamiento "outlier" al mapa digitalizado. (a) Palmera de coco. (b) Limón. (c) Letra 'a'.

La Figura 5 muestra las imágenes de 13 x 13 pixeles de ambos símbolos, después de aplicarle un filtraje de mediana con una cruz de 5 x 5 a la imagen original con 'artefactos'. La palmera resultó con 35 colores, el limón con 27 y la letra 'a' con 17 colores diferentes.

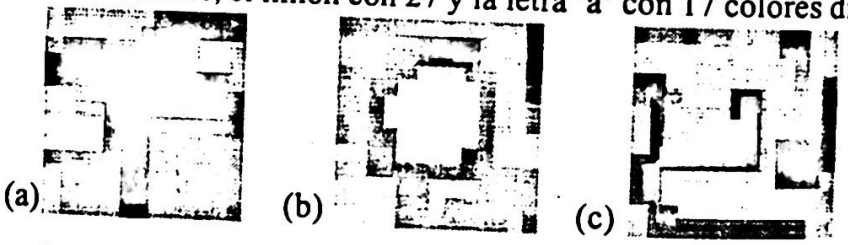



**Figura 5.** Imágenes después de aplicar un filtraje de mediana con una máscara en forma de cruz de 5 x 5 al mapa digitalizado. (a) Palmera de coco. (b) Limón. (c) Letra 'a'.

En todos los casos, podemos percatarnos de que la calidad de los símbolos ha empeorado por lo que su reconocimiento e interpretación con tal diversidad de colores, sobre esa base, no constituye una tarea fácil. De hecho, han quedado "camuflados" o simplemente han desaparecido, como el carácter 'a' de la Figura 5 (c).

## *Mejoramiento selectivo condicional* de las imágenes

### Efectos en la digitalización de los semitonos

este análisis, en el que se ha utilizado como ejemplo un punto de semitono de color (R), se han hecho las consideraciones siguientes:

colores de los puntos de semitono de la mezcla sustractiva CMY(K) utilizados en la impresión, se consideran saturados. El color rojo se ha creado por la superposición de puntos de semitono coincidentes, uno de color magenta y otro de color amarillo.

error de registro, es decir, la falta de coincidencia en la superposición de los puntos colores CMY(K), es cero.

ganancia del punto al realizarse la impresión es mínima.

color del sustrato es "blanco" y su distribución es uniforme.

ırante el proceso de digitalización ocurre lo siguiente respecto al color que adquiere cada Si el muestreo de la imagen tiene lugar dentro de un punto de un semitono saturado el píxel habrá adquirido una componente de color con el matiz de dicho punto y un dependiente de su brillantez (Figura 6). El resto de los pixeles adquirirá una mponente dependiente tanto del por ciento coincidente con el punto de semitono en estión, como del color del fondo o del punto de semitono subyacente.

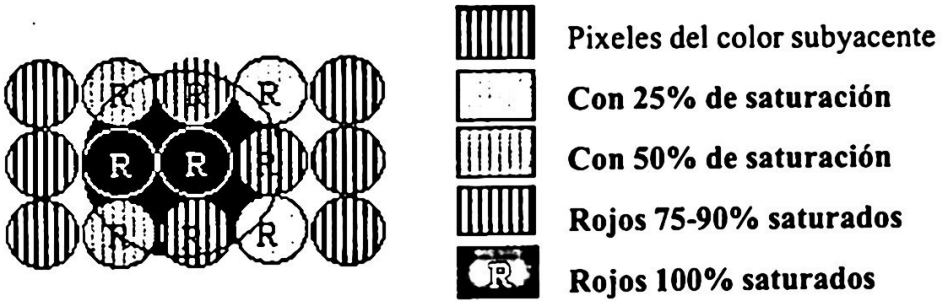



**Figura 6.** Muestreo de un punto de semitono de color rojo.

color del fondo es blanco (Figura 7), color que aporta todos los colores, el valor de las mponentes de color correspondientes del píxel en cuestión (la magenta y la amarilla) se duce, y aumenta a su vez el valor de la otra componente (la ciano). Como resultado se tendrá un color menos saturado, del mismo matiz y más brillante. A medida que el píxel menos del color del punto de semitono, su color se acercará más al blanco.

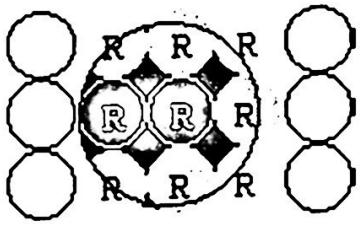


**Figura 7.** Muestreo de un punto de semitono de color rojo sobre fondo blanco.

Si el color del fondo es negro (Figura 8), el cual no contribuye en nada con también el valor de la componente en cuestión se reduce, pero sin aportarse otras componentes. El resultado es que el píxel correspondiente adquirirá un de sombra que se manifiesta como un píxel del mismo matiz, también menos saturad menos brillante. A medida que el píxel recibe menos del color del punto de semito color se acercará más al negro

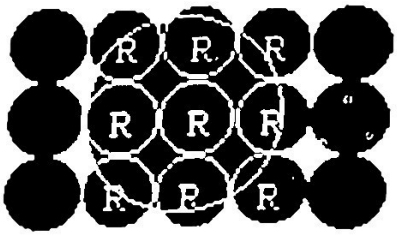



**Figura 8.** Muestreo de un punto de semitono de color rojo sobre fondo negro.

Si en lugar de tenerse fondo blanco o negro, se tiene un punto de semitono con otro y un probable realengo entre los puntos circulares cercanos, la diversidad de valore adquieren las componentes del color de los pixeles es mucho mayor. Esta es la razón p que al digitalizar una imagen impresa en semitono, aparecen pixeles de muchos diferentes, alterados, los que en definitiva son los 'artefactos' que deben eliminar medida que la frecuencia de muestreo de la imagen de semitonos aumenta, la proporció 'artefactos' respecto al total de pixeles de la imagen tiende a disminuir.

La forma en que se eliminan los pixeles, que por su frecuencia de aparición y valores particulares de sus componentes se consideran 'artefactos', es asociándolos aquellos pixeles de colores esenciales cuyas componentes cumplen las caracterís regidas por las ocho reglas heurísticas detalladas en el epígrafe 5. De esa forma, l pixeles pasan a tomar el color que desde un inicio deberían haber tenido al realizar digitalización.

## 4.2 Colores esenciales

Durante el *mejoramiento selectivo condicionado* se establecieron como colores esenci aquellos que aparecen en la leyenda del mapa que se mejora y en los fondos, en favo facilitar la decisión sobre cuáles colores de píxel deben prevalecer y cuáles deben transformados para agruparse a ellos. Estos son los colores que finalmente debe prevalecer en el mapa mejorado; el resto se considerarían los 'artefactos' que finalm debían desaparecer. En general, en el método propuesto, el usuario tendrá siempr opción de seleccionar los colores esenciales particulares del mapa digitalizado que de mejorar.

## 4.3 Paleta de Color Seguro (PCS)

La Paleta de Color Seguro PCS ("Color Safe Palette" en inglés) es una paleta que manej número limitado de colores, a saber, 216 colores, que se crean con un máximo de sól valores por componente RGB para cada píxel de la imagen [6]. Los valores de pí

reestablecidos para cada una de las componentes R, G y B de cada color de la PCS son: 0, 102, 153, 204 y 255. Los posibles colores que finalmente se producen, son obtenidos de combinación de las tres componentes con estos 6 valores para cada una.

Durante el proceso de *mejoramiento selectivo condicionado* de las imágenes, como primera lternativa para agrupar los numerosos colores que originalmente aparecieron al digitalizar mapa y según los seis niveles preestablecidos para cada una de las componentes RGB Paleta de Color Seguro, se evaluaron tres opciones: *hacia arriba*, que lleva los valores las componentes de cada píxel al nivel inmediato superior; *hacia abajo*, ídem al nivel ınmediato inferior; y *hacia el centro*, que los lleva hacia los valores inferior y superior, a de los valores intermedios de la PCS.

Finalmente se siguió la pauta de agrupar los colores hacia el centro, por ser la que rindó los mejores resultados al no alejarse mucho los colores obtenidos, de los riginalmente resultantes de la digitalización. El agrupamiento en los seis grupos para cada omponente de color RGB siguiendo esta pauta se muestra en la Tabla 2.

| Grupo de color | Rango de la componente original | Valor de la componente final |
|---|---|---|
| 1 | $0 \leq R, G, B \leq 25$ | 0 |
| 2 | $26 \leq R, G, B \leq 76$ | 51 |
| 3 | $77 \leq R, G, B \leq 127$ | 102 |
| 4 | $128 \leq R, G, B \leq 178$ | 153 |
| 5 | $179 \leq R, G, B \leq 229$ | 204 |
| 6 | $230 \leq R, G, B \leq 255$ | 255 |

**Tabla 2.** Agrupamiento de las componentes de cada color a los de la Paleta de Color Seguro (PCS).

### Principios básicos para desarrollar la heurística de mejoramiento

cuatro los principios básicos que han servido de base para el desarrollo de las reglas ieurísticas de mejoramiento propuestas:

La reducción del número final de colores de la imagen.

La consideración de que los 'artefactos' son los pixeles con colores no esenciales con menor frecuencia de ocurrencia en la imagen.

Debido a las características periódicas y repetitivas de los procesos de digitalización y de impresión en semitonos, se supuso que el proceso de mejoramiento puede ser llevado a cabo mediante estrictas reglas surgidas de la experiencia y de la observación.

Que las imágenes en colores no tienen niveles de gris, excepto el negro y el blanco.

## Reglas heurísticas del *mejoramiento selectivo condicional*

ocho reglas heurísticas creadas se exponen a continuación.

**I. Si un color Cj tiene dos componentes cualesquiera iguales y la tercera diferente.**

El color $C_j$ poco frecuente se asociará con el color $C_i$ más frecuente, si sus componentes iguales coinciden y en la diferente discrepan por exceso o por defecto unidades.

Ejemplo: A los 1254 pixeles del color con componentes R: 255, G: 102, B: 153, asociárseles en una búsqueda los 121 pixeles con componentes R: 255, G: 51, B: 153.

**II. Si un color Cj tiene sus tres componentes diferentes.**

El color $C_j$ poco frecuente se asociará con el color $C_i$ más frecuente, si cualesquiera sus componentes coinciden y en la tercera discrepan por exceso o por defecto unidades.

Ejemplo: A los 892 pixeles del color con componentes R: 153, G: 102, B: 153, asociárseles en una búsqueda los 67 pixeles con componentes R: 153, G: 51, B: 153.

**III. Si un color $C_j$ tiene dos componentes cualesquiera iguales y la tercera diferente.**

Si dos componentes de un color $C_j$ coinciden con las correspondientes de un color esencial y difieren en la tercera componente por exceso o por defecto en 51, el color esencial absorberá al color $C_j$ menos frecuente (más cercano) de entre los posibles.

Ejemplo: Los 1317 pixeles del color esencial con componentes R: 204, G: 102, B: absorberá en una búsqueda a los 2289 pixeles del color no esencial con componentes 153, G: 102, B: 102.

**IV. Independientemente de que las tres componentes sean iguales (grises) o no, todas iguales o inferiores al nivel 153, se agruparán como color negro con componente RGB iguales a 0, 0, 0.**

Ejemplo: A los 254 pixeles del color con componentes R: 0, G: 0 B: 0, asociárseles en la misma búsqueda los 12 pixeles con componentes R: 102, G: 51, B: los 420 pixeles con componentes R: 0, G: 51, B: 102.

**V. Si un color Cj tiene dos componentes cualesquiera iguales y la tercera diferente.**

El color $C_j$ poco frecuente se asociará con el color $C_i$ más frecuente, si su componente diferente coincide en ambos y en las dos iguales discrepan por exceso o por defecto unidades. (Nota: en esta regla los colores grises no absorben, pero sí pueden ser absorbidos.)

Ejemplo: A los 2544 pixeles del color con componentes R: 102, G: 153, B: 153, asociárseles en una búsqueda los 132 pixeles con componentes R: 102, G: 204, B: 204. una nueva búsqueda, los 2676 pixeles resultantes deben absorber los 211 pixeles con componentes R: 102, G: 102, B: 102.

**VI. Si un color Cj tiene cualesquiera dos componentes iguales y la tercera diferente.**

El color $C_i$ más frecuente se asociará con el color $C_j$ menos frecuente, si las componentes discrepan por exceso o por defecto en 51 unidades.

Ejemplo: A los 545 pixeles del color con componentes R: 102, G: 153, B: 153, deben asociárseles en una búsqueda los 132 pixeles con componentes R: 51, G: 102, B: 102.

**VII. Si un color $C_j$ tiene sus tres componentes diferentes.**

un color $C_j$ poco frecuente coincide en una componente con la de un color $C_i$ más frecuente, y las otras dos componentes siendo diferentes están correspondientemente invertidas, el más frecuente debe absorber al menos frecuente.

Ejemplos: A los 3245 pixeles del color con componentes R: 204, G: 102, B: 51, deben asociárseles en una búsqueda los 12 pixeles con componentes R: 204, G: 51, B: 102.

**Si un color Cj tiene cualesquiera dos componentes iguales y la tercera diferente.**

un color $C_j$ poco frecuente tiene las componentes correspondientemente invertidas de color $C_i$ más frecuente, el color más frecuente debe absorber al menos frecuente.

Ejemplo: A los 3645 pixeles del color con componentes R: 102, G: 153, B: 153, deben asociárseles en una búsqueda los 281 pixeles con componentes R: 153, G: 102, B: 102.

## Algoritmo resumido del *mejoramiento selectivo condicional*

algoritmo se resume de la forma siguiente:
**Entrada**: imagen de semitonos en colores digitalizada, en formato bmp.
**Salida**: imagen mejorada condicionalmente, en formato bmp.

{
1. **Entrada** de la imagen digital
2. **Determinación** del tamaño de la imagen
3. **Conversión** a los colores de la Paleta de Color Seguro PCS
4. **Selección** de los colores esenciales de entre los de la imagen

{
5. **Exploración** (secuencial) de los pixeles de la imagen
6. **Control** de las componentes RGB de cada píxel
7. **Ordenamiento** en la lista TO de los colores de los pixeles por su frecuencia de ocurrencia desde los más a los menos frecuentes
8. Para toda la imagen, hasta que no haya nuevos agrupamientos, realizar la Regla X, para $1 \leq X \leq 8$

{
9. **Agrupamiento** de los pixeles al ejecutarse la Regla X
10. **Reordenamiento** de la lista TO desde los más a los menos frecuentes
11. **Sustitución** el color de los pixeles alterados por los nuevos colores determinados con las reglas

}
}

Aunque el procedimiento descrito tiende a hacer pensar que no es general, las imágenes resultantes obtenidas rebaten esta apreciación. La generalidad radica en la intrínseca estructura periódica de la impresión en semitonos y en la uniformidad del muestreo, independientemente de las pequeñas alteraciones que pueden ocurrir debido a

imperfecciones de la impresión, errores de registro, calidad del escáner presencia de pliegues del papel, etc.

## 7 Resultados obtenidos

Al aplicar las ocho reglas heurísticas definidas sobre partes del mapa de imágenes, hemos obtenidos los siguientes resultados. En todos los casos, sólo ha quedado con colores esenciales.

Con el fin de comparar los resultados que se obtienen al mejorar las Figura 2, ahora de mayor tamaño, las Figuras 9-11 muestran las imágenes palmera de coco original, ahora con texto incorporado, con los colores mejorada, respectivamente. Similarmente, las Figuras 12-14 muestran ampliadas del limón original con texto incorporado. Finalmente, la Figura subimagen de 11 x 11 pixeles de la tercera letra 'a' del texto Guanajuato Original (44 colores), (b) PCS (20 colores) y (c) Mejorada (4 colores).

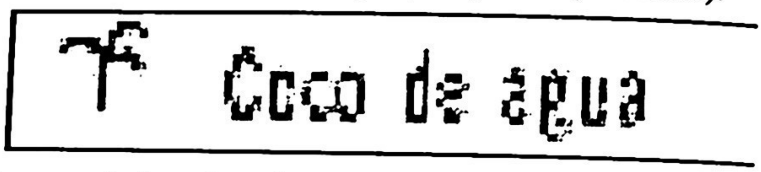



**Figura 9.** Imagen de la palma de coco original (de la leyenda). Tiene

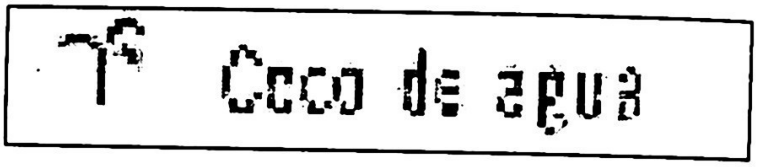



**Figura 10.** Imagen de la palma de coco con la PCS. Tiene 11 colores.

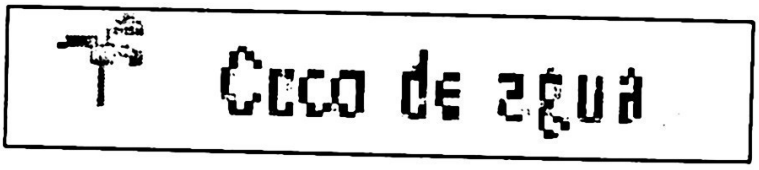



**Figura 11.** Imagen de la palma de coco mejorada. Sólo tiene 6 colores.




**Figura 12.** Imagen del limón original (de la leyenda). Tiene 58 colores.




**Figura 13.** Imagen del limón con la PCS. Tiene 21 colores.



**Figura 14.** Imagen del limón mejorada. Sólo tiene 7 colores.

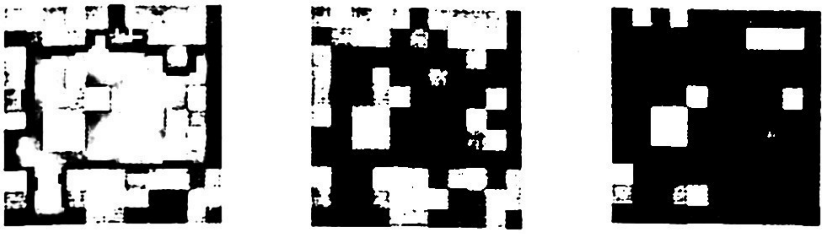


**Figura 15.** Subimagen de la tercera letra 'a' del texto Guanajuato del mapa. (a) Original (44 colores). (b) PCS (20 colores). (c) Mejorada (4 colores).

En la Figura 16 se muestran las imágenes cuadradas de 251 001 pixeles (501 pixeles de lado) de una parte del mapa de México con 8 bits/píxel para el color: 1A, original; 1B, con los colores de la PCS; y 1C, ya mejorada selectivamente.

En la Tabla 3 se muestra para cada una de las imágenes el número total de colores, el número de colores esenciales y no esenciales inicialmente, el por ciento de colores esenciales y no esenciales de cada una de las imágenes, en las tres etapas del mejoramiento selectivo condicionado. La Tabla 4 muestra el número total de pixeles de la imagen, el número de pixeles no agrupados y el % de error de agrupación, calculado sobre la base de los pixeles no agrupados respecto al total, para los colores esenciales seleccionados. En ambas tablas se destacan los resultados en las celdas sombreadas.

| Imagen → | 1A | 1B | 1C |
|---|---|---|---|
| No. total de colores | 256 | 42 | 11 |
| No. de colores esenciales | 0 | 11 | 11 |
| No. de colores no esenciales | 256 | 30 | 0 |
| % de colores no esenciales | 100 | 71.4 | 0 |
| % de colores esenciales | 0 | 28.6 | 100 |

**Tabla 3.** Número total de colores de cada imagen y por cientos de colores esenciales y no esenciales relativos a ese número.

| Imagen | 1 |
|---|---|
| No. total de pixeles | 251001 |
| No. de pixeles no agrupados | 0 |
| % de error de agrupación | 0 |

**Tabla 4.** Total de pixeles de las imágenes y por ciento de error en la agrupación.



1A 1B 1C

**Figura 16.** Imagen 1A, original. Imagen 1B, con los colores de la Paleta de Color Seguro (PCS). Imagen 1C, imagen mejorada selectivamente.

## 8 Análisis de los errores del método

El error en el mejoramiento se ha calculado sobre la base del número de pixeles con colores no agrupados, respecto al total de pixeles de la imagen, para los colores esenciales predefinidos en la operación. La Tabla 5, sobre esa base, muestra que el error para imagen mejorada es cero. El resultado visual obtenido con la imagen mejorada, no puede apreciarse aquí por lo pequeño de las imágenes y por la pérdida del color en la impresión. La evaluación general de los resultados del método puede apreciarse en [9].

## 9 Ventajas y desventajas del método de *mejoramiento selectivo condicional* propuesto

Las ventajas del método propuesto son las siguientes:

1. No se requiere el uso de costosos escáneres de tambor para la digitalización de los mapas impresos en semitonos digitalizados, excepto cuando el tamaño de la imagen lo exige.
2. No se requiere disponer de reproducciones impresas de mapas de gran calidad, ni con óptimas condiciones de conservación.
3. Se mantiene las conectividades de 4 y 8 vecinos en los contornos de los patrones gráficos puntuales, lineales y de caracteres.

No se requiere conocer a priori ninguna información relacionada con el contenido de los mapas, ni sobre la forma, tamaño o colores reales de los diversos patrones de todo tipo presentes en el mapa, salvo los que aparecen en la leyenda y que son previamente seleccionados por el usuario como esenciales.

No se hace crítica la frecuencia del muestreo con las características del contenido de los mapas, ni con el método de impresión en semitonos convencional utilizado.

Puede usarse en el mejoramiento de cualquier tipo de imagen digital, independientemente haya sido impresa en semitonos o captada con algún otro dispositivo digitalizador.

Por ser un método de agrupamiento de pixeles, puede ser utilizado en la segmentación de imágenes sobre la base del color.

No se requiere ningún mejoramiento previo de la imagen digitalizada, con los métodos clásicos "ciegos" creados para la limpieza del ruido uniforme e impulsivo.

Una vez seleccionados los adecuados colores esenciales, todo se resuelve en una sola corrida.

Por otro lado, las desventajas encontradas son las siguientes:

No es aplicable a imágenes digitalizadas con semitonos en niveles de gris.

Tiene lugar una cierta pérdida de información que surge con la primera agrupación que se hace hacia los colores de la Paleta de Color Seguro.

Por las características propias de la impresión en semitonos, la validez de los colores que se obtienen en la digitalización de algunos símbolos puntuales, lineales o caracteres alfanuméricos compuestos por trazos finos o estrechos, pierden algunos de los trazos al asociarse al color del fondo o al negro.

Puede haber cambios no decisivos en algunos de los matices de los pixeles originales.

## 0 Conclusiones

El algoritmo de mejoramiento selectivo condicional de imágenes propuesto, ha constituido una necesidad para reconocer e interpretar más fiablemente la información existente en las imágenes en colores impresas en semitonos. Su propósito ha sido, mediante la aplicación de ocho reglas de transformación halladas heurísticamente, eliminar los 'artefactos' que se presentan al alterarse eventualmente algunos colores de los pixeles durante el proceso de digitalización de los puntos que componen los semitonos.

Las ventajas fundamentales que brinda son: permite la utilización de escáneres baratos convencionales para la digitalización; mantiene los contornos de los patrones gráficos presentes; no se requiere conocer ninguna información a priori sobre su contenido; no se requiere ningún tipo de preprocesamiento; por ser un método de agrupamiento de pixeles, puede ser utilizado en la segmentación de imágenes sobre la base del color, además de mejorarse selectivamente la imagen en una sola corrida. Si una vez mejorada la imagen sólo quedan pixeles con los colores seleccionados como esenciales, el error calculado sobre la

base del número de pixeles casuales que no pudieron transformarse respecto al total de pixeles de la imagen puede considerarse cero. El número de pixeles que quedan sin transformarse, en general, depende de los colores esenciales seleccionados por el usuario.

**Referencias**


[1] S. Levachkine, Digitalización automatizada de mapas raster, Revista Digital Universitaria, Vol. 1, No. 0, UNAM, marzo 2000, http://www.revista.unam.mx/vol.0/art3/antece.html

[2] S. Levachkine, E. Polchkov, Integrated technique for automated digitization of raster maps, Revista Digital Universitaria, Vol. 1, No. 1, UNAM, julio 2000, (http://www.revista.unam.mx/vol.1/art4/index.html).

[3] E. M. Felipe, M. E. Ramos, S. Levachkine, Problemas potenciales en la digitalización automática de los mapas cartográficos en colores, Congreso Internacional de Computación CIC '2000, 15-17 de noviembre de 2000, México D.F, p. 189 –200.

[4] S. Levachkine, A. Velázquez, V. Alexandrov M. Kharinov, Semantic Analysis and Recognition of Raster-Scanned Color Cartographic Images (http://geo.cic.ipn.mx).

[5] E. M. Felipe, M. E. Ramos, Mejoramiento de imágenes de semitonos en colores digitalizadas, Memorias del VI Taller Iberoamericano en Reconocimiento de Patrones TIARP 2001, CIC-IPN, México D. F., p. 13-24.

[6] http://www.websitetips.com/color/colourchart.html (Lynda Weinman pages)

[7] HP PrecisionScan LTX. User´s Manual.

[8] Atlas NUTESA, Gráfica Internacional, S. A., ISBN 84-499-3077-4, España, 1979, 75 pág.

[9] Informe Técnico del CIC (serie azul), Edgardo M. Felipe Riverón, Mario E. Ramos García, Mejoramiento Selectivo Condicional: una nueva estrategia para el mejoramiento de imágenes digitales en colores impresas en semitonos. ISBN 970-18-8365-9.

# Segmentation of numerous tiny objects in very noisy gray-levels images

**Ingrid Castellanos Bisset [a], Edgardo M. Felipe Riverón [a,b], Leudis Sánchez Cuello [a]**

[a] Group of Digital Signal Processing
Institute of Cybernetics, Mathematics and Physics, CITMA, Havana, Cuba,
[b] Laboratory of Image Processing
Computer Research Center, National Polytechnic Institute, Mexico, DF.
edgardo@cic.ipn.mx

## ABSTRACT

This paper presents a new approach to isolate tiny objects in very noisy gray-levels images. Our objects of interest consist of blood (micro)vessels present in histological cuts of malign tumors which grow in soft parts of the human body, through a natural process known as angiogenesis. Blood vessels appear in images as small, dark objects of irregular shapes with non-uniform backgrounds. The proposed strategy is based on the application of a conditioned morphological closing, which considers some statistic properties of pixel values explored by the structuring element. The approach gives better segmentation results than those obtained by other two strategies used before and briefly analyzed in this paper.

**Keywords:** Noise cleaning; Blood vessel segmentation; Conditioned closing; Angiogenesis; statistically based structuring element.

## 1 Introduction

Noise is a common and serious cause of deterioration of digital images. When high levels of noise are present in an image, it is often nearly impossible to analyze its contents. On the other hand, when efforts are done to eliminate the noise, it is common that some important information disappears together with the noise. Then, obviously, this constitutes a serious obstacle for the future analysis of the image. Hence, eliminating the noise from images without (or with a minimum) loss of the information of interest is of fundamental importance. This task is known in digital image processing as noise filtering, noise cleaning, or noise smoothing. The lower the signal-to-noise ratio in an image, the lesser its quality.

Bad quality of images used in our particular experiments is product of a poor quality digitizer. Images obtained with such a digitizer (6 in total) are unique, and they cannot be created again. It does exist a particular interest by physicians to conserve them. For that reason, in spite of this, we attempted to enhance and segment them, in order to realize a posterior morphometrical analysis.

In this paper we present an efficient approach developed to eliminate the additive and structured noise in images having such numerous irregular tiny objects: the blood vessels. They are successfully isolated from the background. The strategy, oriented to clean the additive noise, includes the selection of the best variant from several procedures composing a sequence of four steps. The last step uses two conditioned morphological closing based on the standard deviation of gray levels in the image, which later is compared with two different numerical values $\xi$. The first closing uses a

$\xi$ value lesser than the second one. Objects obtained from the second closing act internal markers. The combination of markers with the objects resulting from the firs closing, offer finally a high quality segmented image.

## 2 Characteristics of images used in the study

Images used in our experiments represent histological cuts of malign tumors tha eventually grow in soft parts of the human body, where blood (micro)vessels are created by tumors to feed themselves. This natural process is known as *angiogenesis*. Two out of six images used in our analysis are shown in Fig. 1. They are colored, bu they were transformed to gray-level in our experiments to facilitate the hardcopy reproduction later. Blood vessels appear as small, dark objects of irregular shapes with non-uniform backgrounds. The total area of blood vessels in these pictures is an important indicator for specialists to determine the age (or level of development) this kind of tumor.

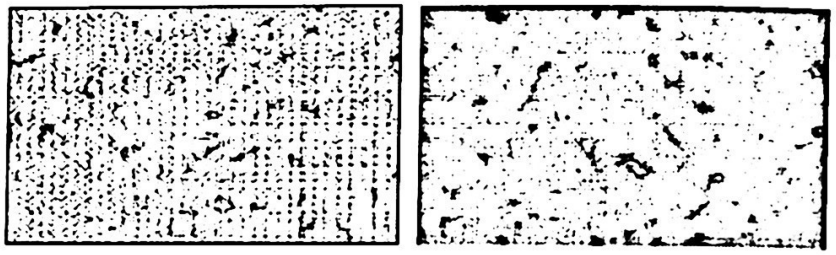


**Figure 1.** Example of two out of six images used in our procedure.

The presence of noise and the low contrast of images are caused generally by illumination deficiencies. The low quality and sensitivity of the capturing system used to create them contributed also in this sense. Eventually, in the segmentation process of gray-level images, the bad selection of the threshold level in non-elaborated algorithms of binarization produces also some artifacts, similar to tiny blood vessels present in our images. Then, it is necessary to isolate properly the useful objects, in order to calculate later the actual area of blood vessels by morphometrical analysis. In that task an idoneous system described in Rodriguez et al., 1992, is used for that purpose.

Low quality images used in our analysis present additive (uniform), multiplicative (impulsive) and structured (coherent) noises, together with some areas of higher brightness. The intensity and position of the non-uniform illumination source arriving to the mirror of the optical microscope, gives place to the central luminous spot in all images. Originally, images appeared bluish-greenish because of its artificially coloring with methyl green to highlight the vascular formations and thus distinguish more clearly the blood vessels from the background.

Images were observed through a binocular microscope with 200X magnification coupled to a color video camera. Signals from the camera were digitized and sent to the microcomputer. RGB images obtained have spatial resolution of 512 x 512 and 5 bits per pixel for color. For segmentation purposes, color images were transformed to HSI color space, where the intensity gray-level phase of images was selected. Because of the common small size of blood vessels in pictures, it was particularly

difficult to distinguish the artifacts (or eventually impulsive noise) from the actual blood vessels.

## 3 Related topics

In Digital Image Processing (DIP), elimination of different classes of noise in images is commonly required. So far, many articles have been written for this purpose with by using different segmentation methods oriented to such goals (Otsu, 1978; Pun, 1980, 1981; Johannsen-Bille, 1982; Chiralo-Berdan, 1985; Kapur et al., 1985; Wen-Hsiang, 1985). However, elimination of noise in images where the gray level, the size, and eventually the shape of objects is similar to the additive and/or multiplicative noise, have not been frequently found.

Additive noise is usually cleaned from gray-level images by means of low-pass filters and by averaging pixels values during the capture of images, as illustrated in Gonzalez-Wood, 1992, 1996. To eliminate impulsive noise, many methods have been proposed so far, amongst them, the median filter (Huang-Yang, 1978); rank-order filters (Heygster, 1982); and more recently the so-called morphological filters (Maragos-Schafer, 1990). Structured, periodical, or signal dependent noise is commonly eliminated by using the Fast Fourier Transform or other in order to transform images to frequency space, and then interactively suppress undesirable frequencies producing the structured effect. Afterwards, applying the Inverse Fast Fourier Transform (or correspondent) over the filtered image, a cleaned final image is obtained. Examples of this method are encountered in Gonzalez-Wood, 1992, 1996. Thus, in our experiments, periodical noise present in all images was eliminated by means of a narrow band-pass filter in the Fourier frequency domain.

Sometimes in the segmentation process, as a strategy, a (real or false) color image is changed to a gray-scale image and then binarized by thresholding. The main goal of this method is to obtain a binary image with objects highly differentiated with respect to the background. In this case many artifacts are produced, together with the real impossibility to guarantee good segmentation results due to the loss of useful information in the frontiers of the objects of interest. Example of this approach is given in Rodriguez et al., 1999, by using generalized morphological filters and components labeling to overcome in some measure the loss of information.

## 4 Strategies developed

In order to isolate the blood vessels in images of example, two approaches have been developed in our Group.

### 4.1. First strategy

Taking into account color models most frequently used in these purposes, the RGB model gave the best results. Amongst the three-color components resulting from this model (red, green and blue), the green component was selected as the best one according to the amount of valid information, probably due to the very particular color characteristic acquired by the histological cuts as a consequence of their preparation in the pathology laboratory. For that reason, all subsequent steps were

carried out upon the image on the green channel. Images were segmented by means of the following sequence of procedures (Rodriguez et al., 1998; Alarcon, 1998):

- Filtering the additive noise with a low-pass filter using a 3x3 square mask;
- Using a heuristic combination of a punctual and local thresholding (as in Sharon, 1986), with a modification in the thresholding value. The local method used a window size selected from the experience;
- Finally, applying the median filter.

Results of this sequence of procedures are shown in Fig. 2 for both images taken as examples (Fig. 1). Circles and arrows indicate some clearly observed errors in the segmentation process. They are detected when comparing them with images manually segmented by a physician shown in Fig. 3(b). Errors in percents for all 6 images are shown in Table 1.

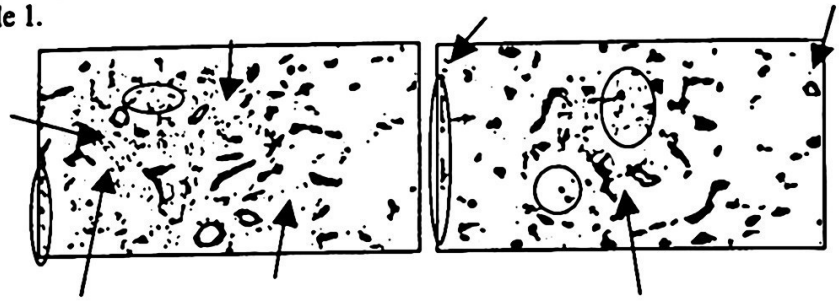


**Figure 2.** Resultant images from the first strategy.

### 4.2.Second strategy

In order to eliminate the artifacts still present in noisy images after the segmentation strategy given before (4.1), images were processed later by generalized morphological filters (GMF) and with components labeling (Rodriguez et al., 1998). Results of this procedure are shown in Fig. 3. Main differences (or errors) clearly observed after elimination of artifacts, shown in 3 (b), when compared with images segmented manually by pathologists, shown in 3 (c), are indicated with circles and arrows. Errors in percents are shown also in Table 1.

## 5 The new approach

The new strategy presented in this paper proposes to deal with the blood vessels and the background in a differentiated form. To achieve this, it was taken into account that the blood vessels in the images have lower gray-levels (darker) than the background. Therefore, the standard deviation respect to the local arithmetic mean (or another measure of central tendency) is greater than that in points of the background, considering windows size bigger than the size of blood vessels.

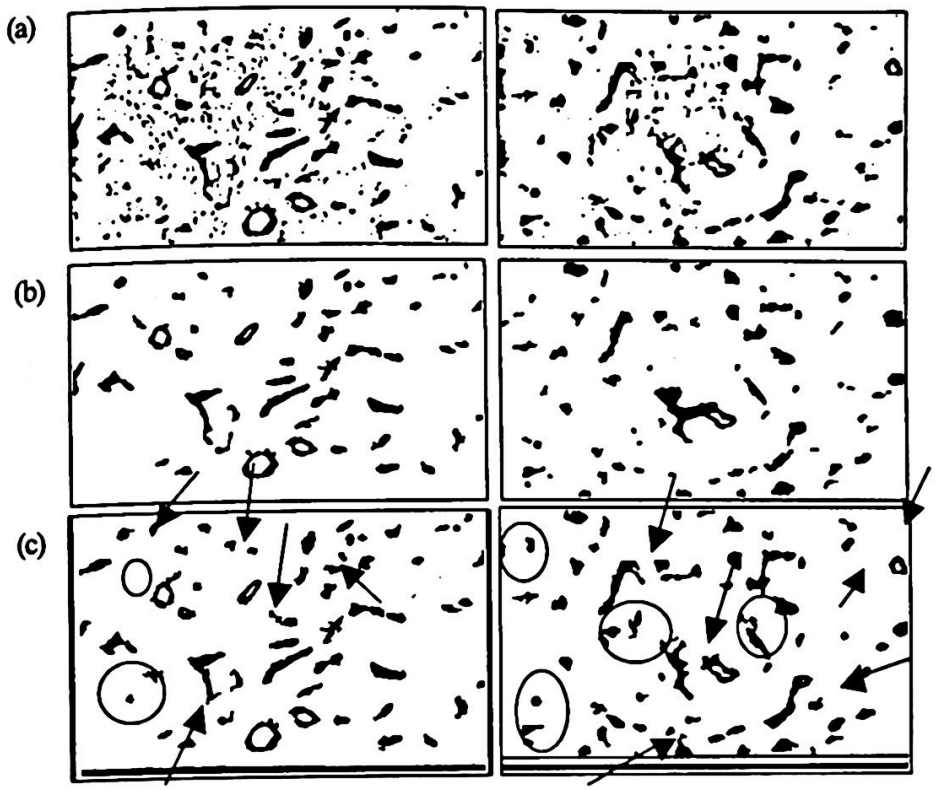



**Figure 3.** (a) Original images, (b) Images obtained by manual segmentation, (c) Results obtained by using generalized morphological operators and components labeling.

In Fig. 4 (b), the gray level distribution in an arbitrary column of an image is shown. Valleys are related to the lesser (darker) level corresponding to blood vessels in the image. Variations due to the additive noise are clearly observed. When a morphological closing is applied by the structure element shown in Fig. 4.(a) with a "height" equal to 55, the profile shown in Fig. 4 (c) is obtained. As a result, only valleys remain onto a uniform background. This result is produced, because of when the dilation (as a part of closing) is done, pixel values greater than 255 *are limited* to this value. Finally, after the erosion, those pixels acquire the same limited value equal to *(255 – height)*.

In our experiments, the height of the structure element for each image was automatically calculated on a basis of a local processing, considering pixel values within contiguous equal-size rectangular windows, and the calculation of a measure of central tendency $M$ amongst the arithmetic mean, the median and the harmonic mean. Thus, according to this, the *limit* is given by the expression *(255-M)*.

However, standard morphological filters applied globally to images, affect both the objects and the background. For that reason, new types of morphological filters were defined in our application. In such a way, dilation (or erosion) and closing (or aperture) are done if the standard deviation of pixels values covered by the structure element *(Dev)*, with respect to the measure of the selected central tendency $M$, is lesser than a given positive integer value $\xi$. Then, when processed, the *limit* considered for each window of the image was finally calculated by *(255 – M + Dev)*.

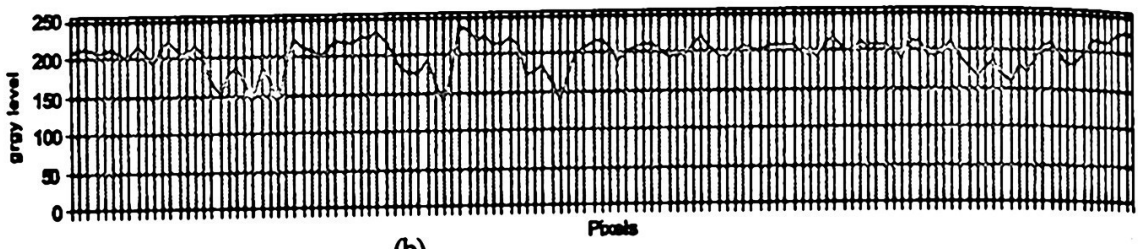
In this sense, a new morphological filter was defined, where the dilation and erosion required by the closing are done only if the standard (or mean) deviation of the pixels values covered by the structure element, with respect to the local measure of central tendency, is lesser than a given value $\xi$. This special closing we have called *conditioned closing*.

With this in mind, to achieve a homogeneous background, the *conditioned closing* was applied by a specific structure element (rhombus of 5 pixels, obtained from the experience) with gray-levels equal to *(255 – M + Dev)*, where *M* is the local measure of the central tendency measure selected, and *Dev* is either the standard (or the mean) deviation. In such a way, the resultant structure element used will have different gray level for each window.

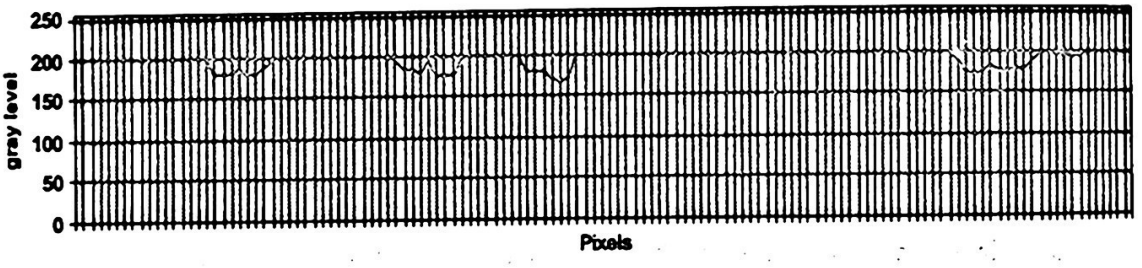
Thus, the new strategy carries out the following four procedures:

$$\begin{bmatrix} 0 & 55 & 0 \\ 55 & 55 & 55 \\ 0 & 55 & 0 \end{bmatrix}$$

(a)




(b)




(c)

**Figure 4.** (a) Structure element, (b) Pixel values of an arbitrary column of an image, (c) Result of closing with a rhomb with height 55.

- ***Application of standard morphological filters.*** Images were dealt with standard open-close, close-open, close and open filters, each independently, in order to select that with the best performance.

This step is included in order to attenuate gray-levels variations produced by the additive noise, which will influence on central tendency measures to be further calculated.

- ***Windowing.*** Images were dealt locally with two different rectangular windows: 32 x 16 and 64 x 32 pixels.

Due to the non-homogeneous background of images, it is required local processing in order to take into account particular conditions in small areas.

- ***Local calculation of different measures of central tendency.*** They were calculated locally (for each rectangular window) the arithmetic mean, the harmonic mean and the median.

This step is required to calculate the appropriate gray-levels of the structure element, locally referred to each window, and the standard deviation respect to the measure of the particular central tendency measure selected, required later by our modified *conditioned closing.*

- ***Application of conditioned closing.*** *Conditioned* morphological closing by a rhombus of 5 pixels of side and gray-level *(255 - M + Dev)* is applied *Dev* represents either the standard deviation or the mean deviation.

In this step, two values of a positive integer magnitude $\xi$ were selected from experience for comparison. Firstly, a little value of $\xi$ to obtain an image with blood vessels still together with artifacts, but in a homogeneous background. Later a greater value of $\xi$ to obtain an image without artifacts, but with internal markers in places where the blood vessels are located. By using the marker image, the blood vessels are finally isolated from the previous image. Thus, a clean (binary) segmented image has been accomplished in a straightforward form.

In our case, the best variant selected in each respective previous step was the following:

- Opening filter,
- 32 x 16 window,
- Median, as a measure of central tendency,
- Modified (*conditioned*) closing using a standard deviation, with a numerical value $\xi = 10$ to get the cleaned image, and $\xi = 30$ to get the marker image.

Images obtained though this sequence, are shown in Fig. 5 for two images of example. Fig. 6 shows resultant images obtained for six images considered.

It is well known that, due to the lack of ground truth, quantitative evaluation of a segmentation algorithm is difficult to achieve. Our alternative has been to use manual-segmentation results as ground truth, which although it is not perfect, provides a useful indication. Error are given in two metrics: false target count (FTC), that is, those background pixels that are wrongly segmented as target by the algorithm, and false non-target counts (FNTC), that is, those pixels that are in but fail to be segmented as the target. In our case, calculation of respective errors considering all images analyzed gave as result values lesser than 1,19% and 1,73%, respectively. Some differences with respect to images manually segmented are indicated in Fig. 5 with circles and arrows. Comparison of our results with the original ones shows that, qualitatively, certainty is enough high.

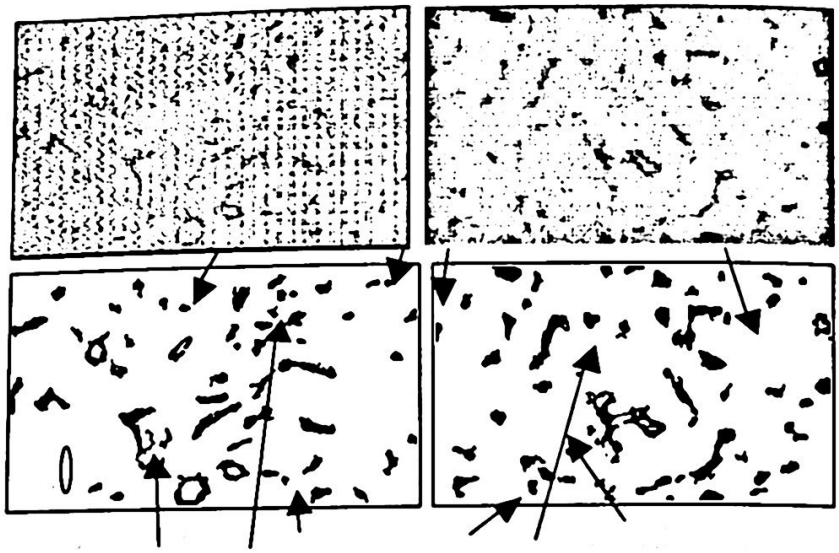


**Figure 5.** Images obtained with the new approach.

## 6 Errors analysis

Numerical evaluation of errors in percents is shown in Table 1, obtained with both previous strategies and the new approach when they are compared with images manually segmented.

| Image | 1st strategy (4.1) | | 2nd strategy (4.2) | | New approach | |
|---|---|---|---|---|---|---|
| | FTC | FNTC | FTC | FNTC | FTC | FNTC |
| 1 | 0.76 | 6.15 | 1.16 | 2.29 | 0.67 | 1.24 |
| 2 | 1.09 | 6.36 | 1.49 | 3.31 | 1.05 | 1.57 |
| 3 | 0.91 | 5.99 | 1.17 | 1.86 | 0.63 | 1.54 |
| 4 | 1.49 | 4.82 | 1.79 | 2.67 | 1.19 | 1.29 |
| 5 | 1.41 | 7.20 | 2.25 | 3.62 | 0.52 | 1.73 |
| 6 | 2.13 | 5.25 | 3.19 | 3.12 | 0.58 | 1.12 |

**Table 1.** Percents of false target count (FTC) and false non-target count (FNTC) errors, for images segmented with first and second strategies and with the approach proposed. Higher values of error are highlighted.

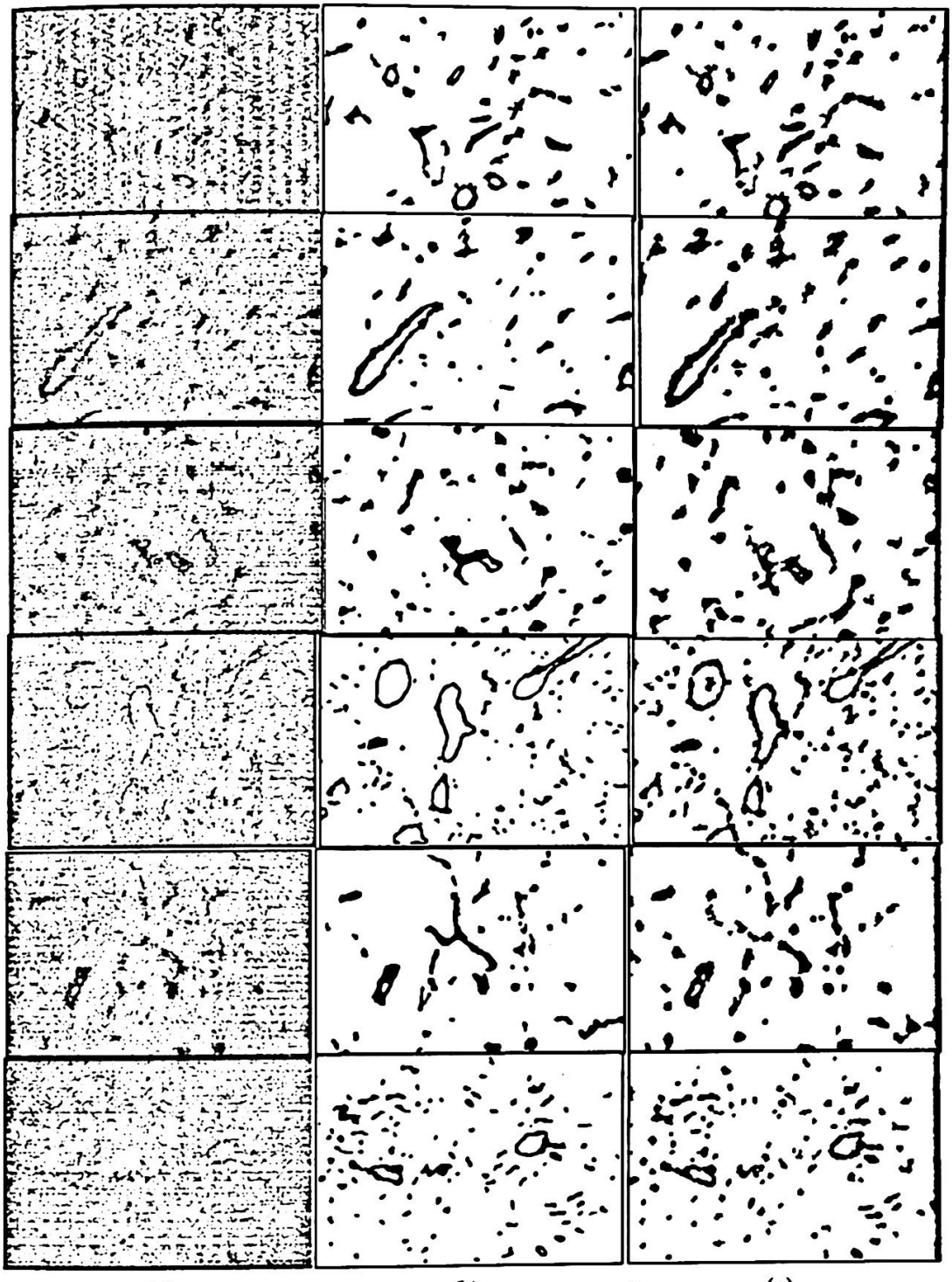



**Figure 6.** Final results of segmentation for all six images. (a) Original images, (b) Images manually segmented, (c) Images segmented with the new approach.

## 7 Conclusions

Comparing results obtained with strategies described in 4.1 and 4.2, with respect to our new approach described in epigraph 5, it is demonstrated that the new approach gives best results. Comparison is carried out with respect to the manual segmentation of blood vessels shown in Fig. 3 (b). The best variant of the sequence proposed,

guarantees the best segmentation of blood (micro)vessels present in our particular very noisy images. Errors calculated between 1,19% and 1.73% for target and non-target counts, respectively, for all images, indicate this assessment. Additionally, resultant images were obtained in a relatively more straightforward form.

## 8 References


Alarcón, T., 1998. Segmentación de imágenes en colores. Su empleo en el estudio de la angiogénesis. Tesis de Maestría en Ciencias, ISPJAE.

Chiralo, R. P., Berdan, L. L., 1985. Adaptive digital enhancement of latent fingerprints. Proc. of IEEE 20.

Gonzalez, R. C., Woods, R. E., 1992. Digital Image Processing. Addison-Wesley.

González, R. C., Woods, R. E., 1996. Tratamiento Digital de Imágenes. Addison-Wesley / Díaz Santos.

Heygster, G., 1982. Rank filters in Digital Image Processing. Computer Graphics and Image Processing 19, 148-164.

Huang, T. S., Yang, G. J., 1978. A fast two dimensional median filtering algorithms. IEEE Transactions on Acoustic, Speech, and Signal Processing, ASSP-27 (1), 13-18.

Johannsen, G., Bille, J., 1982. A threshold selection method using information measures, Proceeding of 6$^{Th}$ International Conference On Pattern Recognition, Munich, Germany, 140-143.

Kapur, J. N., Sahoo, P. K., Wong, A. K. C., 1985. A new method for gray level picture thresholding using the entropy of the histogram. Computer Graphics and Image Processing, 29, 273-277.

Maragos, P., Schafer, R. W., 1990. Morphological System for Multidimensional Signal Processing. Proceeding of the IEEE 78 (4), 690 - 708.

Otsu, N., 1978. A threshold selection method from gray level histogram. IEEE Trans. Systems, Man and Cybernetics SMC-8, 62-66.

Pun, T., 1980. A new method for gray level picture thresholding using the entropy of the histogram. Signal Processing 2, 223-237.

Pun, T., 1981. Entropic thresholding: A new Approach. Computer Graphics and Image Processing 16, 210-239.

Rodriguez, R., Alarcon, T., Castellanos, I., Wong, R., Sanchez, I., 1998. Color segmentation. Its use in the study of angiogenesis. Part I, Journal of Vascular Research 35, Sup.3.

Rodriguez, R., Castellanos, I., Alarcon, T., Navarro, R., Felipe, E., Sanchez, L., 1999. Elimination of noise via morphological filters and components labeling. Its use in the study of angiogenesis. Research Report of ICIMAF No. 99-83.

Rodríguez, R., Fernández-Britto, J. Wong, R., Campos, R., 1992. MADIP, Morfometría Analítica y Digitalización en Patología. Revista Cubana de Investigaciones Biomédicas, 11(2), 126 – 128.

Sharon, A. S., 1986. ANGY: A rule-based Expert System for Automatic Segmentation of Coronary Vessels From Digital Subtracted Angiograms. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI 8 (2), 188 - 199.

Wen-Hsiang, T., 1985. Moment-Preserving Thresholding: A new approach. Computer Graphics and Image Processing, 29, 377-393.

# INTELLIGENT SEGMENTATION IN THE TASK OF RASTER-TO-VECTOR CONVERSION OF COLOR CARTOGRAPHIC IMAGES

Serguei Levachkine***, Aurelio Velázquez, Miguel Torres, Marco Moreno
Laboratory of Geoprocessing
Centre for Computing Research (CIC),
National Polytechnic Institute (IPN),
UPALM, Edificio-CIC,
07738 Mexico City, Mexico
{palych, avela, mtorres, marcomoreno}@cic.ipn.mx

**Abstract**

We present an approach to a raster-to-vector conversion of color cartographic images. This is simultaneous segmentation-recognition-interpretation system when successfully segmented objects of interest (alphanumeric, punctual, linear, and area) are labeled by the system in the same, but are different for each type of objects, gray-level values. We call it the composite image technique. We exchange the source image by a number of composite image representations. Every composite image representation is associated with certain feature. Some of the composite images that contain the objects of interest serve to be used in the following object detection-recognition-interpretation. The specification of features associated with perspective composite representations is regarded as a type of knowledge domain. The results of gray-level color image conversion are presented.

**Key words:** Raster-to-Vector Conversion, Image Segmentation and Representation, Image Recognition, Image Segments Hierarchy, Image Synthesis

*** Corresponding author

# 1. Introduction.

Segmentation is fundamental to the field of image processing because it is used to provide the basic representation on which understanding algorithms operate. This is especially true for the task of raster-to-vector conversion systems. The ability to build up a representation from individual pixels of a scanned image, which exploits relationships such as local proximity and highlights the structures of the underlying components, is important for the extraction of features during interpretation and recognition [1]. In general, the nature of this representation is application dependent. In the present work, we developed an application independent segmentation.

Up to the now a great variety of segmentation algorithms for gray-level images has been proposed. The majority of color segmentation approaches is based on monochrome segmentation approaches operating in different color spaces [8]. Gray level segmentation methods can be directly applied to each component of a color space; thus, the results can be combined in some way to obtain a final segmentation result. However, one of the problems is *how to employ the color information as a whole for each pixel.* When the color is projected onto three *RGB* color components, the color information is so scattered that the color image becomes simply multispectral image and the color information that humans can perceive is lost [13]. Another problem is *how to choose the color representation for segmentation* [2,8]. There is no single color representation that can surpass others for segmenting all kinds of color images. The use of nonlinear spaces, such as *HSI* and the normalized color space can solve the problem to certain approximation. However, the nonlinear spaces have essential, non-removable singularities and there are spurious modes in the distribution of values [14].

An alternative solution presented in this work is invariant image representation (composite images, or simply *composites*) that does not depend on choice of particular color space. The processing of a color image is individual segmentation by each color component into image meaningful (or invariant with respect to a given, unnecessary color feature) regions, first and, then – image's integrated segmentation-recognition (or "objects of interest designing"). Moreover, the prescribed set of features is regarded as a type of *knowledge domain.*

The *composite image technique* includes object-fitting compact hierarchical segmentation, binarization of segmented images, and synthesis of binary representations. The main goal of image synthesis consists of the object linking by its associated names.

In the following sections, we build up composite image representations based on object-fitting compact hierarchical segmentation [3-6].

## 2. Object-fitting compact hierarchical segmentation.

Object-fitting compact hierarchical segmentation [3,4] is a sequence of embedded partitions without repetition of composed segments in different partitions (Fig. 1).

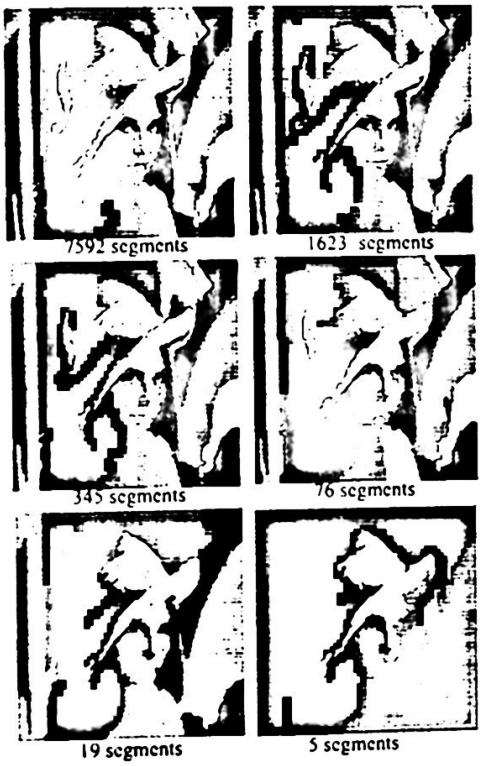



Fig. 1. Compact hierarchical image segmentation

A partition is obtained by iterative segment splitting or merging. In the merging mode, any segment in each iteration merges into the nearest adjacent segment. Number $2^i$ bound total number of segments, where $i$ is the number of iteration.

Number of segments $N$ has to be taken into account for automatic color image analysis. Indeed, the merging of segments into objects defines the *image semantics*.

This number decreases approximately as $(4 \div 5)^{-i}$, where $i$ is the iteration number (Fig. 2). From our point of view, deviation from this exponential dependence leads to semantics violation. Disclosed regularity can be useful for automatic analysis of gray-scale and color images.

Our experiments have shown (Fig. 2a) that the compressed volume of an image is decreased in the same exponential mode (image compression with information-lossless standard algorithms: RAR, LZH, etc.). Eliminating the dependence on iteration number $i$, we can obtain the exponent-mode relation of compressed data volume $V$ on number of segments $N$ as follows:

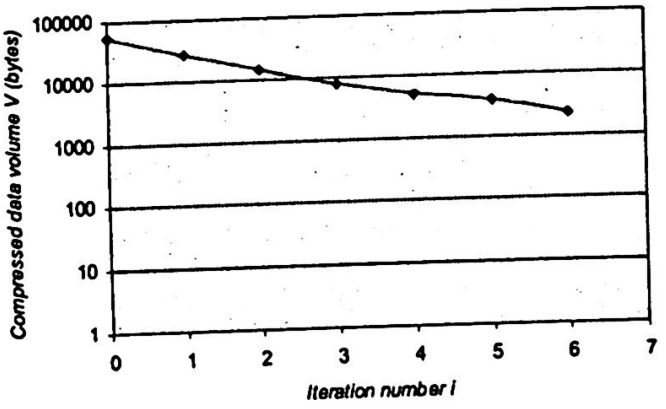



Fig. 2a. Linear dependence of compressed image volume on iteration number $i$ (in experiments carried out by A. Zaitseva (SPIIRAS))

$$N/N_0 = (V/V_0)^{\alpha}$$

Where $N_0$, $V_0$ denote number of segments and compressed volume of the source image respectively; $\alpha$ is some real coefficient.

We obtained that in the case of object-fitting compact hierarchical segmentation (Fig. 1) the exponent $\alpha$ is approximately *2.9*. Note that for non-adaptive pyramid segmentation [2], $\alpha$ is approximately *1.4*.

It is known that the volume of compressed data is closely related to the *amount of information into data*. Thus, a theoretical explanation of the obtained experimental dependencies (Figs. 2 and 2a) represents an interesting research topic of Pattern Recognition.

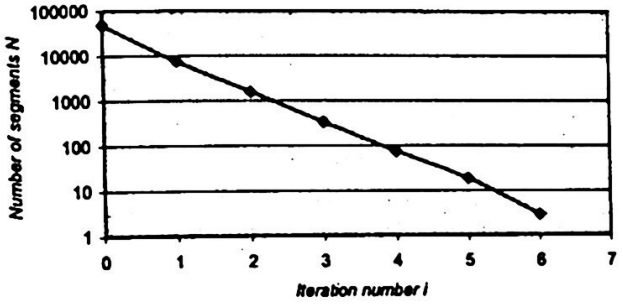



Fig. 2. Linear dependence of number of segments $N$ on iteration number $i$ (in experiments carried out by H. Rumbo (CIC))

# 3. Composite image representation.

We have found that in addition to natural decomposition (e.g., $R$, $G$ or $B$ – component splitting) of color images, artificial representations can also be useful for objects of interest detection-recognition [5,6]. Our approach provides composite representations of the source image by means of reduced number of color or tone components and segments. Composite image representation is a sequence of binary representations, which are packed into different bit planes. These binary images are the result of two-valued classification of source image by some feature (intensity, area, invariant moments, etc.).

A bit component of composite image (Fig. 3) computes by means of thresholding of the current segmented image. The threshold is equal to the average all over image of intensity, geometric or otherwise (features), e.g., $I^{(i)}$. To threshold the image, $I^{(i)}$ is compared with its average over pixels of each segment, e.g., $I^{(s)}$ as follows: $I^{(i)} \geq (<) \xi I^{(s)}$, where $\xi$ is a tuning parameter.

By substitution of intensity feature with any other feature (pixel number, segment linear dimension, orientation angle, etc.; see Section 6), we can obtain another composite representations which are useful for detection of polygons, contour, noises, etc. Examples of composite image representations are shown in Fig. 3a.

This number decreases approximately as $(4 \div 5)^{-i}$, where $i$ is the iteration number (Fig. 2). From our point of view, deviation from this exponential dependence leads to semantics violation. Disclosed regularity can be useful for automatic analysis of gray-scale and color images.

Our experiments have shown (Fig. 2a) that the compressed volume of an image is decreased in the same exponential mode (image compression with information-lossless standard algorithms: RAR, LZH, etc.). Eliminating the dependence on iteration number $i$, we can obtain the exponent-mode relation of compressed data volume $V$ on number of segments $N$ as follows:

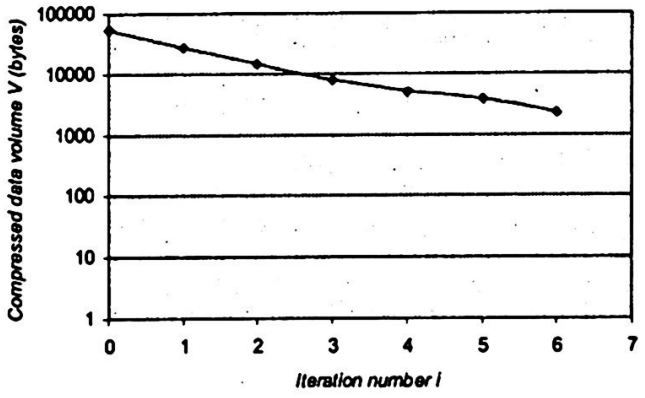



Fig. 2a. Linear dependence of compressed image volume on iteration number $i$ (in experiments carried out by A. Zaitseva (SPIIRAS))

$$N/N_0 = (V/V_0)^{\alpha}$$

Where $N_0$, $V_0$ denote number of segments and compressed volume of the source image respectively; $\alpha$ is some real coefficient.

We obtained that in the case of object-fitting compact hierarchical segmentation (Fig. 1) the exponent $\alpha$ is approximately *2.9*. Note that for non-adaptive pyramid segmentation [2], $\alpha$ is approximately *1.4*.

It is known that the volume of compressed data is closely related to the *amount of information into data*. Thus, a theoretical explanation of the obtained experimental dependencies (Figs. 2 and 2a) represents an interesting research topic of Pattern Recognition.

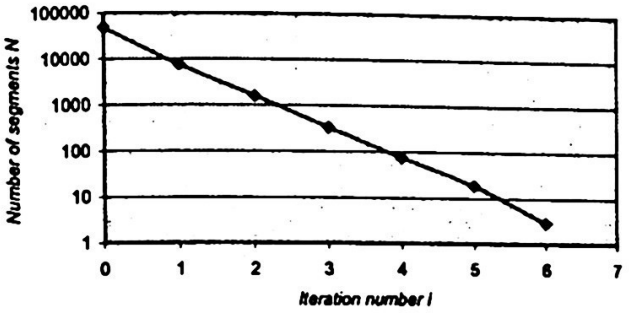



Fig. 2. Linear dependence of number of segments $N$ on iteration number $i$ (in experiments carried out by H. Rumbo (CIC))

# 3. Composite image representation.

We have found that in addition to natural decomposition (e.g., $R$, $G$ or $B$ – component splitting) of color images, artificial representations can also be useful for objects of interest detection-recognition [5,6]. Our approach provides composite representations of the source image by means of reduced number of color or tone components and segments. Composite image representation is a sequence of binary representations, which are packed into different bit planes. These binary images are the result of two-valued classification of source image by some feature (intensity, area, invariant moments, etc.).

A bit component of composite image (Fig. 3) computes by means of thresholding of the current segmented image. The threshold is equal to the average all over image of intensity, geometric or otherwise (features), e.g., $I^{(i)}$. To threshold the image, $I^{(i)}$ is compared with its average over pixels of each segment, e.g., $I^{(s)}$ as follows: $I^{(i)} \geq (<) \xi I^{(s)}$, where $\xi$ is a tuning parameter.

By substitution of intensity feature with any other feature (pixel number, segment linear dimension, orientation angle, etc.; see Section 6), we can obtain another composite representations which are useful for detection of polygons, contour, noises, etc. Examples of composite image representations are shown in Fig. 3a.



Fig. 3. Bit components of composite images obtained by means of thresholding of Lena's source and segmented images



Fig. 3a. Composite image representations

The first representation of Fig. 3a is obtained from the Lena source image by means of intensity thresholding. As a rule, sharpness is achieved for this composite image. Other composite images of Fig. 3a are obtained with different values of the tuning parameter $\xi$ from the images of Fig. 1. To compose these images, we also used the geometric features from the feature set (Section 6) in addition to intensity feature.

The bit components are packed in the resulting representation, where the extrema of intensity indicate the pixels associated with unchanged binary feature.

Essentially, the composite images form a "book" in which the objects of interest can be found on appropriated page(s). Thus, a "page number" defines the method of thresholding and the tuning parameter $\xi$.

# 4. Color composites.

Compact hierarchical segmentation of a color image is performed by each independent color components (*R*, *G*, and *B*) considering these as semi-tone images. In this way, coinciding intensities of resulting *R*, *G*, and *B* composite images indicate the segments of equal color with respect to using feature. This can be used for invariant color image description.

As a rule, compact hierarchical image segmentation implies that color segments are enlarged simultaneously in accordance to regularities presented in Section 2. Due to the self-consistence of *RGB*-segmentation behavior, visual quality improvement in composite intensities becomes available (Fig. 4).

The method requires significant operative memory space. To overcome this disadvantage, we used special data organization in the form of irregular dynamic trees [3,4], that provides optimal in memory space computing for the successive scanning of image scales. Due to data organization, a practical use of our program package does not require further algorithmic development. The user needs only to make adequate choice to carry out task features from prescribed feature set (the *knowledge domain*).

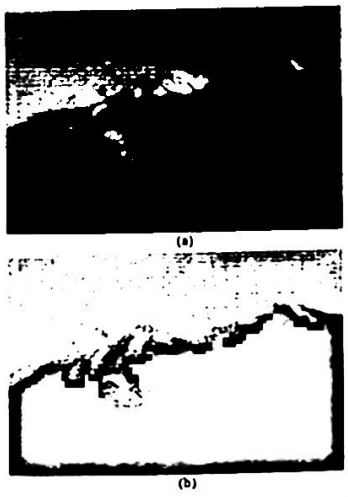


Fig. 4. Image enhancement with composites: (a) source and (b) composite image

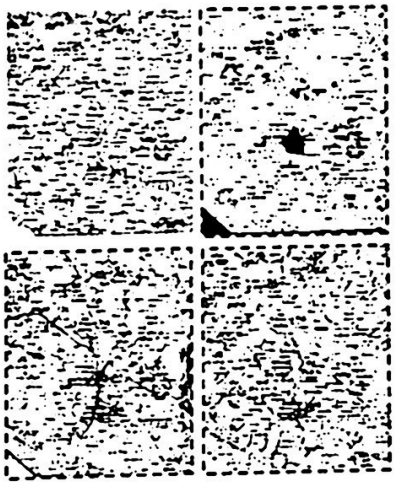


Fig. 5. Cartographic pattern retrieval from a color map image 1082x1406 pixels (upper left corner)

## 5. Applications of composites.

Image two-valued classification (binarization) is one of the most important tasks in modern recognition methods. In the frameworks of the composite image technique, we obtained a few solutions for this task [3-6]. By applying composites, we are able to extract cartographic data using *R*, *G*, and *B*-components of full-size color raster-scanned image (Fig. 5).

Fig. 6 shows how our method insures object detection in the task of recognition of inclined digits embedded in graphics. This illustrates that each composite image contains machine-treatable bit-planes for target object detection and also purposeless bit-planes. Indeed, to effectively recognize the objects of interest, it is better to search these objects on appropriated bit-planes.

Although, our system can generate some errors in interpretation, it is much more useful for the following understanding algorithms because its output is nearly recognized objects of interest.

Owing to use the geometric features in addition to intensity feature, our method detects hidden for other methods image "technological history", i.e., how the image was drawn and printed (Fig. 7).

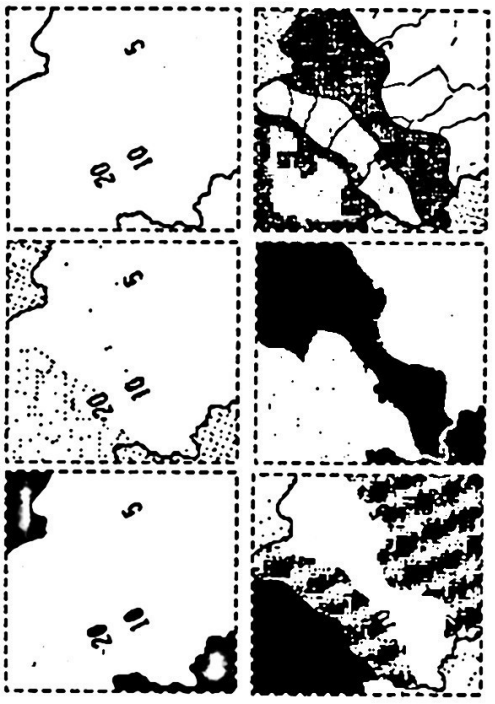


Fig. 6. Bit-planes suitable for digit recognition (left aligned) and other purposes (right aligned)

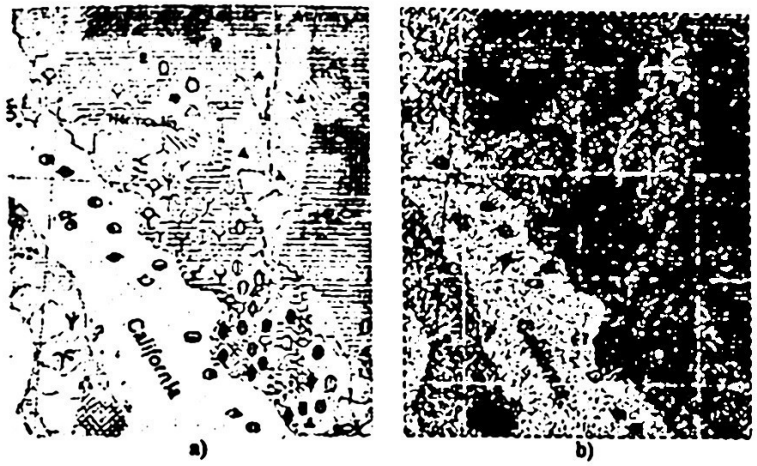



Fig. 7. Hidden network pattern reveal: (a) source and (b) composite image

# 6. Comments.

In previous sections, the details of the composite image technique have been described for very first time. Presented approach exploits the user's experience providing the knowledge domain in the form of the prescribed feature-attribute set. This set contains a number of attributes and numerous features. The attributes are a primary set of segment characteristics estimated and dynamically stored for all image segments at any level of the composite image representation. This provides a full-value use of object-fitting hierarchical segmentation. The features are numerical segment characteristics, which are obtained as output of data conversion, and are selected in function of the processing stage and the problem context. Thus, prescribed segment attributes are the following:

1. Extrema of numerical characteristics
   - Global (for the whole image);
   - Local (for a neighborhood of the segment).
2. Additive
   - Integral intensity (the sum of pixel intensities);
   - Number of pixels;
   - Integral first and second moments computed with respect to the origin.
3. Non-additive perimeter.
4. Description of the adjacent segments in terms of binary relationships.

These data provide an estimation of the intensity and geometric segment features: pixel intensity range, average intensity, invariant moments, parameters of linear sizes and shapes, etc. In this manner, the features used in generation of object-fitting hierarchy of the segments can be different from the features used in object recognition [3,4]. As a rule, at first step of image treatment, only intensity features are useful, because the source image pixels do not form geometrically meaningful segments and objects. Consequently, up to reaching image invariant representation other features are used for object designing and recognition. Moreover, the image composite representation with respect to a given feature means the recognition of "corresponding" to this feature, objects of interest.

The method has numerously been tested on complex raster-scanned color cartographic and semi-tone images rendering good results [6].

To our knowledge, this is one of the first attempts to design a segmentation-recognition computer system for complex color images of arbitrary type [7-10].

# 7. Conclusions.

The problem of *how and to what degree the semantic information should be employed in image segmentation* has led us to novel conception of composite image representation for mutual object detection-recognition at low level processing. We conjecture that modern segmentation systems must support mutual object detection-recognition-interpretation, starting at low-level, memorizing results at the intermediate level, and effectively communicating these results to the high level. The approach proceeding from this conjecture is called *composite image technique*. The idea is to prepare the source image as much as possible for subsequent high- level processing of image regions. In most of the existing color image segmentation approaches, definition of a region is based on similarity of color. This assumption often makes it difficult for any algorithms to separate the objects with highlights, shadows, shadings or texture, which cause inhomogeneity of colors of the object's surface. Using *HSI* can solve this problem to some extent, except that hue is unstable at low saturation. Some physics-based models have been proposed to solve this problem [8,9]. These models take into account the color formation, but they have too many restrictions, which prevent them from being extensively applied. Homogram thresholding and fuzzy logic algorithms applied to the problem have also exhibited limitations [11,12].

We saw an alternative solution of the problem defining image regions by additional *quantitative, qualitative, and nominal features* (in addition to

color feature), which on the whole render the *user's knowledge domain.* We think that this is a kind of advanced simulation of the human's visual perception. However, it is necessary to emphasize that for optimization of labor-intensive program training a strong formalization of composite image technique is now required. We are under way to solve this problem.

In our experiments, we used between others, complex raster-scanned color cartographic images, which being intermediate between drawings and natural images provide a nearly ideal model for testing because they have characteristics of both [6]. We obtained promising results in this *subject domain.*

At the same time, automatic interpretation of color images presents certain difficulties for state-of-the-art in image processing and also artificial intelligence. To date, it appears unrealistic to obtain fully automatic computer-based interpretation system free of errors [8,10].

We believe that only integrated approach to the problem can be fruitful. In the context of the present work, this means first, decomposition of source image by multiple hierarchical components to achieve a stable, accurate representation in the presence of degraded images. Second is the segmentation with mutual recognition of appropriate primitives. Finally, there is the development of a unified knowledge-based trainable and self-trainable system with optimal human-machine interaction for color image treatment. Our future research will be concerned with this approach.

## ACKNOWLEDGEMENTS

The authors of this paper wish to thank to the Centre for Computing Research (CIC-IPN), General Coordination of Postgraduate Study and Research (CGEPI), National Polytechnic Institute (IPN) and the Mexican National Council for Science and Technology (CONACYT) for their supports.

## REFERENCES

[1] D.S. Doermann, An introduction to vectorization and segmentation, *Lecture Notes in Computer Science, Vol. 1389*, 1998, 1-8

[2] V.V. Alexandrov & N.D. Gorsky, *Image representation and processing. A recursive approach* (Mathematics and Its Applications, *Vol. 261.* Kluwer Academic Publishers, Dordrecht/Boston/London, 1993)

[3] M. Kharinov & M. Nesterov, Intelligent program for automatic image recognition based on compact object-fitting hierarchical

image representation in terms of dynamic irregular ramified trees, *Proc. International Academy for Informatics, Communications and Management*, St. Petersburg, Russia, Special Issue 12-C 1997, 1-35 (Library of Congress Number: 98646239)

[4] V. Alexandrov, M. Kharinov & S. Levachkine, Conception of hierarchical dynamic structure in application to audio and video data recognition, *Proc. IASTED International Conference on Artificial Intelligence and Soft Computing, ASC 2001*, Cancun, Mexico, 2001, 348-353

[5] S. Levachkine, A. Velázquez & V. Alexandrov, Color image segmentation using false colors and its applications to geo-image treatment: Alphanumeric characters recognition, *Proc. IEEE International Geosciences and Remote Sensing Symposium, IGARSS 2001*, Sydney, Australia, 2001 (IEEE Catalog Number (CD-ROM): 01CH37217C; Library of Congress Number: 01-087978; ISBN CD-ROM: 0-7803-7033-3)

[6] S. Levachkine, A. Velázquez, V. Alexandrov & M. Kharinov, Semantic analysis and recognition of raster-scanned color cartographic images, *Proc. IAPR Int. Workshop on Graphics Recognition, GREC 2001*, Kingston, Ontario, Canada, 2001, 255-266

[7] K. Sunil Kumar & U.B. Desai, Joint segmentation and image interpretation, *Pattern Recognition*, *32*(4), 1999, 577-589

[8] H.D. Cheng, X.H. Jiang, Y. Sun & Jingli Wang, Color image segmentation: advances and prospects, *Pattern Recognition*, *34*(12), 2001, 2259-2281

[9] S. Levachkine & J.H. Sossa, Image segmentation as an optimization problem, *Computation and Systems*, *3*(4), 2000, 245-263 (ISSN 1405-5546)

[10] Definiens Imaging GmbH, e-Cognition: Object Oriented Image Analysis, http://www.definiens-imaging.com/ecognition/

[11] H.D. Cheng, X.H. Jiang & J. Wang, Color image segmentation based on homogram thresholding and region merging, *Pattern Recognition, 35*(2), 2002, 373-393

[12] T. Q. Chen & Y. Lu, Color image segmentation – an innovative approach, *Pattern Recognition*, *35*(2), 2002, 395-405

[13] R.C. Gonzalez & R.E. Woods, *Digital Image Processing* (3rd edn. PTR, NJ: Prentice-Hall, 2002)

[14] J. Kender, Saturation, hue, and normalized color: calculation, digitization effects, and use, *Computer Science Technical Report, Carnegie Mellon University*, 1976

# Filtrado en Tiempo Real Basado en Estimadores Robustos

Francisco J. Gallegos-Funes[1], Volodymyr I. Ponomaryov[1], Oleksiy B. Pogrebnyak[2], Luis Niño de Rivera[1]

Instituto Politécnico Nacional
[1]Escuela Superior de Ingeniería Mecánica y Eléctrica
Av. Santa Ana 1000, Col. San Francisco Culhuacan, 04430, México D.F., México
[2]Centro de Investigación en Computación
Av. Juan de Dios Batiz s/n, Col. Lindavista, C.P. 07738, México D.F., México
Tel/Fax (52-55)5656-2058, Email: fcogf@hotmail.com

**Resumen**

En este articulo se presentan las capacidades y características en tiempo real de un filtro robusto para suprimir el ruido impulsivo en aplicaciones de procesamiento de imágenes. La implementación en tiempo real del filtro propuesto fue realizada en el DSP TMS320C6701. Extensivas simulaciones en imágenes conocidas demostraron que desempeño del filtro propuesto es mejor que otros filtros no lineales tanto en supresión de ruido y preservación de detalles. Finalmente, simulamos secuencias de video corrompidas por ruido impulsivo para demostrar que el método propuesto potencialmente provee una solución en tiempo real para mejorar la calidad de las transmisiones de televisión y video.

**Palabras clave:** Procesamiento de imágenes, DSP TMS320C6701.

## 1. Introducción.

Diferentes clases de filtros han sido propuestas para remover el ruido en las imágenes [1-3]. Los filtros son clasificados en diferentes categorías dependiendo de aplicaciones especificas. Los filtros lineales son eficientes para remover el ruido Gaussiano pero tienen un pobre desempeño ante el ruido impulsivo. Los filtros no lineales son diseñados para suprimir ruido de diferente naturaleza, pueden remover el ruido impulsivo y garantizan la preservación de detalles. Los filtros basados en el orden de rangos han recibido considerable atención debido a sus excelentes propiedades de supresión de ruido y preservación de detalles [1, 2]. Ejemplos de filtros basados en el orden de rangos son los bien conocidos *mediano*, KNN (*K-nearest neighbor*), filtro *L*, diferentes filtros de switcheo, filtros neuronales, etc. Los filtros *mediano multietapa*, CWM (*center weighted median*), WM (*weighted median*), WOS (*weighted order statistics*), *stack*, *permutación* y RCRS (*rank conditioned rank selection*) han sido utilizados como filtros suavizantes en aplicaciones de restauración donde una imagen es corrompida por ruido [1-4]. Los filtros CS (*comparison and selection*), LUM (*lower-upper-middle*) son propuestos para mejoramiento de contornos [2, 4].

En este articulo, presentamos un esquema robusto de filtrado para aplicaciones en procesamiento de imágenes. Nosotros introducimos el filtro MM-KNN que esta basado en estimadores robustos combinados $R$ y $M$ para calcular la estimación robusta de los pixeles encerrados en la ventana de filtrado [5, 6]. El filtro propuesto usa el valor del pixel central en la ventana de filtrado para proveer la preservación de detalles y el estimador $M$ combinado con el estimador de la *mediana* proveen suficiente supresión de ruido impulsivo. Varios tipos de funciones de influencia [1, 7] en el estimador $M$ son usados para proveer una mejor supresión de ruido impulsivo.

El resto de este articulo es organizado como sigue: Formulamos el esquema del filtrado propuesto. Resultados experimentales del filtrado de imágenes conocidas mediante el uso de nuestro método y otros métodos basados en la mediana son presentados. Se muestra el tiempo de procesamiento de nuestro método y de varios filtros utilizados como comparativo mediante el uso del DSP de punto flotante TMS320C6701. Finalmente, presentamos algunos resultados experimentales en secuencias de vídeo usando nuestro algoritmo, y mostramos nuestras conclusiones.

## 2. Formulación e Implementación.

Huber propuso los estimadores $M$ como una generalización de los estimadores de máximo de verosimilitud (MLE) [2, 4]. Su definición es dada por una función $\rho$ $\{\rho(X) = \ln(F(X))\}$ conectada con la función de densidad de probabilidad $F(X)$ de una muestra de datos $X_i$, $i = 1, \ldots, N$ [2]:

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \sum_{i=1}^{N} \rho(X_i - \theta) \quad . \quad (1)$$

La estimación del parámetro local $\theta$ puede ser encontrada mediante el cálculo de la derivada parcial de $\psi$ (con respecto a $\theta$).

$$\sum_{i=1}^{N} \psi(X_i - \theta) = 0 \quad , \quad (2)$$

donde $\theta$ es un parámetro local.

La solución del estimador robusto $M$ para $\theta$ es determinado imponiendo ciertas restricciones en la función de influencia $\psi(X)$ o en las muestras $X_i - \theta$, llamada censorización o "trimming". La técnica estándar para el estimador asume el uso del método iterativo de Newton que puede ser simplificado por un algoritmo de un paso para calcular la estimación $M$ del valor de la media de $\theta$ [2]

$$\theta_{\mathrm{M}} = \frac{\sum_{i=1}^{N} X_i \tilde{\psi}(X_i - \mathrm{MED}\{X\})}{\sum_{i=1}^{N} 1_{[-r,r]}(X_i - \mathrm{MED}\{X\})}, \qquad (3)$$

donde $\tilde{\psi}$ es la función normalizada de $\psi$: $\psi(X) = X\tilde{\psi}(X)$. Es evidente que la ultima fórmula representa la media aritmética de $\sum_{i=1}^{n} \psi(X_i - \mathrm{MED}\{\mathbf{X}\})$, la cual es evaluada en el intervalo $[-r,r]$.

La simple restricción del rango de $\psi(X)$ es el estimador $M$ de Huber [5, 7]

$$\tilde{\psi}_b(X) = \mathrm{MIN}(b, \mathrm{MAX}(X,-b)) = [X]_{-b}^{b} \qquad . \qquad (4)$$

Otra manera para encontrar la función $\tilde{\psi}(X)$ es la de eliminar los valores lejanos de una muestra de datos. Este método es llamado estimaciones $M$ mínimas. Hampel demostró en [7] que la función de corte mediana es la más reducida estimación robusta $M$. Anteriormente, usamos la función simple de corte. Existen diferentes funciones de influencia propuestas en la literatura. Nosotros también usamos las funciones de influencia Hampel, Andrews, Tukey y Bernoulli [2, 7].

Para mejorar las propiedades robustas de los estimadores $R$ y $M$ mediante el uso de estimaciones $R$ consiste en un procedimiento similar a la media aritmética [5, 6, 9]:

$$\theta_{\mathrm{medM}} = \mathrm{MED}\{X_i \tilde{\psi}(X_i - \mathrm{MED}\{\mathbf{X}\}), i = 1, \ldots, N\} \qquad . \qquad (5)$$

Este estimador es llamado estimador combinado RM. Hay que hacer notar que el estimador RM (5) es usualmente la mediana cuando la función $\psi$ es representada por la ec. (4). Si la función $\tilde{\psi}$ es descrita por la función simple de corte, esta se convierte en la función de corte mediana. Es posible suponer que las propiedades de los estimadores robustos RM excedan las propiedades robustas de los estimadores de base $R$ y $M$.

Para incrementar la robustez de los filtros estándar, es posible emplear diferentes métodos conocidos en la teoría de estimación robusta [1, 2]. Una propuesta para incrementar la calidad de filtración vía preservación de detalles en las imágenes, consiste en el uso de $K$ elementos de una muestra de valores los cuales son encerrados con respecto al valor del pixel central en la ventana de filtrado. Este método es conocido como el algoritmo de filtrado KNN [2].

Para mejorar la robustez del filtro KNN para que pueda preservar los contornos y detalles de pequeña escala en presencia de ruido impulsivo, propusimos el uso de la estimación RM (5). El filtro resultante "Median M-type K-Nearest Neighbor", que nosotros llamamos simplemente como el algoritmo MM-KNN es escrito como [6, 9]:

$$\hat{f}^{(w)}_{\text{MMKNN}}(i,j) = \text{MED}\left\{h^{(w)}(i+m, j+n)\right\}, \quad (6)$$

donde $h^{(w)}(i+m, j+n)$ es un conjunto $K_{close}$ de valores de pixeles con peso de acuerdo con la función de influencia $\tilde{\psi}(X)$ encerrados en la ventana de filtrado en la estimación obtenida en el paso previo $\hat{f}^{(w-1)}_{\text{MMKNN}}(i,j)$. La estimación inicial es $\hat{f}^{(0)}_{\text{MMKNN}}(i,j) = x(i,j)$ y $\hat{f}^{(w)}_{\text{MMKNN}}(i,j)$ denota la estimación en la iteración $w$. $x(i,j)$ es el pixel actual o pixel original contaminado por ruido en la ventana de filtrado. La ventana de filtrado satisface la expresión $N = (2L+1)^2$ y $m,n = -L,\ldots,L$. El número actual de pixeles vecinos cercanos $K_{close}(i,j)$ refleja la actividad local de los datos y la presencia de ruido impulsivo y es determinado como [5, 6]

$$K_{close}(i,j) = [K_{min} + aD_S(i,j)] \le K_{max} \quad (7)$$

Aquí, el parámetro $a$ controla la sensitividad del filtro para detectar los detalles en las imágenes. $K_{min}$ es el número mínimo de pixeles vecinos para remover el ruido y $K_{max}$ es el número máximo de vecinos que restringen los contornos y el emborronamiento de los detalles. $D_S(i,j)$ es el detector de picos [6], y MAD es la mediana de las desviaciones absolutas de la mediana [2], los cuales son definidos como:

$$D_S(i,j) = \frac{\text{MED}\left\{|x(i,j) - x(i+m, j+n)|\right\}}{\text{MAD}\{x(i,j)\}} + 0.5\frac{\text{MAD}\{x(i,j)\}}{\text{MED}\{x(i+k, j+l)\}}, \quad (8)$$

$$\text{MAD}\{x(i,j)\} = \text{MED}\left\{|x(i+m, j+n) - \text{MED}\{x(i+k, j+l)\}|\right\} \quad (9)$$

En nuestros experimentos, una ventana 3x3 es aplicada. El algoritmo finaliza cuando $\hat{f}^{(w)}_{\text{MMKNN}}(i,j) = \hat{f}^{(w-1)}_{\text{MMKNN}}(i,j)$. Además, los valores de los parámetros del filtro MM-KNN y de las funciones de influencia fueron encontrados después de numerosas simulaciones [8, 9]. Las funciones de influencia anteriormente mencionadas en el filtro propuesto (6) proveen una buena supresión de ruido

impulsivo [2, 6, 7]. También propusimos la aplicación del filtro mediano estándar para mejorar la habilidad de supresión de ruido del filtro MM-KNN en presencia de ruido impulsivo. Las simulaciones mostraron que cuando $K_{close} > 7$ el filtro MM-KNN puede ser sustituido por el filtro *mediano 3x3* y cuando $K_{close} > 350$ usamos el filtro *mediano 5x5* [6, 8, 9].

## 3. Rendimiento del Filtro MM-KNN.

El filtro MM-KNN con diferentes funciones de influencia fue evaluado, y su desempeño fue comparado con diferentes filtros no lineales [8-12]. El criterio usado para comparar el rendimiento en supresión de ruido de diferentes filtros fue el *pico de la relación señal a ruido* (PSNR) y para la evaluación de la preservación de detalles el criterio usado fue el *error absoluto medio* (MAE) [1,2].

$$\mathrm{PSNR} = 10 \cdot \log\left[\frac{(255)^2}{\mathrm{MSE}}\right] \ \mathrm{dB} \quad , \quad (10)$$

$$\mathrm{MAE} = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} f(i,j) - \hat{f}(i,j) \quad , \quad (11)$$

donde $\mathrm{MSE} = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \left[f(i,j) - \hat{f}(i,j)\right]^2$ es el *error cuadrático medio*, $f(i,j)$ es la imagen original; $\hat{f}(i,j)$ es la imagen restaurada; y $M, N$ es el tamaño de la imagen.

Para determinar las propiedades de supresión de ruido impulsivo de diferentes filtros, las imágenes "Airfield", "Barbara", "Boat", "Bridge", "Goldhill" y "Peppers" (imágenes en niveles de gris de 256x256 pixeles), fueron degradadas con 20% de ruido impulsivo. El rendimiento PSNR y MAE del filtro MM-KNN 3x3 con las funciones de influencia de *corte mediana*, *Hampel*, *Andrews*, *Tukey* y *Bernoulli* fueron comparados con el rendimiento de los filtros WM 3x3 (*weighted median*), TSM 3x3 (*tri-state median*), ACWM 3x3 (*adaptive center weighted median*), ROM 3x3 (*rank order mean*), MMEM 3x3 (*minimum-maximum exclusive mean*) y LMMSE 3x3 (*Local Linear Minimum Mean Square Error*). Los filtros usados para comparar nuestro método fueron computados de acuerdo a sus referencias [10-16]. La razón de haber escogido estos filtros para compararlos con el nuestro es que su rendimiento ha sido comparado con diferentes filtros conocidos y sus ventajas han sido demostradas. Los valores PSNR y MAE son mostrados en las Tablas 1 y 2 respectivamente.

La Tabla 1 muestra que el rendimiento PSNR del filtro propuesto MM-KNN es mejor en comparación a los filtros utilizados como comparativo, (especialmente cuando el porcentaje de ruido es mayor al 5%). Resultados similares fueron obtenidos al calcular los valores MAE (ver Tabla 2). Uno puede ver que el filtro ROM tiene en algunos casos los mejores valores de supresión de ruido y preservación de detalles, pero la ventaja de nuestro procedimiento es que el filtro propuesto MM-KNN no usa datos de entrenamiento y los parámetros del filtro propuesto pueden ser tomados como constantes. Además, el filtro ROM presenta los mejores resultados visuales en preservación de detalles, pero su desventaja es que tiende a variar grandemente el valor del pixel central en la ventana de filtrado, esto puede ser fácilmente visualizado en imágenes a color. El parámetro $a$ del filtro propuesto y los parámetros $\alpha$, $\beta$ y $r$ de las funciones de influencia son muy importantes para obtener la suficiente supresión de ruido y preservación de detalles.

Tabla 1. Valores PSNR en dB para diferentes imágenes corrompidas por 20% de ruido impulsivo.

| Filtro | Airfield | Barbara | Boat | Bridge | Goldhill | Peppers |
|---|---|---|---|---|---|---|
| 3x3 WM | 22.40 | 21.38 | 22.20 | 22.88 | 24.21 | 24.68 |
| 3x3 TSM | 21.13 | 21.77 | 21.47 | 20.57 | 23.11 | 22.96 |
| 3x3 ACWM | 22.97 | 23.67 | 23.26 | 23.14 | 24.84 | 25.18 |
| 3x3 ROM | 23.08 | 23.84 | 23.40 | 23.06 | 24.82 | 25.04 |
| 3x3 MMEM | 22.69 | 22.93 | 22.92 | 22.47 | 24.16 | 24.40 |
| 3x3 LMMSE | 23.03 | 22.79 | 22.29 | 22.58 | 24.15 | 24.75 |
| MM-KNN (Corte) | 23.21 | 23.83 | 23.67 | 22.90 | 25.45 | 26.37 |
| MM-KNN (Hampel) | 23.24 | 23.87 | 23.60 | 22.90 | 25.50 | 26.48 |
| MM-KNN (Andrew) | 23.23 | 23.85 | 23.63 | 22.93 | 25.48 | 26.40 |
| MM-KNN (Tukey) | 23.23 | 23.86 | 23.64 | 22.92 | 25.50 | 26.41 |
| MM-KNN (Berno) | 23.24 | 23.86 | 23.64 | 22.92 | 25.50 | 26.41 |

Tabla 2. Valores MAE para diferentes imágenes corrompidas por 20% de ruido impulsivo.

| Filtro | Airfield | Barbara | Boat | Bridge | Goldhill | Peppers |
|---|---|---|---|---|---|---|
| 3x3 WM | 10.67 | 11.61 | 10.61 | 12.76 | 9.96 | 7.81 |
| 3x3 TSM | 13.44 | 13.86 | 13.13 | 15.62 | 12.43 | 11.38 |
| 3x3 ACWM | 10.57 | 11.08 | 10.57 | 11.85 | 10.43 | 9.21 |
| 3x3 ROM | 10.42 | 10.96 | 10.55 | 11.81 | 10.57 | 9.62 |
| 3x3 MMEM | 12.23 | 12.09 | 11.36 | 13.57 | 11.09 | 9.67 |
| 3x3 LMMSE | 11.24 | 12.10 | 10.28 | 12.91 | 11.08 | 9.70 |
| MM-KNN (Corte) | 10.45 | 10.42 | 9.75 | 12.61 | 9.58 | 7.18 |
| MM-KNN (Hampel) | 10.42 | 10.39 | 9.75 | 12.59 | 9.50 | 7.18 |
| MM-KNN (Andrews) | 10.44 | 10.40 | 9.76 | 12.59 | 9.53 | 7.15 |
| MM-KNN (Tukey) | 10.45 | 10.41 | 9.76 | 12.63 | 9.56 | 7.18 |
| MM-KNN (Bernoulli) | 10.46 | 10.41 | 9.76 | 12.63 | 9.56 | 7.18 |

Para las imágenes de prueba el valor óptimo de la función de *Hampel* es $\beta$=90. El valor óptimo del parámetro $r$ para la función *Andrews* es $r$ =35, para la función de *Tukey* $r$ =15 y para la función de *Bernoulli* $r$ =20. En las Figuras 1 y 2 se presentan las imágenes procesadas para las imagenes "Barbara" y "Goldhill" mostrando la supresión de ruido impulsivo de acuerdo a las Tablas 1 y 2.

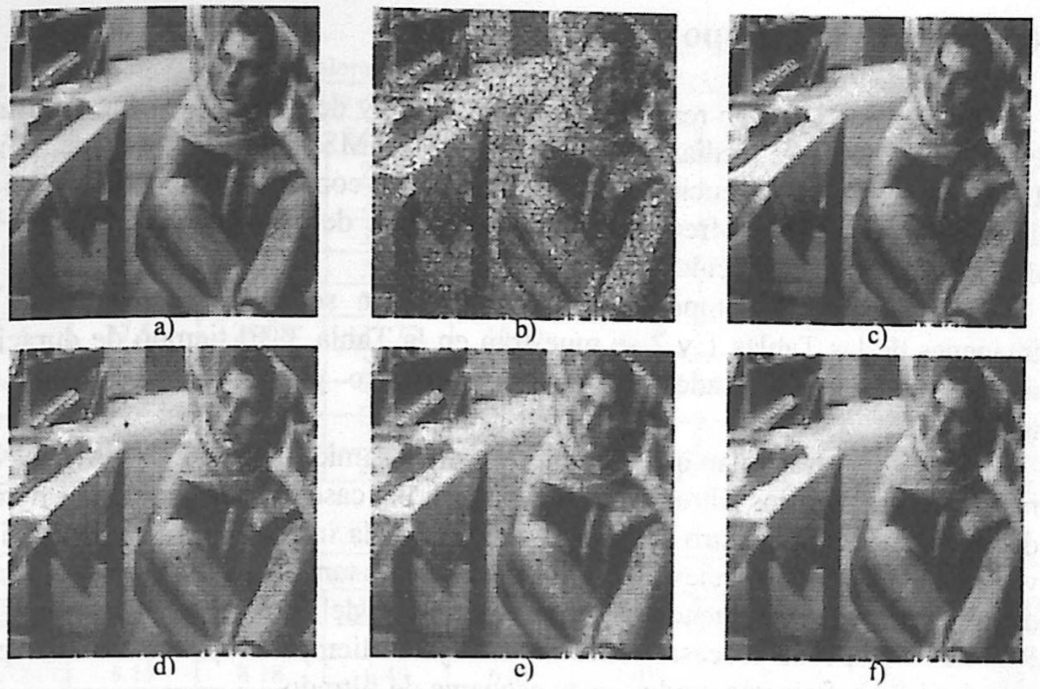



Figura 1. Supresión de ruido impulsivo en la imagen "Barbara". a) Imagen original, b) Imagen degradada con 20% de ruido impulsivo, c) Imagen restaurada con el filtro ACWM, d) Imagen restaurada con el filtro ROM, e) Imagen restaurada con el filtro MMEM, f) Imagen restaurada con el filtro MM-KNN (Corte).

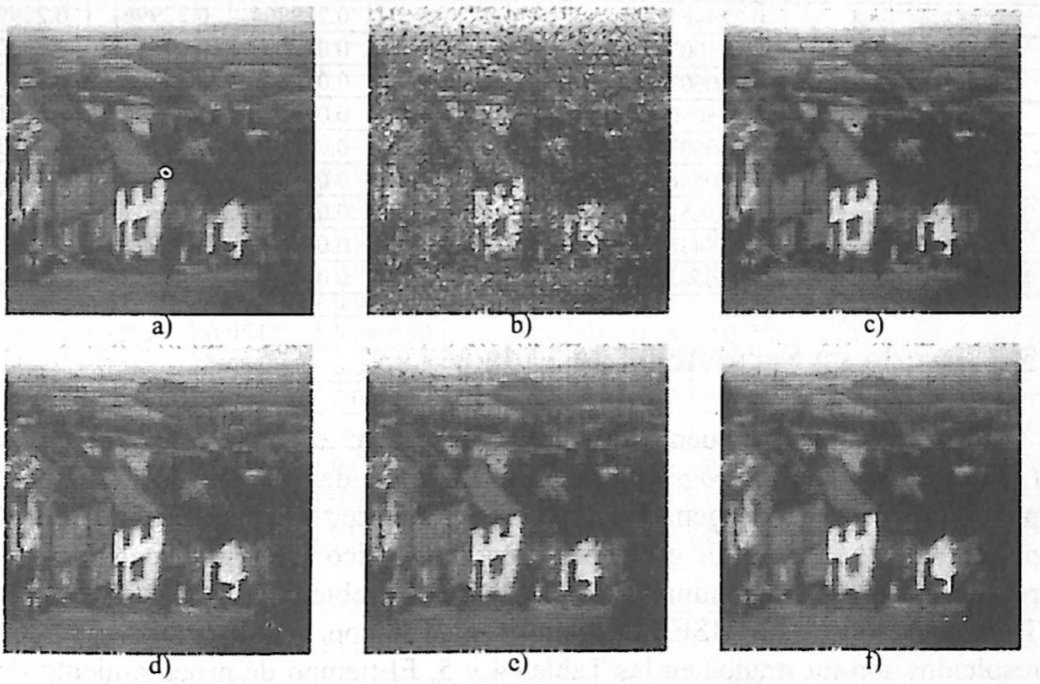



Figura 2. Supresión de ruido impulsivo en la imagen "Goldhill". a) Imagen original, b) Imagen degradada con 20% de ruido impulsivo, c) Imagen restaurada con el filtro ACWM, d) Imagen restaurada con el filtro ROM, e) Imagen restaurada con el filtro LMMSE, f) Imagen restaurada con el filtro MM-KNN (Andrews).

## 4. Análisis en Tiempo Real.

El análisis en tiempo real del filtro MM-KNN y de otros filtros fue realizado en varias imágenes mediante el uso del DSP TMS320C6701 [15] de Texas Instruments. Con la ejecución de hasta 1 GFLOPS con una velocidad de reloj de 167 MHz, el 'C6701 ofrece soluciones efectivas de alto desempeño para los desafíos en la programación de DSPs [17-19].

Los valores del tiempo de procesamiento en segundos realizado en las imagenes de las Tablas 1 y 2 se muestran en la Tabla 3. El tiempo de duración incluye el tiempo de adquisición, procesamiento y almacenamiento de las imágenes.

Los resultados revelan que el tiempo de procesamiento del filtro MM-KNN es mayor que el de otros filtros en la mayoría de los casos. La duración del tiempo de procesamiento del filtro propuesto depende de la imagen a procesar y casi no varia para diferentes niveles de ruido, estos valores también dependen del cálculo de las funciones de influencia y de los parámetros del filtro. En el caso del filtro ROM su tiempo de procesamiento no incluye el tiempo de procesamiento para calcular los coeficientes usados en su esquema de filtrado.

Tabla 3. Tiempo de procesamiento para diferentes imágenes corrompidas por 20% de ruido impulsivo.

| Filtro | Airfield | Barbara | Boat | Bridge | Goldhill | Peppers |
|---|---|---|---|---|---|---|
| 3x3 WM | 0.020341 | 0.020168 | 0.020341 | 0.020341 | 0.020341 | 0.020341 |
| 3x3 TSM | 0.054756 | 0.054758 | 0.054757 | 0.054756 | 0.054758 | 0.054759 |
| 3x3 ACWM | 0.229967 | 0.229954 | 0.229960 | 0.229964 | 0.229961 | 0.229951 |
| 3x3 ROM | 0.075007 | 0.075008 | 0.075008 | 0.075007 | 0.075008 | 0.075008 |
| 3x3 MMEM | 0.040607 | 0.040620 | 0.040606 | 0.040607 | 0.040606 | 0.040618 |
| 3x3 LMMSE | 0.075070 | 0.075118 | 0.075101 | 0.075105 | 0.075087 | 0.075140 |
| MM-KNN (Corte) | 0.060503 | 0.059214 | 0.061038 | 0.060503 | 0.060806 | 0.060382 |
| MM-KNN (Hampel) | 0.070546 | 0.070992 | 0.072360 | 0.071816 | 0.073086 | 0.071449 |
| MM-KNN (Andrews) | 0.075051 | 0.074246 | 0.074308 | 0.077871 | 0.079414 | 0.076526 |
| MM-KNN (Tukey) | 0.059436 | 0.059172 | 0.060846 | 0.060314 | 0.059951 | 0.059672 |
| MM-KNN (Bernoulli) | 0.066561 | 0.065529 | 0.069023 | 0.067937 | 0.067169 | 0.067001 |

## 5. Filtrado en Secuencias de Vídeo.

Simulamos una secuencia de vídeo mediante el uso de imágenes PET (*Positron Emission Tomography*) con una relación de muestreo de 10 imágenes por segundo. Cada imagen de la secuencia de vídeo es muestreada a 256x256 pixeles con 256 niveles de gris. La secuencia de vídeo fue degradada con 20% de ruido impulsivo. Los rendimientos PSNR y MAE obtenidos con los filtros WM, TSM, MMEM y LMMSE fueron comparados con el filtro propuesto y los resultados son mostrados en las Tablas 4 y 5. El tiempo de procesamiento de la secuencia de vídeo es presentado en la Tabla 6. Las imágenes procesadas 1 a 4 de la secuencia de vídeo son mostradas en la Figura 3 mediante el uso del filtro LMMSE y el filtro MM-KNN con la función de influencia de corte mediana.

Tabla 4. Valores PSNR en dB para la secuencia de video PET.

| Núm. Sec. | Filtros de orden estadistico | | | | Filtro MM-KNN | | | |
|---|---|---|---|---|---|---|---|---|
| | WM | TSM | MMEM | LMMSE | Corte | Hampel | Andrews | Bernoulli |
| 1 | 23.53 | 27.90 | 27.68 | 29.23 | 33.04 | 33.75 | 32.76 | 33.00 |
| 2 | 27.49 | 28.21 | 27.78 | 30.12 | 35.21 | 36.63 | 34.93 | 35.52 |
| 3 | 29.01 | 28.34 | 28.16 | 30.07 | 37.06 | 39.03 | 36.83 | 37.14 |
| 4 | 30.04 | 28.79 | 27.73 | 31.53 | 38.51 | 42.75 | 37.28 | 38.47 |
| 5 | 26.58 | 27.96 | 27.72 | 30.66 | 35.19 | 35.94 | 34.39 | 34.67 |
| 6 | 27.58 | 28.08 | 27.77 | 30.70 | 36.97 | 38.75 | 36.43 | 36.62 |
| 7 | 32.41 | 28.12 | 28.54 | 31.44 | 45.90 | 42.09 | 41.23 | 42.46 |
| 8 | 32.36 | 28.17 | 28.16 | 31.89 | 39.67 | 42.64 | 38.42 | 39.51 |
| 9 | 30.65 | 28.48 | 28.17 | 32.31 | 44.12 | 42.50 | 41.74 | 43.23 |
| 10 | 30.22 | 29.08 | 28.29 | 31.42 | 43.18 | 42.45 | 40.80 | 41.90 |

Tabla 5. Valores MAE para la secuencia de video PET.

| Núm. Sec. | Filtros de orden estadistico | | | | Filtro MM-KNN | | | |
|---|---|---|---|---|---|---|---|---|
| | WM | TSM | MMEM | LMMSE | Corte | Hampel | Andrews | Bernoulli |
| 1 | 8.16 | 8.26 | 6.63 | 7.05 | 5.17 | 5.26 | 5.13 | 5.24 |
| 2 | 8.17 | 8.18 | 6.49 | 6.85 | 4.97 | 5.05 | 4.90 | 5.03 |
| 3 | 8.00 | 8.18 | 6.41 | 6.80 | 4.80 | 4.82 | 4.72 | 4.79 |
| 4 | 7.73 | 8.01 | 6.38 | 6.30 | 4.56 | 4.60 | 4.38 | 4.55 |
| 5 | 7.96 | 8.23 | 6.41 | 6.57 | 4.77 | 4.85 | 4.70 | 4.80 |
| 6 | 8.08 | 8.25 | 6.68 | 6.68 | 4.83 | 4.91 | 4.78 | 4.86 |
| 7 | 7.81 | 8.08 | 6.16 | 6.54 | 4.46 | 4.45 | 4.38 | 4.44 |
| 8 | 7.60 | 8.02 | 6.01 | 6.71 | 4.20 | 4.22 | 4.14 | 4.19 |
| 9 | 7.37 | 7.96 | 6.09 | 6.18 | 4.24 | 4.19 | 4.16 | 4.18 |
| 10 | 7.48 | 7.66 | 5.88 | 6.22 | 4.02 | 4.06 | 3.96 | 4.01 |

Tabla 6. Tiempo de procesamiento para la secuencia de video PET.

| Núm. Sec. | Filtros de orden estadistico | | | | Filtro MM-KNN | | | |
|---|---|---|---|---|---|---|---|---|
| | WM | TSM | MMEM | LMMSE | Corte | Hampel | Andrews | Bernoulli |
| 1 | 0.020341 | 0.054756 | 0.040582 | 0.075108 | 0.060746 | 0.069584 | 0.082740 | 0.068537 |
| 2 | 0.020341 | 0.054759 | 0.040588 | 0.075115 | 0.061255 | 0.071004 | 0.081302 | 0.067388 |
| 3 | 0.020341 | 0.054757 | 0.040592 | 0.075070 | 0.061181 | 0.070612 | 0.082707 | 0.067331 |
| 4 | 0.020340 | 0.054755 | 0.040590 | 0.075124 | 0.061824 | 0.071712 | 0.086542 | 0.068776 |
| 5 | 0.020341 | 0.054757 | 0.040592 | 0.075116 | 0.061456 | 0.072268 | 0.084483 | 0.070175 |
| 6 | 0.020341 | 0.054757 | 0.040585 | 0.075080 | 0.062739 | 0.072810 | 0.084752 | 0.069196 |
| 7 | 0.020341 | 0.054758 | 0.040597 | 0.075076 | 0.062859 | 0.072597 | 0.087664 | 0.068960 |
| 8 | 0.020341 | 0.054756 | 0.040596 | 0.075078 | 0.061319 | 0.070744 | 0.084072 | 0.069399 |
| 9 | 0.020341 | 0.054754 | 0.040591 | 0.075087 | 0.062859 | 0.073974 | 0.083702 | 0.070640 |
| 10 | 0.020341 | 0.054758 | 0.040596 | 0.075088 | 0.061536 | 0.071503 | 0.084774 | 0.070706 |

De los resultados obtenidos en las Tablas 4 y 5 se observa que el filtro MM-KNN tiene mejores propiedades de supresión de ruido y preservación de detalles en comparación con otros filtros, en el caso de imágenes PET su propia constitución en niveles de gris ayuda grandemente a un mejor filtrado. El filtro LMMSE fue el filtro comparativo que obtuvo los mejores resultados ya que esta diseñado especialmente para secuencias de vídeo [14]. El filtro ROM no se

utilizo debido a que su esquema de filtrado requiere de datos de entrenamiento y no es factible su implementación en tiempo real. Con lo que respecta al tiempo de procesamiento, nuestra técnica tiene el mayor tiempo de procesamiento a excepción del filtro LMMSE. De los resultados de las Tablas 3 y 6, es claro que el sistema puede procesar 10 imágenes de 256x256 con un tiempo de espera de 0.01 a 0.03 seg., dependiendo de la función de influencia usada. En el caso de imágenes de 64x64 la técnica de filtrado propuesta puede suprimir el ruido impulsivo y preservar los detalles finos prácticamente a la velocidad de una película estándar en sistemas de visión por computadora.



a) Secuencia de video original PET (imágenes 1 a 4).



b) Secuencia de video degradada de a) con 20% de ruido impulsivo (imágenes 1 a 4).



c) Secuencia de video restaurada con el filtro LMMSE (imágenes 1 a 4).



d) Secuencia de video restaurada con el filtro MM-KNN (corte mediana) (imágenes 1 a 4).

Figura 3. Resultados de supresión de ruido impulsivo en una secuencia de video PET.

## 6. Conclusiones.

Se presentó la implementación en tiempo real de un filtro robusto para supresión de ruido impulsivo y preservación de detalles mediante el uso del DSP TMS320C6701. El filtro robusto MM-KNN fue diseñado con diferentes funciones de influencia para aplicaciones en procesamiento de imágenes.

Extensivas simulaciones demostraron que desempeño del filtro propuesto tiene mejores propiedades de supresión de ruido y preservación de detalles que otros filtros no lineales presentados en la literatura.

El filtro propuesto provee una solución en tiempo real para la supresión de ruido y preservación de detalles en las transmisiones de televisión y vídeo para mejorar la calidad de éstas.

Es claro que en el caso de imágenes en niveles de gris de 64x64 la técnica de filtrado propuesta puede suprimir el ruido impulsivo y preservar los contornos y detalles finos prácticamente a la velocidad de una película estándar en sistemas de visión por computadora.

Los tiempos de procesamiento pueden ser reducidos si se utiliza un DSP con mayor desempeño que el TMS32C6701 tanto en velocidad de reloj como en ciclos de instrucción.

**Agradecimientos.** Los autores dan las gracias al CONACYT por el apoyo dado al proyecto de investigación 31978-A. Al Instituto Politécnico Nacional y al programa PROMEP SUPERA por los apoyos brindados.

## Referencias

[1] A. Bovik, *Handbook of Image and Video Processing*, Acadamic Press, San Diego CA, 2000.

[2] J. Astola and P. Kuosmanen, *Fundamentals of Nonlinear Digital Filtering*, CRC Press, Boca Raton-New York, 1997.

[3] A. Murat, *Digital Video Processing*, Prentice Hall, Upper Saddle River, New Jersey, 1995.

[4] G. R. Arce, "Multistage order statistic filters for image sequence processing," *IEEE Trans. Signal Process.* 39(5), 1146-1163, 1991.

[5] V. I. Ponomaryov and O. B Pogrebnyak, "Novel robust RM filters for radar image preliminary processing," *Journal of. Electronic Imaging* 8(04), pp. 467-477, 1999.

[6] V. I. Ponomaryov, F. J. Gallegos Funes, O. B. Pogrebnyak, Luis N. de Rivera, "Two-stage robust rank filter with fine detail preserving for the image processing," *Proc. SPIE Visual Information Processing X*, vol. 4388, 275-286, 2001.

[7] F. R. Hampel, E. M. Ronchetti, P. J. Rouseew, W. A. Stahel, *Robust Statistics. The approach based on influence function*, Wiley, New York, 1986.

[8] Francisco J. Gallegos Funes, Volodymyr I. Ponomaryov, Oleksiy B. Pogrebnyak, L. Niño-de-Rivera, "Estimadores robustos RM con diferentes funciones de influencia en el filtro KNN", Memorias del VI Taller Iberoamericano de Reconocimiento de Patrones "Avances Recientes en Análisis de Imágenes y Reconocimiento de Patrones TIARP 2001, pp. 1-21, México, D. F., Noviembre 2001.

[9] F. J. Gallegos-Funes, V. Ponomaryov, S. Sadovnychiy, L. Niño-de-Rivera, "Median M-type K-nearest neighbour (MMKNN) filter to remove impulse noise from corrupted images", *IEE Electronics Letters*, 38(15), pp. 786-787, 2002.

[10] T. Chen, K. Ma and L. Chen, "Tri-State Median filter for image denoising," *IEEE Trans. Image Process.* 8(12), pp. 1834-1838, 1999.

[11] T. Chen and H. R. Wu, "Adaptive impulse detection using center-weighted median filters," *IEEE Signal Processing Letters*, 8(1), 1-3, 2001.

[12] E. Abreu, M. Lightstone, S. K. Mitra and K. Arakawa, "A new efficient approach for the removal of impulse noise from highly corrupted images," *IEEE Trans. Image Process.* 5(6), pp. 1012-1025, 1996.

[13] H. Wei-Yu, and Lin Ja-Chen, "Minimum-maximum exclusive mean (MMEM) filter to remove impulse noise from highly corrupted images,". *Electronics Letters*, 33(2), pp. 124-125, 1997.

[14] M. K. Özkan, M. I: Sezan and A. Murat, "Adaptive motion-compensated filtering of noisy image sequences," *IEEE Trans. Circuits and Syst. For Video Tech.* 3(4), 277-290, 1993.

[15] T. S. Huang, G. J. Yang and G. Y. Tang, "A Fast two-dimentional median filtering algorithm," *IEEE Trans. Acoust. Speech, Signal Process.* ASSP-27, pp. 13-18, 1979.

[16] I. Pitas, "Fast algorithms for running order and max/min calculations," *IEEE Trans. Circuits and Systems*, 36(6), pp. 795-904, 1989.

[17] Texas Instruments, *TMS320C6201/6701 Evaluation Module User's Guide, SPRU269D*, Texas Instruments Incorporated, 1998.

[18] Texas Instruments, *TMS320C6000 Optimizing Compiler User's Guide, SPRU187G*, Texas Instruments Incorporated, 2000.

[19] Texas Instruments, *TMS320C6000 Code Composer Studio Tutorial, SPRU301C*, Texas Instruments Incorporated, 2000.

# Arquitectura Neuronal Convolucional para la Detección de Defectos en Productos en Continuo

**José A. Calderón Martínez[1,2] y Pascual Campoy Cervera[1]**

[1] Departamento de Automática, Ingeniería Electrónica
e Informática Industrial
Universidad Politécnica de Madrid
C/ José Gutiérrez Abascal 2, 28006 Madrid, España
campoy@etsii.upm.es
http://www.disam.upm.es/vision/
[2] Instituto Tecnológico de Aguascalientes
Av. López Mateos 1801 Ote., 20256 Aguascalientes, Ags., México
SEP, CONACYT
acaldero@etsii.upm.es

**Abstract.** Los sistemas de inspección visual automática suelen ser cruciales en los procesos de fabricación de hoy en día. Aquellos sistemas basados en redes neuronales permiten un compromiso entre resolución y velocidad de procesamiento. Este trabajo presenta el desarrollo de una nueva arquitecura neuronal, denominada *Arquitectura Espiral Descendente Convolucional* (AEDC), para generar de forma automática filtros digitales aplicables en sistemas de inspección mediante visión artificial. Se presentan los resultados experimentales de ésta arquitectura aplicada en la detección de defectos en pasta de papel de imágenes obtenidas en un medio ambiente de fabricación real.

**Palabras clave:** Redes neuronales convolucionales, filtros digitales, visión artificial, inspección automática.

## 1 Introducción

La demanda de los sistemas de inspección automática mediante visión artificial en las industrias de fabricación se incrementa día con día. Debido a sus características de velocidad, flexibilidad y confiabilidad se convierten en uno de los sistemas de inspección más altamente competitivos. Reducen costos de inspección, de producción fuera de calidad y pueden llegar a inspeccionar hasta el 100% de productos fabricados incrementado con ello la confianza de los sistemas de producción.

Varios autores han trabajado en detección de defectos en diferentes tipos de procesos. Conners [1] propuso metodos de analisis de texturas para detectar defectos en laminas de madera automaticamente. Dewaele et al. [2] uso metodos de procesamiento de señales para detectar puntos y lineas de defecto en imagenes de texturas. Lee et al. [3] recientemente ha usado redes neuronales para clasificar defectos mediante caracteristicas de energia y entropia de imagenes de piezas

de acero. Stojanovic et al. [4] propone una mezcla de técnicas de inspección basada en procesamiento de imagenes binarias, análisis estadístico y clasificación mediante redes neuronales, en un sistema de inspección de productos textiles. Campoy et al. [5] recientemente ha propuesto la aplicación de filtros digitales generados mediante redes neuronales en la inspección de pasta de papel.

## 2 Reconocimiento de patrones

La figura 1 ilustra como se realiza un *clásico* reconocimiento de patrones. Ha de mencionarse que la etapa de *segmentación* no se realiza en todas las aplicaciones (ejm. inspección de acabado de superficies), por lo que algunos autores la contemplan como parte de la etapa de *preprocesamiento.*

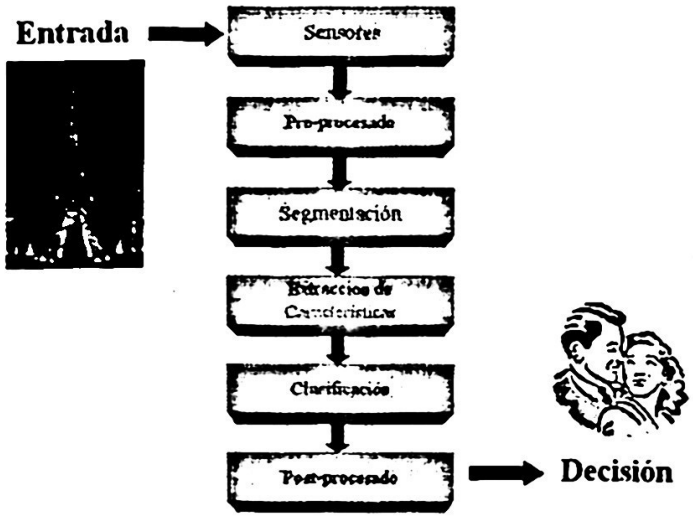



**Fig. 1.** Sistema de Reconocimiento de Patrones

Un problema de reconocimiento bién definido y suficientemente restringido, conducirá a una representación de patrones compacta y a una estrategia de toma de decisiones simple. Los cuatro enfoques mejor conocidos en el reconocimiento de patrones[6], y no necesariamente independientes, son: 1) template matching (comparación con plantillas), 2) clasificación estadistica, 3) comparación estructural o sintáctico, y 4) redes neuronales.

## 3 Redes Convolucionales

El neocognitron introducido por Fukushima[7] puede ser considerado como la primera implementación de una red convolucional.

Las redes neuronales convolucionales (Convolutional Neural Networks) proveen un método eficiente para restringir la complejidad de redes neuronales unidireccionales (feedforward) mediante la compartición de pesos (weight sharing) y la restricción a conexiones locales. Esta topología ha sido aplicada en particular a clasificación de imágenes cuando se tiene que evitar pre-procesamiento sofisticado y clasificar directamente imágenes brutas o no procesadas.

Existen varias posibilidades para diseñar una arquitectura de red neuronal convolucional (CNN), mediante la combinación de diferentes tipos de neuronas y reglas de aprendizaje. En la figura 2 se ilustra una arquitectura básica de una red neuronal convolucional, en ella se muestran tres capas, una imagen de entrada, tres grupos de imágenes en la capa oculta, y una salida. El número de capas y el número de grupos en cada capa depende de la aplicación.

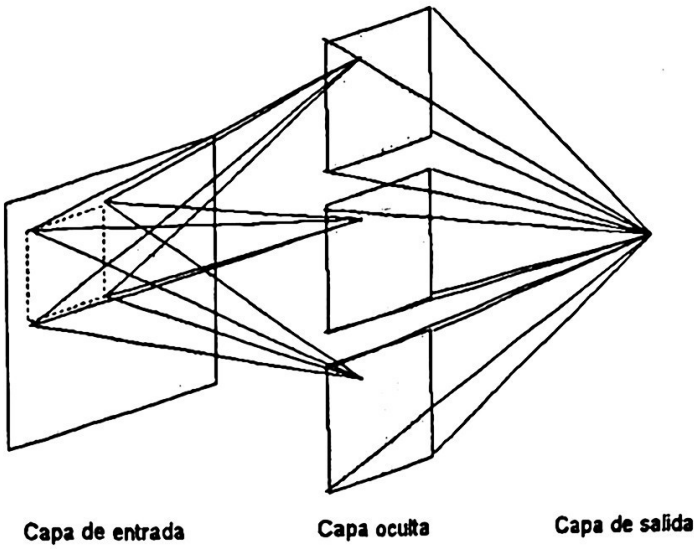



**Fig. 2.** Arquitectura básica de una CNN

Las redes neuronales convolucionales combinan tres ideas arquitecturales para asegurar algún grado de invarianza en desplazamiento, escala y distorsión: *campos receptivos locales*, *pesos compartidos* (o replicar pesos), y algunas veces *sub-muestreo* espacial o temporal [8, 9]. El uso de pesos compartidos también reduce el número de parámetros en el sistema ayudando a la generalización.

El proceso de aprendizaje de una red convolucional está basado en el algoritmo de retro-propagación, que actualiza los pesos $w$ de las neuronas

$$w(t+1) = w(t) + \eta\delta(t)x(t) \tag{1}$$

donde $\eta$ es la tasa de aprendizaje, $x(t)$ es la entrada a la neurona, y $\delta(t)$ es un término de error para la neurona.

Las principales aplicaciones que se han hecho de las redes neuronales convolucionales son el las de LeCun[10] en reconocimiento de caracteres (impresos y manualmente escritos), de rostros por Lawrence[11, 12], y en el análisis de imágenes médicas por Sahiner et al [13].

## 4 Arquitectura Neuronal Desarrollada

En este apartado se presenta el sistema que se ha desarrollado para llevar a cabo la inspección de pasta de papel. Primeramente se analiza la fase de obtención de los filtros digitales mediante redes neuronales y posteriormente la fase de detección y clasificación de los diferentes tipos de defectos.

### 4.1 Generación automática de filtros digitales

La fase de obtención de los filtros está compuesta por varias etapas ilustradas en el diagrama de bloques de la figura 3 y que a continuación se describen:

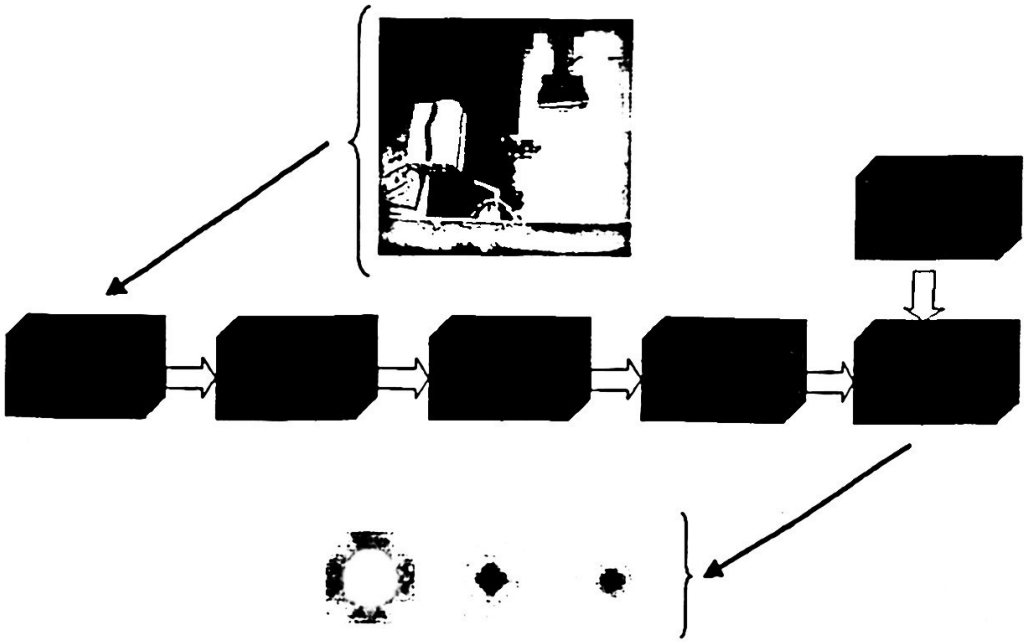


**Fig. 3.** Actividades en la generación de los filtros digitales.

### 4.2 Adquisición de imágenes para entrenamiento

Las imágenes se captan en la línea de producción con una cámara TDI de 2048 pixels de resolución horizontal, 96 líneas de acumulación lumínica y 44 Klíneas/s

de velocidad máxima de sincronización, con una óptica de 35 mm f/1:3, y utilizando iluminación a contraluz por fibra óptica con potencia lumínica de 150 W. La resolución de las imágenes es de 15 pixels/mm, obteniendo un área de 3x3 pixels para los defectos de 0.04 $mm^2$.

## 4.3 Clasificación supervisada

Una vez que se han obtenido las imágenes, estas son analizadas por el experto humano que anteriormente realizaba la tarea de inspección, y él (o ella) clasifica los defectos en los diferentes tipos basándose en las normas establecidas y de acuerdo con su experiencia. Los once tipos de defecto a identificar se muestran en la tabla 1.

**Table 1.** Tipos de defecto

| Tipo | Tamaño, $mm^2$ |
|---|---|
| E | 0.08 |
| D | 0.20 |
| C | 0.45 |
| B | 0.80 |
| A1 ... A7 | 1.0 ... 5.0 |

## 4.4 Selección de imágenes

Esta etapa consiste en seleccionar las imágenes más representativas de los diferentes tipos de defectos, así como de imaágenes sin defecto.

## 4.5 Normalización de imágenes

Para minimizar las diferencias entre los niveles de gris de las imágenes obtenidas, estas son normalizadas mediante el ajuste de la distribución de los niveles de gris a una distribución Gaussiana con media igual a 0 y con una desviación de 1.

$$N' = \frac{N - M}{\sigma} \tag{2}$$

## 4.6 Generación de filtros para cada tipo de defecto

La *Arquitectura Espiral Descendente Convolucional* (AEDC) mostrada en la figura 4 es la que se se ha diseñado y utilizado para generar los filtros digitales que se aplican durante la inspección en línea.

La red neuronal se entrena fuera de línea utilizando muestras con el tipo de defecto para el cual se va a obtener el filtro. así como muestras sin defecto.

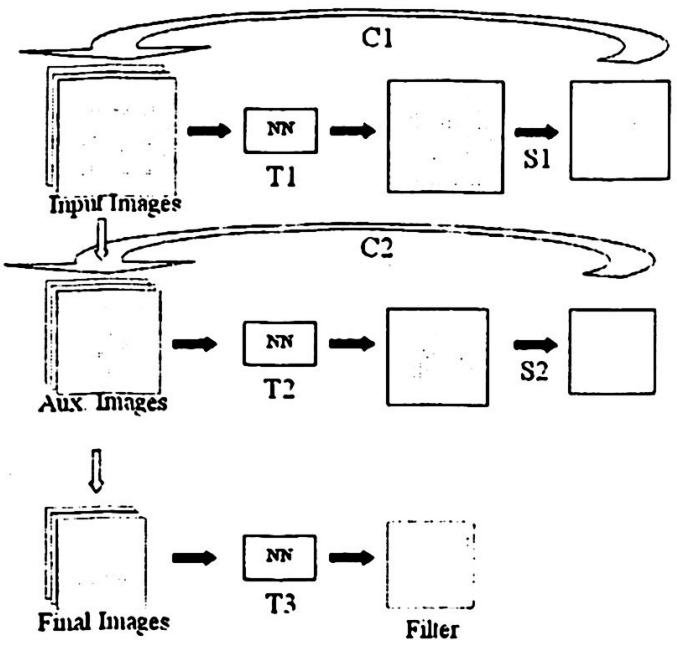



**Fig. 4.** Arquitectura Espiral Descendente Convolucional (AEDC).

Como puede observarse en la figura 4 el sistema es alimentado con imágenes de 30x30 pixels, con y sin defecto, con las que se entrena (T1) a una red neuronal mediante retropropagación la cual genera los coeficientes con los que se obtiene un filtro de dimensiones similares, posteriormente se realiza un sub-muestreo (S1) elimminando los pixeles más alejados del centro, obteniendo una imagen de 28x28 pixels las cuales son utilizadas para efectuar una convolución (C1) a las imágenes originales con el objeto de obtener unas imágenes auxiliares mejor centradas que las primeras, para ser utilizadas en la siguiente etapa de entrenamiento. El proceso se repite varias veces en forma de espiral descendente hasta obtener el filtro de tamaño reducido y adecuado, siendo el límite el tamaño del defecto para el cual se está diseñando el filtro.

### 4.7 Generación de imágenes sintéticas

Este es un módulo auxiliar en el que se aplica un algoritmo para generar imágenes sintéticas utilizando las imágenes originales. La utilidad es que de esta forma se cuenta con un mayor número de imágenes para realizar el entrenamiento de la red neuronal. Se obtienen siete imágenes sintéticas por cada imagen original. Las potenciales aplicaciones de este algoritmo son para los casos en los que se cuenta con pocas imágenes para el entrenamiento. y/o en donde la obtención de muestras es costosa o imposible. En la figura 5 se muestra un ejemplo de imágenes sinteticas generadas por el sistema.

Con esta generación de imágenes sintéticas se garantiza la isotropía de los filtros digitales que se obtienen tras el proceso de aprendizaje. Esta isotropía es

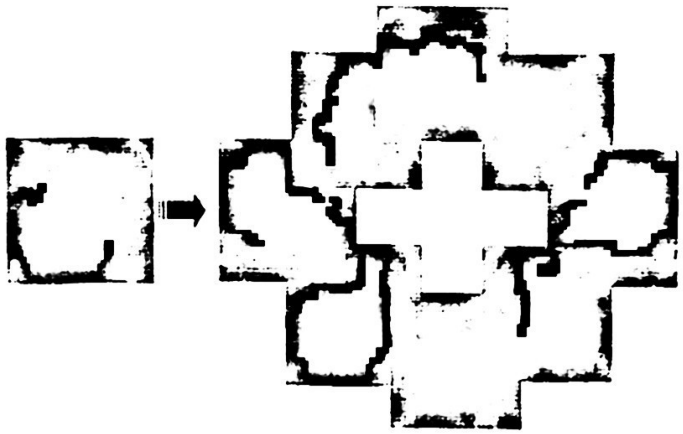


**Fig. 5.** Ejemplo de imágenes sintéticas.

un conocimiento a priori del problema y su introducción mediante la generación de imágenes sintéticas da lugar a unos resultados similares al mencionado método de las ligadouras de los pesos de la red neuronal.

## 5 Detección e identificación de defectos

La figura 7 muestra como se lleva a cabo la inspección de las imágenes de pasta de papel.

### 5.1 Adquisición de imágenes

Se adquieren imágenes de 682x2024 pixels de la pasta de papel, con la cámara TDI mencionada anteriormente, en la línea de producción a la velocidad con que se esté efectuando el proceso.

### 5.2 Detección de regiones de interés

A las imágenes captadas se les aplica el algoritmo de detección de defectos basado en etiquetado de regiones de interés, siendo estas regiones las posibles zonas que contengan los defectos.

El algoritmo incluye una binarización y una escaneo de la imagen completa, obteniendo como salida una imagen auxiliar con las regiones de interés perfectamente delimitadas.

### 5.3 Obtención de características

En ésta etapa se obtienen características de las regiones detectadas en cada una de las imágenes analizadas, como *área* de la región detectada, *reflexiones en x,y*, y *centroide*, que son utilizadas para la extracción o aislamiento de la zona de defecto (imagen de 30x30 pixels) para su análisis y clasificación.

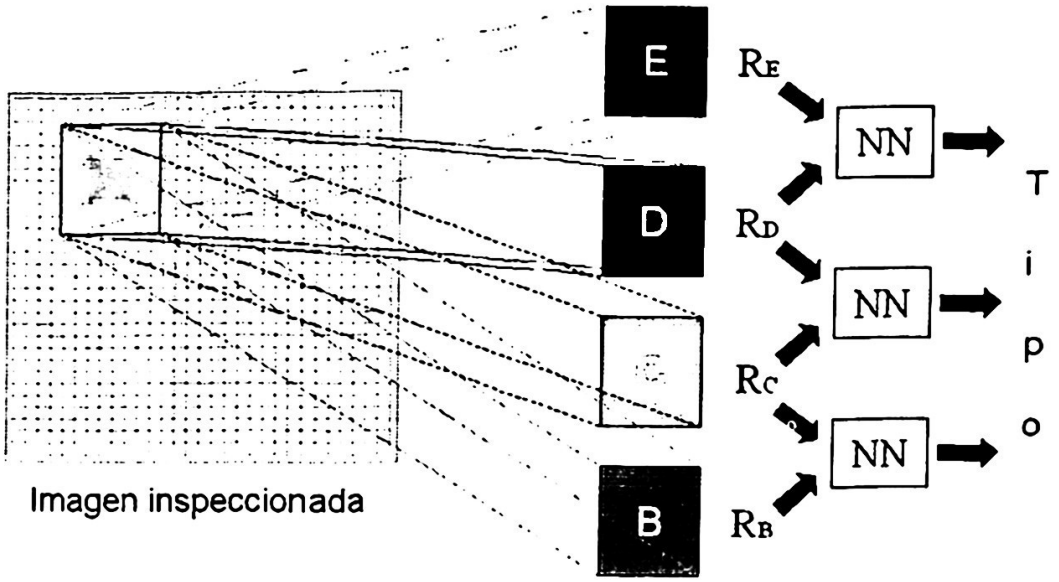



**Fig. 6.** Arquitectura utilizada para realizar la inspección.

### 5.4 Identificación de tipos de defecto

Los filtros obtenidos en la primera fase del sistema desarrollado se aplican de acuerdo con la técnica de redes convolucionales. como se muestra en la figura 7. obteniendo como respuesta a los filtros la identificación del tipo de defecto en cuestión. Los casos en los que el defecto analizado responda a la aplicación de dos filtros "vecinos" de forma similar, su tipo se define mediante la utilización de una red neuronal posteriormente a los filtros.

## 6 Resultados experimentales

El tamaño del filtro es importante por cuestiones de tiempo de ejecución durante la fase de identificación de los defectos, así como por el área de la imagen cubierta durante la convolución. Al generar los filtros digitales mediante la *Arquitectura Espiral Descendente Convolucional* (AEDC) se observa un rápido descenso en su tamaño. a 87% en el primer submuestreo. 75% en el segundo. 64% en el tercero. y así sucesivamente. La figura 7 ilustra los diferentes tamaños de filtros que se generan para la identificación de defectos de tipo E. D. y C.

En la tabla 2 puede observarse una mayor capacidad de muestro del método automático respecto al método manual. tomando en cuenta la Norma ENCE 404 [14] que establece que debe inspeccionarse un área de 500x500 $mm^2$ de una muestra de una hoja de pasta de papel de 800x800 $mm^2$. tomada de uno de cada 7 lotes. de 8 balas. de 410 hojas.

Los resultados obtenidos en una de las pruebas en la etapa de identificación de diferentes tipos de defectos se muestran en la tabla 3.

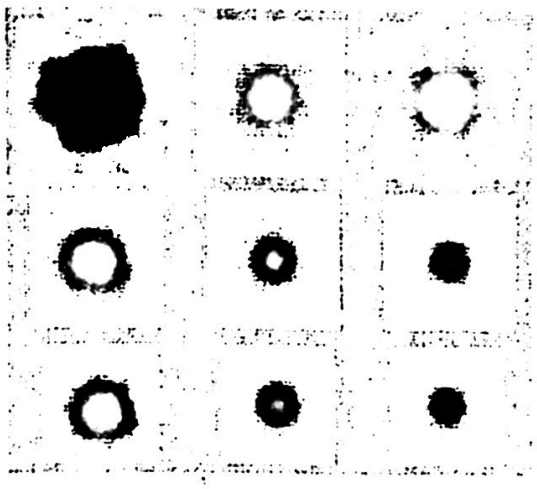


**Fig. 7.** Filtros generados para identificar defectos tipo C, D y E.

**Table 2.** Comparación de métodos

| Método | Área ($mm^2$) | % Inspeccionado | Tiempo (seg) |
|---|---|---|---|
| Manual | 500x500 | 0.0017% | n.a. |
| Automático | 136.5x136.5 | 0.118% | 2.5 |

**Table 3.** Identificación de defectos

| Tipo | N Muestras | % Identificados |
|---|---|---|
| E | 300 | 86% |
| D | 250 | 92% |
| C | 300 | 93% |

## 7 Conclusiones

Se ha presentado una arquitectura que puede ser aplicable en procesos de inspección automática en diversos entornos de fabricación, la cual hace uso de redes neuronales convolucionales para generar filtros digitales. Los filtros generados mediante la arquitectura AEDC han sido probados con imágenes de defectos obtenidas en una fábrica de pasta de papel a velocidad normal de producción, con resultados satisfactorios.

## References

1. Conners, R. *Identifying and locating surface defects in wood*, IEEE Transactions PAMI 5 (6) (1983), pp 573-583.
2. Dewaele, P., Van Gool, L., and Oosterlinchk, A. *Texture Inspection with self-adaptive convolution filters*, Proceedings of the Ninth International Conference on Pattern Recognition, 1, 1988, pp 14-17.
3. Lee, C.S., Choi, C.H., Choi, J.Y., Kim, Y.K. and Choi, S.H., *Feature extraction algorithm based on adaptive wavelet packet for surface detect classification*, IEEE International Conference on Image Processing, 1996, pp. 673-675.
4. Stojanovic, R., Mitropulos, P., Koulamas, C., Karayiannis, Y., Koubias, S., and Papadopoulos, G., *Real-Time Vision-Based System for Textile Fabric Inspection*, Real-Time Imaging 7, Academic Press, pp. 507-518, 2001.
5. Campoy-Cervera, P., Munoz-Garcia, D.F., Pena, D., and Calderon-Martinez, J.A., *Automatic Generation of Digital Filters by NN Based Learning: An application on Paper Pulp Inspection*, IWANN'2001 6th International Work-Conference on Artificial and Natural Neural Networks, June 2001, Proceedings, Part II, pp. 235-245
6. Jain, A.K., Duin, R.P.W., and Mao, J., *Statistical Pattern Recognition: A Review*, IEEE Transactions on Pattern Analysis and Machine Intelligence, January 2000, vol. 22, n 1, pp. 4-37.
7. Fukushima, K., *Neocognitron: A self organizing neural-network model for a mechanism of pattern recognition unaffected by shift in position*, Biological Cybernetics, vol. 36, pp. 193-202, 1980.
8. LeCun, Y., and Bengio, Y., *Convolutional networks for images, speech and time-series*, In M. A. Arbib, editor, The Handbook of Brain Theory and Neural Networks. MIT Press, 1995.
9. Li, B-Q., and Li, B., *Building Pattern Classifiers Using Convolutional Neural Networks*, IEEE International Joint Conference on Neural Networks, 1999. Vol. 5, pp. 3081-3085.
10. LeCun, Y., Bottou L., Bengio Y., and Haffner P., *Gradient-Based learning applied to document recognition*, Proceedings of the IEEE, Nov. 1998, pp. 1-45.
11. Lawrence, S., Lee, Giles C., and Chung Tsoi, A., *Convolutional neural networks for face recognition*, 1996. Proc. CVPR '96 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 217-222, 1996.
12. Lawrence, S., Lee, Giles C., Chung Tsoi, A., and Back, A.D., *Face recognition: A convolutional neural-network approach*, IEEE Transactions on Neural Networks, vol. 8, n 1, pages 98-113, 1997.

13. Sahiner, B., Chan, H.P., Petrick, N., Wei, D., Helvie, M.A., Adler, D.D., and Goodsitt, M.M., *Classification of Mass and Normal Breast Tissue: A Convolution Neural Network Classifier with Spatial Domain and Texture Images*, IEEE Transactions on Medical Imaging, Vol. 5, pp. 598-610, Oct. 1996.
14. Normas ENCE: *Technical Specification of the Product*, Empresa Nacional de Celulosas, S.A., 1994.

# Incremental Construction of LSTM Recurrent Neural Network

Sabrine Ribeiro René Alquézar
Universitat Politècnica de Catalunya
Departament de Llenguatges e Sistemes Informàtics
Campus Nord, Jordi Girona Salgado, 1-3, C6-201,
08034, Barcelona, España.
{eslopes,alquezar}@lsi.upc.es

**Abstract**

Long Short-Term Memory (LSTM) is a recurrent neural network (RNN) that uses structures called memory blocks to allow the net remember significant events distant in the past input sequence in order to solve long time lag tasks, where other RNN approaches fail. Throughout this work we have performed experiments using LSTM networks extended with growing abilities, which we call GLSTM. Four methods of training growing LSTM have been compared. These methods include cascade and fully connected hidden layers as well as two different levels of freezing previous weights in the cascade case. GLSTM has been applied to a forecasting problem in a biomedical domain, where the input/output behavior of five controllers of the Central Nervous System control has to be modelled. We have compared growing LSTM results against other neural network approaches, including our previous work where conventional LSTM was applied to the task at hand.

## 1 Introduction

In this work, the application of four growing methods are studied to improve the learning of a recurrent neural network (RNN) called Long Short-Term Memory (LSTM) in signal prediction tasks. LSTM is a recurrent network that uses structures called memory blocks to allow the net remember significant events distant in the past input sequence in order to solve long time lag tasks, where other RNN approaches fail. LSTM have been quite successfully applied to standard benchmarks related to classification problems [GS00b, HS97], and more recently to signal forecasting problems [RA01, GS00a].

A key issue in neural network (NN) design is determining the number of hidden units required to perform input/output mapping with satisfactory performance. In recent years, attempts have been made to build NNs incrementally

in an automatic way. The techniques used to solve this problem are called constructive or growing methods.

Growing methods are designed to automate the process of determining the network topology, by modifying both the weights and connectivity of the network during learning. Thus, these methods eliminate the need to guess the network size what makes the use of NNs more user–friendly.

In the latest years a great deal of effort has been directed towards finding efficient growing algorithms [KY97]. Many RNN growing methods have been proposed. The Dynamic Node Creation [Ash89] is a method that adds a new hidden unit when the average error curve begins to flatten out too quickly. A more sophisticated method in order to train the network that uses sophisticated nonlinear least squares and quasi-Newton optimization techniques can be found in [Bel94]. In [HRM+94] an approach called Projection Pursuit Learning Network is modelled as a one hidden layer Multi-layer perceptron that learns neuron by neuron, and layer by layer cyclically after all the training patters are presented. The most used neural network growing method is the Cascade Correlation (CC) [Fah91], that combines two key ideas. The former is the cascade architecture, in which hidden units are added only one at a time. The latter is the learning algorithm, which create and installs the new hidden units by maximizing the correlation of their outputs with the residual error. In [GAP98] a sequential orthogonal approach to the building and training of single hidden layer neural networks is described. A more recent method called Sequential Approximation with Optimal Coefficients and Interacting Frequencies [RA02] has been reported that combines two key ideas. The first one is the optimization of the coefficients (or output layer weights), which provide the linear part of the approximation. The second one is the flexibility to choose the frequencies (or hidden layer weights), which provide the non-linear part.

Regarding to RNNs the most popular growing method used up to now is the Recurrent Cascade-Correlation (RCC) [Fah91] which builds an architecture that adds recurrent operation to the Cascade-Correlation architecture. Therefore, in RCC each hidden unit is provided with a single weighted self–recurrent link that feeds back its own activation value on the previous time step. As an alternative to the original RCC architecture, [KTA95] introduced the Parallel-modular RCC that is trained with natural connectionist glue, which is a concept for modularity and scaling in large phonemic neural networks [AW90, KL90]. Another more recent effort attempting to learn weights and topology of neural nets is the work of Angeline [ASP94]. The GeNeralized Acquisition of Recurrent Links Algorithm is an evolutionary algorithm that non-monotonically constructs recurrent networks to solve a given task.

Throughout this work we have performed experiments using LSTM networks extended with growing abilities, which we call GLSTM. Four methods of training growing LSTM have been compared. GLSTM has been applied to a forecasting problem in a biomedical domain, where the input/output behavior of five controllers of the Central Nervous System control has to be modelled. We have compared growing LSTM results against those reported in [BVA98] using other NN approaches and our previous work applying conventional LSTM

[RA01] to the task at hand.

In the following section we describe the LSTM architecture. In section 3 we introduce the growing LSTM approaches. In section 4 we show the case of study. In section 5 the experimental methodology used is described. In section 6 the obtained results are presented. Finally, some conclusions are given in section 7.

# 2 LSTM Recurrent Neural Network

LSTM [HS97] belongs to a class of recurrent networks that has time-varying inputs and targets. That is, points in the time series or input sequence are presented to the network one at a time. The network can be asked to predict the next point in the future or by classify the sequence or to perform some dynamic input/output association. Error signals are either generated at each point of the sequence or at the end of the sequence.

## 2.1 LSTM Structure

A fully connected LSTM architecture is a three-layer neural network composed of an input layer, a hidden layer and an output layer. The hidden layer has a feedback loop to itself, i.e., at time step $t$ of a sequence with $n$ time steps, presented to the network, the hidden layer receives as input the activation values of the input layer and the activation values of the hidden layer at time step $t-1$. Fig. 1 illustrates a LSTM with a fully connected hidden layer consisting of two memory blocks, each one consisting of two cells. The LSTM showed has an input dimension of two and an output dimension of one. Only a limited subset of connections are shown.

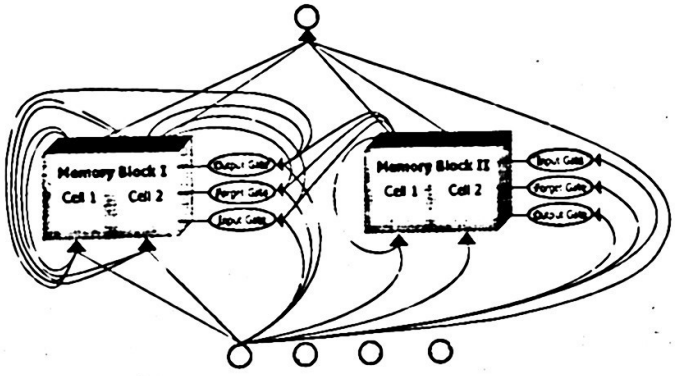



Figure 1: Example of LSTM net consisting of 4 inputs units, 1 output unit and 2 memory blocks of size 2. Only a limited subset of connections are shown.

The basic unit in the hidden layer is known as a memory cell block. A memory cell block (Fig. 2) consists of $S$ memory cells and three multiplicative gates, called the input gate. output gate and forget gate. Each memory cell has at its core a recurrently self-connected linear unit called *Constant Error*

*Carousel* (CEC), whose activation is called the cell state. The CECs solve the vanishing error problem: in the absence of a new input or error signals to the cell, the CEC's local error back flow to remains constant, neither growing nor decaying. Input and output gates regulate write and read access to a cell whose state is denoted $S_c$. The CEC is protected from both flowing activation and backward flowing error by the input and output gates respectively. When gates are closed (activation around zero), irrelevant inputs and noise do not enter the cell, and the cell state does not perturb the remainder of the network. The forget gate feed the self–recurrent connection with its output activation and is responsible for do not allow the internal state values of the cells grow without bound by resetting the internal states $S_c$ as long as it needs. In addition to the self–recurrent connection, the memory cells receive input from input units, other cells and gates.

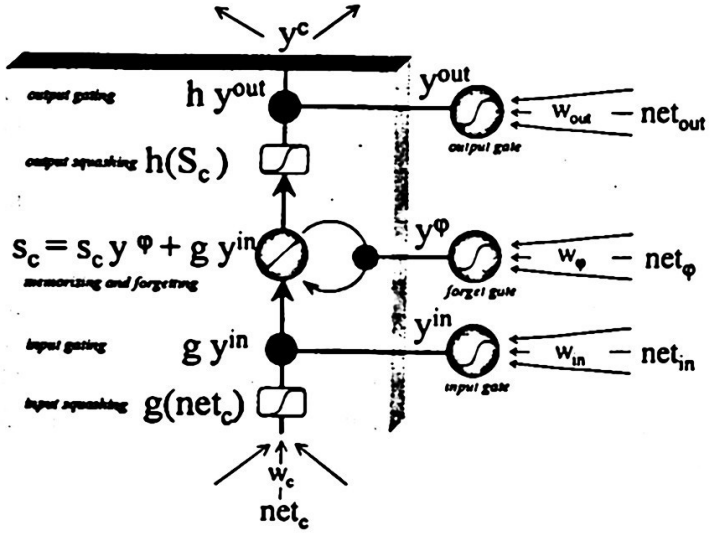



Figure 2: The standard LSTM cell with a recurrent self-connected connection and its respective gates.

While the cells are responsible for maintaining information over long periods of time, the responsibility for deciding what information to store, and when to apply that information lies with the input and output gate units, respectively.

A single step involves the update of all units (forward pass) and the computation of error signals for all weights (backward pass). The equations that describe the dynamics of the LSTM network can be found in [7].

## 2.2 Learning

LSTM's backward pass [HS97] is basically a fusion of slightly modified truncated back-propagation through time (BPTT) [WP90], which is obtained by truncating the backward propagation of error information, and a customized version of RTRL [RF87] which properly takes into account the altered (sigma-pi-like) dynamics caused by input and output gates (see details in [HS97]).

Output units use BPTT; output gates use a truncated version of BPTT. However, weights to the cells and forget gates use a truncated version of RTRL. Truncation means that all errors are cut off once they leak out of a memory cell or gate, although they do serve to change the incoming weights. The effect is that the CECs are the only part of the system through which errors can flow forever. So, the error signals flowing out of the CEC and the multiplicative gates are truncated after they are used to update the incoming, weighted connections.

# 3 Growing LSTM

Aiming to improve the learning abilities of LSTM neural net, this work deals with a version of LSTM where the network topology is incrementally adapted by adding new memory blocks. We call this version Growing LSTM (GLSTM).

In growing algorithms, many heuristics can be used to guide the search in the possible solution space. An important problem is how to set the weight connections of a newly added block. LSTM starts training with just one memory block and grows by inserting blocks, one at time, on the hidden layer in two basic ways: *cascade* and *fully connected*, that are discussed posteriorly.

In both architectures, each new memory block receives a connection from each of the network's original inputs. However the connections that comes from other blocks changes according to the architecture used. Every new block has the same number of cells than the first initial block.

## 3.1 Cascade Growing LSTM

In the recurrent cascade architecture, the new units are added one at a time and each new unit receives inputs from every preexisting units and also from itself. Thus, carrying out this concept on LSTM, each new memory block in addition to have a self-connect connection will also receives a link from each of the network's original inputs and also from every memory cell on preexisting blocks, as can be seen in Fig. 3.

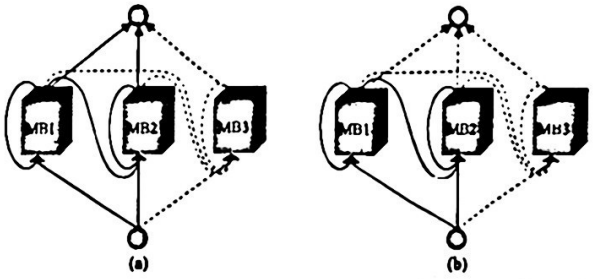



Figure 3: Cascade growing. Whereas in (a) all preexisting weights are frozen, in (b) only weights arriving at already existing blocks are frozen, letting the hidden-output weights free.

In this work, the weights arriving and leaving from the memory blocks were

frozen in two different ways. In the first one, all the preexisting weights in the whole net were frozen, leaving only the new block weights to be trained. In the second way, previous weights were frozen except for those of the output units, that remained trainable.

Fig. 3 illustrates these two configurations. In both, it can be supposed that there exist just two memory blocks (MB1 and MB2) and a new memory block (MB3) will be added. For the first configuration (Fig. 3(a)), the weights arriving at already existing blocks (MB1 and MB2) are kept frozen (solid lines) and those arriving at new memory block (MB3) are trained repeatedly (dashed lines). The output unit weights are configured in a similar way. In the second configuration (Fig. 3(b)) the same procedure is carried out, but now all hidden-output weights are modified during subsequent training.

A third configuration, not shown in the figure, is not to freeze any weight in the net, so the preexisting weight connections still can be trained further after the addition of a new memory block.

### 3.2 Fully Connected Growing LSTM

In addition to the fact that the new memory block receives connections from every preexisting blocks, as occurs in the cascade architecture, in the fully connected architecture the preexisting blocks also receives weight connections from each new added block. As can be seen in Fig. 4 no weight in the net is frozen during training, since every cell of each memory block receives new connections after adding a new block.

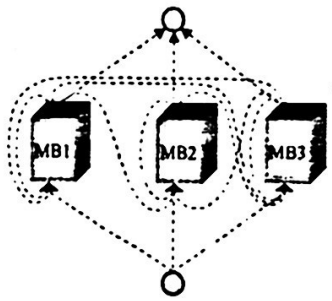



Figure 4: Fully GLSTM.

## 4 Case of Study

The human cardiovascular system is composed of the hemodynamical system and the Central Nervous System (CNS) control. In this work we try to model the latter by capturing its input/output dynamic behavior.

The CNS generates the regulating signals for the blood vessels and the heart, and it is composed of five controllers: the heart rate controller (HRC), the peripheric resistance controller (PRC), the myocardiac contractility controller

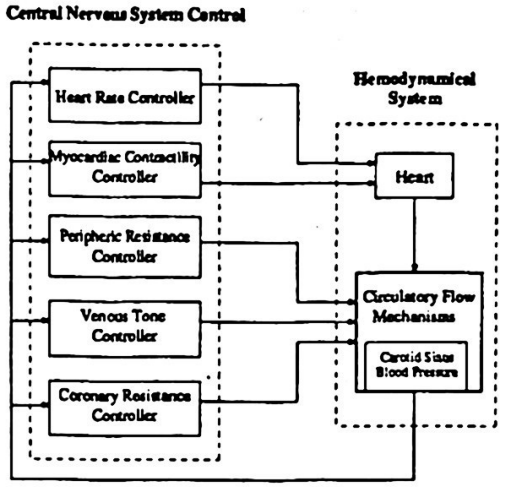



Figure 5: Simplified diagram of the cardiovascular system model, composed of the hemodynamical system and the CNS control.

(MCC), the venous tone controller (VTC), and the coronary resistance controller (CRC). A simplified diagram of the cardiovascular system is shown in Fig. 5. All of these controllers are single–input/single-output (SISO) systems driven by the same input variable, namely the carotid sinus pressure. Although the Carotid Sinus Pressure is not easily measurable, it can be extracted from the differential equation model describing the hemodynamics of the cardiovascular system [Val93]. The five output variables of the controller models are not even amenable to a physiological interpretation, except for the heart rate controller variable, which is the inverse heart rate, measured in seconds between beats.

Whereas the structure and functioning of the hemodynamical system are well known and a number of quantitative models, mostly based on differential equations, have been developed, the functioning of the central nervous system control is of high complexity and still not completely understood. Although some differential equation models for the central nervous system have been postulated [SS74], these models are not accurate enough, and therefore, the use of other modelling approaches like neural networks may offer an interesting alternative for capturing the behavior of the CNS control [LK90].

# 5 Experimental Methodology

## 5.1 Prediction Strategy

Prediction tasks involve the use of currently available input and output points to predict a future output point, i.e., given two finite sequences $x(1), x(2), x(3), \ldots, x(t)$ of input signal values and $y(1), y(2), y(3), \ldots, y(t-1)$ of output signal values, predict the value $y(t-1+T)$ of the output signal.

To prepare the data conveniently, we have replaced the original target output

$y(t)$ by the difference between the $y(t)$ output value and the previous value $y(t-1)$ multiplied by a scaling factor $fs$, so that the target is calculated as $t_g(t) = fs * (y(t) - y(t-1)) = \Delta y(t) * fs$. $fs$ scales $\Delta y(t)$ between $-1$ and 1.

Stepwise and iterated predictions are made. In single-step prediction the network predicts the next output point, $y(t)$, after being fed with the current input $x(t)$ and the last known value of the output, $y(t-1)$. In this case, $T = 1$. It should be noted that, both the inputs and the desired response are provided from the known training points.

During iterated prediction with $T = n$ the output is clamped to the $y-$input and the predicted values are fed back $n$ times, i.e. the $y-$input samples are progressively substituted by the output of the network. This closed loop system is illustrated in Fig. 6.

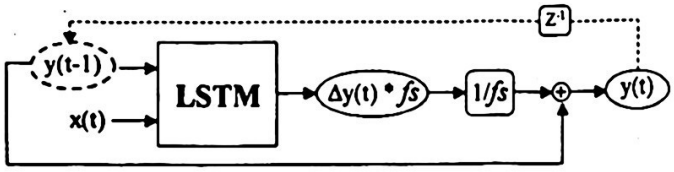



Figure 6: Setup for the output signals.

## 5.2 The Data

The data used in the training and test phases are composed of the five controllers mentioned in §4: HRC, PRC, MCC, VTC and CRC, that were recorded with a sampling rate of 0.12 seconds from simulations of a purely differential equation model. This model had been tuned to represent a specific patient suffering from coronary arterial obstruction, by making the four different physiological variables (right auricular pressure, carotid pressure, coronary blood flow, and heart rate) of the simulation model agree with the measurement data taken from the patient.

The training set consists of 1,500 data points for each controller. Each trained network was validated by using it to forecast six data sets that had not been employed in the learning process. Each one of these six test sets (for each controller), with a size of 300 points each, contains signals representing specific morphologies, allowing the validation of the model for different system behaviors.

## 5.3 Experimental Setup

The GLSTM network architecture used is made up of an input layer with 2 inputs units, an output layer with 1 output unit and a hidden layer consisting of memory cell blocks of size 1. A new memory block is added to the hidden layer when the previous configuration has been trained at least 1000 epochs and the mean of the error in the last 10 epochs has not improved the previous error

mean up to 15 memory blocks or a maximum number of epochs, which depends on the task.

After preliminary experiments, the bias weights for input and output gates in successive blocks were fixed as: $-0.5$, $-1.0$, $-1.5$, and so forth. The initialization of output gates pushes initial memory cells activations towards zero, whereas that of the input gates prevents memory cells from being modified by incoming inputs. As training progresses, the biases become progressively less negative, allowing the serial activation of cells as active participants in the network computation.

- The forget gates were initialized with symmetric positive values: $+0.5$ for the first block, $+1.0$ for the second, $+1.5$ for the third, and so forth. The bias initialization must be positive in this case, since it prevents the cells from forgetting everything [GS00a].

For the cell's input squashing function $g$ different configurations were used, depending on the controller. More specifically, for the HRC and CRC training sets the antisymmetric logarithm function [Alq97] was used, whereas for VTC, MCC and PRC controllers $g$ was the logistic sigmoid in $[-1, 1]$. As regard to the output squashing function $h$ and the activation function of the output unit, they were fixed as the linear identity function.

After some preliminary experiments, the number of epochs chosen to stop training was 5000 for HRC, MCC and VTC controllers, 2500 for PRC and 2000 for CRC control. The learning rate has been changed according to the dataset to be learned, but, in essence, it took on values from 0.01 to 0.025. The momentum parameter for each controller was 0.0 for HCR, PRC and CRC training sets and 0.5 for VTC and MCC training sets.

The error measure is given by the normalized mean square error (NMSE), in percent, between the predicted output value and the target value, $y$:

$$NMSE = \frac{E[(y(t) - \hat{y}(t))^2]}{y_{\text{var}}} \cdot 100\% \tag{1}$$

where $y_{var}$ is the variance of $y$. During training, the above NMSE error is used to determine when to finish the learning process, as explained earlier.

# 6 Results

In this section, two kinds of analysis are carried out. In the former, comparisons among different growing architectures are made and in the latter, GLSTM's results obtained throughout the present study are showed and compared with those reported in previous studies on the same task using neural net approaches.

Table 1 displays the outcomes obtained by each controller on the training and test set using the different growing architectures and configurations (see §3). *Fully* stands for fully connected growing without freezing weights. In order to investigate the effects of freezing weights, we tested three different choices for cascade growing LSTM. *Cascade Freezing* refers to the configuration illustrated by Fig. 3(a), *Cascade Not-Freezing* refers to growing cascaded without freezing

| | Fully | | Cascade Freezing | | Cascade Not-Freezing | | Cascade Trainable-Out | |
|---|---|---|---|---|---|---|---|---|
| | Train | Test | Train | Test | Train | Test | Train | Test |
| HRC | 4.85 | 3.99 | 3.09 | 4.11 | 3.26 | 4.15 | 2.11 | 3.27 |
| PRC | 0.12 | 0.69 | 0.13 | 0.66 | 0.17 | 1.33 | 0.09 | 0.61 |
| MCC | 0.39 | 1.11 | 0.14 | 0.99 | 0.11 | 1.02 | 0.09 | 0.98 |
| VTC | 0.08 | 0.96 | 0.10 | 0.96 | 0.18 | 1.05 | 0.08 | 0.96 |
| CRC | 0.22 | 0.37 | 0.11 | 0.35 | 0.18 | 0.38 | 0.09 | 0.26 |
| Av. | 1.35 | 1.42 | 0.71 | 1.41 | 0.78 | 1.58 | 0.48 | 1.21 |

Table 1: Average of three trials NMSE errors (in percent) for training and test sets of each controller using different growing architectures.

any weights, and *Cascade Trainable-Out* refers to Fig. 3(b) scheme. The results showed are the average of three different training trials using different initial random weights. As it can be seen, the results achieved by *Cascade Trainable-Out* is far better than those obtained by the other configurations. From now on, GLSTM results this configuration.

Next, refer to the outcomes accomplished in [RA01], which applies LSTM to the task at hand, are used as a baseline to illustrate GLSTM improvements on LSTM. Moreover, the results reported in [BVA98] are also recalled for comparison purposes. In [BVA98] four different approaches were performed over the task at hand, where three of them are TDNNs [KL90, HKP91] that differ in the training method used: a standard backpropagation algorithm (TDNN-BP), a hybrid procedure composed by repeated cycles of simulated annealing coupled with conjugate gradient algorithm (TDNN-AC), and a genetic algorithm (TD-HNN). This last network uses indeed a different neuron model based on a similarity computation. The other one is a RNN approach, an ASLRNN net, similar to Elman's SRN net, except that is trained by a true gradient descent algorithm that does not truncate error propagation backwards in time.

Concerning to stepwise prediction results, Table 2 shows the average NMSE errors (in percent) of each of the above mentioned architectures against those yielded by GLSTM. For each controller, three different training trials using different random weight initialization in the range $[-0.1, 0.1]$ were carried out. The net built in each trial was applied to the six test sets associated with the controller.

In order to compare the long-term prediction results, where the whole test set is attempted to be predicted ($T = 300$), Table 3 shows the average NMSE errors for different dynamic input/output architectures of three independent training trials for the six test sets of each controller. As can be seen, GLSTM outperforms in most cases those results achieved by ASLRNN, TDNN-BP and LSTM.

Finally, Fig. 7 shows the error curve of LSTM trained with 1, 2 and 3 memory blocks and LSTM trained with cascade growing method (GLSTM) for the HRC controller. The black points showed in the graph indicate the epoch in which a new memory block was added to the net. As it can be observed,

| | TD-HNN | | TDNN-BP | | TDNN-AC | | ASLRNN | | LSTM | | GLSTM | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Train | Test | Train | Test | Train | Test | Train | Test | Train | Test | Train | Test |
| HRC | 0.11 | 0.18 | 1.15 | 1.52 | 0.15 | 0.13 | 1.63 | 1.91 | 2.16 | 3.41 | 2.11 | 3.27 |
| PRC | 0.09 | 0.12 | 0.94 | 1.27 | 0.26 | 0.14 | 0.84 | 1.10 | 0.25 | 0.65 | 0.09 | 0.61 |
| MCC | 0.03 | 0.06 | 0.81 | 1.33 | 0.09 | 0.08 | 0.71 | 1.18 | 0.19 | 1.04 | 0.09 | 0.98 |
| VTC | 0.03 | 0.06 | 0.81 | 1.33 | 0.09 | 0.08 | 0.71 | 1.18 | 0.19 | 1.01 | 0.08 | 0.96 |
| CRC | 0.10 | 0.11 | 0.47 | 0.66 | 0.03 | 0.04 | 0.41 | 0.53 | 0.18 | 0.31 | 0.09 | 0.26 |
| Av. | 0.07 | 0.11 | 0.84 | 1.22 | 0.12 | 0.09 | 0.86 | 1.18 | 0.59 | 1.28 | 0.48 | 1.21 |

Table 2: Average NMSE errors (in percent) for step-wise prediction on training and test sets.

| | | TDNN-BP | ASLRNN | LSTM | GLSTM |
|---|---|---|---|---|---|
| HRC | Data Set 1 | 24.31 | 28.25 | 32.81 | 25.52 |
| | Data Set 2 | 7.47 | 8.62 | 28.66 | 21.36 |
| | Data Set 3 | 13.48 | 16.77 | 28.40 | 19.52 |
| | Data Set 4 | 6.87 | 8.16 | 23.25 | 19.23 |
| | Data Set 5 | 32.12 | 38.24 | 31.70 | 24.73 |
| | Data Set 6 | 7.86 | 9.80 | 27.86 | 33.96 |
| Average Error | | 15.35 | 18.31 | 28.78 | 24.05 |
| PRC | Data Set 1 | 58.15 | 50.07 | 12.30 | 6.68 |
| | Data Set 2 | 17.80 | 16.11 | 17.29 | 6.55 |
| | Data Set 3 | 41.56 | 36.89 | 12.15 | 6.53 |
| | Data Set 4 | 29.09 | 26.97 | 7.15 | 35.00 |
| | Data Set 5 | 34.73 | 38.54 | 21.15 | 71.47 |
| | Data Set 6 | 21.22 | 18.40 | 14.22 | 5.48 |
| Average Error | | 33.76 | 31.16 | 14.04 | 21.95 |
| MCC | Data Set 1 | 41.72 | 55.83 | 27.48 | 17.16 |
| | Data Set 2 | 20.92 | 17.18 | 42.88 | 26.26 |
| | Data Set 3 | 40.22 | 35.60 | 26.26 | 17.60 |
| | Data Set 4 | 39.80 | 42.08 | 14.44 | 33.88 |
| | Data Set 5 | 34.32 | 36.87 | 16.15 | 19.27 |
| | Data Set 6 | 27.20 | 23.38 | 30.91 | 20.01 |
| Average Error | | 34.04 | 35.16 | 26.35 | 22.36 |
| VTC | Data Set 1 | 41.68 | 54.25 | 27.01 | 11.36 |
| | Data Set 2 | 20.90 | 16.93 | 35.43 | 9.99 |
| | Data Set 3 | 40.22 | 35.68 | 10.16 | 10.02 |
| | Data Set 4 | 39.80 | 41.86 | 14.58 | 32.37 |
| | Data Set 5 | 34.41 | 36.77 | 15.50 | 40.19 |
| | Data Set 6 | 27.22 | 23.12 | 29.90 | 9.41 |
| Average Error | | 34.04 | 34.77 | 22.02 | 18.89 |
| CRC | Data Set 1 | 147.73 | 148.65 | 3.70 | 6.67 |
| | Data Set 2 | 28.35 | 36.17 | 4.63 | 15.75 |
| | Data Set 3 | 84.35 | 83.75 | 3.00 | 4.29 |
| | Data Set 4 | 4.69 | 4.49 | 5.48 | 6.59 |
| | Data Set 5 | 56.20 | 58.50 | 72.29 | 44.01 |
| | Data Set 6 | 12.32 | 11.16 | 2.99 | 4.64 |
| Average Error | | 55.69 | 57.12 | 14.73 | 13.65 |

Table 3: Average NMSE errors for long-term prediction

GLSTM finds a certain stability by adding memory blocks incrementally and keeping previous hidden-layer weights frozen.

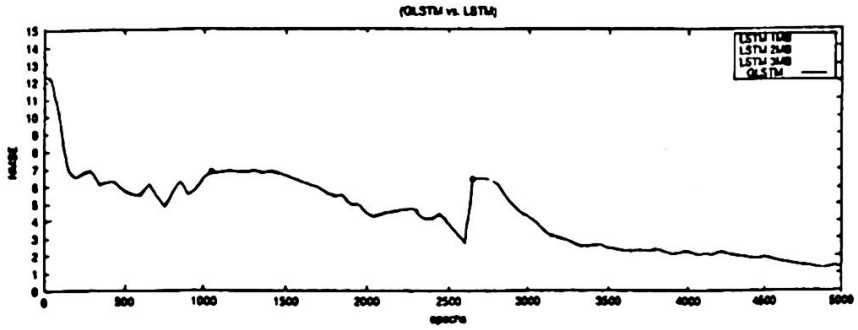


Figure 7: GLSTM vs. LSTM prediction.

# 7 Conclusions

The need to fix the size and topology of a network before its training phase is one of the most important practical problems when using common neural network approaches. Therefore, the use of incremental growing methods that build the network automatically are seen as a practical way of making the use of neural networks less complex. This is specially desirable in the case of recurrent neural networks, where the topology can be extremely complex.

Most of the previous work on NN growing methods has dealt with feedforward NNs and only a few techniques have been proposed for some particular types of recurrent NNs. In this work, we have studied the incremental construction of the LSTM net, which is maybe the more powerful RNN architecture proposed so far. Some different alternatives have been identified and tested on a signal forecasting task concerning the learning of models for the Central Nervous System Control. These include cascade and fully connected hidden layers as well as different levels of freezing previous weights in the cascade case.

It has been shown that, in addition to remove the need to fix the number of hidden units in advance, the growing LSTM can yield a better performance both in the training and test phases. Moreover, the behavior of the error minimization during the training phase appears to be more stable when using Growing LSTM with frozen weights.

Nevertheless, the experiments carried out here need to be complemented with new studies on many other different problems. In addition, the final objective of our work is not only to build LSTM nets incrementally, but rather to develop a methodology for the incremental construction of recurrent NNs for prediction tasks that can combine LSTM units (memory blocks) with other RNN architectures or even with Time-Delay Neural Networks. The underlying idea is to build a RNN as simplest as possible for a given problem and only adding more sophisticated elements (such as LSTM memory blocks) in a

parsimonious way when strictly required to improve the approximation and/or generalization performance.

# References


[Alq97] R. Alquézar. *Symbolic and Connectionist Learning Techniques for Grammatical Inference.* PhD thesis, Technical University of Catalonia, 1997.

[Ash89] T. Ash. Dynamic Node Creation in Backpropagation Networks. *Connection Science*, 1(4):365–375, 1989.

[ASP94] P.J. Angeline, G.M. Saunders, and J.B. Pollack. An Evolutionary Algorithm that Constructs Recurrent Neural Networks. *IEEE Transactions on Neural Networks*, 5(1):54–64, 1994.

[AW90] K. Shikano A. Waibel, H. Sawai. Consonant Recognition by Modular Construction of Large Phonemic Time-Delay Neural Networks. In *Proc of the Int. Conf. on Acoust, Speech, and Signal Processing*, volume S3.9, pages 112–115, 1990.

[Bel94] M.G. Bello. Enhanced Training Algorithms, and Integrated Training/Architecture Selection for Multilayer Perceptron Networks. *IEEE Transactions on Neural Networks*, 3(6):864–875, 1994.

[BVA98] L. Belanche, J.J. Valdés, and R. Alquézar. Fuzzy Heterogeneous Neural Networks for Signal Processing. In *Proc. of Int. Conf. of Artificial Neural Network*, 1998.

[Fah91] Fahlman, S.E. The Recurrent Cascade-correlation Architecture. *Advances in Neural Information Processing Systems*, 3:190–196, 1991.

[GAP98] A.H. Gee, S.V.B. Aiyer, and R.W. Prager. A Sequential Learning Approach for Single Hidden Layer Neural Networks. *Neural Computation*, 11:65–80, 1998.

[GS00a] F.A. Gers and J. Schmidhuber. Applying LSTM to Time Series Predictable Through Time-Window Approaches. Technical Report 22-00, IDSIA, 2000.

[GS00b] F.A. Gers and J. Schmidhuber. Recurrent Nets that Time and Count. In *Proc. of the Int. Joint Conf. on Neural Networks*, page 273, Como, Italy, 2000.

[HKP91] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation.* Addison Wesley, Redwood City, CA, 1991.

[HRM+94] J.N. Hwang, S.R. Ray, M. Maechler, D. Martin, and J. Schimert. Regression Modelling in Backpropagation and Projection Pursuit Learning. *IEEE Transactions on Neural Networks*, 5(3):54–64, 1994.

[HS97] S. Hochereiter and J. Schmidhuber. LSTM can Solve Hard Long Time Lag Problems. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 473–479, Cambridge, MA, 1997. MIT Press.

[KL90] G. Hinton K. Lang, A. Waibel. A TimeDelay Neural Network Architecture for Isolated Word Recognition. *Neural Computation*, 3:23–34, 1990.

[KTA95] I. Kirschning, H. Tomabechi, and J.I. Aoe. A Parallel Recurrent CascadeCorrelation Neural Network with Natural Connectionist Glue. In *Proc. of the Int. Conf. on Neural Networks*, volume 2, pages 953–956, Perth, Australia, 1995.

[KY97] T.Y. Kwok and D.Y. Yeung. Constructive Algorithms for Structure Learning in Feedforward Neural Networks for Regression Problems. In *IEEE Transactions on Neural Networks*, volume 8, pages 630–645, 1997.

[LK90] A. Law and D. Kelton. *Simulation Modelling and Analysis*. McGraw Hill, New York, 1990.

[RA01] S. Ribeiro and R. Alquézar. A Comparative Study on a Signal Forecasting Task applying Long Short-Term Memory (LSTM) Recurrent Neural Networks. In *VI Simpósio Ibero-Americano de Reconhecimento de Padrões*, pages 487–495, Florianopolis, Brazil, 2001.

[RA02] E. Romero and R. Alquézar. A New Incremental Method for Function Appoximation using Feed-forward Neural Networks. In *Proc. of the Int. Joint Conf. on Neural Networks*, Honolulu, Hawaii, 2002.

[RF87] A. J. Robinson and F. Fallside. The utility driven dynamic error propagation network. Technical Report CUED/F-INFENG/TR.1, Engineering Department, Cambridge University, 1987.

[SS74] H. Suga and K. Sagawa. Instantaneous Pressure-volume Relationships and their Ratio in the Excised, Supported Canine Left Ventricle. *Circulation Research*, 53:117–126, 1974.

[Val93] M. Vallverdú. *Modelado y Simulación del Sistema de Control Cardiovascular en Pacientes con Lesiones Coronarias*. PhD thesis, Technical University of Catalonia, 1993.

[WP90] R.J. Williams and J. Peng. An Efficient Gradient-based Algorithm for on-line Training of Recurrent Network Trajectories. *Neural Networks*. 2:491–501, 1990.

# Agrupador de complejos QRS basado en Aprendizaje Competitivo Selectivo

Pablo Aguirre, Juan Cardelino, Nicolás Loeff & Franco Simini
Instituto de Ingeniería Eléctrica,
Facultad de Ingeniería,
Universidad de la República.

**Resumen**

El complejo QRS es una de las señales más importantes presentes en el ECG en cuanto a la detección de anomalías cardíacas. El algoritmo presentado descompone cada complejo en sus primeros componentes principales y le incorpora información temporal. Los complejos QRS son agrupados en este espacio utilizando el algoritmo de aprendizaje competitivo selectivo (Competitive Selective Learning, CSL) el cual está basado en un resultado óptimo de cuantización vectorial. Este procedimiento presentó una tasa de clasificación del 99.0% al agrupar los complejos de la base de datos MIT/BIH en 25 grupos.

**Palabras clave:** complejo QRS, agrupamiento, CSL, aprendizaje competitivo, base de datos MIT/BIH.

## 1. Introducción

Varias décadas de investigación han hecho de la electrocardiografía una disciplina básica para el diagnóstico de desórdenes cardíacos. Debido a su simplicidad, bajo costo y al ser una técnica no-invasiva, sigue siendo ampliamente utilizada a pesar de la aparición de técnicas más modernas.

En los últimos años el descenso progresivo en el costo del procesamiento digital de alta performance ha impulsado la introducción de técnicas de análisis automático de Electrocardiogramas (ECG). Las mismas son fundamentales en diversos entornos, como por ejemplo en los centros de cuidados intensivos donde es esencial la detección y clasificación en tiempo real de arritmias cardíacas. En estos casos la clasificación incorrecta o tardía de la señal puede significar una amenaza a la vida del paciente. Otro campo de aplicación es el análisis de señales de larga duración, el cual puede resultar tedioso y consumir mucho tiempo de un especialista. Esto lleva a que el mismo se concentre en algunos intervalos determinados, en desmedro del resto de la señal. La automatización permite seleccionar los segmentos relevantes para que el especialista pueda estudiarlos con detenimiento, y prestar menos atención a los tramos rutinarios. Para que esto sea posible, es necesario que la performance del algoritmo sea muy alta para que el resultado sea confiable.

El presente trabajo se centrará en el estudio del complejo QRS [1] que esta asociado con la actividad eléctrica ventricular. El mismo contiene importante información clínica, por ejemplo su morfología sufre cambios importantes en los latidos ventriculares. También está presente en la mayoría de los latidos y su relación señal-ruido es la más alta de todas las ondas presentes en la señal.

El principal problema a los que se enfrenta un analizador de ECG es la gran variabilidad de las morfologías, no sólo entre pacientes sino también debido a movimientos del paciente, modificaciones de la características de conducción eléctrica del cuerpo, etc.

Además, la señal de ECG se ve contaminada por varias fuentes de ruido, tanto externas (interferencia de la red de alimentación, movimiento de los electrodos) como de origen biológico (el movimiento muscular que provoca interferencia en alta frecuencia y la respiración que provoca un desplazamiento de la línea base). Es por eso que debe tomarse un cuidado especial en la elección de técnicas adecuadas

[1] El latido cardíaco tiene una expresión bio-eléctrica compuesta por varias ondas, llamadas P, Q, R, S y T respectivamente. En este trabajo se utiliza la expresión *complejo QRS* como referencia a la señal formada por las tres ondas centrales de cada latido.

| Código | Explicación | Código | Explicación |
|---|---|---|---|
| ! | *flutter* ventricular | R | Bloqueo de rama derecha |
| / | Latido de marcapaso | S | Latido supraventricular prematuro o ectópico (atrial o nodal) |
| ? | No clasificado durante el entrenamiento | a | Prematuro aberrante atrial |
| A | Latido prematuro atrial | e | Escape atrial |
| E | Escape ventricular | f | Fusión entre Normal y Paced |
| B | Bloqueo de rama (sin especificar) | j | Escape nodal |
| F | Fusión entre Normal y Ventricular | n | Escape suapraventricular (atrial o nodal) |
| J | Latido prematuro nodal | r | Prematuro ventricular R-sobre-T |
| N | Latido normal | L | Bloqueo de rama izquierda |
| V | Prematuro ventricular | Q | Inclasificable |

Cuadro 1: Códigos de anotaciones de la base MIT-BIH y sus explicaciones.

de acondicionamiento de la señal (pre-procesamiento), ya que la calidad de la señal a la entrada del clasificador tiene impacto directo sobre el desempeño del mismo.

El enfoque propuesto en este trabajo consiste en la extracción de cada complejo QRS un vector n-dimensional de características que describen al mismo. Este vector se obtiene truncando el desarrollo en componentes principales de la señal, y agregando información temporal del mismo. De esta manera se logra representar los latidos en un espacio conveniente para el agrupamiento, reducir el costo computacional y lograr una mayor inmunidad al ruido. Los vectores así obtenidos son agrupados en 25 clases utilizando una variante del aprendizaje competitivo (CL), desarrollada por Ueda y Nakano ([1]), denominada aprendizaje competitivo selectivo (CSL). Este algoritmo resuelve la muchos de los problemas usuales del CL, principalmente la convergencia a mínimos locales, y por lo tanto, en general, presenta un mejor desempeño. La validación del algoritmo se realiza sobre la base de datos de arritmias MIT-BIH ([2]). A continuación se presenta una breve descripción de esta.

### Base de datos MIT-BIH

La misma fue construida a partir de un conjunto de unos 4000 registros Holter. Consta de 48 registros de aproximadamente 30 minutos de duración, de los cuales se encuentran disponibles gratuitamente 25 registros completos y los primeros diez minutos de los restantes. Los registros están numerados como 1XX o 2XX y aquellos incompletos se identifican con el prefijo 'x_'. Las señales fueron tomadas de hombres y mujeres entre los 23 y 89 años. Las mismas fueron registradas analógicamente con dos canales por el mismo modelo de instrumento en todos los casos, y más tarde digitalizadas con una frecuencia de muestreo de 360 Hz y resolución de 11 bits (para $\pm 5$ mV).

Los registros fueron anotados por dos grupos de cardiólogos independientemente, y las discrepancias fueron resueltas por consenso. Las anotaciones de la base se consideraron en este estudio como el verdadero tipo de arritmia y los distintos tipos presentes en la base se pueden apreciar en la tabla 1.

### Organización del artículo

En la sección 2 se describe el proceso de filtrado y extracción de características. El algoritmo CSL es presentado en la sección 3. En la sección 4 se explica la implementación del agrupador de complejos QRS. Los resultados del mismo son discutidos en la sección 5, y son comparados con algoritmos similares presentes en la literatura. Por último, las conclusiones se desarrollan en la sección 6.

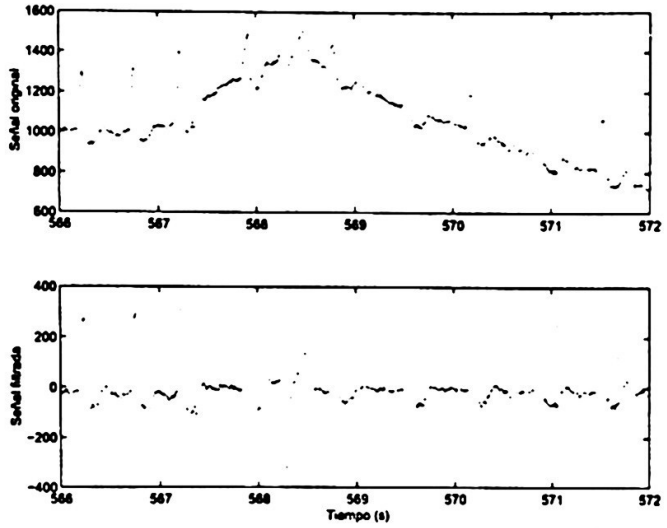



Figura 1: Filtrado del *baseline drift* en el registro 203

# 2. Preprocesamiento

## 2.1. Filtrado

La señal del ECG se ve contaminada por una variedad de fuentes de ruido. Se destaca entre ellas la respiración del paciente que causa el *baseline drift*. El mismo se manifiesta como un movimiento de la línea de base del ECG (ver señal original en la figura 1), que afecta la forma de los complejos QRS. Debido a su naturaleza, la mayor parte del espectro del *baseline drift* se concentra en las frecuencias bajas, del orden de 1 Hz. Para atenuar su presencia en la señal, así como el componente de continua, se diseñó un filtro pasaalto FIR de fase lineal, de 512 retardos, con una ganancia de 1 en la banda pasante, frecuencia de -3dB en 1.25 Hz y una ganancia de -50 dB en 0.1 Hz. Un ejemplo de este filtrado se puede apreciar en la figura 1 donde se muestra su comportamiento en una sección del registro 203.

La ubicación del pico R es determinada a partir de las anotaciones de la base de datos y el complejo QRS es extraído del canal 1 de la señal en una ventana de ancho $t_0 = 200$ ms (73 muestras a 360Hz) centrada en el pico R.

## 2.2. Transformada de Karhunen-Loève

Para la clasificación de las arritmias en los latidos es necesario antes extraer de la señal un conjunto de características que describan la misma. Varios enfoques han sido propuestos, siendo el uso directo de las muestras temporales el más utilizado ([3, 4, 5, 6]). Aunque tiene un buen desempeño en condiciones favorables, su performance baja con la presencia de ruido en la señal y es sensible a la ubicación temporal de la ventana del QRS. Otra propuesta es la medición de características temporales de la señal como son pendientes, ancho del QRS, área del QRS, etc. (por ejemplo en [7]) y su comparación con umbrales establecidos experimentalmente. Desgraciadamente, este enfoque no es robusto frente a la variabilidad de las morfologías de los complejos QRS dado que su efectividad depende de la elección de los umbrales.

Para enfrentar varias de estas dificultades, especialmente la sensibilidad al ruido, fueron propuestos espacios de transformadas cuyas métricas son más robustas. Además, la reducción de la dimensionalidad dada por el truncamiento de los coeficientes de la transformada ayuda en la velocidad de comparación entre latidos. Como contrapartida la dimensión de la trasformada no debe ser muy baja para no perder información clínica relevante.

Es común en la literatura para problemas de clasificación de complejos QRS trabajar con la transformada de Karhunen-Loève y la de Hermite ([8, 9]).

La transformada de Karhunen-Loève (K-L, también conocida como *Análisis de Componentes Principales*) es una transformación lineal dependiente de la señal, que es óptima en el sentido del error cuadrático medio. Tiene otras propiedades interesantes como son entropía de representación mínima y coeficiente no correlacionados.

La transformada K-L ha sido empleada extensamente en el análisis del segmento ST y la onda T ([10, 11, 12, 13, 14, 15]), compresión de ECG ([16, 17]) y clasificación de latidos ([18]). Dado que los vectores correspondientes a valores propios más altos representan mayor proporción de energía proyectada, un extractor de características de orden $N$ debe utilizar los primeros vectores propios ordenados por valor propio descendente.

En [19] se muestra que la representación obtenida con 6 coeficientes de K-L presenta un error cuadrático medio de aproximadamente 3 % sobre la base de datos MIT-BIH.

### 2.3. Generación de vectores de características

Para generar los vectores de características se utilizaron los primeros nueve coeficientes de la proyección en el espacio de la transformada, y además se agregó información temporal de la señal. En particular se consideraron las siguientes características:

- Intervalo $RR$ instantáneo. Calculado como,

$$RR_i = R_i - R_{i-1} \tag{1}$$

  donde $R_i$ representa la muestra en donde se estima se encuentra el pico R. Esta característica denota el ritmo cardíaco.

- Cociente entre el $RR$ instantáneo y el medio. Calculado como,

$$k_i = \frac{RR_i}{\overline{RR_i}} \tag{2}$$

  donde $\overline{RR_i}$ se calcula como el promedio de los últimos ocho $RR$s. El $k_i$ da una idea de cuan prematuro ($k_i < 1$) o retardado ($k_i > 1$) es el latido.

- Ancho del complejo QRS. No hay una definición formal del ancho del QRS debido a la gran variabilidad de las morfologías. Se utilizó el ancho en el cual el latido cae a 75 % de su altura pico.

El vector generado de esta forma fue normalizado sobre toda la base del MIT-BIH restando la media y dividiendo entre la desviación estándar en cada dimensión. Dado que los coeficientes de las transformadas contienen diferente cantidad de información, los mismos se dividen por el mismo factor para preservar dicha relación. Este factor es la desviación estándar del primer coeficiente.

Como consecuencia de la normalización cada una de las características tiene el mismo peso relativo. Para darle mayor importancia a la morfología se le asignaron diferentes pesos, asignándose pesos menores a las características temporales ya que la mayor parte de la información morfológica se encuentra contenida en los coeficientes de la transformada. De acuerdo a esta idea los pesos utilizados en la implementación fueron: 0.25 al ancho, 0.5 al RR y 0.5 al $K_i$ y 1 a los coeficientes de la Karhunen-Loève.

## 3. Algoritmo CSL

### 3.1. Introducción

El problema de cuantización vectorial se define como el mapeo de un espacio de entrada k-dimensional en un conjunto $Y$ de $N$ vectores representativos (*code-vectors*)

$$R^k \rightarrow Y \subset R^k \text{ donde } Y = \{y_1 \dots y_N\}$$

A cada *code-vector* se le asocia una región $S_i$ del espacio de entrada definido a partir de la regla del vecino más cercano:

$$S_i = \{x \in R^k / d(x, y_i) \leq d(x, y_j) \; \forall \, j \neq i\} \tag{3}$$

donde, por ejemplo para la norma Euclídea,

$$d(x, y_i) = ||x - y_i||_2^r \tag{4}$$

tal que $r$ denota la potencia de la norma.

Por lo tanto el agrupador (o cuantizador) se puede definir como la función $Q(x)$ que cumple que

$$Q(x) = y_i \; si \; x \in S_i \tag{5}$$

En la mayoría de los casos para el proceso de diseño de un cuantizador se dispone de un conjunto $X = x_1 \ldots x_T \subset R^k$ de $T$ vectores de entrenamiento, el cual es utilizado repetidamente para encontrar los valores óptimos de los *code-vectors* $y_j$. Cada presentación completa de los $T$ vectores se denomina *epoch*. La optimalidad se busca en el sentido de minimizar la *distorsión* definida como

$$D = \frac{1}{k} E\{d(x, Q(x))\} \tag{6}$$

que se puede escribir como

$$D = \frac{1}{k} \int_{R^k} d(x, Q(x)) p(x) dx \text{ en el caso continuo} \tag{7}$$

$$D = \frac{1}{k} \sum_i d(x_i, Q(x_i)) p_x(x_i) \text{ en el caso discreto} \tag{8}$$

Los algoritmos tradicionales de aprendizaje competitivo presentan cada vector de entrada $x \in R^k$ a todos los *code-vector* para calcular la distancia entre ellos y elegir al ganador como el *code-vector* $y_c$ más cercano. Para reducir la distancia media al ganador, el nuevo *code-vector* se calcula como

$$y_c(t+1) = y_c(t) + \eta(t)[x(t) - y_c(t)] \tag{9}$$

donde $\eta(t)$, denominado *learning rate*, es un parámetro no negativo monótonamente decreciente que controla la convergencia del algoritmo. La ecuación 9 es conocida usualmente como la regla *winner-takes-all*.

Debido a este entrenamiento de descenso por gradiente, el resultado final es altamente dependiente de las condiciones iniciales y lleva al algoritmo a converger a mínimos locales. Así, algunos *code-vectors* puede que nunca resulten ganadores, sub-utilizando la capacidad del algoritmo. Diversas propuestas han aparecido para atenuar este problema, incluyendo aprendizaje competitivo sensible a al frecuencia (FSCL) y Mapas Auto-Organizados (SOM). Sin embargo, ninguno de estos enfoques logran un desempeño óptimo.

## 3.2. Principio de Equidistorsión

El principio de equidistorsión fue propuesta por Gerscho en 1979, quien unificó y generalizó los resultados existentes sobre la optimalidad asintótica de los cuantizadores vectoriales. En su trabajo [20] se demuestra que, para una densidad de probabilidad "suave" $p(x)$, una cantidad grande de vectores representativos $N$ y una medida Euclídea de la distorsión, el mínimo de la distorsión es

$$D^* = C(k,r) N^{-\beta} \left( \int p(x)^{\frac{1}{1+\beta}} dx \right)^{1+\beta} \tag{10}$$

donde $r$ y $\beta = \frac{r}{k}$ (normalizada por la dimensión) denotan la potencia de la norma Euclídea usada (6) y el término $C(x,k)$ es un coeficiente que depende solo de la dimensión $k$ y de la potencia $r$.

El mínimo de la distorsión (10) se alcanza sólo cuando la densidad de probabilidad de los vectores de entrenamiento es proporcional a $p(x)^{(\frac{1}{1+\beta})}$[2]. Gerscho ([20]) muestra que en ese caso todas las sub-distorsiones, $D_i$ con $i = 1, 2, ..., N$, en una partición óptima son iguales. Donde la sub-distorsión $D_i$ se define como:

$$D_i = \frac{1}{k} E\{d(x, Q(x))\}, \text{ para } x \in S_i \tag{11}$$

de esta forma $D = \sum_i D_i$.

En el desarrollo anterior se asumió que la densidad de probabilidad subyacente era de soporte conexo, lo cual no es una suposición muy realista, dado que en los casos prácticos en general se tiene un conjunto

[2]La densidad de los vectores de entrenamiento es discreta, pero como T es grande se puede aproximar por una densidad continua.

1. Inicialización
   - $N$: número de *code-vectors* .
   - $X$: conjunto de entrenamiento.
   - $Y^0$: conjunto de *code-vectors* iniciales.
   - $\epsilon$: criterio de parada (sobre la distorsión).
   - $m$: contador de *epochs* ($m = 1$).
   - $D(0) = \infty$: distorsión inicial.
   - $s(m)$: función de selección de la cantidad de *code-vectors* a seleccionar en el paso $m$.
2. Entrenamiento: Para todos los $x \in X$. ejecutar 2*a* y 2*b* al igual que en el aprendizaje competitivo usual.
   *a*) Encontrar el *code-vector* más cercana al vector $x$.
   *b*) Calcular el nuevo *code-vector* usando la ecuación 9.
3. Selección: Si $s(m) > 2$. elegir $s(m)$ *code-vectors* y ejecutar el algoritmo de selección para los *code-vectors* elegidos. Incrementar el contador $m$ y volver al paso 2.
   En caso contrario, avanzar al paso 4.
4. Test de convergencia:
   *a*) Calcular la distorsión media $D(m)$ usando los *code-vectors* obtenidos en el paso 2.
   *b*) Si $\left|\frac{D(m)-D(m-1)}{D(m)}\right| \leq \epsilon$ termina el algoritmo dando como resultado final los *code-vectors* $Y^{(m)}$.
   En caso contrario, incrementar el contador $m$ y volver al paso 2.

Figura 2: Resumen de Algoritmo CSL

de clusters disjuntos de vectores de entrada. Ueda y Nakano ([1]) extendieron el resultado de Gerscho para este caso.

Considere el caso que la distribución es un región ($\Omega$) compuesta por $M$ clusters disjuntos ($\Omega_1, \ldots, \Omega_M$). Formalmente, $\Omega$ satisface:

1. $\Omega_i \subset R^k$
2. $i \neq j \Rightarrow \Omega_i \cap \Omega_j = \phi$
3. $\bigcup_{i=1}^{M} \Omega_i = \Omega \subset R^k$

Sea $p_j(x)$ una densidad de probabilidad continua para $\Omega_j$. Entonces la condición (2) implica que $p_j(x) = 0$ fuera de $\Omega_j$. Por esto, podemos decir que el caso anterior ($p(x)$ de soporte conexo) corresponde al caso $M = 1$.
Este resultado se puede resumir en el siguiente teorema.

**Teorema 1 (Principio de Equidistorsión)** *Para un número grande de vectores representativos $N$, aunque la densidad de probabilidad subyacente este formada por clusters disjuntos, la distorsión es mínima cuando todas las regiones presentan la misma sub-distorsión.*

## 3.3. Algoritmo CSL

De acuerdo al principio de equidistorsión, los clusters con mayor sub-distorsión deberían ser divididos en más regiones y en aquellos clusters con menor sub-distorsión debe haber menos regiones. Para lograr esto se agrega un mecanismo de *selección* al algoritmo de aprendizaje competitivo tradicional.

Este mecanismo consiste en quitar *code-vectors* de las zonas donde la distorsión es menor para agregarlos a las zonas donde la distorsión es mayor. El número de *code-vectors* a agregar(eliminar) se calcula mediante una función de *fitness* (que se explicará más adelante) calculada a partir de la sub-distorsión. De esta manera se puede ajustar el número de *code-vectors* en las regiones de acuerdo a sus sub-distorsiones sin cambiar el número total de *code-vectors*. A medida que el aprendizaje avanza las sub-distorsiones se van igualando hasta aproximar la óptima.

Cuando se dispone de un conjunto finito de vectores de entrenamiento, éste se utiliza repetidamente para ajustar los *code-vectors* hasta que el algoritmo converge. Asimismo los vectores se eligen aleatoriamente para facilitar el escape de los mínimos locales. Además, como se dispone de un conjunto discreto, el valor esperado de la distorsión se aproxima por la media de la misma sobre el conjunto de entrenamiento.

Además del ajuste usual, debido al aprendizaje competitivo, se realiza la selección en cada *epoch*. Para asegurar la convergencia de la selección, es necesario que el número de *code-vectors* a seleccionar decrezca de forma monótona luego de cada paso de selección. Por lo tanto, la selección siempre se realiza en un número finito de pasos.

### Resumen del algoritmo CSL

La figura 2 muestra un resumen del algoritmo CSL. La distorsión media en el paso 4*a* en la iteración $m$ se calcula como

$$D(m) = \frac{1}{T} \sum_{x \in X} min_{y \in Y^{(m)}} d(x, y) \tag{12}$$

donde $Y^{(m)}$ es el conjunto de *code-vectors* luego del paso 2 en la iteración $m$.

En el paso 3 se eligen los $s(m)$ *code-vectors* que se encuentran en los extremos de la distorsión (los que tienen más y menos). De esta forma, es claro ver que eligiendo los *code-vectors* de esta manera se equipara más rápido la distorsión de cada región.

Como se puede ver el algoritmo CSL es básicamente un aprendizaje competitivo clásico, excepto que se realiza la selección cada cierto intervalo de tiempo. Como consecuencia de esto, durante el proceso de aprendizaje, no solamente la distorsión total se reduce debido a la actualización de los *code-vectors* realizada por el paso competitivo (paso 2), sino que también se reduce la varianza de las sub-distorsiones en las distintas regiones debido al mecanismo de selección (3).

A continuación se presentará en detalle el algoritmo de selección.

### Algoritmo de Selección

La selección puede ser interpretada como la multiplicación y desaparición de *code-vectors* en cada región del espacio de entrada. Como es una función discontinua la selección puede ejecutarse correctamente incluso cuando un cluster de vectores de entrada pertenezca a varias regiones.

Claramente con un sólo paso de selección no se logra la equidistorsión, pero luego de ejecutarla varias veces esta es aproximadamente alcanzada.

En el algoritmo se tienen $s(m)$ *code-vectors* a seleccionar. que han de ser repartidos de forma de tal que en la regiones con mayor distorsión haya más *code-vectors*. Para elegir como se distribuirán los *code-vector*, se calcula una medida no-lineal de la distorsión (función de *fitness* $g_i$). Está función representa la fracción de los $s(m)$ *code-vectors* que corresponden en la región $i$, es decir que $\sum_{i=1}^{s(m)} g_i = 1$.

Como el número de *code-vectors* en cada región debe ser un número entero. la cantidad de *code-vectors* $u_i$ que habrá en la región $i$ se calculará como la parte entera del producto $g_i s(m)$. Pero entonces $\sum_{i=1}^{s(m)} u_i \leq s(m)$, por lo tanto pueden sobrar $s'(m) = s(m) - \sum_{i=1}^{s(m)} u_i$ *code-vectors*.

A cada región se le deberían asignar $g_i s(m)$ *code-vectors*, pero en realidad se le asignan $u_i$, por lo tanto *queda* por repartir la fracción $u_i' = g_i s(m) - u_i$ de *code-vector*. Entonces se le asigna un *code-vector* más a cada una de las $s'(m)$ regiones cuyo $u_i'$ es más cercano a uno.

En la figura 3 se puede apreciar un resumen del algoritmo de selección.

1. Calcular la función de *fitness* a partir de la subdistorsión $D_j$ como

$$g_i = \frac{D_i^\gamma}{\sum_{l=1}^{s(m)} D_l^\gamma}$$

donde $\gamma$ es una constante no negativa y menor que uno.

2. Calcular el número de *code-vectors* a reproducir en cada región:

   a) Calcular $u_i \equiv [g_i s(m)]$ ($[a]$ denota la parte entera del número $a$)

   b) Ordenar los *code-vectors* según el valor de $u_i' = g_i s(m) - u_i$.
   Para los $s'(m) = s(m) - \sum_{l=1}^{s(m)} u_l$ *code-vectors* que tengan el valor de $u_i'$ más cercano a uno, sumar uno al $u_i$ correspondiente.

3. Para cada $i$, el *code-vector* $y_i$ se reproducirá según $u_i$, de forma que:

   - Si $u_i = 0$ el *code-vector* desaparece.
   - Si $u_i = 1$ el *code-vector* permanece sin cambios.
   - Si $u_i > 1$ se agregan $u_i - 1$ *code-vectors* en la región sumando a $y_i$ una perturbación aleatoria $\delta_i$ tal que $||\delta_i|| \ll ||y_i||$.

Figura 3: Resumen del algoritmo de Selección

| Tipo | ! | / | A | E | F | J | L | N | |
|---|---|---|---|---|---|---|---|---|---|
| Nro.Lats | 472 | 7020 | 1273 | 106 | 786 | 31 | 4994 | 50407 | |
| $S_i$ | 95,55 % | 99,66 % | 90,97 % | 96,23 % | 78,50 % | 48.39 % | 99,92 % | 99,79 % | |
| $+P_i$ | 92,42 % | 99,49 % | 94,69 % | 94,44 % | 87,64 % | 78,95 % | 99,84 % | 99,31 % | |
| **Tipo** | **Q** | **R** | **S** | **V** | **a** | **e** | **f** | **j** | **Total** |
| Nro.Lats | 33 | 5072 | 2 | 5561 | 147 | 5 | 982 | 16 | 76907 |
| $S_i$ | 9,09 % | 99,78 % | 0,00 % | 96,94 % | 79,59 % | 0.00 % | 95,62 % | 12,50 % | 99,00 % |
| $+P_i$ | 100,00 % | 99,53 % | x | 98,14 % | 99,15 % | x | 95,43 % | 100,00 % | 98,98 % |

Cuadro 2: Performance por clase del agrupador sobre la base de datos.

**Observaciones**

La subdistorsión $D_i$ en el paso 1 se calcula como

$$D_i = \frac{1}{T} \sum_{x \in S_i} d(x, y_i)$$

debe notarse que en el cálculo de $D_i$ se usa la cantidad total de vectores de entrenamiento $T$ y no la de la región, $t_i$. Esto se desprende de la definición de subdistorsión (11).

En el paso 1 si $\gamma = 1$ la función $g_i$ es lineal, y si además la varianza de las sub-distorsiones es muy grande la selección será excesiva. Para evitar esto es conveniente tomar valores más pequeños por ejemplo $\gamma = 0{,}5$. En la práctica se puede observar que la selección es poco sensible al parámetro $\gamma$ en un entorno de este valor.

# 4. Agrupador

## 4.1. Implementación

El agrupador implementado utiliza el algoritmo CSL, realizando el entrenamiento con registros enteros, sin utilizar información sobre la clase de los latidos. Se trabajó sobre el espacio 12-dimensional formado por los vectores de características explicados en la sección 2.3, que se mapea en los *code-vectors*

| Tipo | ! | / | A | E | F | J | L | N | Q | R | S | V | a | e | f | j | Σ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ! | 451 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 36 | 0 | 0 | 0 | 0 | 37 |
| / | 0 | 6996 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 31 | 0 | 36 |
| A | 2 | 0 | 1158 | 0 | 0 | 1 | 1 | 52 | 0 | 1 | 0 | 1 | 7 | 0 | 0 | 0 | 65 |
| E | 0 | 0 | 0 | 102 | 0 | 0 | 2 | 0 | 0 | 3 | 0 | 1 | 0 | 0 | 0 | 0 | 6 |
| F | 0 | 0 | 1 | 0 | 617 | 0 | 0 | 26 | 1 | 0 | 0 | 55 | 4 | 0 | 0 | 0 | 87 |
| J | 0 | 0 | 1 | 0 | 0 | 15 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 4 |
| L | 1 | 0 | 0 | 1 | 0 | 0 | 4990 | 0 | 2 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 8 |
| N | 0 | 0 | 104 | 1 | 124 | 1 | 0 | 50299 | 9 | 1 | 2 | 65 | 15 | 5 | 10 | 14 | 351 |
| Q | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 6 | 2 | 1 | 14 | 0 | 1 | 0 | 5061 | 0 | 0 | 0 | 0 | 0 | 0 | 24 |
| S | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| V | 18 | 0 | 2 | 0 | 44 | 0 | 1 | 27 | 2 | 2 | 0 | 5391 | 4 | 0 | 2 | 0 | 102 |
| a | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 117 | 0 | 0 | 0 | 1 |
| e | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| f | 0 | 24 | 0 | 0 | 0 | 0 | 0 | 2 | 16 | 0 | 0 | 3 | 0 | 0 | 939 | 0 | 45 |
| j | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 |
| Σ | 21 | 24 | 115 | 4 | 169 | 16 | 4 | 108 | 30 | 11 | 2 | 170 | 30 | 5 | 43 | 14 | 766 |

Cuadro 3: Agrupamiento de los latidos sobre la base de datos, por tipo.

del agrupador. Los latidos se agruparon en 25 clases de forma tal de poder comparar los resultados con [8].

## 4.2. Entrenamiento

Usualmente los datos de interés constituyen un subconjunto *pequeño* del espacio de entrada, por lo tanto para la inicialización se buscó que los vectores representativos se ubicaran cerca del centro de los datos de entrada, en este caso como éstos fueron normalizados, este centro estará cerca del origen. Por lo tanto se inicializaron los *code-vectors* en el origen. Esto no representa un problema dado que el algoritmo CSL es robusto frente a variaciones de las condiciones iniciales.

Durante el entrenamiento los vectores de entrada son presentados en forma aleatoria, actualizando los *code-vectors* con cada vector presentado. El *learning rate* se inicializa en 0.1 y se decrementa un 6 % por cada *epoch*.

Para el algoritmo de selección se tomó un valor inicial de la función de selección $s(m)$ igual al número de *code-vectors* menos uno. $s(m)$ se eligió de manera que se decremente un *code-vector* por *epoch*. Si durante cinco *epoch* seguidas ningún *code-vector* fue modificado por el algoritmo de selección, éste finaliza.

Una vez finalizada la selección el algoritmo termina cuando el cambio relativo en la distorsión es menor que $10^{-5}$ o cuando se llega al número máximo de quince *epochs*.

# 5. Resultados

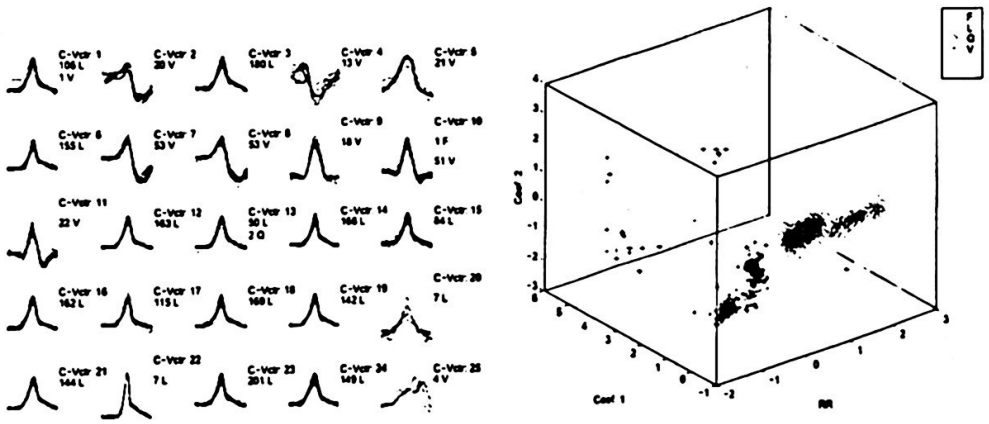
Para medir la performance de un algoritmo no supervisado, es necesario conocer la clase a la que pertenece cada latido agrupado. Con esta información disponible es posible asociar a cada *code-vectors* una clase de latido. Esta clase se determina de acuerdo al tipo de latido presente en mayor proporción dentro de la región correspondiente. En este trabajo se discriminaron los latidos en los 20 tipos disponibles[3] en la base MIT-BIH ([2]). Los códigos de los tipos de latidos pueden encontrarse en la tabla 1.

La performance del algoritmo se medirá en función de la *sensibilidad* (S) y el *valor predictivo positivo* (+P). Para definir estas magnitudes se considerarán la siguientes categorías de latidos (por clase)

**$TP_i$**: Verdaderos Positivos (True Positive) son los latidos de clase $i$ asignados a la clase $i$.

**$FP_i$**: Falsos Positivos (False Positive) son los latidos de otra clase asignados a la clase $i$.

[3]Cabe aclarar que si bien existen 20 tipos de latidos definidos en la base, sólo aparecen en los registros 16 de dichos tipos



(a) Agrupamiento de complejos QRS (b) Visualización del espacio de entrada

Figura 4: Agrupamiento del registro 214.

**$FN_i$:** Falsos Negativos (False Negative) son los latidos de clase $i$ asignados a otra clase.

Entonces se definen sensibilidad y valor predictivo positivo por clase ($S_i$ y $+P_i$) como

**$+P_i$:** Relación de los latidos de clase $i$ correctamente clasificados sobre el total de latidos asignados a la clase $i$ por el algoritmo. Representa que tan seguro se puede estar de que el latido pertenece al grupo asignado por el agrupador, y no a otro.

$$+P_i = \frac{TP_i}{TP_i + FP_i} \tag{13}$$

**$S_i$:** Relación de los latidos de la clase $i$ correctamente clasificados sobre el total de latidos efectivamente de clase $i$. Representa que tan seguro se puede estar de que los latidos de esa clase sean asignados correctamente por el algoritmo.

$$S_i = \frac{TP_i}{TP_i + FN_i} \tag{14}$$

De la misma forma se definen las performances globales[4]

**+P:** Es el promedio de las predictividades positivas por clase, pesadas por el número de latidos de cada clase

**S:** Es el promedio de las sensibilidades por clase, pesadas por el número de latidos de cada clase.

En la tabla 2 se pueden apreciar las performances por clase obtenidas por el algoritmo sobre toda la base MIT-BIH. La sensibilidad global fue 99.00% y el valor predictivo positivo global 98.98%. En la tabla se puede observar que los tipos más comunes de latidos son los que presentan las mayores sensibilidades. Esto se debe a la naturaleza del algoritmo CSL, que asigna un mayor número de *code-vectors* a las regiones con mayor densidad. Por otro lado, al haber tantos *code-vectors* asignados a estas clases, algunos latidos de clases minoritarias caen en las regiones correspondientes a dichos *code-vectors*. Por esta razón generalmente la predictividad positiva de clases más comunes es ligeramente menor que su sensibilidad.

A la inversa, las clases minoritarias, presentan una baja sensibilidad y un alto valor predictivo positivo, en términos relativos.

Por ejemplo, la mayoría de los latidos son normales (65.6%) y consecuentemente la sensibilidad en el agrupamiento es alto (99.8%), y el valor predictivo positivo es de 99.3%. Los latidos $J$ (latidos

[4]Una vez que se tienen las performances por clase por registro existen dos formas de calcular las globales: la primera es calcular las performances totales por registro y luego promediar sobre todos los registros, y la segunda es hacerlo a la inversa, primero sumando sobre todos los registros y luego promediando sobre las clases. En este trabajo se usó la segunda.

prematuros nodales) representan el 0.04 % de toda la base, y presentan una sensibilidad de 48.4 % y un valor predictivo de 79.0 %.

En la tabla 3 se muestra con mayor detalle el agrupamiento de los latidos sobre toda la base de datos. El eje horizontal representa la clase verdadera del vector y el vertical la clase asignada por el agrupador. Por ejemplo, hay 18 latidos *flutter* ventriculares (!) que son asignados incorrectamente a latidos ventriculares prematuros (V). La diagonal representa los latidos correctamente agrupados. La columna y la fila $\tilde{\Sigma}$ denota la suma en sentido horizontal y vertical respectivamente de los latidos erróneamente asignados (excluyendo entonces la diagonal). El error en sentido horizontal representa los latidos de otras clases asignados a una clase , y en sentido vertical los latidos de una clase asignados a otras.

Observando esta tabla se puede verificar lo afirmado anteriormente sobre las performances de agrupamiento en las clases mayoritarias. Por ejemplo hay 108 latidos de clase normal (N) asignados a otras clases y 351 latidos de otras clase asignados a la clase normal. En el caso de la clase J, las cantidades son 16 y 4 respectivamente.

En la tabla 5 se muestra la performance global por registro. La sensibilidad global varía de 96.1 % (registro 213) hasta 100 % (varios registros). El valor predictivo global lo hace entre 95.3 % (x_124) y 100 % (varios).

Como ejemplo, en la figura 4(a) se puede apreciar la superposición de los complejos QRS (después de filtrados) asociados a cada *code-vector* para el registro 214 (sensibilidad 99.8 %, predictividad 99.7 %). En la misma se puede apreciar que los errores corresponden a un latido fusión entre ventricular y normal (F) asignado a un grupo de ventriculares (V), un V a un grupo de bloqueo de rama izquierda (L), y dos inclasificables (Q) a uno de L.

En la figura 4(b) se puede observar la estructura de los latidos en el espacio de entrada al agrupador (la salida de la extracción de características). Obviamente no se pueden representar las 12 dimensiones al mismo tiempo, por lo que se optó por hacerlo con las dos primeras direcciones de la transformada (que deberían tener la mayor parte de la información) y el período cardíaco instantáneo (RR). En la leyenda de la misma se pueden observar los distintos tipos de latidos representados, mientras que los *code-vectors* figuran como estrellas (*). Las observaciones hechas en el párrafo anterior pueden aclararse. En la misma se observa que el latido ventricular (V) mal clasificado en el *code-vector* 4 se debe a que tiene un período (RR) muy distinto al del resto de los demás ventriculares.

## 5.1. Comparación con otros algoritmos

En esta sección se comparará la performance del agrupador desarrollado con la de otros algoritmos similares en la literatura.

### Clustering ECG complexes using Hermite functions and self-organizing maps

Lagerholm *et al.* [8] utilizan la transformada de Hermite para extraer características de los complejos QRS y un mapa auto-organizado de 25 *code-vectors* para realizar el agrupamiento. Dichos autores reportan una performance global (sensibilidad) sobre *toda* la base de datos MIT-BIH de 98.5 % frente a 99.00 % de este trabajo. No se calcula el valor predictivo positivo. Cabe aclarar que en dicho trabajo la performance se calcula también dividiendo los latidos en los 16 tipos no vacíos que aparecen en la base de datos.

El trabajo de Lagerholm *et al.* presenta las siguientes diferencias básicas, que es necesario aclarar previo a comparar performances:

- Se utilizó la base de datos de 48 registros completos.
- No se usaron en este trabajo las anotaciones de la base de datos para ubicar los picos R sino un algoritmo de detección que tiene una sensibilidad de 99.7 % (aparentemente se ignoran en todo el estudio los Falsos Positivos del detector). Por lo tanto, la performance *real* del algoritmo de Lagerholm en las condiciones de nuestro trabajo está acotada por 98.8 %, que se daría en el caso en que todos los latidos que no fueron detectados fueran agrupados correctamente.
- Lagerholm *et al.* utilizan ambos canales de la base de datos, mientras que este trabajo utiliza el canal 1 solamente.

No se reportaron las performances por registro. Sin embargo, para comparar los resultados con el trabajo de Hu *et al.* [18] (que es un algoritmo supervisado) se calcula la medida de performance: [5],

$$R = \frac{\text{No. de latidos V y E correctamente agrupados}}{\text{No. total de latidos}} \tag{15}$$

En la tabla 4 se puede apreciar la comparación de esta medida para este trabajo con el trabajo citado [8]. Como se puede apreciar en la misma en los registros completos (200-219), que son directamente comparables, la performance obtenida por este trabajo es ligeramente superior a la reportada por Lagerholm *et al.*. En los demás registros se sigue la misma tendencia.

| | Lagerholm *et al.* | | CSL | |
|---|---|---|---|---|
| Reg. | Latidos | R | Latidos | R |
| 200 | 2605 | 98,1 % | 2599 | 99,6 % |
| 201 | 1858 | 99,9 % | 1961 | 100,0 % |
| 202 | 2038 | 99,4 % | 2134 | 100,0 % |
| 203 | 2882 | 96,8 % | 2978 | 98,3 % |
| 205 | 2491 | 99,9 % | 2654 | 99,9 % |
| 207 | 2215 | 96,9 % | 2330 | 97,2 % |
| 208 | 2823 | 98,5 % | 2953 | 98,9 % |
| 209 | 2860 | 100,0 % | 3002 | 100,0 % |
| 210 | 2502 | 98,3 % | 2648 | 99,3 % |
| 213 | 3071 | 97,6 % | 3249 | 98,7 % |
| 214 | 2134 | 99,9 % | 2259 | 99,9 % |
| 215 | 3172 | 99,9 % | 3361 | 100,0 % |
| 219 | 2156 | 99,6 % | 2152 | 99,8 % |
| 221 | 2302 | 100,0 % | 825 | 100,0 % |
| 223 | 2480 | 99,1 % | 837 | 99,3 % |
| 228 | 1978 | 100,0 % | 695 | 99,9 % |
| 230 | 2303 | 100,0 % | 727 | 100,0 % |
| 231 | 1877 | 99,9 % | 503 | 100,0 % |
| 233 | 2936 | 99,8 % | 1021 | 99,4 % |
| 234 | 2577 | 100,0 % | 919 | 100,0 % |
| Tot | 49260 | 99,1 % | 39807 | 99,5 % |

Cuadro 4: Comparación con algoritmo SOM de Lagerholm *et al.*[8]

### Classifying Multichannel ECG Patterns with an Adaptive Neural Network

En este trabajo S. Barro *et al.* [3] aplican una red neuronal ART multicanal para agrupar los QRS utilizando las muestras temporales. Se reporta una sensibilidad global de 98.72 % y una predictividad positiva de 98.78 % frente al valor predictivo de 98.98 % de este trabajo. Es necesario hacer varias aclaraciones sobre el trabajo de Barro:

- Utiliza 20 registros completos de la base MIT-BIH, que tienen latidos ventriculares.
- El número de clases del algoritmo no es fijo sino que se adapta a cada registro, llegando a más de 80 en el registro 108 frente a las 25 clases de este trabajo.
- Se usan ambos canales de la base y se le asignan créditos a cada uno basado en el ruido estimado, mientras que este trabajo utiliza el canal 1 solamente.
- Es un algoritmo *online*; el agrupador CSL es *offline*.

Debido a estas diferencias y dado que los autores no incluyeron el número de latidos por registro, no se efectuó una comparación por registo con este trabajo.

[5] Es de notar que aunque entre los VEB (latidos ectópicos ventriculares) de [18] se encuentran los latidos prematuros ventriculares R-sobre-T que Lagerholm *et al.* ignoran al definir la medida ya que no hay ninguno en la base de datos. Los *flutter* ventriculares no se colocan en este grupo (aunque deberían pertenecer) ya que [18] los ignora.

| Reg. | Tipo | total | Reg. | Tipo | total | Reg. | Tipo | total |
|---|---|---|---|---|---|---|---|---|
| 100 | Nro.Lats | 2271 | 208 | Nro.Lats | 2953 | x_116 | Nro.Lats | 795 |
| | S | 100,00 % | | S | 96.95 % | | S | 100,00 % |
| | +P | 100,00 % | | +P | 96,89 % | | +P | 100,00 % |
| 101 | Nro.Lats | 1863 | 209 | Nro.Lats | 3002 | x_117 | Nro.Lats | 503 |
| | S | 99,95 % | | S | 98,60 % | | S | 100,00 % |
| | +P | 99,95 % | | +P | 98,56 % | | +P | 100,00 % |
| 102 | Nro.Lats | 2185 | 210 | Nro.Lats | 2648 | x_121 | Nro.Lats | 607 |
| | S | 98,49 % | | S | 98,83 % | | S | 100,00 % |
| | +P | 98,26 % | | +P | 98,75 % | | +P | 100.00 % |
| 103 | Nro.Lats | 2082 | 212 | Nro.Lats | 2746 | x_122 | Nro.Lats | 835 |
| | S | 99,90 % | | S | 99.96 % | | S | 100,00 % |
| | +P | 99,81 % | | +P | 99,96 % | | +P | 100,00 % |
| 104 | Nro.Lats | 2227 | 213 | Nro.Lats | 3249 | x_123 | Nro.Lats | 504 |
| | S | 97,49 % | | S | 96,12 % | | S | 99,80 % |
| | +P | 96,67 % | | +P | 95.88 % | | +P | 99,60 % |
| 105 | Nro.Lats | 2570 | 214 | Nro.Lats | 2259 | x_124 | Nro.Lats | 522 |
| | S | 99,69 % | | S | 99,82 % | | S | 96,17 % |
| | +P | 99.49 % | | +P | 99,69 % | | +P | 95,27 % |
| 106 | Nro.Lats | 2025 | 215 | Nro.Lats | 3361 | x_220 | Nro.Lats | 697 |
| | S | 99,90 % | | S | 99.88 % | | S | 99,71 % |
| | +P | 99,90 % | | +P | 99.86 % | | +P | 99,71 % |
| 107 | Nro.Lats | 2135 | 217 | Nro.Lats | 2206 | x_221 | Nro.Lats | 825 |
| | S | 100,00 % | | S | 99,59 % | | S | 100,00 % |
| | +P | 100,00 % | | +P | 99.59 % | | +P | 100,00 % |
| 118 | Nro.Lats | 2276 | 219 | Nro.Lats | 2152 | x_222 | Nro.Lats | 737 |
| | S | 99,65 % | | S | 99.40 % | | S | 98,10 % |
| | +P | 99,67 % | | +P | 99,03 % | | +P | 97,43 % |
| 119 | Nro.Lats | 1985 | x_108 | Nro.Lats | 560 | x_223 | Nro.Lats | 837 |
| | S | 100,00 % | | S | 99.64 % | | S | 97,37 % |
| | +P | 100,00 % | | +P | 99,31 % | | +P | 96,00 % |
| 200 | Nro.Lats | 2599 | x_109 | Nro.Lats | 855 | x_228 | Nro.Lats | 695 |
| | S | 98,38 % | | S | 99,88 % | | S | 99,86 % |
| | +P | 97.19 % | | +P | 99,92 % | | +P | 99,86 % |
| 201 | Nro.Lats | 1961 | x_111 | Nro.Lats | 696 | x_230 | Nro.Lats | 727 |
| | S | 97,86 % | | S | 99.86 % | | S | 100,00 % |
| | +P | 98,25 % | | +P | 99,71 % | | +P | 100,00 % |
| 202 | Nro.Lats | 2134 | x_112 | Nro.Lats | 852 | x_231 | Nro.Lats | 503 |
| | S | 98,50 % | | S | 100,00 % | | S | 99,60 % |
| | +P | 98,47 % | | +P | 100,00 % | | +P | 99,41 % |
| 203 | Nro.Lats | 2978 | x_113 | Nro.Lats | 579 | x_232 | Nro.Lats | 600 |
| | S | 98.15 % | | S | 100,00 % | | S | 99,83 % |
| | +P | 97.91 % | | +P | 100,00 % | | +P | 99,83 % |
| 205 | Nro.Lats | 2654 | x_114 | Nro.Lats | 554 | x_233 | Nro.Lats | 1021 |
| | S | 99,47 % | | S | 99,28 % | | S | 99,22 % |
| | +P | 99,19 % | | +P | 98,60 % | | +P | 98,57 % |
| 207 | Nro.Lats | 2330 | x_115 | Nro.Lats | 633 | x_234 | Nro.Lats | 919 |
| | S | 96,91 % | | S | 100,00 % | | S | 100,00 % |
| | +P | 96,77 % | | +P | 100,00 % | | +P | 100,00 % |

Cuadro 5: Performance global por registro del agrupador CSL.

## 6. Conclusiones

En el presente trabajo se investigó el problema de agrupamiento de complejos QRS, y en particular se estudió la aplicabilidad del algoritmo CSL al mismo, lo cual constituye el aporte fundamental de este trabajo, dado que no se encontró en la literatura ningún trabajo que lo utilizara en este tipo de problemas.

Se adoptó como metodología de trabajo la utilización de criterios claros de evaluación de desempeño y la presentación objetiva de resultados para facilitar su reproducción, así como la comparación con los resultados publicados en la bibliografía consultada.

El agrupador desarrollado mostró una sensibilidad de 99.00% y un valor predictivo positivo de 98.98%, ligeramente superior a los desempeños reportados en trabajos similares ([3, 8]). Estos resultados muestran que utilizar el algoritmo de aprendizaje competitivo selectivo (CSL) en el desarrollo de un agrupador es una alternativa válida para enfrentar este problema. Esto es coherente con los resultados teóricos que muestran la superioridad de CSL frente a los demás algoritmos de aprendizaje competitivo.

Los resultados obtenidos sugieren la posibilidad de aplicar una variante supervisada del algoritmo CSL en problemas de clasificación. Una posible implementación de esta idea, desarrollada por los autores, se puede encontrar en [19].

Existen algunas mejoras, que escapan al alcance de este trabajo, que pueden contribuir a mejorar el desempeño obtenido. Por ejemplo la utilización de más canales en todo el proceso ([3, 4]). Esto implica que se dispone de más información sobre un mismo latido, lo cual puede influir favorablemente en la tasa de clasificación.

Por otra parte, el algoritmo CSL es adaptivo en el número de *code-vectors* por clase, pero el número total permanece constante. Se puede ver [19] que la performance se ve alterada significativamente según el número total de *code-vectors*. Por lo tanto es razonable pensar que permitiendo variar el número total del *code-vectors*, se puedan lograr performances más altas en los clasificadores. En la literatura existen varios trabajos en torno a esta idea, por ejemplo [4, 21, 22] y [23].

## Referencias

[1] N. Ueda; R. Nakano. A new competitive learning approach based on an equidistortion principle for designing optimal vector quantizers. *Neural Networks*, 7(8):1211–1227, January 1994.

[2] Massachusetts Institute of Technology. MIT-BIH Database Distribution. http://www.physionet.org/physiobank/database/mitdb/.

[3] S. Barro; M. Fernandez-Delgado; J.A. Vila-Sobrino; C. V. Regueiro; and E. Sanchez. Classifying multichannel ECG patterns with an adaptive neural network. *IEEE Eng. in Med. and Bio.*, January/February 1998:45–55.

[4] M. Fernandez-Delgado and S. Barro. Mart: A multichannel ART-based neural network. *IEEE Trans. on Neural Networks*, 9:139–150.

[5] R. L. Watrous; and G. Towell. A patient-adaptive neural network ECG patient monitoring algorithm. *Computers in Cardiology, Vienna, Austria*, pages 229–232, September 10-13, 1995.

[6] R. Silipo; and C. Marchesi. Artificial neural networks for automatic ECG analysis. *IEEE Trans. Signal Processing*, Vol. 46, no. 5:1417–1425, 1998.

[7] R. L. Watrous; G. Towell; M. S. Glassman; M. S. Shahraray and D. Theivanayagam. Synthesize, optimize, analyze, repeat (SOAR): Application of neural network tools to ECG patient monitoring. *Proceedings of the Third International Congress on Air- and Structure-Borne Sound and Vibration, Montreal*, pages 997–1004, 1994.

[8] M. Lagerholm; C. Peterson; G. Braccini; L. Edenbrandt; L. Sörnmo. Clustering ECG complexes using hermite functions and self-organizing maps. *IEEE Trans. on Biomedical Engineering*, pages pp. 838–848, 2000.

[9] P. Laguna; R. Jané and P. Caminal. Adaptive estimation of QRS complex by the hermite model for classification and ectopic beat detection. *Medical and Biological Engineering and Computing*, vol. 34:58–68, 1996.

[10] P. Laguna; G. B. Moody; R. Jané; P. Caminal and R.G. Mark. The KL transform as a tool to analyze the ST segment: Comparison with QT interval. *Journal of Electrocardiology*, vol. 28 (suplement):41–49, 1995.

[11] P. Laguna; G. B. Moody and R. G. Mark. Analysis of the cardiac repolarization period using the KL transform: Aplications on the ST-T database. *Computers in Cardiology, IEEE Computer Society Press, Bethesda*, pages 233–236, 1994.

[12] R. Jané; S. Olmos. Análisis de la repolarización cardiaca en el ECG, registrado durante angioplastia, mediante la transformada de KL. *XIII Congreso Anual de la Asociación Española de Bioingeniería, Barcelona*, pages 41–42, 1995.

[13] P. Laguna; G.B. Moody; J. Garcia; A.L. Goldberger and R.G. Mark. Analysis of the ST-T complex using the KL transform: Adaptive monitoring and alternans detection. *Medical and Biological Engineering and Computing*, vol. 37:175–189, 1999.

[14] S. Olmos; J. Garcia; R. Jane and P. Laguna. Análisis de la señal ECG con expansiones ortogonales de reducido número de coeficientes (in spanish). *Comunicaciones del XV Congreso anual de la Sociedad Española de Ingenieria Biomedica (SEIB), Valencia*, pages 89–92, 1997.

[15] I. Roncal and P. Laguna. Detector de isquemia utilizando la transformada de karhunen-loève. *Comunicaciones del XIV Congreso anual de la Sociedad Española de Ingeniería Biomédica, Pamplona*, pages 91–93, 1996.

[16] S. Olmos; D. Echeverria; R. Jane and P. Laguna. Compresion de señal ECG multiderivacionales mediante expansiones ortogonales optimas (in spanish). *Comunicaciones del XV Congreso anual de la Sociedad Española de Ingenieria Biomedica (SEIB), Valencia*, pages 321–324, 1997.

[17] S. Olmos; M. Millan; J. Garcia and P. Laguna. ECG data compression with the KL transform. *Computers in Cardiology, IEEE Computer Society Press, Indianapolis*, 1996.

[18] Y. H. Hu; S. Palreddy; and W. J. Tompkins. A patient-adaptable ECG beat classifier using a mixture of experts approach. *IEEE Trans. on Biomed. Eng.*, Vol. 44, No. 9:891–900, 1997.

[19] Pablo Aguirre; Juan Cardelino:Ñicolás Loeff. Sistema de detección, clasificación e indentificación en línea de complejos QRS. Tesis de grado, Facultad de Ingeniería, Universidad de la República, Mayo 2002.

[20] A. Gerscho. Asymptotically optimal block quantization. *IEEE Trans. on Information Theory*, IT-25(4):1211–1227, July 1979.

[21] S. G. Miaou; H.L. Yen. Quality driven gold washing adaptive vector quantization and its application to ECG data compression. *IEEE Trans. on Biomed. Eng.*, Vol. 47, No. 2:209–218, 2000.

[22] D. Alahakoon; S. Halgamuge; and B. Srinivasan. Dynamic self-organizing maps with controlled growth for knowledge discovery. *IEEE Trans. on Neural Networks*, 11(3), May 2000.

[23] Stergios Papadimitriou: Seferina Mavroudi: Liviu Vladutu: and Anastasios Bezerianos. Ischemia detection with a self-organizing map supplemented by supervised learning. *IEEE Trans. on Neural Networks*. vol. 12. No. 3:503–515.

# Identificación de Células Cervicales a través de Redes Neuronales Pulsantes.

Aldrin Barreto Flores [1], Leopoldo Altamirano Robles [1], Olga Pérez Chalini [2]
[1] Instituto Nacional de Astrofísica, Óptica y Electrónica.
Ciencias Computacionales. Luis Enrique Erro No. 1, Santa María Tonantzintla, Puebla, 51&216, 72000. México.
[2] Instituto Mexicano del Seguro Social. San Alejandro. Av. 10 Poniente 2721. Puebla, Puebla, 72090. México.

**Resumen**

La identificación de células cervicales ha sido un problema explorado a través de diversas técnicas computacionales, principalmente técnicas espaciales en imágenes y redes neuronales [1] [9]. En este trabajo se propone el uso de redes neuronales pulsantes para la generación de vectores de características a través del Modelo de Eckhorn en imágenes de células cervicales, lo cuál no ha sido reportado anteriormente en la literatura. Uno de los aspectos importantes de estas redes, es la propiedad de obtener diversas características de textura por medio de la generación de pulsos en las neuronas al momento de procesar la imagen. La cuantificación de estos pulsos en el tiempo [4] para cada una de las imágenes de las células permite la generación de vectores de características. Estos vectores se emplean en una red neuronal no supervisada basada en la arquitectura de Kohonen para poder determinar el grado de precisión en la clasificación de células normales y anormales.

Empleando un total de 10 características se tienen resultados preliminares, que han permitido tener un 86% de confiabilidad en un conjunto de muestras de 132 células a la hora de identificar células normales y anormales.

**Palabras Claves:** Reconocimiento de Patrones, Redes Neuronales Pulsantes, Cáncer Cérvico Uterino.

## 1. Introducción.

La búsqueda de métodos que permitan tener una mayor eficiencia en sistemas de reconocimiento en el área de la medicina (usando imágenes digitales médicas) es una labor continua que inició hace varias décadas [7]. Para el caso de imágenes de células cervicales, generalmente las soluciones propuestas han implicado el uso de técnicas espaciales como la Matriz de Co-ocurrencia para la caracterización de textura [1] y en algunos otros casos aspectos morfológicos [5] [8]. Así mismo, otras técnicas como la lógica difusa han permitido mejorar los vectores de características para la clasificación de las mismas; mientras que otras alternativas han propuesto el uso de características multiresolución o de frecuencia para determinar patrones más confiables [6].

Una de las principales desventajas de los métodos anteriores es la necesidad de etapas de pre-procesamiento de la imagen, debido a que generalmente se desea tener la mejor calidad posible al momento de analizar y procesar una imagen. El llevar a cabo

este tipo de procesamiento trae por consecuencia el aumento de parámetros a ser definidos, ya que normalmente no se realizan de forma automática.

Una manera de evitar este problema consiste en tener la mayor calidad posible en las imágenes celulares a la hora de digitalizar las imágenes de las células; esto se puede realizar a través de la aplicación de soluciones químicas a las muestras [1], lo que permite la separación de las células y la eliminación de elementos celulares no deseados y que degradan la calidad de las imágenes. Esta alternativa representa desafortunadamente un aumento en los costos económicos del proceso de diagnóstico por lo cuál sólo en algunos países desarrollados es posible su uso.

En este trabajo se propone el uso de redes neuronales pulsantes para su evaluación en el reconocimiento de imágenes digitales de células cervicales. Estas redes neuronales tienen como su base el modelo de Eckhorn [3] y su principal característica es que permiten emular la forma en la cuál funciona el sistema visual de algunos mamíferos. Estas redes neuronales permiten obtener características de bordes y textura de una manera intrínseca, sin necesidad de otros procesamientos en las imágenes (Matriz de co-ocurrencia, filtros de Sobel, etc). Estas características se obtienen por medio de la generación de pulsos en las neuronas a la hora de procesar las imágenes.

Las redes neuronales pulsantes se han aplicado en otras áreas de la medicina, tales como en el análisis de mamografías y tomografías con el objetivo de tener segmentaciones mas exactas e identificación de algunas patologías [3]. En nuestro caso se plantea el uso de estas redes con el objetivo de tener un procesamiento rápido en la obtención de vectores de características a través de la cuantificación de pulsos en las neuronas de la red. El procesamiento se realiza en imágenes médicas de células cervicales obtenidas en condiciones que dificultan su identificación (sin ningún pre procesamiento especial químico), para posteriormente evaluar la propiedad de clasificación de las mismas por medio de una red neuronal no supervisada con la arquitectura de Kohonen [11].

## 2. Imágenes Digitales de Células.

El proceso de diagnóstico de cáncer cervico uterino (por medio de la prueba de Papanicolau) consiste en la observación directa de las células cervicales. Este proceso de revisión es efectuado por los cito-tecnólogos (personal capacitado para la inspección y diagnóstico celular) por medio de un microscopio óptico.

La búsqueda de anormalidades en las células se basa principalmente en el análisis visual de diferentes atributos en el núcleo y citoplasma. Estos atributos comprenden la regularidad del núcleo, la relación núcleo / citoplasma, la distribución y textura de la cromatina que forma parte del núcleo, junto con otros elementos (poli-nuclearidad, inflamación, presencia de leucocitos, etc) que permiten determinar el estado en el que se encuentran las células.

Uno de los parámetros más importantes que permiten determinar el estado de una célula, se reflejan de manera principal en las características de la textura de la cromatina del núcleo, ya que es el lugar donde se empiezan a reflejar anormalidades en el ADN de la célula. Por lo tanto la textura de la cromatina y de la célula en general, es uno de los parámetros importantes a considerar a la hora de determinar un diagnóstico.

Para la generación de imágenes de prueba se solicito al personal encargado del área de diagnóstico la selección de dos láminas, una que contuviera células normales y otra que presentará células anormales (nivel NICIII).

Las láminas seleccionadas no tenían ningún tipo de tratamiento especial para la eliminación de tejido que pudiera ocluir información de las células, esto con el objetivo de determinar la robustez del método en imágenes cotidianas en una Institución de Salud Pública Mexicana.

El proceso de adquisición de imágenes comprendió la adaptación de una cámara digital de alta resolución (SONY DFW-SX900) al ocular de un microscopio óptico (con una lente de 40x) que se emplea para la observación de las laminas de tejido cervical. En la digitalización de las imágenes se empleó una tarjeta digitalizadora PCI Matrox-Meteor II–1394, junto con una computadora PC convencional. Las imágenes que se obtuvieron fueron en color de 16 bits y con una resolución de 1280 x 960 pixeles, ejemplos de estas imágenes se muestran en la Fig. 1.

A partir de estas imágenes (convertidas a escalas de grises) se procedió a la selección de manera manual de aquellas células que tuvieran características normales y anormales con el objetivo de determinar el conjunto de muestras a evaluar en la propuesta de este trabajo. Las diferencias entre células normales y anormales se aprecia al observar las Figs. 1a y 1b, por ejemplo en la Fig.1a se puede observar una coloración muy fuerte y concentrada de la cromatina del núcleo que es bastante puntual, así mismo se aprecia una textura del citoplasma y del núcleo de manera uniforme. Para el caso de la Fig. 1b los núcleos son más grandes, además de presentar una distribución más dispersa de la cromatina del núcleo; en algunos casos el citoplasma es demasiado tenue o no se presenta en las células a diferencia de las células en la Fig.1a. Ambas imágenes presentan diversos elementos celulares que dificultan la apreciación de características en las células, como por ejemplo leucocitos, flora bacteriana, etc.



(a) (b)

**Fig. 1.** Imágenes de células cervicales.(a) Células normales y (b) anormales (NICIII).

# 3. Obtención de Parámetros a través de Redes Neuronales Pulsantes.

## 3.1 Redes Neuronales Pulsantes.

La teoría de redes neuronales pulsantes tiene como principal finalidad la emulación y el estudio del funcionamiento de la corteza visual en algunos mamíferos, como por ejemplo el gato. La principal virtud de esta teoría es la generación de modelos [3] que permiten obtener diferentes características de las imágenes por medio de la generación de pulsos en las neuronas. Este funcionamiento es similar a la manera en la cuál opera la corteza visual (en una forma muy reducida en capacidad), ya que diversos atributos obtenidos por la retina se van procesando y propagando (en el tiempo) a través de pulsos en el cerebro que permiten aprender y reconocer diferentes patrones. Algunos de estos atributos comprenden textura, bordes y regiones con características similares, entre muchos otros.

La característica pulsante de este tipo de redes se debe a que el disparo de cada neurona se encuentra condicionado a la superación de un umbral dinámico que se reduce con el tiempo (con la finalidad de que en algún momento en el tiempo todas las neuronas pulsen). Este umbral se compara con la contribución de las neuronas vecinas en un radio determinado, junto con la entrada del píxel de la imagen que corresponde a esa neurona. Por lo tanto si un conjunto de neuronas pulsan en un determinado momento, esto significa que las neuronas presentan contribuciones similares que se traducen en regiones de textura similares, bordes o diversos elementos de textura en la imagen.

Una característica interesante de las redes neuronales pulsantes es la ausencia de entrenamiento en comparación con otros esquemas de redes neuronales tradicionales; al mismo tiempo una desventaja de este método es la ausencia de un método de paro o un número fijo de ciclos que permitan estimar cuando se tienen resultados óptimos. Generalmente se estima de manera experimental el número de ciclos en base a los resultados que se van obteniendo.



(a) Imagen Original (b) Ciclo de ejecución No. 4 (c) Ciclo de ejecución No. 8

**Fig. 2.** Obtención de características de textura del citoplasma (b) y bordes (c) en imágenes de células cervicales.

Las imágenes de salida generadas por las redes neuronales pulsantes facilitan el manejo y el proceso de las diversas características en las imágenes ya que solo tendrán valores binarios. Cada una de las imágenes representa en realidad diversas características que encuentra la red neuronal pulsante, es decir un píxel en blanco

significa que la neurona (i,j) generó un pulso al superar el valor de su umbral establecido. Para el caso de un píxel negro su significado es que aún la neurona no ha logrado superar el umbral que tiene asignado, por lo cuál su estado no generará un pulso. Estos pulsos representan diversas características encontradas en las imágenes. En la Fig. 2 se muestra el procesamiento de una imagen de 752 x 608 píxeles en escala de grises de un conjunto de células normales. La Fig. 2a) representa la imagen original en donde cada píxel de esta imagen corresponde al estimulo de entrada de cada neurona. La Fig. 2b) muestra los pulsos que generan la imagen binaria y que representan la textura del citoplasma en la imagen, mientras que la Fig. 2c) representa la separación de los bordes celulares de la imagen.

Uno de los principales modelos de redes neuronales pulsantes es el propuesto por Eckhorn [3]. La estructura comprende una neurona por cada píxel y se alimenta de tres componentes de entrada: alimentación ($F_{ij}$) y enlaces ($L_{ij}$) entre neuronas vecinas, junto con el píxel ($S_{ij}$) de la imagen que corresponde al estimulo de la neurona.

Cada uno de estos componentes se encuentran en comunicación con las neuronas vecinas a través de los pesos sinápticos $M_{jk}^n$ y $W_{jk}^n$. Estos pesos almacenan su estado anterior, pero con un factor exponencial negativo ($\alpha_L$ y $\alpha_F$). Una representación de este modelo se muestra en la Fig. 3. Las ecuaciones que lo definen se muestran a continuación:

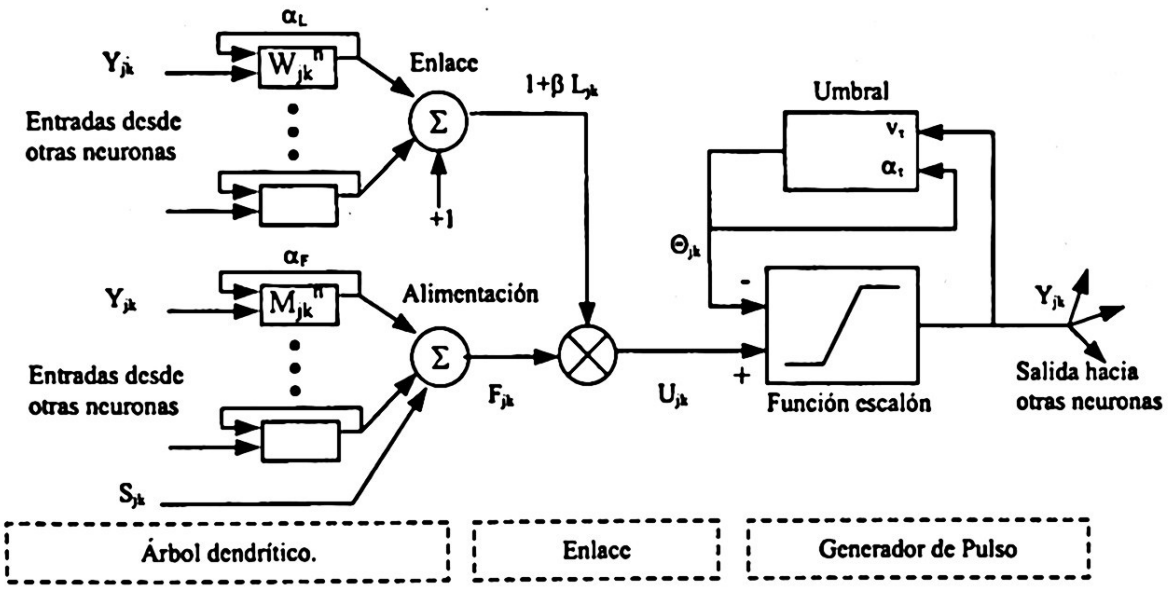



**Fig. 3.** Representación esquemática de un elemento en la red neuronal pulsante.

$$F_{ij}[n] = e^{\alpha} F_{ij}[n-1] + S_{ij} + V_F \sum_{kl} M_{ijkl} Y_{kl}[n-1] \tag{1}$$

$$L_{ij}[n] = e^{\alpha} L_{ij}[n-1] + V_L \sum_{kl} W_{ijkl} Y_{kl}[n-1] \tag{2}$$

$$U_{ij}[n] = F_{ij}[n](1 + \beta L_{ij}[n]) \tag{3}$$

$$Y_{ij}[n] = \begin{cases} 1 & U_{ij}[n] \rangle \Theta_{ij}[n-1] \\ 0 & \text{en otro caso} \end{cases} \quad (4)$$

$$\Theta_{ij}[n] = e^{\alpha_\Theta}\Theta_{ij}[n-1] + V_\Theta Y_{ij}[n] \quad (5)$$

El cálculo iterativo de cada una de las ecuaciones anteriores permite la generación de las imágenes binarias para cada uno de los ciclos de ejecución *n*. La ecuación (1) de la alimentación, corresponde a las entradas que se tienen en la neurona. Estas entradas contienen la versión anterior de la alimentación con un factor de decaimiento, el estímulo o píxel de entrada y la contribución de las neuronas vecinas que comparten sus entradas a la neurona en estudio. Para el caso de la ecuación 2 se omite el estímulo y su diferencia radica en que la contribución de las neuronas vecinas se obtiene de su salida. Los términos constantes $V_F$ y $V_L$ se emplean como constantes de normalización para evitar una saturación debida a los cambios en los pesos $M_{jk}^n$ y $W_{jk}^n$. Estas ecuaciones corresponden al *árbol dendrítico* en la Fig. 3

El estado de estos dos grupos se combinan por medio de la ecuación (3), en donde la contribución del grupo de enlace se controla por medio de β. Esta unión permite obtener el estado interno de la neurona. Este proceso se muestra en la sección *enlace* de la Fig. 3.

La ecuación (4) permite evaluar el estado interno de la neurona con un umbral dinámico $\Theta_{ij}$ y es donde se decide si la neurona efectúa un pulso o no. La ecuación (5) ilustra el comportamiento dinámico del umbral, ya que cuando se genera un pulso, el umbral aumenta de manera significativa su valor. La neurona no volverá a disparar hasta que el umbral haya reducido su valor (exponencialmente) de manera importante. La constante $V_\Theta$ determina la amplitud en el aumento del umbral, una vez que la neurona dispara. Este proceso se muestra en la Fig. 3 en la sección *generador de pulso*.

### 3.2 Vectores de características.

El uso de redes neuronales pulsantes para la detección e identificación de objetos generalmente ha sido propuesto como elemento de pre-procesamiento a la hora de segmentar las imágenes en estudio. Un ejemplo de esto se presenta en [2] donde se hace uso de redes neuronales pulsantes para segmentación.

Un estudio realizado por Johnson [10] propone otra manera de aprovechar las características de las redes neuronales en la identificación y caracterización de objetos. Johnson determinó una forma sencilla de convertir las imágenes binarias que representan los pulsos producidos por las neuronas en vectores de información. Su cálculo es para cada imagen binaria que se va produciendo en el momento en que las neuronas efectúan sus pulsos. El vector se ha denominado G (señal en el tiempo) y su obtención se muestra en la ecuación (6).

Este vector cuantifica en un instante del tiempo, la cantidad de neuronas que efectuaron pulsos.

$$G[n] = \sum_{ij} Y_{ij}[n] \tag{6}$$

Las redes neuronales pulsantes en cada ciclo de procesamiento generan imágenes binarias que representan características en las imágenes. Por ejemplo en la Fig. 2 se aprecian características de textura en el citoplasma y en los bordes de las células. También se puede apreciar que el valor G[4] y G[8] correspondiente a cada uno de estos ciclos difieren en su magnitud de manera importante, por lo tanto se propone la obtención de una firma o patrón para imágenes de células cervicales normales y anormales. Esto con el objetivo de valorar la propiedad de discriminación a través del cálculo de este vector.

En la generación de los vectores G para evaluar la identificación de células cervicales normales y anormales, se siguió el siguiente procedimiento:

- Segmentación manual de células normales y anormales de las imágenes muestra. Las imágenes que se obtuvieron abarcaron el núcleo y el citoplasma de la célula, ya que estos son los puntos de diagnóstico más importantes. Las imágenes analizadas fueron en escala de grises.
- Las imágenes de las células se procesaron por medio de una red neuronal pulsante usando valores de inicialización de variables y constantes empleados en el modelo de Eckhorn [3].
- Para la determinación del número de ciclos se seleccionó un total de 10, debido a que en un número mayor los resultados iniciales volvían a generarse sin proporcionar información adicional.
- A partir de cada una de las imágenes binarias producidas por la red se procedió a la generación del vector de información G[n] con n=10.

Un ejemplo del procesamiento de una imagen de célula normal y anormal se muestra en la Fig.4, donde también se ilustra el proceso anteriormente comentado.

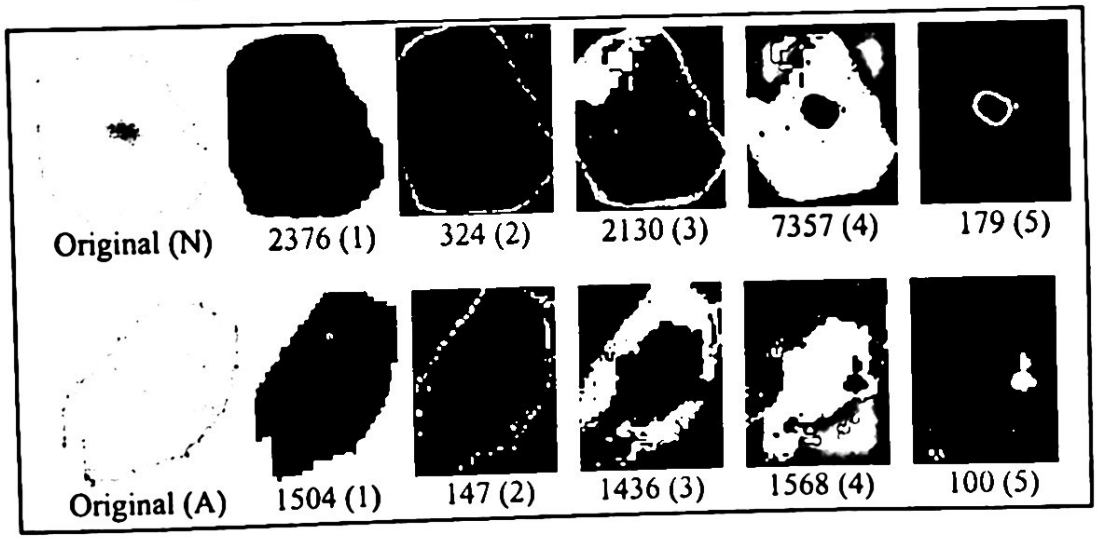



**Fig. 4.** Procesamiento de célula normal (N) y anormal (A) en una red neuronal pulsante. Los valores en la parte inferior de cada imagen corresponden al calculo de G para 5 ciclos y el valor de la derecha corresponde al número de iteración o *n*.

Las imágenes binarias de la Fig. 4 ilustran la diferencia en el procesamiento, ya que por ejemplo en la célula normal (iteración 4) las neuronas correspondientes al citoplasma han generado pulsos debido a la regularidad de la textura; mientras que para el caso de la anormal la poca regularidad presente en su textura permite extraer solo partes parciales de su citoplasma generando diferencias importantes en el valor de G[4]. El mismo caso se presenta en la Iteración 3, donde la regularidad de la célula normal permite extraer su borde, mientras que en la célula anormal debido a su anormalidad sólo se extraen partes parciales del mismo.

Estas características se cuantifican a través del vector G observándose diferencias importantes en su magnitud para cada iteración como se muestra en la gráfica de la Fig. 5. Debido a la escala seleccionada, en las últimas iteraciones no se logra apreciar la diferencia de magnitudes entre los vectores. La diferencia en magnitud se debe a que en las últimas iteraciones las neuronas que efectúan pulsos son aquellas referentes a los núcleos de las células, por lo cual su valor es mucho menor con respecto a los pulsos iniciales de la red neuronal. La Tabla 1 muestra valores numéricos donde se aprecian estas diferencias.

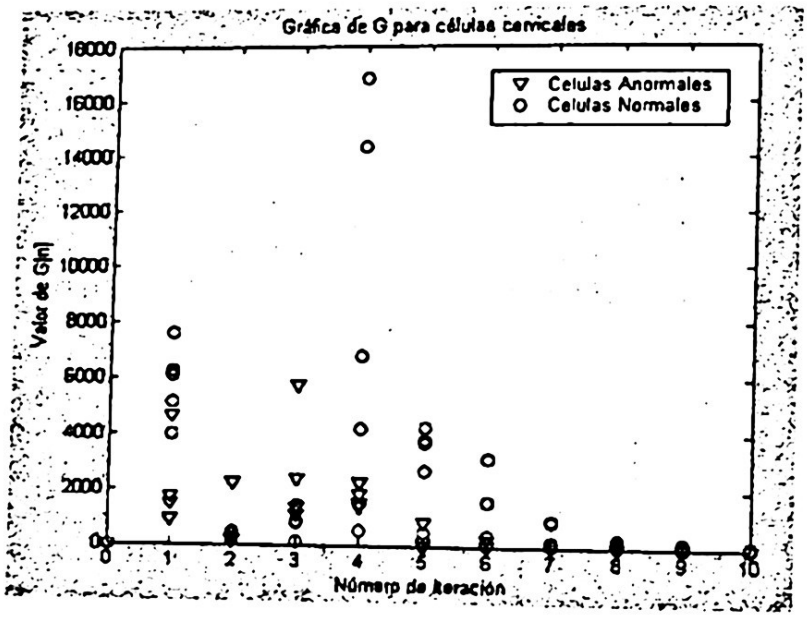



**Fig. 5.** Gráfica de G en una muestra de células cervicales.

**Tabla 1.** Ejemplos de vectores de características a partir de los valores de G (sin normalizar).

| No de Iteración / Célula | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Anormal | 1504 | 147 | 1436 | 1568 | 100 | 2 | 1 | 0 | 0 | 0 |
| Anormal | 1169 | 830 | 921 | 758 | 7 | 0 | 0 | 0 | 0 | 0 |
| Anormal | 1232 | 520 | 2297 | 131 | 0 | 0 | 0 | 0 | 0 | 0 |
| Anormal | 1240 | 68 | 197 | 1309 | 1018 | 62 | 0 | 0 | 0 | 0 |
| Normal | 2376 | 324 | 2130 | 7357 | 179 | 130 | 125 | 48 | 0 | 0 |
| Normal | 5602 | 464 | 2269 | 10846 | 1450 | 210 | 195 | 153 | 1 | 0 |
| Normal | 3961 | 72 | 141 | 969 | 2656 | 1097 | 458 | 143 | 88 | 83 |
| Normal | 3264 | 361 | 734 | 6558 | 5472 | 671 | 84 | 77 | 41 | 0 |

## 4. Clasificación.

La Tabla No 1 contiene los vectores de información para cada una de las células normales y anormales que se obtuvieron mediante el uso de redes neuronales pulsantes en la sección anterior. Estos valores permiten determinar características de citoplasma y núcleo, las cuáles se pueden apreciar con ayuda de la Fig. 4 y la Tabla No 1. En la Tabla No 1 se observa que los primeros valores de las iteraciones indican magnitudes muy grandes. Esto se debe a que en estas etapas las neuronas pulsan con respecto a tonalidades suaves, que corresponden al citoplasma abundante que existe en células normales y al poco citoplasma en las anormales. Así mismo se puede apreciar en las ultimas iteraciones de los valores de la Tabla No 1, que las neuronas se encargan de analizar los núcleos correspondientes a las células. Las redes neuronales pulsantes muestran un comportamiento interesante, ya que mientras en células normales los últimos valores representan un análisis de textura del núcleo, en las células anormales la totalidad del núcleo se obtiene en iteraciones más tempranas. Esto se debe que el tamaño y textura del núcleo ha cambiado con respecto a una célula normal.

El análisis de los pulsos de las neuronas a través de la ecuación (6) permite la generación de características correspondientes al citoplasma y al núcleo de la célula.

Para evaluar la generalización del vector de información en un conjunto de muestras normales y anormales se propuso el uso de la arquitectura de clasificación del modelo de Kohonen [11]. Esta arquitectura se seleccionó por su característica de aprendizaje competitivo y por emplear un esquema de centros de clases que cubren las diversas variantes que pudieran tener las células cervicales.

La arquitectura de la red neuronal de Kohonen consistió en una entrada de 10 neuronas. Estas entradas corresponden a los vectores de características de las células que se encuentran en el vector G de cada célula procesada. Estos vectores fueron previamente normalizados con respecto a los máximos de cada ciclo pulsante $n$.

La última capa de Kohonen contiene dos neuronas de salida, una que representaba las células normales y otra que se encargaba de células anormales. Esta separación de clases se determinó en base al material obtenido en la sección 2.

En el entrenamiento de la red de Kohonen se emplearon un total de 50 células, las cuáles incluían 25 normales y 25 anormales. Esto permitió la generación de los centros de las clases para estos tipos de células cervicales por medio del vector G. A partir de estos centros se procedió a la validación de los patrones aprendidos por la red neuronal de Kohonen. Este proceso se efectuó a través de la presentación de un total de 82 células entre las que se encontraban células normales y anormales. Los resultados de esta clasificación se presentan en la siguiente sección.

## 5. Resultados.

Las pruebas de clasificación realizadas por medio de la red de Kohonen obtuvieron un total de 19 células clasificadas erróneamente. En este total se encontraron 12 falsos positivos (células normales clasificadas como cancerosas) y 7 falsos negativos (células cancerosas clasificadas como normales).

Estos resultados se obtuvieron al presentar en la entrada de la red de Kohonen el conjunto total de las características de las muestras de imágenes de células cervicales

obtenidas en la sección 3.2 (un total de 132 células). El porcentaje de exactitud en la clasificación de células cervicales normales y anormales fue de 86%.

La clasificación de las células se basó en los valores de textura de la imagen de las células, los cuales se obtuvieron por medio de la generación del vector G para cada imagen binaria producida al procesar la célula. Los valores del vector G permitieron cuantificar propiedades de regularidad o anormalidades en los núcleos y citoplasmas celulares, reflejándose estas características de las células en el número de neuronas que generaron pulsos en un determinado ciclo de procesamiento.

El error de clasificación en las células se debió principalmente a que en algunas células las condiciones de iluminación a la hora de obtener las imágenes no fueron similares para todas. Esto ocasionó que algunas neuronas generaran pulsos en ciclos en los cuáles no se esperaban, generando diferencias con respecto a los patrones originales.

Un factor importante a considerar, fue la presencia de algunas imágenes celulares que presentaban zonas de traslape del citoplasma, poli-nucleación, y elementos celulares no deseados sobre el núcleo y el citoplasma del núcleo. Estas imágenes (incluidas en el conjunto de validación) se probaron en el clasificador para probar la robustez del método a variaciones en las células. Los resultados mostraron que estos factores afectaron de manera más notoria la clasificación de células normales (12 de 19), ya que en el caso de anormalidades se tuvieron menos errores (7 de 12). Esto es interesante debido a que es importante tener siempre un índice menor de falsos negativos.

El motivo por el cuál se afectó más el comportamiento de los pulsos en las neuronas normales, se debe a que una célula normal presenta regularidad en el núcleo y su citoplasma; por lo cuál la inclusión de otros elementos afecta esta regularidad produciendo que la neuronas generen pulsos que representan estas irregularidades. En células anormales este tipo de detalles, no afectan de manera importante debido a que las células presentan irregularidades en su citoplasma y núcleo.

En la Fig. 6 se muestra la diferencia en la generación del vector de información debido a problemas de iluminación en las imágenes. La diferencia en los valores se pueden comparar con respecto a los mostrados en la Tabla No. 1, en donde para el caso de células anormales los últimos ciclos no presentaban neuronas que generaran pulsos en sus salidas. Esta diferencia en los vectores tiene su fundamento en la manera en la cuál operan las neuronas de la red neuronal pulsante. El aumento o disminución en los valores de grises permitirá que las neuronas aceleren o retarden sus pulsos en base a los umbrales dinámicos de cada una. Por lo tanto esto genera cambios en la cantidad de neuronas que se disparan en un determinado tiempo $n$, como se observa en los vectores de la Fig. 6 con los de la Tabla 1.

$G_1[n]$= [1003, 39, 59, 718, 1632, 743, 320, 201, 179, 6]
$G_2[n]$= [6442, 50, 40, 231, 1432, 1058, 1094, 520, 340, 174]
$G_3[n]$= [670, 12, 5, 18, 79, 306, 896, 425, 310, 31]

**Fig. 6.** Valores de G para imágenes de células anormales, las cuáles presentan diferencias en las condiciones de iluminación a la hora de su obtención.

## 6. Conclusiones.

En este trabajo se presentó el uso de las redes neuronales pulsantes por medio del modelo de Eckhorn [3], con el objetivo de identificar células cervicales en dos clases: normales y anormales. Una de las propiedades de las redes neuronales es la propiedad de la obtención de características similares de textura, bordes y regiones con valores de grises similares. Este tipo de arquitecturas emplea una neurona para cada píxel de la imagen en análisis.

Las redes neuronales efectúan disparos o pulsan con respecto a un umbral dinámico que se compara con la contribución de las neuronas vecinas dentro de un radio predeterminado. Esto permite la obtención de características de textura en el núcleo y citoplasma de las células en estudio, de manera dinámica en el tiempo. Cada uno de los pulsos generados en las neuronas se cuantifican en el tiempo a través de un vector de información G[n]. A través del cálculo del vector G para cada una de las células se procedió al entrenamiento y validación de una arquitectura de reconocimiento de Kohonen. Esta red de identificación obtuvo un 86% de exactitud en la clasificación de los dos tipos de clases.

Los resultados preliminares obtenidos permiten establecer que a través de la cuantificación de los pulsos por medio del vector G, es posible tener una caracterización de células cervicales. El modelo de Eckhron funciona de manera adecuada aún a pesar de que las imágenes que se tuvieron como muestras presentaban problemas de iluminación y bajo contraste. A pesar de las fallas de reconocimiento (19 células) el cálculo de los pulsos en las neuronas presenta menores errores en la identificación de células anormales, lo cual es un punto importante en la búsqueda de métodos que reduzcan el porcentaje de falsos negativos.

Como trabajo futuro se propone el análisis de las regiones producidas por los pulsos en cada ciclo de ejecución de las células con el objetivo de agregar características de morfología y espaciales del núcleo y citoplasma. También se plantea evaluar la distancia que existe entre la repetición de los patrones de G (usando ciclos de ejecución elevados) con el objetivo de reducir el porcentaje de error, adicionando esta característica a los vectores de aprendizaje.

En la búsqueda de aumentar el porcentaje de eficiencia se pretende ampliar el tamaño de las muestras, así como también el análisis y prueba de otras arquitecturas de redes neuronales, tales como Hopfield y ART. Así mismo se plantea la normalización de la iluminación para tener una mayor uniformidad a la hora de la obtención de las imágenes celulares.

Finalmente se considera que este tipo de redes neuronales pulsantes representa una alternativa interesante en el tratamiento de imágenes médicas desde el punto de vista de la obtención de características y la segmentación. Uno de sus puntos más favorables es la ausencia de técnicas de pre-procesamiento, ya que a partir de las imágenes binarias producidas por la red es posible obtener diversos elementos importantes en las imágenes en estudio. Su estudio y aplicación en un conjunto pequeño de muestras médicas de células cervicales permite comprobar lo anteriormente comentado con buenas perspectivas de resultados favorables en conjuntos de muestras mucho mas amplios.

## Referencias

[1] R. F. Walker, P. Jackway, B. Lovell and I. D. Longstaff (1994). Classification of Cervical Cell Nuclei Using Morphological Segmentation and Textural Feature Extraction. *Second Australian and New Zealand Conference on Intelligent Information Systems.*

[2] H. S. Ranganath and G. kuntimad (1999). Object Detection Using Pulse Coupled Neural Networks. *IEEE Transactions on Neural Networks, 10,* 615-620.

[3] T. Lindblad and J. M. Kinser (1998). *Image Processing using Pulse-Coupled Neural Networks.* Springer-Verlag London.

[4] T. Lindblad and J. M. Kinser (1999). Inherent Features of Wavelets and Pulse Coupled Networks. *IEEE Transactions on Neural Networks, 10,* 607-614.

[5] C. Di Ruberto, A. Dempster, S. Khan and B. Jarra (2002). Analysis of Infected Blood Cell Images Using Morphological Operators. *Image and Vision Computing, 20,* 113-146.

[6] F. Arman and J.A. Pearce (1990). Unsupervised Classification of Cell Images Using Pyramid Node Linking. *IEEE Transactions on Biomedical Engineering, 37,* 647-650.

[7] J. S. Duncan and N. Ayache (2000). Medical Image Analysis: Progress Over Two Decades and The Challenges Ahead. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22,* 85-105.

[8] J. P. Thiran, and B. Macq (1996). Morphological Feature Extraction for The Classification of Digital Images of Cancerous Tissues. *IEEE Transactions on Biomedical Engineering, 43,* 1011-1020.

[9] K.Tumer, N. Ramanujam, J. Ghosh and R. Richards-Kortum (1998). Ensembles of radial basis function networks for spectroscopic detection of cervical precancer. *IEEE Transactions on Biomedical Engineering, 45,* 953-961.

[10] J. L. Johnson (1994). The Signature of Images. *IEEE International Conference on Neural Nets,* Orlando.

[11] A. S. Pandya and R. B. Macy (1996). *Pattern Recognition with Neural Networks in C++.* CRC Press, Inc.

# Using Neural Networks for Pattern Recognition in Robotic Tasks

I. Lopez-Juarez, J. Corona

CIATEQ A. C., Centro de Tecnología Avanzada,
Mechatronics and Intelligent Manufacturing
Systems Research Group (MIMSRG)
Manantiales 23A, Fracc. Ind. B.Q.,
El Marqués, Querétaro. CP 76246. México.
(ilopez, jcorona@ciateq.mx)

## Abstract

Assembly tasks by robots have traditionally depended on simple sensing systems and the robot manufacturers programming language. However, this restricts the use of robots in complex manufacturing operations. An alternative to robot programming is the creation of self-adaptive robots based on Artificial Neural Networks (ANN's) in order to use industrial manipulators in unstructured working environments.

The research presented in this article shows how force sensing data can be used by an ANN to map contact force states with robot's motion. The methodology has been succesfully tested during assembly tasks showing that an industrial robot can learn different operations. The method is generic and it has been tested using two industrial manipulators.

## 1 Introduction

Industrial robots are reliable machines. The success of the operation is based on the accuracy of the robot itself and the precise knowledge of the environment, i.e. information about the geometry of the parts to be assembled, located, machined, welded, etc. Combining these elements and using the manufacturers programming language efficient programs can be written. When parameters change, the robot program has to be amended to take into account new conditions. The adaptation to these new conditions is explicitly given by the programmer. Industrial robots are currently being programmed using this technique, hence, robots are still unable to be self-adaptive to varying conditions and this is possibly one of the major drawbacks that has limited their extensive use in manufacturing.

Techniques are sought to provide self-adaptation for robots. Robot manipulators operate in real world situations with a high degree of uncertainty and require sensing systems to compensate from potential errors during operations. Uncertainties come from a wide variety of sources such as robot positioning errors, gear backlash, arm deflection, ageing of mechanisms and disturbances. Controlling all the above aspects would certainly be a very difficult task, therefore a simpler approach based on force control is preferred. By using force control the overall effect of the contact force between the environment and the manipulator are considered as a whole. Robots can also recognise force/torque patterns and operate in unstructured environments specifically performing assembly tasks.

Related work using ANN is reviewed first. The system architecture is described and the nature of the force patterns explained. Based on this background the ANN Adaptive Resonance Theory (ART) is presented continuing with the description of the Neural Network Controller (NNC). Experimental results showing the NNC's performance are discussed and finally some conclusions given.

## 2 Connectionist Models and Related Work

Force control can be roughly divided in Model-based and Connectionist-based approaches. The model-based approach takes as much information of the system and environment as possible. This information includes localisation of the parts, geometry of the parts, materials, friction, etc. The connectionist-based approach is based on connectionist models and its robustness relies on the information given during the training stage that implicitly considers all of the above parameters. Model-based methods do not offer a complete solution due to the uncertainties associated during assembly as mentioned earlier. On the other hand, connectionist-based techniques have proved to work reliably when uncertainty is involved due to their generalisation property.

The use of connectionist models in robot control to solve the problem of assembling parts under uncertainty has been demonstrated in a number of publications, either in simulations [1], [4], [5], or being implemented on real robots [6], [7], [8]. In these methods, Reinforcement Learning (RL), unsupervised and supervised type networks have been used. The reinforcement algorithm implemented by V. Gullapalli demonstrated to be able to learn circular and square peg insertions. The network showed a good performance after 150 trials with insertion times lower than 100 time steps [9]. Although the learning capability demonstrated during experiments improved over time the network is unable to generalise over different geometries. Insertion are reported with both circular and square geometries, however, when inserting the square peg, its rotation around the vertical axis was restricted otherwise the insertion would not have been possible. M. Howarth followed a similar approach, using Backpropagation in combination with reinforcement learning. During simulation it was demonstrated that 300 learning cycles were needed to achieve a minimum error level

with his best network topology during circular insertions [8]. A cycle meant to be an actual motion that diminished the forces acting on the peg. For the square peg, the number of cycles increased dramatically to 3750 cycles. These figures are important, especially when fast learning is desired during assembly. On the other hand, E. Cervera using SOM networks and a Zebra robot (same used by Gullapalli) developed similar insertions as the experiments developed by Gullapalli. Cervera in comparison with Gullapalli improved the autonomy of the system by obviating the knowledge of the part location and used only relative motions. However, the trade-off with this approach was the increment of the number of trials to achieve the insertion [6], the best insertions were achieved after 1000 trials. During Cervera's experiments the network considered 75 contact states and only 8 out of 12 possible motion directions were allowed. For square peg insertions, there were needed 4000 trials to reach 66% success of insertion and that did not improved any further. According to Cervera's statement, " We suspect that the architecture is suitable, but the system lacks the necessary information for solving the task". The situation clearly recognises the necessity to embed new information in the control system as it is needed, which is likely to be achieved with an architecture such as ART.

# 3 The Research

In our research, the robot is provided only with contact force information and a Primitive Knowledge Base (PKB), which is an initial contact force-action mapping that bias its initial reactions to constrained forces. No information is given about the localisation of the parts. The arm increases its knowledge on-line based on the success of the predicted motion. The robot actually increases and enhances its knowledge as the operation progresses. The time that the robot takes to complete a similar operation is reduced and also mistakes made earlier do not recur, which demonstrates the new expertise of the robot.

The design of the novel Neural Network Controller (NNC) is founded on the strength of ART networks to learn incrementally. The new information is acquired as the operation develops without affecting the knowledge that was previously learnt. The Fuzzy ARTMAP algorithm is used and the NNC training made on-line. The number of contact force patterns that the NNC can accommodate in its knowledge is limited only to memory storage. The switching mechanism of the NNC is regulated by the development of the operation. New knowledge information is only accepted in the Knowledge Base when it has strongly contributed towards the success of the assembly. The resulting Enhanced Knowledge Base (EKB) at the end of the assembly can be used for similar operations. Results on industrial robots demonstrate that the robot's skill improves effectively and the insertion times and the errors diminish over time. Furthermore this is, to the best knowledge of the authors, the first time the Fuzzy ARTMAP network has been applied to an industrial robot manipulator.

# 4 System Architecture

The hardware architecture is formed by an industrial master computer in whic the DSP based F/T sensor card resides, the industrial manipulator KUK KR15, KRC2 controller, Kuka Control Panel (KCP) and JR3 F/T sensor illustrated in Figure 1. The main units of the robot system are the KRC controller and the robot arm itself. Power and data are transmitted betwee the two units through two interconnecting cables. The KRC2 controller hous the components that control and power the robot arm. The master comput communicates with the controller via serial port using the Xon/Xoff proto col. Additionally, the robot is also provided with vision and speech recognition systems, which enable the robot to recognise simple 2D geometries and to commanded via voice. For further details on these other systems the reader referred to [3]

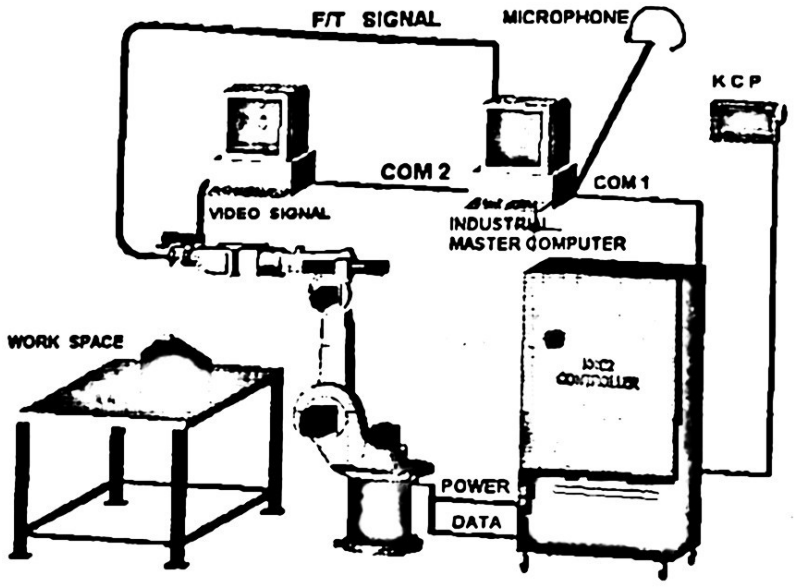



Figure 1: System Architecture

# 5 The problem and nature of forces

Figure 2(a) shows a typical peg in hole insertion, which is a canonical operation for performance assessment. The force traces occurring during this type operation are given in Figure 2(b).

This type of signal is normally acquired by using a F/T sensor mounted in robot's wrist. The sensor provides the required input information to the NNC. The signal patterns contain information regarding the force and torque "felt" at the robot wrist. With this information is possible to determine how much force is being applied to the end-effector or gripper[1] and how these forces affect

[1]The gripper is a mechanical device to grasp and hold the assembly part. which normally consist of two or more fingers.

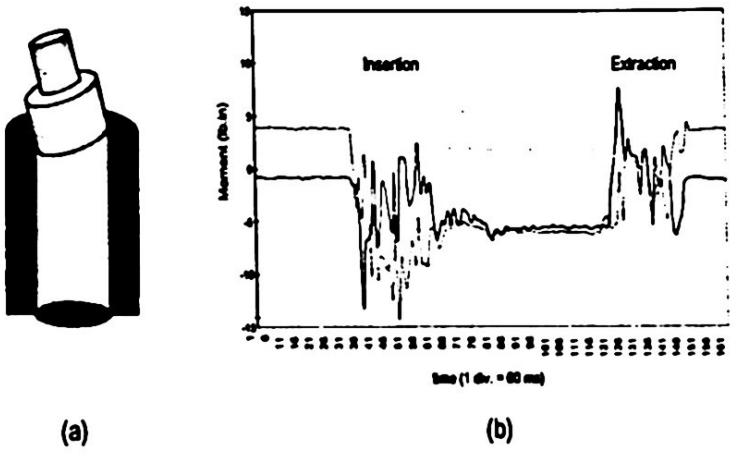



Figure 2: (a) Peg-in-hole insertion (b)Contact force

the orientation of the peg by means of the moment value and sign. Although the graph only shows the moment information, the information available to the NNC is a F/T vector containing six elements i.e. fx, fy, fz, mx, my and mz.

## 5.1 Nature of contact forces

Peg-in-hole operations were carried out using pegs with different cross-section: circular, square and square with a rounded corner, (this peg will henceforth be referred to as a radiused-square peg). The female parts were made with chamfer at 45°. A typical force trace of the insertion and extraction of the square peg is shown in Figure 3a. It should be noted that magnitudes were scaled to have better interpretation of the proportionality and similarity properties between force and moment signals. Signals corresponding to $Z$ axis (insertion direction in tool coordinates) are not given since they were completely different.

It was observed that signals corresponding to peg insertions with symmetric cross-section (square and circular) followed a similar pattern, while for the radiused-square peg insertion the patterns were totally different. This occurred because the distribution of the contact forces on the pegs were also dissimilar due to the non-symmetric shape of the peg. Therefore, it can be said that the type of pattern depends on the *magnitude* of the force applied during assembly and the *shape* of the peg but, it is important to mention that this assumption is valid only when both mating pairs are aligned, that is, the peg has been aligned perpendicular to the female component. Additionaly, the type of proportionality and cross-correlation changes according to the *offset* location of the peg whitin the $X$-$Y$ plane quadrants shown in Figure 3b.

The peg and the female block are shown in top view. The female component has been enlarged to make obvious the placement of the peg within quadrants. The level of cross-correlation between force and moment patterns depends on the placement of the peg within these quadrants. An important observation is that the correlation was related to the symmetry of the mating pairs. Circular and

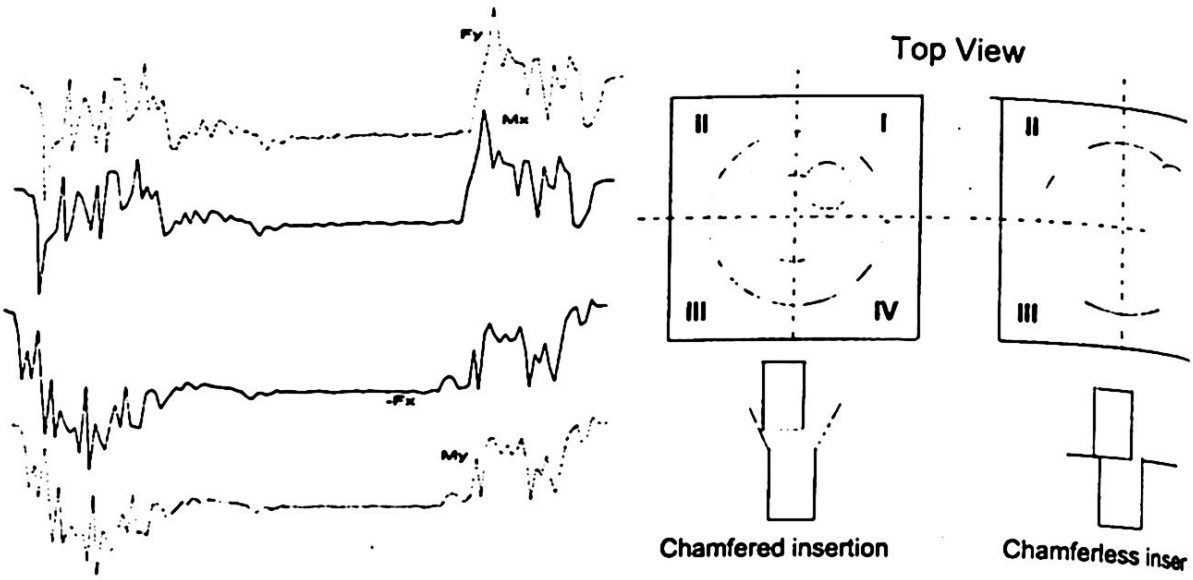



Figure 3: (a) *Insertion-extraction square peg* *(b) Symmetry and contact forces*

square pegs showed higher correlation (infinite rotational symmetry) compared to the radiused-square peg (one rotational symmetry).

The complexity of the system becomes evident, especially if the infinite number of patterns that can be generated through the assembly process considered. To deal with this complexity the proposed NNC has to classify recognise these patterns first.

# 6 ART and FuzzyARTMAP

The Adaptive Resonance Theory (ART) was developed by Stephen Grossberg and Gail Carpenter at Boston University. Different model variations have been developed to date based on the original ART-1 algorithm [10]. The mechanics a basic ART module are as follows: It consists of two subsystems as illustrated in Figure 4. The attentional subsystem is made up of two layers of nodes $F_1$ and $F_2$. In an ART network, information in the form of processing-element output reverberates back and forth between layers. If a stable resonance takes place learning or adaptation can occur. On the other hand, the orienting subsystem is in charge of resetting the attentional subsystem when an unfamiliar event occurs.

A *resonant state* can be attained in one of two ways. If the network has learned previously to recognise an input vector, then a resonant state will be achieved quickly when that input vector is presented. During resonance, the adaptation process will reinforce the memory of the stored pattern. If the input vector not immediately recognised, the network will rapidly search through its stored patterns looking for a match. If no match is found, the network will enter a resonant state whereupon the new pattern will be stored for the first time. Thus, the network responds quickly to previously learned data, yet remains able

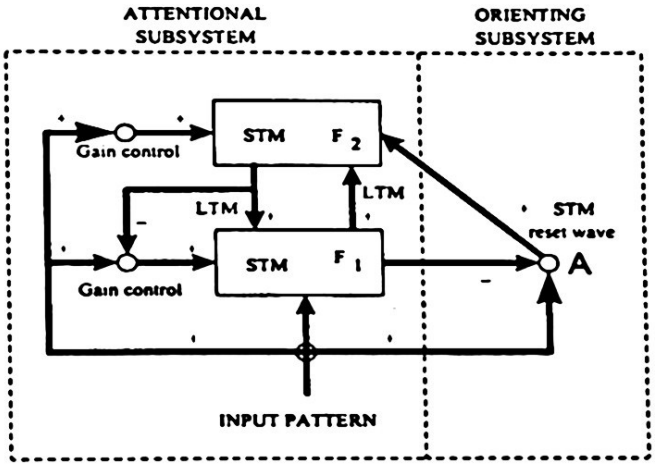



Figure 4: ART architecture

learn when novel data is presented, hence solving the so-called *stability-plasticity* dilemma. The activity of a node in the $F_1$ or $F_2$ layer is called *short-term* memory (STM) whereas the adaptive weights are called *long-term* memory (LTM). Gain controls handle the discrete presentation of the input signals. A vigilance parameter measures how much mismatch is tolerated between the input data and the stored patterns, which can be used to control the category coarseness control of the classifier.

Supervised learning is also possible through ARTMAP that uses two ART modules or its variants, such as Fuzzy ARTMAP (FAM) that incorporates fuzzy set theory operations in order to handle analogue data between 0 and 1 [11]. The NNC was designed based on this FAM network due to its capabilities of fast incremental learning (typically one epoch). The mechanics of the NNC which incorporates the FAM network is explained in the next section.

# 7 Neural Network Controller (NNC)

The functional structure of the assembly system is illustrated in Figure 5. The FAM is the heart of the NNC. The controller includes three additional modules. The Knowledge Base that stores the initial information related to the geometry of the assembling parts. This information is used only during the first assembly operation, later this is enhanced by patterns that favour the assembly and whose inclusion is regulated by the Pattern Selector Module. The Pattern Selector section keeps track of the F/T patterns and verifies whether the action is good enough to allow the FAM network to be retrained. If this is the case, the switch SW is closed and the corresponding pattern-action provided to the FAM for on-line retraining.

Future predictions will be based on this newly trained FAM network. The Automated Motion module basically is in charge of sending the incremental motion request to the robot controller. External components to the NNC are

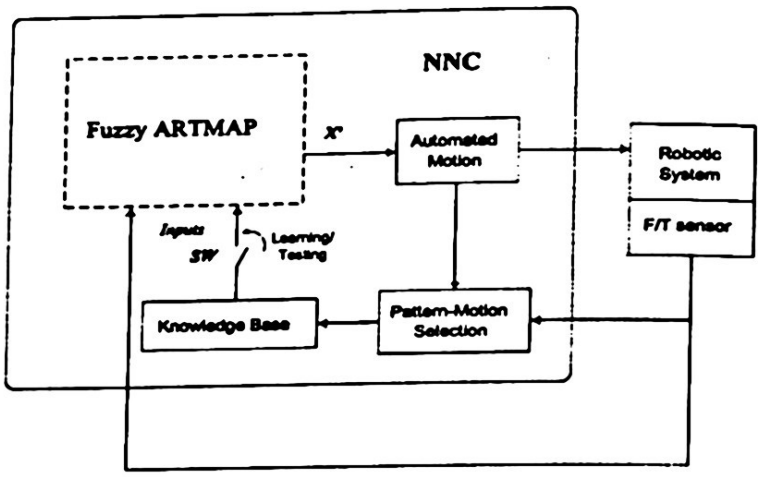



Figure 5: Neural Network Controller (NNC)

the robot controller, image system, the manipulator itself and the F/T sensor that provides the pattern information. The programs for the NNC were created using Visual C++ 6.0 and implemented in a 800MHz Pentium III Industrial Computer.

## 7.1 Initial training and PKB formation

The formation of the PKB basically consists of showing the robot how to act to individual components of the F/T vector. The influence of each vector component requires a motion opposite to the direction of the applied force diminish its effect. The procedure is illustrated in Figure 6. For simplicity, only the lower arm of the manipulator has been shown.

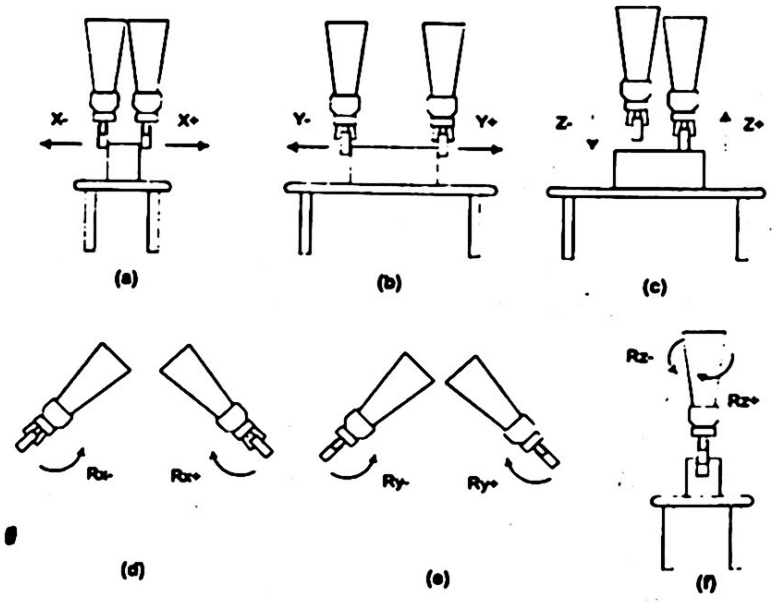



Figure 6: Training Procedure

Every motion of this type is referred to as a Primitive Motion (PM) and the idea is to teach the robot *where* to move when single F/T components, i.e. fx, fy, fz, mx, my, or mz are applied to the workpiece. Figure 6 illustrates the PM needed to diminish the corresponding constraint force in all possible motions. The storage of the F/T vector and the PM will form the PKB that is required to start the assembly for the very first time. Once the first insertion has been completed, the robot may possibly have increased its knowledge. If so, the PKB is enhanced and an Enhanced Knowledge Base (EKB) version will be used during the following insertion.

The PKB used during our experiments is shown in Figure 7, The F/T data from the sensor was scaled to the range [0,1], where the extreme values 0 and 1 corresponded to a force of -15 lb and +15 lb respectively. Negative values were assigned to the interval [0,0.5) and positive values were assigned to the interval (0.5,1]. It should be noted that the origin in the graph is set to 0.5, where positive and negative values are represented in the upper and lower halves of the graph respectively. Every column corresponded to an input vector to the network. The corresponding assigned output vector is shown at the top of the graph for each pattern.

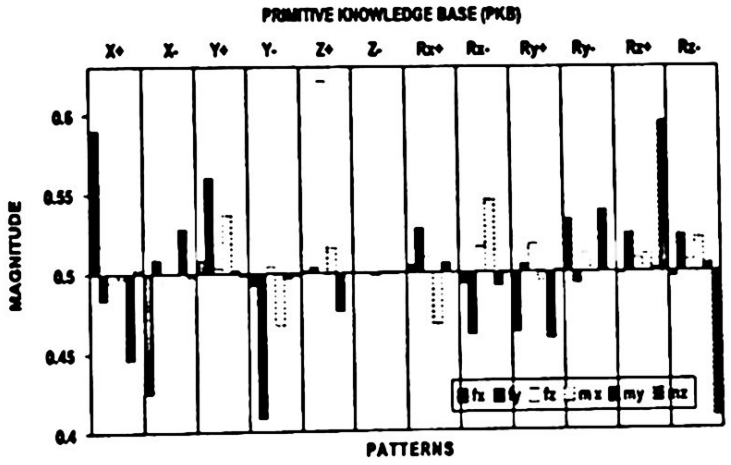



Figure 7: PKB

# 8 Pattern-Motion selection and knowledge enhancement

There are potential problems associated with the learning mechanism which are solved by the Pattern-Motion Selection module. The robot should continue moving in the insertion direction if, and only if, a minimum force value has been reached. This situation should trigger the learning mechanism in order to allow the acquisition and learning of the pattern-action pair that produced such a

situation. In the event of continual learning after having reached this point, the performance of the NNC might decay. This situation is similar to what is known as overtraining, overfitting or overlearning in ANNs. At this point the learning should be stopped because if the robot learns other patterns under the above circumstances, eventually the minimum force value will be different leading wrong motions. The same applies to the condition when the end-effector meets a force higher than the force limit. There should not be any further learning during this situation since learning a higher force would probably damage the sensor.

The above situations can be resumed in three fundamental questions:

1. What is a good motion?
2. How to recover from errors?
3. Which motions should or should not be learned?

Having an assembly system which is solely guided by contact force states, the criterion to decide whether the motion was good enough to be learnt based on the following expression:

$$F_{after} < 0.1 * F_{before} \tag{1}$$

$F_{after}$ and $F_{before}$ are computed using the following equation:

$$F = \sqrt{fx^2 + fy^2 + fz^2 + mx^2 + my^2 + mz^2} \tag{2}$$

Expression 1 means that if the total force after the incremental motion significantly reduced then that pattern-action will be considered good to be included in the knowledge base. Experiments showed that if this threshold value was set higher (*i.e.* $\geq 0.3 * F_{before}$) the network became very sensitive and showed overtraining behaviour.

Forces that are higher than the value given by $0.1 * F_{before}$ and lower than the $F_{limit}$ are still good values. However, the corresponding pattern-action pair will only be used during network recall. This situation is illustrated in Figure 8 that shows three possible situations: learning, recall and error recovery. The third area is a situation where $F \geq F_{limit}$. In this situation the user is alerted and asked to reposition the arm.

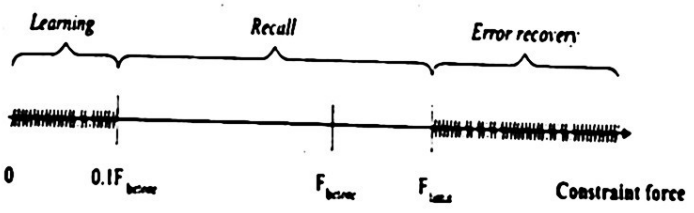



Figure 8: Learning, Recall and Error Recovery

There will be ambiguous situations in which learning should not be permitted. This applies to patterns in the insertion direction (usually Z direction). Consider downward movements in the Z- direction. At the time the peg makes

contact with the female block, there may well be a motion prediction in the Z+ direction. This recovery action will certainly diminish the contact forces and will satisfy the condition given by the expression 1 in order to learn the force-action pair. However, this situation is redundant since it was given when the PKB was formed and it is likely that it will corrupt the PKB. Similarly, learning should not be allowed when the arm is in free-space. In this situation, $F_{after}$ and $F_{before}$ will be very similar and again learning another pattern in the Z- direction will be redundant. Both situations were tested experimentally by the author and revealed that an unstable situation may appear if further learning is allowed in the insertion direction.

After the pattern-action has satisfied expression 1 and the prediction direction is not in the Z direction, the pattern is allowed to be included in the new "expertise" of the robot, the EKB. Patterns that do not satisfy expression 1 and whose values are lower than the $F_{limit}$ will only be used to recall previous knowledge. The knowledge refinement process will continue in the NNC until the end-condition is satisfied.

# 9 Results

The testbed for the assembly experiments is shown in Figure 9a and a typical peg in chamfered hole insertion is shown in Figure 9b. The Intelligent assembly was carried out using aluminium pegs with different cross-sectional geometry: circular, square and radiused-square. Clearances between pegs and mating pairs were 0.1 mm. The assembly was ended when 3/4 of the body of the peg was inside the hole.



Figure 9: (a) Testbed (b) Peg in hole assembly operation

The Fuzzy ARTMAP network parameters during experiments were set for fast learning (learning rate = 1). The base vigilance $\overline{\rho_a}$ had a low value since it has to be incremented during internal operations. $\rho_{map}$ and $\rho_b$ were set much higher to make the network more selective creating as many clusters as possible.

The vigilance parameters used for the experiments reported in this article are as follows: $\overline{\rho_a} = 0.2$ (base vigilance), $\rho_{map} = 0.7$ and $\rho_b = 0.9$.

## 9.1 Expertise test

Results from several peg-in-hole operations have shown that the robot can acquire manipulative assembly skills. At the starting of the operation the robot has only the PKB which is being enhanced though the assemblies. The robot demonstrates its expertise by reducing the number of alignment motions and consequently the insertion speed. The acquisition of the skill takes approximately 1 minute [2].

Figure 10a and Figure 10b illustrate two different learning situations in this operation. Figure 10a shows an insertion directed by the NNC and with learning enabled, this meant that the NNC was allowed to learn new patterns if the expression $F_{after} < 0.1F_{initial}$ was satisfied. After 14 insertions the robot had learned 9 new patterns, which complemented the knowledge base. From this insertion onwards there were no further patterns learned.

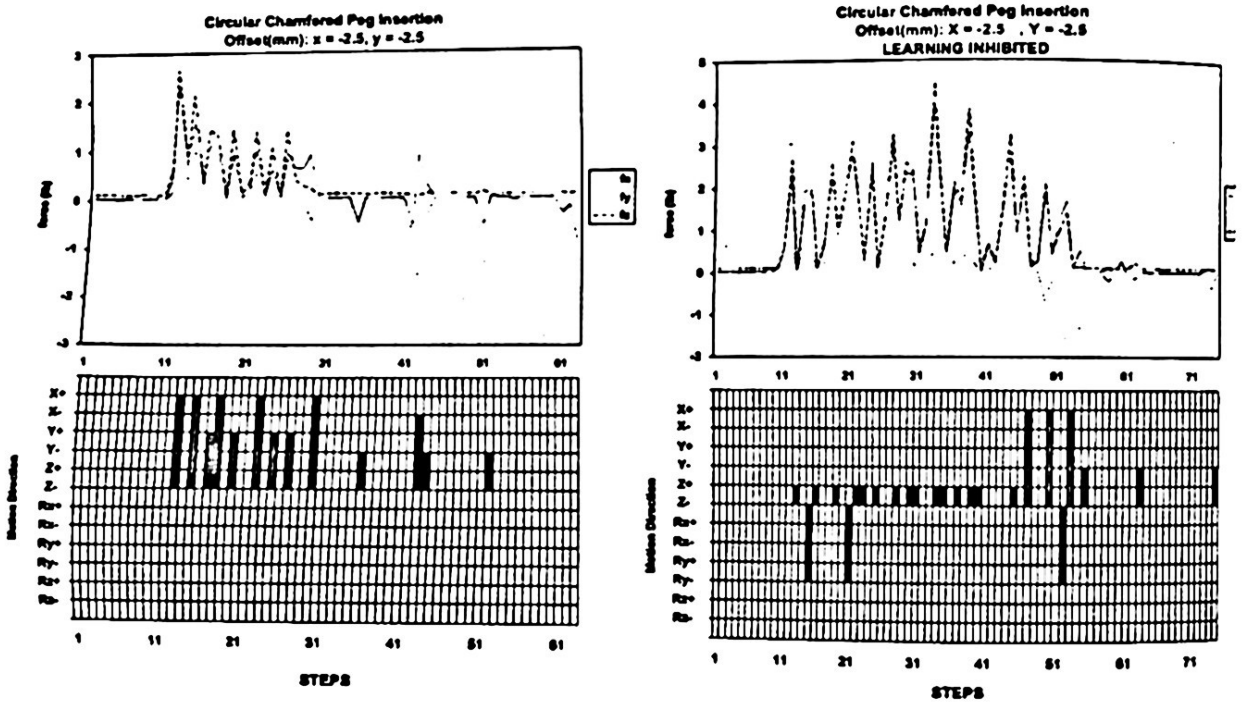



Figure 10: (a) learning enabled (b) learning inhibited

For comparison purposes, another insertion with the same offset was carried out, but in this case the robot's learning capability was inhibited (See Figure 10b). This means that the robot uses solely the PKB and no patterns are allowed to be learned during the operation. In both Figures, the upper graph represents the force traces whereas the motion directions commanded by the NNC are given in the lower graph. In the Motion Direction graph, the horizontal axis corresponds with the Z- direction. Bars above the horizontal axis represent linear alignments and below the horizontal axis represent angular alignments.

Despite that the offset was the same, the number of alignment motions and insertion time were higher. With the learning inhibited, the robot was not

allowed to learn contact states within the chamfer hence the NNC generated motions based only on its initial PKB. This resulted in motions that produced an excessive fz. As a result, the NNC predicted a series of compensatory movements in Z+ and Ry- to recover from these situations. The robot was ultimately able to insert the workpiece, however the performance was poorer in terms of alignment and consequently speed.

It can clearly be observed that the same operation with the same offset can be achieved more efficiently and faster if the robot uses the EKB. In other words, the robot shows its *dexterity* when it is allowed to use its expertise.

## 9.2 Density of data and knowledge acquisition

The capability of generalisation and knowledge acquisition of the NNC has been demonstrated. Patterns that reduce significantly the contact forces during manipulations were acquired into the knowledge base and learnt. In the circular chamfered insertion example the network was initially trained with the PKB containing the 12 possible patterns associated with the robot's 6 DOF. This information biased the initial learning by creating 12 categories to allocate every possible motion direction. From these results, it was verified that subsequent patterns corresponding to contact states within the chamfer were effectively allocated into these categories. However, the pattern population within certain categories produced high density of data within regions in the feature space. For instance in X+ direction, which is explained below.

During the chamfered circular peg insertion only four patterns were learnt. These patterns corresponded to the X+, X-, Y+ and Y-. This can be appreciated in Figure 11a that shows the nature of learned patterns. The new patterns were valuable to speed up the insertion and to improve the insertion trajectory as it was shown during the test. However, these patterns were present within the data more than once and a total of 13 patterns were acquired after 14 insertions which implied that certain categories were more populated.

As it can be seen, patterns belonging to the same category were very similar. The patterns corresponding to the X+ direction were allowed to be learnt 8 times. This implied that the contact forces were significantly reduced in eight occasions. This high number of patterns populated more the feature space in that area, which is represented in Figure 11b.

For simplicity, only four major areas of action (X+, X-, Y+ and Y-) are represented. Initially, the main groups are formed, this is represented by the big black dots as illustrated at the beginning of the operation using what has been termed PKB. The smaller dots represent additional patterns that have been clustered within the same major region. As it is observed, the region belonging to the X+ direction was more populated than in the others. The high density of data only implies that there are more data in the region and the cost is memory space. However, since the criteria to learn new patterns was the condition given by the expression $F_{after} < 0.1 * F_{before}$, then as the learning progresses, a reduction in contact forces is expected, as it was demonstrated during the experiments, since the robot became more skillful. Being this statement true, it

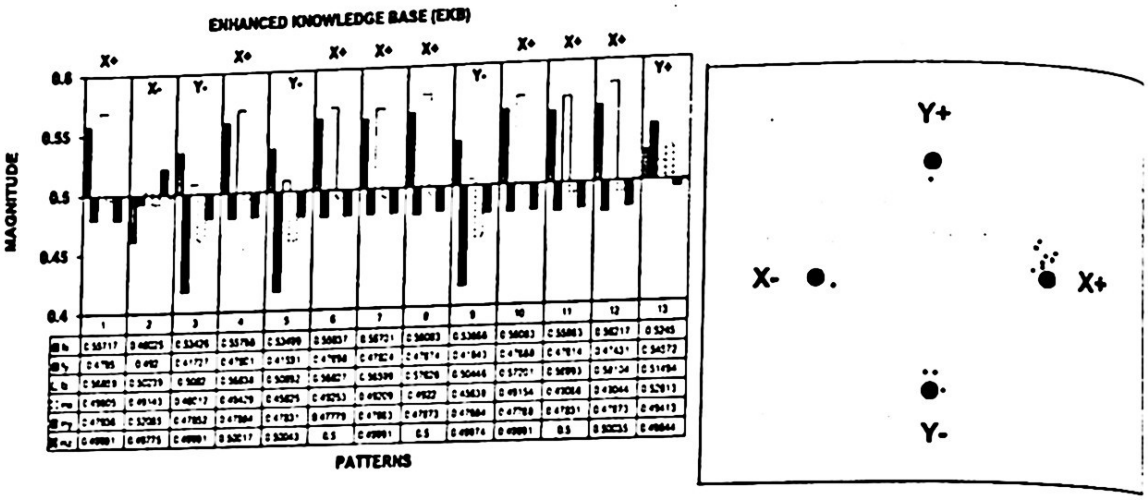



Figure 11: (a) Learned patterns during insertion (b) Data density

is also true that the knowledge acquisition becomes more strict. This obeys to the fact that forces are smaller as the robot is more skillful and from the above expression forces have also to be smaller to be accepted into the EKB. Also, as the robot's dexterity improved, the trend in the number of patterns that were accepted into the EKB decreased. The above expression for allowing the patterns to be learnt resulted to be a criterion to stop automatically the learning. With this reasoning in mind, it can be demonstrated that the density does not corrupt the selectivity of the NNC, but only affects the memory resources to allocate the learned patterns.

# 10 Conclusions

Results from our experiments demonstrate that industrial manipulators can learn manipulative skills on-line using only contact force pattern information. The use of the Fuzzy ARTMAP based Controller has provided the on-line learning capability needed by the task. The robot is able to learn incrementally new patterns and consequently new assembles and to improve effectively its skills from experience. The methodology is generic and has been tested in two industrial manipulators [12], [2]. On going work in looking at the implementation of an adaptive preprocessing stage which will allow the system to have the input pattern values scaled according to the current input range. The design of a Multimodal ART architecture to reinforce the confidence in the motion prediction has been envisaged. With this architecture we expect to fuse image and tactile information to aid complex operations.

## Acknowledgements

The authors wish to thank The Mexican and The Queretaro's Science and Technology Councils (CONACyT, CONCyTEQ) for partially funding this project.

# References

[1] I. Lopez-Juarez, M. Howarth, K. Sivayoganathan. *Robotics and Skill Acquisition.* A. Bramley; T. Mileham; G. Owen. (eds.): In Advances in Manufacturing Technology X. ISBN 1 85790 031 6. (1996) 166-170

[2] I Lopez-Juarez, M Howarth *Knowledge Acquisition and Learning in Unstructured Robotic Assembly Environments.* The International Journal of Information Sciences, Special Issue on Intelligent Learning and Control of robotics and Intelligent Machines in Unstructured Environments. To appear.

[3] I. Lopez-Juarez, M Pena, J Corona, K Ordaz, F Aliew. *Towards the Integration of Sensorial Perception in Industrial Robots* Fifth International Conference on Application of Fuzzy Systems and Soft Computing Milan, Italy. September 17-19, 2002. 218-223. ISBN 3-933609-13-5.

[4] H. Asada. *Teaching and Learning of Compliance Using Neural Nets.* IEEE Int Conf on Robotics and Automation. (1990) 1237-1244

[5] Enrique Cervera, Angel P. del Pobil. *Learning and Classification of Contact States in Robotic Assembly Tasks.* Proc of the 9th Int. Conf. IEA/AIE. (1996)725-730

[6] Enrique Cervera, Angel P. del Pobil. *Programming and Learning in Real-World Manipulation Tasks.* IEEE/RSJ Int. Conf. on Intelligent Robot and Systems. (1997) 471-476

[7] Vijaykumar Gullapalli, Judy A Franklin, Hamid Benbrahim. *Acquiring Robot Skills via Reinforcement Learning* IEEE Control Systems. (1994) 13-24

[8] Martin Howarth. *An Investigation of Task Level Programming for Robotic Assembly.* Ph.D. Thesis. The Nottingham Trent University. (1998)

[9] Vijaykumar Gullapalli. *Skillful control under uncertainty via direct reinforcement learning.* Robotics and Autonomous Systems. (1995) 237-246

[10] Gail A. Carpenter, Stephen Grossberg. Computer Vision, Graphics, and Image Processing. *A Massively Parallel Architecture for a Sel-Organizing Neural Pattern Recognition Machine.* Academic Press, Inc. (1987) 54-115

[11] Gail A. Carpenter, Stephen Grossberg, Natlya Markunzon, John H. Reynolds; David B. Rosen. *Fuzzy ARTMAP: A Neural Network Architecture for Incremental Learning of Analog Multidimensional Maps.* IEEE Transactions on Neural Networks. Vol. 3 No. 5 (1992) 698-713

[12] I. Lopez-Juarez, M. Howarth, K. Sivayoganathan. *An Adaptive Learning Approach to Control Contact Force in Assembly.*IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'98). Vol 3.(1998)1443-1448

# Cellular Neural Networks and Mobile Robot Vision

Marco Balsi[1] and Xavier Vilasís-Cardona[2]

[1]Dipartimento di Ingegneria Elettronica, Università "La Sapienza", via Eudossiana 18, 00184 Rome, Italy - balsi@uniroma1.it

[2] Departament d'Electrònica, Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, Pg. Bonanova 8, 08022 Barcelona, Spain - xvilasis@salleURL.edu

**Abstract**

We propose Cellular Neural Networks (CNNs) for solving the visual-feedback problem in robot guidance. In particular, we show how CNNs provide the necessary image processing to guide an autonomous mobile robot in a maze made of black lines on a light surface. The system consists of a fuzzy controller performing the elementary navigation tasks fed by the result of processing the image only by CNN techniques. We use this solution to make some considerations on more difficult problems such as curved line following and obstacle avoidance.

**keywords:** Cellular Neural Networks, Robot Vision, Fuzzy Logic.

## 1 Introduction

A challenge in automous robotics is navigation in unstructured environments with vision-based algorithms. This problem has been tackled with sophisticated image processing techniques, ranging from mathematical morphology to artificial neural networks, and partial solutions have been achieved: algorithms exist to solve obstacle avoidance, cliff detection, autonomous highway driving or active stereo vision for tracking. However, these methods require a large amount of computing effort, forcing a trade-off between real autonomy and real-time operation. To break this compromise, we propose using Cellular Neural Networks (CNNs) [1, 2, 3], which are arrays of locally connected cells or neurons. CNNs have two good points for solving the vision-based navigation problem : first of all, they show a natural suitability for image processing and second, they have direct VLSI implementation, allowing for real-time parallel computation. Still, one needs to prove that CNNs have enough image processing capacity to enable robot navigation. Therefore, our goal is to start with a simple problem, yet with real life relevance: line following and landmark recognition in a maze [4, 5]. The

solution shows to be a first step towards tackling more complex problems such as curved line following or obstacle avoidance.

Line (or equivalent marking) following is a classical problem in robot navigation. Solutions found in the literature are either simple but with limited capability, either complex, facing realistic problems and using rather complicated hardware. An example of simple system is the ARGO partially autonomous vehicle [6]. It is a normal passenger car fitted with an automatic steering mechanism, controlled by a 486PC. The PC processes the images taken by two B/W cameras with the aim of finding the right line of the road, and keeping it in the appropriate position in the field of view. The processing is very simple, yet effective for the purpose. More complex systems are generally based on a computer which is either on board (for large vehicles, e.g. [7, 8]), or remotely connected (e.g. [9]). In these examples images are processed by an additional unit, due to the necessity of high computing power. Besides line following, these vehichles are capable of performing unstructured road navigation, target following and obstacle avoidance.

The paper is organised as follows. First of all, we make a fast review of Cellular Neural Networks and previous uses in robotics. Then, we present our solution for guidance in a maze: a fuzzy controller fed by the information coming from the CNN processing. We describe, then, the image processing in CNN terms and we give some possibilities for hardware implementation and some experimental results. Finally, we study how can we use the maze solution to deal with curved line tracking and obstacle avoidance.

# 2 Cellular Neural Networks

## 2.1 The CNN paradigm

Cellular Neural Networks (CNNs) [1, 2, 3] are arrays of dynamical artificial neurons (cells) with local connections only. This essential point has made hardware implementation of large networks possible on a single VLSI chip [10, 11].

Typically, cells are organised in a fairly large two-dimensional grid and an image pixel is assocated to one cell. In this way, CNNs can be used for massively parallel focal-plane processing, avoiding the bottleneck caused by electronic image loading. Moreover, distributed memory and a global control module can be present on chip. In this way the CNN becomes a stored-program massively parallel processor known as the *CNN Universal Machine* (CNN-UM) [12], which is capable of executing a complex sequence of operations (*analogic* programs) in real time. Therefore, CNNs are very good candidates as hardware platforms for robot vision.

In this paper, we shall refer to the Discrete-Time CNN model (DTCNN hereafter) [3]. However, all operations described in the following can also be achieved using continuous-time CNNs. The choice for DTCNNs relies on the limited availability of CNN chips. We are then forced to resort to simulated solutions, on DSPs or either on FPGAs and simulations are faster if DTCNNs

are used. The DTCNN core operation is described by the following system of iterative equations:

$$x_{ij}(n+1) = \left( \sum_{kl \in N(ij)} A_{k-i,l-j} \text{ sign } x_{ij}(n) + \sum_{kl \in N(ij)} B_{k-i,l-j} u_{ij}(n) + I \right) \quad (1)$$

where:

$x_{ij}$ is the state of the cell (neuron) in position $ij$, that corresponds to the image pixel in the same position;

$u_{ij}$ is the input to the same cell, representing the luminosity of the corresponding image pixel, suitably normalised;

$A$ is a matrix representing the interaction between cells, which is local (as specified by the fact that summations are taken over the set $N$ of indexes neighbour cells) and space-invariant (as implied by the fact that weights depend on the difference between cell indexes, rather than their absolute values);

$B$ is a matrix representing forward connections issuing from a neighbourhood of inputs.

$I$ is a bias.

$N(i,j)$ is the set of indexes corresponding to cell $ij$ itself and a small neighbourhood (e.g. cell $ij$ and its 8 nearest neighbours). Due to the locality of the computation, implied by the summation over this neighbourhood, and space-invariance, implied by the differences of indexes in matrices $A$ and $B$ in equation (1), it is sufficient to define $A$ and $B$ for a few instances of the indexes, so that they may be represented by small matrices.

The operation performed by the network is fully defined by the so-called *cloning template* $\{A, B, I\}$ (see examples in Table 2). Moreover, under suitable conditions, and with time-invariant input $u$, a steady state is reached. This steady state depends, in general, on initial state values and input $u$. Images to be processed are fed to the network as initial state and/or input, and the result taken as steady state value, which realistically means a state value after some time steps (ranging normally from 10 to 100 according to the task).

Equation (1) can be generalised by including nonlinear and delayed interactions [12]. While non-linearity in the feed-forward (connection to input) path can be exchanged for a suitable linear-connection-based algorithm [13], nonlinear feedback connections allow for real additional functionality.

Unlike most neural networks, very few parameters determine the CNN functionality. So, it is possible to explicitly design [14] a cloning template for a specific task (besides learning it, e.g. [15]), and to broadcast such parameters to all neurons. The cloning template can be understood as an *analogic instruction*

for the CNN. *Analogic* instructions can then be combined into *analogic* (analog/logic) programs, using the global controller and local memory. Sophisticated image processing can then be achieved in real time by the suitable sequence cloning templates. Many cloning templates and analog algorithms have been designed for the most diverse tasks [16].

## 2.2 CNN in Robotics : Previous Approaches

The previous applications of Cellular Neural Networks in robotics can be classified in three main groups: optical flow and time-to-contact estimation, depth estimation by stereo vision and simple line following.

Optical flow estimation is a very rich source of information for assessing the relative motion with respect to the surrounding environment or for tracking a moving object. Although usual algorithms are computationally costly, regularisation-based approach to optical flow can be cast into a CNN-like architecture to achieve local parallel computation [17]. Also, a few one-dimensional arrays are sufficient to make an estimation of the velocity field source (focus of expansion) and magnitude permits evaluation of direction and distance objects for obstacle avoidance purposes [18]. However, the CNN models used for this task are not standard and do not accomodate into general purpose CNN-UM chips.

Stereo vision is widely used for global estimation of depth information. is also a computationally-intensive task, very demanding for traditional architectures. Cellular Neural Networks can be applied to real-time solution of the stereo-matching problem [19]. Hardware implementation of such algorithms [20] is very efficient, but it can be necessary to resort to a multi-chip assembly deal with high-definition images [21].

A simple line-following task has been addressed in reference [22]. Even the task is not complex, simplicity and robustness of the solution is remarkable. Very limited CNN-based hardware resources are proved to perform efficiently in real-time.

# 3 Navigation in a maze

## 3.1 The Visual Control System

We have decided to start our robot vision CNN programme by dealing with a simple problem such as driving a robot in a maze made of black lines on light background. Despite its simplicity, this problem involves key features guidance such as navigation (line following) and landmark recognition (crossings and turns). Although there exist simpler solutions, we consider it to be significant benchmark to prove the efficiency of CNNs.

The mobile robot architecture considered is very close to a that of a standard car, with one driving and one steering wheel. In this case, the line following problem can be solved by adapting a well established fuzzy controller solving

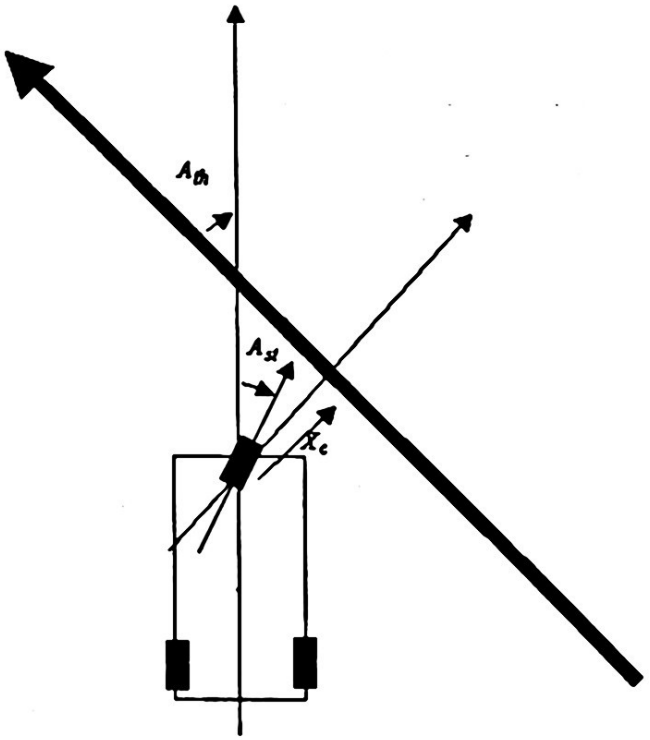



Figure 1: Line parameters. In the situation portrayed $A_{st}$ is positive, $A_{th}$ is positive and $X_c$ is negative (*i.e.* left)

the car backer-upper problem [23]. This fuzzy controller delivers the steering angle given the distance $X_c$ of the centre of the steering wheel from the line to be followed and the angle $A_{th}$ made by the axis of the robot and the line. This is portrayed in Figure 1, where $A_{st}$ is the steering angle.

The robot guidance problem is then split into two processes: a visual processing module and the fuzzy controller. The first is in charge of exctracting the relevant information from the image, namely the number of lines in sight, and its mathematical parameters, and take a decision on what line to follow. The parameters of this line (distance and angle) shall be delivered to the fuzzy controller. The fuzzy rule base is built out of driving experience inspired in reference [23]. The consequent sets are simplified to singletons: this scheme allows for a simpler on-line implementation because it requires fewer calculations. The values for the singletons are adjusted according to the robot platform steering capabilities. The controller, fed with distance $X_c$ and angle $A_{th}$ to the line uses 15 rules to deliver the steering angle $A_{st}$. They are listed in Table 1. Membership functions for $X_c$ and $A_{th}$ are shown in Figure 2. Consequents for $A_{st}$ are 0 for Zero and $\pm 20$, $\pm 40$, $\pm 80$ for Small, Medium, Large Left/Right respectively.

The global control strategy works as follows. With a single line in sight, the robot tries to align itself on it as fast as possible. When a second line gets into the view, the type of bend or crossing is assessed, by examining the

Table 1: Fuzzy control rule set for $A_{st}$.

| | | $A_{th}$ | | | | |
|---|---|---|---|---|---|---|
| | | Positive Large | Positive Small | Zero | Negative Small | Negative L |
| $X_c$ | Left | Medium Left | Small Right | Medium Right | Medium Right | Large Ri |
| | Centre | Medium Left | Small Left | Zero | Small Right | Medium R |
| | Right | Large Left | Medium Left | Medium Left | Small Left | Medium R |

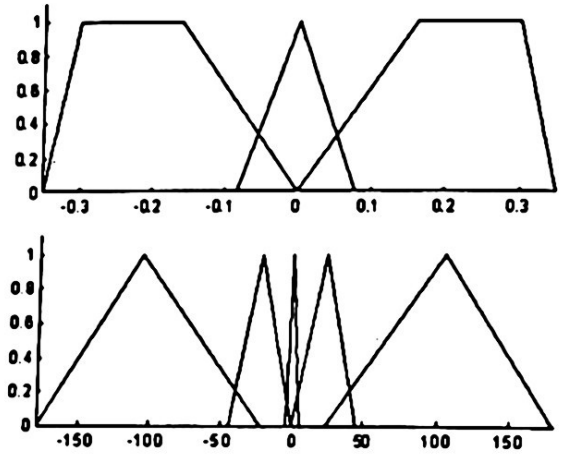



Figure 2: Membership functions for $X_c$ and $A_{th}$.

relative positions of the two lines. The possible actions are, then, evaluated. If more than one is possible (e.g. *go straight on* or *turn right*), then a command from a higher order level is required (in our experiments we just implemented a pre-defined list of actions). If a turn is called for, the robot continues on the current line until the crossing is at a pre-defined optimal distance (determined by the maximum steering angle). Then, the algorithm just switches the line to be followed. In this manner, we are capable of guiding the robot over the entire maze.

## 3.2 Image Processing

In the previous section we have defined the tasks of the image processing module, namely, landmark recognition (crosses and turns) and extraction of line parameters. Actually, landmark recognition just amounts to determine how many lines are in sight. From those, we need to compute $X_c$ and $A_{th}$. However, prior to line recognition and parameter extraction, the image needs to be acquired and preprocessed. Acquisition shall depend on the actual implementation and will be treated below, while preprocessing will consist in cleaning and enhancing the image contrast. Our goal is to perform all image processing operations using only sequences of CNN templates, in CNN terms, a CNN Universal Machine programme.

For preprocessing we use the so-called *small-object-killer* template (see Ta-

Table 2: Templates used in the processing stage.

| | $A$ | $B$ | $I$ |
|---|---|---|---|
| small object killer | $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ | 0 | 0 |
| left edge | 2 | $\begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ | -1 |
| vertically-tuned filter | 2 | $\begin{pmatrix} -1 & 0.5 & 1 & 0.5 & -1 \\ -1 & 1 & 1 & 1 & -1 \\ -1 & -1 & 5 & -1 & -1 \\ -1 & 1 & 1 & 1 & -1 \\ -1 & 0.5 & 1 & 0.5 & -1 \end{pmatrix}$ | -13 |
| connected-component detector | $\begin{pmatrix} 0 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & -1 & 0 \end{pmatrix}$ | 0 | 0 |

ble 2) to make a preliminary cleaning and binarisation of the acquired image (Figure 3 (a) and (b)).

The calculation of the parameters of lines visible in an image is universally performed by the Hough transform [24], but this is a computationally intensive technique that does not lend itself to efficient implementation on a CNN. We have then to devise a different approach.

In order to perform direction and position evaluation, first we need to get efficiently a thin line. For this purpose, we resort to design [14] a cloning template that extracts only one of the two edges of a stripe (it actually extracts only those parts of edges of black-and-white objects that lie on the left of the object itself). This template is given in Table 2, and its effect is shown in Figure 3 (c). To deal with approximately horizontal lines, we extract the upper edge by using a rotated version of this template.

The line position and orientation computation is based on two steps. We assume that a maximum of two, approximately orthogonal lines are within sight of the camera. This is not very restrictive, because we chose a setup in which the camera is oriented at an angle that was chosen as a compromise between looking a reasonably long distance forward, and avoiding a large deformation due to perspective.

The first step is a directional filtering that extracts lines approximately oriented along two orthogonal directions. During normal operation these directions are the vertical and horizontal ones, corresponding to the line being followed and a possible orthogonal line following a bend or crossing. However, when the robot is turning or is largely displaced from all lines, it is necessary to switch to diagonal directions. As this situation is known to the controller, it will signal to the image processing stage which filter should be used.

Operation of a vertically-tuned filter (Table 2) is depicted in Figures 4 and 5. Other direction-selective filters are obtained by rotation of this cloning template. After the tuned filters have been applied, we get two images containing at most one line. Direction and position of the line is then extracted by performing




Figure 3: Original image, taken from the actual camera with size $60 \times 45$ pixels (a), cleaning and binarisation (b), left edge (c), connected-component detector -intermediate result (d), final result (e).

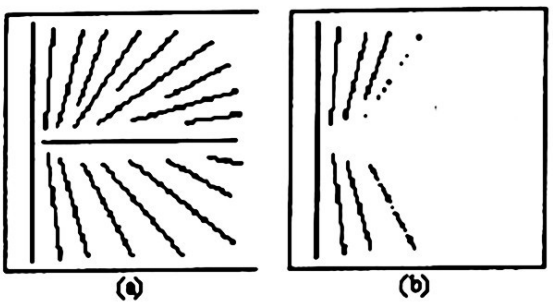



Figure 4: Vertical line extracting filter: original image (a), result (b).

a horizontal and vertical projection in order to read the positions of the first and last black pixels of the two projections. These four numbers, together with the information about which extreme of the line is closer to one of the borders of the image, are enough to compute direction and position of the line to the maximum precision allowed by image definition.

Extraction of the needed projections can be done by means of the so-called *connected-component-detector* cloning template (Table 2). Operation of such template at an intermediate and final stage of processing is depicted in Figure 3(d,e). It is apparent that besides obtaining the desired projections, also the information about which extreme is closer to the border can be obtained from examination of intermediate results.

Using the CNN operation described above, by simple trigonometry, it is easy to obtain the parameters that are necessary for navigation, namely, angle and distance. Such parameters are passed to the controller.

A strong-point of CNN processing is its capability of dealing with low quality or ill defined images, as shown in Figure 6.

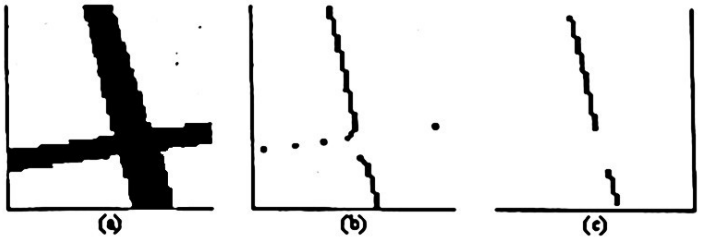



Figure 5: Vertical line extracting filter applied to the image of a crossing (a) and after edge extraction (b): result (c).

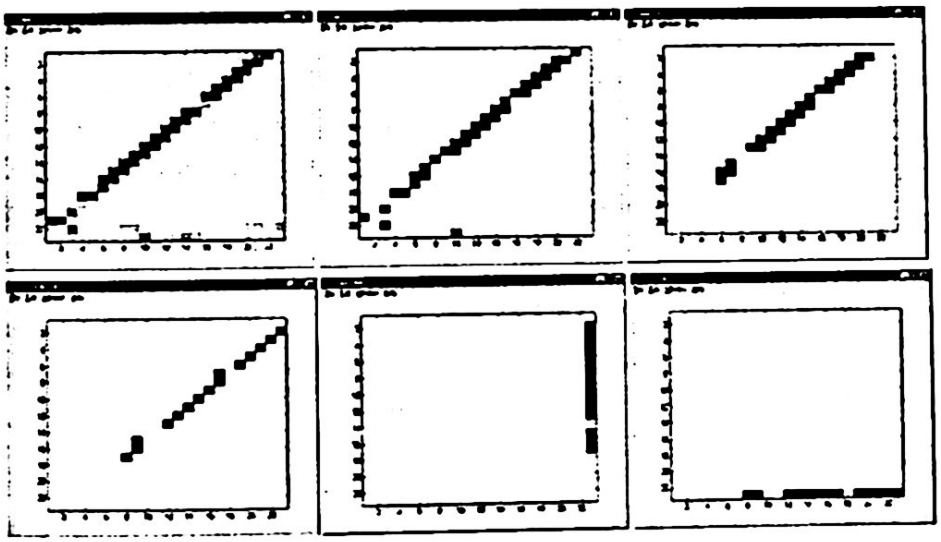


Figure 6: Successful processing of an image with low contrast.

### 3.3 Hardware Implementation

We have been considering two hardware implementations for a robot to be guided with such algorithms.

The first one is a small autonomous three-wheeled cart driven by a motor mounted on the single front wheel, that also steers by means of a second motor. The cart is approximately 27cm long, 18cm wide, 16cm tall, and weights about 4kg. A PAL camera is fitted on the front of the robot, oriented downwards, 30° from the horizontal. Images are grabbed and digitised by dedicated circuitry implemented in a CPLD (ALTERA EPM7128STC100), which also performs image decimation to reduce unnecessary definition. Image processing (CNN simulation) is performed in a DSP (TMS320c32), and the control system (fuzzy rule base) is implemented in a 386-microprocessor-based microcontroller. Image size is $60 \times 45$ pixels and lines on the floor are obtained using a 2cm wide black tape. A power board feeds the motors, and batteries are carried on board. In this way the robot it is completely autonomous. The image processing stage can currently process one image per second. We estimated by simulation that this allows the robot to move smoothly at a speed of 2.5 cm/s. Of course if CNN chip were used instead of the DSP, it would be possible to reach a much higher speed.

The second has similar mechanic characteristics but is controlled by an Xilinx FPGA development board. This solution is not fully autonomous since requires external power supply.

### 3.4 Experimental Results

The robot layout and the algorithms have been thoroughly tested by employing a realistic simulation of the vehicle realised in Working Model, interfaced with simulation of the fuzzy control and image processing system realised in Matlab. All mechanical and physical parameters of the actual robot were taken into account (going from size and weight to wheel and floor materials), as well as actual processing times of the mounted board, that has already been successfully tested. Mechanical mount and electrical testing of the robot is currently being completed, and we expect the robot to be fully functional soon. Meanwhile, Figure 7 shows a simulation of the path followed by the robot, starting a little displaced ($A_{th} = 10°, X_c = 4$cm) and turning at a crossing. For each position of the robot, a camera shot is assumed and the geometrical parameters of the robot position are calculated and passed to the controller.

## 4 Further Perspectives

The setup described in section 3 can be extended to other navigation problems provided we are able to define a virtual line to be followed. The parameters of this virtual line are then sent to the fuzzy controller. Bearing this in mind, we can consider more evolved problems such as curved line following or obstacle avoidance.

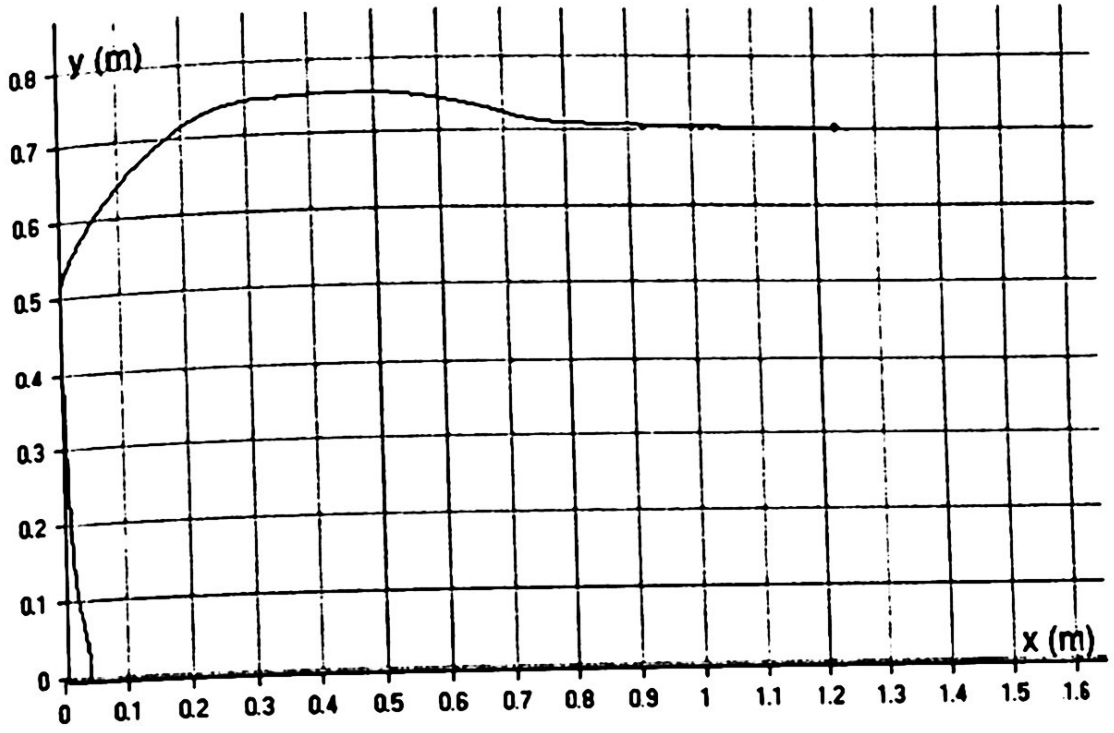



Figure 7: Path followed by the front wheel of the robot. Axis marking in meters.

## 4.1 Curved Line Following

Curved lines can be addressed by modifying the algorithms used for continuous straight lines. An accurate driving strategy has to take into account the following requirements: small curvature lines should be followed as close as possible, while for large curvature ones, the manoeuvre should be globally optimised, much in the same way as turning at a crossing. For this purpose, we estimate a *short-term* approximation to the curved line which is a the tangent line at the closest end (line *a* in Figure 8), and a *long-term* one which is the secant line joining farthest ends of curve in the image (line *b* in Figure 8). When the long-term approximation is similar to the short-term one, close following is possible using of the tangent line. When the angle difference between the two approximations is large, then substantial correction of the steering angle is necessary. This can be performed by adding, for instance, a supervisory fuzzy control algorithm using the information of both the tangent and the secant. Rules are of the form *if secant is more to the right than tangent, then correct steering angle towards right,* and *if secant is more to the left than tangent, then correct steering angle towards left.*

This correction can also be applied in two other cases: when coarse sampling in time is being used or when speed is relatively high so that the robot has moved a long path before the evaluation of the following image. The tangent line can then be estimated by the projection approach described in Section 3, but just projecting an area of few lines near the bottom of the image instead of the

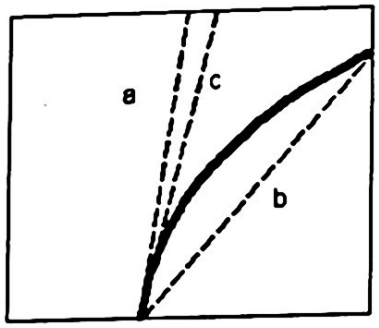



Figure 8: Tangent and secant to the curve.

whole image, so as to obtain a short-term secant, resembling the tangent (line *c* in Figure 8). This solution, however, is very sensitive to the image noise quantisation.

An alternative approach is based on computing one or more intermediate projections. It is equivalent to making a piece-wise-linear approximation of curve. The bottom projection delivers the short-term approximate line, while other projections can be used for correction.

## 4.2 Obstacle Avoidance

Obstacle avoidance involves two main tasks which are recognising obstacles such and, then, driving to avoid collision.

Obstacle recognition depends strongly on the characteristics of the environment. For our purposes, we consider an obstacle anything that develops in vertical dimension, departing from the floor level. This means that we shall consider colour, luminosity, texture or shape as indicators. These could just patterns on the floor and make no obstacle for the robot.

Solutions exist for binocular vision by stereo-matching-based range-finding (see for instance references [19, 20]). For high-definition or foveated imaging we can also use an optical-flow-based strategy (e.g. [17, 18]). Still we look solutions using our current hardware based only on monocular low-definition vision. We believe that working with such limitations will prompt us to look for solutions that optimise simplicity and cost.

A vision-only approach is based on checking the deformation of a suitable light pattern projected ahead. For instance, if we project a straight line light it will remain straight and in in a known position. When an obstacle (or level change, such as a step) is present, the line is deformed and displaced (Figure 9 (a)). It is easy to recognise such situation by CNNs, for instance, by projecting shadows of perceived objects on the expected line (Figure 9 (b)). Since obstacle recognition takes a relatively large amount of resources, it may be convenient to track objects once they are recognised rather than performing

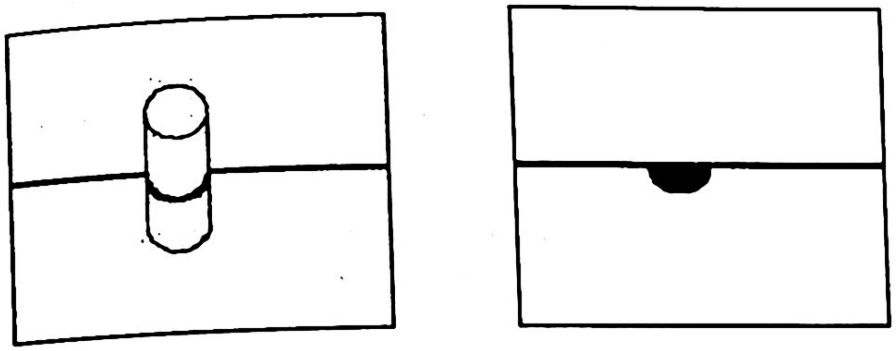



Figure 9: Line deformation in the presence of an object (a) and projection of the deformed line (b).

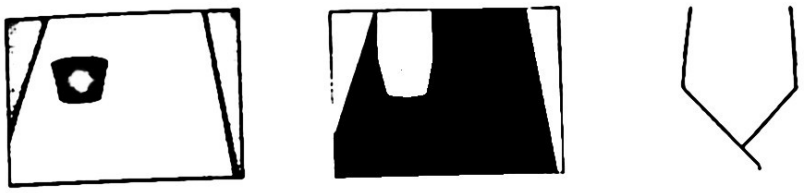



Figure 10: A corridor with an obstacle (a). A CNN template can fill the area available to move the robot (b). A skeletonisation of the available area (c).

recognition from scratch at every step. Efficient CNN-based tracking strategies can be based on active contours [25, 26].

An obstacle-avoiding driving strategy can be built on the line-following approach as follows. Suppose the robot is driving along a corridor with an obstacle in its way (Figure 10 (a)). Once the obstacle is recognised, we can use its contours and boundaries of the hallway as limits for the path. We apply then a CNN template to fill all available area by propagation (Figure 10 (b)). We can then take the skeleton of the shape obtained (Figure 10 (c)), or just draw a line pointing to the centre of the farthest and widest end of the shape obtained.

# 5 Summary and Conclusions

We have shown how the combination of Cellular Neural Network image processing and a fuzzy control can solve the robot vision problem in a simple case such as the motion in a maze. The image processing steps can be understood as an *analogic programme* in CNN-UM terms.

While the current implementation is based on software realisation of build-

ing blocks on programmable hardware, the target implementation is based special-purpose VLSI CNN hardware, yielding substantially enhanced performance.

The solution for this problem gives a hint for studying more complicated problems such as curved line following or obstacle avoidance. Still, this first step towards a wider use of CNNs in robotics for visual feedback. Further contributions should stress the use of CNN chips on a hardware implementation and the formulation of the image processing steps in CNN terms.

**Acknowledgements** This work is partially supported by Funitec (PGR2001-05). Authors would like to thank Jordi Solsona, Sonia Luengo, Alessandro Maraschini, Giada Apicella, Valentina Giovenale and Ruben Funosas for their collaboration in the project.

# References

[1] Chua, L.O., Yang, L. (1988) Cellular Neural Networks: Theory. IEEE Trans. Circ. Syst. **CAS-35** 1257-1272

[2] Chua, L.O., Roska, T. (1993) The CNN Paradigm. IEEE Trans. Circ. Syst. **CAS-I-40** 147-156

[3] Harrer, H., Nossek, J.A. (1992) Discrete-time Cellular Neural Networks. Int. of Circ. Th. Appl.**20** 453-467

[4] Vilasís-Cardona, X., Luengo, S., Solsona, J., Apicella, G., Maraschini, A., Balsi, M. (2002) Guiding a mobile robot with Cellular Neural Networks. *to appear Int. J. of Circ. Th. Appl.*

[5] Balsi, M., Vilasís-Cardona, X. (2002) Robot Vision using Cellular Neural Networks mobile robot vision. *Zhou, C., Maravall, D., Ruan, D. Eds. Autonomous Robotic Systems*, Physica-Verlag 2002.

[6] ARGO Project (2000), http://nannetta.ce.unipr.it/ĀRGO/english/index.html (as of December 2000)

[7] Chen, K.H., Tsai, W.H. (1997) Vision-Based Autonomous Land Vehicle Guidance in Outdoor Road Environments using Combined Line and Road Following Techniques. Journal of Robotic Systems **14** (10) 711-728

[8] Cheok, K.C., Smid, G.E., Kobayashi, K., Overholt, J.L., Lescoe, P. (1997) Fuzzy Logic Intelligent Control System Paradigm for an In-Line-of-Sight Leader-Following HMMWV. Journal of Robotic Systems 14(6) 407-420

[9] Maeda, M., Shimakawa, M., Murakami, S. (1995) Predictive Fuzzy Control an Autonomous Mobile Robot with Forecast Learning Function. Fuzzy Sets and Systems **72** 51-60

[10] Liñan, L., Espejo, S., Domínguez-Castro, R., Rodríguez-Vázquez, A. (2002) ACE4K: An Analog VO 64x64 Visual MicroProcessor Chip with 7-bit Analog Accuracy. Int. J. of Circ. Th. Appl., in press

[11] Kananen, A., Paasio, A., Laiho, M., Halonen, K. (2002) CNN Applications from the Hardware Point of View: Video Sequence Segmentation. Int. J. of Circ. Th. Appl., in press

[12] Roska, T., Chua, L.O. (1993) The CNN Universal Machine: an Analogic Array Computer . IEEE Trans. Circ. Syst. **CAS-I-40** 163-173

[13] Kék, L., Zarándy, A. (1998) Implementation of large-neighbourhood non-linear templates on the CNN universal machine. Int. J. of Circ. Th. Appl. **26** 551-566

[14] Zaràndy, À. (1999) The Art of CNN Template Design. Int. J. of Circ. Th. Appl. **27** 5-23

[15] Kozek, T., Roska, T., Chua, L.O. (1993) Genetic algorithm for CNN Template Learning. IEEE Trans. Circ. Syst., **CAS-I-40** 392-402

[16] Roska, T., Kék, L., Nemes, L., Zaràndy, À., Brendel M. (2000) CSL-CNN Software Library. Report of the Analogical and Neural Computing Laboratory, Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, Hungary

[17] Balsi, M. (1998), Focal-Plane Optical Flow Computation by Foveated CNNs. Proc. of Fifth IEEE Int. Workshop on Cellular Neural Networks and their Applications (CNNA-98), London, UK, Apr. 14-17, 1998, 149-154

[18] Shi, B.E. (1999) A one-dimensional CMOS focal plane array for Gabor-type image filtering. IEEE Trans. Circ. Syst. **CAS-I-46** 323-327

[19] Zanela, A., Taraglio, S. (1998) Seeing the Third Dimension in Stereo Vision Systems: a Cellular Neural Network Approach. Eng. Appl. Artif. Intel. **11** 203-213

[20] Sargeni, F., Bonaiuto, V. (2001) CNN Cell for Computing Disparity Map. Elec. Lett. **37** 682-683

[21] Titus, A.H., Drabik, T.J. (2000) Analog VLSI Implementation of the Help if Needed Stereopsis Algorithm. IEEE Trans. Circ. Syst. **CAS-II-47** 1328-1337

[22] Szolgay, P., Katona, A., Eröss, Gy., Kiss, Á. (1994) An Experimental System for Path Tracking of a Robot using a 16*16 Connected Component Detector CNN Chip with Direct Optical Input. Proc. of Third IEEE Int. Workshop on Cellular Neural Networks and their Applications (CNNA-94), Rome, Italy, Dec. 18-21, 1994, 261-266

[23] Kosko, B. (1992) Neural Fuzzy Systems, Prentice Hall Englewood Cliffs, NJ

[24] Hough, P.V.C. (1962) Method and means of recognizing complex patterns. U.S.Patent 3,069,654. *See also* Jain, A.K. (1989) Fundamentals of digital Image Processing, Prentice Hall Englewood Cliffs NJ.

[25] Vilariño, D.L., Brea, V.M., Cabello, D., Pardo, J.M. (1998) Discrete-time CNN for image segmentation by active contours. Patt. Rec. Lett, **19** 721-734

[26] Vilariño, D.L. (2001) Contornos activos a nivel de pixel: diseño e implementación sobre arquitecturas de redes no lineales celulares, Ph.D. thesis, Univ. of Santiago de Compostela, Spain



# Formación Integral de Especialistas en Reconocimiento de Patrones

José Ruiz Shulcloper
Grupo de Reconocimiento Lógico Combinatorio de Patrones, Instituto de Cibernética, Matemática y Física. Calle E#309 e/ 13 y 15 Vedado, CP 10400, Ciudad de la Habana, Cuba
recpat@icmf.inf.cu; shul59@hotmail.com; shulmx.@yahoo.com.mx

**Resumen**

A partir de una concepción integradora del Reconocimiento de Patrones (RP), se hace un análisis de las tendencias actuales que subyacen a muchas investigaciones y a los programas de formación de especialistas en esta área del conocimiento. Teniendo estos elementos como premisa, se propone un plan de acciones encaminadas a buscar un especialista en Reconocimiento de Patrones con una formación científica y técnica más completa, más adecuada para enfrentar los problemas actuales y sobre todo futuros, que debe encarar la disciplina.

**Palabras Claves:** Reconocimiento de Patrones, formación integral de especialistas.

## 1 Introducción

Es natural pensar que para lograr un mayor y más efectivo desarrollo de cualquier disciplina científica, tanto en sus fundamentos teóricos como en sus herramientas para la solución de problemas de la realidad y en la introducción eficiente de las mismas en la práctica social, es imprescindible contar con una *mayor cantidad* y de *mayor calidad* de lo más importante de todo este proceso: el especialista en esta zona del conocimiento.

Es de esperar que en cuanto a la cantidad todos convendremos en la necesidad de aumentar la formación de nuevos especialistas en todas las ramas de lo que genéricamente, siguiendo concepciones generalizadoras de las que hablaremos más adelante, aquí llamamos Reconocimiento de Patrones.

El aspecto de la *calidad* ya se hace más difuso. ¿Qué queremos decir con ese término? Aquí no nos estamos refiriendo a la calidad desde el punto de vista pedagógico, ni tan siquiera nos referimos al modo con que el

conocimiento ha sido asimilado por el estudiante, sino a *qué conocimiento* le estamos impartiendo a nuestros especialistas.

Las cuestiones que queremos abordar son entre otras las siguientes. Sabemos que no es posible enseñar lo que no se conoce y es natural pensar que educamos en consonancia a la forma en que entendemos las cosas, es decir, educamos en virtud de nuestras concepciones. ¿Son nuestras concepciones acerca de esta área del conocimiento las adecuadas? ¿Podemos hablar de RP como una disciplina científica? ¿Qué debería saber un especialista de esta zona del conocimiento a los diferentes niveles de educación: estudiante universitario, maestría, doctorado? ¿Estamos formando *adecuadamente* a nuestros especialistas, en el sentido de darles los conocimientos necesarios para resolver la compleja problemática que la disciplina tiene enfrentar en la actualidad y sobre todo en el futuro no muy lejano?

Es natural pensar que abordar el problema de la idoneidad o no de la formación de especialistas en nuestra área del conocimiento nos llevará de manera ineludible a sus fundamentos teóricos, a sus principios.



Desdichadamente, este tema no ha recibido una atención análoga a otros problemas del Reconocimiento de Patrones. No ha sido mucha la atención dispensada en congresos y talleres. De las organizaciones científicas internacionales relacionadas de alguna manera con esta zona del conocimiento, sólo IEEE Computer Society ha promovido Talleres sobre Educación [1-3], aunque limitado a Visión por Computadora y Procesamiento de Imágenes. En torno a estas dos zonas del RP se han desarrollado en la última década varios talleres y se han promovido interesantes discusiones acerca de los contenidos, medios computacionales para la enseñanza y textos, tanto de nivel de postgrado como para estudiantes universitarios. Estos últimos cursos son relativamente recientes y se concentran en las universidades de los países con mucho más desarrollo científico y técnico en estas materias.

En estos eventos el foco fundamental ha estado concentrado en cómo impartir mejor, con mejores recursos pedagógicos, los contenidos de esas materias, independientemente de las otras ramas vinculadas estrechamente con la temática. Es decir, se forman especialistas para la Visión por Computadora y/o para el Procesamiento de Imágenes. Un aspecto interesante que podemos extraer de los materiales de estos eventos es el hecho que, de los textos que se han elaborado, son excepciones aquellos que abordan la temática del RP en una óptica general. Trabajos como el Pattern Classification and Scene Analysis [4] de Duda y Hart (1973) o el Pattern Recognition Principles [5] de Tou y González (1974), son contados.

Es obvio que 30 años después un texto análogo a los dos mencionados tendría modificaciones sustanciales y además, la complejidad para su redacción no habría disminuido, por el contrario es muy superior.

En nuestra región, son pocas las universidades en pocos de nuestros países, que imparten algún curso para estudiantes universitarios y en todos los casos se trata de alguno de los aspectos de RP, casi siempre también sobre Procesamiento de Imágenes y/o Visión por Computadora.

Es una de las tareas actuales del Comité de Educación de la International Association on Pattern Recognition (IAPR) no sólo realizar la labor de selección de los mejores trabajos de estudiantes en los International Congress on Pattern Recognition (ICPR), sino de atender problemas como los que aquí se introducen.

Este trabajo explora algunos problemas relacionados con la formación de especialistas en RP que han sido abordados, en el mejor de los casos, de manera parcial por algunas instituciones relacionadas con esta zona del conocimiento. En la sección 2 se hace un análisis de las tendencias actuales que subyacen a muchas investigaciones. En la sección 3 se ofrece una concepción integradora de lo que entendemos por Reconocimiento de Patrones y sus implicaciones no sólo para el objetivo docente que nos proponemos sino también, para el desarrollo de las investigaciones tanto teóricas como aplicadas. En la sección 4 se propone un plan de acciones encaminadas a buscar un especialista en Reconocimiento de Patrones con una formación científica y técnica más completa, más adecuada para enfrentar los problemas futuros que debe encarar la disciplina.

## 2. Tendencia actual de las investigaciones en RP

Para tener una valoración en cuanto a la idoneidad de las concepciones y planes de formación de especialistas vigentes en esta zona del conocimiento, analicemos cuáles son las tendencias temáticas actuales de los grupos de investigación. Un análisis exhaustivo de este asunto es prácticamente imposible por lo que tomaremos como muestra de referencia a algunos de los Comités Técnicos (TC) que existen en la IAPR [6] en la actualidad y algunos de los grupos de RP en Ibero América y veamos someramente su contenido temático.

Aunque la IAPR no refleja todo lo que se está haciendo en el mundo dado que en la actualidad sólo 38 países están integrados a ella, no cabe duda que entre ellos se encuentran todos los países de mayor desarrollo en RP a escala mundial.

Entre los 18 TC existentes en este momento, como era de esperar hay comités dedicados al Reconocimiento Estadístico de Patrones, al Reconocimiento Sintáctico Estructural, Geociencias y Teledetección,

Visión por Computadora, Análisis de Señales, entre otros. Aparecen sin embargo, otros grupos que llaman la atención por poner en evidencias una tendencia a la interacción de áreas antes no relacionadas. Citemos algunos de los problemas que se pretenden resolver en el marco de esos grupos para ilustrar las ideas que planteamos:

*Graphics-based information retrieval*
*Use of linguistic information to improve shape classification*
*Geometrical and logical document structure*
*Data fusion problems*
*Hierarchical graphs for image segmentation and for control of perceptual strategies*
*Strategies for matching graphs having large set of nodes*
*Are graphs a useful representation for temporal data?*
*Video mining*
*Mining images, temporal-spatial data, images from remote sensing*
*Knowledge extraction from text, video, signals and images*
*Algebraic and discrete mathematical techniques in pattern recognition and image analysis*
*Discrete geometry*

En Ibero América, entre algunos de los muchos grupos de investigaciones podemos encontrar problemas tales como:

*técnicas de reconocimiento basadas en isomorfismos de grafos*
*interpretación inteligente de planos de arquitectura dibujados a mano*
*sistemas de interpretación de imágenes basado en conocimientos*
*máquina paralela para el procesamiento digital de señales*
*reconocimiento lógico combinatorio de patrones*
*morfología matemática*
*modelos difusos de reconocimiento de patrones*

Se van haciendo cada vez más familiar en la literatura de RP algunos términos como:

*fractales*
*modelos estructurales estocásticos*
*teoria de testores*
*conjuntos rugosos (rough sets)*
*evolutionary pattern recognition*
*hybrid architectures for machine vision and signal processing including evolutionary components*

Estos contenidos están apuntando hacia zonas del conocimiento tales como: teoría de hipergrafos, sistemas de recuperación de información, lingüística matemática y computacional,. Minería de datos, textos e imágenes, modelos algebraicos, geometría finita, paralelismo, computación

evolutiva y otras ramas que hace pocos años atrás poco o nada tenían que ver con el RP.

¿Qué es lo que se desprende de estos datos? Que a pesar de la existencia de relativos aislamientos conceptuales e insuficiente aún cooperación en las investigaciones tanto teóricas como prácticas, se aprecia un incremento en la integración de diferentes ramas del RP y de herramientas matemáticas y computacionales en las investigaciones que antes no habían sido empleadas. Es obvio que esta tendencia aumentará aún más en los próximos años.

Sin embargo, esto aún no se refleja en la docencia, en la formación de los especialistas de los tres niveles. No tenemos conocimiento que exista un plan de estudio en cualquiera de los tres niveles en el cual el estudiante adquiera una visión global del área actual de la disciplina y menos aún que conozca los rudimentos necesarios de las diferentes zonas de trabajo en las que actualmente se mueve el RP. Quizás lo más nocivo de esta situación es la no adecuada preparación para aprender, en relativamente poco tiempo, el manejo de herramientas que cada vez más muestran su eficacia en la solución de problemas teóricos y prácticos. El desconocimiento de herramientas y modelos puede lastrar la eficiencia de la solución de algunos problemas y en ciertos casos pudiera impedir alcanzar alguna solución.

Podemos afirmar además, que se aprecia un incremento, en los países más desarrollados en esta zona del conocimiento, de la enseñanza universitaria de algunas de las temáticas como Visión por Computadora y Procesamiento de Imágenes. Esta docencia en nuestros países es aún más escasa.

Como una muestra, a juicio del autor, de una insuficiente personalidad independiente, prácticamente toda la docencia postgraduada en el mundo está supeditada a un programa "sombrilla" que en la mayoría de los casos es un programa de Ciencias de la Computación o de Ingeniería Eléctrica, fundamentalmente. Existen algunas maestrías en Visión por Computadora que son independientes de otros programas, pero no es lo más frecuente.

Estos elementos nos deben llevar a la reflexión, al análisis de lo que deberíamos hacer ante un hecho que cada día se hace más evidente: las fronteras del RP se amplían, las interacciones entre sus ramas son más frecuentes y productivas, nuevas zonas de la matemática y las ciencias de la computación se integran al arsenal de herramientas del RP algunas de ellas sufriendo modificaciones que la hacen convertirse en herramientas propias de la disciplina.

Consideramos que se hace necesaria una conceptualización integradora del RP y en consonancia con ella un plan de *formación integral* de especialistas de RP a los tres niveles que contemple la tendencia actual de interacción y extensión de las investigaciones en RP. En este plan, los materiales que se elaboren para los estudiantes universitarios revisten una

extrema importancia y complejidad ya que ellos determinarán en gran medida ese enfoque integral al que hacemos referencia para esta área del saber.

## 3. ¿Qué es Reconocimiento de Patrones?

A pesar de que es una zona del conocimiento relativamente joven, apenas unos 45 años [7-9], su carácter interdisciplinario hace que este problema tenga una alta complejidad. Ya en 1973, Duda y Hart se referían a la dificultad de escribir un texto en el que se expusiera de manera sistemática las partes fundamentales del RP, que era entonces considerada por ellos como una rama de la ciencia relacionada con el reconocimiento por máquinas de leyes ante la presencia de ruido y en condiciones complejas. Según estos autores, el desarrollo de las técnicas computacionales había atraído hacia estos problemas a muchos especialistas de la estadística matemática, la teoría de las comunicaciones, la teoría de los sistemas de conmutación, la teoría del control, la investigación de operaciones, la biología, la psicología, la lingüística y las ciencias computacionales. Afirmaban desde esa época no sin pesar, la no-existencia de una teoría única del RP. Hoy todavía padecemos esa ausencia.

En 1998, en ocasión del III Taller Iberoamericano de Reconocimiento de Patrones [10], planteamos la necesidad de una mayor integración conceptual con el propósito de buscar una más eficiente colaboración científica y un más efectivo desarrollo de nuestros grupos de trabajo. Allí esbozamos algunas ideas en torno a lo que podría ser una aproximación a la concepción integral que requerimos ahora para nuestros fines docentes.
*Por RP podríamos identificar a la zona del conocimiento (de carácter interdisciplinario) que se ocupa del desarrollo de teorías, métodos, técnicas, y dispositivos computacionales para la realización de procesos ingenieriles, computacionales y/o matemáticos, relacionados con objetos físicos y/o abstractos, que tienen el propósito de extraer la información que le permita establecer propiedades y/o vínculos de o entre conjuntos de dichos objetos sobre la base de los cuales se realiza una tarea de identificación o clasificación.*

Ejemplos de *objetos físicos* lo son: fotos; hologramas; escrituras; jeroglíficos; símbolos; señales bioeléctricas; señales acústicas; etc.

Los *objetos abstractos* son n-uplos de un cierto producto cartesiano de conjuntos de cualquier naturaleza (duros, es decir, de la Teoría Clásica de Conjuntos; difusos; rugosos o de cualquier otra Teoría de Conjuntos que un futuro pueda crear el hombre). Es el caso, por ejemplo, de la descripción del cuadro sintomatológico de un paciente; de la descripción geólogo-geofísica

de una zona; de la enumeración de las características económicas y sociales de un conjunto de jóvenes formado por delincuentes y no delincuentes; de la descripción de las posibles anomalías que se pueden presentar en los servicios eléctricos de una compañía de luz; de las características socio-económicas de un solicitante de crédito bancario y muchas otras más que están presentes en disciplinas tales como la Medicina, las Geociencias, la Sociología, la Astronomía, la Criminalística, la Psicología, etc., disciplinas a las que se ha dado en llamar *poco formalizadas*. Esta denominación es un poco en la intención de contrastarlas con otras tales como la Matemática y la Física, en las que las leyes, el formalismo, están presentes en un grado apreciable. En las ciencias poco formalizadas es muy frecuente la presencia de la subjetividad proveniente de evaluaciones hecha por el hombre. Aquí no hay demostraciones al estilo de la Matemática ni tampoco resultados experimentales como podemos obtener en un laboratorio de Física.

En consonancia con esta conceptualización de RP presentamos un esquema, para nada exhaustivo, de las partes componentes del RP y su posición relativa respecto a las Ingenierías, la Matemática y la Computación. Todas apuntando de alguna manera hacia las aplicaciones. Porque, es menester decir que es esta una disciplina eminentemente dirigida a las aplicaciones. También es oportuno señalar que, como sentenció K. Levin "Rien n'est aussi practique qu'une bonne theorie".

## 3.1 Esquema constituyente de RP

*Matemática* *Ingeniería* *Computación* *Matemática*

Procesamiento de Imágenes

Procesamiento de Señales

Análisis e Interpretación de Imágenes

Análisis e Interpretación de Señales

Visión por Computadora

Percepción Remota

Redes Neuronales para RP

Algoritmos Genéticos en RP

Técnicas de Inteligencia Artificial para RP

Morfología Matemática

Reconocimiento Estadístico

Reconocimiento Sintáctico-Estructural

Reconocimiento Lógico Combinatorio

La posición relativa de las zonas del RP aquí mencionadas quiere expresar la medida de la presencia de la Ingeniería, la Computación y la Matemática en la generalidad de las investigaciones que en ellas se llevan a

cabo. Obviamente no debe entenderse estas ubicaciones al pie de la letra, más bien se trata de dar una idea de las posiciones relativas de dichas zonas.

## 3.2 Implicaciones del esquema constituyente

1. El Reconocimiento de Patrones es una zona del conocimiento interdisciplinaria, como se expresaba anteriormente, cuyas fuentes integrantes son las Ingenierías, la Computación y la Matemática.
2. Su estructura interna ha estado históricamente fraccionada en pequeños feudos que no nos conviene mantener. No se trata de que dejemos de hacer lo que cada cual ha querido venir haciendo hasta la fecha. Se trata de que nos reconozcamos mutuamente como miembros de una única familia, la de Reconocimiento de Patrones. Al igual que ocurre en la Matemática, la Física y muchas otras disciplinas con estructuras complejas como la del RP. Un especialista en Ecuaciones Diferenciales es matemático; un especialista en Lógica Matemática es matemático. Un especialista en Estado Sólido es físico; un especialista en Física Atómica es físico. ¿Por qué entonces no podemos entender que todo el esquema anterior (más quizás algunas omisiones involuntarias) sea RP?
3. El terreno de trabajo que tenemos en la disciplina de RP es inconmensurable. Tanto desde el punto de vista de las investigaciones fundamentales, fundamentales orientadas a las aplicaciones, como de las propias aplicaciones. Independientemente que existen zonas de investigaciones en RP donde la Ingeniería (genéricamente hablando), la Computación o la Matemática tienen un peso mayoritario, hay muchas zonas de investigación / aplicación que tienen un carácter híbrido. En ocasiones están presentes dos de ellas: Ingeniería – Computación, en ocasiones Computación – Matemática, etcétera, muchas veces todas.

Al margen de las posibles deficiencias que como *definición* propiamente dicha pueda tener la dada de RP en este trabajo, se ha formulado una conceptualización de nuestra disciplina tendente a promover la unidad académica entre zonas que históricamente se han reconocido de manera independiente. De hecho no es común referirnos a nuestra disciplina con un solo nombre. *Reconocimiento de Patrones* es la denominación que estamos proponiendo adoptar para nuestra disciplina entendida en los términos antes mencionados y en el espíritu del esquema constituyente del RP aquí también introducido. Obviamente, al igual que como en otras ciencias (Matemática, Física) cada una de sus zonas constituyentes tiene su propio desarrollo, no pierde con esta unidad propuesta su identidad propia.

Esta unidad académica puede tener repercusiones prácticas importantes: Se trata de promover una unidad académica para obtener una unidad

práctica, una unidad de acción. ¿Cómo pueden interactuar diferentes zonas del RP sin tan siquiera escucharnos, en ocasiones ni nos reconocemos? Claro esto último no es suficiente. Tiene que existir además una motivación, una necesidad, una voluntad de hacerlo. Las necesidades, querámoslo o no, existen.

¿Por qué no podemos echar andar un programa de Postgrado en el que se vean las tres partes constituyentes del RP? ¿Por qué no formar a nuestros jóvenes profesionales y científicos en la idea de una disciplina única, poderosa, interdisciplinaria, con todas las especializaciones que se requieran?

¿Por qué no podemos trabajar en la creación de una entidad, institución, centro, en el que se desarrolle no una, sino todas las partes constituyentes del RP? Un Centro que precisamente promueva no sólo esa unidad académica sino también esa unidad de acción.

Las líneas propuestas de trabajo conjunto pueden ser un interesante problema de desarrollo científico y de la consecución de aplicaciones de cierta relevancia social.

# 4. Proyecto de Programa de Postgrado en RP

Un programa de estudio que responda a las exigencias antes planteadas no existe. Para su elaboración debe tenerse en cuenta, entre otras cosas, los conocimientos **básicos** necesarios de matemática, ingeniería eléctrica, procesamiento de señales, óptica, física, teoría de la computación y algoritmos, entre otras materias, que serán imprescindibles para una formación sólida e integral de esta área del conocimiento. Es importante subrayar que un especialista en RP no será un matemático, ni un físico, ni un ingeniero eléctrico, ni un especialista en computación. Será simplemente un especialista en RP. Hasta el momento actual tenemos especialistas de esas áreas que se dedican al RP. ¿No sería factible formar especialistas en RP de manera directa? Esto obviamente apunta a la creación de una carrera universitaria en RP.

Este postgrado tendría las siguientes características distintivas:

1.- No es un apéndice de otro postgrado, es decir, no es un postgrado en Computación, o en Ingeniería, o en Matemática con especialidad en RP

2.- Tampoco sería un programa en Visión por Computadora, ni en Procesamiento de Imágenes, ni en ninguna de las que aquí consideramos ramas del RP

3.- Estaría basado en una visión integradora del RP como una disciplina interdisciplinaria que tiene una influencia de las Ingenierías, la Computación y la Matemática

4.- El conjunto de asignaturas básicas y optativas estaría concebido en una forma única, respondiendo a un mismo objetivo: formar al estudiante en una visión integradora del RP, trasmitirle la esencia de la disciplina y sus ramas fundamentales, darles los conocimientos y habilidades necesarias para posteriormente especializarse en un área en particular sin dejar de ver y usar la disciplina como un todo. Por tal motivo desde el mismo inicio todas las asignaturas harían referencia a su utilidad práctica en diferentes problemas del RP

5.- La colección de textos que se elaboraría al efecto respondería a un formato único en el que se hace énfasis tanto en la teoría como en las potencialidades aplicaciones del RP.



## 4.1 Programa de Estudio

Programa de Maestría a realizar en 6 cuatrimestres (2 años). Cursos de 60 horas lectivas de 45 minutos. Ese tiempo incluye conferencias, clases prácticas y seminarios si los tuviera. Los métodos de evaluación son variados, en dependencia de la asignatura y el profesor. Habrá un texto básico por asignatura que podrá ser complementado por cada profesor como estime oportuno. Carga semanal de docencia para el alumno de 16 horas lectivas.

Asignaturas Básicas

*Cuatrimestre Primero*

1. Principios del Reconocimiento de Patrones
2. Lógica Matemática y Matemática Discreta
3. Reconocimiento Estadístico de Patrones
4. Reconocimiento Sintáctico Estructural

*Cuatrimestre Segundo*

5. Reconocimiento Lógico Combinatorio de Patrones
6. Morfología Matemática
7. Redes Neuronales y Algoritmos Genéticos
8. Procesamiento Digital de Imágenes y Señales

*Cuatrimestre Tercero*

9. Análisis de Imágenes y Señales
10. Visión por Computadoras
11. Teledetección
12. Dispositivos RP

*Cuatrimestre Cuarto*

13. Cursos optativos

*Cuatrimestre Quinto y Sexto*

14. Seminario de Tesis

Asignaturas Optativas. Entre otras: Reconocimiento de Voz; Procesamiento y Análisis de Señales Biomédicas; Procesamiento y Análisis de Imágenes Biomédicas; Métodos de Segmentación; Algoritmos Conceptuales y Objetos Simbólicos; Teoría de Testores; Álgebra de Clasificadores; Computación Evolutiva; etcétera.

# 5. Conclusiones

Una de las razones por las que el autor ha puesto a su consideración esta problemática y ha brindado algunas ideas primarias para su solución, es la necesidad que se tiene de una mayor convergencia, una mayor integración entre las diferentes zonas del RP que se han ido estableciendo como zonas aparentemente independientes como el Procesamiento de Imágenes, la Visión por Computadora, los modelos matemáticos, las herramientas de la Inteligencia Artificial entre otras, bajo una concepción única. Esto no obedece a razones teóricas. Se trata de dotar de herramientas y métodos que den solución más efectiva y eficiente a los problemas que esta área del saber enfrenta. Muchas de esas herramientas y métodos no existen y deberán ser desarrolladas.

En este trabajo se han puesto de manifiesto dos problemas esenciales. El primero, la ausencia de una conceptualización, de unos fundamentos teóricos que sustenten, organicen, cohesionen el actual desarrollo de nuestra disciplina. Se hace necesaria la creación de las bases teóricas, de los principios del RP. El segundo problema es que, consecuentemente con la situación conceptual mencionada, la formación de nuestros especialistas es parcial y adolece de una visión global, de una forma integral de pensamiento, de conocimientos básicos necesarios para enfrentar los complejos problemas que en la actualidad se presentan ante nuestra disciplina.

Ante estos dos problemas se puede optar por uno de los siguientes dos caminos. El primero, trabajar en los principios del RP, esperar a tenerlos para entonces comenzar a formar a los nuevos especialistas a la luz de dichos postulados. El segundo camino es de una naturaleza más práctica e implica una aproximación sucesiva a la solución total de los problemas planteados. Se trata de ir modificando las curricula de estudio en la medida que vamos avanzando en los fundamentos teóricos. Esta variante es la que el autor propone y para ello sugiere un concepto de RP que en alguna medida constituye una paso en la senda necesaria a recorrer y consecuentemente con ello, una primera aproximación a un plan de estudio para la enseñanza postgraduada, que quizás sirva de motivación para un plan universitario de RP.

Finalmente, se sugiere lo siguiente:

1) El tema de los Principios del RP y consecuentemente los problemas de la formación de especialistas deben recibir una mayor atención por la importancia práctica que revisten
2) En nuestra región debemos promover en la enseñanza universitaria cursos básicos ***integrales*** de RP
3) Es conveniente lograr una mayor independencia de nuestros programas de postgrado respecto a otros programas "sombrillas" y lograr que estos sean programas integrales de RP
4) Es muy importante que sigamos promoviendo la interacción, la colaboración entre las diferentes zonas del RP y que nos unamos a escala regional con tales propósitos. Esta unión podría repercutir muy beneficiosamente en el desarrollo de todos.

# References


[1] B. A. Maxwell Comparative review of image processing and computer vision textbooks. *Proceedings of SPIE Medical Imaging '98*, February, 1998.

[2] B. A. Maxwell Teaching Computer Vision to Computer Scientists. *IEEE Computer Society Workshop on Undergraduate Education and Image Computation*, June, 1997.

[3] B. A. Maxwell. A survey of computer vision education and text resources. *http://www.palantir.swarthmore.edu/~maxwell:*

[4] R.O. Duda, P.E. Hart. *Pattern Recognition and Analysis Scene.* John Wiley & Sons. 1973.

[5] J.T. Tou, R.C. González. *Pattern Recognition Principles*. Addisson-Wesley. 1974.

[6] www.iapr.org.

[7] F. Rosenblatt. The Perceptron: A perceiving and recognizing automaton. Project PARA, Cornell Aeronautical Laboratory. Rept. 85-460-1. 1957.

[8] F. Rosenblatt. On the convergence of reinforcement procedures in simple Perceptrons. Cornell Aeronautical Laboratory. Rept. VG-1196-G-U. 1960.

[9] F. Rosenblatt. Principles of neurodynamics: Perceptrons and the theory of brain mechanisms. Spartan Books, Washington D.C. 1961.

[10] J. Ruiz-Shulcloper. Aplicaciones de Reconocimiento de Patrones en zonas del conocimiento poco formalizadas. Memorias del III Taller Iberoamericano de Reconocimiento de Patrones, TIARP'1998 México, D.F. pp. 91-101, 1998.

# Seismic Pattern Recognition Using Wavelet Transform and Seismic Attributes

Margarita Fernández[1], Adriana Mavilio[1], Juan Ramón Jimenez[3], Nieves Henriquez[3]
[1] Technical University of Havana, ISPJAE
Ciudad Habana (Cuba)
mflimia@electrica.ispjae.edu.cu,
adriana@electrica.ispjae.edu.cu
[2] PDVSA, INTEVEP,Caracas (Venezuela)
Jrjim@intevep.pdv.com, henriqueznp@pdvsa.com

**Abstract**

Seismic pattern recognition was carried out based on a comparison of seismic reflection traces of an interest region with those taken of a geologically well understood zone. The pattern was constructed from seismic traces corresponding to this reference zone. A segmentation task was performed, shifting a window through the 3D seismic data of the whole study region following seismic horizons.

We obtained similarity maps considering texture feature and seismic attributes respectively. On one hand texture features were computed based on wavelet decomposition coefficients of the seismic trace segments inside the window. In this case, the Mahalanobis distance to the pattern was used to determine the gray tonalities of the similarity map. On the other hand, a set of seismic attributes was extracted from seismic traces and compared with the pattern ones. From this comparison a parameter, seismic similarity to the pattern, was obtained.

A similarity map was also generated according to the values of this parameter. A satisfactory agreement between both similarity maps was obtained

**Key words:** seismic pattern, texture, wavelet.

## 1. Introduction.

The determination of areas of seismic similarity to those of good petrophysical properties can be a powerful tool to support exploratory work by highlighting areas with higher priority of prospecting. This can be done generating similarity maps based on the recognition of seismic reflection patterns [1, 2, 3], which are extracted around two good producing wells belonging to the region.

In this work, we generate similarity maps by using two different methods: wavelet approach [4] and seismic attribute technique [5].

Wavelet transform is one of the main techniques for analysis of signals and images. This description provides information about the signal contained in ever-

smaller regions of the frequency domain, and thus provides a very powerful tool for the discrimination of similar textures. In the last decade much research work on texture analysis has been done using this technique [ 6, 7].

In another paper we report the results of the use of these techniques and Gabor filters for texture segmentation of a 3D seismic section [4].

Correlating various attributes with reservoir properties generally performs seismic attribute analysis. This correlation cannot be extrapolated from one reservoir to another. This task becomes more difficult as the number of attributes involved becomes larger. We use a similarity analysis [5] to summarize seismic attribute information into just one parameter. This method eases the analysis task.

## 2. Seismic Data

The size of seismic data volume is (567 × 166 × 325) pixel$^3$. The z-axis corresponds to time (one division equivalent to 4 ms). The seismic reflection signals with 325 samples are seismic trace segments recorded with a bin size of 30×30 m.

The $x, y$ position of a good producing well was known. A window of (10 × 10 × 9) pixel$^3$ around the well (calibration point), corresponding to 100 seismic trace segments, was taken as reference area. The seismic trace segments inside this window, characterizing the internal stratification of this area, were considered to construct the pattern. The time length of this window plays an important role, since the good focus of the structural characteristics inside the seismic data volume depends on it. The time length of 9 elements (32 ms) was chosen taking into account the results of a correlation analysis [8], since for this time length the stratigraphic faults inside the seismic data volume were delineated more precisely.

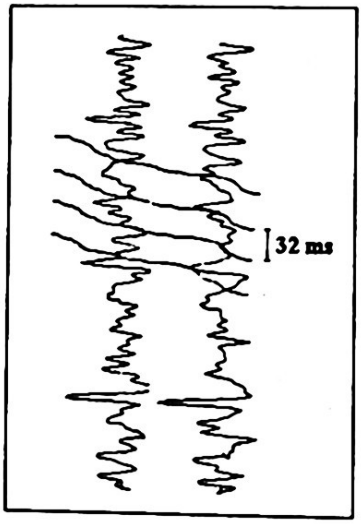



**Figure 1. Trace segments extracted from the seismic data volume. The orientation of the sketched line segments follows the seismic horizon. They delimit the segmentation zones.**

It was supposed that non-significant changes of the stratification take place inside the window.

The segmentation was carried out following a seismic horizon, at three fferent levels, as can be seen in Fig. 1.

The seismic data were supplied by the Venezuelan Enterprises PDVSA.

## Texture Features.

The Wavelet Transform uses a family of wavelet functions and its associated aling functions to develop the original signal in different frequency sub-bands. decomposition procedure is successively applied to the lowest frequency sub- to obtain the next level of resolution. The wavelet filters {g} (bandpass) and aling {h} (lowpass) are followed by a two to one sub-sampling in the signal.

The results are a detailed signal containing high frequencies and one pproximate that corresponds to the lowest frequency sub-band. The procedure is repeated on the approximated signal to obtain the next resolution level. Thus, wavelet transform is performed through a pyramidal algorithm [6].

The Daubechies wavelet bases are orthonormal, of compact support and have been broadly used in signal and image processing.

In this work, Daubechies-2 orthogonal basis was applied until a ecomposition level $J = 2$, taking into account the results of a numerical experiment erformed in a prior work with other seismic data [4].

The next steps were followed for pattern construction:

- Each trace segment (with 32 time elements, after a cubic spline interpolation) from the reference area is decomposed by means of the 1D wavelet transform, obtaining finally 3 sub-signals (frequency channels), one approximate and two detailed signals.
- The pattern feature vectors are determined. The three components of these vectors correspond to the three frequency channels. These components are calculated according to (1-4). A gaussian-weighted average of them is performed on the reference zone. Also, the feature covariance matrices $C_{cov}$ are calculated.

The features: energy ($E$), modulus ($md$), standard deviation ($sd$) and verage residual ($rms$) were calculated for each frequency channel through the llowing expressions:

$$E = 1/M \sum_{j=1}^{M} |c_j|^2 \qquad (1)$$

$$md = 1/M \sum_{j=1}^{M} |c_j| \qquad (2)$$

$$sd = \sqrt{\sum_{j=1}^{M} \frac{(c_j - vm)^2}{(M-1)}} \qquad (3)$$

$$rms = 1/M \sum_{j=1}^{M} |c_j - vm| \qquad (4)$$

where $c_j$ are the decomposition coefficients on a frequency channel, $M$ is the number of coefficients corresponding to this channel, and $vm$ is the mean value of the channel coefficients.

## 4. Seismic attributes

Seismic attributes, defined as "any or all the extracted observations of the seismic data, that can help direct or indirectly to the exploration of hydrocarbons" [9] have been used for the prediction, estimate or extrapolation of the stratigraphic characteristics of the location. In this work we used them in order to evaluate similarity to a pattern.

The mean values (gaussian-weighted average) and the standard deviations of each attribute were calculated in the reference zone. The following set of independent seismic attributes was used: top amplitude, composite envelope amplitude, average instantaneous frequency, amplitude wt.average, instantaneous frequency, average vibration energy, instantaneous real amplitude and instantaneous quadrature amplitude. They were computed by using appropriate software (IC2 de Scott & Pickford).

The pattern attribute vector was conformed. The components of this vector are the mean values of these attributes.

## 5. Segmentation

In order to discriminate zones of different similarity to the pattern inside the seismic data volume, we generate the similarity maps.

A segmentation window with size $(10 \times 10 \times 9)$ pixel$^3$ is shifted pixel to pixel following the seismic horizon for three different levels (see figure 1). The size of the window is a compromise between reliability of the estimates of the seismic textures and seismic attributes, and a wish of high resolution on the entire section.

In order to obtain the feature or the attribute vector of the window sample for each window location, we followed the same steps as in the case of the pattern construction. The sample vector was assigned to the window's central point.

The Mahalanobis distance:

$$D = \sum_{j=1}^{4} (\mathbf{d}_j - \mathbf{d}_{pj})^T \left( \frac{Cov_{pj} + Cov_j}{2} \right)^{-1} (\mathbf{d}_j - \mathbf{d}_{pj}) \quad (5)$$

between the reference feature vector and the sample feature vector were calculated for wavelet approach. The degree of similarity between the sample inside the window and the reference texture can be expressed in terms of the inverse of this distance. In the expression (5) $\mathbf{d}_j$ and $\mathbf{d}_{pj}$ are the 3-dimensional feature vectors of the window sample and the pattern respectively. Here the index $j$ specifies the type of feature. $Cov_{pj}$ and $Cov_j$ are the covariance matrices of the $j$ -type feature for the pattern and the sample respectively.

For attribute analysis, the similarity parameter was calculated in two steps: classification and sum. In the first step, to each component of the sample attribute vector is assigned one or zero, depending on whether its value falls within an interval centered in the corresponding component of the pattern attribute vector plus-minus the standard deviation of this component. The second step is simply to add up all the binary components. The result is the value of the similarity parameter for that window location.

Obviously, the similarity parameter ranges from zero to the number of attributes employed in the process. The areas with a greater quantity of attributes similar to those of the reference point possess the highest values of this parameter.

# 6. Results.

Fig. 2 shows one of the similarity maps obtained through wavelet approach and attribute analysis for one of the three studied levels and for one of the two reference zones. In the first case is showed the Mahalanobis distance and in the second the similarity parameter. Lightest gray tones correspond to more similarity to reference zone and darkest tones correspond to zones, which are very different from it.



(a)

(b)

Figure 2. Similarity maps of a reservoir at a given level for reference well (producer). (a) Applying Wavelet Transform, Daubechies-2 orthogonal basis. (b) Applying seismic attribute similarity analysis.

It is important to point out the good qualitative agreement between the two similarity maps generated by both approaches in spite that is impossible to quantify the similarity between both methods because they are based in different features and used different discrimination

functions. This agreement is present also in the three studied levels and for the two reference zones.

The black zone at the left on the images has not any meaning, since it corresponds to a lack of seismic information.

Black points centered in black circles on Fig.2 indicate the locations of two good producing wells of similar known rock properties.

The reference zone is located around the control point represented in the similarity maps by the black point on a lightest gray zone, near the center of the images demonstrating its great similarity with the pattern. It is expected since we know that both wells have similar rock properties.

## 7. Summary and Conclusions.

We have used two approaches, attribute technique and wavelet transform to perform pattern identification in a 3D seismic section, following a seismic horizon.

A satisfactory agreement was obtained among both methods and it was shown that both provide effective tools for seismic pattern recognition.

As a result of these segmentations the location of a stratigraphic fault was found. Experts in this matter have corroborated this result. Both methods performed the right identification of zones around the good production wells.

The used features and attributes showed high discrimination power for different seismic textures.

The generated similarity maps can be used to identify zones with similar seismic response to the reference zone, in this case a good producing well. A comparative analysis of these two maps provides a guide to highlight prospective areas with higher priority.

## |References.


[1] Vinther, R. Mosegaard, K., Kierkegaard, K. (1995). Seismic texture classification: To Computer-Aided Approach to Stratigraphic Analysis", *65th Annual Internat. Mtg., Soc. Expl.. Geophys., Expanded Abstracts*, CT1.5 pp. 153-155.

[2] Simaan, M. A.,(1991). Knowledge-Based Computer System for Segmentation Seismic Based on Texture, *Soc. Expl. Geophys*, CM1.2, pp.141-159.

.[3] Love, P. L. Simaan, M. (1984). "Segmentation of Stacked Seismic Dates by the Classification of Image Texture", *54th Annual Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts*, Session: S7.3, pp. 480-482

.[4] Fernández M, Mavilio A., Tejera M.(2000).Texture Segmentation of a 3D Seismic Section with Wavelet Transform and Gabor Filters, *Proceedings of 15th International Conference on Pattern Recognition, IEEE, Barcelona 2000, Image, Speech and Signal Proccessing*, Vol. 3, pp. 358-361.

[5] Michelena, R. González, E. M. Capello, (1998). Similarity analysis: A new tool to summarize seismic attributes information, *The Leading Edge*, Vol. 17, No. 4, April, pp545-548.

[6] Chang, T. Kuo, C.C.J (1992). A Wavelet Transform Approach to Texture Analysis. ", *Proc Int. Conf. Acoust. Speech, Signal Proc.*, Vol. 3, pp. 661-664.

[7] Chang, T. Kuo, C.C.J (1993) .Texture Analysis and Classification with Tree-Structured Wavelet Transform, *IEEE Transactions on Image Processing*, Vol. 2, No 4 pp.429-441

[8] Bahorich, M. Farmer, S. (1995). 3-D Seismic discontinuity for faults and,

stratigraphic features: The coherence cube, *The Leading Edge*, Vol. 14, No. 10, pp.1053-1058
Taner, M. Schuelke, J. O'Doherty, R Baysal, E. (1994). Seismic attributes revisted, *SEG Expanded Abstract.*





# Un algoritmo incremental para la obtención de particiones con datos mezclados

Aurora Pons-Porrata[1], José Ruiz-Shulcloper[2], Rafael Berlanga-Llavori[3], Yovanis Santiesteban Alganza[1]
[1] Universidad de Oriente, Santiago de Cuba (Cuba)
aurora@app.uo.edu.cu, yosva@csd.uo.edu.cu
[2] Instituto de Cibernética, Matemática y Física (Cuba)
recpat@cidet.icmf.inf.cu
[3]Universitat Jaume I, Castellón (España)
berlanga@lsi.uji.es

**Resumen**

En este trabajo se introduce un algoritmo incremental eficiente para estructurar un conjunto de datos en conjuntos compactos. Este algoritmo se apoya en algunas propiedades de los conjuntos compactos que son demostradas en el trabajo. El algoritmo propuesto crea una partición única del conjunto de datos, por lo que no depende del orden de presentación de los objetos. La complejidad computacional del algoritmo incremental presentado sigue siendo la misma que la de su variante no incremental y, por tanto, mucho más eficiente que la aplicación reiterada de la variante no incremental. El algoritmo propuesto puede ser utilizado en todas las tareas que requieran del agrupamiento de objetos en compactos y del procesamiento dinámico de la información, tales como, por ejemplo, la organización de la información, navegación, filtrado, ruteo y detección y seguimiento de sucesos en un flujo de documentos, entre otras aplicaciones. Es bueno resaltar que este algoritmo puede utilizarse, además, en problemas donde se deseen agrupar objetos de cualquier naturaleza, descritos por rasgos cuantitativos y cualitativos mezclados, incluso con ausencia de información.

**Palabras clave:** algoritmos incrementales, algoritmos de agrupamiento.

## 1. Introducción.

El agrupamiento de datos es uno de los problemas centrales en la Minería de Datos. Entre las técnicas de agrupamiento existentes, el criterio de agrupamiento en conjuntos compactos tiene la propiedad de que la semejanza entre un par de objetos de un mismo grupo es máxima. Los grupos obtenidos por este criterio son relativamente pequeños y densos.

Existen muchas aplicaciones donde este criterio resulta de utilidad: la determinación de zonas perspectivas de ciertos minerales, el análisis de datos en problemas de clasificación supervisada y otros. En estos problemas se procesan conjuntos de datos estáticos, es decir, que no varían.

Existen otros problemas en los que el conjunto de datos se incrementa en tiempo. Por ejemplo, el análisis del comportamiento de las facturas de una empresa, el estudio de las características sociales de los turistas que arriban a un hotel, identificación y seguimiento de sucesos en un flujo de noticias, etc. Para este tipo de aplicaciones se requiere de un algoritmo incremental que obtenga los conjuntos compactos.

En este trabajo se introduce un algoritmo incremental eficiente para estructurar un conjunto de datos en conjuntos compactos. Este algoritmo se apoya en algunas propiedades de los conjuntos compactos que son demostradas en el trabajo.

## 2. Conceptos y resultados necesarios.

Sea $\zeta$ una colección de objetos y S una función de semejanza entre objetos simétrica. Aquí sólo consideraremos este tipo de función de semejanza. Sea, además, $\beta_0$ un umbral de semejanza definido por el usuario.

**Definición 1.** *Se dice que dos objetos $O_i$ y $O_j$ son $\beta_0$-semejantes si $S(O_i, O_j) \geq \beta_0$ para todo $O_j$ de la colección de objetos $\zeta$ se cumple que $S(O_i, O_j) < \beta_0$ entonces $O_i$ se denomina $\beta_0$-aislado.*

**Definición 2.** *Diremos que $NU \subseteq \zeta$, $NU \neq \emptyset$ es una componente conexa si se cumple que* [1]:

1. $\forall O_i, O_j \in NU,\ O_i \neq O_j,\ \exists O_{i_1}, \ldots, O_{i_q} \in NU\ [O_i = O_{i_1} \wedge O_j = O_{i_q} \wedge \forall p = 1, \ldots, q\text{-}1\ S(O_p, O_{p+1}) \geq \beta_0]$.
2. $\forall O_i \in \zeta\ [O_j \in NU \wedge S(O_i, O_j) \geq \beta_0] \Rightarrow O_i \in NU$.
3. *Todo elemento $\beta_0$-aislado es una componente conexa (degenerada).*

**Definición 3.** *Diremos que $NU \subseteq \zeta$, $NU \neq \emptyset$, es un conjunto compacto si* [1]:

1. $\forall O_j \in \zeta\ [O_i \in NU \wedge \max_{\substack{O_t \in \zeta \\ O_t \neq O_i}} \{S(O_i, O_t)\} = S(O_i, O_j) \geq \beta_0] \Rightarrow O_j \in NU$.
2. $[\max_{\substack{O_i \in \zeta \\ O_i \neq O_p}} \{S(O_p, O_i)\} = S(O_p, O_t) \geq \beta_0 \wedge O_t \in NU] \Rightarrow O_p \in NU$.
3. $|NU|$ *es mínimo.*

*Todo elemento $\beta_0$-aislado constituye un conjunto compacto (degenerado).*

La condición 1 dice que todo objeto de *NU* tiene en *NU* a sus objetos más $\beta_0$-semejantes. La condición 2 dice que no existe fuera de *NU* un objeto cuyo objeto más $\beta_0$-semejante esté en *NU*. La condición 3 dice que *NU* es el conjunto más pequeño que cumple las condiciones 1 y 2.

El criterio de agrupamiento basado en los conjuntos compactos forma, al igual que el de las componentes conexas, una partición. Esta partición es, además, única para cada conjunto de datos dado.

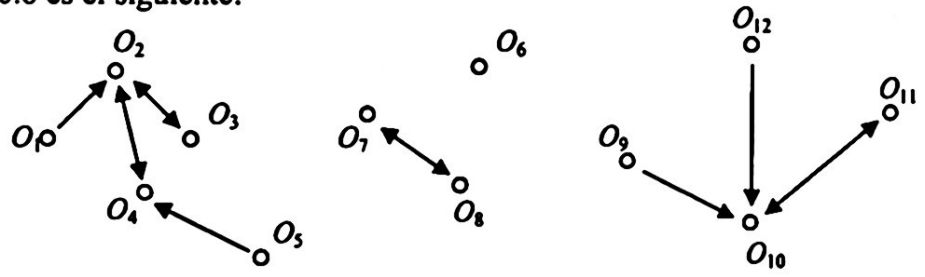
**Ejemplo 1.** Sean $\zeta=\{O_1,O_2,O_3,O_4,O_5,O_6,O_7,O_8,O_9,O_{10},O_{11},O_{12}\}$ y la matriz de semejanza siguiente, obtenida a partir de una función de semejanza $S$:

$$MS = \begin{array}{c} \\ O_1 \\ O_2 \\ O_3 \\ O_4 \\ O_5 \\ O_6 \\ O_7 \\ O_8 \\ O_9 \\ O_{10} \\ O_{11} \\ O_{12} \end{array} \begin{pmatrix} 1.0 & 0.85 & 0.75 & 0.78 & 0.63 & 0.70 & 0.51 & 0.38 & 0.43 & 0.27 & 0.22 & 0.13 \\ & 1.0 & 0.90 & 0.90 & 0.76 & 0.62 & 0.58 & 0.45 & 0.56 & 0.38 & 0.36 & 0.27 \\ & & 1.0 & 0.88 & 0.83 & 0.50 & 0.60 & 0.47 & 0.65 & 0.47 & 0.47 & 0.33 \\ & & & 1.0 & 0.85 & 0.58 & 0.68 & 0.55 & 0.63 & 0.45 & 0.42 & 0.30 \\ & & & & 1.0 & 0.48 & 0.73 & 0.65 & 0.80 & 0.65 & 0.58 & 0.50 \\ & & & & & 1.0 & 0.51 & 0.36 & 0.27 & 0.11 & 0.05 & 0.00 \\ & & & & & & 1.0 & .0.83 & 0.66 & 0.58 & 0.48 & 0.51 \\ & & & & & & & 1.0 & 0.66 & 0.63 & 0.53 & 0.61 \\ & & & & & & & & 1.0 & 0.83 & 0.80 & 0.70 \\ & & & & & & & & & 1.0 & 0.88 & 0.85 \\ & & & & & & & & & & 1.0 & 0.80 \\ & & & & & & & & & & & 1.0 \end{pmatrix}$$

Si consideramos $\beta_0 = 0.8$, entonces los conjuntos compactos son los siguientes: $NU_1=\{O_1,O_2,O_3,O_4,O_5\}$, $NU_2=\{O_9,O_{10},O_{11},O_{12}\}$, $NU_3=\{O_7,O_8\}$ y $NU_4=\{O_6\}$

**Definición 4.** *Sea $S$ una función de semejanza entre objetos. Llamaremos grafo basado en la máxima $\beta_0$-semejanza según $S$, y lo denotaremos máx-$S$, al grafo orientado $G=(\zeta,E)$ cuyos vértices son los objetos de la colección $\zeta$ y existe un arco del vértice $O_i$ al $O_j$ si se cumple que $O_j$ es el objeto más $\beta_0$-semejante a $O_i$.*

**Ejemplo 2.** El grafo *máx-S* correspondiente a la matriz de semejanza del ejemplo 1 con $\beta_0=0.8$ es el siguiente:




**Proposición 1.** *El conjunto de todos los conjuntos compactos de $\zeta$ coincide con el conjunto de todas las componentes conexas del grafo máx-$S$ asociado a $\zeta$ sin tener en cuenta la orientación.*

***Demostración.*** Es inmediato, a partir de la definición 4, que toda componente conexa del grafo *máx-S* asociado a $\zeta$ sin tener en cuenta la orientación, es un conjunto compacto de $\zeta$.

Sea $C=\{O_{i_1},O_{i_2},...,O_{i_k}\}$, $k>1$, un conjunto compacto y $G=(\zeta, E)$ el grafo *máx-S* asociado a $\zeta$ sin tener en cuenta la orientación.

Supongamos que $C$ no es una componente conexa de $G$. Por lo tanto, existen $O_{i_j},O_{i_r}\in C$, tales que entre ellos no existe un camino en $G$.

Entonces, construyamos el conjunto $NU=\{O_{i_j}\}$ y agreguemos a él todos los objetos $O_{i_t}$ de $C$ tales que:

$$\max_{\substack{O_{i_s}\in C \\ O_{i_s}\neq O_{i_j}}} \{S(O_{i_j},O_{i_s})\} = S(O_{i_j},O_{i_t}) \geq \beta_0$$

$$\max_{\substack{O_{i_s}\in C \\ O_{i_s}\neq O_{i_t}}} \{S(O_{i_s},O_{i_t})\} = S(O_{i_j},O_{i_t}) \geq \beta_0$$

Se repite el proceso para cada objeto de $C$ agregado a $NU$ hasta que no se pueda agregar a nadie más.

Por construcción de $NU$ el grafo *máx-S* asociado a $NU$ sin tener en cuenta la orientación es conexo, $O_{i_r}\notin NU$ por la suposición y, además, $NU$ es el conjunto más pequeño que satisface las condiciones 1 y 2 de la definición de conjunto compacto. Como $NU\subset C$, por la condición 3 de la definición de conjunto compacto, $C$ no puede ser un conjunto compacto.

Si $C$ fuera un conjunto unitario, es decir, $O_{i_j}$ es un objeto $\beta_0$-aislado, entonces una componente conexa de $G$ sería $\{O_{i_j}\}$, pues no existe ningún arco de $O_{i_j}$ cualquier otro vértice de dicho grafo. ∎

# 3. Nuevas propiedades de los compactos.

**Definición 5.** *Sean $C$ un conjunto compacto de una colección de objetos $\zeta$, $S$ una función de semejanza y $\beta_0$ un parámetro definido por el usuario. Decimos que un objeto $O$ está conectado con $C$ si existe al menos un $O'\in C$ que cumpla una de las dos condiciones siguientes:*

1. *$O'$ es el objeto más $\beta_0$-semejante a $O$ según $S$.*
2. *$O$ es el objeto más $\beta_0$-semejante a $O'$ según $S$.*

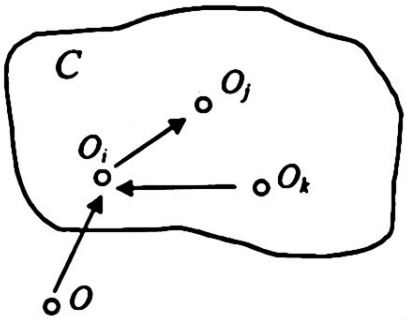
Un objeto $O$ puede conectarse con un conjunto compacto $C$ de varias formas. La figura 1 muestra todos los casos que pueden presentarse. En los casos I y II el objeto $O$ no rompe ningún arco del grafo *máx-S* asociado a $C$. En los casos III y IV sí rompen arcos.

Sean $\zeta$ una colección de objetos, $S$ una función de semejanza, $\wp$ el conjunto todos los conjuntos compactos de $\zeta$, $C\in\wp$ un conjunto compacto, $G_c=(C,U)$ el grafo *máx-S* asociado a $C$ y $NG_c$ el grafo $G_c$ sin tener en cuenta la orientación. Sea, además, $O$ un objeto tal que $O\notin\zeta$.

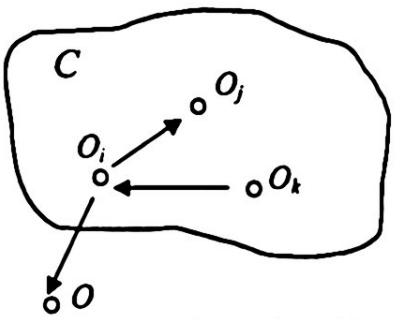
**Lema 1.** *Si O no está conectado con C, entonces C será un conjunto compacto de* $\zeta \cup \{O\}$.

***Demostración.*** Sea $G$ el grafo *máx-S* asociado a $\zeta$ sin tener en cuenta la orientación. Por la proposición 1, $NG_c$ es una componente conexa de $G$. Si $O$ no está conectado con $C$, por definición de componente conexa de un grafo, $O$ no puede pertenecer a dicha componente conexa. Por tanto, $NG_c$ será también una componente conexa del grafo *máx-S* asociado a $\zeta \cup \{O\}$ sin tener en cuenta la orientación y, en consecuencia, $C$ será un conjunto compacto de $\zeta \cup \{O\}$. ∎

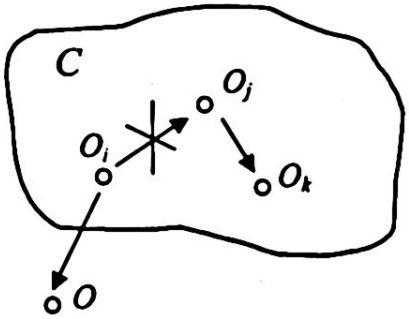
**Corolario.** *Si O es* $\beta_0$*-aislado, entonces el conjunto de todos los conjuntos compactos de* $\zeta \cup \{O\}$ *es* $\wp \cup \{\{O\}\}$.



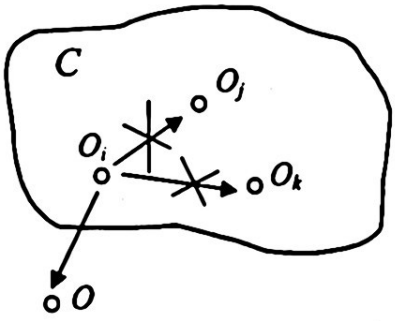

Caso I: $O_i$ es el objeto más $\beta_0$-semejante a $O$.




Caso II: $O$ y $O_j$ son los objetos más $\beta_0$-semejantes a $O_i$.




Caso III: $O$ es el objeto más $\beta_0$-semejante a $O_i$ y $O_j$ deja de serlo.




Caso IV: $O$ es el objeto más $\beta_0$-semejante a $O_i$ y $O_j$ y $O_k$ dejan de serlo.

Figura 1: Formas de conexión entre un objeto y un conjunto compacto.

**Lema 2.** *Si O está conectado con C y O no rompió ningún arco de* $G_c$*, entonces O y todos los objetos de C pertenecerán a un mismo conjunto compacto de* $\zeta \cup \{O\}$.

***Demostración.*** Por la proposición 1, $NG_c$ es una componente conexa del grafo *máx-S* asociado a $\zeta$ sin tener en cuenta la orientación. Si $O$ no rompió ningún arco de $G_c$, entonces $NG_c$ sigue siendo conexa en el grafo *máx-S* asociado a $\zeta \cup \{O\}$ sin tener en

cuenta la orientación. Si $O$ está, además, conectado con $C$, entonces $O$ pertenece a la misma componente conexa de los objetos de $C$ y, por tanto, $O$ y todos los objetos de $C$ pertenecerán a un mismo conjunto compacto de $\zeta \cup \{O\}$. ∎

**Definición 6.** *Un punto de articulación de un grafo no orientado es un vértice v tal que cuando removemos v y todos los arcos incidentes en él, la componente conexa de v se divide en dos o más partes* [2].

Sean $G'$ el grafo *máx-S* asociado a $\zeta \cup \{O\}$ y $H$, el subconjunto de objetos de $C$ que pertenecen a la componente conexa de $O$ en $G'$ sin tener en cuenta la orientación.

**Lema 3.** *Si O está conectado con C y O rompe uno y sólo un arco de $G_c$ (caso III), entonces si el conjunto $C \setminus H \neq \emptyset$, $C \setminus H$ es un conjunto compacto de $\zeta \cup \{O\}$.*

***Demostración.*** Sea $O_i \in C$ el objeto al cual $O$ es el más $\beta_0$-semejante y tal que se rompe su único arco $a$ que incide al exterior. Si $O_i$ no es un punto de articulación en $NG_c$, entonces al eliminar $a$ el grafo sigue siendo conexo y, por tanto, $H=C$.

Si $O_i$ es un punto de articulación en $NG_c$, entonces el grafo $G''=(C, U \setminus \{a\})$ sin tener en cuenta la orientación no es conexo. Si $C \setminus H \neq \emptyset$, entre cualesquiera dos objetos de $C \setminus H$ existe un camino en $NG_c$ porque, de lo contrario, $C$ no sería un conjunto compacto. Como $O_i$ es un punto de articulación en $NG_c$, no existe en $H$ ningún otro objeto conectado con $C \setminus H$.

Por tanto, $C \setminus H$ es una componente conexa en $G'$ sin tener en cuenta la orientación, es decir, $C \setminus H$ es un conjunto compacto en $\zeta \cup \{O\}$. ∎

**Teorema.** *El conjunto de todos los conjuntos compactos de $\zeta \cup \{O\}$ es:*

$$\wp' = [\wp \setminus (N \cup R \cup M)] \cup \left[\bigcup_{i=1}^{|R|} \{R_i \setminus H_i\}\right] \cup \left[\{O\} \cup \bigcup_{i=1}^{|N|} N_i \cup \bigcup_{i=1}^{|R|} H_i \cup \bigcup_{i=1}^{|M|} J_i\right] \cup K$$

*donde:*

*N es el conjunto de todos los conjuntos compactos de $\wp$ que están conectados con O y donde no se rompieron arcos y $N_i \in N$.*

*R es el conjunto de todos los conjuntos compactos de $\wp$ que están conectados con O y donde se rompió uno y solo un arco y $R_i \in R$.*

*M es el conjunto de todos los conjuntos compactos de $\wp$ que están conectados con O y donde se rompió más de un arco y $M_i \in M$.*

*$H_i$ es el conjunto de todos los objetos de $R_i$ que pertenecen a la misma componente conexa de O en G' sin tener en cuenta la orientación.*

*$J_i$ es el conjunto de todos los objetos de $M_i$ que pertenecen a la misma componente conexa de O en G' sin tener en cuenta la orientación.*

*K es el conjunto de todos los conjuntos compactos que pueden formarse en cada $M_i \setminus J_i$, $i=1,...,|M|$.*

***Demostración.*** Como dijimos anteriormente, un nuevo objeto $O \notin \zeta$ puede conectarse sólo de cuatro formas diferentes con los conjuntos $C \in \wp$. Es importante

observar que estos 4 casos no son excluyentes, incluso pudieran ocurrir todos a la vez varias veces cada uno. Por esta razón no es posible, ni necesario, analizar todas las posibles combinaciones. La ocurrencia de cada caso conllevará a una determinada modificación en el conjunto $\wp$.

La demostración de este teorema se apoya en los lemas 1, 2 y 3 y en la proposición 1.

A $\wp'$ pertenecerán por el lema 1, todos los conjuntos compactos de $[\wp \setminus (N \cup R \cup M)]$, pues con ellos el nuevo objeto $O$ no está conectado.

Además, por el lema 3, dado un conjunto compacto $R_i$, el conjunto $R_i \setminus H_i$ mantiene su propiedad de ser compacto si $H_i \neq \varnothing$ y, por tanto, los conjuntos de $\left[\bigcup_{i=1}^{|R|}\{R_i \setminus H_i\}\right]$ también pertenecen a $\wp'$.

La componente conexa de $G'$ a la que pertenece $O$ $\left(\left[\{O\} \cup \bigcup_{i=1}^{|N|} N_i \cup \bigcup_{i=1}^{|R|} H_i \cup \bigcup_{i=1}^{|M|} J_i\right]\right)$ es por la proposición 1 un conjunto compacto. Note que a ella pertenecen todos los conjuntos compactos de $N$ (según lema 2).

No es difícil ver que:

$$\zeta \setminus \left\{\left[\wp \setminus (N \cup R \cup M)\right] \cup \left[\bigcup_{i=1}^{|R|}\{R_i \setminus H_i\}\right] \cup \left[\{O\} \cup \bigcup_{i=1}^{|N|} N_i \cup \bigcup_{i=1}^{|R|} H_i \cup \bigcup_{i=1}^{|M|} J_i\right]\right\}$$

puede ser diferente del vacío. Esto es debido a la posibilidad de que $O$ provoque la ruptura de más de un arco en al menos un conjunto compacto $M_i$. Es inmediato que no podemos afirmar que $M_i \setminus J_i$ es un conjunto compacto. Luego, a partir del conjunto $\left[\bigcup_{i=1}^{|M|}\{M_i \setminus J_i\}\right]$ se hace necesario construir el conjunto $K$ de todos sus posibles conjuntos compactos. Observe que el cardinal de este conjunto es mucho más pequeño que el de $\zeta \cup \{O\}$. Con la inclusión de los conjuntos compactos existentes en $K$ se construye completamente $\wp'$. ∎

# 4. Algoritmo compacto incremental.

Paso 1. Cada vez que se presenta un nuevo objeto $O$ se calcula la similaridad con todos los objetos de los conjuntos compactos existentes.

Paso 2. Se seleccionan los objetos que son más $\beta_0$-semejantes a $O$ y aquellos a los que $O$ es el más $\beta_0$-semejante. Si no existen tales objetos, en virtud del corolario del lema 1, el conjunto $\{O\}$ es un nuevo conjunto compacto y terminar.

Paso 3. Los objetos seleccionados en el paso 2 pertenecen a conjuntos compactos de $N$, $R$ ó $M$, luego:

a) Se crea un nuevo conjunto compacto formado por $O$ y todos los objetos que pertenecen a la misma componente conexa de $O$ en el grafo *máx-S* de $\zeta \cup \{O\}$ sin tener en cuenta la orientación. Note que esta componente conexa está formada por todos los objetos de los conjuntos de $N$ y una parte de los objetos de cada uno de los conjuntos de $R$ y $M$.
b) Estos objetos son eliminados de los conjuntos compactos a los que pertenecían.
c) Los conjuntos compactos que quedan vacíos se eliminan.
d) Las partes que quedaron de los conjuntos de $R$ siguen siendo conjuntos compactos, en virtud del lema 3.

Paso 4. Puede suceder que al eliminar un objeto (que pertenece a la componente conexa de $O$) del conjunto compacto $M_i$ de $M$ al que pertenecía provoque que se quede inconexo y, por tanto, deje de ser un conjunto compacto. En cada parte de los conjuntos $M_i$ que quedaron hacer:
a) Seleccionar un objeto.
b) Construir su componente conexa (por la proposición 1 sus objetos forman un conjunto compacto).
c) Eliminar los objetos de esa componente conexa de la parte de $M_i$.
d) Mientras queden objetos ir al paso 4a).

# 5. Análisis de la complejidad computacional.

En este algoritmo cada objeto $O$ tiene asociado el grupo al que pertenece y tres informaciones: $A(O)$ que contiene el o los objetos más $\beta_0$-semejantes a $O$, es decir, $\{O' \in \zeta \,/\, S(O,O') = \max_{\substack{O'' \in \zeta \\ O'' \neq O}} \{S(O,O'')\} \wedge S(O,O') \geq \beta_0\}$, $SimilMax(O)$, el valor de esa máxima similitud y $De(O)$ que contiene los objetos a los que $O$ es el más $\beta_0$-semejante, es decir, $\{O' \in \zeta \,/\, O \in A(O')\}$.

La complejidad computacional de este algoritmo es $O(n^2)$, pues en el paso 1 cada objeto se compara con todos los existentes. El paso 3 conforma la componente conexa a la que el nuevo objeto pertenece. Este paso tiene complejidad $O(e)$, pues se trabaja con listas de adyacencia [3]. Aquí $e$ es la cantidad de aristas del grafo. Aunque desde el punto de vista teórico $e$ es del orden $n^2$, en la práctica los grafos de máxima semejanza no son grafos completos y lo que ocurre con mucha frecuencia es que la cantidad de objetos más semejantes a uno dado es 1. Por eso, en nuestro caso, hemos estimado experimentalmente que $e=cn$, donde $c$ es la cantidad máxima de objetos más semejantes a uno dado. En el paso 4 se reconstruyen las componentes conexas de los grupos que quedaron inconexos, lo cual, es $O(n+e)$.

Con relación a la complejidad espacial, nuevamente si tenemos en cuenta lo anterior, la complejidad sería $O(n)$, pues para cada objeto sólo tendríamos que almacenar el valor de su máxima semejanza y la lista de los objetos más $\beta_0$-semejantes, cuyo cardinal nuevamente podría aproximarse por una constante.

# 6. Experimentación.

La efectividad del algoritmo incremental propuesto ha sido evaluada usando dos conjuntos de datos.

## Conjunto de datos 1.

El primer conjunto de datos que usamos es una porción de *flag* tomado de [4]. Este conjunto está formado por 50 banderas de países descritas por 23 rasgos: 8 numéricos, 12 booleanas y 3 cualitativas *k*-valentes. Ejemplos de estos rasgos son: número de barras horizontales y verticales, colores presentes, si tienen o no triángulos, número de diagonales y estrellas, etc. Se utilizaron los siguientes criterios de comparación para cada tipo de variable:

$$C(x_i(O), x_i(O')) = \begin{cases} 1 & si\ x_i(O) = x_i(O') \\ 0 & en\ otro\ caso \end{cases}, \text{ para las variables booleanas}$$

$$C(x_i(O), x_i(O')) = \begin{cases} 1 & si\ x_i(O), x_i(O') \in A_p \\ 0 & en\ otro\ caso \end{cases}, \text{ para las variables } k\text{-valentes.}$$

$$C(x_i(O), x_i(O')) = \begin{cases} 1 & si\ |x_i(O), x_i(O')| \le \varepsilon \\ 0 & en\ otro\ caso \end{cases}, \text{ para las variables numéricas}$$

$A_p$ son subconjuntos de valores del rasgo $x_i$ y ε un umbral. Ambos parámetros son dados por el usuario.

Como función de semejanza utilizamos la suma ponderada de los criterios de comparación por rasgos y como umbral de semejanza =0.9.

Para el subconjunto de datos formado por 48 banderas se obtuvieron los siguientes 26 agrupamientos:

$G_1$={Bulgaria,Venezuela}, $G_2$={Gambia,Costa Rica}, $G_3$={Uruguay}, $G_4$={Fiji}, $G_5$={Bhutan},$G_6$={Zimbabwe},$G_7$={Uganda},$G_8$={Guinea_Bissau},$G_9$={Panama}, $G_{10}$={Somalia,Mauritania}, $G_{11}$={NorthKorea},$G_{12}$={Greece}, $G_{13}$={Japan,Brazil}, $G_{14}$={Ethiopia,Afghanistan,Colombia,Ecuador}, $G_{15}$={Cameroon,Mexico,Belgium, Mali,Chad,Andorra,Senegal},$G_{16}$={Niger,India},$G_{17}$={Haiti,Canada},$G_{18}$={Belize}, $G_{19}$={Cuba,Mozambique,Puerto_Rico},$G_{20}$={Argentina,Egypt,Austria},$G_{21}$={USA} $G_{22}$={Australia, Tuvalu, New_Zealand}, $G_{23}$={South_Africa}, $G_{24}$={Malta}, $G_{25}$={Bahamas}, $G_{26}$={Sweden,Dominican_Republic}.

Al incorporarse la bandera de Francia se mantuvieron los mismos 26 grupos, incorporándose Francia al grupo 15:

$G_{15}$={France, Cameroon, Mexico, Belgium, Mali, Chad, Andorra, Senegal}

Todas las banderas de este grupo se caracterizan por tener 3 franjas verticales de 3 colores diferentes donde uno de ellos es el rojo.

Al incorporarse, finalmente, la bandera de España se forman 26 grupos pero, en este caso, se particiona el grupo $G_{14}$ y se fusionan una parte de este grupo con el grupo $G_{20}$, quedando de esta forma:

$G_{14}$={Colombia, Ecuador},

$G_{20}$={Spain, Ethiopia, Afghanistan, Argentina, Egypt, Austria}

Atendiendo a los rasgos considerados para la descripción, la bandera de España es ahora la más parecida a la de Etiopía, por lo que las banderas de Colombia y Ecuador dejan de serlo. Por su parte la bandera más parecida a la de Afganistán es la de Etiopía, lo que justifica que éstas dos últimas se mantengan unidas en el nuevo agrupamiento $G_{20}$.

$S$(Etiopía,Colombia)=0.9828 $S$(Etiopía,Ecuador)=0.9828
$S$(Etiopía,Afghanistan)=0.9814 $S$(Etiopía, España)=0.9885

Esto provoca la partición del Grupo 14. Por otra parte, la bandera de Austria es la más parecida a la de España, $S$(Austria,España)=0.9985; lo que provoca la fusión de estos grupos al incorporarse en él la bandera de España.

Como ya hemos visto en el ejemplo, al incorporarse una nueva bandera al conjunto de datos, los grupos pueden cambiar en su composición y también pueden hacerlo en número.



## Conjunto de datos 2.

El segundo conjunto de datos con el que experimentamos es una colección de 526 artículos de periódicos publicados en la sección de noticias internacionales del periódico español "El País" durante el mes de junio de 1999. En este tipo de problema, es claro el carácter dinámico de la colección de datos (en este caso, de documentos) y, por tanto, imprescindible la utilización de un algoritmo incremental que identifique los sucesos que acontecen. Los documentos, denotados aquí por el título del artículo, fueron representados en función de la frecuencia relativa de sus términos y de sus ocurrencias temporales [5]. A continuación mostramos algunos de los grupos formados utilizando como umbral de semejanza 0.25:

Grupo #1:
Kofi Annan cree que es prematuro ponerse a dar saltos de alegría.
La secuencia de la pacificación.
La ONU ve imposible el regreso de los refugiados antes del otoño.
Aznar dice que espera ver un Kosovo liberado de la represión.
Clinton deja abiertas todas las opciones.
La OTAN consigue imponer sus principales objetivos.
El plan para la paz en Yugoslavia y el regreso de los refugiados.
Siete discrepancias iniciales.
Los refugiados carecen de papeles para regresar y recuperar sus casas.
Aznar dice que el próximo paso debe ser colaborar con el Tribunal Internacional.
Proyecto de resolución de la ONU.
Solana se niega a recibir al líder de la guerrilla kosovar en su visita privada a la OTAN.
Aznar pide ante Rugova que no se creen falsas ilusiones sobre la independencia de Kosovo.

Grupo #2:
Amnistía Internacional destaca el 'caso Pinochet' como hito de los derechos humanos.
Baltasar Garzón amplía en 34 casos la acusación por torturas contra el ex dictador Pinochet.
Documentos oficiales secretos chilenos pueban la existencia de la Operación Cóndor.
La desaparición de personas era una 'industria', según un informe.
'Embustes de los señores de España'.
Repercusión política del caso.
El Supremo dice que la actitud de Fungairiño en el 'caso Pinochet' puede ser 'criticable'.

El primer grupo contiene las noticias sobre los acontecimientos de la guerra de Kosovo y el segundo, las que abordan el caso Pinochet. Observe que, incluso al analizar el contenido de los artículos por su título, y teniendo en cuenta que el umbral seleccionado es bajo, los grupos formados presentan una alta coherencia temática.

# 7. Conclusiones.

En este trabajo se presenta un nuevo algoritmo incremental de agrupamiento que crea una partición en conjuntos compactos de la colección de objetos.

La variante no incremental de este criterio de agrupamiento necesitaba almacenar la matriz de similaridad (de orden $n^2$) de los objetos de la colección lo que, sin dudas, constituye una gran limitación cuando se desea trabajar con colecciones dinámicas o grandes de objetos. La variante incremental desarrollada no necesita almacenar esta matriz.

Otra ventaja del algoritmo propuesto es que permite encontrar grupos con formas arbitrarias en oposición a los algoritmos incrementales como *K-Means* [6] y *Single-Pass* [7] que utilizan medidas centrales para generar los grupos, y como consecuencia, se restringe a los grupos a tener formas esféricas o elipsoidales.

Además el agrupamiento obtenido por el algoritmo propuesto es único, es decir, no depende del orden de presentación de los objetos, a diferencia del *K-Means* y el *Single-Pass*, que pueden producir diferentes agrupamientos al cambiar el orden de entrada de los objetos. Esta unicidad hay que entenderla de la siguiente manera: dados $m$ objetos, la estructuración en conjuntos compactos es única, independientemente del orden en que los $m$ objetos sean considerados. Es claro que en una población dinámica, la llegada de un nuevo objeto puede producir nuevas estructuraciones, pero de poblaciones diferentes. Sin embargo, en el caso de la estructuración en compactos, una vez que ha sido considerada una población determinada, los objetos que la conforman pudieron haber arribado en cualquier orden produciendo la misma estructuración.

El algoritmo propuesto no necesita fijar a priori el número de grupos a obtener, es decir, es aplicable a problemas de clasificación no supervisada libre.

El criterio de agrupamiento basado en los conjuntos compactos forma grupos disjuntos y más cohesionados y pequeños que los formados por las componentes conexas basadas solamente en la $\beta_0$-semejanza. Este último criterio es el utilizado en el algoritmo incremental *GLC* [8].

La complejidad computacional del algoritmo incremental presentado sigue siendo la misma que la de su variante no incremental, lo que es muy superior a la aplicación reiterada del algoritmo no incremental.

El algoritmo propuesto puede ser utilizado en todas las tareas que requieran del agrupamiento de objetos en compactos y del procesamiento dinámico de la información, tales como la organización de la información, navegación, filtrado, ruteo y detección y seguimiento de sucesos en un flujo de documentos, el análisis del comportamiento de las facturas de una empresa, el estudio de las características sociales de los turistas que arriban a un hotel, entre otras aplicaciones.

Es bueno resaltar que este algoritmo puede utilizarse, además, en problemas donde se deseen agrupar objetos de cualquier naturaleza, descritos por rasgos cuantitativos y cualitativos mezclados, incluso con ausencia de información.

## Referencias


[1] Martínez-Trinidad, J. F., Ruiz-Shulcloper J., Lazo-Cortés, M. (2000). Structuralization of Universes, *Fuzzy Sets and Systems*, Vol. 112 (3), 485-500.

[2] Aho, A. V. , Hopcroft, J. E., Ullman, J. D. (1983). *Data Structures and Algorithms*, Addison-Wesley Publishing Company.

[3] Horowitz, E., Sahni, S. (1975). *Fundamentals of Data Structures*, Computer Science Press, Woodland Hills, California.

[4] ftp://ftp.ics.uci.edu/pub/machine-learning-databases/

[5] Pons-Porrata, A., Berlanga-Llavori, R., Ruiz-Shulcloper, J. (2002). Temporal-Semantic Clustering of Newspaper Articles for Event Detection, *Pattern Recognition in Information Systems*, Eds. José M. Iñesta y Luisa Micó, ICEIS Press, 104-113.

[6] Larsen, B., Aone, C. (1999). Fast and Effective Text Mining Using Linear-time Document Clustering, *in KDD'99*, San Diego, California, 16-22.

[7] Hill, D. R. (1968). A vector clustering technique, *in Samuelson (ed.). Mechanized Information Storage, Retrieval and Dissemination*, North-Holland, Amsterdam.

[8] Sánchez-Díaz, G. (2001). *Desarrollo de algoritmos para el agrupamiento grandes volúmenes de datos mezclados*, Tesis doctoral, Centro Investigación en Computación, México.

# Combining evolution techniques to estimate features' weights and the support sets system for ALVOT

Carrasco-Ochoa Jesús. A. & Martínez-Trinidad José Fco.
Departamento de Ciencias Computacionales
Instituto Nacional de Astrofísica, Optica y Electrónica.
Luis Enrique Erro No. 1 Sta. Ma. Tonanzintla, Puebla, México.
CP: 72840.
email: {ariel, fmartine}@inaoep.mx

**Abstract**

In this paper we propose an alternative method, based on evolution strategies and genetic algorithms, to estimate features' weights and the support sets system for the supervised classification model ALVOT (voting algorithms). Usually, this system is taken as the set of all typical testors, and features' weights are calculated using these testors. However, algorithms for calculating all typical testor have very high complexity, exponential in the worst case. Also, in some practical cases, the number of typical testors can be too large. This can become typical testors inapplicable for the classification stage. This problem, of high cardinality of the set of all typical testors, can cause that features' weights become too little for features which appear only in a few typical testors, including features which appear only in one typical testor. The proposed method allows generating features' weights, which maximizes ALVOT classification quality, with a support sets systems of limited size, but with a high efficiency for classification. Some performance examples, of the new method, are exposed. Quality for classification, of the results, is compared against typical testors option.

## 1 Introduction

ALVOT [1, 2] is a supervised classification model. It was developed in the framework of the logical combinatorial approach to pattern recognition [1]. This model is based on partial precedence. For ALVOT application it is necessary to define some parameters including features' weights and the support sets system, that indicates which parts of the objects are relevant to compare. It is very difficult to obtain values for features' weights from the specialist at the modeling process, so these values must be estimated from data. There are several

methods to estimate features' weights. In the framework of the logical combinatorial pattern recognition these weights are calculated from the set of all typical testors. Also, usually the set of all typical testors is taken as the support sets system. The use of all typical testors as support sets system gives good results at the classification stage, but there are two problems for using them: on the one hand, algorithms for calculating all typical testors have exponential complexity with regard to the number of features, so, for problems with many features the calculation of all typical testor is very expensive; on the other hand, the number of typical testors has an exponential bound, for this cause, typical testors can be too many to be useful at the classification stage (we have an example with 42 features where there are more than 300,000 typical testors). Then, use them to calculate features' weights becomes inadequate when they are too many, due to the number of typical testors is part of the denominator, in almost all expressions developed until now.

In this work we propose a new method based on evolution strategies, which allows estimating values for features' weights. This new method has as objective classification quality maximization. To evaluate classification efficiency it is necessary to have a support sets system, but it will be incongruent to use the set of all typical testors. Because of this, a method based on genetic algorithms is used to estimate the support sets system for each step. The last process uses as features' weights, the values that are been evaluated. Since the search strategy tries to maximize classification efficiency, the final values can have high classification efficiency, many times better than the efficiency of using the set of all typical testors. Additionally, the size of the support sets system is bounded, about its cardinality, by the size of the population maintained by the genetic algorithm.

In order to test the proposed method, we applied it over different data sets. Size and efficiency of the obtained features' weights and support set systems were compared against the typical testors option.

In the following sections the ALVOT model is described, concepts of testor and typical testor are given, testor theory approach to features' weights estimation is explained, a new method for estimating features' weights and the support sets system is exposed and experimentations are showed.

## 2 ALVOT

First, we define some concepts, which are useful to describe the voting algorithm model (ALVOT).

**Definition 1** *A support sets system denoted by $\{\Omega\}$, is a set of subsets of features. This system indicates which parts of the objects will be compared. Each $\Omega \in \{\Omega\}$ is called support set.*

**Definition 2** *Let $\Omega$ be a subset of the features. The $\Omega$-part of an object $O$ corresponding to $\Omega$, denoted as $\Omega O$, is the subdescription of $O$ using only features of $\Omega$.*

The voting algorithms model is based on the following basic ideas:

*Analogy.* There is a similarity function, which reflects how the analogy is made in the real problem.

*Partialprecedence.* The comparisons are not made between complete object descriptions but between subdescriptions previously selected.

*Frequency.* In voting algorithms an object belongs to a class if it is similar to enough objects of this class.

Each voting algorithm is determined by 6 parameters. The possibility of changing any of these parameters makes a family of this kind of algorithms.

The parameters that define a voting algorithm are:

$\{\Omega\}$ *SupportSetsSystem*

The support sets system determines which parts of the objects will be compared when the algorithm is applied. Any subset of the power set of the features can be used as support sets system, for example, all the subsets with a fixed cardinal, the set of all typical testors, etc.

$\beta$ *SimilarityFunction*

The similarity function determines how subdescriptions will be compared. This function should reflect how the analogy between objects is handled in the real problem.

$f$ *ObjectEvaluationFunctionforaFixSupportSet*

This function determines how much information is given by the similarity of a new object with each one of the sample objects, for a fix support set. The result of this function is called the vote given by each sample object to a new object with regard to a fix support set.

This function can take into consideration the weight of the evaluated sample object, and also, the weights of the features of the considered support set. Examples of this function are (1) and (2), where $\gamma(O_i)$ is the weight of the object $O_i$ and $P(\Omega) = \sum_{x_i \in \Omega} P(x_i)$ being $P(x_i)$ the weight of the feature $x_i$.

$$f(O_i,O,\Omega) = \beta(\Omega O_i, \Omega O) \qquad (1)$$

$$f(O_i,O,\Omega) = g(O_i)P(\Omega)\beta(\Omega O_i, \Omega O) \qquad (2)$$

$\varphi$ *ClassEvaluationFunctionforaFixSupportSet*

This function summarizes all object evaluations for a new object within each class, for a fix support set. The result of this function is called the vote given by each class to a new object with regard to a fix support set. Examples of this function are (3) and (4).

$$\varphi(j,O,\Omega) = \frac{1}{|K_j|} \sum_{O_i \in K_j} f(O_i,O,\Omega) \qquad (3)$$

$$\varphi(j,O,\Omega) = \begin{cases} 1 & if \ \frac{1}{|K_j|} \sum f(O_i,O,\Omega) \geq \lambda; \lambda > 0 \\ 0 & otherwise \end{cases} \qquad (4)$$

$\psi$ *ClassEvaluationFunctionfortheSupportSetsSystem*

This function summarizes all class evaluations for a new object for the whole support sets system. The result of this function is called the vote given by each

class to a new object with regard to the whole support sets system. Examples of this function are (5) and (6).

$$\psi(j,O) = \frac{1}{|\{\Omega\}|} \sum_{\Omega \in \{\Omega\}} \varphi(j,O,\Omega) \tag{5}$$

$$\psi(j,O) = \begin{cases} 1 & if \ \frac{1}{|\{\Omega\}|} \sum_{\Omega \in \{\Omega\}} \varphi(j,O,\Omega) \geq \lambda; \lambda > 0 \\ 0 & otherwise \end{cases} \tag{6}$$

*rSolutionRule*

This function takes all global evaluations for each class (given by y) and decides which class or classes the new object belongs to. The solution rule takes the form $r(\psi(1,O),...,\psi(l,O)) = (\alpha_1(O),...,\alpha_n(O))$, where $\alpha_i(O)$ is 1 if the new object is assigned to the $i$th class, and 0 in otherwise. Examples of this function are (7) and (8).

$$\alpha_i(O) = \begin{cases} 1 & if \ \psi(i,O) > \psi(j,O) \ \forall j \neq i \\ 0 & otherwise \end{cases} \tag{7}$$

$$\alpha_i(O) = \begin{cases} 1 & if \ \psi(i,O) - r_1 > \psi(j,O) \ \forall j \neq i \\ 0 & otherwise \end{cases} \tag{8}$$

With these parameters voting algorithms work in the following 5 stages:

1. Model parameters determination.
2. Apply Object evaluation function for the new object with each support set.
3. Apply Class evaluation function for the new object with each support set.
4. Apply Class evaluation function for the new object with the whole support sets system.
5. Apply solution rule to determine which class or classes the new object belongs to.

# 3 Typical Testors

Into the framework of the Logical Combinatorial Pattern Recognition, feature selection is done using typical testors[1, 3]. For the ALVOT model, typical testors can be used as support sets system. A testor is defined as follows:

**Definition 3** *A subset of features $T$ is a testor if and only if when all features are eliminated, except those from $T$, there is not any pair of equal subdescriptions in different classes.*

The definition 3 says us that a testor is a subset of features, which allows complete differentiation of objects from different classes. Within the set of all testors, there are some testors, which are irreducible. This kind of testors is called typical testors. Typical testors are defined as follows:

**Definition 4** *A subset of features $T$ is a typical testor if and only if $T$ is a testor and there is not any other testor $T'$ such that $T' \subset T$.*

The definition 4 says us that a typical testor is a testor where every feature essential, this is, if any of them is eliminated the resultant set is not a testor.

The calculation of all typical testors is important because they are all subsets features that can completely differentiate objects from different classes and they are not redundant. But, if the definition 4 is used to calculate all typical testors, all $2^n$ different subsets of features must be verified. For each one, you have to decide if it is a testor, then you must decide if it is typical.

Since this process has exponential complexity with regard to the number features, many different algorithms to calculate all typical testors have been developed, but all of them have exponential complexity for the worst case.

# Feature Weights Estimation

The approach based on Testor Theory was first proposed by Dimitriev et al. and the basic idea is the following: A testor is a feature subset, which does not confuse any pair of subdescriptions from different classes. Moving from a testor to a typical testor (eliminating features, when it is possible) we get an irreducible combination of features, where each feature is essential in order to keep differences between classes. This is the property that distinguishes each typical testor. It is natural to suppose that if a feature appears many times in the typical testors, it is more difficult to disregard it. That is, we could say it is more useful to differentiate between classes. Based on this idea Dimitriev et al. give a definition of feature's weight as the relative frequency of the occurrence each feature in the set of all typical testors. Let $\tau$ be the number of typical testors for a certain problem. Let $\tau(i)$ be the number of typical testors, which contain feature $x_i$. Feature's weight of $x_i$ is given by (9).

$$P(x_i) = \frac{\tau(i)}{\tau} \qquad (9)$$

where $i = 1, ..., n$, $x_i \in R$.

This expression is based on the following idea: a feature is more important when it appears more times into the set of all typical testors.

# Our proposal

this section, we present an evolution strategy to compute features' weights and a genetic algorithm to compute support sets systems associated to each combination of features' weights. First, we will explain the genetic algorithm, which is used as fitness function by the evolution strategy.

## 5.1 Genetic Algorithm

The individuals handled by the genetic algorithm [5, 6] will be the support sets. These sets are represented as $n$-uples formed by the values 0 and 1 (genes), these values represent the absence or presence of a feature, and $n$ represents the total number of features.

We will denote the individuals as $\Omega_i$ $i = 1, ..., p$, where $p$ is the number of individuals in the population.

The fitness function used by the algorithm is (10).

$$Fitness(\Omega_i) = \frac{Number\ of\ well\ classified\ objects}{Total\ number\ of\ classified\ objects} \qquad (10)$$

This function evaluates how many objects were well classified by ALVOT regarding the total of classified objects. The support sets system in this case is $\{\Omega_i\}$. If an individual (a support set) classifies more objects, then it will be more capable.

The crossover operator used in the algorithm is as follows: First, the population is ordered in descending form regarding individuals' fitness. The first individual (that has the highest fitness) and the last one (that has the smallest fitness) are taken and crossed using the one-point crossover using $n/2$ (the half of the individual) as crossing point. After, the second individual is crossed with the penultimate one; this process is repeated until finishing with the population.

In the genetic algorithm the generative mutation is used, this operator randomly takes an individual's gene and changes its value, if the gene has value 0, the new value will be 1 and vice versa.

In general way, the proposed algorithm woks as follows:

1. First, randomly the initial population is generated. The population size and iterations number are entrance parameters of the algorithm.

2. Then, the population's individuals are evaluated to determine their fitness, and then they are crossed using the crossover operator. The fitness is also evaluated for the new obtained population.

3. After, for each population's individual the mutation operator is applied. The fitness is also evaluated for the new obtained population.

4. Finally, the population's individuals together with the individuals obtained by crossing and mutation operators are ordered according to their fitness and the best are chosen (taking account the population size) to form a new population that will be used in the next iteration of the algorithm.

When the algorithm finishes then the efficiency of ALVOT is evaluated using both the computed support sets system (final population) and the features' weights W. This efficiency value will be used by the evolution strategy.

```
Genetic Algorithm (SSS-GA)
Input
 W: features' weights.
 size: size of population
 Num_iter: Number of iterations
1.Generate_initial_population(population,size)
2.Evaluate_population(population)
3.Repeat for i=1 until i=Num_iter
4.Crossover(population,population2)
5.Evaluate_population(population2)
6.Mutation(population,population3)
7.Evaluate_population(population3)
8.Pick_out_next_population(population,population2,population3)
9.Write(population)
10.Return ALVOT(population,W) END.
```

In the algorithm, population2 has a new population after applying crossover operator to individuals of the original population; population3 has new individuals after applying the mutation operator to individuals of the original population.

The function Pick_out_next_population chooses from population, population2, and population3 to the most capable individuals (those with highest fitness) that will form a new population for the next iteration.

## 5.2 Evolution Strategy

The individuals handled by the evolution strategy [7] will be the features' weights. These weights are represented as n-uples formed by values in the interval [0,1], these values represent the weight of each feature, and n represents total number of features.

We will denote the individuals as $W_i$ $i = 1, ...., p$, where $p$ is the number of individuals in the population.

The fitness function used by the strategy was the genetic algorithm above described, where population size and number of iterations were 15 and 10 respectively. That is to say, $Fitness(W_i) = SSS\text{-}GA(W_i)$. This function evaluates how many objects were well classified by ALVOT regarding the total of classified objects. In this case, the genetic algorithm ($SSS\text{-}GA$) is called passing as parameter the individual $W_i$. In this way, a support sets system is computed for and the efficiency of ALVOT using $W_i$ and $\{\Omega\}_i$ as parameters is evaluated. an individual $W_i$ (using their $\{\Omega\}_i$) correctly classifies more objects, then it will be more capable.

The crossover operator used in the strategy is one-point crossover. But in this case the individuals are n-uples of real numbers. The uniform mutation is used; this operator randomly takes an individual's gene and changes its value a random number in the interval [0,1].

```
Evolution Strategy (ES)
Input:
 size:  size of population
 Num_iter:  number of iterations
1.Generate_initial_population(population,size)
2.Evaluate_population(population)
3.Repeat for i=1 until i=Num_iter
4.Crossover(population,population2)
5.Evaluate_population(population2)
6.Mutation(population,population3)
7.Evaluate_population(population3)
8.Pick_out_next_population(population,population2,population3)
9.Write(population) END.
```

The evolution strategy has the same structure than the genetic algorithm except that the function Evaluate_population calls to *SSS-GA* for each individual $W_i$ in order to compute a support sets system $\{\Omega\}_i$. In this way, the fitness of $W_i$ is computed as the efficiency of ALVOT using $W_i$ and $\{\Omega\}_i$ as parameters. This is shown in the figure 1. At end of the process we can select the individual with highest fitness (and its support sets system) as the features' weights and the support sets system for ALVOT in the classification stage.

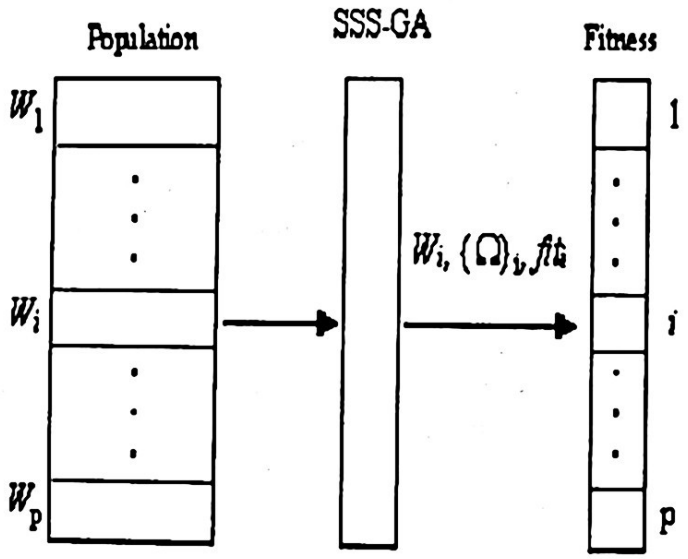



Figure 1: Diagram of the method proposed to estimate features weights and support sets system.

# 6 Experimental results

In this section, some experiments to estimate the features' weights using an evolution strategy and the estimation of the support sets system with the genetic algorithm are presented. In all the experimentations, a comparison of the classification efficiency using the features' weights and the support sets system obtained with the proposed method, against the typical testor option, is made.

The databases for experimentations were taken from [8].

The first experiment was carried out with the zoo database. This database contains 101 descriptions of animals grouped in 7 classes, each description is given in terms of 16 features.

In this case there are 33 typical testors. The efficiency of classification of ALVOT using these 33 testors like support sets system is of 1.0. The measure of efficiency was obtained classifying the training sample and taking the number of successes divided by the number of objects in the sample.

In the figure 2, the classification efficiency of ALVOT when $W_f$, the most capable individual in the last iteration of the evolution strategy, and $\{\Omega\}_f$ (the support set system associated to $W_f$) were used as parameters is shown. In this case, the horizontal axis represents the variation of population size in the genetic algorithm to compute the support sets system. The vertical axis is the reached efficiency using the typical testors and the features' weights computed on basis of them. For the evolution strategy we handle a fixed population size.

In the figure 2a it is seen that, for zoo database the efficiency cannot be improved but with a population of 10 support sets, three times less than the total number of typical testors, the same efficiency can be achieved as with the 33 typical testors.

The second experiment was carried out with the import85 database. This database has 205 car specifications and they are distributed in 7 classes, each specification is made using 25 features. The number of computed typical testors is 6. The classification efficiency of ALVOT using the 6 testores is 0.7512. In the figure 2b, the obtained results are shown.

The third experimentation was made with the crx database. This database has 690 credit applications, there are 2 classes, and each application is given in terms of 15 features. This example only has one typical testor. The classification efficiency of ALVOT taking account the typical testor is 0.9623. In the figure 2c, the results obtained are shown.

The last experimentation was made using the hepatitis database. This database has 155 patients with hepatitis, there are 2 classes; each patient description has 19 features. Here, the number of typical testors is 404. The classification efficiency of ALVOT using the typical testors is 0.9870. In the figure 2d it is seen the classification efficiency of ALVOT using the features' weights and the support sets system computed with our method.

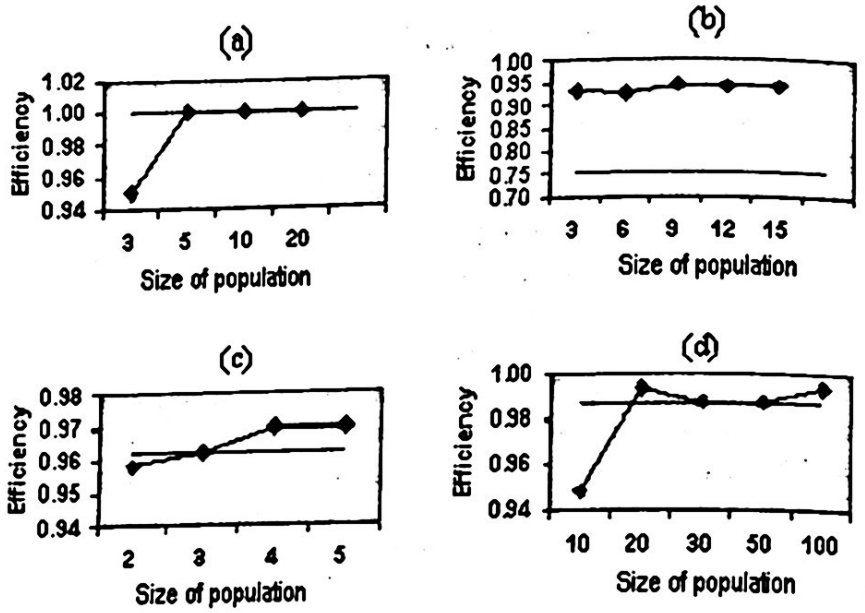



Figure 2: Efficiency of ALVOT for a)zoo, b)import85, c)crx, and d)hepatitis, using the features' weights and the support sets systems computed by our method.

# 7 Conclusions

In this paper, a new method, based on evolution strategy and genetic algorithms, to estimate the features' weights and the support sets system for the model of supervised classification ALVOT, is presented.

From experimentation, we can conclude that a better classification efficiency than using typical testors and features' weights based on them may be reached; but applying our method we can get a support sets system with lesser cardinality. It results in a lesser time for classification stage.

Additionally, since the problem of computing all the typical testors has exponential complexity and the proposed method can be proved that has polinomial complexity. For problems with many features may be impossible in the practice to calculate the typical testors, however the proposed method can be applied.

Another point to highlight is the fact that the number of typical testors is bounded exponentially, so that, in some cases they can be too many to be useful in the classification stage. Contrarily, in the proposed method the size of the support sets system is one of the parameters, which allows fixing the size of the system according to the practical requirements.

The main application of the here presented method is for big problems where using typical testors is not feasible. Since they cannot be computed or they turn out to be too many.

**Acknowledgement.-** This work was financially supported by CONACyT (Mexico) through projects J38707-A and I38436-A.

# References

Martínez-Trinidad J. F.& Guzmán-Arenas A. (2001). The logical combinatorial approach to pattern recognition an overview through selected works, *Pattern Recognition* 34(4) 741-751.

Ruiz-Shulcloper J. & Lazo-Cortés M. (1999). Mathematical Algorithms for the Supervised Classification Based on Fuzzy Partial Precedence, *Mathematical and Computer Modelling*, 29(4), 111-119.

Lazo-Cortes M., Ruiz-Shulcloper J. & Alba-Cabrera E. (2001). An overview of the evolution of the concept of testor, *Pattern Recognition*, 34(4), 753-762.

Dmitriev A.N., Zhuravlev Y.I., & Krendeliev F.P. (1966). About mathematical principles of objects and phenomena classification, *Diskretni Analiz* 7, 3-15. (In Russian).

Mitchel M. (1996). *An introduction to genetic algorithms*, MIT Press.

Goldberg D. (1989). *Genetic algorithms in search, optimization and machine learning*, Addison Wesley.

Beyer H. G. (2001). *Theory of Evolution Strategies*, Springer Verlag.

http://www-old.ics.uci.edu/pub/machine-learning-databases/

# LCARS un Lenguaje de Especificación de Problemas en Reconocimiento de Patrones bajo el enfoque Lógico-Combinatorio

Salvador Godoy-Calderón[a], José Fco. Martínez-Trinidad[b] y Manuel Lazo Cortés[c]

[a]Centro de Investigación en Computación, CIC-IPN, Instituto Politécnico Nacional, México.
[b]Instituto Nacional de Astrofísica Optica y Electrónica INAOE, Ciencias Computacionales, Sta María Tonanzintla, Puebla C.P. 72840, México.
[c]Instituto de Cibernética Matemática y Física, Ministerio de Ciencia Tecnología y Medio Ambiente, E#309 esq. a 15 Vedado, Ciudad de La Habana. C.P. 10400, Cuba.

**Abstract**

En este trabajo se presenta el lenguaje LCARS para especificar problemas de Reconocimiento de Patrones bajo el enfoque lógico-combinatorio (RPLC). Se revisa el contexto en que surge la necesidad del lenguaje y la solución que éste brinda a los problemas planteados. El desarrollo de esta herramienta computacional representa un claro ejemplo de la integración de algunas técnicas de la teoría de lenguajes formales para modelar problemas en el enfoque lógico-combinatorio. Por último, se revisan un Motor de Reconocimiento de Patrones llamado " SHELL" y un ambiente de solución de problemas llamado "STUDIO" que están siendo desarrollados a partir del lenguaje LCARS.

*Keywords*: Logical-combinatory approach, covering, fuzzy environments.

## 1 Motivación

El enfoque lógico-combinatorio del reconocimiento de patrones [1-3], como varios otros enfoques [4-6], proporciona un marco teórico para el estudio y solución de problemas de clasificación, pronóstico, optimización, soporte en la toma de decisiones, etc. Cada enfoque presenta un marco teórico distinto, fundamentado en principios filosóficos diversos y, por tanto, un conjunto de axiomas diferente.

Esta incompatibilidad de teorías genera la necesidad, para cada enfoque, de procurarse las herramientas necesarias para el desarrollo de investigaciones con su propia modalidad. Procurar estas herramientas, para cualquier disciplina, exige el desarrollo de meta-conocimiento: modelos sobre la forma de desarrollar, organizar, interpretar y manipular el conocimiento propio de la disciplina. Sólo cuando se dispone de tales modelos se pueden diseñar y construir herramientas de automatización, diagnóstico y soporte a la investigación.

En particular, en el enfoque lógico-combinatorio, varios modelos de meta-conocimiento han sido desarrollados, se encuentran publicados en diversos textos y artículos de investigación y son aplicados tanto por investigadores como por estudiantes en diversas partes del mundo [1]. Sin embargo, la gran mayoría de las herramientas computacionales construidas, han sido diseñadas para resolver instancias particulares de problemas, no bajo algún modelo general. Esta limitación se ha debido fundamentalmente a la complejidad en el manejo de dominios de definición para las variables o rasgos descriptivos y la definición de funcionales de comparación.

En este artículo se presenta el desarrollo de una herramienta computacional, fundamentada en modelos generales; un lenguaje formal de especificación de problemas. Notoriamente, el desarrollo de este lenguaje requirió de la aplicación de modelos y técnicas propias de otro enfoque distinto en el reconocimiento de patrones, el enfoque sintáctico-estructural. La característica esencial de este enfoque es modelar cualitativamente los problemas, a partir de la teoría de lenguajes formales y la teoría de máquinas. Por ello, el desarrollo de este lenguaje reviste una importancia especial, no sólo por sus aplicaciones en el enfoque lógico-combinatorio, sino como un ejemplo de integración de áreas en el desarrollo de meta-conocimiento.

## 2 El modelo subyacente

Exponer con detalle los modelos meta-cognitivos en los que se fundamenta el diseño del lenguaje LCARS, escapa definitivamente del alcance de este trabajo. Baste, por el momento, con una revisión acerca de algunos de los elementos más relevantes que dichos modelos incluyen. Para un estudio a profundidad consúltese [7-12 ].

En el enfoque lógico-combinatorio de reconocimiento de patrones (RPLC), los problemas que involucran uno o más cubrimientos presentan siempre un conjunto de elementos comunes. El enfoque incluye, por supuesto, varias metodologías y técnicas especiales para el tratamiento de estos elementos comunes. El modelo que sirve de fundamento al desarrollo del lenguaje LCARS (*Logic-Combinatory Algoritmic Recognition Specification*) contempla tres asociaciones conceptuales para cada uno de los objetos del universo en estudio:

1. su evaluación o descripción en términos de un conjunto de características o rasgos descriptivos (Descripción de los objetos).

2. su pertenencia a cada una de las posibles categorías en estudio (Pertenencia a las categorías).

3. información necesaria para evaluar, ya sea de manera cuantitativa o cualitativa, la comparación entre:

   - el valor de descripción de un objeto en un rasgo particular, y el valor de otro objeto en el mismo rasgo descriptivo (Criterios de Comparación Descriptiva).
   - el conjunto de valores de descripción de un objeto y el conjunto de valores de otro objeto (Función de Analogía entre patrones).

En términos computacionales lo anterior implica que las tres necesidades fundamentales en la especificación de problemas en RPLC pueden ser expresadas de la siguiente manera:

1. Se requiere de un formato estándar que permita expresar todos los aspectos constituyentes de un problema en RPLC, incluyendo su estructura, la descripción de los objetos y las restricciones involucradas en su solución. Este formato deberá ser suficiente para especificar problemas en entornos difusos.

2. Se requiere disponer de tipos de datos para representar el dominio de definición de cada rasgo descriptivo. Además de los tipos de datos comunes (enteros, reales, etc) debe existir la posibilidad de definir nuevos tipos de datos (tipos enumerados, rangos, etc.).

3. Se requiere poder expresar los criterios de comparación y funciones de analogía de manera tal que se puedan validar de acuerdo a los tipos de datos involucrados y se puedan realizar múltiples evaluaciones de dichos funcionales.

Durante la formalización de un problema de clasificación o de selección de rasgos, es necesario especificar elementos en dos categorías: estructurales y algorítmicos. Los elementos estructurales determinan la composición del problema; el universo de objetos en estudio, los rasgos descriptivos, el conjunto de categorías, etc. Los elementos algorítmicos, en cambio, determinan el comportamiento de los componentes; el dominio de definición de los rasgos descriptivos, los diversos funcionales de comparación, las restricciones en la solución del problema, etc.

# 3 Estructura del lenguaje

El lenguaje LCARS fue diseñado para satisfacer todos los requerimientos de especificación de problemas expresados arriba, tanto estructurales como algorítmicos. Estructuralmente se compone de secciones llamadas bloques, posiblemente anidados, que especifican todos los elementos componentes de un problema y la

forma jerárquica en que éstos se relacionan entre sí. Algorítmicamente contiene tipos de datos que determinan el dominio de definición de diversas variables e incluye estructuras algorítmicas de control para la especificación de funcionales de comparación.

En términos generales el lenguaje LCARS presenta las siguientes características:

1. Se trata de un lenguaje de propósito específico fuertemente estructurado y tipificado.
2. Lenguaje algorítmico completo.
3. Con funciones pre-definidas adecuadas para el área de aplicación RPLC (como especificación de valores de criterios de comparación en cualquier rasgo, etc.)
4. Con una sintaxis orientada al matemático no-computólogo.

A la especificación en lenguaje LCARS de un problema le llamaremos *programa*. Así pues, un programa LCARS es en sí mismo un único bloque llamado *Proyect* que contiene dentro de sí tres sub-bloques principales: *Problem*, *Covering* y *Patterns*. Cada bloque se delimita por su nombre y un terminador con el término *End* seguido por el nombre del bloque que termina.

```
Proyect "nombre del proyecto" is
    Problem
        ...
    End Problem ;
    Covering
        ...
    End Covering ;
    Patterns
        ...
    End Patterns ;
End Project
```

El bloque *Problem* especifica la estructura del problema, las restricciones impuestas a su solución, así como las constantes y tipos de datos involucrados en el problema:

```
Problem
    ProblemType : tipo de problema ;
    ClassStyle : naturaleza de las clases ;
    ClassInteraction : relación entre las clases ;

    Restrictions
        atributo de resticción : valor ;
        ...
    End Restrictions ;

    DataTypes
```

```
            nombre del tipo : definición ;
            ...
        End DataTypes ;
        Constants
            nombre de constante : tipo de datos := valor ;
            ...
        End Constants ;
    End Problem ;
```

El bloque *Covering* especifica todos los elementos del cubrimiento inicial del problema:

```
    Covering
        Features
            nombre del rasgo : tipo de datos ;
            ...
        End Features ;
        Descriptions
            nombre objeto : [ev rasgo#1 \ pertenencia], etc... ;
            ...
        End Descriptions ;
        Classes
            nombre de clase#1, etc... ;
        End Classes ;
        Memberships
            nombre de objeto : pert. a clase#1, etc... ;
            ...
        End Memberships ;
        Criteria
            CC [ rasgo ] : tipo_de_datos is definición ;
            ...
        End Criteria ;
        Analogy
            AF [ ] : tipo_de_datos is definición ;
        End Analogy ;
    End Covering ;
```

Por último, el bloque *Patterns* especifica los objetos a clasificar:

```
    Patterns
        nombre objeto : [ev rasgo#1 \ pertenencia], etc... ;
        ...
    End Patterns ;
```

# 4 Tipos de Datos

Si se pretende que LCARS realmente sirva como base para la construcción de diversas soluciones en RPLC, su diseño y especificación deben ser tan dinámicos y generales como el enfoque mismo. Los tipos de datos y estructuras internas que conforman el lenguaje deben adaptarse a la mayor cantidad posible de áreas de aplicación. Evidentemente la especificación de LCARS será revisada y actualizada continuamente para adaptarla a este entorno dinámico. Para la primera versión se ha diseñado un conjunto básico de tipos de datos agrupados en dos clases: tipos de datos de sistema y tipos de datos definidos por el usuario. Los tipos de datos de sistema son aquellos pre-definidos en el lenguaje y representan la naturaleza más común de los datos:

| Nombre del tipo | Significado |
|---|---|
| *Boolean* | Datos de naturaleza binaria |
| *Natural* | Números naturales |
| *Integer* | Números enteros |
| *PositiveReal* | Números reales positivos |
| *Real* | Números reales |
| *Character* | Caracteres |
| *Alphanumeric* | Cadenas |



Por su parte, la capacidad de admitir tipos de datos definidos por el usuario, imprime la flexibilidad necesaria para la formalización de problemas sin deterioro de la legibilidad ni la facilidad de mantenimiento a los programas construidos. Para la primera versión se han especificado tres clases de tipos de datos que pueden ser definidos por el usuario:

| Nombre del tipo | Significado |
|---|---|
| *List* | Listas de cadenas (tipos enumerados) |
| *Subrange* | Subrangos discretos de tipos ordinales |
| *Linguistic* | Variables |

Para futuras versiones de LCARS se contempla la posibilidad de agregar tipos para la representación de datos binarios (imágenes) y datos vectoriales. Esto debido a que el manejo de imágenes es una de las aplicaciones más trabajadas en cualquier enfoque del reconocimiento de patrones. Otras posibles extensiones se determinarán de acuerdo a las necesidades prácticas de la comunidad de usuarios.

En lenguaje LCARS el uso de los tipos de datos se concentra en los bloques *Problem* y *Covering*. Dentro de *Problem* el bloque *DataTypes* se usa para definir los tipos de datos de usuario para el problema especificado. En el bloque *Constants* se definen las constantes usadas en el problema, y el tipo de datos de cada una. En LCARS todas las constantes son tipificadas.

En los diferentes sub-bloques dentro de *Covering* se hace uso, tanto de los tipos de datos de sistema, como de los tipos de datos definidos por el usuario,

para especificar los dominios de definición de rasgos descriptivos, criterios de comparación y función de analogía.

Obsérvese que la sintáxis usada en el bloque *Patterns*, así como en el subbloque *Descriptions* dentro de *Covering*, está diseñada para admitir la especificación de objetos en términos difusos. De igual forma, las pertenencias especificadas dentro del bloque *Memberships* son siempre datos de tipo *PositiveReal*. Estas dos características posibilitan al lenguaje LCARS para especificar problemas en entornos difusos.

# 5 Comparación y Analogía

La capacidad de definir tipos de datos, criterios de comparación y funciones de analogía es, sin lugar a dudas, la característica más relevante del lenguaje LCARS y la que lo distingue de otras herramientas computacionales para solucionar problemas específicos.

Como ya se mencionó, en un programa LCARS, el bloque *Covering* contiene, entre otros, los bloques *Criteria* y *Analogy*. Es en estos bloques que se definen los funcionales de comparación con la siguiente sintaxis:

| Criterios de Comparación | Función de Analogía |
|---|---|
| **CC** [*rasgo*] : *tipo_de_datos* **is** | **AF** [ ] : *tipo_de_datos* **is** |
| **Variables** | **Variables** |
| *definición de variables* | *definición de variables* |
| **Begin** | **Begin** |
| *instrucciones algorítmicas* | *instrucciones algorítmicas* |
| ... | ... |
| **Return** (*respuesta*) ; | **Return** (*respuesta*) ; |
| **End;** | **End;** |

Dentro de cada bloque delimitado por **Begin-End** se define el funcional de comparación y el valor del funcional se regresa mediante la instrucción **Return**. Para ello se dispone tanto del conjunto de operadores aritméticos y lógicos fundamentales, como de las seis estructuras algorítmicas de control. Las estructuras de control tienen la siguiente sintaxis:

Estructuras de decisión

| | |
|---|---|
| **If** (*condición*) **do** | **Case**(*variable*) **equals** |
| *bloque* | *valor#1* **do** *bloque #1* |
| **End If ;** | **end** |
| | *valor#2* **do** *bloque #2* |
| | **end** |
| | ... |
| **If** (*condición*) **do** | **otherwise do** |
| *bloque #1* | *bloque #n* |

**End Case ;**

**end else**
*bloque #2*
**End If ;**

Estructuras de Repetición

**While** (*condición*) **loop**
*bloque*
**End While ;**

**Loop**
*bloque*
**Until** (*condición*);

**For** *variable* **in** [ *min, max , incremento* ] **loop**
*bloque*
**End For ;**

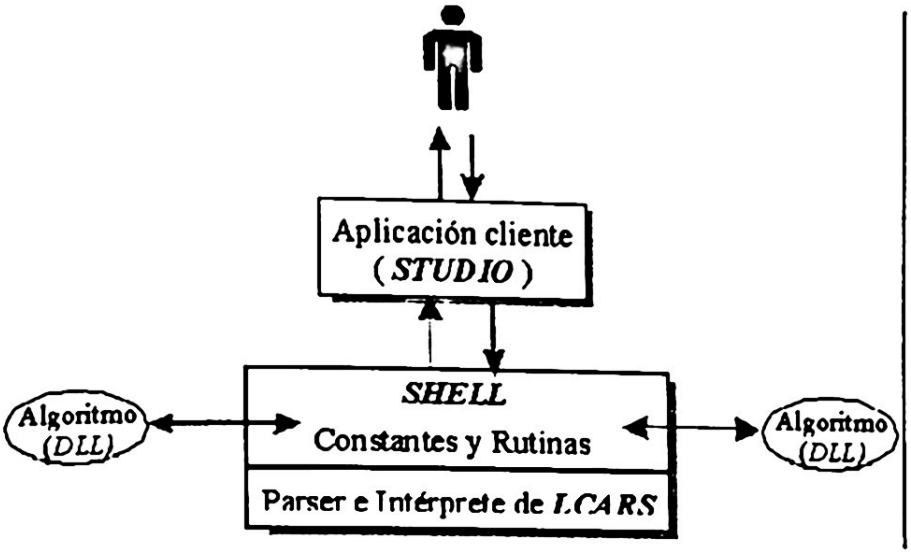
**Loop**
*bloque*
**While** (*condición*);

Con esto se muestra que el lenguaje LCARS efectivamente soluciona los problemas de especificación planteados en los requerimientos anteriores.

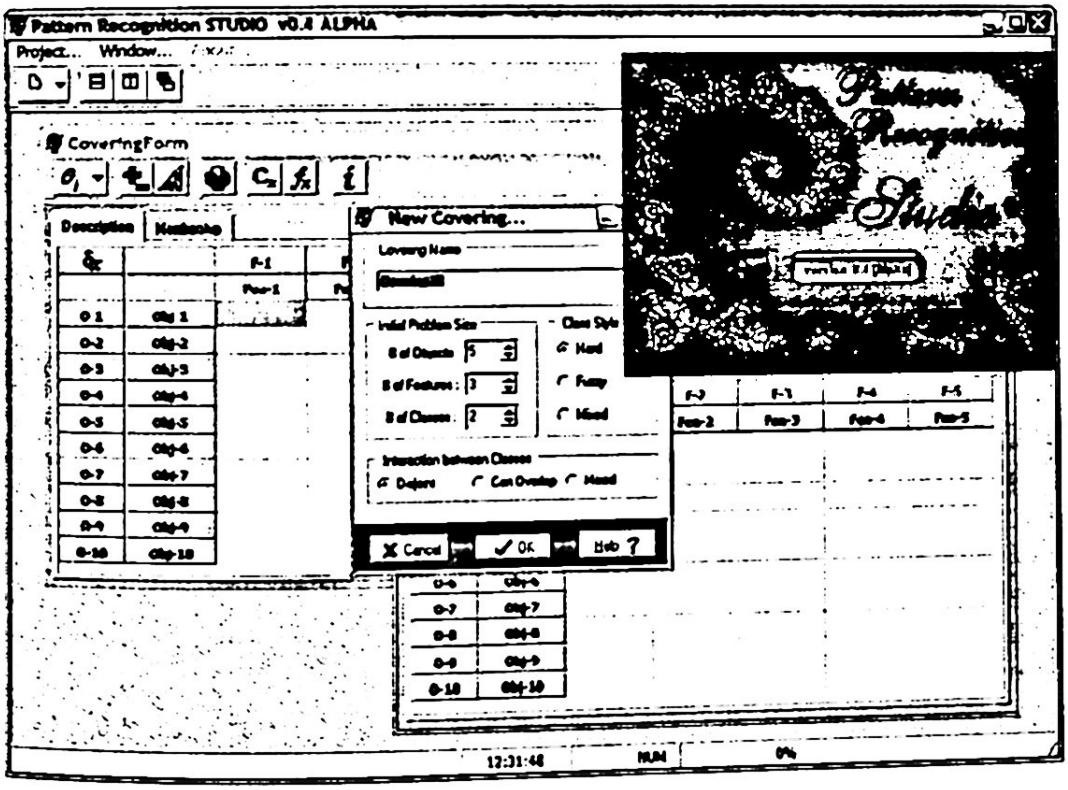
# 6 Construcción de un Motor de Reconocimiento de Patrones

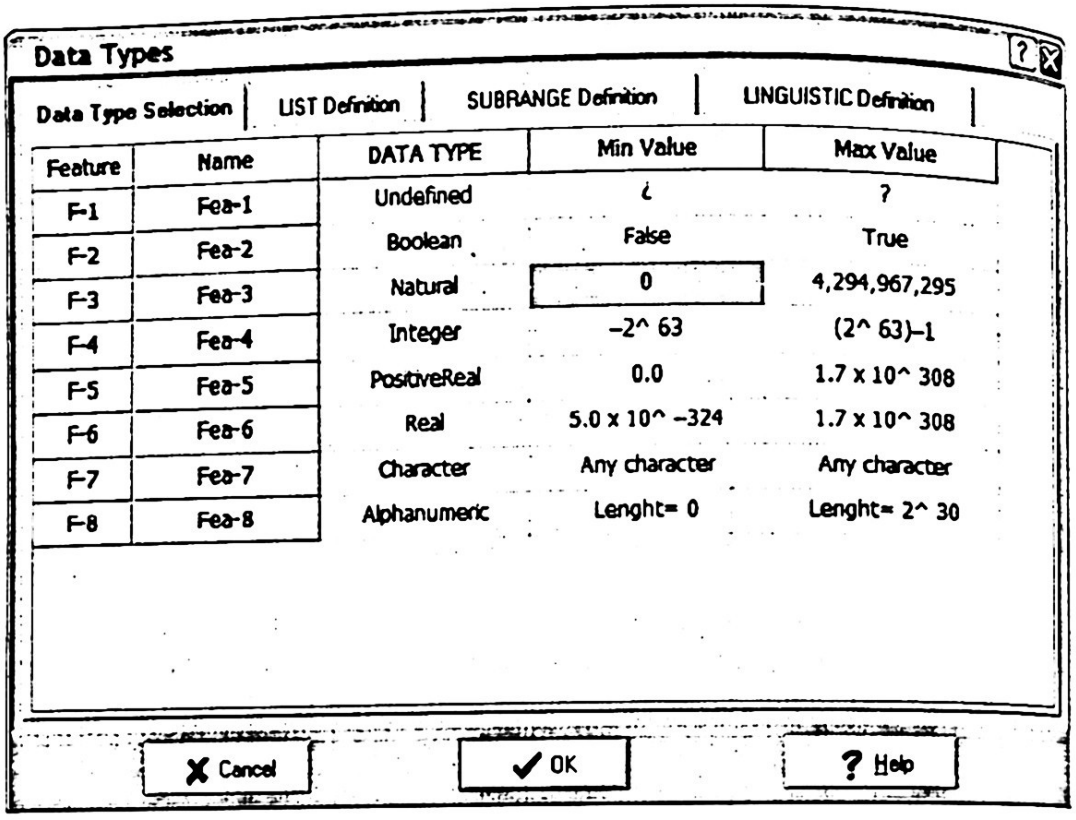
Disponer de un lenguaje de especificación de problemas no constituye una solución completa para las necesidades de investigación y docencia en RPLC. Si bien LCARS puede especificar problemas con diferentes características, es sólo eso, una herramienta de especificación. Para realmente disponer de una solución completa, es necesario agregar otra herramienta capaz de generar las estructuras de datos derivadas de la especificación (matriz de diferencias, matriz básica, etc.), así como aplicar sobre el problema especificado diversos algoritmos de clasificación, selección de rasgos, etc.

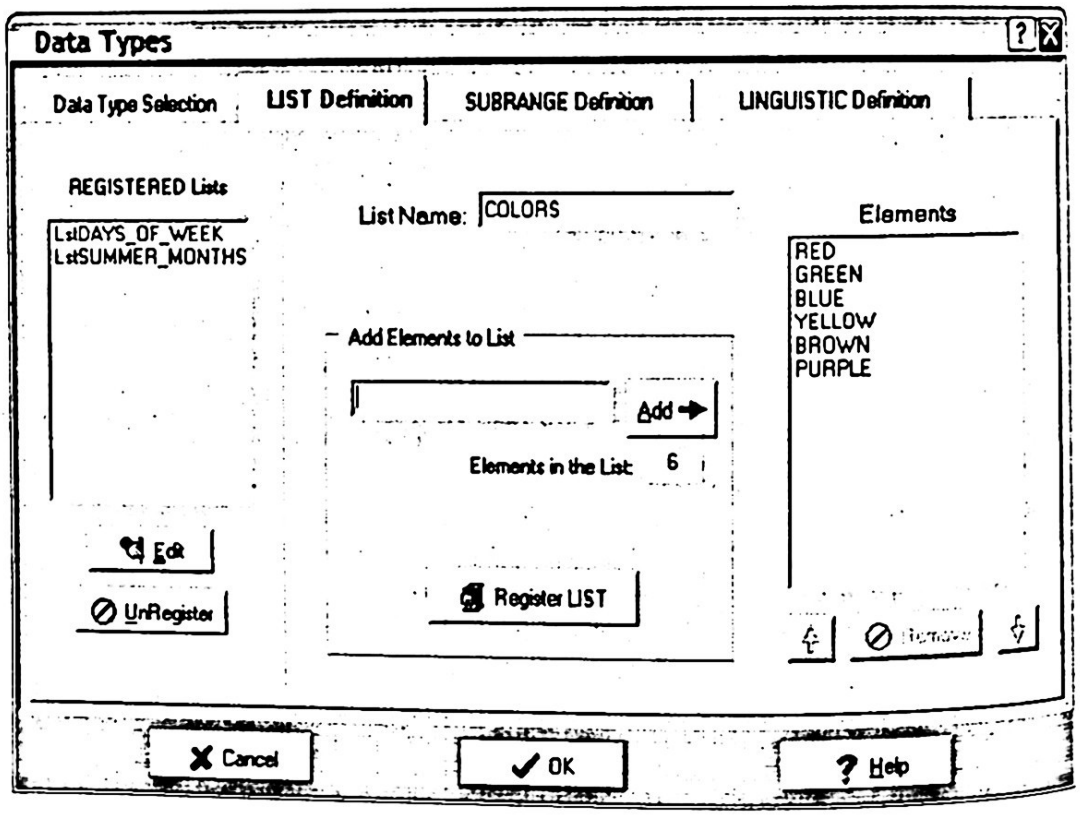
Por ello, de forma paralela a diseño de LCARS, se ha abordado el diseño y construcción de un motor de reconocimiento de patrones llamado SHELL. Este motor es una biblioteca de constantes y rutinas diseñadas para trabajar en conjunto con una especificación LCARS. De hecho, los principales elementos constituyentes del SHELL son un parser y un intérprete de LCARS. Para lograr una solución que impacte al mayor número de grupos de investigación, el SHELL se ha organizado y compilado de acuerdo con el estándar COM de *Microsoft*. Ello permitirá que cualquier investigador pueda hacer uso del SHELL desde cualquier lenguaje de programación que soporte COM. En esencia, el SHELL se ha diseñado como un componente para la construcción de aplicaciones de RPLC bajo una arquitectura cliente-servidor. En dicha arquitectura el SHELL juega el papel de servidor o motor de reconocimiento de patrones, proporciona una interfaz hacia las aplicaciones cliente (API) y tiene la capacidad de ser extendido mediante algoritmos programados en forma de bibliotecas dinámicas (DLLs). Las aplicaciones cliente pueden entonces generar soluciones de un mayor nivel de abstracción.

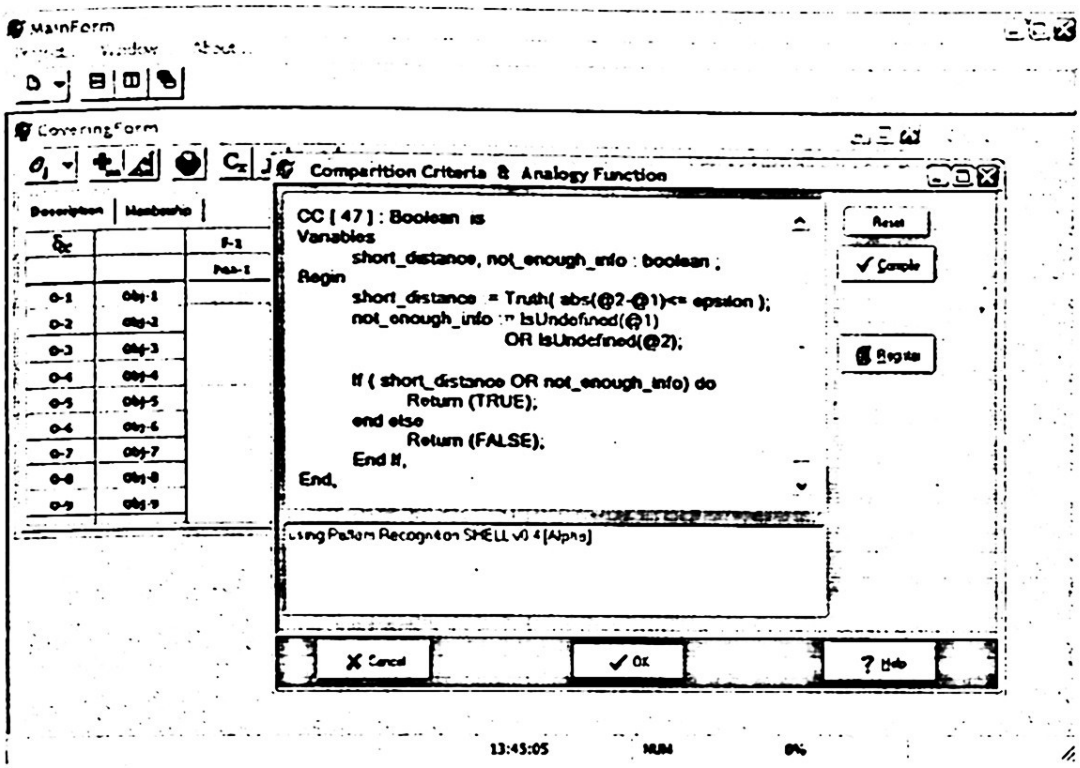



Como prueba y ejemplo de la integración del SHELL en aplicaciones de RPLC, se está construyendo también una aplicación cliente llamada Pattern Recognition STUDIO. Como su nombre lo indica, este STUDIO pretende ser un espacio para la experimentación sobre problemas y algoritmos de RPLC. Fundamentalmente se trata de una interfaz gráfica para ayudar al investigador a definir, de forma amigable, problemas en lenguaje LCARS y a procesarlos mediante las rutinas del SHELL. Algunas imágenes de este STUDIO, en su estado de desarrollo actual, se muestran a continuación.




Data Types
Data Type Selection
LIST Definition
SUBRANGE Definition
LINGUISTIC Definition
Feature
Name
F-1
Fea-1
F-2
Fea-2
F-3
Fea-3
F-4
Fea-4
F-5
Fea-5
F-6
Fea-6
F-7
Fea-7
F-8
Fea-8
DATA TYPE
Min Value
Max Value
Undefined
¿
?
Boolean
False
True
Natural
0
4,294,967,295
Integer
–2^ 63
(2^ 63)–1
PositiveReal
0.0
1.7 x 10^ 308
Real
5.0 x 10^ –324
1.7 x 10^ 308
Character
Any character
Any character
Alphanumeric
Lenght= 0
Lenght= 2^ 30
Cancel
OK
Help


Data Types
Data Type Selection
LIST Definition
SUBRANGE Definition
LINGUISTIC Definition
REGISTERED Lists
LstDAYS_OF_WEEK
LstSUMMER_MONTHS
List Name:
COLORS
Elements
RED
GREEN
BLUE
YELLOW
BROWN
PURPLE
Add Elements to List
Add
Elements in the List:
6
Edit
UnRegister
Register LIST
Cancel
OK
Help




# 7 Conclusiones

La actividad meta-cognitiva es un signo de madurez en cualquier disciplina. Sólo cuando se dispone de modelos meta-cognitivos en cualquier rama del conocimiento es posible evaluar el avance logrado hasta un cierto momento. ubicarlo en su justo contexto y avanzar hacia un nuevo estado con mayor nivel de abstracción que potencíe la construcción de nuevo conocimiento. Es así como evoluciona cualquier rama del saber humano y en particular, la forma en que las matemáticas han evolucionado. Para lograr ese cambio es necesario sistematizar e incluso automatizar aspectos de menor nivel de abstracción y aprovechar esa sistematización para la construcción de nuevos modelos más abstractos.

La computadora es precisamente un dispositivo para la automatización de procesos, por ello, representa una gran ayuda para la construcción de herramientas como el lenguaje LCARS y el SHELL que se pretende jueguen ese papel de impulsores del avance en RPLC. Una vez terminada la construcción de estas herramientas, su éxito dependerá de la administración tecnológica que de ellos se haga. Si y sólo si se logra construir una verdadera comunidad epistémica alrededor de ellas, será posible mantenerlas en sincronía con las necesidades tecnológicas de RPLC así como en constante actualización.

# 8 Referencias


[1] Ruiz-Shulcloper, J. & Mongi, A. .A. (2002), Logical Combinatorial Pattern Recognition: A Review. In *Recent Research Developments in Pattern Recognition*, ed. Pandalai, Pub. Transword Research Networks, USA., (To appear).

[2] Ruiz-Shulcloper J., Guzman-Arenas A., & Martínez-Trinidad J. F. (1999). *Logical Combinatorial Approach to Pattern Recognition: I. Feature selection and supervised classification.* Editorial Politecnica, Mexico.

[3] Martínez-Trinidad J. F.& Guzmán-Arenas A. (2001). The logical combinatorial approach to pattern recognition an overview through selected works, *Pattern Recognition* 34(4), 741-751.

[4] Fukunaga K. (1990). *Introduction to Statistical Pattern Recognition*, Academic Press, London.

[5] Duda R. O., Hart P. E., & Stork D. G. (2000). *Pattern Classification* (2nd ed), Wiley, New York , NY.

[6] Schalkoff Robert J. (1992). *Pattern Recognition: Statistical, Structural and Neural Approaches*, John Wiley & Sons, Inc.

[7] Ruiz-Shulcloper J. & Lazo-Cortés M. (1999). Mathematical Algorithms for the Supervised Classification Based on Fuzzy Partial Precedence, *Mathematical and Computer Modelling*, 29(4), 111-119.

[8] Godoy-Calderon S. (1996). "Generalization of Testor and Typical Testor concepts for fuzzy environments". M. Sc., Thesis, CINVESTAV-IPN, Mexico City, (In Spanish).

[9] Lazo-Cortes M., Ruiz-Shulcloper J. & Alba-Cabrera E. (2001). An overview of the evolution of the concept of testor, *Pattern Recognition*, 34(4), 753-762.

[10] Alba-Cabrera E. (1998). "Nuevas extensiones del concepto de testor para diferentes tipos de funciones de semejanza". Tesis Doctoral. Universidad de la Habana, Cuba.

[11] Carrasco-Ochoa J. A. (2001). "Sensibilidad en Reconocimiento Lógico Combinatorio de Patrones". Tesis Doctoral, Centro de Investigación en Computación, IPN, México, D.F.

[12] Sánchez-Díaz G. (2001). "Desarrollo de algoritmos para el agrupamiento de grandes volúmenes de datos mezclados". Tesis Doctoral, Centro de Investigación en Computación, IPN, México, D.F.

# Chunking Suave: Un Método de Descomposición para el Entrenamiento de Máquinas de Soporte Vectorial

M.C. Ricardo del Angel Pérez Flores
Departamento de Cómputo e Informática
Universidad Autónoma del Carmen
e-mail: rperez@pampano.unacar.mx

Dr. Arturo Hernández Aguirre
Area de Ciencias de la Computación
Centro de Investigación en Matemáticas, A.C.
e-mail: artha@cimat.mx

**Resumen**

Este artículo propone el método de Chunking Suave para el rápido entrenamiento de una Máquinas de Soporte Vectorial. Este método, basado en Conjuntos de Trabajo, es una modificación del método de Chunking propuesto originalmente por Vapnik. Las ventajas del Chunking Suave son su velocidad y una menor cantidad de vectores de soporte. Describimos nuestro método, se reportan experimentos con la base de datos MNIST, y explicamos porqué un SVM de Chunking Suave generaliza de manera comparable a los algoritmos tradicionales.

**Palabras clave:** Máquinas de Soporte Vectorial, Chunking, Chunking Suave, Teoría Estadística del Aprendizaje, Reconocimiento de Patrones.

## 1. Introducción.

Las Maquinas de Soporte Vectorial (Support Vector Machines o SVM) constituyen una de las técnicas mas reciente para reconocimiento de patrones. En 1995, Vapnik [8] y su equipo de colaboradores establecieron las bases teóricas de la Teoría Estadística del Aprendizaje (Statistical Learning Theory o SLT), dando como resultado práctico las SVMs. Estas, afirma Burges [1], son un paradigma aparte de las Redes Neuronales que, aunque tienen muchas similitudes, están mejor fundamentadas en la teoría. Tienen mejor capacidad de generalización; es decir, capacidad para clasificar correctamente datos distintos al conjunto de entrenamiento. La diferencia entre los métodos es notable; el entrenamiento de la Red Neuronal no controla la generalización, sino que asume que ésta será correcta si el error de entrenamiento es pequeño sobre un conjunto de entrenamiento lo suficientemente grande. El entrenamiento de una SVM, en cambio, controla simultáneamente ambos parámetros: error de entrenamiento y generalización.

Una SVM efectúa la clasificación de objetos puntuales en dos clases por medio de una superficie de decisión determinada por ciertos puntos del conjunto de entrenamiento,

conocidos como vectores de soporte. Dicha superficie se obtiene al resolver un problema de Programación Cuadrática Convexa con restricciones lineales, cuyo número de variables es igual al número de datos de entrenamiento. La solución de este problema es el objetivo de este artículo.

# 2. Formulación del Problema de Clasificación con SVM.

Tenemos $l$ muestras independientes e idénticamente distribuidas, provenientes de una distribución de probabilidad $P(x,y)$ desconocida, consistentes en pares conformados por un vector y una etiqueta de clase $(x_1,y_1),(x_2,y_2),...,(x_l,y_l)$, donde $x_i \in R^n$ y $y_i \in \{1,-1\}$. Deseamos construir una SVM capaz de mapear los valores de los vectores $x_i$ a los valores de las etiquetas $y_i$ correspondientes por medio de una función de decisión $y = f : R^n \to \{1,-1\}$.

## 2.1. Caso Linealmente Separable.

Es el caso más simple, Figura 2.1, donde los datos pueden ser clasificados por un hiperplano orientado, conocido como hiperplano de separación, con ecuación $w \cdot x + b = 0$, donde $w$ es un vector normal al hiperplano llamado vector de pesos y $b$ conoce como sesgo.

Decimos que el conjunto de entrenamiento es linealmente separable si existen $w \in R^n$ y $b \in R$, tales que todos los datos de entrenamiento satisfacen las siguientes restricciones:

$$w \cdot x_i + b \geq 1 \text{ para } y_i = 1 \tag{2.1}$$

$$w \cdot x_i + b \leq -1 \text{ para } y_i = -1 \tag{2.2}$$

La distancia entre $H_1$ y $H_2$ nos da el margen de separación para las dos clases, el cual es $M = \frac{2}{\|w\|}$. Por lo tanto, para encontrar el par de hiperplanos que dan el máximo margen, y al mismo tiempo el hiperplano de separación, es necesario maximizar distancia $M$, lo que equivale a minimizar el valor de $\|w\|$, sujeto a las restricciones dadas.

De esta forma, llegamos al siguiente problema de Programación Cuadrática Convexa:

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2 = \frac{1}{2} w \cdot w \tag{2.3}$$

sujeto a las restricciones

$$-y_i (w \cdot x_i + b) + 1 \leq 0 \text{ para } i = 1,2,...,l. \tag{2.4}$$

La solución de este problema de optimización se encuentra en el Punto Silla Lagrangiano primal del mismo, dado por

$$L_P(w,b,\alpha) = \frac{1}{2} w \cdot w + \sum_{i=1}^{l} \alpha_i \left[ -y_i (w \cdot x_i + b) + 1 \right]$$

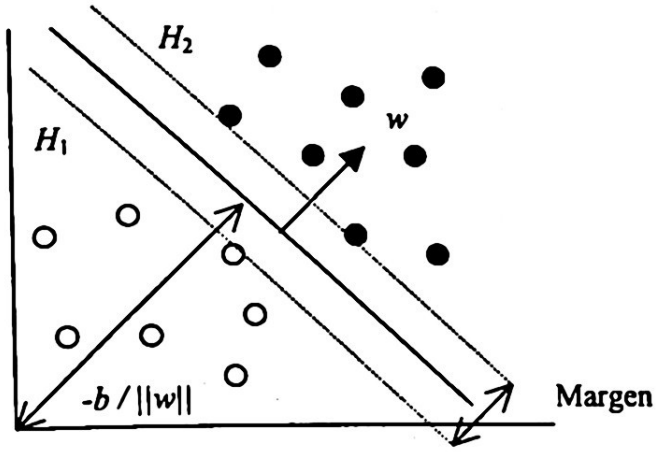



*Figura 2.1. El hiperplano de separación para el caso linealmente separable en $R^2$.*

donde cada $\alpha_i$ para $i = 1,2,...,l$, es un multiplicador de Lagrange. Dicha solución es la misma que para el problema dual. Para encontrarlo debemos minimizar el Lagrangiano respecto a $w$ y $b$, al tiempo que es maximizado con respecto a $\alpha = (\alpha_1, \alpha_2, ..., \alpha_l)$.

Así, podemos replantear el problema como

$$\max_{\alpha} \quad L_D(\alpha) = -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j x_i \cdot x_j + \sum_{i=1}^{l}\alpha_i \tag{2.6}$$

sujeto a las restricciones

$$\sum_{i=1}^{l}\alpha_i y_i = 0 \tag{2.7}$$

$$\alpha_i \geq 0 \text{ para } i = 1,2,...,l. \tag{2.8}$$

Este problema cumple con la forma estándar del problema de Programación Cuadrática Convexa con una restricción de igualdad (2.7) y $l$ de desigualdad (2.8). Puede manejado con mayor facilidad, pues las restricciones originales son sustituidas por restricciones que incluyen a los multiplicadores de Lagrange, además de que los datos de entrenamiento aparecen como simples productos punto de vectores. Dicha situación resulta ventajosa en el caso no lineal, que se revisará en la siguiente sección.

la solución, a cada punto de entrenamiento le corresponde un multiplicador de Lagrange óptimo $\alpha_i^*$. Los puntos para los cuales su respectiva $\alpha_i^* > 0$ están sobre alguno los hiperplanos $H_1$ o $H_2$, y se conocen como vectores de soporte. Todos los demás puntos de entrenamiento tienen $\alpha_i^* = 0$ y están a los lados de $H_1$ o $H_2$ de modo que se cumple la desigualdad de la restricción.

Para las SVM, los vectores de soporte son los elementos críticos del conjunto de entrenamiento. Ellos definen la región de decisión, cumplen con la igualdad en las restricciones (2.8) y si los eliminaramos cambiaría la solución encontrada, contrario a lo que sucedería, si todos los demás puntos de entrenamiento fueran removidos o movidos de lugar sin cruzar $H_1$ o $H_2$.

Una vez solucionado el problema, tendremos un vector $\alpha^*$ de multiplicadores de Lagrange, con los cuales podemos calcular el vector de pesos óptimo y el sesgo óptimo

$$w^* = \sum_{i=1}^{l} \alpha^*_i x_i y_i \tag{2.9}$$

$$b^* = \frac{1}{k}\left[\sum_{s=1}^{k}\left(y_s - w^* \cdot x_s\right)\right] \tag{2.10}$$

donde $k$ es el número de vectores de soporte.

Así, obtenemos la ecuación del hiperplano de decisión $u(x) = w^* \cdot x + b$.

## 2.2. Caso No Lineal.

En la mayoría de los casos, la separación lineal en el espacio de entrada es demasiado restrictiva para un uso práctico. La teoría puede extenderse a regiones de decisión no lineales mapeando los puntos de entrada hacia puntos en un espacio de rasgos, buscando luego el hiperplano óptimo de separación en dicho espacio.

Dados los punto de entrada, utilizamos la función $\varphi : R^n \to R^m$ para mapear dichos puntos hacia el espacio de rasgos $R^m$ (que por lo general es un Espacio de Hilbert). A un hiperplano óptimo de separación (una región de decisión lineal) en $R^m$ le corresponde una región de decisión no lineal en el espacio de entrada, que solo puede determinarse si la función $\varphi$ es completamente conocida. La ventaja es que $\varphi$ interviene solamente en el producto punto de dos puntos del espacio de rasgos, mediante una función simétrica conocida como kernel, que se calcula a partir de los puntos en el espacio de entrada como $k(u,v) = \varphi(u) \cdot \varphi(v)$. Aquí no es necesario conocer $\varphi$.

Cabe señalar que tendría un altísimo costo computacional efectuar explícitamente estos mapeos y luego calcular los productos punto. Sin embargo, cuando un kernel cumple las Condiciones de Mercer (ver [6]), el mapeo es implícito. Esto es, trabajamos en $R^n$ aunque la solución se encuentre en $R^m$.

La región de separación no lineal puede encontrarse como la solución al problema de Programación Cuadrática antes mencionado, pero con $\varphi(x_i)$ en vez de cada $x_i$. Es necesario modificar el planteamiento del problema como:

$$\max_{\alpha} \quad L_D(\alpha) = -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j k(x_i, x_j) + \sum_{i=1}^{l}\alpha_i \tag{2.11}$$

sujeto a las restricciones

$$\sum_{i=1}^{l}\alpha_i y_i = 0 \tag{2.12}$$

$$\alpha_i \geq 0 \text{ para } i = 1,2,...,l. \tag{2.13}$$

La ventaja del problema dual sobre el problema primal, por depender de los multiplicadores de Lagrange y de productos punto de vectores de entrenamiento se hace evidente aquí: si el problema dependiera del vector de pesos, tendríamos tantas incógnitas como la dimensión del espacio de rasgos, que se ha visto que tiene una dimensión demasiado grande.

El sesgo óptimo se obtiene como

$$b^* = \frac{1}{k}\left[\sum_{s=1}^{k}\left(y_s - \sum_{i=1}^{l}\alpha_i y_i \cdot k(x_s, x_i)\right)\right] \tag{2.14}$$

donde $k$ es el número de vectores de soporte.

La clasificación se efectúa mediante la misma función que en el caso linealmente separable, pero en este caso no es posible calcular el vector de pesos explícitamente debido a que necesitaríamos conocer $\varphi$, por lo tanto tenemos

$$u(x) = \sum_{i=1}^{l}\alpha_i y_i k(x_i, x) + b^* \tag{2.15}$$

Así, la extensión de la teoría se traduce en encontrar kerneles que identifiquen a ciertas familias de regiones de decisión que puedan escribirse como productos punto en algún espacio de rasgos. En la práctica, será necesario emplear el enfoque no separable para el caso no lineal y escoger el kernel que combine la mejor generalización con un mapeo al espacio de rasgos de la menor dimensión posible. Los kerneles mas conocidos y utilizados, según describe Gunn [3], son:

| Tipo de Kernel | Función |
|---|---|
| **Polinomiales de Grado d** | $k(u,v) = (u \cdot v)^d \quad k(u,v) = (u \cdot v + 1)^d$ |
| **Funciones de Base Radial (RBF) Gaussianas** | $k(u,v) = e^{\frac{\|u-v\|^2}{2\sigma^2}}$ |
| **Perceptrones Multicapas** | $k(u,v) = \tanh(ku \cdot v - \delta)$ |

Existen muchos otros tipos de kerneles, tales como RBF exponenciales, series de Fourier, splines y splines B. Además, es posible construir kerneles más complicados mediante la suma o el producto de varios kerneles [3].

## Métodos de Conjunto de Trabajo.

Aquí podemos distinguir dos clases de problemas: los problemas "pequeños", que se puede resolver utilizando cualquier paquete de optimización de propósito general que resuelva problemas de Programación Cuadrática Convexa con restricciones lineales. Estas técnicas requieren que los datos sean mantenidos en memoria en forma de una matriz, llamada Hessiano. La complejidad del problema de entrenamiento crece con el tamaño de esta matriz, limitando los enfoques a conjuntos de pocos miles de datos.

Para problemas mayores deseamos tomar las ventajas que forman las bases de los métodos de conjunto de trabajo (working set) en optimización: si uno supiera por adelantado cuales restricciones estuvieran activas, podría ser posible descartar todas las restricciones inactivas y simplificar el problema. Esto lleva a varias estrategias, todas basadas de alguna manera en conjeturas sobre el conjunto activo, y restringiendo el entrenamiento a dichas conjeturas. El punto importante es seleccionar el conjunto de trabajo de tal forma que la optimización del correspondiente subproblema de Programación Cuadrática lleve a una mejora en la totalidad de la función objetivo.

Se ha reportado que estos métodos trabajan muy bien en la práctica y hacen posible repartir con conjuntos de datos de varias decenas de miles de puntos [5].

## 3.1. El Método de Chunking.

El método de Chunking [8] (por pedazos) se inicia entrenando una SVM a partir de un subconjunto pequeño de muestras elegidas arbitrariamente, llamado "chunk" ("pedazo"). Las demás muestras se prueban con el clasificador resultante y se eligen los $N_V$ puntos que más violen las condiciones de Karush-Kuhn-Tucker (KKT), junto con los $N_S$ vectores de soporte del pedazo entrenado, para formar un nuevo pedazo de tamaño $M = N_S + N_V$. Este proceso se realiza iterativamente, inicializando el vector $\alpha$ para cada nuevo subproblema con los valores de salida del anterior, hasta que todos los puntos satisfacen las condiciones KKT (explicadas en la Sección 4).

En cada iteración se utiliza un optimizador cualquiera para entrenar el pedazo. Dicho pedazo puede cambiar de tamaño en forma variable, dependiendo del valor de $N_V$ dado por el usuario. Generalmente, el pedazo crece (aunque puede decrecer) hasta que en la última iteración contiene a los vectores de soporte de todo el conjunto de entrenamiento que representan a las restricciones activas, lo cual hace que se cumpla la condición de salida. Es por eso que los problemas en que se aplique este método deben tener un número esperado de vectores de soporte bajo.

## 3.2 Los Métodos de Descomposición y SMO.

El método anterior requiere que el número de vectores de soporte $N_S$ sea lo suficientemente pequeño para que un Hessiano de tamaño $(N_S + N_V) \times (N_S + N_V)$ quepa en la memoria. Un algoritmo de conjunto de trabajo alternativo que supera esta limitación fue propuesto por Osuna, Freund y Girosi [5]. En este algoritmo, como en el anterior, se entrena una pequeña porción de los datos de entrenamiento en cada iteración, con la diferencia de que el tamaño del conjunto de trabajo permanece fijo y pueden haber más vectores de soporte que datos en dicho conjunto.
Consiste en particionar el conjunto de entrenamiento en dos conjuntos $B$ y $N$, donde $B$ será el conjunto de trabajo de tamaño $M$. El vector $\alpha$ también se divide en $\alpha_B$ y $\alpha_N$, de modo que $\alpha_N$ permanecerá fijo y $\alpha_B$ cambiará en cada iteración. En cada iteración, se realiza un intercambio de datos entre los dos conjuntos, reemplazándose algunos valores de $B$ por otros de $N$, de acuerdo a alguna heurística determinada.

El problema es independiente del número de vectores de soporte, y puede probarse que la función objetivo mejora en cada iteración, por lo que este algoritmo debe converger hacia el óptimo global en un número finito de iteraciones.

El algoritmo de Optimización Secuencial Mínima (Sequential Minimal Optimization o SMO) [7] se deriva tomando la idea del método de descomposición hasta su extremo optimizando un subconjunto mínimo de dos puntos en cada iteración. Fue propuesto por Platt [7]. El poder de esta técnica reside en el hecho de que el problema de optimización para dos puntos permite una solución analítica. En cada paso, SMO escoge dos elementos $\alpha_i$ y $\alpha_j$, con $i \neq j$, para optimizarlos, encuentra los valores óptimos para estos dos parámetros dado que todos los otros están fijos, y actualiza el vector $\alpha$ de acuerdo a esto. La opción de los dos puntos es determinada por una heurística, mientras que la optimización de los dos multiplicadores es realizada analíticamente.

# 4. El Método de Chunking Suave.

Una buena manera de revisar que algún algoritmo trabaja es comprobando que la solución satisface todas las condiciones KKT para el problema primal, pues son condiciones necesarias y suficientes para que la solución sea óptima. En los métodos de conjuntos de trabajo se realizan iteraciones en cada una de las cuales se construyen SVM, con el fin de probar si todos los datos cumplen con las condiciones KKT, las cuales están dadas por las siguientes ecuaciones:

$$\frac{\partial f}{\partial w} = -\sum_{i=1}^{l} \alpha_i^* \frac{\partial g_i}{\partial w} \Rightarrow w = -\sum_{i=1}^{l} \alpha_i^* (-y_i x_i) \Rightarrow w = \sum_{i=1}^{l} \alpha_i^* y_i x_i \tag{4.1}$$

$$\frac{\partial f}{\partial b} = -\sum_{i=1}^{l} \alpha_i^* \frac{\partial g_i}{\partial b} \Rightarrow 0 = -\sum_{i=1}^{l} \alpha_i^* y_i \Rightarrow \sum_{i=1}^{l} \alpha_i^* y_i = 0 \tag{4.2}$$

$$\alpha_i^* \geq 0 \text{ para } i = 1,2,...,l, \tag{4.3}$$

$$\alpha_i^* g_i(w^*, b^*) = \alpha_i^* \left[ -y_i \left( w^* \cdot x_i + b^* \right) + 1 \right] = 0 \text{ para } i = 1,2,...,l, \tag{4.4}$$

$$g_i(w^*, b^*) = -y_i \left( w^* \cdot x_i + b^* \right) + 1 \leq 0 \text{ para } i = 1,2,...,l. \tag{4.5}$$

Así, para verificar si una SVM entrenada cumple con las condiciones KKT, es suficiente con verificar que todas las $\alpha_i$ satisfacen las ecuaciones (4.2) a (4.5).

Cuando termina un entrenamiento tenemos valores de $\alpha^*$ y, por consiguiente, de $w^*$ y $b^*$. Generalmente los valores del vector $\alpha^*$ encontrado cumplen con las condiciones establecidas por (4.2) y (4.3), pues son las condiciones del problema de Programación Cuadrática ya resuelto. Por lo tanto, nos resta verificar si para cada punto de entrenamiento los multiplicadores cumplen con (4.4) y (4.5). Para ello, hagamos $u(x_i) = w^* \cdot x_i + b^*$ y veamos a las ecuaciones (4.4) y (4.5) como

$$\alpha_i^* \left[ -y_i u(x_i) + 1 \right] = 0 \text{ para } i = 1,2,...,l, \tag{4.6}$$

$$-y_i u(x_i) + 1 \leq 0 \text{ para } i = 1,2,...,l. \tag{4.7}$$

De acuerdo a (4.4), hay dos factores que pueden hacer 0 dicha igualdad: $\alpha_i$ y $-y_i u(x_i) + 1$; y de acuerdo a (4.3), tenemos dos casos posibles para cada $x_i$ de un conjunto de entrenamiento: $\alpha_i^* = 0$ y $\alpha_i^* > 0$.

**Caso 1.** Cuando $\alpha_i^* = 0$, entonces $-y_i u(x_i) + 1$ puede o no ser distinto de 0. Pero como ese factor debe cumplir con (4.5), no puede ser positivo y, por lo tanto, al despejar adecuadamente tenemos que $y_i u(x_i) \geq 1$. El cero en el multiplicador nos dice que el vector de entrenamiento $x_i$ no es un vector de soporte y por lo tanto no cae en ninguno de los hiperplanos que acotan a éste; por lo tanto su valor no puede ser 1. Así, para toda $\alpha_i^* = 0$ debe cumplirse que $y_i u(x_i) > 1$.

**Caso 2.** Cuando $\alpha_i^* > 0$, entonces necesariamente $-y_i u(x_i) + 1 = 0$. De esa manera, el factor cumple con (4.5) y, por lo tanto, al despejar adecuadamente tenemos que $y_i u(x_i) = 1$. En este caso, el vector de entrenamiento $x_i$ sí es un vector de soporte. Así, para toda $\alpha_i^* > 0$ debe cumplirse que $y_i u(x_i) = 1$.

Es claro que cualesquiera vectores que no hayan tomado parte de un conjunto de entrenamiento no tienen un valor de α determinado, por lo cual para ellos no se toman en cuenta las condiciones (4.1) a (4.4). Sin embargo, cuando al utilizar el método de Chunking queremos probar si un vector de entrenamiento cumple con las condiciones KKT, asumimos valores de $\alpha = 0$ para los vectores que no fueron parte del pedazo.

Hemos visto que cierta cantidad de vectores de entrenamiento entran y salen del conjunto de trabajo, de acuerdo a si cumplen o no las condiciones descritas anteriormente. Generalmente, y de acuerdo a como haya sido elegido el primer conjunto de trabajo, nuestra primera región de decisión clasificará incorrectamente una cantidad proporcionalmente alta de vectores de entrenamiento. Conforme vayan sucediéndose las iteraciones, dicha proporción disminuirá hasta que en la última iteración haya cero intercambios entre el conjunto de trabajo y los vectores restantes.

## 4.1. El Algoritmo de Chunking Suave y su Implementación.

A continuación describiremos el algoritmo para el Método de Chunking Suave. Dicho método esta basado en el de Chunking, por lo que cada vez que entrena con un pedazo de los vectores de entrenamiento, considera como cero a los multiplicadores de Lagrange del resto. Esta implementación se hizo en un solo programa para ambos métodos, de modo que la única diferencia entre ellos es el rigor con el que se toman las condiciones KKT.

**Parámetros.** Es necesario determinar algunos parámetros del algoritmo. El primero de ellos es el tamaño del conjunto de trabajo inicial. Esto depende del tamaño del problema y del número esperado de vectores de soporte para ese problema. A continuación debemos saber en que cantidad se le permitirá crecer al conjunto de trabajo, cada vez que sea insuficiente, como se verá mas adelante.

**Selección del Conjunto de Trabajo Inicial.** Se construye el conjunto de trabajo de manera aleatoria, pero estableciendo un balance en el numero de vectores de cada clase; es decir, se van eligiendo uno de cada clase de manera intercalada hasta completar el conjunto de trabajo. Si acaso la cantidad de vectores de una clase es menor a la mitad del tamaño del conjunto de trabajo inicial, entonces todos ellos serán incluidos en él.

**Primera Fase (Suave).** En esta fase, lo que sigue se repetirá hasta que no existan vectores fuera del conjunto de trabajo que estén mal clasificados. Se resuelve el problema de Programación Cuadrática para los elementos del conjunto de trabajo, se extraen los vectores de soporte y se obtiene una región de decisión.

Se calculan los valores de $y_i u(x_i)$ para todos los vectores del conjunto de entrenamiento y se hacen dos listas, ordenadas de acuerdo a los valores recién calculados: una para los vectores que violan la condición de no estar bien clasificados (candidatos a entrar al conjunto de trabajo), y otra para los que no fueron vectores de soporte en el conjunto de trabajo (candidatos a salir).

Se determina un número de intercambios entre los dos conjuntos, de acuerdo a la menor de las cantidades de vectores entrantes y salientes. En caso de haber cambios posibles, estos se efectúan intercambiando al vector de afuera que mas viola la condición con el no vector de soporte de adentro que este mas alejado del margen, y así sucesivamente con los segundos, terceros, etc., de cada lista.

En caso de no haber candidatos a salir pero si a entrar, o no haber obtenido vectores de soporte, se incrementa el tamaño del conjunto de trabajo de acuerdo al parámetro ya

establecido, y esos lugares se ocupan con los vectores candidatos a entrar con mayor violación a la condición.

Si hubieron cambios o un incremento para el conjunto de trabajo, se pasa a la siguiente iteración con el nuevo conjunto de trabajo. En caso de no haber candidatos a entrar, no importa si hay o no candidatos a salir, la fase iterativa termina.

**Segunda Fase (Iteración Adicional).** Se calculan los valores de $y_i u(x_i)$ para todos los vectores del conjunto de entrenamiento que no estaban en el conjunto de trabajo, y se hace una lista para los vectores que quedan dentro del margen de la última región de decisión calculada. Se reúnen estos datos con los vectores de soporte en el conjunto de trabajo para formar otro conjunto de trabajo. Finalmente, se obtiene la región de decisión para este otro conjunto de trabajo.

# 5. Resultados Experimentales.

Se ha probado este nuevo método utilizando la base de datos de caracteres manuscritos MNIST [4] que consta de imágenes de $28 \times 28$, lo que resulta en vectores de 784 elementos enteros de 0 a 255; es decir, nuestro espacio de entrada es $R^{784}$. Los experimentos aquí mostrados fueron hechos para el problema de separar el dígito 0 de los demás. El Experimento 1 corresponde a 250 datos de entrenamiento, y el Experimento 2 500 datos. Se ha calculado la generalización para 4500 datos de prueba en ambos casos.

## 5.1 Experimento 1.

Primeramente veremos el caso del entrenamiento de SVM utilizando 25 muestras de cada dígito, lo que nos da 250 patrones de entrenamiento. Se encontraron las regiones de decisión para cada enfoque y se tomaron algunos datos resultantes. Las tablas 5.1 a 5.3 nos muestran en forma comparativa dichos resultados. Cabe señalar que el tiempo que aparece en el renglón de la iteración adicional del método de Chunking Suave es exclusivo de la iteración, efectuada al final de la primera parte del método.

Cuando se menciona a la función QuadProg, nos referimos a que se entrenó una SVM con todos los vectores utilizando dicha función. Cabe destacar que para entrenar las SVM cada iteración de los métodos de descomposición se utilizó también QuadProg.

Resulta interesante observar las tendencias en el número de vectores de soporte que se obtienen en cada método. Al utilizar QuadProg y tener a todo el conjunto de entrenamiento, se obtiene un porcentaje alto de vectores de soporte, mientras que con los métodos de Chunking (original y suave), esas cantidades disminuyen. Podemos observar que la iteración adicional aproxima esta cantidad a la de la función QuadProg, debido a que toma una buena cantidad de vectores que quedaron dentro del margen del Chunking Suave. Esto se debe a que el tamaño del margen es muy pequeño, como se discutirá mas adelante. Otra situación es que el número de vectores de soporte aumenta conforme aumenta el grado del kernel polinomial y los porcentajes de estos sobre el total de vectores de entrenamiento aumenta de maneras similares.

No solamente la velocidad de entrenamiento y el número de vectores de soporte encontrados son factores a favor del Chunking Suave, sino la capacidad de generalización que se obtiene resulta ser casi la misma. Incluso al efectuar la iteración adicional lo que gana no es significativo. Puede decirse que la región de decisión del Chunking Suave

es tan buena como las de las SVM convencionales, con la gran ventaja de la rapidez con que se obtienen.

| *Algoritmo* | *Número de Iteraciones* | *% Correctos Entrenamiento* | *% Correctos Prueba* | *Tiempo (Segundos)* | *Núm. VS (%)* |
|---|---|---|---|---|---|
| QuadProg | 1 | 98.80 | 95.13 | 32.95 | 179 (71.60%) |
| Chunking | 4 | 98.80 | 95.13 | 51.11 | 114 (45.60%) |
| Ch. Suave | 2 | 98.80 | 95.22 | 23.23 | 99 (39.60%) |
| I. Adicional | 1 | 99.00 | 95.20 | 17.52 | 165 (66.00%) |

*Tabla 5.1. Resultados para un kernel polinomial de grado 4 con 250 muestras.*

| *Algoritmo* | *Número de Iteraciones* | *% Correctos Entrenamiento* | *% Correctos Prueba* | *Tiempo (Segundos)* | *Núm. VS (%)* |
|---|---|---|---|---|---|
| QuadProg | 1 | 98.60 | 94.58 | 31.09 | 190 (76.00%) |
| Chunking | 9 | 98.60 | 94.64 | 135.14 | 130 (52.00%) |
| Ch. Suave | 2 | 98.40 | 94.71 | 22.81 | 103 (41.20%) |
| I. Adicional | 1 | 98.60 | 94.64 | 21.27 | 178 (71.20%) |

*Tabla 5.2. Resultados para un kernel polinomial de grado 5 con 250 muestras.*

| *Algoritmo* | *Número de Iteraciones* | *% Correctos Entrenamiento* | *% Correctos Prueba* | *Tiempo (Segundos)* | *Núm. VS (%)* |
|---|---|---|---|---|---|
| QuadProg | 1 | 98.40 | 94.07 | 29.30 | 191 (76.40%) |
| Chunking | 8 | 98.40 | 94.11 | 222.27 | 146 (58.40%) |
| Ch. Suave | 2 | 98.40 | 94.13 | 22.78 | 112 (44.80%) |
| I. Adicional | 1 | 98.40 | 94.11 | 18.86 | 182 (72.80%) |

*Tabla 5.3. Resultados para un kernel polinomial de grado 6 con 250 muestras*

## 5.2 Experimento 2.

Ahora nos ocuparemos del caso del entrenamiento de SVM utilizando 50 muestras de cada dígito, lo que nos da 500 patrones de entrenamiento. Como en el ejemplo anterior, se encontraron las regiones de decisión para cada enfoque y se tomaron algunos datos resultantes. Las tablas 5.4 a 5.6 nos muestran en forma comparativa dichos resultados.

| *Algoritmo* | *Número de Iteraciones* | *% Correctos Entrenamiento* | *% Correctos Prueba* | *Tiempo (Segundos)* | *Núm. VS (%)* |
|---|---|---|---|---|---|
| QuadProg | 1 | 100.00 | 96.36 | 350.11 | 330 (66.00%) |
| Chunking | 3 | 100.00 | 96.36 | 256.26 | 203 (40.60%) |
| Ch. Suave | 3 | 100.00 | 96.51 | 229.38 | 181 (36.20%) |
| I. Adicional | 1 | 100.00 | 96.51 | 192.62 | 261 (52.20%) |

*Tabla 5.4. Resultados para un kernel polinomial de grado 4 con 500 muestras.*

| *Algoritmo* | *Número de Iteraciones* | *% Correctos Entrenamiento* | *% Correctos Prueba* | *Tiempo (Segundos)* | *Núm. VS (%)* |
|---|---|---|---|---|---|
| QuadProg | 1 | 100.00 | 95.73 | 365.80 | 333 (66.60%) |
| Chunking | 7 | 100.00 | 95.87 | 546.30 | 206 (41.20%) |
| Ch. Suave | 3 | 100.00 | 95.80 | 223.44 | 188 (37.60%) |
| I. Adicional | 1 | 100.00 | 95.93 | 156.33 | 280 (56.00%) |

*Tabla 5.5. Resultados para un kernel polinomial de grado 5 con 500 muestras.*

| *Algoritmo* | *Número de Iteraciones* | *% Correctos Entrenamiento* | *% Correctos Prueba* | *Tiempo (Segundos)* | *Núm. VS (%)* |
|---|---|---|---|---|---|
| QuadProg | 1 | 100.00 | 95.29 | 372.72 | 361 (72.20%) |
| Chunking | 7 | 100.00 | 95.44 | 591.83 | 241 (48.20%) |
| Ch. Suave | 3 | 100.00 | 95.00 | 202.91 | 181 (36.20%) |
| I. Adicional | 1 | 100.00 | 95.44 | 137.37 | 308 (61.60%) |

*Tabla 5.6. Resultados para un kernel polinomial de grado 6 con 500 muestras.*

Ahora la diferencia en velocidades se ha acentuado pero el porcentaje de ahorro sigue siendo superior al 50%. Aún si sumamos el tiempo de la iteración adicional, la ventaja sigue siendo del Chunking Suave. Las diferencias con el total, aún con la iteración adicional son mucho menores en porcentaje y en ocasiones gana un método y en otras el otro. La capacidad de generalización, como era de esperarse, ha mejorado debido a la cantidad de vectores de entrenamiento.

# Conclusiones.

Para terminar explicaremos porqué el Chunking Suave es tan "bueno" para generalizar como el Chunking original. En esta clase de problemas el tamaño del margen ínfimo y ciertamente despreciable, como puede verse en las Tabla 6.1.

Concluimos que la simple región de decisión (correctamente ubicada como en nuestro caso) es suficientemente buena para darnos una generalización adecuada, de la misma forma en que lo haría una SVM entrenada con el método de Chunking o con la función QuadProg. La gran diferencia es que el Chunking Suave requiere de un menor esfuerzo computacional para hacer válidas todas las restricciones del problema, que los otros dos métodos de entrenamiento. La consecuencia de esto es una mayor velocidad a la hora de encontrar la región de decisión. Por lo tanto, si podemos obtener una región de decisión similarmente buena y en un tiempo menor, tenemos un método mucho mejor, sobre todo se utiliza para construir SVMs con grandes cantidades de datos de entrenamiento.

| *Kernel* | Grado 4 | Grado 5 | Grado 6 |
|---|---|---|---|
| *Margen con 250 Patrones* | $5.0501\times10^{-26}$ | $1.3834\times10^{-32}$ | $3.9758\times10^{-39}$ |
| *Margen con 500 Patrones* | $1.485\times10^{-25}$ | $4.2029\times10^{-32}$ | $1.2878\times10^{-38}$ |

*Tabla 6.1. Anchos de los márgenes de seis SVM.*

Hemos descrito el Chunking Suave como un método muy veloz para entrenar SVMs. Otra aportación importante de este trabajo es el reconocimiento de que en muchos problemas que requieren de kerneles, el tamaño del márgen es ínfimo. Aunque no hemos probado con otros problemas, por analogía la gran mayoría de los SVMs que clasifican esta clase de datos a través de kerneles van a tener un ancho de márgen despreciable. De esta forma, una región de decisión que se encuentra dentro de un márgen óptimo tan angosto es equivalente al mismo óptimo. Con la ventaja de menor tiempo de cálculo, menor número de vectores de soporte y similar generalización que otorga, el Chunking Suave supera a los algoritmos con los que fue comparado.

## Agradecimientos.

El segundo autor reconoce ayuda parcial para este trabajo a través del proyecto CONACyT No. I-39324-A.

# Referencias.

[1] Christopher C. Burges (1998). *A Tutorial on Support Vector Machines for Pattern Recognition*, Data Mining and Knowledge Discovery, Kluwer Academis Publishers.

[2] Philip E. Gill, Walter Murray, Margaret H. Wright (1991). *Numerical Linear Algebra and Optimization, Volume 1*, Addison-Wesley Publishing Company.

[3] Steve Gunn (1998). *Support Vector Machines for Classification and Regression*, ISIS Technical Report, Image Speech and Intelligent Systems Group, University of Southampton.

[4] Yann Le Cun (1999). *The MNIST Database of Handwritten Digits*, Dirección de Internet: http://yann.lecun.com/exdb/mnist/.

[5] Edgar Osuna, Robert Freund, Federico Girosi (1997). *An Improved Training Algorithm for Support Vector Machines*, Proceedings of IEEE Workshop on Neural Networks for Signal Procesing.

[6] Ricardo del A. Pérez Flores (2002). *Entrenamiento de Máquinas de Soporte Vectorial y su Aplicación en el Reconocimiento de Patrones*, Tesis de Maestría, CIMAT, A.C.

[7] John C. Platt (1998). *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*, Technical Report MSR-TR-98-14, Microsoft Research.

[8] Vladimir Vapnik (1995). *The Nature of Statistical Learning Theory*, Springer-Verlag.

# Teoría para Reconocimiento de Patrones Colaborativo

Dr. Argelio Victor de la Cruz Rivera[1], M. en C. María de los Ángeles Alonso Lavernia[1]

[1] Centro de Investigación en Tecnologías de Información y Sistemas
Universidad Autónoma del Estado de Hidalgo (CITIS-UAEH)
argelioc@uaeh.reduaeh.mx, marial@uaeh.reduaeh.mx

## Resumen

Se presentan los resultados preliminares obtenidos en el desarrollo de una teoría que permite la integración de varios métodos de Reconocimiento de Patrones Supervisado.

La solución propuesta consiste, en construir una Base de Conocimientos a partir de los datos contenidos en la Información de Aprendizaje, la cual describe las características de las clases existentes. Dado que esta transformación se ejecuta utilizando una estructura semejante a la de un sistema basado en conocimientos, este resultado permite construir sistemas que se nutran con experiencias humanas y datos provenientes del problema, además de posibilitar el uso colaborativo de diversos clasificadores.

Para realizar lo anterior, se desarrolla un procedimiento de descomposición de la información además de adecuaciones a varios métodos, para permitir su utilización bajo los mismos criterios. Algunos de estos métodos son: la regla de los k vecinos más cercanos, la regla del análisis global por grupo, la regla del análisis distinguido, árboles de decisión, etc.

**Palabras Claves:** Reconocimiento Supervisado, Base de Conocimientos, Clasificación Colaborativa.

## 1. Introducción

Como es conocido, los métodos de Reconocimiento de Patrones pueden ser divididos en dos grupos fundamentales: Reconocimiento Supervisado y Reconocimiento No Supervisado, aunque algunos autores también incluyen problemas de selección de variables y variantes de los anteriores [17], [19].

En ambos casos, se parte de suponer que se tiene una matriz de datos, que denominaremos M, la cual está formada por "n" filas y "m" columnas, donde las filas representan los individuos bajo estudio (también llamados objetos o entes, que denotaremos como $I_i$) y las columnas los parámetros o variables que han sido medidos a esos individuos (que se denotarán como $V_j$).

Los métodos de Reconocimiento No Supervisado tratan de resolver el problema de encontrar una clasificación adecuada para los individuos en Nc clases o grupos, mientras que los métodos de Reconocimiento Supervisado, parten de suponer que los individuos de la matriz M, ya se encuentran clasificados en Nc clases y el problema a resolver consiste en: clasificar otro conjunto de individuos, cuya pertenencia a las clases de M es desconocida. Es común, denominar como Matriz de Aprendizaje o Información de Aprendizaje a la matriz M y denominar como patrones a los individuos [6], [7], [8], [17].

Son diversos los trabajos que se han realizado en esta disciplina, sin embargo, continúan presente diversos problemas a la hora de su utilización práctica:

1. La aplicación de varios métodos para un mismo problema, puede provocar aparición de varias soluciones y es muy difícil encontrar una vía formalizada para tomar una decisión con todos esos resultados.
2. Ante un problema real no es posible determinar, a priori, cuál o cuáles métodos utilizar para garantizar que los resultados sean correctos.

En Reconocimiento de Patrones, la idea de combinar la decisión de varios clasificadores ha sido explorada con vistas a resolver estos problemas. Diversos métodos han sido usados para combinar los resultados de distintas redes neuronales [5], [16], [19], [21], utilizan árboles de decisión [2], [3] u otros modelos [1], [4], [12], [14].

En la mayoría de los casos se trata la combinación de modelos derivados de múltiples versiones del mismo método de aprendizaje o reconocimiento y por lo general se parte los resultados brindados por cada uno de ellos de forma independiente, para buscar una manera de combinarlos y obtener un único resultado [15], [18], [22].

Las razones para intentar combinar los resultados de múltiples clasificadores son fuertes, puesto que cada método puede representar aspectos diferentes de un problema de los datos de entrada como tal.

En el presente trabajo, se proponen soluciones a la utilización combinada clasificadores supervisados, pero mediante la integración de los métodos a arquitectura única, en lugar de la integración de sus resultados como tal.

Sin embargo, los resultados obtenidos no se centran únicamente en esta dirección, que la aproximación que se propone, permite el diseño de sistemas capaces de abordar la solución de problemas complejos de diagnóstico, pronóstico o toma de decisiones general, mediante la combinación de datos provenientes de una Información Aprendizaje y conocimientos provenientes de expertos humanos en el tema de forma colaborada.

# 2. Uso Combinado de Datos y Conocimientos

En la mayoría de los problemas reales de clasificación, diagnóstico, pronóstico o toma de decisiones en general, se pueden identificar dos fuentes de información fundamentales, que pueden ser usadas para buscar las soluciones:

- Casos conocidos representados a través de datos
- Experiencias Humanas

No se puede asegurar que una u otra información sea más importante, puesto que depende de cada caso en particular, y ambas pueden ser importantes fundamentales.

Para poder construir un sistema que sea capaz de manejar ambas fuentes de información es necesario encontrar formas de unificación de sus estructuras de representación.

La solución propuesta en este trabajo, consiste en la creación de nuevas estructuras migración de los datos hacia una estructura de Base de Conocimientos. Con esto se logra permitir la combinación de clasificadores, y se brinda una herramienta que permita el combinado de datos y conocimientos mediante estructuras híbridas [20].

En resumen, se brindan soluciones generales a problemas como los siguientes:

- Se posibilita el uso de diferentes criterios para Reconocimiento de Patrones de forma tal, que los resultados que se brinden sean obtenidos producto de la combinación los diversos criterios empleados. Cada método puede tener un aporte diferente resultado final que se brinda.

- Se permite el uso combinado de experiencias (conocimientos humanos) y casos conocidos (datos) para la solución de un mismo problema.
- Se brinda una herramienta computacional que permite el diseño de sistemas interactivos para Reconocimiento de Patrones, con posibilidades para justificar su comportamiento y los resultados que brinde [9], [10], [11].

## Base de Conocimientos de la Información de Aprendizaje

En la teoría que será tomada como base [9], [10], [11], el término Base de Conocimientos (BC) se interpreta, hablando de forma global (no se pretende construir una definición formal) como la unión de dos componentes fundamentales:

1) Un conjunto formado por *hechos o conceptos* provenientes de un dominio del conocimiento específico.
2) Un conjunto de *relaciones* entre los elementos del conjunto anterior.

Una Forma de Representación de Conocimientos será considerada como toda aquella estructura teórica que permita expresar de alguna forma, los elementos de alguno de los conjuntos anteriores y a su vez posibilite procesar la información que ellos representada.

El trabajo que se presenta se enfoca hacia los problemas de Reconocimiento Supervisado, por tanto, la información de partida será la Matriz de Aprendizaje de algún problema particular y el objetivo es construir una BC, que describa las características de cada clase de esta información por separado.

## Construcción de Conceptos

La vía más simple que utilizaremos para representar los elementos del primer conjunto una BC, es a través de las llamadas **Proposiciones**.

Una Proposición constituye una estructura con diferentes componentes, que permiten representar conceptos de diversa forma [9], [10], [11] pero sólo analizaremos los aspectos generales relacionados con la utilización que se le dará en el presente trabajo.

Una situación del mundo real, se puede representar en la computadora mediante una proposición, si a ésta se le puede asociar un valor de seguridad o certidumbre.

Las componentes básicas que utilizaremos para representar una proposición serán: textos identificadores, para expresar en lenguaje natural la situación que se desea representar, valor de certidumbre asociado y atributos naturales.

El valor de certidumbre C(Pj) asociado a una proposición denotada como Pj, es un número del intervalo [-1,1], donde 1 indica *verdad absoluta*, -1 falsedad absoluta, (lo contrario de lo anterior), y 0 *desconocimiento total*, es decir, que no se sabe. El resto de los valores se encuentran asociados a gradaciones de la veracidad.

Los atributos naturales pueden ser: pregunta, objetivo o intermedio, pero estos quedan definidos una vez que se introduce el segundo conjunto. Las proposiciones pregunta representan información conocida sobre el problema que debe ser suministrada como dato entrada. Los objetivos constituyen los resultados que se brindarán, y los intermedios representan subobjetivos dentro del problema que también deberán ser obtenidos.

### 4.1. Construcción de Proposiciones Objetivo

Como los problemas de Reconocimiento Supervisado se enfocan a clasificar un objeto una o varias clases entre las existentes en la Matriz de Aprendizaje, entonces se pueden definir las proposiciones objetivo como se expresa a continuación:

Si suponemos que existen Nc clases reconocidas en la matriz M, entonces se puede definir el texto general de las proposiciones objetivo como sigue:

"El individuo o patrón a reconocer, denotado como $I_i$, se clasifica en la clase $l$ perteneciente a la Matriz de Aprendizaje", donde $I_i$ representa un individuo caracterizado por el mismo conjunto de variables que los miembros de la Matriz de Aprendizaje, pero que no se encuentra en ella (se denotará $I_i \notin M$) y $l \in \{1, 2, \ldots, Nc\}$.

Si denotamos además, como O$l$ a la proposición objetivo $l$, entonces los valores de certidumbre asociados a O$l$ se interpretarían como sigue: si C(O$l$) > 0 ello indicará que existen evidencias a favor de la clasificación o pertenencia del individuo $I_i$ a la clase claro está que mientras mayor sea este valor mayor seguridad habrá al respecto, pero por el contrario, C(O$l$) < 0 entonces ello indica que existen razones para considerar que no pertenece a la clase $l$ y C(O$l$) = 0 indica que no se sabe nada acerca de la pertenencia del individuo a la clase.

Se puede entonces plantear que la primera tarea de un sistema utilizando la teoría propuesta, consistirá en lo siguiente:

"*Dado una Matriz de Aprendizaje M y un individuo $I_i \notin M$, obtener los grados certidumbre asociados a las Nc proposiciones objetivo derivadas de M, que expresan seguridad que se tiene sobre la pertenencia de $I_i$ a cada clase*".

De lo anterior se evidencia la necesidad de definir como ejecutar este proceso y para ello establecer como relacionar la información de partida con los resultados expresados.

Por otra parte, no es evidente cuáles proposiciones preguntas o intermedios se generan, puesto que ello va a depender de las relaciones (elementos del segundo conjunto en BC) que se establezcan, de los métodos utilizados y de la naturaleza de los datos.

# 5. Construcción de Relaciones

Las relaciones entre los conceptos de una BC constituyen la vía utilizada para ejecutar los procesos de inferencia que permiten dar solución al problema planteado.

En nuestro caso, estas relaciones se conciben como los diferentes métodos reconocimiento a ser aplicados, pero dado los objetivos y extensión del presente trabajo nos concentraremos en estudiar sólo algunos casos generales, que permiten construir arquitectura sobre la que se pueden incorporar otros métodos o variantes.

## 5.1. Método de Descomposición de la Información

Como su nombre lo indica este método basa su funcionamiento en buscar descomposición de la información para que el proceso de clasificación se ejecute pasos. Comenzaremos el análisis planteando algunas definiciones para las variables numéricas.

### *5.1.1. Proposiciones de Tipo 1*

**Definición 1.-** Una variable numérica j para el individuo $I_i$ con valor $V_{ij}$ es semejante variable j de la clase $l$ si cumple que se encuentra dentro del intervalo [Minj$l$ , Maxj$^n$ donde Minj$l$ denota al mínimo de la variable j para la clase $l$ y Maxj$l$ el máximo variable j para la clase $l$.

Si denotamos como $m_n$ a la cantidad de variables numéricas de la matriz, entonc utilizando esta definición se puede construir un conjunto de proposiciones pregunt precisamente con $m_n$ elementos de la siguiente forma:

**Proposición general Tipo1**: La variable numérica j para el individuo $I_i$ es semejante a variable j de la clase *l*. Se denota $PTipo1'_{ji}$.

Como se puede observar, se obtienen $m_n$ proposiciones con atributo pregunta, de este planteamiento general.

### *1.1.1. Cálculo de la certidumbre asociada a las proposiciones Tipo 1*

Denotaremos como $C(PTipo1'_{ji})$ la certidumbre asociada a las proposiciones Tipo1 y consideraremos que la zona alrededor de los valores máximo y mínimo la variable pueden ser dudosos, para lo cual se introduce un Factor F ∈ [0,1] que permitirá definir una función, para cada caso particular, en la que se represente la parte típica del intervalo, de la siguiente forma:

Si F = 0 ello indica que $C(PTipo1'_{ji}) = 1$ sólo para $V_{ij} = (Minj_l + Maxj_l) / 2$, es decir, cuando el valor de la variable j para el individuo $I_i$ ($V_{ij}$) este exactamente en el centro del intervalo.

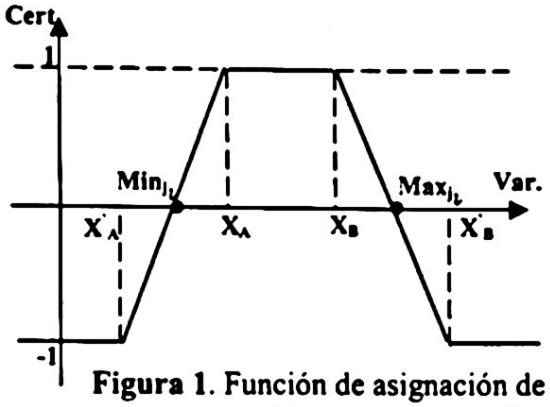



**Figura 1**. Función de asignación de certidumbre para F=1/2.

Para F = 1 se indica que si $V_{ij} \in [Minj_l, Maxj_l]$ entonces $C(PTipo1'_{ji}) = 1$ y si $\notin [Minj_l, Maxj_l]$ entonces $C(PTipo1'_{ji}) = -1$ .

Los valores entre 0 y 1 son situaciones intermedias que permiten ampliar o disminuir la zona de dudas alrededor del mínimo y máximo. En la figura 1 se muestra un diagrama de función para un caso donde F = 1/2.

### *5.1.2. Reglas de Tipo 1*

A partir de las proposiciones Tipo1 se obtiene información sobre la pertenencia del individuo a cada clase para una variable en particular como sigue:

**Definición 2.-** Si la variable numérica j para el individuo $I_i$, es semejante a la variable j de la clase *l* con certidumbre 1 ($C(PTipo1'_{ji}) = 1$) entonces el individuo $I_i$ pertenece a la clase *l* para la variable j con valor de certidumbre: $W_l = \dfrac{Nc - 1 - n_{jk}}{Nc - 1 + n_{jk}}$ donde Nc denota la cantidad de clases y $n_{jk}$ la cantidad de clases para las cuales $V_{ij} \in [Minjk , Maxjk]$, $k \in \{1,2,\ldots,Nc\} / \{l\}$.

También se puede considerar que si la variable numérica j para el individuo $I_i$, no es semejante a la variable j de la clase *l* con certidumbre 1 ($C(PTipo1'_{ji}) = -1$) entonces constituye una proposición verdadera que el individuo $I_i$ no pertenece a la clase *l* para j.

Se puede construir entonces, un conjunto de proposiciones de Tipo2 como sigue:

**Proposición general Tipo2**: El individuo $I_i$ pertenece a la clase *l* para la variable numérica j. Se denota $PTipo2'_{ij}$.

Y como se observa en la definición 2 existe una relación entre las proposiciones Tipo1 las Tipo2, que puede ser expresada mediante una Regla de Producción como sigue:

Regla Tipo 1 (RT1): $PTipo1'_{ji} \Rightarrow PTipo2'_{ij}(W_l, -1)$;

Está regla significa lo siguiente: Si se obtiene con completa seguridad que la variable numérica j para el individuo $I_i$ es semejante a la variable j de la clase *l* ($C(PTipo1'_{ji}) = 1$)

entonces esto contribuye con $W_l$ a la seguridad que se tiene sobre la pertenencia del individuo $I_i$ a la clase *l* para la variable numérica j, pero si esta premisa no se satisface con completa seguridad ($C(PTipo1'_{ji}) = -1$) entonces contribuye con -1.

Claro está, que si la premisa toma algún valor no extremal entonces la contribución debe ser re-calculada. Esto puede ser obtenido de diversas formas, pero una vía inmediata que garantiza obtener valores proporcionales al incremento o decremento de los valores asociados a la premisa es la siguiente: si $C(PTipo1'_{ji})$ denota la certidumbre asociada a $PTipo1'_{ji}$, y ésta es positiva, entonces la contribución será $C(PTipo1'_{ji})^* W_l$ y si es negativa entonces contribuirá con $-C(PTipo1'_{ji})$.

Como se puede observar, las proposiciones de Tipo2 son intermedios ya que no representan datos de entrada ni los objetivos definidos en el epígrafe 4.1. Veamos entonces como construir las reglas que permiten obtener los objetivos.

### *5.1.3. Reglas de Tipo 2*

**Definición 3.-** Si el individuo $I_i$ pertenece a la clase *l* para la variable numérica j con completa seguridad ($C(PTipo2'_{ij}) = 1$) entonces esto contribuye en $C(PTipo2'_{ij})/m$ (m denota la cantidad de variables) a la certidumbre global de la pertenencia de este individuo $I_i$ a la clase *l*. $C(PTipo2'_{ij})$ denota la certidumbre de las proposiciones Tipo2.

Como se puede observar aquí se plantea una relación entre las proposiciones de Tipo2 y los objetivos, que se denominarán de Tipo3 (se denotarán $PTipo3'_i$), lo cual puede ser expresado a través de la siguiente regla general:

Regla Tipo 2 (RT2): $PTipo2'_{ij} \Rightarrow PTipo3'_i$ (1/m , -1/m); lo cual significa lo siguiente:

Si se obtiene con completa seguridad que el individuo $I_i$ pertenece a la clase *l* para la variable numérica j ($C(PTipo2'_{ij}) = 1$) entonces esto contribuye en 1/m a la seguridad que se tiene sobre la pertenencia de $I_i$ a la clase *l*, y si por el contrario se obtiene que $C(PTipo2'_{ij}) = -1$, entonces esto contribuye con -1/m.

Se puede observar que este criterio define aportes semejantes para todas las variables a la clasificación, sin embargo, es común en los problemas reales que unas variables sean más importantes que otras, de aquí que de ser conocida la importancia o peso de las variables, se puede utilizar esta información para ponderar las contribuciones.

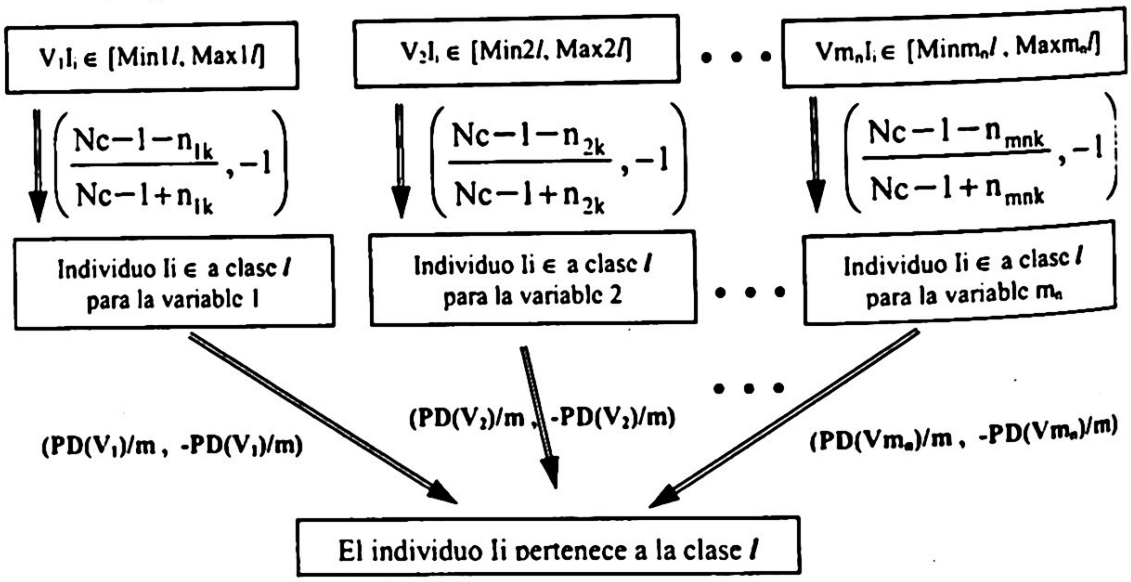



**Figura 2.-** Estructura de reglas para el Método de Descomposición de la Información

Si denotamos como $PD(V_j) \in [0,1]$ al peso de la variable $V_j$ entonces la Regla Tipo 2 se puede expresar de la siguiente forma: $PTipo2'_{ij} \Rightarrow PTipo3'_i (PD(V_j)/m, -PD(V_j)/m)$;

Si analizamos las estructuras que se obtienen para las proposiciones y reglas que se han generado hasta este punto, se obtiene un esquema como el representado en la figura 2, donde se observa la estructura obtenida para un objetivo.

### *5.1.4. Análisis para Variables No Numéricas*

**Definición 4.-** Una variable no numérica j (cualitativa o binaria) para el individuo $I_i$ con valor $V_{ij}$ pertenece a la clase *l* si $V_{ij} = X_{kj}$, donde $X_{kj}$ es el valor del individuo k para la variable j y $k \in l$.

Como se observa, el criterio planteado considera que si se encuentra algún individuo en clase *l* tal que su valor para la variable j coincide con el valor correspondiente de la misma variable para $I_i$, entonces esto constituirá un elemento favorable con relación a la clasificación en esta clase.

Se puede entonces construir una nueva proposición de Tipo4 como sigue:

**Proposición general Tipo4**: La variable no numérica j para el individuo $I_i$ pertenece a clase *l*. Se denota $PTipo4'_{ji}$.

Y a partir de ella definir la siguiente regla para la contribución a la proposición Tipo2:

Regla Tipo 3 (RT3): $PTipo4'_{ji} \Rightarrow PTipo2'_{ij} (W_l, -1)$;

Existen varias formas de definir la contribución $W_l$ en este tipo de regla para el caso de variables no numéricas; dos posibilidades son las siguientes:

Caso 1.- $W_l = 1$. Se considera que al existir un individuo en la clase con valor igual al de $I_i$ para j, esto asegura la pertenencia de $I_i$ para la variable j a la clase *l*.

Caso 2.- $W_l = \dfrac{NE_{lji}}{NE_l}$ donde $NE_{lji}$ denota la cantidad de individuos de la clase *l* cuyos valores para la variable j son iguales al del individuo a clasificar $I_i$ ($V_{ij}$) mientras que $NE_l$ denota la cantidad de individuos de la clase *l*. En este caso se considera como importante la cantidad de individuos con este valor para la variable j.

## 5.2. Criterios de los Individuos más Semejantes

El método de los k-vecinos más cercanos se encuentra entre los más conocidos y usados Reconocimiento Supervisado de Patrones [6], [13]. En nuestro caso hemos considerado varios criterios que permiten incorporar a la estructura anterior las ideas de este método.

Consideraremos el conjunto $I_{knn} = \{X_1, X_2, \ldots, X_{knn}\}$ de los knn patrones más semejantes a $I_i$ (Individuo a clasificar) dentro de la Matriz de Aprendizaje M ($1 \leq knn \leq n$, un parámetro suministrado por el usuario y $X_k \in M$, $k=1,\ldots,knn$) y supondremos que este conjunto está ordenado de acuerdo a su semejanza con $I_i$ de mayor a menor, es decir, se parece más a $I_i$ que $X_2$ y $X_2$ más que $X_3$ y así sucesivamente.

Criterio 1.- Si $X_k \in I_{knn}$, $X_k \in l$ y $SX_kI_i \in [0,1]$ denota la semejanza o parecido entre $X_k$ e $I_i$ entonces, esto contribuye en $SX_kI_i$ a la pertenencia de $I_i$ a la clase *l*.

partir de este criterio se puede construir una regla considerando las proposiciones:

PTipo5: El individuo $X_k \in I_{knn}$.

PTipo6: El individuo $X_k \in l$.

Regla Tipo 4 (RT4): $PTipo5 \;\&\; PTipo6 \Rightarrow PTipo3'_i (SX_kI_i, 0)$;

Esta regla se interpreta como sigue: Si se cumple que el individuo $X_k$ perteneciente a $M_l$ se encuentra entre los knn más semejantes a $I_i$ y $X_k$ es miembro de la clase $l$, entonces esto contribuye con el valor de la semejanza $SX_kI_i$, a la pertenencia de $I_i$ a la clase $l$.

Se puede observar, que no todas las clases recibirán contribuciones producto de esta regla, dado que puede ocurrir que ninguno de sus elementos este en el conjunto $I_{knn}$, lo cual pudiera ser considerado como evidencia negativa en lugar de 0.

Criterio 2.- Si $X_k \in I_{knn}$ y $X_k \in l$ entonces esto contribuye en CertI – CertI(k-1)/knn a la pertenencia de $I_i$ a la clase $l$, donde CertI es un parámetro externo que denota un valor de certidumbre inicial que permite controlar la influencia de este criterio.

Ejemplo: Si suponemos que knn = 3 y CertI = 0.8 entonces $X_1$ contribuye en 0.8, $X_2$ contribuye en 0.53 y $X_3$ contribuye en 0.27 lo cual implica una contribución menor según exista un menor parecido con $I_i$.

De la misma forma que en el caso anterior, se construyen proposiciones y una regla general para este criterio (RT5).

Criterio 3.- El usuario puede suministrar como datos las contribuciones de cada uno de los integrantes del conjunto $I_{knn}$ (RT6).

Criterio 4.- Utilizar los mismos criterios anteriores, pero obteniendo conjuntos de individuos más semejantes a cada clase en lugar de a la matriz de aprendizaje completa. En este caso se tendrían reglas (RT7) para cada clase en particular.

## 5.3. Criterios del Análisis Global por Grupos

La Regla del Análisis Global por Grupos surge como una variante a la regla de clasificación clásica de los k vecinos más cercanos, en la cual se considera el cálculo de semejanzas con toda la clase en general, en lugar de obtener semejanzas individuales [8].

La semejanza entre un individuo $I_i$ a clasificar y una clase $l$ de la Información de Aprendizaje, según esta regla, se obtiene de la siguiente forma: $SI_i l = \sum_{i=1}^{m} 1 - \frac{1}{1 + dI_i^j l_j}$

donde $dI_i^j l_j$ representa una semejanza entre el valor de la variable j de $I_i$ y los valores de esta variable, para todos los integrantes de la clase $l$, para lo cual también se proponen distintas variantes [8].

Lo interesante de esta regla de clasificación consiste, en que se demuestra que la expresión de semejanza entre un individuo y una clase ($SI_i l$) cumple con los axiomas de una distancia clásica.

Desde el punto de vista del presente trabajo la idea consiste en permitir la integración de este procedimiento a la estructura anterior, lo cual se puede ejecutar de varias maneras. Algunos criterios son los siguientes:

Criterio 1.- La semejanza entre $I_i$ y una clase $l$ contribuye en $1/(1+SI_i l)$ a la pertenencia de $I_i$ a la clase $l$. La regla correspondiente a este criterio se expresa como: $0 \Rightarrow PTipo3_i^l$ $(1/(1+SI_i l), 0)$; lo cual indica que bajo cualquier condición existe una contribución de $1/(1+SI_i l)$ a la pertenencia de $I_i$ a la clase $l$.

Criterio 2.- Se puede definir que solamente las clases de mayor semejanza reciban contribuciones positivas, mientras que el resto reciban contribuciones negativas.

Esto es, si la clase *l* se encuentra entre las k*l* de mayor semejanza al individuo $I_i$ (k*l* es un parámetro externo) entonces existirá una contribución favorable a la clasificación proporcional a la semejanza y en caso contrario una contribución negativa.
Si definimos una proposición como sigue:
PTipo7: La clase *l* se encuentra entre las k*l* de mayor semejanza al individuo $I_i$.
Se obtiene la Regla Tipo 8 (RT8): $PTipo7 \Rightarrow PTipo3^{l}_{i}\ (1/(1+S_{I_i}l),\ 1 - 1/(1+S_{I_i}l))$;
Existen otras variantes definidas en el algoritmo original donde se utiliza para el cálculo las semejanzas individuo-clases, solamente aquellos integrantes de la clase que sean suficientemente parecidos de forma individual al patrón que se desea clasificar. Esta variante se denomina Análisis Global Distinguido [8] y también se pueden incorporar a las estructuras anteriores utilizando las mismas ideas presentadas.

# 6. Clasificación Colaborativa

Como se puede deducir del estudio de los epígrafes anteriores, la idea que se presenta consiste en construir una BC constituida por un conjunto de Proposiciones y un conjunto de Reglas de Producción, donde estas últimas son la representación de diversos métodos de clasificación supervisada.

El usuario del sistema que implementa esta teoría, tendrá la posibilidad de escoger los criterios o métodos a utilizar, que pueden ser uno o varios, de forma tal que el resultado de clasificación será la combinación de las contribuciones que aporta cada uno de los métodos a los objetivos finales, obteniéndose como resultado la seguridad que se tiene en relación con la pertenencia del patrón a clasificar a cada clase de la Información de Aprendizaje.

Para ejecutar el procesamiento de los grados de certidumbre se ha tomado una teoría utilizada para el desarrollo de Sistemas Expertos, que generaliza otras teorías clásicas en este campo, mediante una aproximación algebraica [10], [11].

Si se considera que $x_1, x_2, \ldots, x_r$ denotan valores de certidumbre provenientes de las contribuciones de r reglas de producción, entonces se necesita definir una operación $\oplus_o$ (o denota el tipo de operación) que permita combinar estos valores dos a dos y que cumpla con las siguientes propiedades:

1. $x_i \oplus_o x_j \in [-1,1]$ (operación interna)
2. Asociatividad: $x_i \oplus_o (x_j \oplus_o x_z) = (x_i \oplus_o x_j) \oplus_o x_z$
3. Conmutatividad: $x_i \oplus_o x_j = x_j \oplus_o x_i$
4. Existencia de elemento neutro: $x_i \oplus_o 0 = x_i$
5. Existencia del opuesto: $x_i \oplus_o -x_i = 0$
6. Monotonía: Si $x_i \leq x_j$ entonces $x_i \oplus_o x_z \leq x_j \oplus_o x_z$
7. $x_i \oplus_o 1 = 1,\ x_i \neq -1;\ x_i \oplus_o -1 = -1,\ x_i \neq 1$
8. $-1 \oplus_o 1$ indefinido

Como se puede deducir, esta operación sobre el conjunto de definición de valores de certidumbre [-1,1], define una estructura algebraica que es un Grupo Abeliano.

Para obtener una operación como ésta se busca un isomorfismo f, con el grupo de los números reales con la operación suma de la siguiente forma:

$$x_i \oplus_o x_j = f^{-1}(f(x_i) + f(x_j))$$ ($f^{-1}$ denota la inversa de f)

Existen diversas funciones que pueden ser usadas en esta operación; dos ejemplos interesantes para los que se demuestra que las operaciones resultantes son Grupos

Abelianos isomorfos a los utilizados en los Sistemas Expertos clásicos Prospector Mycin [20] son: $f_1(x) = Ln[1/(1-x)]$ y $f_2(x) = Ln[(1+x)/(1-x)]$ respectivamente.

Ejemplo: Si se obtienen contribuciones de -0.2 y 0.6 producto de la aplicación de dos reglas, entonces la contribución global de ambas utilizando $f_1$ es:

$$-0.2 \oplus_1 0.6 = f^{-1}(f(-0.2) + f(0.6)) = 0.455.$$

# 7. Incorporación de Conocimientos

Con la teoría propuesta, además de obtenerse un procedimiento para clasificación colaborativa entre métodos de Reconocimiento Supervisado, con la teoría propuesta, también se logra una estructura semejante a la de una BC construida utilizando experiencias humanas.

Lo anterior tiene diversas implicaciones, la primera tiene que ver con el proceso clasificación, puesto que no existe ningún problema en agregar los criterios de expertos humanos, mediante reglas de producción.

Supongamos que se agrega una regla de producción suministrada por los expertos como la siguiente:

$Ant \Rightarrow PTipo3'_i (W_1, W_2)$; donde "Ant" representa una proposición o combinación de proposiciones y $W_1$, $W_2$ son los pesos o grados de certidumbres para la afirmación negación absoluta de Ant, respectivamente.

Si consideramos que $W_p$ denota la contribución obtenida producto de la aplicación uno o varios criterios de reconocimiento y $W_e$ la contribución obtenida después de aplicar una o varias reglas suministradas por los expertos humanos, entonces el resultado combinar ambas contribuciones es: $W_p \oplus_o W_e$, lo cual brinda un resultado híbrido.

## 7.1. Elaboración de los Resultados

Otro aspecto a considerar es la respuesta final que debe brindar el sistema diseñado esta forma, puesto que como ya sabemos, lo que se obtiene es la seguridad, favorable desfavorable de la pertenencia de un patrón a clasificar a cada una de las clases existentes en la Información de Aprendizaje.

Sin embargo, la solución a esta situación no es única, ya que guarda cierta relación el problema particular que se este tratando.

Existen algunos criterios de conclusión suministrados por el sistema que el usuario puede elegir. Como ejemplos tenemos los siguientes:

Criterio 1.- Brindar como clasificación aquella clase con mayor valor de certidumbre positiva.

Criterio 2.- Brindar como solución hasta un máximo de $C_r$ clases cuyos valores certidumbre sean máximos valores positivos y la diferencia entre la de mayor valor la de menor no exceda a $W_d$ (valor externo).

Criterio 3.- Presentar la clasificación como dudosa si no existe una clase cuya diferencia con las demás es significativa. El concepto de significativo lo define el usuario introduciendo un parámetro de diferencia $W_d$ entre la clase de mayor peso y la siguiente en valor.

Así por ejemplo, si se está ante una situación donde no se admite la pertenencia de patrón a más de una clase, entonces el primer criterio puede resultar apropiado y se puede

introducir una variante de clasificación dudosa, para el caso en que dos o más clases estén muy cerca en los valores de certidumbre, lo cual es una combinación con el tercer criterio.

Finalmente, puede existir también el caso para el cual ninguno de los criterios resulte apropiado, ante tal situación el sistema propuesto permite que el usuario defina su proceso de elaboración de las conclusiones, utilizando una teoría que está diseñada especialmente con ese objetivo y que es parte de un ambiente para el desarrollo de sistemas inteligentes, denominado HAries [9], [10], [11].

Esto es posible ya que el sistema de Reconocimiento Supervisado que se propone puede trabajar de forma independiente o de forma colaborada con HAries, lo cual es necesario si se incorpora conocimiento humano al sistema. Esto ha permitido aprovechar las herramientas propias de los sistemas basados en conocimientos para construir sistemas híbridos.

# 8. Consideraciones Finales

Para lograr la combinación de varios clasificadores se ha construido una teoría que propone la construcción de una Base de Conocimientos a partir de la Información de Aprendizaje, para métodos de Reconocimiento Supervisado de Patrones.

Se observa, que no se realiza una combinación de los resultados provenientes de diversos métodos, sino que se utilizan sus teorías para producir contribuciones a la clasificación, cuyo resultado final es la combinación global de todas.

Se puede asegurar, que la arquitectura generada constituye una plataforma de trabajo en la que se pueden integrar otros métodos o criterios de clasificación supervisada, lo cual posibilita la ampliación de sus alcances y un ambiente para la experimentación.

Un caso típico son los árboles de decisión [20], puesto que a partir de estos se puede obtener reglas de producción con la misma estructura que las explicadas en los epígrafes anteriores.

Otro ejemplo son ios Algoritmos de Votación o de Zhuravliov [17] cuya idea fundamental yace en considerar que un objeto puede parecerse a otro, pero no en su totalidad y esto puede brindar información para la clasificación buscada.

De hecho en estos algoritmos ya se considera, como su nombre lo indica, ciertas ideas de clasificación colaborada y resulta muy natural también su incorporación a la arquitectura que se propone.

Estos resultados son muy útiles desde el punto de vista práctico, ya que posibilitan la inclusión de conocimientos humanos como parte del proceso de clasificación o a la inversa, de manera que se pueda utilizar una Base de Datos como parte del diseño de un Sistema Experto, de forma tal que los resultados sean producto de la combinación de ambas fuentes de información.

No obstante, quedan todavía diversas preguntas planteadas, como por ejemplos: ¿La incorporación de muchos métodos mejora la efectividad de la clasificación?, los resultados obtenidos en diversos trabajos apuntan a que la unificación tiende a mejorar los resultados, pero, ¿cuáles métodos usar ante un problema dado y hasta qué punto incorporar más métodos provoca una mejora en los resultados? sigue siendo aún, un problema abierto.

# Referencias


[1] Alexandre, L., Aurelio C. Campilho, A., Kamel, M.: "On combining classifiers using sum and product rules". Pattern Recognition Letters, Vol. 22 (12) pp. 1283 – 1289 (2001)

[2] Aliand, K. M., Pazzani, J. "On the link between error correlation and error reduction in decision tree ensembles". Technical Report UCI TR-95-38, University of California, Irvine, California (1995)

[3] Bauer, E., Kohavi, R. : "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants". Machine Learning, 36(1/2): 105-139 (1999)

[4] Breiman, L: "Bagging Predictors". Machine Learning, 24(2): 123-140 (1996)

[5] Cho, S. B., Jin Kim, J.: "Combining multiple neural networks by fuzzy integral for robust classification". IEEE Transactions on Systems, Man and Cybernetics, 25(2): 380-384 (1995)

[6] Cover, T.M., Hart, P.E.: "Nearest Neighbor Pattern classification". IEEE trans. Information Theory, IT-13 Pp 21-27 (1967)

[7] De la Cruz, A. V.: "Un sistema de algoritmos para reconocimiento de patrones supervisado con posibilidades de auto-aprendizaje. Ejemplos de aplicación". IV Jornada Científica del Instituto de Geofísica y Astronomía A.C.C. 18-20 Octubre (1984)

[8] De la Cruz, A. V.: "SRPS: Un sistema para reconocimiento de patrones supervisado" Informática'90, Reporte de participación. Pp 227-240 (1990)

[9] De la Cruz A.V., Valdés J.J., Pérez A., Jócik E., Balsa J., Rodríguez A., "The ARIES Environment for the Development of Knowledge Based Expert Systems". Industrial Applications of Artificial Intelligence. Elsevier Science Publishers. B.V. North-Holland, pp 203-209 (1991)

[10] De la Cruz, A.V., Valdés J.J., Jocik E., Balsa J., Rodríguez A.: Fundamentos y Práctica de la Construcción de Sistemas Expertos. Editorial Academia, La Habana, Cuba (1993)

[11] De la Cruz, A.V.: Construcción Práctica de Sistemas Expertos. Memorias Foro Computación de la teoría a la práctica, México DF, pp. 262-278 (1999)

[12] Dhillon, I.S., Modha, D.S.: "A Data-Clustering Algorithm on Distributed Memory Multiprocessors". Large-Scale Parallel Data Mining. Lecture Notes in Artificial Intelligence 1759. Subseries of Lecture Notes in Computer Science. Pp 245-260 (2000)

[13] Duda, R. O., Hart, P. E.: "Pattern Classification and Scene Analysis". John Wiley, New York (1973)

[14] Freund, Y., Schapire, R.: "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". J. Computer and System Sciences, 55:119-139 (1997)

[15] Johnson, E.L., Kargupta, H.: "Collective, Hierarchical Clustering from Distributed, Heterogeneous Data". Large-Scale Parallel Data Mining. Lecture Notes in Artificial Intelligence 1759. Subseries of Lecture Notes in Computer Science. Pp 221-244 (2000)

[16] Krogh, A., Vedelsby, J.: "Neural network ensembles cross validation, and active learning" NIPS 7. Lee 97 Peter M. Lee 1997. Bayesian Statistics. New York: Wiley (1995)

[17] Ruíz, J., Guzmán, A., Martínez, J. F.: "Enfoque lógico combinatorio al reconocimiento patrones". Instituto Politécnico Nacional, México (1999)

[18] Schaal, S., Christopher G.: "From isolation to cooperation: an alternative view of a system experts". NIPS 7 (1995)

[19] Schalkoff, R.: "Pattern Recognition Statistical, Structural and Neural Approaches". John Wiley and Sons, Inc. 364 Pp. (1992)

[20] Shapiro S. C.: "Encyclopedia of Artificial Intelligence". Volume 1. John Wiley & Sons (1992)

[21] Tumer, K., Joseph Ghosh, J.: "A framework for estimating performance improvements hybrid pattern classifiers". Proceedings of the World Congress on Neural Networks, III:220-225 (1994)

[22] Valev, V., Asaithambi, A.: "Multidimensional pattern recognition problems and combining classifiers". Pattern Recognition Letters, Vol. 22 (12) pp. 1291 – 1297 (2001)

# Reconocimiento de Patrones Auto-Supervisado. Método de las Clases Cerradas

Dr. Argelio Victor de la Cruz Rivera[1], M. en C. María de los Ángeles Alonso Lavernia[1]

[1] Centro de Investigación en Tecnologías de Información y Sistemas
Universidad Autónoma del Estado de Hidalgo (CITIS-UAEH)
argelioc@uaeh.reduaeh.mx, marial@uaeh.reduaeh.mx

**Resumen**

Se presenta un nuevo algoritmo para resolver problemas de Reconocimiento de Patrones No Supervisado, que puede ser considerado como un método híbrido, por una parte, dada su capacidad para trabajar como método de partición o jerárquico y por otra, debido a su estrategia de funcionamiento que utiliza ideas provenientes de los métodos supervisados.

Se introduce el concepto de Clase Cerrada, y un algoritmo para el cálculo de éstas en una matriz de datos cualquiera, el cual se utiliza como base para el desarrollo de cuatro variantes de clasificación.

Entre las ideas que se utilizan, se encuentra realizar un Reconocimiento No Supervisado sobre los datos, utilizando como información de aprendizaje la propia pertenencia a las clases de cada individuo en el momento que se encuentre el proceso, de aquí que en cierto sentido la clasificación sea también auto-supervisada.

Los resultados pueden ser una partición de los datos en una cantidad de grupos predefinido o la representación mediante un Dendrograma, a través del cual se puede extraer una clasificación para los datos procesados.

## 1. Introducción

Los métodos de Reconocimiento de Patrones pueden ser divididos en dos grupos fundamentales: Reconocimiento Supervisado y Reconocimiento No Supervisado [9], [12], [13], aunque algunos autores también incluyen métodos de selección de variables y variantes de los anteriores [5], [12], [14], [15].

En general, se parte de suponer que se tiene una matriz de datos M, formada por "n" filas y "m" columnas, donde las filas representan los individuos observados y las columnas los parámetros o variables que han sido medidos a esos individuos, es decir:

$$M = \begin{pmatrix} X_{11} & \dots & X_{1m} \\ & \vdots & \\ X_{n1} & \dots & X_{nm} \end{pmatrix}$$

donde $X_{ij}$ representa el valor del individuo i para la variable j ($i \in \{1, \dots, n\}$, $j \in \{1, \dots, m\}$)

Los métodos de Reconocimiento No Supervisados tratan de resolver el problema de encontrar una clasificación adecuada de los individuos en "g" clases o grupos, de forma tal que la semejanza entre los integrantes de una clase sea puesta de manifiesto, al igual que las diferencias entre individuos de clases diferentes.

Por otra parte, los métodos de Reconocimiento Supervisados, parten de suponer que los individuos de la matriz M, ya se encuentran clasificados en "g" clases (por cualquier

vía) y el problema a resolver consiste en: clasificar otro conjunto de individuos, caracterizados por las mismas "m" variables, cuya pertenencia a las clases de M es desconocida. Es común en este caso denominar como Información de Aprendizaje a la matriz M.

En el presente trabajo, se introduce un tipo de método con características especiales, puesto que la tarea que desempeña está dirigida a resolver problemas de Reconocimiento No Supervisado, pero la forma en que se ejecuta la clasificación se asemeja a la de un método Supervisado [3], [4] y específicamente a la conocida regla de los k vecinos más cercanos [2].

La idea que se desarrolla consiste en considerar que la clasificación se ejecuta por pasos, de forma tal, que en cada momento los individuos ya clasificados se utilizan como información de aprendizaje para clasificar los restantes.

En el método propuesto se parte de suponer que al inicio del proceso de clasificación, cada individuo constituye una clase independiente y que al estilo de los métodos jerárquicos aglomerativos [1], [6], [7], [11], se ejecuten fusiones en pasos sucesivos según determinados criterios, hasta que la cantidad de clases se haga igual a un número "NC", que constituye un parámetro de entrada a los algoritmos.

Dada esta organización del proceso, es posible la representación de los pasos de clasificación a través de un Dendrograma, lo cual permitirá representar gráficamente configuración de los datos y determinar la cantidad de grupos si es desconocida.

Por otra parte, el método también hereda algunas características de los algoritmos partición [1], [7], [10], puesto que el resultado que se brinda es precisamente una partición de los individuos en una cantidad fija de clases.

La idea general del procedimiento se resume en los siguientes hechos:

1) Para cada grupo existente en el estado "e" de la clasificación (el concepto de estado se define formalmente en el epígrafe 3) se deben obtener los "KNN" individuos más semejantes entre los externos a la clase como tal. El número" KNN" es un dato externo que recibe el algoritmo.
2) Este proceso se ejecuta para cada integrante de la clase de forma individual y toman los "KNN" individuos de máxima semejanza, entre todos los obtenidos forma particular, como los elementos asociados al grupo completo.
3) La clasificación o proceso de fusión, se ejecutará supervisado por aquellas clases las que pertenezcan los individuos más semejantes.
4) La influencia en la clasificación de los "KNN" individuos, estará condicionada el orden que ocupen respecto a los valores de semejanzas, es decir, el individuo más cercano será el de mayor influencia y así sucesivamente en orden decreciente.

Partiendo de éstas ideas básicas, se definen un conjunto de conceptos y propiedades que permiten construir el método propuesto.

## 2. Vector de Votación

El Cálculo de la influencia mencionada en el cuarto punto se ejecuta correspondencia con un vector de votación que se define a continuación.

Utilizaremos las siguientes notaciones:

- "KNN" denotará el número de individuos más semejantes que serán considerados entre lo n individuos que forman la matriz de datos ($1 \leq KNN \leq n$).

- "KNNx" representará al conjunto formado por los KNN individuos más semejantes al individuo "x" ordenados de mayor a menor parecido, de acuerdo con alguna medida de la semejanza.
- "KNNx(j)" representará al individuo j del conjunto KNNx.
- "a" denota la cantidad de clases a las que pertenecen los individuos del conjunto KNNx. Se observa que $1 \leq a \leq KNN$ dado que más de un individuo puede pertenecer a la misma clase.

**Definición 1.-** Se llama vector de votación $V(x) = (V_1, V_2, \ldots, V_a)$ respecto a un individuo "x", en relación con las clases $C_1, C_2, \ldots, C_a$, a las que pertenecen los KNN individuos más semejantes a "x", a aquel cuyas componentes $V_i$ se obtienen como sigue:

$$V_i = \sum_{j=1}^{KNN} CTR_{KNNx(j)}(C_i)$$

donde $CTR_{KNNx(j)}(C_i)$ denota la contribución del individuo j respecto a la clase $C_i$, con

$$CTR_{KNNx(j)}(C_i) = \begin{cases} 0 & \text{si } KNNx(j) \notin C_i \\ \dfrac{KNN-(j-1)}{KNN} & \text{si } KNNx(j) \in C_i \end{cases}$$

Como se puede observar aquellas clases que no estén representadas entre los KNN individuos más semejantes no tendrán votación. Por otra parte, se observa que al encontrarse el conjunto KNNx ordenado de acuerdo a la semejanza, el primer individuo será el de mayor contribución con votación 1 exactamente, mientras que los demás tendrán menor votación, hasta llegar al último cuya votación será 1/KNN.

Se observa también, que en la definición del vector de votación no se toma en cuenta semejanza real existente entre los individuos. Esto puede provocar que un grupo de individuos poco parecido aporte más, lo cual puede provocar resultados no deseados, como se observa en el siguiente ejemplo.

Ejemplo 1.- Si consideramos que KNN=4 y que el primer individuo del conjunto KNNx pertenece a la clase 1 y los restantes a la clase 2, el vector de votación es (1 , 1.5) lo cual inclina la clasificación hacia la clase 2. Sin embargo, pueden ocurrir dos casos como los siguientes:

Parecidos entre el individuo x y los 4 más semejantes

a) 0.9 , 0.88 , 0.87 , 0.86
b) 0.9 , 0.3 , 0.28 , 0.25

Se puede observar que la diferencia en parecido entre el primer individuo y los restantes, para el caso "a", no es sustancial, mientras que en el segundo caso si lo es, y sin embargo, el resultado es el mismo de acuerdo al vector de votación.

Tomando en cuenta esta situación se introduce una variante en la que se utiliza la medida de la semejanza para ponderar el cálculo del vector de votación como sigue:

Componentes del vector de votación ponderado $V_i = \sum_{j=1}^{KNN} CTR_{KNNx(j)}(C_i) * S(KNN_x(j), x)$

donde $S(KNN_x(j), x) \in [0,1]$ representa la semejanza entre el individuo x y el individuo KNNx(j) dentro del conjunto de los más parecidos a "x".

Lo anterior permite tomar en cuenta, tanto la posición entre los más parecidos, como la semejanza entre ellos.

Ejemplo 2.- Si tomamos como referencia los mismos datos del ejemplo 1 y obtenemos los vectores de votación ponderados se obtiene lo siguiente:

a) $V_1=1(0.9)=0.9$, $V_2=3/4(0.88) + 2/4(0.87) + 1/4(0.86)) = 1.31$, luego el vector de votación ponderado es $V(x) = (0.9 , 1.31)$

b) $V_1=1(0.9)=0.9$, $V_2=3/4(0.3) + 2/4(0.28) + 1/4(0.25)) = 0.4275$, luego el vector de votación ponderado es $V(x) = (0.9 , 0.4275)$

Se observa que en el primer caso la clase 2 sigue manteniendo el máximo votación, mientras que en el segundo caso la situación cambia, debido a los bajos valores de semejanzas con relación al primer individuo.

Se puede considerar que cada componente $V_i$ del vector de votación representa medida de la cercanía del individuo "x" a la clase $C_i$ y este hecho puede ser usado como un criterio para ejecutar la fusión de clases semejantes.

Lo anterior significa, que utilizando el vector V(x) se puede decidir cuando individuo "x" será reclasificado como integrante de aquella clase para la cual se obtuvo el máximo de votación, pero que a su vez el resto de los integrantes de la clase, donde estaba clasificado "x", también se verán afectados y por consiguiente reclasificados de la misma forma.

Tal acción provoca una disminución en el número de clases y la desaparición correspondiente al individuo en análisis.

Bajo determinadas condiciones puede ocurrir que algunas componentes del Vector de Votación sean iguales y por tanto se hace necesario establecer criterios solucionar este conflicto.

## 2.1. Criterios de utilización para el Vector de Votación

Existen dos posibilidades a la hora de utilizar el vector de votación en la fusión clases:

Criterio 1.- Máximo a la Izquierda. Siempre busca el máximo lo más a la izquierda posible del vector, es decir, si existen dos valores iguales se toma la clase del primer valor.

Criterio 2.- Máximo a la Derecha. Contrariamente al anterior, se toma el valor alejado dentro del vector.

En el primer criterio se hace énfasis hacia la importancia creciente de los primeros individuos, mientras que en el segundo, se prioriza la cantidad de individuos semejantes que aportan contribución sobre cada clase. Estos son criterios que introducir el usuario como datos.

Ejemplo 3.- Si KNN = 3, las contribuciones potenciales de cada uno de los individuos (sin considerar la semejanza) son las siguientes: 1, 2/3, 1/3, es decir el individuo semejante aporta 1 a la votación, el siguiente en importancia 2/3 y el último 1/3. Si el primer individuo pertenece a la clase $C_1$ y los otros dos a la clase $C_2$, el de votación quedaría como sigue: $V(x) = (1, 1)$, es decir existe la misma contribución hacia ambas clases, sin embargo, bajo el criterio Máximo a la Izquierda, se tomaría como clase más cercana $C_1$, con lo cual el primer individuo tiene prioridad, mientras que en el segundo caso, se tomaría $C_2$ lo cual expresa que al encontrarse

individuos, entre los KNN pertenecientes a una misma clase, esto se tendrá en mayor consideración.

# 3. Clases Cerradas

El método que se propone plantea la utilización del vector de votación para obtener clases con determinadas características de estabilidad. Comenzaremos analizando como se ejecuta el proceso.

**Definición 2.-** Se llama Iteración al proceso de análisis de todos los individuos de la matriz de datos, ordenados según los valores del individuo de mayor semejanza, para su posible reclasificación según el concepto de votación.

Respecto a esta definición se necesita mencionar dos aspectos importantes:

1. El análisis de los individuos es ordenado por las semejanzas al primer integrante de los KNN correspondiente, ya que se van tomando para el análisis los individuos cuyos valores de semejanza al resto sean los mayores.
2. En una misma iteración un individuo pudiera ser relocalizado más de una vez, debido a la característica hereditaria que le brinda un individuo a los integrantes de su grupo.
3. El proceso de relocalización de individuos implica fusiones entre las clases.

**Definición 3.-** Un individuo i se llama Estable en la iteración "t" si no es relocalizado (cambiado de clase) en t.

**Definición 4.-** Un individuo i se llama Absolutamente Estable si luego de alcanzar la estabilidad en la iteración t, éste no tiene posibilidades de ser relocalizado en ninguna iteración posterior.

El siguiente hecho es una consecuencia directa de las definiciones anteriores.

**Proposición 1.-** Si un individuo i que pertenece a la clase $C_L$ es Absolutamente Estable, entonces todos los integrantes de $C_L$ también lo son. A $C_L$ se le denomina Clase Absolutamente Estable.

**Demostración:** Utilizaremos el método de reducción al absurdo, para lo cual supondremos que no se cumple la afirmación planteada para tratar de obtener una contradicción. Supongamos entonces, que a pesar de existir un individuo i absolutamente estable en la clase $C_L$ ($i \in C_L$), existe otro individuo $j \in C_L$, que no lo es. Si esto ocurre, entonces el individuo "j" puede ser reclasificado por no ser absolutamente estable (definición 4) y por tanto, esto significa que al ser reclasificado en una iteración t, también el resto de los integrantes de su clase sería reclasificados, incluyendo el "i" que de hecho no sería entonces absolutamente estable. De lo anterior se obtiene una contradicción con la suposición inicial, quedando demostrada así la afirmación planteada.

De lo anterior se deduce que la propiedad de estabilidad absoluta para un individuo, se puede obtener directamente de la estabilidad del resto, es decir, analizando si algún individuo de su clase puede ser reclasificado o no.

Por otro lado la propiedad de estabilidad absoluta puede ser enfocada según la siguiente propiedad:

**Proposición 2.-** Una clase $C_L$ es Absolutamente Estable sí y sólo si, el valor máximo de cada vector de votación asociado a sus miembros, se corresponde con individuos de la propia clase $C_L$.

La demostración de esta proposición se obtiene de la aplicación de las definiciones y 4 respectivamente.

**Definición 5.-** Sea "$C_L$" una Clase Absolutamente Estable de M. Si para todo individuo $I_j \in M$ no incluido en $C_L$ ($I_j \notin C_L$) se cumple que el valor máximo del vector votación correspondiente V(Ij) no coincide con $C_L$, se dice que ésta forma una Clase Cerrada de Primer Orden o Pura.

De la definición se observa, que la condición de Clase Cerrada implica, que ninguno de sus miembros tiene posibilidades de ser reclasificado en iteraciones posteriores del estado actual y que a su vez, ningún individuo externo a la clase puede ser incluido ésta. Por tanto, una vez alcanzada esta propiedad una clase permanecerá invariante iteraciones posteriores.

En la siguiente proposición se analiza la cantidad de iteraciones necesarias para obtener una configuración de Clases Cerradas.

**Proposición 3.-** El número de iteraciones máximo necesario para obtener todas Clases Cerradas para KNN $\in \{1,2\}$ es exactamente 1.

**Demostración**: Analicemos que ocurre si KNN=1. Para demostar que en una iteración se obtienen todas las Clases Cerradas, es suficiente demostrar, que basta con una iteración para que todos lo individuos adquieran la propiedad de estabilidad absoluta, puesto que al cumplirse esto ningún individuo puede ser reclasificado y por tanto, todas las clases son cerradas. Ahora bien, como el proceso de clasificación se ejecuta relocalizando a cada individuo en la clase donde se encuentre el más semejante a si se ejecuta una segunda iteración no habrá posibilidades de cambios para individuo alguno, pues ya todos se encuentran en la misma clase de su individuo más semejante y por tanto todos son absolutamente estables. Para KNN=2 no hay diferencias, pues la primera componente del vector, siempre será mayor que segunda, lo cual implica que esta última no afecta el resultado con relación primero.

Analicemos qué ocurre cuando se obtienen todas las Clases Cerradas de Primer Orden para una matriz de datos M. Para ello introduciremos el concepto de Estado de Clasificación.

**Definición 6.-** Se llama Estado de la Clasificación de Primer Orden, a una configuración tal que, todas las clases existentes son cerradas de primer orden.

No obstante, necesitamos analizar qué ocurre cuando se obtiene un estado de primer orden y el número de clases obtenidas 'c' es mayor que el número de clases buscada (NC), lo cual significa que se ha obtenido un estado parcial de la clasificación.

El cambio de un estado a otro se ejecuta de la siguiente forma:

1.- Se obtienen los KNN individuos más semejantes a cada individuo de la matriz datos, considerando solamente aquellos que no pertenezcan a su misma clase.

2.- Para cada clase *l* se consideran las KNN individuos obtenidos para sus integrantes. Si $N_l$ denota la cantidad de elementos de esta clase, entonces se obtiene subconjunto de individuos que oscila entre KNN y KNN*$N_l$.

3.- Se extraen los KNN de mayor semejanza entre los elementos de este conjunto obtenido y estos serán considerados como los asociados a cada uno de integrantes de la clase.

Bajo esta nueva situación se puede proceder como antes para obtener nuevas Clases Cerradas formadas por fusión de las anteriores. Esta idea nos lleva a la siguiente definición.

**Definición 7.-** Se llama Clase Cerrada de Orden "e" (e>1), a aquella obtenida producto de la fusión de 2 o más Clases Cerradas de orden e-1 y que cumple con la definición 5. El orden "e" representa el estado actual de la clasificación.

Tomando como base los conceptos explicados se puede implementar un algoritmo de clasificación con varias variantes.

# 4. Algoritmo de clasificación utilizando el concepto de Clase Cerrada

Utilizando los conceptos estudiados, se puede construir un algoritmo de clasificación con diferentes variantes y formas de presentar los resultados.

Variante 1.- Encontrar las clases cerradas de primer orden.

Variante 2.- Encontrar una partición de los individuos en Nc clases.

Variante 3.- Obtener el último nivel de clases cerradas posible donde el número de clases sea diferente de 1.

Variante 4.- Obtener un dendrograma del proceso de fusión de los individuos y clases, partiendo de n clases y terminado el proceso en una.

De lo anterior se puede observar, que el resultado que se busca no tiene que ser, necesariamente, una configuración de clases cerradas, aunque en todos los casos se utilizará este concepto para obtener los resultados.

Por otro lado, serán necesarios varios parámetros como entrada.

1. Valor de KNN.
2. Criterio para el vector de votación si KNN > 2.
3. Función de semejanza a utilizar.
4. Tipo de resultado buscado (Variante 1, 2, 3 o 4).

Los pasos generales del algoritmo se pueden resumir en los siguientes pasos:

Paso 0. Considerar que cada uno de los "n" individuos a clasificar constituye un grupo individual, es decir, al inicio existirán n clases.

Paso 1. Calcular la matriz de semejanza entre los "n" individuos. Esto puede ser ejecutado opcionalmente.

Paso 2. Obtener para cada individuo "i" de la matriz de datos, los KNN más semejantes (usando la matriz de semejanza o calculándola en este momento), tomando en cuenta que no pertenezcan a su misma clase.

Paso 3. Obtener para cada clase *l* los KNN individuos más semejantes entre los obtenidos para cada uno de sus miembros y considerar que a todos los integrantes de forma individual le corresponden estos, en lugar de los obtenidos en el paso 2.

Paso 4. Obtener el vector de votación según el criterio definido y a partir de este, la clase más parecida a cada individuo.

Paso 5. En correspondencia con el tipo de resultado buscado, ejecutar lo siguiente:

Variante 1.- Obtener las clases cerradas e ir al paso 7.

Variante 2.- Obtener las clases cerradas para el estado actual "e", tomando en cuenta que el proceso se ejecute hasta el final, sólo si el número de clases actual "Na" es mayor que el número de clases buscado "Nc"; puesto que en caso contrario se

salta al paso 7. Esto significa que nunca se obtendrá un número de clases menor que el buscado. Si el proceso finaliza ir al paso 6.

Variante 3.- Guardar el estado actual de la clasificación "e". Obtener un nuevo estado e+1 obteniendo las clases cerradas. Si el número de clases es mayor que 1 ir paso 2. Si el número de clases es uno considerar como estado actual el "e" e ir al paso 7.

Variante 4.- Obtener las clases cerradas para el estado actual "e". Si el número clases es uno ir al paso 7, mientras que en caso contrario ir al paso 2.

Paso 6. Si el número de clases cerradas en el estado actual es igual a Nc entonces obtuvo el estado final concluyendo la clasificación, mientras que en caso contrario, se regresa el Paso 2. Este paso solo se ejecuta para la variante 2.

Paso 7. Para las variantes 1, 2 y 3 se presenta como resultado el estado actual de clasificación. Para la variante 4 se brinda como resultado el Dendrograma resultante del proceso ejecutado.

## 5. Un Ejemplo

Para mostrar el funcionamiento del algoritmo vamos a considerar una matriz de datos hipotética muy sencilla, la cual está formada por 10 individuos caracterizados por variables numéricas. En la figura 1 se presentan los datos y su representación espacial, mientras que en la tabla 1 se muestra la correspondiente matriz de Distancias Euclidianas, como medida de la semejanza.

| | 1 | 2 |
|---|---|---|
| 1 | 4 | 2 |
| 2 | 9 | 6 |
| 3 | 7 | 7 |
| 4 | 6 | 6 |
| 5 | 1 | 1 |
| 6 | 7 | 4 |
| 7 | 4 | 1 |
| 8 | 5 | 1 |
| 9 | 2 | 2 |
| 10 | 3.5 | 1.5 |

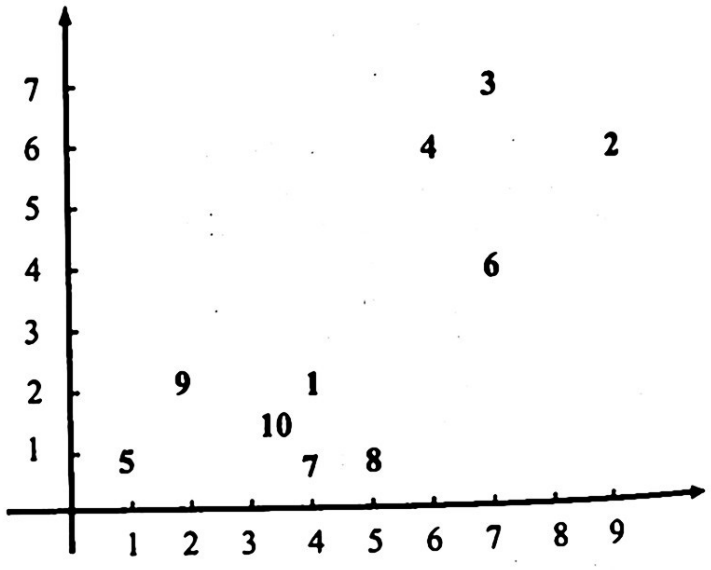



**Figura 1.** Matriz de Datos y su representación gráfica.

La tarea planteada consiste en clasificar los individuos en 2 grupos utilizando variante 2 del algoritmo presentado. Esto significa que el objetivo no es encontrar clases cerradas exactamente. Por simplicidad utilizaremos como parámetro KNN = así no será necesario utilizar el vector de votación, además de necesitarse una iteración para obtener cada estado de la clasificación (ver proposición 3).

Para ejecutar el algoritmo nos apoyaremos en dos tablas, en las que se resumen distintos estados e iteraciones producto de la aplicación del método.

En la tabla 2 se presenta para cada individuo, el más semejante en cada estado de la clasificación y en la tabla 3, la clase en que se encuentra clasificado en cada estado e iteración específica.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | | | | | | | | | |
| 2 | 6.40 | 0 | | | | | | | | |
| 3 | 5.83 | 2.24 | 0 | | | | | | | |
| 4 | 4.47 | 3 | 1.41 | 0 | | | | | | |
| 5 | 3.16 | 9.43 | 8.49 | 7.07 | 0 | | | | | |
| 6 | 3.61 | 2.83 | 3 | 2.24 | 6.71 | 0 | | | | |
| 7 | 1 | 7.07 | 6.71 | 5.39 | 3 | 4.24 | 0 | | | |
| 8 | 1.41 | 6.40 | 6.32 | 5.1 | 4 | 3.61 | 1 | 0 | | |
| 9 | 2 | 8.06 | 7.07 | 5.66 | 1.41 | 5.39 | 2.24 | 3.16 | 0 | |
| 10 | 0.71 | 7.11 | 6.52 | 5.15 | 2.55 | 4.3 | 0.71 | 1.58 | 1.58 | 0 |

**Tabla 1**. Matriz Triangular Inferior de Distancias Euclidianas.

| | Estado | Individuos 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Individuos más Semejantes | 1 | 10 | 3 | 4 | 3 | 9 | 4 | 10 | 7 | 5 | 1 |
| Distancia | | 0.71 | 2.24 | 1.41 | 1.41 | 1.41 | 2.24 | 0.71 | 1 | 1.41 | 0.71 |
| Individuos más Semejantes | 2 | 9 | 1 | 1 | 1 | 10 | 1 | 9 | 9 | 10 | 9 |
| Distancia | | 2 | 6.4 | 5.84 | 4.47 | 2.55 | 3.61 | 2.24 | 3.16 | 1.58 | 1.58 |

**Tabla 2**. Individuos más semejantes en cada estado de la clasificación.

| | Estado | Iteración | Individuos 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Clases | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| | | 1 | 10 | 4 | 4 | 4 | 9 | 4 | 10 | 10 | 9 | 10 |
| Clases | 2 | 1 | 10 | 4 | 4 | 4 | 10 | 4 | 10 | 10 | 10 | 10 |

**Tabla 3**. Clase a la que pertenece cada individuo después de terminada la iteración.

Como se puede observar en la Tabla 3, para el estado inicial (iteración 0) se considera que cada individuo constituye una clase independiente.

A continuación se comienza el proceso de reclasificación analizando para cada individuo a qué clase pertenece el más semejante a él.

El proceso iterativo se organiza utilizando los parecidos obtenidos entre los individuos, por tanto el orden de análisis para estos, en la primera iteración es: 1, 7, 10, 8, 3, 4, 5, 9, 2, 6.

Lo anterior significa que el orden en que se coloquen los individuos a clasificar no influye en los resultados, dado que el proceso iterativo es ordenado por las propias

semejanzas entre los individuos. Esto garantiza que las fusiones que se ejecuten siempre sean entre aquellos individuos o clases de mayor parecido.

Lo anterior se observa claramente en las clases que prevalecen después de la primera iteración (tabla 3, fila de la iteración 1).

Por ejemplo se observa (tabla 2, estado 1) que el individuo más semejante a 1 10, pero como este último pertenece a la clase 10, entonces se reclasifica el individuo pasando a formar parte de esa misma clase, seguidamente se tiene que el individuo semejante al 7 es a su vez el 10, luego también queda incluido en esta clase.

El proceso continúa en el orden señalado hasta completar una iteración. El resultado hasta aquí, son tres clases cerradas, la primera formada por los individuos 2, 3, 4 y segunda por 5 y 9 y la tercera formada por 1, 7, 8 y 10 (tabla 3, estado 1, iteración 1).

Como el objetivo planteado es obtener una partición en dos clases, se necesita obtener otro estado de la clasificación.

En la tabla 2, filas del estado 2, se observa el resultado de obtener los individuos semejantes para cada uno, tomando en cuenta que no pertenezcan a su clase.

Se puede apreciar que el proceso iterativo comienza en el individuo 9, el cual, con el resto de los integrantes de su grupo (en este caso 5) se unen al grupo 10. Como e este paso se obtienen dos clases el proceso se detiene obteniéndose como resultado:

Grupo 1: {1, 5, 7, 8, 9, 10}

Grupo 2: {2, 3, 4, 6}

## 6. Construcción del Dendrograma

Como se había mencionado, otra característica interesante del método propuesto posibilidad de representar el resultado mediante un Dendrograma, lo cual es muy como método exploratorio de los datos en caso de ser completamente desconocido estructura de clases [9].

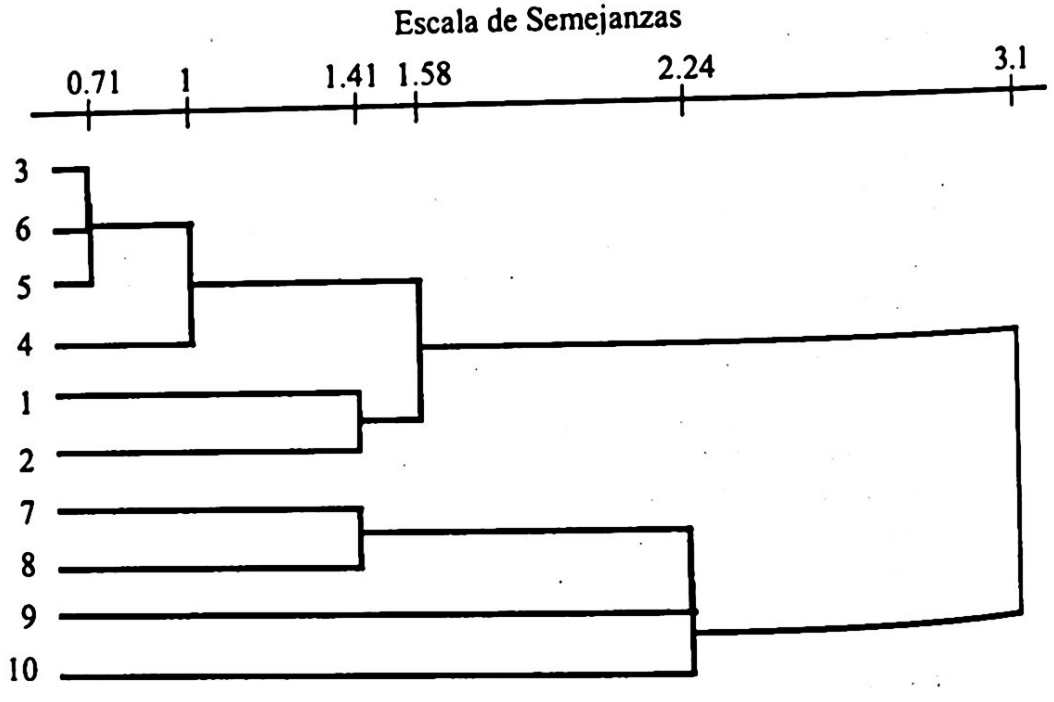



**Figura 2.** Dendrograma del proceso de agrupación.

En la figura 2 se muestra el Dendrograma resultante del proceso de clasificación efectuado en el conjunto de datos del ejemplo anterior.

A la izquierda se ubican los individuos y en la parte superior la escala que mide los valores de semejanza de los pasos de fusión ejecutados en el proceso de clasificación.

Tomando como punto de partida las tablas 2 y 3, se puede seguir el proceso de construcción del Dendrograma como se muestra en la tabla 4.

| Pasos de Fusión | Individuos agrupados | Semejanza |
|---|---|---|
| 1 | 3 con 6 | 0.71 |
| 2 | (3, 6) con 5 | 0.71 |
| 3 | (3, 6, 5) con 4 | 1 |
| 4 | 1 con 2 | 1.41 |
| 5 | 7 con 8 | 1.41 |
| 6 | (7, 8) con 9 | 2.24 |
| 7 | (7, 8, 9) con 10 | 2.24 |
| 8 | (3, 6, 5, 4) con (1, 2) | 1.58 |
| 9 | (3, 6, 5, 4, 1, 2) con (7, 8, 9, 10) | 3.1 |

**Tabla 4.** Proceso de construcción del Dendrograma.

# 8. Conclusiones

Como se ha podido apreciar, el método propuesto representa una herramienta para abordar el problema de reconocimiento no supervisado que utiliza algunas ideas híbridas provenientes de diversos tipos de métodos.

Así podemos reconocer la estrategia utilizada por los métodos jerárquicos de considerar a cada individuo como grupo independiente, para posteriormente ejecutar sucesivas fusiones, mientras que el resultado constituye una partición de los individuos en una cantidad de clases que preferentemente debe ser conocida.

Por último, la idea de utilizar los mismos individuos de la matriz para supervisar la clasificación y específicamente los KNN más semejantes, recuerda la forma en que trabajan los métodos de reconocimiento supervisado.

El uso de la matriz de semejanza como punto de partida presenta la dificultad propia del crecimiento cuadrático que ésta experimenta con el aumento de la cantidad de individuos a clasificar, pero presenta la ventaja de contar con toda la información relativa a todas las semejanzas dos a dos entre los individuos, disminuyendo sensiblemente la cantidad de cálculos, no obstante, un algoritmo utilizando directamente la Matriz Observacional se encuentra también implementado, pero ello no varia la teoría presentada.

Es de destacar que estamos en presencia de un método con el que se puedan abordar problemas donde las variables pueden ser tanto numéricas como no numéricas, permitiendo incluso la existencia de información incompleta [4], [8].

Es importante, mencionar que este método permite obtener una historia basada en el concepto de clases cerradas, en la cual se muestra, utilizando el Dendrograma, la clasificación de los individuos de un extremo a otro, es decir, partiendo de considerar

cada individuo como un grupo independiente, hasta considerarlos todos como miembros de uno solo.

Finalmente, es de mencionar que se han realizado algunos estudios comparativos de este método con algoritmos jerárquicos y de partición clásicos, utilizando datos simulados y casos reales conocidos, pero dada la extensión del trabajo estos resultados no se presentan.

Es de destacar, que en todos los casos se observó un mejor comportamiento utilizando el concepto de clases cerradas que se propone.

## Referencias


[1] Anderberg, M. R.: "Cluster Analysis for applications", Academic Press, New York (1973).

[2] Cover, T.M., Hart, P.E.: "Nearest Neighbor Pattern classification". IEEE trans. Information Theory, IT-13 Pp. 21-27 (1967)

[3] De la Cruz, A. V.: "Un sistema de algoritmos para reconocimiento de patrones supervisado con posibilidades de auto-aprendizaje. Ejemplos de aplicación". Jornada Científica del Instituto de Geofísica y Astronomía A.C.C. 18-20 Octubre 1984

[4] De la Cruz, A. V.: "SRPS: Un sistema para reconocimiento de patrones supervisado". Informática'90, Reporte de participación. Pp. 227-240 (1990)

[5] De la Cruz, A. V.: "Reconocimiento Semi-Supervisado. Método de las Clases Cerradas.". International Conference, Science and Technology for Development, CIMAF'95. La Habana, Cuba, 23-27 Enero (1995)

[6] Dhillon, I.S., Modha, D.S.: "A Data-Clustering Algorithm on Distributed Memory Multiprocessors". Large-Scale Parallel Data Mining. Lecture Notes in Artificial Intelligence 1759. Subseries of Lecture Notes in Computer Science. Edited by J.G. Carbonell and J. Siekmann. Pp. 245-260 (2000)

[7] Everitt, Brian: "Cluster Analysis", Social Science Research Council (1974)

[8] Gower, J.C.: "A general coefficient of similarity and some of its properties" biometrics Vol.27, No.4, Pp. 857-874 (1971)

[9] Han, J., Kamber, M.: "Data Mining Concepts and Techniques". Morgan Kaufmann Publishers. 550 Pp. (2000)

[10] Hartigan, J. A.: "Clustering Algorithms". Wiley (1975)

[11] Johnson, E.L., Kargupta, H.: "Collective, Hierarchical Clustering from Distributed, Heterogeneous Data". Large-Scale Parallel Data Mining. Lecture Notes Artificial Intelligence 1759. Subseries of Lecture Notes in Computer Science. Edited by J.G. Carbonell and J. Siekmann. Pp. 221-244 (2000)

[12] Kanal, L.N., Dattatteya, G.R.: "Pattern Recognition". Encyclopedia of Artificial Intelligence. Vol. 2. Second Edition. John Wiley & Sons. Pp. 1116-1129 (1992)

[13] Ripley, B. D.: "Pattern Recognition and Neural Networks": Cambridge UNIVERSITY PRESS. 403 Pp. (1996)

[14] Ruiz, J., Guzmán, A., Martínez, J. F.: "Enfoque lógico combinatorio reconocimiento de patrones". Instituto Politécnico Nacional, México (1999)

[15] Schalkoff, R.: "Pattern Recognition Statistical, Structural and Neural Approaches" John Wiley and Sons, Inc. 364 Pp. (1992)

# MATRIZ DE CONOCIMIENTO SOBRE LA COMPLEJIDAD DE PREDICCIÓN EN SERIES DE TIEMPO.

**E. Bautista-Thompson & J. Figueroa-Nazuno**
Centro de Investigación en Computación.
Instituto Politécnico Nacional.
Unidad Profesional "Adolfo López Mateos" Zacatenco
Col. Linda Vista, C.P. 07738
México D. F.
ebautista@correo.cic.ipn.mx jfn@cic.ipn.mx

**Resumen**

Se presentan los resultados de la evaluación de la capacidad de diferentes técnicas para la predicción y modelado de series de tiempo. Utilizando mapas autoorganizados (SOM), se identificaron relaciones entre la capacidad de predicción de las técnicas y la complejidad de las series de tiempo. Se construyeron clases de complejidad de las series y clases de capacidad de predicción de las técnicas.

**Palabras clave:** Predicción, Series de Tiempo, Complejidad, SOM.

## 1. Introducción.

En la literatura constantemente se reportan nuevas metodologías para la predicción de series de tiempo [1, 2, 3], sin embargo a pesar del gran interés en el desarrollo de nuevas técnicas de predicción, no hay criterios homogéneos para la evaluación de la capacidad de predicción de las técnicas desarrolladas, también se observa que los conceptos de predicción y modelado de series de tiempo se usan de forma indistinta y ambigua en la literatura [4]; cuando en realidad la predicción y el modelado de una serie de tiempo operativamente son conceptos diferentes. Por ultimo, se han hecho algunos trabajos sobre clasificación de series de tiempo basado en el análisis de parámetros característicos determinados con técnicas de sistemas dinámicos no lineales [5, 6, 7], sin embargo no se encuentra en la literatura información sobre las relaciones que pueden existir entre la complejidad de las series de tiempo (determinada en base a los parámetros que caracterizan a las series), y el tipo de técnicas de predicción o modelado utilizadas. Los argumentos anteriores nos motivan a desarrollar el análisis que se presenta en este trabajo con el objetivo de generar una matriz de conocimiento que nos permita extraer reglas sobre la predicción de series de tiempo.

Para realizar el análisis sobre la capacidad de predicción se utilizaron dos clases de series de tiempo: series experimentales, las cuales presentan comportamiento complejo (económicas, sísmicas, fisiológicas) y series sintéticas en las cuales se tiene control sobre los parámetros de la función generadora (Seno, Lorentz, Henon, Ikeda). Las técnicas de predicción y modelado evaluadas, fueron seleccionadas bajo el criterio de que fuesen representativas de los diferentes modelos y teorías usados para la predicción de series de tiempo como son: Estadística Moderna, Teoría de Sistemas Dinámicos No Lineales e Inteligencia Artificial (e. g. Redes Neuronales). Para la determinación de la complejidad de las series de tiempo se hizo uso de diversas herramientas de análisis: Chaos Data Analyzer, VRA 4.2, NLyzer, y Técnica de Extracción de Gramáticas, con las cuales se calcularon los parámetros característicos de las series de tiempo: Exponente Principal de Lyapunov, Entropías, Mapa de Recurrencia, Dimensiones, Coeficiente de Hurst, Reglas de Producción Gramatical [ 8, 9, 10, 11].

La información generada fue analizada utilizando mapas autoorganizados (*SOM*) [12, 13], los cuales permiten clasificar la información y así determinar las relaciones existentes entre complejidad de la serie y capacidad de predicción de las técnicas.

## 2. Evaluación de las Técnicas y Caracterización de las Series de Tiempo.

Se seleccionaron 16 series de tiempo (ver Tabla 1) representativas de las diferentes clases de series de acuerdo a la clasificación propuesta por Figueroa-Nazuno et. al. [14]. Las técnicas evaluadas (ver Tabla 2) son de dominio publico en su mayoría y están basadas en diferentes teorías y modelos computacionales: Inteligencia Artificial (Redes Neuronales y Lógica Difusa) [15, 16], Estadística Moderna (Modelos Autoregresivos) [15] y Teoría de Sistemas Dinámicos No Lineales [3, 10].

| *Serie de Tiempo* | *Tipo de Serie de Tiempo.* |
|---|---|
| Sinc | Periódica |
| Qperiodic2 | Cuasiperiódica |
| Mackey-Glass | Caótica |
| Logistic | Caótica |
| Lorentz | Caótica |
| Rossler | Caótica |
| Ikeda | Caótica |
| Henon | Caótica |
| D1 | Compleja |
| Laser | Compleja |
| Down Jones | Compleja |
| Kobe | Compleja |
| ECG | Compleja |
| EEG | Compleja |
| Brownian Motion | Estocástica |
| White Noise | Estocástica |

**Tabla 1.** *Series de Tiempo.*

| *Técnica de Predicción* | *Tipo de Técnica de Predicción.* |
|---|---|
| FIRNet | Heurística (AI): Red Neuronal de Respuesta de Impulso Finito (Generalización de la red estándar de tipo Feedforward) |
| Npredict-PNNGauss | Heurística (AI): Red Neuronal Probabilística (Gauss) |
| Npredict-MLFFN | Heurística (AI): Red Neuronal de Multicapa con alimentación hacia delante |
| Pittnet-FFNBP | Heurística (AI): Red Neuronal de tipo FeedForward con aprendizaje de Backpropagation |
| Pittnet-RBFNBP | Heurística (AI): Red Neuronal de tipo Radial Basis Function con aprendizaje de Backpropagation |
| Matlab-Anfis | Heurística (AI): Modelo Neuro-Difuso |
| Tisean-Predict | Teoría de Sistemas Dinámicos No Lineales: Modelo de ajuste local de funciones lineales |
| Tisean-Polynom | Teoría de Sistemas Dinámicos No Lineales: Modelo de ajuste global de funciones no lineales |
| Tisean-Polynomp | Teoría de Sistemas Dinámicos No Lineales: Construcción de Ansatz de polinomio. |
| Tisean-rbf | Teoría de Sistemas Dinámicos No Lineales: Modelo de ajuste global de funciones no lineales |
| Tisean-nstep | Teoría de Sistemas Dinámicos No Lineales: Construcción de Ansatz para ajuste local lineal |
| Sugimay | Teoría de Sistemas Dinámicos No Lineales: Método simplex mas el uso de la dimensión embebida y vecinos cercanos. |
| Npredict-ARMA | Estadística: Modelo de promedios de autoregresión |

**Tabla 2.** *Técnicas de Predicción.*

La metodología para la realización de los experimentos consistió en la evaluación de las 16 series para cada técnica, el número de experimentos por cada serie de tiempo podía variar desde 1 hasta un máximo de 10, dependiendo de las características de cada técnica, por ejemplo en el caso de la predicción o modelado con redes neuronales se exploraba el posible espacio de soluciones en forma limitada dado que explorar en forma exhaustiva requeriría de una cantidad de experimentos extensa (el número de posibilidades es combinatoria en base a la cantidad de parámetros que pueden ser modificados). La comparación entre los datos originales y los predichos se hizo mediante la graficación de ambas series (original y predicha), el número de datos predichos por serie fue de 50, además se evaluó el error resultante de la predicción exitosa (error medio raíz cuadrada RMSE y el error bias BE) [4], ver Tabla 3.

| *Error* | *Formula* | *Definición.* |
|---|---|---|
| RMSE (Error Medio Raiz Cuadrada) | $RMSE = \left[ \frac{1}{N} \sum_{i=1}^{N} \left( x_i^{\,p} - x_i^{\,o} \right)^2 \right]^{\frac{1}{2}}$ | Representa la medida típica de error de predicción. Valores cerca de cero indican una buena predicción. El término de la diferencia al cuadrado da mayor peso a las discrepancias grandes entre la el valor original y el predicho. |
| BE (Error Bias) | $BE = \frac{1}{N} \sum_{i=1}^{N} \left( x_i^{\,p} - x_i^{\,o} \right)$ | Es una medida de la tendencia a sub o sobre predecir un valor. Un error bias positivo indica una tendencia a sobrepredecir una dato y un valor negativo a subpredecirlo. |

**Tabla 3.** Fórmulas de errores de predicción.

La selección de los parámetros que caracterizan a las series de tiempo se realizo considerando fuesen representativos del comportamiento dinámico del sistema, los parámetros cuantifican características de tipo estadístico, topológico, computacional, espacial y temporal. A continuación se describen los parámetros que fueron calculados con las diferentes herramientas utilizadas:

*Exponente de Lyapunov (EL).* El exponente principal de Lyapunov, mide la evolución de trayectorias vecinas en el espacio fase. Mide la inestabilidad de la dinámica del sistema debido a cambios en sus condiciones iniciales [9, 17, 18, 19].

*Exponente de Hurst (EH).* El exponente de Hurst permite determinar si el fenómeno representado por la serie de tiempo presenta correlaciones de largo alcance (memoria y persistencia de largo alcance). Si la serie de tiempo posee un comportamiento con persistencia de tendencia positiva entonces EH > 0.5, no hay comportamiento predecible si EH = 0, o hay un comportamiento con persistencia de tendencia negativa si EH < 0.5 [20].

*Dimensión de Capacidad (DCA).* La dimensión de capacidad es similar a la dimensión de Hausdorff y mide el grado de auto-similitud del sistema (comportamiento invariante ante cambios de escala espacial), permite cuantificar el grado de heterogeneidad de la señal a diferentes escalas [9].

*Dimensión de Correlación (DCO).* La dimensión de correlación mide la cantidad de veces que la trayectoria del atractor del sistema pasa por una vecindad dada en el espacio fase, cuantifica la correlación espacial local entre puntos de la trayectoria en el espacio fase, sin tomar en cuanto el grado de correlación temporal [9, 10].

*Frecuencia Dominante (FD).* Permite determinar si existe alguna frecuencia característica de la señal, las series de carácter aleatorio poseen espectros amplios sin ninguna frecuencia

dominante, lo mismo se aplica en cierto grado para las series caóticas, las series periódicas y cuasiperiódicas poseen picos bien definidos [20].

*Entropía Espacio-Temporal (EET).* La entropía espacio-temporal, cuantifica de forma global el grado de no correlación de los datos mediante el análisis de recurrencia (Mapas de Recurrencia). A mayor porcentaje de EET menor cantidad de estructura en el mapa de recurrencia y por consiguiente menor correlación entre los datos [6, 7].

*Entropía de Shannon (ES).* La entropía de Shannon, es una medida de la cantidad de información que se obtiene al tomar una medida para especificar el estado del sistema [10].

*Porcentaje de Determinismo (DET).* Permite medir el grado de determinismo en el sistema, por medio del análisis de mapas de recurrencia [6, 7].

*Porcentaje de Recurrencia (REC).* Permite medir el grado de recurrencia (periodicidad y estructura) entre los datos de la serie de tiempo, que es indicativo de patrones repetitivos en la serie de tiempo, por medio del análisis de mapas de recurrencia [6, 7].

*Reglas de Producción (RP).* La generación de gramáticas a partir de una serie de tiempo permite dar una medida de complejidad (computacional) en la cual a mayor número de reglas de producción necesarias para generar una gramática, mayor es la dificultad para la predicción o modelado de la serie de tiempo [11].

## 3. Resultados.

La Figura 1, muestra un ejemplo de predicción con la técnica nstep (TISEAN) basada en la Teoría de Sistemas Dinámicos No Lineales.

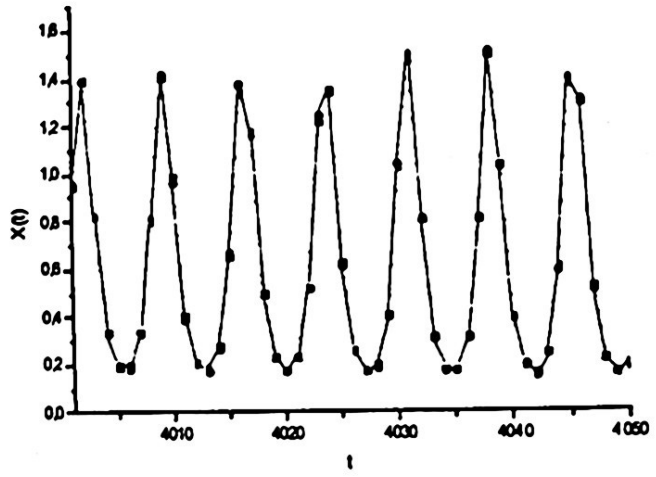



**Figura 1.** *Experimento 1. Serie Láser. Técnica nstep (TISEAN).*

Las Figuras 2 y 3 muestran el comportamiento de los diferentes errores de predicción calculados durante la evaluación de las técnicas. El error RMSE (ver Figura 2), presenta un comportamiento creciente de la serie periódica hacia las series complejas, también se observa un máximo de error en la predicción de las series de tipo caótico tanto para las técnicas heurísticas como para las técnicas basadas en dinámica no lineal. Por su parte el error BE (ver Figura 3), indica poco efecto bias (de sub predicción o sobre predicción) por parte de las técnicas de predicción o modelado para la mayoría de las series evaluadas.

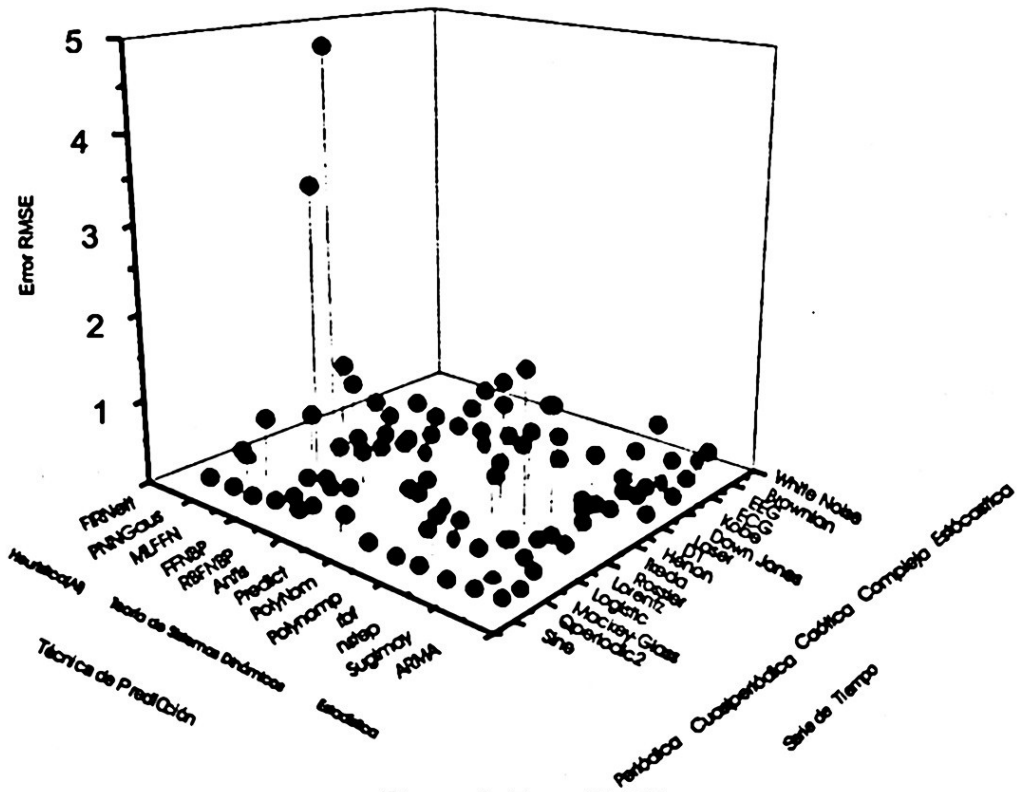



**Figura 2.** *Error RMSE*

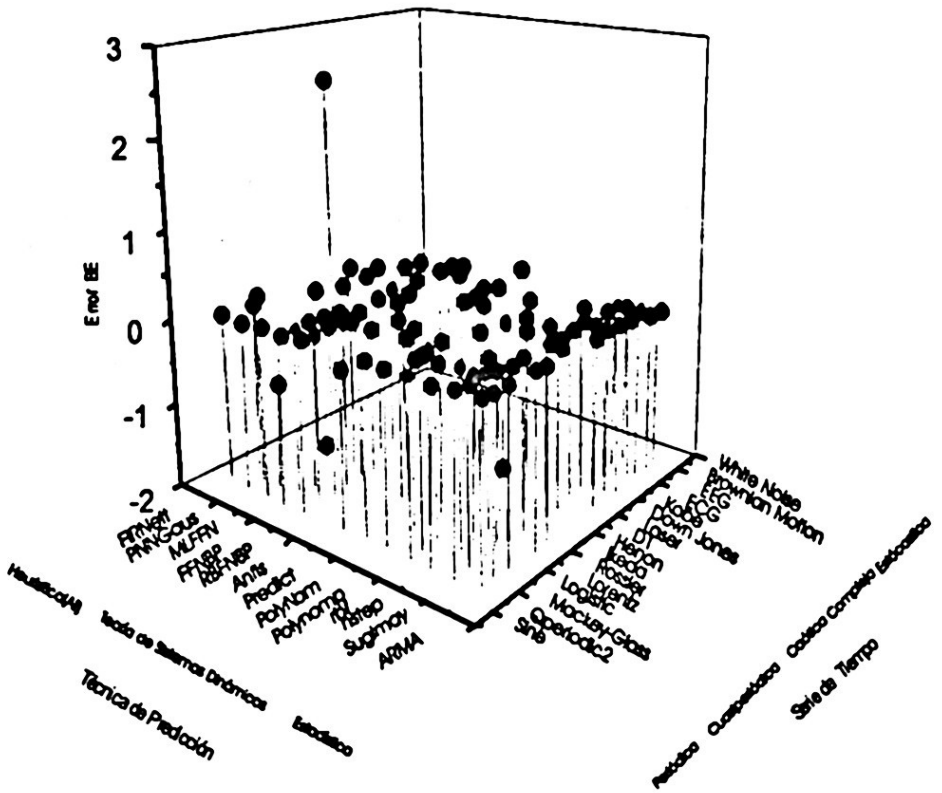



**Figura 3.** *Error BE.*

La Tabla 4 muestra los parámetros calculados para las series de tiempo y la Tabla 5 muestra la escala de dichos parámetros, esta tabla se construyo en base a la integración de diversos criterios definidos en la literatura sobre los parámetros calculados **[6, 7, 8, 20]**, la escala de parámetros se definió considerando el mayor valor y el menor valor obtenido para cada parámetro y su subdivisión en intervalos iguales, se incluye además el impacto en la parámetro y su subdivisión en intervalos iguales, se incluye además el impacto en la dificultad de predicción de la serie de tiempo debido a cada propiedad, el criterio de

asignación de los valores es de carácter empírico (para cada parámetro el valor numérico asignado de degradación en la capacidad de predicción es mayor para el valor del parámetro que denota la mayor complejidad de la serie de tiempo). El objetivo de esta asignación numérica es permitir cuantificar el grado de complejidad de la serie de tiempo en base al conjunto de los parámetros calculados para la misma y poder relacionarlo con la capacidad de las diferentes técnicas para predecir y modelar las series.

| *Serie de Tiempo.* | *Exponente de Hurst* | *Dimensión de Capacidad* | *Dimensión de Correlación* | *Exponente de Lyapunov* | *Frecuencia Dominante* | *Número de Reglas de Producción* | *Entropía (Shannon)* | *Entropía Espacio Temporal (%)* | *Recurrencia (%)* | *Determinismo (%* |
|---|---|---|---|---|---|---|---|---|---|---|
| Sine | 0,94768 | 0,246 | 0,228 | 0,517 | -0,0005 | 27 | 3,711 | 0 | 19,556 | 62,19 |
| Qperiodic2 | 0,00155 | 0,679 | 0,923 | 0,925 | 0,13 | 66 | 2,445 | 0 | 8,148 | 97,760 |
| Mackey-Glass | 0,0781 | 0,983 | 1,025 | 1,481 | 0,0685 | 75 | 0,15914 | 50 | 0,51034 | 37,5178 |
| Logistic | -0,0593 | 0,941 | 0,930 | 0,760 | 0,3795 | 61 | 0,03522 | 78 | 8,61304 | 0,85112 |
| Lorentz | 0,75551 | 0,965 | 1,025 | 0,601 | 0,0005 | 63 | 4,78931 | 57 | 32,8839 | 92,0699 |
| Rossler | 0,56174 | 0,994 | 1,026 | 1,049 | 0,017 | 72 | 2,45897 | 0 | 6,38852 | 88,0434 |
| Ikeda | -0,0195 | 0,983 | 1,019 | 1,452 | 0,3235 | 71 | 0 | 53 | 0,54849 | 0 |
| Henon | -0,03338 | 0,997 | 0,991 | 2,301 | 0,4605 | 69 | 0,0204 | 51 | 9,43468 | 0,63758 |
| D1 | 0,31231 | 0,986 | 2,053 | 1,288 | 0 | 92 | 0,8521 | 39 | 3,30067 | 60,2251 |
| Laser | 0,08799 | 0,961 | 2,096 | 0,949 | 0,1305 | 66 | 2,93549 | 47 | 10,716 | 71,4658 |
| Down Jones | 0,56318 | 0,887 | 1,035 | 0,144 | 0 | 65 | 6,43519 | 2 | 98,8534 | 98,8812 |
| Kobe | 0,01593 | 0,904 | 1,053 | 1,102 | 0,2205 | 79 | 3,74401 | 78 | 57,046 | 44,3674 |
| ECG | 0,74842 | 0,934 | 1,051 | 0,242 | 0 | 53 | 3,51860 | 0 | 13,7886 | 89,8581 |
| EEG | 0,20447 | 0,890 | 1,876 | 1,199 | 0,041 | 82 | 0 | 73 | 2,7157 | 0 |
| Brownian Motion | 0,5374 | 0,971 | 1,103 | 2,043 | 0 | 85 | 2,74038 | 0 | 11,7750 | 66,2786 |
| White Noise | 0,00179 | 0,983 | 2,086 | 1,606 | 0 | 73 | 0 | 79 | 0 | -1 |

**Tabla 4.** *Parámetros de caracterización de las series de tiempo.*

| *Escala de los parámetros (Valor de degradación en la capacidad de predicción)* | *Interpretación* |
|---|---|
| *Exponente de Hurst (Parámetro Estadístico)* | |
| EH > 0.5 (0) | La tendencia local sobre el intervalo continuará (persistencia). La mejor predicción se basa en la extrapolación de la *tendencia positiva* en la serie de tiempo. |
| EH = 0.5 (1) | Valores sucesivos son independientes y la mejor predicción es el último valor medido. |
| EH < 0.5 (0) | La tendencia local puede revertirse (anti persistencia). La mejor predicción tiende al valor medio sobre el intervalo. *Tendencia negativa* en la serie de tiempo. |
| *Dimensión de Capacidad (Parámetro Topológico)* | |
| DC > 0.750 (3) | Autosimilaridad alta |
| 0.500 < DC < 0.750 (2) | Autosimilaridad media |
| DC < 0.500 (1) | Autosimilaridad baja |
| *Dimensión de Correlación (Parámetro Topológico)* | |
| DCO > 1.474 (3) | Cantidad de información alta |
| 0.851 < DCO <1.474 (2) | Cantidad de información media |
| DCO < 0.851 (1) | Cantidad de información baja |
| *Exponente Principal de Lyapunov (Parámetro Topológico)* | |
| EL < 0.863 (1) | Predictibilidad alta |
| 0.863 < EL < 1.582 (2) | Mediana Predictibilidad |
| EL > 1.582 (3) | Predictibilidad baja |
| *Frecuencia Dominante (Parámetro Temporal)* | |
| Existe (0) | Puede haber componente aleatoria |
| No existe (1) | Fuerte componente aleatoria |
| *Número de Reglas de producción (Parámetro Computacional)* | |
| RP > 71 (3) | Alta complejidad computacional y dificultad de predicción |
| 49 < RP < 71 (2) | Mediana Complejidad Computacional y dificultad de predicción |
| RP < 49 (1) | Baja complejidad computacional y dificultad de predicción. |
| *Entropía (Shannon) (Parámetro Computacional)* | |
| ES > 4 (1) | Alto contenido de información en una medición en la serie de tiempo |
| 2 < ES < 4 (2) | Mediano contenido de información en una medición en la serie de tiempo |
| ES < 2 (3) | Bajo contenido de información en una medición en la serie de tiempo |
| *Entropía Espacio Temporal (%) (Parámetro Espacio-Temporal)* | |
| EET > 52 % (3) | Poco estructurado el Mapa de Recurrencia |
| 26% < EET < 52% (2) | Medianamente estructurado el Mapa de Recurrencia |
| EET < 26% (1) | Altamente estructurado el Mapa de Recurrencia |
| *Recurrencia (%) (Parámetro Espacial)* | |
| REC > 66% (1) | Alto grado de correlación en los datos |
| 33% < REC < 66% (2) | Mediano grado de correlación en los datos |
| REC < 33% (3) | Bajo grado de correlación en los datos |
| *Determinismo (%) (Parámetro Espacial)* | |
| DET > 66% (1) | Alto grado de determinismo |
| 33% < DET < 66% (2) | Mediano grado de determinismo |
| DET < 33% (3) | Bajo grado de determinismo |

**Tabla 5.** *Escala de parámetros calculados.*

La Figura 4 muestra la gráfica de la predicción o el modelado de las series de tiempo por parte de las diferentes técnicas evaluadas.

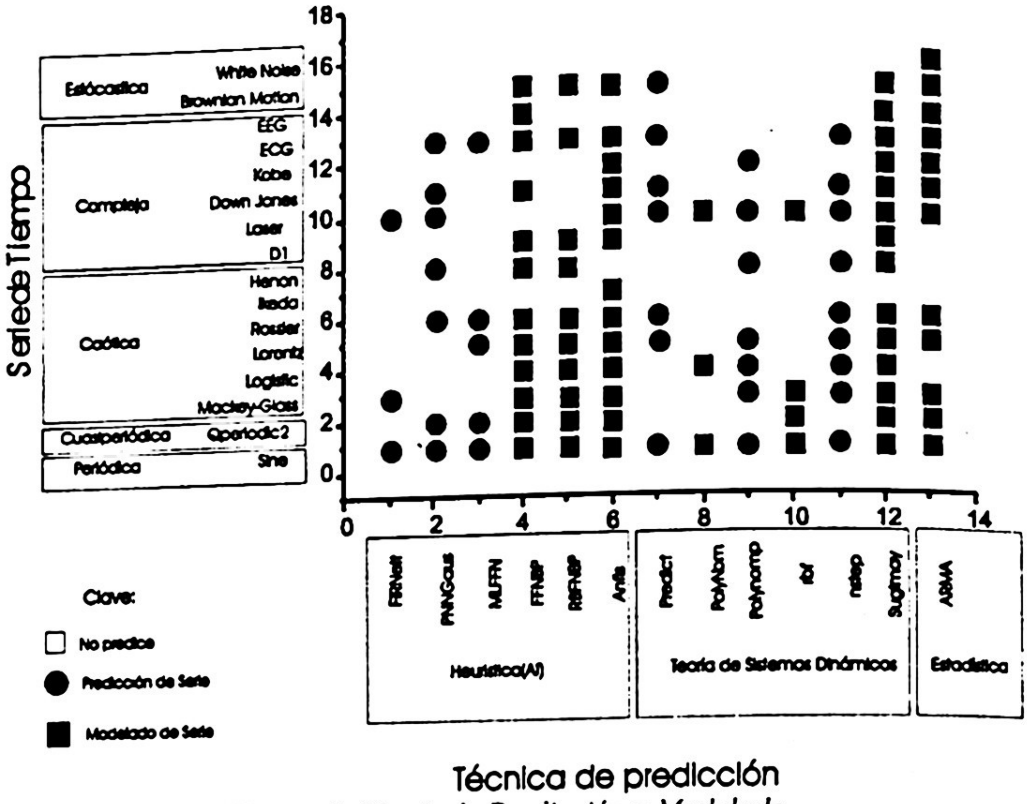



**Figura 4.** *Matriz de Predicción y Modelado.*

Se observa que las técnicas de predicción o modelado más eficientes (por el número de series que logran predecir o modelar) son las basadas en la teoría de sistemas dinámicos no lineales (nstep, sugimay) y de origen heurístico ( PNNGauss, Anfis).

Con el objetivo de cuantificar la capacidad de predicción de las técnicas evaluadas se construyo un Coeficiente de Capacidad de Predicción (*CCP*). El coeficiente esta definido como:

$$CCP = \langle Complejidad\ de\ la\ Serie\ Predicha \rangle + Eficiencia\ de\ Predicción - \langle RMSE \rangle - \langle BE \rangle$$

donde se define:

*<Complejidad de la Serie Predicha>*, como el promedio de la complejidad de las series predichas o modeladas por la técnica evaluada.

*Eficiencia de Predicción*, es la razón del número de series predichas por la técnica entre el número total de series evaluadas.

*<RMSE>*, es el promedio del Error Medio Raíz Cuadrada de todas las series predichas por la técnica evaluada.

*<BE>*, es el promedio del Error Bias de todas las series predichas por la técnica evaluada.

El mayor peso en el coeficiente esta dado por el promedio de la complejidad de las series predichas, seguido por la eficiencia de predicción.

La Figura 5 muestra el Coeficiente de Capacidad de Predicción (*CCP*) vs. Técnica de Predicción o Modelado, con excepción de la técnica Anfis que no permite la generación de archivos con los datos predichos (por lo que no se tiene información sobre el error *RMSE* y *BE* para esta técnica) al resto de las técnicas evaluadas se les pudo determinar su *CCP*. Se observa que de las técnicas que modelan, las que poseen mayor capacidad de predicción

predicción

son de carácter heurístico, sin embargo la técnica Sugimay que se basa en análisis no lineal también tiene una buena capacidad de predicción junto con la técnica ARMA de origen estadístico. En el caso de las técnicas de predicción se observa que la capacidad de predicción es similar tanto para las técnicas heurísticas como para las basadas en sistemas dinámicos no lineales.

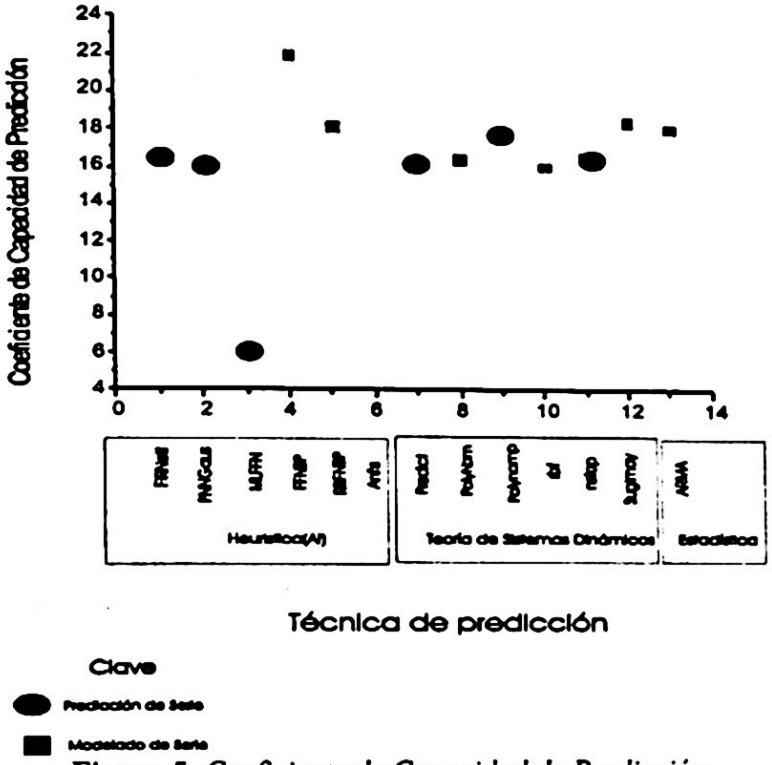



**Figura 5.** *Coeficiente de Capacidad de Predicción.*

Mediante los mapas autoorganizados (*SOM*) se extrajo el conocimiento sobre las relaciones existentes entre el coeficiente de capacidad de predicción de las técnicas y la complejidad de las series de tiempo con el objetivo de generar categorías sobre la capacidad de predicción de las técnicas y sobre la complejidad de las series de tiempo. Los mapas se interpretan de la siguiente forma, en la escala de grises, a menor grado de gris mayor el valor numérico de la propiedad y viceversa, las etiquetas en los mapas corresponden a las unidades que son más representativas de la clase agrupada en dicha zona del mapa, cada propiedad tiene dos mapas uno con etiquetas correspondientes a las series de tiempo predichas y el otro con etiquetas correspondientes a las técnicas de predicción, la correspondencia entre etiquetas en ambos mapas es de uno a uno. Los mapas autoorganizados permiten clasificar la información aun siendo esta incompleta como en el caso de la técnica Anfis en la cual no se tiene información sobre su coeficiente de capacidad de predicción. La Figura 6 muestra un ejemplo de los mapas utilizados en el análisis de extracción de conocimiento, y en la Tabla 6 se presenta la matriz del conocimiento extraído de los mapas autoorganizados.

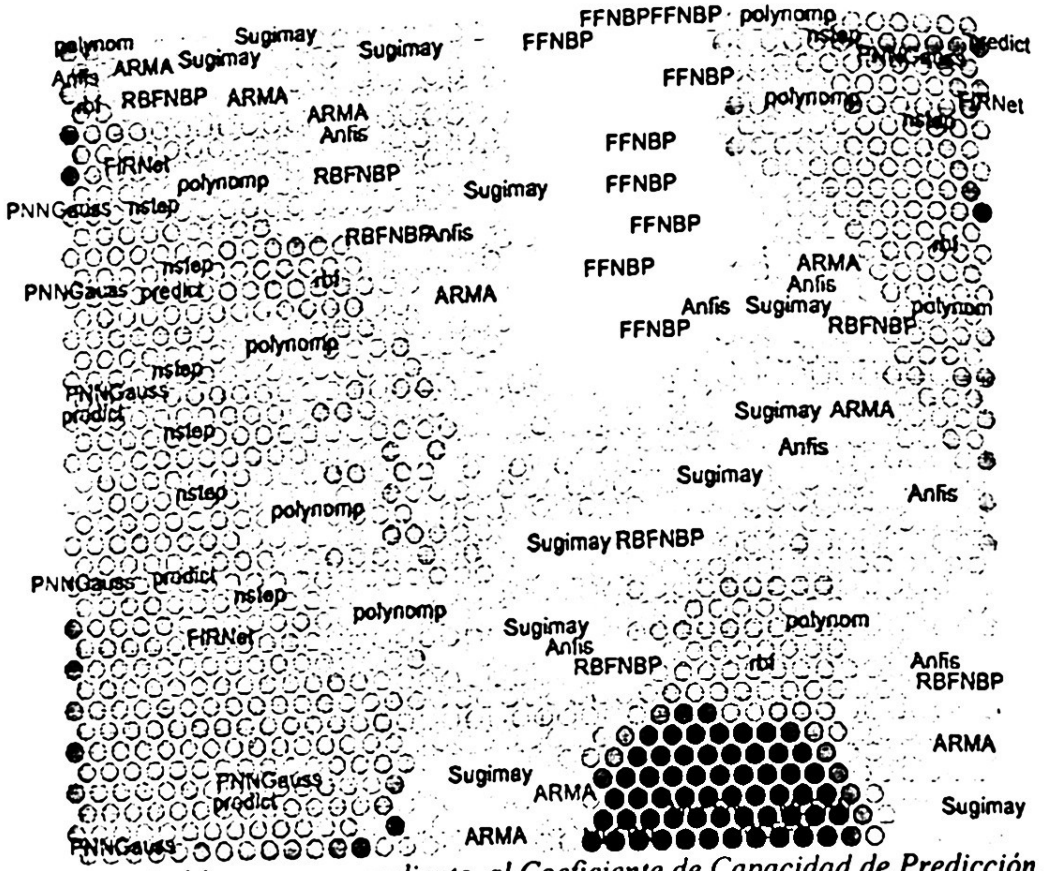



**Figura 6.** *Mapa correspondiente al Coeficiente de Capacidad de Predicción.*

| Técnica de Predicción | Serie de Tiempo | Sinc | Qperiodic2 | Mackey-Glass | Logistic | Lorentz | Rossler | Ikeda | Henon | D1 | Laser | Down Jones | Kobe | ECG | EEG | Brownian Motion | White Noise |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Fimet (P) | CP 2 | COM 1 | | COM 4 | | | | | | | COM 3 | | | | | | |
| PNNGauss (P) | CP 2 | COM 1 | COM 2 | | | | COM 3 | | COM 4 | | COM 3 | COM 1 | | COM 2 | | | |
| MLFFN (P) | CP 1 | COM 1 | COM 2 | | | COM 2 | COM 3 | | | | | | | COM 2 | | | |
| FFNBP (M) | CP 4 | COM 1 | COM 2 | COM 4 | COM 4 | COM 2 | COM 3 | | COM 4 | COM 5 | | COM 1 | | COM 2 | COM 5 | COM 4 | |
| RBFNBP (M) | CP 3 | COM 1 | COM 2 | COM 4 | COM 4 | COM 2 | COM 3 | | COM 4 | COM 5 | | | | COM 2 | | COM 4 | |
| Anfis (M) | CP 3 | COM 1 | COM 2 | COM 4 | COM 4 | COM 2 | COM 3 | COM 5 | | COM 5 | COM 3 | COM 1 | COM 4 | COM 2 | | COM 4 | |
| Predict (P) | CP 2 | COM 1 | | | | COM 2 | COM 3 | | | | COM 3 | COM 1 | | COM 2 | | COM 4 | |
| Polynom (M) | CP 2 | COM 1 | | | COM 4 | | | | | | COM 3 | | | | | | |
| Polynomp (P) | CP 2 | COM 1 | | COM 4 | COM 4 | COM 2 | | | COM 4 | | COM 3 | | COM 4 | | | | |
| rbf (M) | CP 2 | COM 1 | COM 2 | COM 4 | | | | | | | COM 3 | | | | | | |
| nstep (P) | CP 2 | COM 1 | | COM 4 | COM 4 | COM 2 | COM 3 | | COM 4 | | COM 3 | COM 1 | | COM 2 | | | |
| Sugimay (M) | CP 3 | COM 1 | COM 2 | COM 4 | COM 4 | COM 2 | COM 3 | | COM 4 | COM 5 | COM 3 | COM 1 | COM 4 | COM 2 | COM 5 | COM 4 | |
| ARMA (M) | CP 3 | COM 1 | COM 2 | COM 4 | | COM 2 | COM 3 | | | | COM 3 | COM 1 | COM 4 | COM 2 | COM 5 | COM 4 | COM 6 |

**Tabla 6.** *Matriz de Conocimiento de Categorias de Capacidad de Predicción de Técnicas (CP) y Complejidad de Serie de Tiempo (COM).*

Del análisis de los mapas se identificaron 6 categorías de Complejidad y 4 de Capacidad de Predicción, donde la categoría de menor valor corresponde a menor complejidad (COM 1) o capacidad de predicción (CP 1) respectivamente y el orden es creciente en complejidad o capacidad de predicción. Las técnicas con mayor capacidad de predicción (categoría CP 3 y CP 4) modelan las series de tiempo, y en el caso de las técnicas que predicen su capacidad de predicción corresponde a la categoría CP 1 y CP 2. Se observa a partir de la categorización que no hay una relación lineal entre la complejidad de la serie y la dificultad para ser predicha por las técnicas.

## 4. Conclusiones.

La relación entre la complejidad de la serie de tiempo y la dificultad de predicción de la misma no es lineal, la clasificación inicial de las series en base a su comportamiento dominante (periódico, cuasiperiódico, caótico, complejo o estocástico) no es representativo de la dificultad para predecir o modelar la serie de tiempo, el utilizar al conjunto de parámetros que caracterizan a la serie de tiempo nos da mayor información para cuantificar su grado de complejidad y también es posible que permita identificar relaciones entre conjuntos de parámetros y la dificultad o facilidad para predecir dichas series con diferentes técnicas, se esta trabajando en lo anterior por parte de los autores.

No se encontró una técnica de predicción o modelado que fuese superior en su capacidad de predicción en comparación con las otras técnicas, tanto las técnicas heurísticas como las basadas en dinámica no lineal presentan una capacidad de predicción similar, en cuanto a las técnicas de modelado las técnicas de origen heurístico presentan una capacidad mayor que las técnicas basadas en dinámica no lineal. La generación de las categorías de complejidad y capacidad de predicción permiten tener una métrica estándar para la comparación de nuevas técnicas de predicción y series de tiempo, además dicha información puede ser utilizada en el desarrollo de sistemas expertos que identifiquen la mejor o mejores técnicas para la predicción y modelado de las series de tiempo en base a su caracterización y su posterior categorización.

## Referencias

[1] Suhigara, G. & May, R. M. (1990). Nonlinear forecasting as a way of distinguishing chaos from measurement error in time series, Nature, 344, 734-740.

[2] Wan, E. (1994). Time Series Prediction Using a Neural Network with Embedded Tapped Delay-Lines, *Time Series Prediction: Forecasting the Future and Understanding the Past*, Addison-Wesley.

[3] Kantz, H. & Schreiber T. (2000). *Nonlinear Time Series Analysis*, Cambridge: Cambridge University Press.

[4] Diebold, F. X. (2001). *Elements of Forecasting*, South-Western Thomson Learning.

[5] Espinosa-Contreras, A., Suárez-Núñez, R. & Figueroa-Nazuno, J. (1999). Análisis del Comportamiento de Diferentes Sismos en la Ciudad de México con Técnicas de la Dinámica No-Lineal, XLII Congreso Nacional de Física, Villahermosa, Tabasco.

[6] Espinosa-Contreras, A. & Figueroa-Nazuno, J.(2000). Análisis del Comportamiento de la Pérdida de Paquetes en la Red Internet con Técnicas de la Dinámica No-Lineal, Congreso Internacional de Computación CIC 2000. México D. F..

[7] Acevedo-Mosqueda, M. E., León-Vega, C. G. & Figueroa-Nazuno, J. (2001). Medición de Complejidad de Series de Tiempo. XLIV Congreso Nacional de Física, Morelia, Michoacán.

[8] Baker, G. L. & Gollub, J. P. (1993). *Chaotic Dynamics an introduction*, Cambridge: Cambridge University Press.

[9] Williams, G. P. (1997). *Chaos Theory Tamed*, Washington: Joseph Henry Press.

[10] Urbach, R. M. A. (2000). *Footprints of Chaos in The Markets*, Financial Times-Prentice Hall.

[11] Menchaca-Méndez, R., Sánchez-Rodríguez, C., Espinosa-Contreras A. & Figueroa-Nazuno, J. (2000). Modelado y Caracterización de Series de Tiempo por medio de Análisis Gramatical. XLIII Congreso Nacional de Física, Puebla, Puebla.

[12] Deboeck, G. & Kohonen, T. (Eds) (1998). *Visual Explorations in Finance with Self-Organizing Maps*, New York: Springer-Verlag.

[13] Kohonen, T., Hynninen, J., Kangas, J. & Laaksonen, J. (1995). SOM_PACK: Manual de Usuario.

[14] Figueroa-Nazuno, J. (1999). Modelos matemáticos de procesos de contaminación atmosférica. Clean Air 99, Mexico City.

[15] Masters, T. (1995). *Neural, Novel & Hybrid Algorithms for Time Series Prediction*, John Wiley & Sons.

[16] Martín del Brío, B. & Sanz Molina, A. (2002). *Redes Neuronales y Sistemas Difusos*, Alfaomega-RA-MA.

[17] Ruelle, D. (1996). *Chaotic evolution and strange attractors*, Cambridge: Cambridge University Press.

[18] Hilborn, R. C. (2000). *Chaos and Nonlinear Dynamics*, Oxford: Oxford University Press.

[19] Ott, E. (2000). *Chaos in Dynamical Systems*, Cambridge: Cambridge University Press.

[20] Goltz, C. (1997). *Fractal and Chaotic Properties of Earthquakes*, New York: Springer-Verlag.

# TEORÍA DE NÚMEROS: EL PROBLEMA DE COLLATZ.

**E. Bautista-Thompson, A. Villanueva-Becerril & J. Figueroa-Nazuno**
Centro de Investigación en Computación.
Instituto Politécnico Nacional.
Unidad Profesional "Adolfo López Mateos" Zacatenco
Col. Lindavista, C.P. 07738
México D. F.
ebautista@correo.cic.ipn.mx jfn@cic.ipn.mx

**Resumen**

El problema de teoría de números planteado por L. Collatz en 1937 genera series numéricas (Series de Hailstone) que presentan comportamiento complejo difícil de caracterizar en forma analítica. Se presentan los resultados de aplicar técnicas para el Análisis de Sistemas Dinámicos No Lineales, a series obtenidas a partir de números enteros aleatorios entre $1\times10^{50}$ y $1\times10^{150}$, con el objetivo de identificar patrones característicos en dichas series.

**Palabras clave:** Series de Hailstone, Mapas de Recurrencia, Análisis de Gramáticas.

## 1. Introducción.

Tradicionalmente, El Reconocimiento y Clasificación de Patrones son Teorías y Técnicas que se aplican para estudiar si en algún tipo de "señales" o datos registrados de alguna forma, se pueden encontrar Patrones o clases de Patrones. Normalmente se presupone y se sabe de su existencia y los resultados de esos análisis nos dan información o nos hablan de los fenómenos que están atrás de esas señales. Existen cierto tipo de situaciones como las que se analizan en este trabajo, en donde posiblemente las técnicas de análisis de patrones nos ayudan a entender cuáles son los mecanismos o fenómenos que pueden existir o no, en estas series matemáticas. Nuestro interés es aplicar las técnicas modernas de reconocimiento de patrones, en especial las del estudio de los sistemas dinámicos no lineales, para conocer un poco más estas series numéricas, ya que las técnicas analíticas hasta el momento nos han dado muy poca información.

A pesar de que se han desarrollado una gran cantidad de trabajos sobre el Problema de Collatz, (también conocido como Series de Hailstone, Algoritmo de Hasse, Algoritmo de Siracusa entre otros nombres) no ha sido posible resolver dicho problema [1, 2]; la simplicidad de las reglas matemáticas del mismo, contrasta con la dificultad para estudiar en forma analítica el comportamiento de las series numéricas que genera. El problema de Collatz se puede plantear de la siguiente forma, sea el algoritmo definido por la función $h : N \rightarrow N$ sobre el conjunto de los enteros positivos:

$$h(n) = \begin{cases} \dfrac{n}{2} & si \;\; n \;\; es \;\; par \\ 3n+1 & si \;\; n \;\; es \;\; impar \end{cases}$$

donde $n$ es inicialmente cualquier entero positivo. Para generar la secuencia sea $h^k(n)$ la *kth* iteración de *h(n)*. Una secuencia de números de Hailstone es generada mediante un ciclo de retroalimentación donde:

$$h^k(n) = h(h^{k-1}(n)) \text{ para } k \in N$$

y puede ser representada como

$$H(n) = \{h^k(n)\} \; k = 1, 2, 3, ....$$

Ejemplos de conjeturas sobre el comportamiento de estas series que no ha sido posible demostrar en forma analítica son: que para todo número entero se genere una secuencia que converge a 1, y que existan ciclos característicos de números en la Serie de Hailstone **[2]**. La dificultad de este problema llevo al matemático Paul Erdös a afirmar: *"Mathematics is not yet ready for such problems"* **[1, 2]**.

Se ha comprobado numéricamente que el algoritmo genera secuencias de números que convergen a 1 para todos los números iniciales (enteros positivos) $\leq 3 \times 2^{53} \approx 2.702 \times 10^{16}$ **[3]**. El que se pueda generar secuencias de números que presentan un comportamiento complejo a partir de un algoritmo simple, y que no haya sido posible caracterizar dichas secuencias de números en forma analítica, es lo que ha motivado el presente trabajo, en el cual se aplican técnicas utilizadas para el análisis de series de tiempo de sistemas dinámicos no lineales a las secuencias de números, obtenidas mediante el algoritmo ya descrito, que son tratadas como si fuesen series de tiempo generadas por un algoritmo determinista **[4, 5]**. El objetivo es estudiar un conjunto de parámetros e identificar patrones, que caractericen a las Series de Hailstone.

## 2. Técnicas de Análisis.

En esta sección se describen las técnicas que se utilizaron para el estudio de las Series de Hailstone. La primera técnica utilizada fue el Análisis de Mapas de Recurrencia, cual genera un mapa (patrón) de la serie numérica. El mapa es una representación gráfica de la integral de correlación y hace posible visualizar la dependencia temporal espacial de los datos utilizando una medida de distancia euclidiana entre los datos de serie que es vista por medio de una escala de grises. El Mapa de Recurrencia permite detectar gráficamente (ver Figura 2 y 3) patrones escondidos y cambios estructurales los datos, o identificar similitudes entre patrones de diferentes series, a su vez permite calcular los siguientes parámetros: Recurrencia, Determinismo y Entropía de Shannon **[5, 6, 7]**; los cuales son descritos a continuación:

*Porcentaje de Recurrencia.* Permite medir el grado de recurrencia (periodicidad estructura) entre los datos de la serie, que es indicativo de patrones repetitivos en misma **[5, 6]**.

*Porcentaje de Determinismo.* Permite medir el grado de determinismo en el sistema, medio del análisis de mapas de recurrencia **[5, 6]**.

*Entropía de Shannon.* La entropía de Shannon, es una medida de la cantidad de información que se obtiene al tomar una medida para especificar el estado del sistema [7].

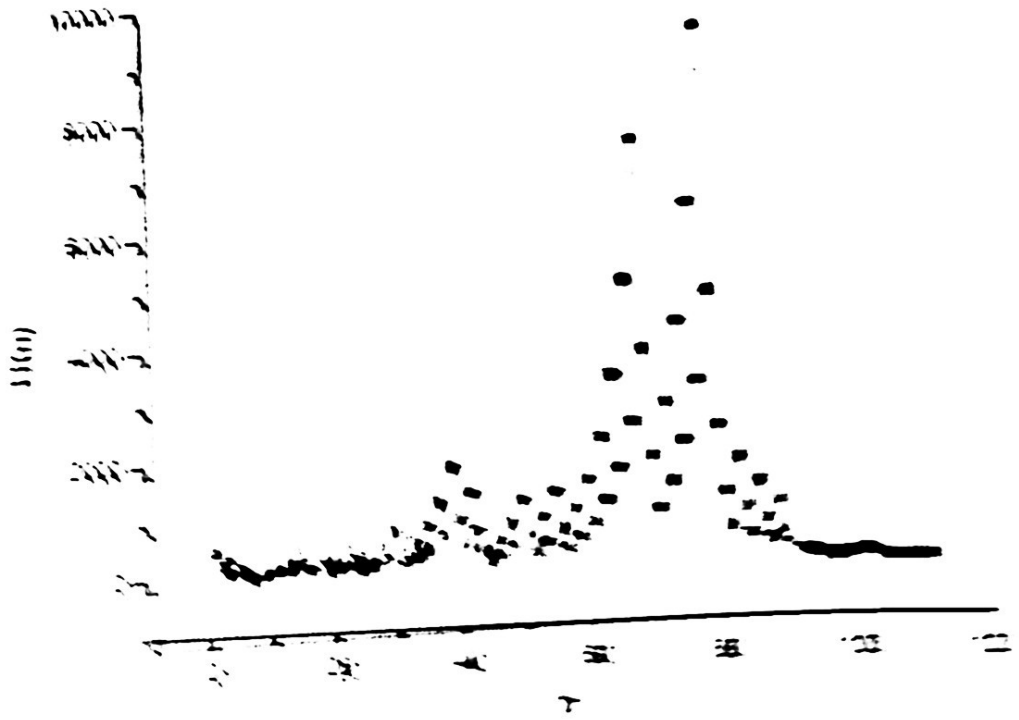
La otra técnica utilizada fue el Análisis de Gramáticas, la cual genera reglas de producción para la serie numérica. El Análisis de Gramáticas consiste en la generación de una gramática libre de contexto en la cual el conjunto de símbolos terminales (reglas de producción), representa la diferencia entre valores contiguos de la serie numérica. De esta forma, si una parte de la serie presenta el mismo comportamiento en repetidas ocasiones, éste quedará representado por una sola variable de la gramática, esta variable a su vez, puede formar parte de la definición de otra variable y así de manera recursiva. La generación de gramáticas a partir de una serie de tiempo permite dar una medida de complejidad (computacional) en la cual, a mayor número de reglas de producción necesarias para generar una gramática, mayor es la complejidad de la serie numérica [8, 9].

## 3. Generación de las Series Numéricas.

Se construyeron series de números a partir de semillas generadas en forma aleatoria. Las series de tiempo utilizadas corresponden a semillas dentro del rango de $1\times10^{50}$ a $1\times10^{150}$, en dicho rango el número de datos para cada serie varía entre 1000 a 5000 datos. Se seleccionaron al azar 10 series por cada incremento de 1000 de tal forma que se obtuvieron 5 grupos de 10 series cada uno. Los extremos del conjunto de 50, fueron series numéricas con 1067 y 5978 datos (ver Figura 1).

Para calcular el mapa de recurrencia y los parámetros derivados, se tuvieron que recortar las Series de Hailstone ya que las fluctuaciones de la secuencia de números son considerables (varios órdenes de magnitud entre valores contiguos) y el tamaño de los valores es sumamente grande (por ejemplo: números enteros del orden de $10^{19}$ ), el recorte se hizo en función de evaluar la serie y observar a partir de qué valor era posible calcular numéricamente el mapa de recurrencia y sus parámetros asociados, lo mismo se hizo para el cálculo de las reglas de producción de gramáticas. Finalmente, el conjunto de 50 series tenía entre 331 y 755 datos por serie.

Una muestra de las series, formada por una de cada grupo, fue recortada nuevamente hasta quedar con los últimos 100 elementos, y se les aplicó el Análisis de Mapa de Recurrencia y de Gramáticas, para estudiar el comportamiento de las series a diferentes escalas. La Figura 1 muestra un ejemplo de una Serie de Hailstone.



[illegible]

## [illegible] de los Resultados

[illegible]

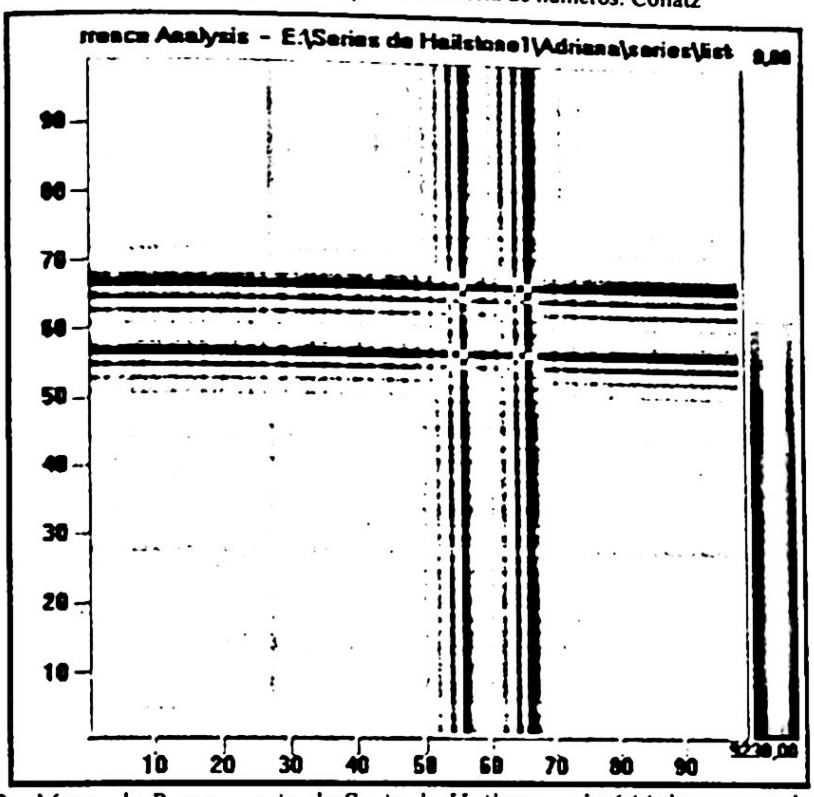



**Figura 2.** *Mapa de Recurrencia de Serie de Hailstone de 644 datos para los últimos 100 datos.*

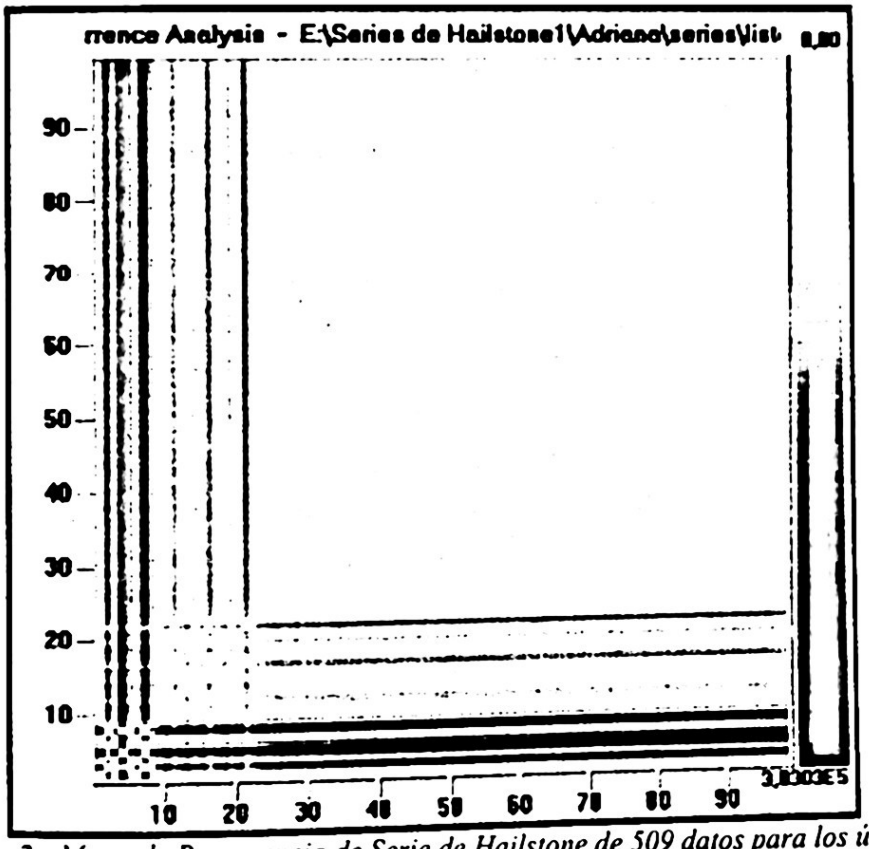



**Figura 3.** *Mapa de Recurrencia de Serie de Hailstone de 509 datos para los últimos 100 datos.*

La Tabla 1 muestra los resultados obtenidos (promedio, desviación estándar, valor máximo y mínimo) del cálculo de los parámetros de Recurrencia, Determinismo, Entropía de Shannon y el Número de Reglas de Producción para las 50 series numéricas seleccionadas para realizar este trabajo.

| | *Recurrencia (%)* | *Determinismo (%)* | *Entropía (%)* | *# Reglas de Producción* |
|---|---|---|---|---|
| *Promedio* | 97,363 | 96,296 | 6,306 | 15,86 |
| *Desviación Estándar* | 3,0531195 | 3,92280357 | 0,24648981 | 5,27995826 |
| *Valor Máximo* | 99,805 | 99,089 | 6,490 | 32 |
| *Valor Mínimo* | 86,476 | 85,629 | 5,594 | 8 |

**Tabla 1.** *Parámetros obtenidos del Análisis de Mapas de Recurrencia.*

La recurrencia promedio es cercana al 100%, esto indica que hay un alto grado de patrones repetitivos. El determinismo también es cercano al 100%, lo que se refleja en lo estructurado del mapa, esto indica que el generador de la señal es determinista. La entropía de Shannon es de 6.3% en promedio, la entropía tan baja indica que a pesar de poseer un comportamiento determinista, la ganancia en información que puede obtenerse de una medición en la serie de tiempo es pobre, consecuencia del comportamiento complejo de la serie (fluctuaciones de varios órdenes de magnitud y a diferentes escalas a lo largo de la serie).

Con respecto al análisis de gramáticas, se encontró que el número de reglas de producción no es alto, en promedio aproximadamente 16 reglas, (el número máximo de reglas de producción para las series fue de 32) en comparación con análisis de gramáticas realizados a diversos tipos de series de tiempo (por ejemplo, serie de electroencefalograma ECG: 82 reglas, serie económica de Down Jones: 65 reglas) [10], lo cual indica que la Serie de Hailstone posee una complejidad computacional baja. Sin embargo se observa lo siguiente: la Figura 4 muestra que ocurre un crecimiento rápido de la cantidad de reglas de producción generadas, lo que corresponde a las fluctuaciones de la serie de Hailstone (ver la Figura 5), y posteriormente disminuye el ritmo de generación de reglas de producción. Este mismo comportamiento se repite para diferentes segmentos de la serie de tiempo (autosimilitud), tal como se muestra en la Figuras 6 y 7 para los últimos 100 datos de la misma serie.

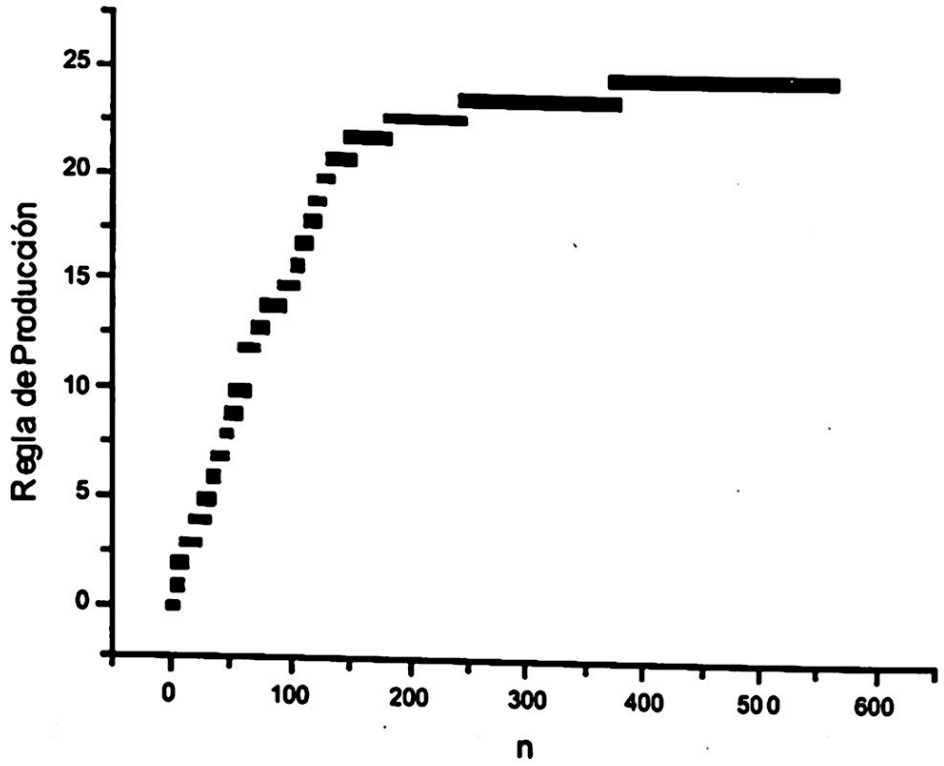



**Figura 4.** *Gráfica de la evolución de las Reglas de Producción para la Serie de Hailstone con n = 557 datos.*

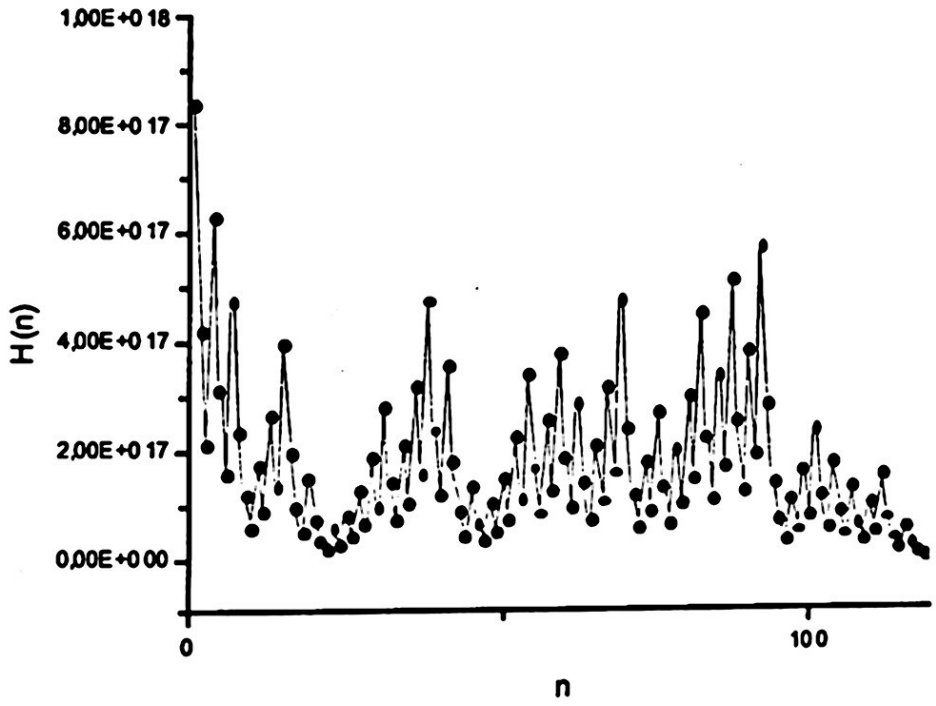



**Figura 5.** *Gráfica de la Serie de Hailstone con n = 557 datos, primeros 120 datos.*

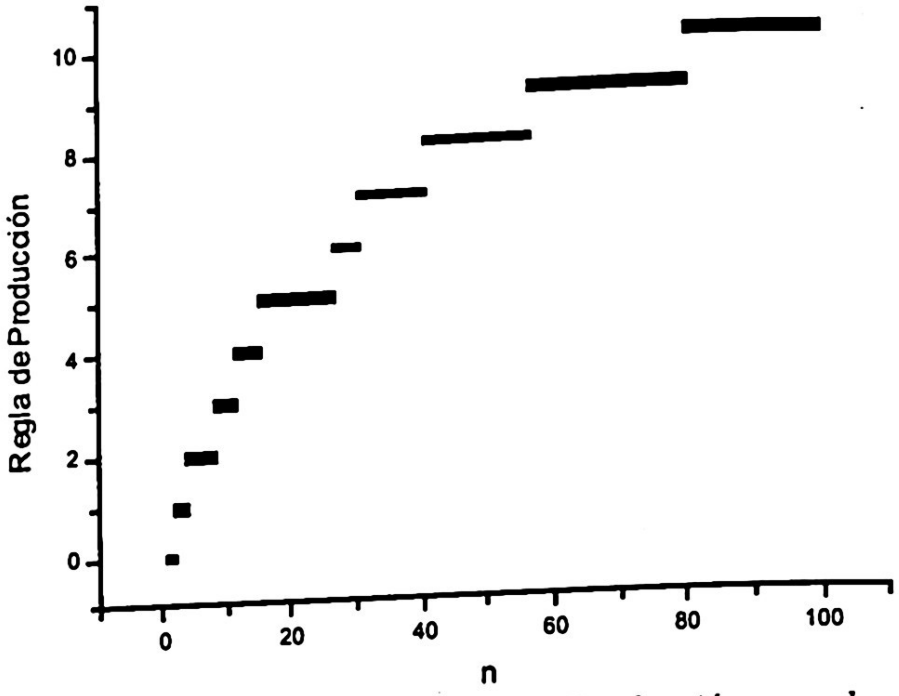



**Figura 6.** *Gráfica de la evolución de las Reglas de Producción para la misma Serie de Hailstone de 557 datos tomando los últimos n = 100 datos.*

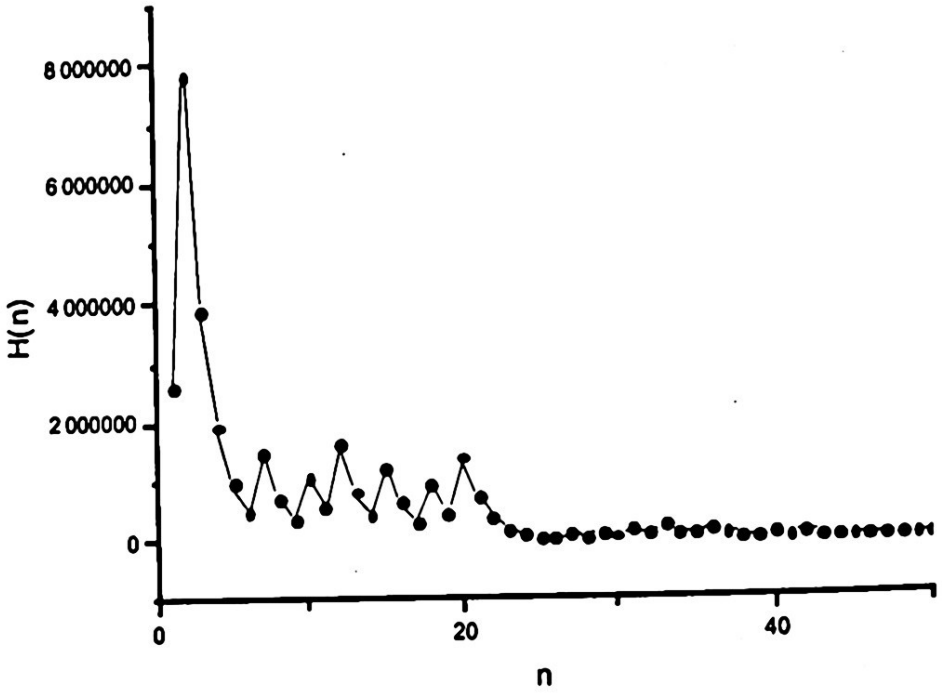



**Figura 7.** *Gráfica de la Serie de Hailstone de 557 datos tomando solo los últimos n = 100 datos, se muestran 50 datos.*

## 5. Conclusiones.

Se probó en forma experimental que series numéricas con comportamiento no suave (fluctuaciones de varios órdenes de magnitud, convergencia lenta) que son difíciles estudiar en forma analítica, pueden ser caracterizadas con técnicas de análisis no lineal y es posible identificar patrones de comportamiento de dichas series.

Mediante los Mapas de Recurrencia se identificó un patrón característico que muestra que la Serie de Hailstone es generada en forma determinista y que existe una estructura que se repite a lo largo de la serie, dicha estructura consiste en las fluctuaciones de varios órdenes de magnitud las cuales dominan en la generación del patrón, dicho comportamiento se observa para diferentes escalas a lo largo de la serie; esto es un fenómeno de autosimilitud.

Analizando los resultados de las series no se observó relación alguna entre la semilla generadora de la serie y los patrones encontrados, es decir los Mapas de Recurrencia son similares para todas las series.

A pesar de que las series poseen un alto grado de recurrencia y determinismo y baja complejidad computacional (expresado en el bajo número de reglas de producción), las fluctuaciones extremas de las series ocasionan la baja Entropía de Shannon observada.

Los resultados en general nos indican que es posible utilizar técnicas de análisis no lineal, en el estudio de estas series numéricas y el hecho de que existan fenómenos de autosimilitud, alta recurrencia y determinismo y baja complejidad computacional son resultados importantes que no se tenían con el estudio exclusivamente analítico de El Problema de Collatz.

## Referencias


[1] Pickover, C. A. (1990). *Computers, Pattern, Chaos and Beauty. Graphics from an Unseen World*, New York: ST. MARTIN'S PRESS.

[2] Lagarias, J. C. (1996). The 3x+1 Problem and its Generalizations, ATT Bell Laboratories Internal Report.

[3] Oliveira e Silva, T. (1999). Maximum Excursion and Stopping Time Record-Holders for the 3x + 1 Problem: Computational Results, Math. Comput. 68, 371-384.

[4] Espinosa-Contreras, A., Suárez-Núñez, R. & Figueroa-Nazuno, J. (1999). Análisis del Comportamiento de Diferentes Sismos en la Ciudad de México con Técnicas de la Dinámica No-Lineal. XLII Congreso Nacional de Física, Villahermosa, Tabasco.

[5] Espinosa-Contreras, A. & Figueroa-Nazuno, J. (2000). Análisis del Comportamiento de la Pérdida de Paquetes en la Red Internet con Técnicas de la Dinámica No-Lineal. Congreso Internacional de Computación CIC 2000, México D. F..

[6] Acevedo-Mosqueda, M. E., León-Vega, C. G. & Figueroa-Nazuno, J. (2001). Medición de Complejidad de Series de Tiempo, XLIV Congreso Nacional de Física, Morelia, Michoacán.

[7] Urbach, R. M. A. (2000). *Footprints of Chaos in The Markets*, Financial Times-Prentice Hall.

[8] Menchaca-Méndez R., Sánchez-Rodríguez C., Espinosa-Contreras, A. & Figueroa-Nazuno, J. (2000). Modelado y Caracterización de Series de Tiempo por medio de Análisis Gramatical, XLIII Congreso Nacional de Física, Puebla, Puebla.

[9] Menchaca-Méndez, R., Sánchez-Rodríguez, C. & Figueroa-Nazuno, J. (2001). Predicción de Series de Tiempo Mediante Análisis Gramatical, XLIV Congreso Nacional de Física, Morelia, Michoacán.

[10] Bautista-Thompson, E. & Figueroa-Nazuno, J. (2002). Análisis de Parámetros que Caracterizan la Dificultad de Predicción en Series de Tiempo, XLV Congreso Nacional de Física, León, Guanajuato.

## Agradecimientos

Los autores desean expresar su agradecimiento al compañero Ricardo Menchaca por apoyo con la Técnica de Análisis Gramatical.

# Recognition techniques

J. M. Medina-Apodaca°, J. Figueroa-Nazuno° & S. García-Benitez*
°Centro de Investigación en Computación, Instituto Politécnico Nacional.
*Instituto de Ingeniería, UNAM.

**Abstract**

In this paper we present a time series modeling tool which enables their easy recognition and treatment using Recurrence Plots and Eigenfaces. We obtain the recurrence plots of a set of time series, and take the values of each plot as a very long vector, as we would with the pixels of a face image when using the Eigenfaces technique. This way, we build the face space of the plots. By expressing each recurrence plot as a point in the face space, we can represent the plots with a short vector. These vectors can be used as input for an analysis process. Using the inverse path, we can recover a time series represented by the output of the analysis.

**Keywords:** Recurrence Plots, Eigenfaces, pattern recognition, time series analysis.

## 1. Introduction.

Time Series modeling and prediction are up to now hard tasks for real phenomena. It is often practically impossible to find a mathematical function which models a real life time series. Furthermore, most of the events in nature and society have too many variables influencing the observed behavior, while several mathematical methods such as autoregression and correlation are limited to linear signals.

One of the most often used methods for data analysis is the Fourier Transform. When used, this transform gives the input signal in the domain of frequency, showing the influence of each frequency component in the whole signal. However, Fourier analysis carries a loss of details about some features of the signals, such as those related to the periodicity or maximums and minimums.

Beside the purely mathematical methods for data analysis are the computational tools. Currently, one of the most powerful and most often used tools are Artificial Neural Networks (ANNs), which can learn the behavior of a time series adapting to a training set of data. However, they have several drawbacks when used for complex models: the correct parameters are difficult to find, the networks can be overfitted to the training set being useless for other data, and a relatively large amount of inputs and/or outputs makes it necessary to count with huge amounts of computational power and time to train the network.

The method described in this paper uses recurrence plots to multiply the information contained in the signal, making some subtle details of the signal visible even for a human observer. Also, recurrence plots may contain information about the underlying process which generated the signal, such as the number of factors involved on its phase space.

A face recognition technique called "eigenfaces" is used to express a family of signals as a combination of others. The result is that a large signal can be expressed by a vector reduced number of entries.

## 2. Recurrence Plots.

Recurrence plots are used to reveal several characteristics of a time series, such degree of aperiodicity and stationarity [1]. Let S be a series with N terms

$$S_0, S_1, S_2, S_3, S_4, S_5, \ldots S_i, \ldots S_{N-1}$$

Let V be the set of all the vectors

$$v_i = (S_i, S_{i+d}, S_{i+2d}, \ldots S_{i+Dd})$$

Where D is called the embedding dimension and d is called the delay. The embedd dimension is formally the number of variables involved in the phase space diagram underlying process which generated the series. However, this dimension can be forced 1 without affecting too much the model.

The Recurrence plot is a map where the value of the point (i, j) is the Euclidean distar between the vectors $v_i$ and $v_j$. This is, the point (i, j) in the plot is equal to $\|vi - vj\|$.

Normally, the values in the map are associated with a color code, so it can be representec a graphical way. There are different color codes that can be assigned to the values plot. The most commonly used is the spectrum, in which the color range runs from white red, green, blue and black. A white point represents a value of 0, and a black represents the longest possible value.

Figure 1 shows the recurrence plot of a sine signal. Note the well defined characteristics of a periodical signal. Since the distance is a commutative measure distance from vector $v_i$ to $v_j$ is the same than the distance from vector $v_j$ to $v_i$), recurre plots are always symmetrical respect to the diagonal.

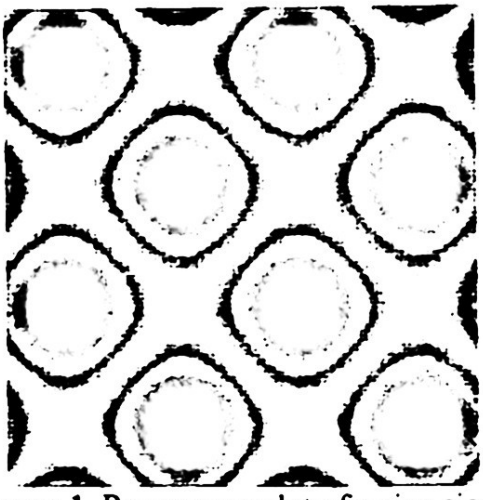


**Figure 1**. Recurrence plot of a sine signal

Figure 2 shows the recurrence plot of a white noise signal. As this is a random signal there are not well defined patterns as in the case of the sine signal. However, the plot is still symmetric respect to the diagonal.

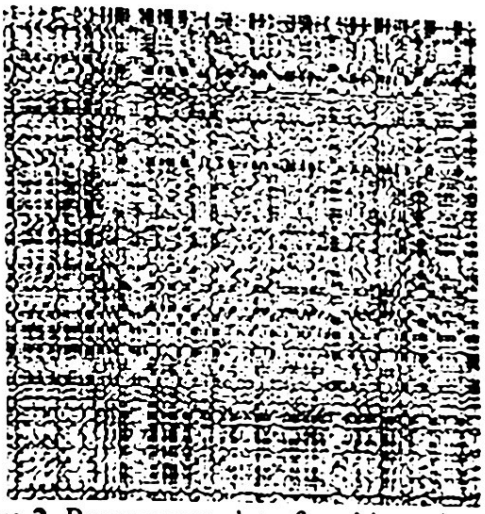


**Figure 2.** Recurrence plot of a white noise signal

Recurrence plots are useful for identifying special features of a time series, which could hardly be seen in a simple graph. For example, in a Recurrence Plot its easy to identify some kind of periodicity and relations among different parts of the series, even if this is related to a complex phenomena.

## 3. Eigenfaces.

Eigenfaces is a face recognition technique based on extracting the most important features of a set of face images, and use them to recognize new face images presented to the system.

In mathematical terms, this technique consists on obtaining the eigenvectors of the covariance matrix whose columns are a set of face images expressed as vectors [2]. These vectors are called the eigenfaces of the set of images.

Each face image of the training set can be exactly represented by a linear combination of the resulting eigenfaces. In principle, the number of face images in the training set is equal to the number of eigenfaces. However, the faces can be also represented using the best eigenfaces of the set, i.e. those which have the largest eigenvalues, and which therefore account for the most variance within the set of images. This way, a multitude of face images can be reconstructed by weighted sums of a small collection of characteristic images.

Once normalized, the complete set of eignenfaces form an orthogonal incomplete basis for a very long dimension space. This space is called the face space. Each face from the set can be totally expressed as a point in the face space. The basic idea of the eigenfaces technique is that, since the eigenfaces are orthogonal vectors and the face images are similar (they are all face images of the persons whose face is already in the training set), the faces which are not in the original set can be approximated by a point in the face space.

The eigenfaces technique is able to recognize a face even with small differences in angle, differences in age, and different illumination environments. This can also easily decide whether or not an image is a face.

Given a point in the face space, it is easy to return to the original face. Since the vector expresses a linear combination of the eigenfaces, the return can be achieved simply multiplying each component of the vector by the corresponding eigenface.

Although the original eigenfaces technique proposed to compute the covariance matrix, technique gives better results by making up the matrix with the original values of the as its columns, and computing its eigenfaces.

## 4. The Developed Technique.

The technique presented here can be applied as preprocessing and post-processing stages for other time series analysis techniques. The preprocessing stage is shown in figure Given a set of time series, we compute the recurrence plot for each one and find the space corresponding to the computed set of recurrence plots.

To find the face space we take the set of recurrence plots and put it in a matrix, where column is a recurrence plot expressed as a very long vector of point values in the Then, we compute the eigenvectors of the matrix, obtaining the face space.

Once we have the face space, we can express each recurrence plot as a linear combinati of the eigenfaces. This results in a point in the face space representing a whole recurren plot with only a few values.

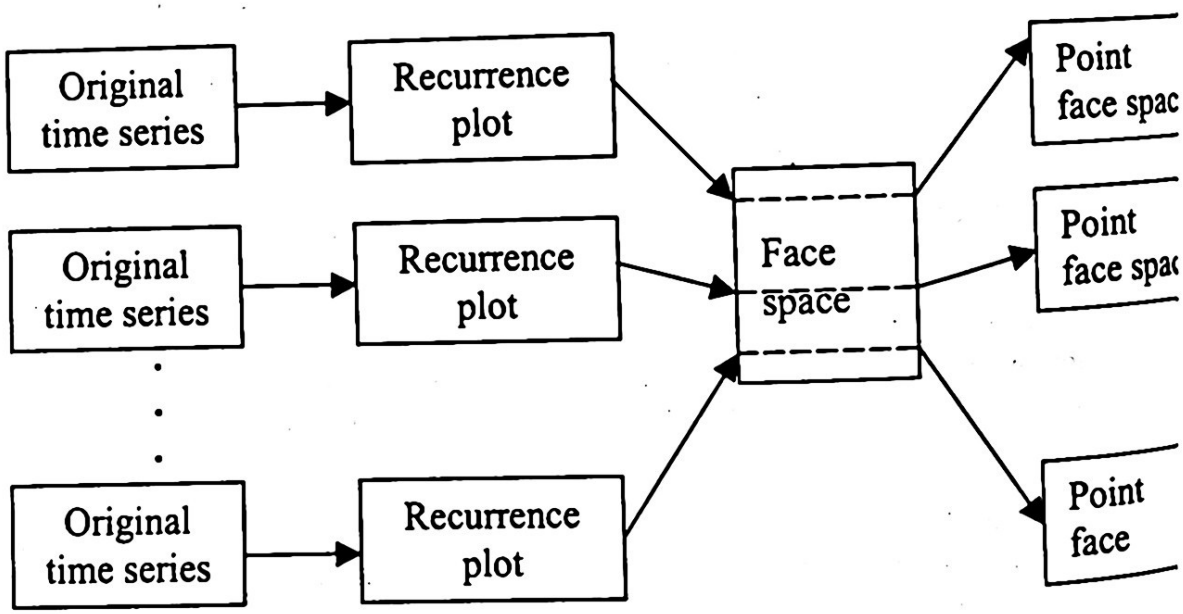



**Figure 3.** Process to train the ANN from the original time series

Note that dimension of the vectors in the face space will be equal to the number of vectors used as the orthogonal incomplete basis for that face space. The number of eigenfaces that can be computed is equal to the number of recurrence plots.

In principle, we may use the whole set of eigenfaces to obtain an accurate representation of the recurrence plots. However, it is possible to take a subset of the eigenfaces, choosing those which have the longest corresponding eigenvalues.

As the recurrence plots have more expression power than time series, at the end of the process we have a detailed description of each series represented by a short number of values. According to this, the whole process can be used as a preprocessing stage for another analysis or treatment which needs an accurate description of the signals and cannot process a large amount of entries (as in the case of Artificial Neural Networks).

For the post-processing stage, if the output of an analysis process (which received points in the face space as inputs) is another point in the same space, the resulting point can be converted to a recurrence plot and, in turn, to the underlying time series. To achieve this inverse process is necessary for the output point to be referred to a known face space, and for the recovered recurrence plot to have an embedding dimension and delay of 1. This is expected to happen when the output point of the analysis process is applied to the same face space as the one which was used to produce the input for the system, and when all the recurrence plots have been generated using embedding dimensions and delays of 1.

Using this preprocessing and post-processing stages we have a complete cycle for time series analysis using a complete but very short description of them.

## 5. Computing the Recurrence Plots.

To compute the recurrence plots, the system first obtains the vector of each given point in the time series according to the given embedding dimension and delay. For example, suppose the system is about to compute the point (5, 3) in the plot. If the embedding dimension is 2 and the delay is 3, then the computed vectors would be $(x_5, x_8)$ and $(x_3, x_6)$.

Then, the value of each point of the plot is computed as the magnitude of the difference between the vectors, and the value is added to the plot. The resultant value can be represented in the plot in several manners according to the choice of the user. For example, a point can be added to a text file, or can be represented as a color in a bitmap file or on the user's screen.

As an optimization, if the embedding dimension is 1, the system does not compute the vectors, but simply computes the value of the point as the difference between the values in the given positions. In the previous example, if the embedding dimension is 1, the point (5, 3) is computed as $x_5$-$x_3$.

## 6. Computing the Eigenfaces.

To compute the eigenfaces, the recurrence plots of the training set are taken as very long vectors. If a NxN plot is given, the corresponding vector will have $N^2$ elements.

In the original technique, the average vector of the plots in the training set is computed, the difference between each vector and the average vector is placed as a column matrix. However, better results are obtained if we take the original vectors instead difference with the average vector.

Call A the matrix which has the plots (or the difference between the plots and the plots) as its columns. The eigenfaces are the eigenvectors of the matrix $AxA^t$.

A common size for a time series is about 1000 values. So, the recurrence plots are expec to have about 1,000,000 points. In this case, the resultant matrix $AxA^t$ will have 1,000,( rows and 1,000,000 columns. Computing the eigenvectors of a 1,000,000x1,000,000 is an intractable problem.

A common technique to overcome this problem is to first compute the eigenvectors matrix $L=A^txA$ [3]. If we have *m* plots in the training set, then L will be a *mxm* smaller than the previous one.

To compute the eigenvectors of L the system uses the QR algorithm[4], which consist finding a matrix Q such that $D=Q^tLQ$, where D is a diagonal matrix, and Q is a sequenc rotations, i.e. $Q=Q_1Q_2...Q_m$ where each $Q_i$ is a rotation matrix. At the end, each value diagonal of D is an eigenvalue, and each column of Q is the corresponding eigenvector.

If we place these eigenvectors as columns of a matrix V, the eigenfaces can be comput the result of normalizing the matrix AxV. The system stores the resultant matrix eigenfaces in order to use it with future plots related to the training set.

Let M be the matrix of eigenfaces obtained from the previous step, and let *u* be a recur plot represented as a vector. To compute the representative vector of recurrence plot product M x *u* is computed. The resultant vector will have as many elements as the set of the eigenfaces.

The representative vectors of the recurrence plots can be reduced even more by taking the most important eigenfaces from the generated set. By doing so, the training totally recovered from the eigenfaces, but just approximated.

The most important eigenfaces for the training set will be those having the eigenvalues generated when the eigenvectors are computed.

## 7. Recovering the Time Series.

If the output of the analysis stage is a point in the face space, it is necessary to trave inverse path in order to obtain the time series represented by this point.

To return to the original series, it is necessary to convert the point in the face space corresponding recurrence plot, and then return to the original time series, as shown 4. The first step can be done easily by multiplying each component of the vector corresponding eigenface. The last step is a little more complicated, because the plot holds only the differences between the vectors of the chosen dimension and dela

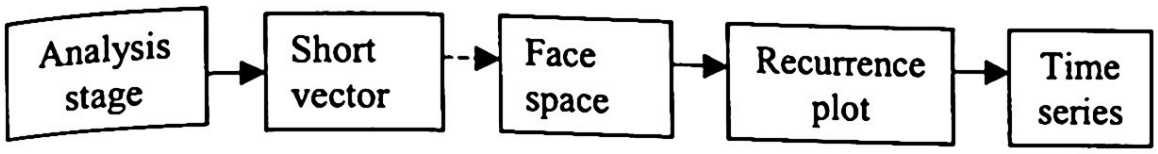



**Figure 4.** Process to obtain a time series from the output of the analysis

We can return from the recurrence plot to the time series only if the dimension and delay used to build the recurrence plot are both 1. Even in this case we cannot always recover exactly the original time series, since it is impossible to recover the original bias, and the recovered plot may be the negative of the original one if the second point of the series is less than the first.

In the case of a recurrence plot with dimension and delay 1, we actually have the absolute differences of each point of the time series with respect to every other point. To recover the original series, we need to choose the recovered values for the time series in a way that all the "lines" in the recurrence plot represent the same information (or similar information in the case of noisy recovered plots).

The basic algorithm to recover the time series is:

1. Take the values of the first line of the recurrence plot as the values of the time series. Remember that the first line in the time series represents the difference of all the point with respect to the first one, so what you are doing is to consider the first point as zero and all the other points as greater or equal to the first point.
2. Take the next line (line $n$) from the recurrence plot
3. For each point in the line, compute the sum and the difference between the value at position $n$ (number of line in the recurrence plot) in the current line, and the value of all other positions $i$ in the line. Take the absolute value of the result and call it $\delta^+$ for the sum, and $\delta^-$ for the difference.
4. Let $\lambda$ be the value in the position $n$ of the recovered time series. If $|\lambda-\delta^+|<|\lambda-\delta^-|$ then the sign of the value at position $n$ in the recovered time series is wrong, and should be changed.
5. Repeat step 2 for each line in the recurrence plot.

Since the recovered recurrence plot may be noisy because of the recovery process, we actually compute the value of each point in the time series as the average of the values extracted from the recurrence plot for that specific point. Also, we do not change the sign of the value each time it is detected to be wrong. Instead, for each position in the time series, we have a counter that stores the number of iterations in which the algorithm detects it should be changed. The sign of the value is only changed when the counter crosses by 0, i.e. only when most of the iterations agree in that the sign should be changed.

To explain how our algorithm works, we can observe the relationship among the involved data in figure 5. The figure shows the original time series, the previous recovered data (*RecValue*) and the value of the plot when analyzing line $n$ ($nl$) of the plot.

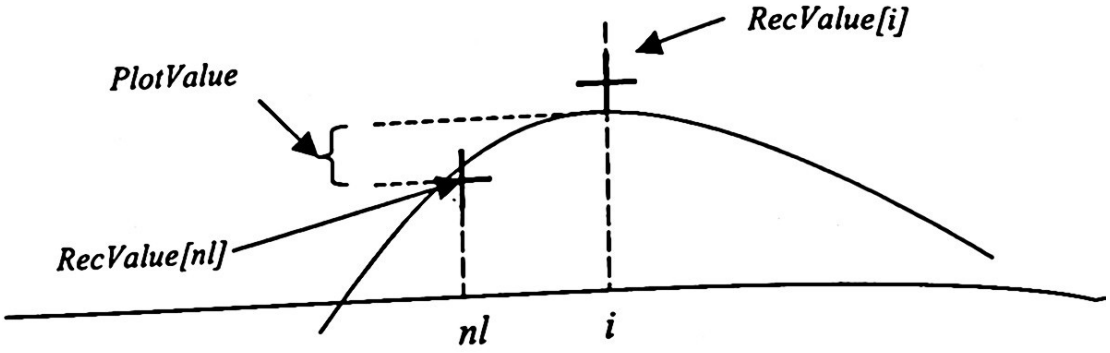



**Figure 5.** Relationship among recovered data, plot values and the original time series when returning from a noisy plot.

Suppose we want to get a better approximation of point *i*. In a first approach, we assume th recovered value in position *nl* (the number of line we are using on the plot) as correct. definition of the recurrence plot, the value of the plot in line *nl*, position *i* (*PlotValu* should be equal to the distance between *RecValue[i]* and *RecValue[nl]*. Note that when dimension and delay of the plot are both 1, the actual distance is simply the absolute of the difference between the values.

If we were sure *RecValue[nl]* and *PlotValue[i]* are correct, we could compute the real in position *i* simply by adding *RecValue[nl]* and *PlotValue[i]*. However, these values not always accurate, so we should better compute the new value of *RecValue[i]* average among all the computed values. This way, in each step we should add or subtract:

$$\frac{\mathrm{Re}cValue[i]-(\mathrm{Re}cValue[nl]-PlotValue)}{Total\ number\ of\ lines\ in\ recurrence\ plot}$$

to *RecValue[i]* in order to compute its new value.

The complete algorithm to return to the original time series from the recurrence shown below:

```
RecoverOriginalValues ()
{
RecValue[] = values in first line of the recurrence plot
positive[] = {0}  //Counter of positive agreements
for nl=1 to data length
       CorrectOriginalValues (RecValue, nl, positive);
}

CorrectOriginalValues (RecValue[], nl, positive[])
{
PlotValue = Value of the plot in line nl, position i
for i=0 to data length
{
if PlotValue<0          // The recurrence plot should not have
PlotValue = 0; // negative values
       if |PlotValue-|RecValue[nl]+RecValue[i]| | <
                              |PlotValue-|RecValue[pos]-RecValue[i]| |
```

```
            //Incorrect sign
            if RecValue[i]>0 {
                  positive[i]--;  // Change the sign every time
                  if positive[i]<0        // positive[i] crosses by zero
                        RecValue[i] = -1.0*RecValue[i];
            }
            else {
                  positive[i]++;
                  if positive[i]>0
                        RecValue[i] = -1.0*RecValue[i];
            }
      }
      else {          // Correct sign. Correct the value
            if RecValue[i]>0
                  positive[i]++;
            else
                  positive[i]--;
            AbsDif = |RecValue[pos]-RecValue[i]|;
            if AbsDif>PlotValue { // Increase the difference
                  if RecValue[i]>RecValue[pos]
                        RecValue[i] = RecValue-(AbsDif-
                              plotValue)/(length of data);
                  else
                        RecValue[i] = RecValue+(AbsDif-
                                    PlotValue)/(length of data);
            }
            else {                // Decrease the difference
                  if RecValue[i]>RecValue[pos]
                        RecValue[i] = RecValue[i]+(PlotValue-
                              AbsDif)/(length of data);
                  else
                        RecValue[i] = RecValue-(PlotValue-
                              AbsDif)/(length of data);
                  }
            }
      }
}
}
```

## 8. Example.

As an example consider three input time series, each with 200 values. These signals are shown in Figure 6.



**Figure 6.** Three input time series

The first step of the preprocessing stage is to compute the recurrence plot of each input time series. The resulting recurrence plots are shown in figure 7.

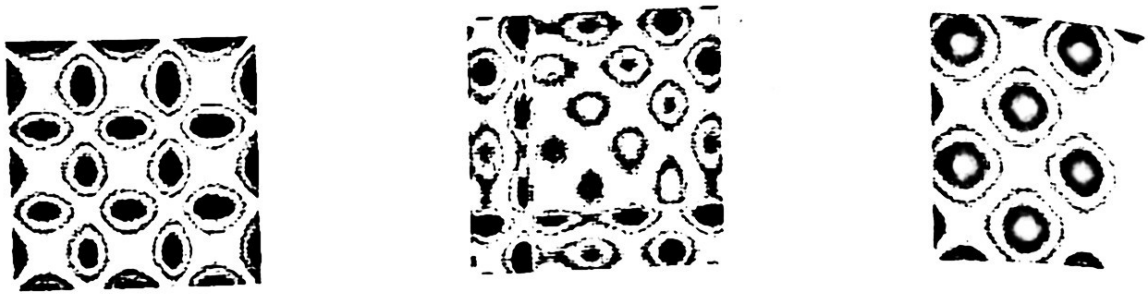


**Figure 7.** Three recurrence plots corresponding to the input time series

The next step is the generation of the set of eigenfaces corresponding to the recurrence shown above. These eigenfaces form the face space, and thus are treated as a However, for this paper the eigenfaces were extracted and their representation with negati values put to zero is shown in figure 8. Note that the eigenfaces do not have a one correspondence with the recurrence plots.

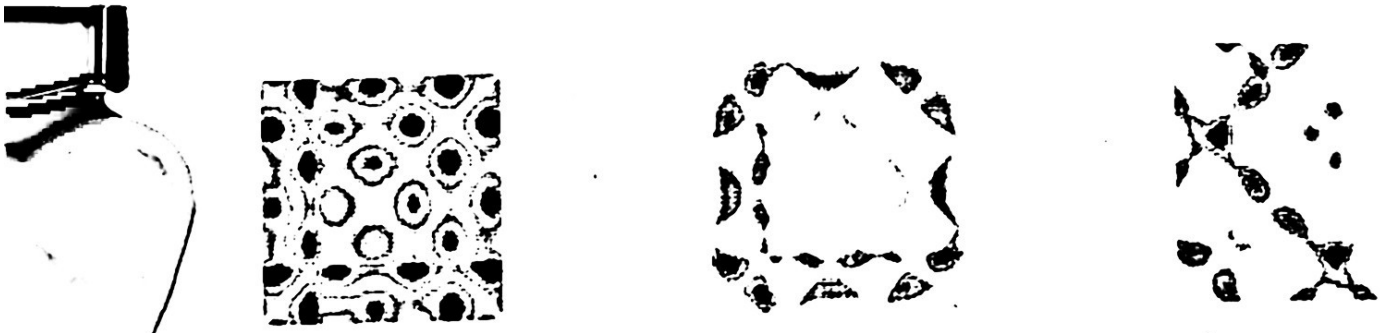


**Figure 8.** The eigenfaces of the face-space

The vectors in the face-space show the weight each eigenface has in the recurrence These non-normalized vectors are shown below, keeping the order of the above recurre plots:

(777.1191573465204, -494.3188957993881, 12.325170500631794)
(1448.5151494857735, 268.12345386239616, 6.803736657263384)
(138.87833187137394, -30.50296638025993, -139.9314167509452)

Given the vectors in the face-space, the original recurrence plots and the underlying series can be recovered, as shown in figures 9 and 10.

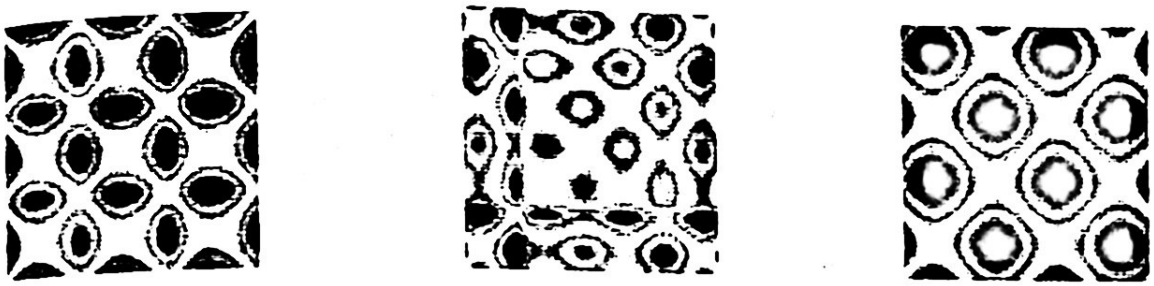


**Figure 9.** Recovered recurrence plots



**Figure 10.** Recovered time series

Note that the first and second recovered time series are the inverse (negative) of the original versions. This happens because the recurrence plots do not have information about the sign of the first point.

## 9. Conclusion.

Powerful tools for time series analysis exist, such as ANN. Sometimes, a technological problem arises when it is necessary to analyze a large amount of inputs in parallel, because this would make necessary to have an input and an output in the analysis tool for each single value in the time series. In the case of ANNs, would be necessary to have a number of neurons of at least two times the number of values in the time series. Furthermore, depending on the number of hidden layers in the ANN, the number of neurons may be several times the number of inputs.

The number of entries can be dramatically reduced using the eigenfaces technique. However, the result can be improved converting the time series to a recurrence plot before applying the process. The recurrence plots can easily show periodicity and many other features hardly visible in the original time series.

The inverse path is needed to process outputs from the ANN and obtain the corresponding time series. Specifically, we have shown an algorithm to recover the original time series from a possibly noisy recurrence plot.

## References


[1] Koebbe, M. & Mayer-Kress, G. (1992). Use of Recurrence Plots in the Analy Time-Series Data. *Nonlinear Modeling and Forecasting, SFI Studies in the Scienc Complexity, Proc. Vol. XXI,* Addison-Wesley, Santa Fe. 361-378.

[2] Turk, M. & Pentland, A. (1991). Face recognition using eigenfaces. *Proc. IEEE Maui, HI.* 586 - 591.

[3] Turk, M. & Pentland, A. (1991). A. Eigenfaces for Recognition. *Journal of Cogn Neuroscience.*

[4] Watkins S. D. *Fundamentals of Matrix Computations.* John Wiley & Sons. 279, 280.

[5] Joliffe, I. T. (1986). *Principal Component Analysis.* Springer-Verlag, New York.

[6] Pentland A., Moghaddam B. & Starner T. (1994). View-Based and Eigenspaces for Face Recognition. *Proc. of IEEE Conf. on Computer Vision Pattern Recognition (CVPR'94)*

# Distributed Bayesian Networks for patterns representation

J. M. Medina-Apodaca & J. Figueroa-Nazuno
Centro de Investigación en Computación, Instituto Politécnico Nacional

**Abstract**

In this paper is presented the design and implementation of a Distributed Bayesian Networks building tool, which can be used for collaborative work in the tasks of human experts knowledge representation. The system allows the users to build a new network, as well as joining an existing one which may be physically distributed along several remote places. Once built, a Bayesian Network contains all the information about conditional probabilities among a set of involved variables. Each group of Bayesian Networks is made accessible to its users by a centralized repository for such group. The interfaces for the system have been specified using CORBA's IDL. At this moment, the system is implemented with the Java language.

**Keywords:** Bayesian Networks, artificial intelligence, knowledge representation, distributed systems.

## 1. Introduction.

Let $X$, $Y$ be random variables for a random experiment with a sample space and probability measure $P$. Suppose $X$ takes values in sample space $S$ and $Y$ takes values in sample space $T$. The distributions of $X$, $Y$ are called marginal distributions.

If $X$ and $Y$ are independent, the probability of $A$ and $B$ is given by the product of their individual probabilities:

$$P(A,B) = P(A)P(B)$$

If $A$ and $B$ are not independent, i.e. the value for $A$ depends on the value of $B$, the probability of $A$ given that $B$'s occurrence is given by the expression:

$$P(A|B) = \frac{P(A,B)}{P(B)}$$

We have $P(A,B) = P(B|A)P(A)$. Using these equations, we obtain the Bayes Rule:

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}$$

The law of total probability asserts that if $\{A_1, A_2, \ldots, A_n\}$ is a set of independent variables whose space is a partition of the sample space, and if a random variable $B$ is dependent of variables $A_1, A_2, \ldots, A_n$, then the probability of $B$ is given by:

$$P(B) = P(B|A_1)P(A_1) + P(B|A_2)P(A_2) + \cdots + P(B|A_n)P(A_n)$$

The joint distribution of variables $X$, $Y$ is a distribution which takes values in a [1]. The importance of the joint distribution is that it stores more information marginal distributions, i.e. the marginal distributions can be obtained from distribution, but the joint distribution cannot be obtained from the marginal distributio

More generally, if we have n variables $V_1$, $V_2$, ..., $V_n$ with corresponding spaces $S_1$, where space $S_i$ contains $m$ values $v_{i1}$, $v_{i2}$, ..., $v_{im}$, the joint distribution space is $S_1 x S_2 x$ The main drawback for using of the joint distribution is that it expands very according with the number of variables and the size of their sample space. If variables are given, each with a sample space of size $m$, the total number of entries joint distribution table is $m^n$.

A Bayesian Network is an augmented directed acyclic graph, represented by the where:

- $V$ is the set of nodes, each of one represent a random variable, along probability distribution table indicating how the probability of this values depend of all the possible combinations of the parents values.
- $E$ is the set of directed edges.

A random variable $X$ has an incident arc coming from random variable $Y$ if and dependent of $Y$, and $X$ is said to be a parent of $Y$. Suppose there is a third variable parent is $Y$. In this case $X$ and $Z$ are said to be independent given $Y$ (see figure 1).

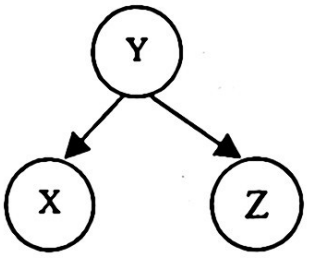



**Figure 1.** Variables $X$ and $Z$ are independent given $Y$

Every node in the Bayesian Network has associated a probability distribution depends only of the its parents value. In the above example, $X$'s distribution must entry for every value in the sample space of variable $Y$.

Any entry in the joint probability distribution can be computed from a Bayesian using the independence and conditional probability rules. This way, a Bayesian achieves a smaller representation of the joint distribution by using context informatio dependency/independency patterns among the random variables.

There are two approaches for reasoning with a Bayesian Network: Top-Down and Up[2]. The Bottom-Up reasoning (also called diagnostic) goes from effects to this approach, the probability of a parent variable can be computed given the occurr

an event in a child variable. The Top-Down reasoning (or causal reasoning), computes the probability of an event given the occurrence of a parent variable.

## 2. The Distributed Model.

The proposed model for the Distributed Bayesian Network is based in domains. A domain is a well delimited knowledge area which can be represented by a set of random variables semantically related. A consistent Bayesian Network could be constructed using only the variables of the domain. However, some of these variables may be involved with the variables in another domain, forcing inter-domain connections.

As an example consider two domains: *"Weather"* and *"Traffic"*. A representative Bayesian Network can be constructed for each of these domains, containing the random variables of interest. Figure 2 shows a connection between these two domains. If it is raining, the probability of a car accident augments, connecting the random variable "Rains" which has the space of possible values *{True, False}*, and the variable *"Car accident"*, which has the same space. When the probability of slow traffic is computed, the variables in *Weather* should be included.

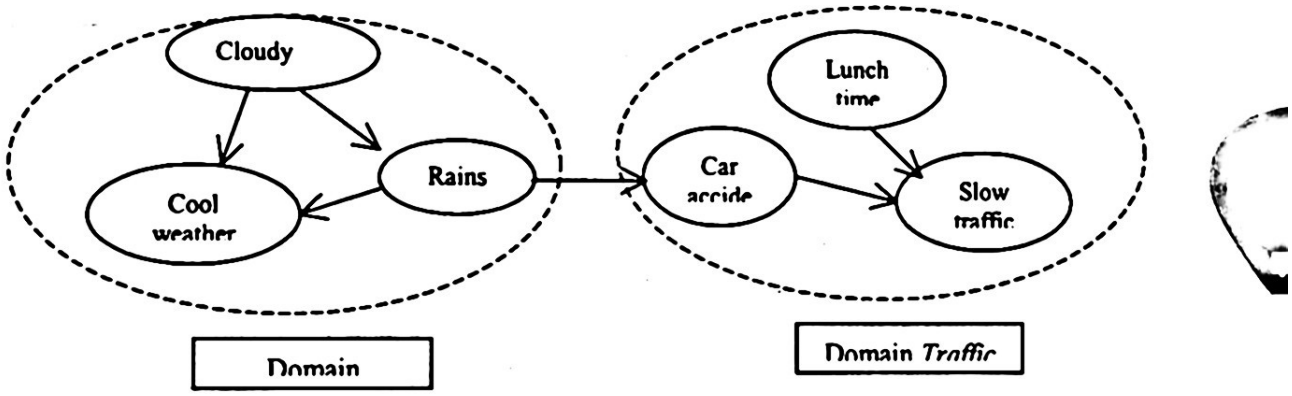



**Figure 2.** The domains *Weather* and *Traffic*

When two domains are related by one or more connections among their variables, the visualization of each domain could expand to see the whole resultant Bayesian Network. The decision of doing so depends of the specific application and user preferences.

The distribution of each variable respect to its parents should be kept local with the variable. This should be totally changed, though, whenever a parent is added or removed to the variable.

The probability of a random variable in the network is evaluated according to a specific scenario. The scenario is the set of values of the involved variables, according to which the resultant probability will be computed. If the value of a variable is omitted, the resultant probability can whether be computed using the probability distribution of that variable, and considering each possible value this can take, or computing a single probability value of the variable according to the probabilities of its predecessors. This way, the resultant

probability may be a new distribution where all the possible values of the omitted varial are considered, or a single probability value.

## 3. Design of the Distributed System.

The system has been designed using remote objects, specifically CORBA[3]. The objects are: BayesNet, Node, Distribution, Scenario, BayesNetsRepository BayesNetObserver. Some of these remote objects are discussed next.

### 3.1. BayesNet.

This object represents the complete Bayesian Network. Contains the nodes of the and is responsible of tasks that involve the whole network, such as computing probability of a given node inside the net or finding the root nodes (those nodes which not have any parents).

When an event occurs in the network (i.e. a node is added or removed, or two nodes connected or disconnected), the Bayesian Network must notify all the interested objects. object can register itself to be notified of events generated in the network.

```
interface BayesNet : NodeListener
{
        typedef sequence <BayesNet> BayesNetArray;
        typedef sequence <Node> NodeArray;
        string getName ();
        NodeArray getNodes ();
        Node getNode (in string name);
        void addNode (in Node n);
        void removeNode (in string name);
        NodeArray getRootNodes ();
        Distribution computeProbabilities (in string nodeName, in Scenario scenario
                                                raises (BayesNetException);
        double computeProbability (in string nodeName, in Scenario scenario)
                                                raises (BayesNetException);
        boolean isMember (in string nodeName);
        void addBayesNetListener (in BayesNetListener nl);
        void removeBayesNetListener (in BayesNetListener nl);
        string toXBN ();
};
```

**Figure 3.** Definition of the Bayesian Network interface

The BayesNet object listens for actions produced for any of its nodes. This network itself can fire a notification to its listeners and take the adequate actions, adding an external node which has been selected as parent or child of any of the the network.

The network fires an event when a node is moved, added or removed, and when a link (a parent-child relationship) is added or removed to the network.

Every Bayesian Network in the system is identified by an unique name given by the user. Along with its name, the network keeps a description of itself, which can be used for helping the user to select a net among a set of them. The IDL interface for the BayesNet object is shown in figure 3.

## 3.2. Node.

The Node object represents an independent node in the system. Each node must maintain references to every parent and child node, a reference to its current probability distribution, its position in the plane of the network, its name, and the listeners to the actions of this node. Every node in the system is identified by a name given by the user. If two nodes have the same name but they are in different networks, each network will see only its own node. But if the nodes are in the same network, then they will be indistinguishable of each other.

A node fires actions when a parent or child is added or removed from it, and when the node is moved to other position in the network. The IDL interface for the node is shown in figure 4.

## 3.3 Distribution.

The Distribution object represents the distribution of a variable according to its parents. The current version of the system supports only distributions with the sample space of values True and False.

This object keeps a mapping between the values of the set of variables and the probability for that specific combination of values. The Distribution object must store the names of every participating variable, except the variable to which the distribution belongs. If there are n variables in the distribution, the object will maintain $2^n$ probability values. According to this, a non-parents variable will have only one probability value: the marginal probability of the variable to be True.

The Distribution object has two ways to return a probability value: receiving a Scenario object or receiving a number. The former way is discussed below. When the last way is used, the number is interpreted as a set of bits indicating the value of each of the variables. The first variable is positioned in the least significant bit position, and the last in the most significant position. A value of zero in the corresponding bit indicates a False value for the variable, and a value of one indicates a True value.

This object must be able to perform the compute sub-distribution operation. In this operation, the Distribution object receives an Scenario object with the values of the known variables, and the object is expected to return a new distribution with the probability values of the unknown variables (those which are not in the scenario) in those cases where the given scenario is fulfilled. So, if variable v has the value True in the scenario object, the

```
interface Node
{
        typedef sequence <Node> NodeArray;
        string getName ();
        void setName (in string name);
        string getDescription ();
        void setDescription (in string desc);
        Distribution getDistribution ();
        NodeArray getParents ();
        long getParentsLength ();
        NodeArray getChildren ();
        long getChildrenLength ();
        Node getParent (in string name);
        boolean isParent (in string name);
        void addParent (in Node parent);
        void removeParent (in string name);
        void setDistribution (in Distribution newDist);
        Node getChild (in string name);
        boolean isChild (in string name);
        void addChild (in Node child);
        void removeChild (in string name);
        void setPosition (in long x, in long y);
        void addNodeListener (in NodeListener nl);
        void removeNodeListener (in NodeListener nl);
        long getXPos ();
        long getYPos ();
        string varToXBN ();
        string structureToXBN ();
};
```

**Figure 4.** Definition of the node interface

resultant sub-distribution must contain just the probability values of those cases wher variable is True. This operation is very useful when computing the probability distril of a given node in the Bayesian Network.

The IDL interface for the Distribution object is shown in figure 5. Although it is Distribution object is one of the most important remote objects of the system.

### 3.4 Scenario.

The Scenario object is used simply as a name-value mapper for a given set of variable used mainly as a friendly interface to retrieve values from the Distribution object. functions allows the user to set the values of the variables as a sequence of bits in in the way described in the Distribution object.

```
interface Distribution
{
        StringArray getNames ();
        long getNamesLength ();
        void setNames (in StringArray names);
        void setValues (in DoubleArray values)
                                raises (DistributionException);
        boolean isElement (in string name);
        double getProbability (in Scenario scenario)
                                raises (ScenarioException);
        double getBinaryProbability (in long bits)
                                raises (DistributionException);
        void setProbability (in Scenario scenario, in double prob)
                                raises (ScenarioException);
        Distribution computeSubdistribution (in Scenario scenario)
                                raises (DistributionException);
        string toXBN (in string nodeName);
};
```

**Figure 5.** Definition of the distribution interface

### 3.5 BayesNetsRepository.

The Bayesian Networks repository is the object that stores references and descriptions of every Bayesian Network in a collaborative session. Each network in the repository is identified by its name. There must be exactly one repository for each collaborative group.

### 3.6 BayesNetObserver.

Sometimes local objects, such as graphic user interfaces, will need to listen for a Bayesian Network's events in order to update their representation or internal state. As local objects, they cannot just register with the Bayesian Network object because they cannot be an argument in a remote object invocation.

The solution used here is to define a listener remote object which listens for the Bayesian Network's events, and whose servant implementation can register (as a local object) the local listeners interested in such events. The events received by the remote observer will be retransmitted to the registered local objects.

## 4. Probability Computing.

For the computation of the probabilities of a node given a scenario, distributions are computed and merged in a recursive fashion. The system gives the option of computing the probability of the node for each possible configuration of the unspecified parents.

In each stage of the compute, the node's distribution is obtained, and the Distribution's `computeSubdistribution()` method is applied with the given scenario. If a parent is

found in the scenario, its value is taken and applied to the actual distribution, eliminating entries in which the parent's value is different from that taken from the scenario. If resultant distribution has only one value (the node's probability), the distribution returned.

After the sub-distribution is computed, the probability of the unknown parents (those are not included in the scenario) is computed with the given scenario in a recursive call. all the possible configurations are to be shown, the distributions are merged in a new that contains the names of all the unknown parents, and the probability entries computed. Otherwise, the parents probabilities are reduced to one probability using probability chain rule[4].

When all the configurations probability is required, for each possible combination of valu in the merged distribution, the probability value when the considered nodes have the of the actual combination is obtained from the sub-distribution. This value is multipl either by the probability value of each parent for that combination if the value of the is positive in the actual combination, or by 1 minus its probability if the value is negati At the end, the merged probability distribution will be completely full and can be retur as the result. If only the total probability is required, these results are added in instead as being put in the merged probability distribution.

## 5. Local Design.

The local or implementation side of the system has been designed for a collaborat interactive graphic environment. Here are explained some of the most important decisions assumed for the developed system.

### 5.1. Probability computing methods.

There are several methods for computing the probability of a given variable in a Bayes Network. The most important are the automatic method (used by the system), stochastic sampling method. Because of this variety of methods, the Strategy pattern[5] has been used, allowing for future use of new methods without any changes the Bayesian Network servant implementation.

### 5.2. Visualization of the networks and other possible options.

When a user is working with a network, he/she should be able to see whether just which is working with or all the involved nodes in the network.

When the user wants to see the complete network (all the involved nodes), then every a parent/child connection is performed between a included node and a not included system must add every connected not included node to the actual network.

This may be just one of several options that may be presented to the user. As Bayesian Network behavior is fixed, and one or more options may be optionally chose the user, the Decorator design pattern has been used. The base network implementation as the basic component, and every extension to this behavior acts as a decorator.

## 6. Storage of the Networks.

The Bayesian Networks generated using the system can be saved and recovered using the XBN format [6]. This XML based format was proposed by the Decision Theory and Adaptive Systems group of Microsoft Research, and is expected to replace older Bayesian Networks Interchange Formats (BNIFs). This format allows the analysis of networks generated with the system using other existing tools, and the collaborative edition of already created networks.

## 7. Implementation Issues.

Currently, the system has been completely implemented using the Java language. Despite of the relative simplicity of the distributed system using CORBA, there are some issues which should be addressed during the implementation stage.

### 7.1 The distributed callback on a synchronized method.

In general, Node events cause the execution of a BayesNet object's method. This method, in turn, will generate an event received by a BayesNetObserver object. This object will communicate the event to the local listeners, which in turn will invoke some methods on the BayesNet object and some Node objects to repaint the network. This sequence of events will produce a callback on the BayesNet object, and probably on the Node object which produced the event, as shown in figure 6. Note that the notifier methods in Node, BayesNet and BayesNetObserver cannot finish before the last listener has finished.

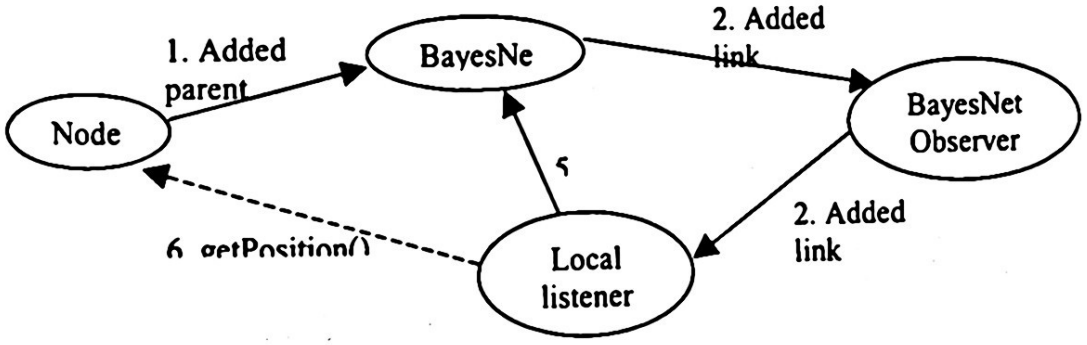



**Figure 6.** The chain of events causes a callback over the BayesNet and Node objects

As the remote objects can be accessed by several threads at the same time, many of their methods must be synchronized.

The Java monitors are reentrant [7]. This means that a thread will not deadlock itself when it invokes a synchronized method from another synchronized method. But in the distributed case, the reentrant thread is not the same as the one that holds the monitor, so a deadlock is produced.

To overcome this problem, a thread is created for each notification. This way, a method can finish after creating one thread for each listener, freeing the monitor allowing the callback.

## 8. Test of the System.

The system has been tested using the Serum Calcium model developed by Greg (Cooper, 1986). The system's input is the XBN file specifying such model, and the are compared with the manually computed results. Figure 7 shows the representation for such file.

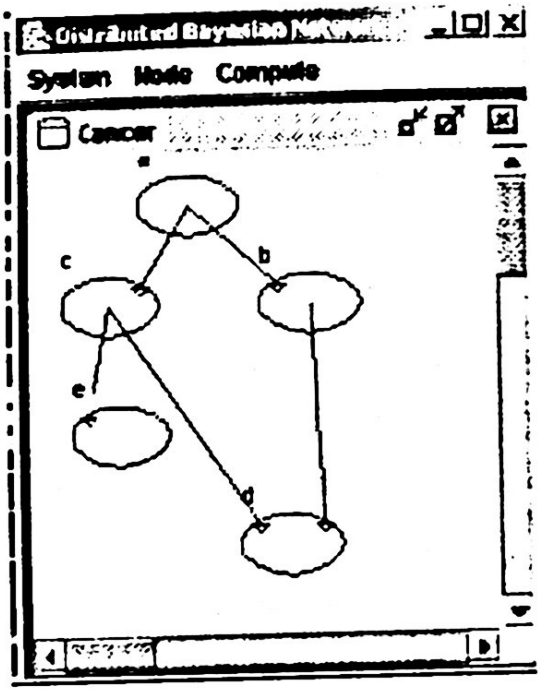



**Figure 7.** Serium Calcium model represented by the system

The probabilities of each node are shown below:

*a*. True: 0.2.
*b*. True given not a: 0.8. True given a: 0.2.
*c*. True given not a: 0.2. True given a: 0.05.
*d*. True given not b and not c: 0.8. True given not b and c: 0.9. True given b 0.7. True given b and c: 0.05.
*e*. True given not c: 0.8. True given c: 0.6.

As a Bottom-Up reasoning test, consider computing the probability of *a=True d=False*. According to the Bayes Rule, this probability is:

$$P(a|d) = \frac{P(d|a)P(a)}{P(d)}$$

*P(a)* is known. To compute *P(d)* we use the law of total probability. According to

$$P(d)=P(d|b,c)P(b,c)+P(d|b,not\ c)P(b,not\ c)+P(d|not\ b,c)P(not\ b,c)+P(d|not\ b,not\ c)P(not\ b,not\ c)$$

As $b$ and $c$ are independent given a, we have $P(b,c)=P(b)P(c)$. Using the same reasoning, we have $P(b)=P(b|a)P(a)+P(b|not\ a)P(not\ a)$ and $P(c)=P(c|a)P(a)+P(c|\ not\ a)P(not\ a)$.

Computing the numbers we get $P(b) = 0.68$, $P(c) = 0.17$ and $P(b,\ c) = 0.1156$.
Similarly: $P(b,\ not\ c) = 0.5644$, $P(not\ b,\ c) = 0.0544$ and $P(not\ b,\ not\ c) = 0.2656$. Finally, we get $P(d) = 0.6623$.

To compute $P(d|a)$, we simply consider $P(x|a)=P(x)$. Doing this we get P(b)=0.2, $P(c)$=0.05, $P(b,c)$=0.01, $P(b,\ not\ c)$=0.19, $P(not\ b,c)$=0.04, $P(not\ b,\ not\ c)$=0.76, and $P(d|a)$=0.7775.

Finally, $P(a|d) = 0.23478786$. This result is the same as the computed by the system.

Now, consider a Top-Down test, in which the probability of $d=True$ is computed given $a=True$.

Making a partition with the values variables $b$ and $c$ in conjunction, and using the law of total probability, we get:

$$P(d|a) = P(d,b,c|a) + P(d,b,\ not\ c|a) + P(d,\ not\ b,c|a) + P(d,\ not\ b,\neg c|a)$$

$$P(d|a) = P(d|b,c,a)P(b,c|a) + P(d|b,\ not\ c,a)P(b,\ not\ c|a) + P(d|not\ b,c,a)P(not\ b,c|a) + P(d|not\ b,\ not\ c,a)P(not\ b,\ not\ c|a)$$

As $d$ does not have a direct dependency on $a$, and as $b$ and $c$ are independent given $a$, the expression above can be rewritten as:

$$P(d|a) = P(d|b,c)P(b|a)P(c|a) + P(d|b,\ not\ c)P(b|a)P(not\ c|a) + P(d|not\ b,c)P(not\ b|a)P(c|a) + P(d|not\ b,\ not\ c)P(not\ b|a)P(not\ c|a)$$

Those probabilities can be directly obtained from the variables' distributions. After computing the products we get.

$$P(d|a) = 0.0005 + 0.133 + 0.036 + 0.608 = 0.7775.$$

Which is the result computed by the system.

## 9. Conclusion.

Bayesian Networks are very useful tools for representing dependency relationships among random variables. Unfortunately modeling these relations is not an easy task, and may require collaboration among field specialists of the involved areas. This system allows a collaborative constructions of Bayesian Networks using the concept of domains to divide

the problem in a set of more manageable parts. Two domains can be joined by establis a parent-child relationship between one or more pairs of their nodes.

Every Bayesian Network for a group of users is maintained in a repository, where it retrieved by its name. More than one user can access the same network, but the net belongs always to a single domain, and is physically kept in a lonely site.

The only cache used is the graphic representation of the network for each user. A through semantic is used for every update to the network. The update to every representation (including the source of the update) is delayed until the node and Baye Network are updated, and the last fires the proper event.

Extensions can be made to this work, such as addition of new probability compu methods. If the system is to be used in a slow network, mechanisms for consiste assurance should be included. These mechanisms, though, may impose limitations the need to have a predefined size of the groups when using vector logical clocks. The presented here is a first step for a Distributed Bayesian Networks Pattern Disco System.

## References

[1] Papoulis, A. *Probability, Random Variables and Stochastic Processes. Internati Student Edition.* Mc Graw-Hill. 165.

[2] Pearl, J. & Russell, S. (2000). *Bayesian Networks.* pp. 2, 3.

[3] Object Management Group. *The Common Object Request Broker: Architecture Specification.* 312 - 335.

[4] Russell, S. & Norvig, P. (1995). *Artificial Intelligence: A Modern Approach.* Hall. Chapter 5.

[5] Gamma, E. Helm, R. Johnson, R. & Vlissides, J. *Design Patterns. Elemen Reusable Object Oriented Software.* Addison Wesley.

[6] Bengtsson, H. *Bayesian networks. A self contained introduction with implement remarks.* 82, 83.

[7] Campione, M. Walrath K. & Huml, A. *The Java Tutorial.* Third Edition. Microsystems.

[8] Dodier, R. (1999). RISO: An Implementation of Distributed Belief Networks. *Symposium on AI in Equipment Service and Maintenance.*

[9] Dodier, R. (1999). Unified prediction and diagnosis in engineering systems of distributed belief networks. *Ph.D. dissertation, Dept. Civil Engineering, of Colorado at Boulder.*

# ALGORITMO DE AGRUPAMIENTO GLC PARALELO

Reynaldo Gil-García[1], José Manuel Badía-Contelles[2]

[1]Universidad de Oriente, Santiago de Cuba (Cuba).

gil@app.uo.edu.cu

[2]Universitat Jaume I, Castellón (España).

badia@icc.uji.es

**Resumen**

En la actualidad los algoritmos de agrupamiento se enfrentan al reto de que los conjuntos de objetos a agrupar son muy grandes, cambian en el transcurso del tiempo y los objetos están descritos por un gran número de rasgos. Esto ha provocado que los algoritmos de agrupamiento tradicionales tomen tiempos demasiado grandes para resolver estos problemas desde el punto de vista práctico. Algunas de las alternativas propuestas para solucionar o aminorar estas dificultades son los algoritmos incrementales y los algoritmos paralelos. En el trabajo se expone un algoritmo paralelo que busca de forma incremental las componentes conexas del grafo de $\beta_0$-semejanza asociado a un conjunto de objetos. Este algoritmo usa el paradigma de programación paralela de paso de mensajes para la paralelización del GLC. Se realiza, además, una valoración del costo computacional del algoritmo diseñado, se implementa en *Message Passing Interface* (MPI) y se realiza un análisis experimental del comportamiento del algoritmo para datos generados de forma aleatoria. El algoritmo paralelo obtenido logra un distribución adecuada de los datos y de los cálculos entre los procesadores. La eficiencia teórica obtenida es la unidad cuando la cantidad de objetos es mucho mayor que la cantidad de procesadores. Se demuestra experimentalmente que cuando los conjuntos de datos son grandes es ventajoso utilizar el algoritmo paralelo propuesto.

**Palabras clave**: algoritmo de agrupamiento, algoritmo incremental, algoritmo paralelo.

# 1. Introducción

Los algoritmos de agrupamiento son herramientas muy utilizadas en distintos contextos como la categorización de documentos [1], agrupamiento de genes y proteínas con similar función [2], seguimiento y detección de sucesos en un flujo continuo de noticias [3], segmentación de imágenes [4], entre otras.

El objetivo del agrupamiento es dado un conjunto de $n$ objetos, descritos a través de $m$ rasgos, crear particiones o cubrimientos de este conjunto. Los grupos formados deben cumplir que la semejanza de los objetos dentro de un agrupamiento sea máxima mientras que la semejanza entre los objetos pertenecientes a grupos diferentes sea mínima.

Los elementos esenciales a tener en cuenta para resolver el problema del agrupamiento son: el espacio de representación de los objetos, la medida de similaridad o semejanza entre objetos (no necesariamente debe ser una métrica) y el criterio agrupamiento o heurística a aplicar. En algunos de estos métodos es necesario defi además, una medida de semejanza entre los grupos, la cual se defi ne en términos de semejanzas entre los objetos que forman dichos grupos.

En muchas aplicaciones el conjunto de objetos a agrupar cambia en el tiempo, pues es necesario agregar nuevos objetos al conjunto o eliminar objetos que ya no existen. Ejemplos de estas aplicaciones son: la detección y seguimiento de sucesos en un continuo de noticias, el análisis del comportamiento de las facturas de una empresa, entre otras.

Los algoritmos de agrupamiento clásicos requieren todo el conjunto de datos realizar el agrupamiento, por tanto, cada vez que se agrega o elimina un objeto habría que realizar de nuevo el agrupamiento. Un paliativo muy usado para resolver situación es acumular un conjunto de cambios y en el horario de menor carga al sistema realizar el agrupamiento. Esta solución hace que se trabaje con datos no actualizados, por lo que se pueden obtener resultados falsos. Además, a medida que crece el conjunto de datos esto es más irrealizable por lo costoso que resulta. Por tanto, se requieren algoritmos que actualicen el agrupamiento a medida que cambia el conjunto de datos, sin necesidad de empezar nuevamente desde el principio. Estos algoritmos reciben nombre de algoritmos incrementales, dinámicos o en línea.

Con el creciente tamaño del conjunto de datos a agrupar, en muchas aplicaciones una sola computadora no puede resolver el problema del agrupamiento en un tiempo razonable. Además, muchas veces una computadora tampoco tiene la memoria necesaria para almacenar los datos que necesita el algoritmo, por lo que hay que recurrir datos en disco, lo que hace más lento aún el procesamiento. Esta es una situación característica donde se puede intentar resolver el problema explotando el poder de cómput y la memoria disponible en un conjunto de procesadores. Así, la paralelización de algoritmos de agrupamiento es la opción para obtener los grupos cuando se tiene conjuntos de datos muy grandes o se necesita utilizar algoritmos de agrupamiento elevada complejidad computacional.

Se han desarrollado paralelizaciones de algunos algoritmos de agrupamiento como por ejemplo, del jerárquico aglomerativo [5] y [6], del *K-Means* [7], del *MAFIA* entre otros.

En el trabajo se expone la paralelización de un algoritmo de agrupamiento incremental que busca las componentes conexas del grafo de $\beta_0$-semejanza asociado conjunto de datos y se hace un análisis de su complejidad computacional. El algoritm paralelo obtenido logra un adecuado balance de carga entre los procesadores al repart equitativamente los objetos entre los procesadores y lograr que cada procesador reali aproximadamente el mismo trabajo. Con este algoritmo paralelo se obtiene una efi cia teórica cercana a la unidad cuando la cantidad de objetos es mucho mayor que cantidad de procesadores.

El artículo se estructura como sigue. En la sección 2 se explican las caracterís cas del algoritmo de agrupamiento GLC. La sección 3 detalla el algoritmo parale propuesto, así como el análisis de su costo computacional. En la sección 4 present mos una valoración de los resultados experimentales obtenidos. Finalmente, damos

conclusiones de nuestro trabajo.

# 2. Algoritmos incrementales

Los algoritmos de agrupamiento incrementales realizan el agrupamiento a medida que van llegando las descripciones de los objetos. Por ello, están en desventaja respecto a los no incrementales que conocen previamente el universo de objetos a agrupar. Esto puede ocasionar que sean menos efi cientes que los no incrementales cuando todos los datos están disponible desde el comienzo (problema estático). Sin embargo, tienen sentido si son más efi cientes cuando el conjunto de datos cambia en el transcurso del tiempo (problema dinámico).

Se han desarrollado diversos algoritmos de agrupamiento incrementales como, por ejemplo, *Single-Pass* [8] y el algoritmo de las estrellas [9]. Estos algoritmos tienen como principal limitación que los grupos obtenidos dependen del orden de presentación de los objetos. Esto lo consideramos una caracteristica indeseable, pues los grupos existentes no dependen del orden en que se presenten los objetos, sino que sólo dependen de las características internas de los datos. Por ello, nuestro propósito fue paralelizar un algoritmo incremental en el que el agrupamiento resultante sea independiente del orden de los objetos.

## 2.1. Algoritmo GLC

GLC [10] es un algoritmo de agrupamiento incremental basado en grafos. Se llama grafo de semejanzas al grafo completo donde los vértices son los objetos a agrupar y las aristas se etiquetan con las semejanzas entre los objetos. Dos objetos cuya semejanza es mayor o igual que un cierto umbral $\beta_0$ (defi nido por el usuario) se denominan $\beta_0$-semejantes [11].

El algoritmo GLC obtiene de forma incremental las componentes conexas del grafo de $\beta_0$-semejanza asociado a un conjunto de objetos. El grafo de $\beta_0$-semejanza es un subgrafo del grafo de semejanzas donde se eliminan las aristas con peso menor que $\beta_0$. Con este algoritmo se garantiza que para todo objeto en un grupo existe al menos en dicho grupo otro objeto $\beta_0$-semejante con él.

Este algoritmo no necesita calcular toda la matriz de semejanzas entre objetos ni requiere tener en memoria dicha matriz, lo cual es una ventaja respecto a otros algoritmos que buscan componentes conexas, pues tiene menor complejidad espacial. Además, el algoritmo GLC permite trabajar con objetos descritos por variables cuantitativas y cualitativas mezcladas, incluso con ausencia de información y no restringe el uso de familias de métricas para realizar la comparación de los objetos. Otra característica del GLC es que nunca compara dos objetos más de una vez.

El algoritmo GLC almacena por cada grupo una lista de los objetos que pertenecen a él y por cada objeto su descripción. Cada vez que llega un nuevo objeto lo compara con los objetos de los grupos existentes. Si el nuevo objeto no fue $\beta_0$-semejante con ningún objeto de los grupos existentes se crea un nuevo grupo unitario con ese objeto. En caso contrario, todos los grupos para los cuales existe al menos un objeto

$\beta_0$-semejante con el nuevo objeto se unen y forman un nuevo grupo al que se agrega también el nuevo objeto. Para conocer los grupos formados en cualquier momento basta con recorrer la lista de grupos e imprimir sus miembros. Los pasos seguidos muestran en el algoritmo 1.

**Algorithm 1** Algoritmo GLC

1. Tomar el siguiente objeto $o$
2. Hacer $L = \emptyset$, lista de grupos a unir
3. Para cada grupo existente $g'$
   a) Para cada objeto $o' \in g'$
      1) Si $Semejanza(o, o') \geq \beta_0$
      Agregar $g'$ a $L$ e ir a 3.
4. Si $L = \emptyset$ crear un nuevo grupo $g$ y agregar en él a $o$
   Sino, unir todos los grupos de $L$ en un grupo y agregar en él a $o$
5. Ir a 1.



Este algoritmo tiene un costo temporal en el peor de los casos de $T_{sec} = t_c n^2 r$ ($t_c$ es una constante que depende del procesador), pues por cada nuevo objeto hay calcular su semejanza con los restantes. No obstante, es bueno destacar que en algoritmo, en muchos casos, no se requiere comparar al nuevo objeto con todos objetos de los grupos existentes. Esto se explica debido a que al encontrar en un grup existente un objeto $\beta_0$-semejante con el nuevo objeto, no se necesita comparar con restantes objetos de dicho grupo.

El costo espacial del GLC es $E_s = cnm$ ($c$ es una constante que depende del de los rasgos de los objetos), pues sólo hay que almacenar las descripciones de objetos y los objetos que están en cada grupo.

## 3. Paralelización

Existen varios paradigmas para implementar algoritmos paralelos como son la me ria compartida y el paso de mensajes [12]. Aunque con el modelo de memoria compa tida los programas son más fáciles de crear, presenta dos limitaciones fundamentale En primer lugar, requiere de un hardware menos escalable que en el modelo de so de mensajes. En segundo lugar, el hardware utilizado en los multiprocesadores memoria compartida, por lo general, es más caro. Por tanto el modelo más difundi es el de paso de mensajes.

En el modelo de paso de mensajes, cada procesador tiene su memoria local e cambia información con los restantes procesadores mediante mensajes a través de red de interconexión. El tipo de esta red de interconexión ejerce una gran influencia la posible efi ciencia a lograr en el algoritmo paralelo.

Para valorar los algoritmos paralelos suele utilizarse un modelo de costo que incluye constantes que dependen del procesador y de la red de comunicación [13]. Las más importantes son:

- $t_c$: tiempo para realizar un cómputo en coma flotante (flop).
- $t_s$: tiempo que tarda la red de comunicación en iniciar un envío de un dato.
- $t_w$: tiempo que tarda en transmitirse una palabra por la red de comunicación entre dos procesadores conectados directamente.
- $s_w$: tamaño de una palabra del procesador.

Para evaluar un algoritmo paralelo se defi nen las siguientes medidas básicas:

- $T_{sec}$: el costo temporal del mejor algoritmo secuencial para resolver el problema tratado.
- $T_{comp}$: el costo de cómputo del algoritmo paralelo.
- $T_{comm}$: el costo temporal de las comunicaciones del algoritmo paralelo.
- $T_{par} = T_{comp} + T_{comm}$: el costo temporal total del algoritmo paralelo.
- $S_{par}$: el costo espacial del algoritmo en cada procesador.

A partir de las medidas anteriores, algunas de las medidas más utilizadas para valorar un algoritmo paralelo son:

- Aceleración: $A = \frac{T_{sec}}{T_{par}}$, mide el incremento de velocidad por utilizar la computadora paralela. Idealmente crece linealmente con el número de procesadores.
- Efi ciencia :$E = \frac{T_{sec}}{pT_{par}}$, mide el grado de aprovechamiento del poder de cómputo paralelo. Tiene el valor de 1 en el caso ideal, cuando el algoritmo paralelo es $p$ veces más rápido que el secuencial, donde $p$ es el número de procesadores.

## 3.1. Algoritmo GLC paralelo

Debido a las ventajas mencionadas anteriormente, en nuestro trabajo utilizamos como paradigma de programación paralela el modelo de paso de mensajes. Dentro de este paradigma utilizamos la estrategia de particionado de datos y el esquema de maestro-esclavos. El procesador 0 actuará como maestro y los restantes procesadores como esclavos.

Para paralelizar el algoritmo GLC repartimos uniformemente los datos entre los procesadores de forma tal que el procesador $i$ almacena la descripción del objeto $j$ si el resto de la división de $j$ entre $p$ (cantidad de procesadores) es $i$. Con esto, tratamos de lograr un equilibrio de carga entre los procesadores, lo cual es una característica deseable en todo algoritmo paralelo.

Cada procesador mantiene una lista de grupos y, a su vez, de cada grupo almacena una lista de los objetos del procesador que pertenecen a ese grupo. No necesariamente

todos los procesadores contienen a todos los grupos. Un procesador sólo tendrá grupos a los que pertenecen sus objetos según la distribución de carga realizada.

Cuando un grupo contenga, al menos, un objeto $\beta_0$-semejante con el nuevo objeto diremos, por convenio, que el grupo es semejante con el nuevo objeto.

Cada vez que el procesador 0 recibe un nuevo objeto $o$ lo difunde a todos los procesadores. Posteriormente, cada procesador verifi ca si el nuevo objeto es $\beta_0$-semejante con alguno de sus objetos. Al fi nalizar este proceso, todos los procesadores informan al procesador 0 cuáles de sus grupos son semejantes con $o$ mediante una operación de comunicación global del tipo recogida. El procesador 0 procesa esta información realiza una operación de comunicación global de tipo difusión donde informa a todos los procesadores los grupos globales a los que fue semejante el nuevo objeto. Con información, cada procesador une a todos sus objetos de los grupos que fueron semejantes con $o$ en un solo grupo. El procesador al que pertenece el nuevo objeto $o$ (según la distribución de carga realizada) lo agrega al nuevo grupo unión formado. En el caso de que $o$ no fue semejante con ninguno de los grupos existentes, en el procesador dueño de $o$ se crea un nuevo grupo donde se incluye a $o$.

Los pasos seguidos se muestran en el algoritmo 2.

Es bueno destacar algunas características del algoritmo paralelo diseñado, como son:

1. Existe un equilibrio de carga entre los procesadores, pues cada procesador tiene aproximadamente $\frac{n}{p}$ objetos.

2. El cálculo de la semejanza entre el nuevo objeto y los objetos de los grupos istentes se realiza entre todos los procesadores. Cada procesador calcula aproximadamente la misma cantidad de semejanzas.

3. El procesador 0 actúa como maestro, pues él difunde los objetos entre los proce sadores y con la información obtenida entre todos los procesadores determin los grupos que fueron semejantes al nuevo objeto.

4. En un procesador, cuando se encuentra en un grupo un objeto $\beta_0$-semejante cc el nuevo objeto, ya no se compara con ningún otro objeto de ese grupo.

5. El algoritmo diseñado trata de lograr su objetivo minimizando la cantidad comunicaciones. Es conocido que para lograr paralelizaciones efi cientes quiere que la cantidad de cómputo sea superior a la cantidad de comunicacione

## 3.2. Análisis de los costos computacionales

En el algoritmo paralelo obtenido cada procesador almacena $\frac{n}{p}$ objetos. En el de los casos, por cada uno de los $n$ objetos que llegan, en cada procesador se calcul. $\frac{n}{p}$ semejanzas entre objetos. El cálculo de cada semejanza entre un par de objetos un costo de $m$.

El costo de cómputo paralelo es, por tanto:

$$T_{comp} = \frac{t_c n^2 m}{p}$$

**Algorithm 2** Algoritmo GLC paralelo

1. El procesador 0 toma el próximo objeto y difunde a todos los procesadores su descripción.
2. Cada procesador realizar:
   *a*) $ListaMarcados = \emptyset$.
   *b*) Recibir la descripción del nuevo objeto $o$.
   *c*) Desmarcar todos los grupos en la lista de grupos del procesador.
   *d*) Para cada grupo no marcado $g'$ en la lista de grupos del procesador:
      1) Para cada objeto $o'$ de $g'$ hacer:
         - Calcular semejanza $S$ entre $o'$ y $o$.
         - Si $S \geq \beta_0$ hacer:
           Marcar $g'$ e ir a 2 (d).
3. Recogida en el procesador 0 de los grupos marcados en cada procesador.
4. El procesador 0 procesa la lista para obtener todos los grupos semejantes con $o$.
5. Difusión desde el procesador 0 con la lista global de los grupos obtenidos en el paso anterior y el identifi cador $id$ del nuevo grupo a formar.
6. Cada procesador que posea grupos en la lista global, une todos estos grupos en un nuevo grupo con identifi cador $id$.
7. Si es el procesador al que pertenece $o$
   *a*) Si ya creó el nuevo grupo $id$ agregar a $o$ en ese grupo
   Sino, crear un nuevo grupo unitario formado por $o$ cuyo identifi cador será $id$
8. Ir a 1.

Por cada nuevo objeto se realizan dos difusiones y una recogida. La primera operación de difusión (paso 1) tiene un costo de $p(t_s + t_w m)$. Después se realiza una recogida (paso 3) con un costo de $p(t_s + t_w K_1)$, donde $K_1$ es la mayor cantidad de grupos semejantes al nuevo objeto existentes en un procesador. Posteriormente, se produce segunda difusión (paso 5), que tiene un costo de $p(t_s + t_w K_2)$, donde $K_2$ es la mayor cantidad de grupos semejantes al nuevo objeto de forma global. Note que $K_2 \geq K_1$.

En el análisis de estos costos hemos supuesto que las difusiones y la recogida efectúan mediante $p$ mensajes, lo cual constituye el peor de los casos. Dependiendo de la red de interconexión que se utilice estas operaciones se pueden lograr con menor cantidad de mensajes. Por ejemplo, en una red de difusión la recogida se logra con mensajes pero la difusión se logra con un solo mensaje.

Por tanto, el costo de las comunicaciones, en el peor de los casos, es:

$$T_{comm} = np(3\,t_s + t_w(m + 2K_2))$$

El costo espacial es:

$$S_{par} = \frac{s_w nm}{p}$$

pues en cada procesador se almacenan los $m$ rasgos de $\frac{n}{p}$ objetos.

El costo temporal del mejor algoritmo secuencial para resolver el problema tratado es:

$$T_{sec} = t_c n^2 m$$

Por tanto, concluimos que la eficiencia del algoritmo paralelo es:

$$E = \frac{t_c n^2 m}{t_c n^2 m + np^2(3\,t_s + t_w(m + 2K_2))}$$

Si consideramos que $t_c nm \gg t_s p^2$, entonces la eficiencia alcanzada con nuestro algoritmo paralelo tiende a 1. Esta suposición no es difícil de lograr, pues se intenta paralelizar un algoritmo, precisamente, cuando la cantidad de objetos es muy grande.

## 4. Resultados experimentales

El algoritmo propuesto se implementó utilizando el estándar de programación paralela de paso de mensajes MPI. Para la realización de los experimentos se utilizó un cluster de computadoras Pentium II con 256K de memoria cache, 600MHz y 128 de RAM. Estas computadoras están enlazadas por una red myrinet conectada mediante switcher. El sistema operativo utilizado fue Linux Suse y la versión de MPI empleada fue MPICH-1.2.2.

Para determinar los parámetros de esta máquina paralela se realizó el test del pon y se obtuvo que $t_s = 1{,}66E-4$ y $t_w = 5{,}34E-8$. Para la estimación propio del algoritmo GLC, es decir, del tiempo que demora el procesamiento objeto en el algoritmo secuencial se midió el tiempo de ejecución del algoritmo máquina del cluster y se llegó a que $t_c = 4{,}31E-7$. Utilizando el resultado anterior (Vea 3.2), se requiere entonces que $nm \gg 385p^2$ para obtener un algoritmo GLC paralelo eficiente en esta multicomputadora.

Para llevar a cabo el estudio experimental del algoritmo diseñado se generaron varios conjuntos de datos aleatorios. Los conjuntos de datos se realizaron variando la cantidad de objetos en el rango [10000,90000] con objetos de 2 y 8 rasgos. Las componentes de los objetos se generaron utilizando el generador de números aleatorios proporcionado en la librería C de GNU, variando los rangos en que se generan los números de forma tal que la densidad promedio de los conjuntos obtenidos es la misma para todos los conjuntos de datos.

Con los datos así obtenidos se ejecutaron las variantes secuencial y paralela del algoritmo, variando la cantidad de procesadores desde 2 hasta 16. Los resultados obtenidos se muestran en el cuadro 1. En el mismo, para cada combinación de cantidad de datos, dimensión de los datos y cantidad de procesadores se especifica en segundos el tiempo de cálculo obtenido.Como puede observarse, en todos los casos se logra que los tiempos de ejecución del algoritmo paralelo sean menores que los del secuencial.

| | | Procesadores | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Tamaño | Rasgos | 1 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 |
| 10000 | 2 | 159 | 81 | 57 | 49 | 48 | 52 | 51 | 50 | 49 |
| 10000 | 8 | 352 | 111 | 69 | 60 | 54 | 59 | 56 | 54 | 55 |
| 30000 | 2 | 1532 | 720 | 500 | 424 | 386 | 387 | 371 | 360 | 354 |
| 30000 | 8 | 3315 | 966 | 645 | 508 | 435 | 428 | 410 | 394 | 375 |
| 50000 | 2 | 4436 | 2230 | 1424 | 1973 | 1102 | 1072 | 1007 | 986 | 963 |
| 50000 | 8 | 10061 | 2968 | 1768 | 2171 | 1197 | 1186 | 1088 | 1058 | 1005 |
| 70000 | 2 | 9529 | 4542 | 2932 | 2467 | 2195 | 2120 | 2006 | 1943 | 1873 |
| 70000 | 8 | 17275 | 5878 | 3472 | 2824 | 2393 | 2334 | 2147 | 2034 | 1939 |
| 90000 | 2 | 14160 | 8225 | 5612 | 4103 | 3618 | 3553 | 3343 | 3223 | 3119 |
| 90000 | 8 | 28150 | 10456 | 6523 | 4650 | 3975 | 3847 | 3564 | 3387 | 3239 |

Cuadro 1: Resultados obtenidos en la experimentación.

En la figura 1 están graficados los tiempos obtenidos contra el tamaño de los conjuntos de datos utilizando los conjuntos de datos con objetos de 8 rasgos. En la misma se muestran las curvas para algunas cantidades de procesadores. En ella puede observarse el comportamiento cuadrático del algoritmo GLC secuencial. Además se observa cómo a medida que aumenta la cantidad de procesadores disminuyen los tiempos de cómputo. Un comportamiento similar se observa con las restantes dimensiones y cantidades de procesadores.

La figura 2 muestra las aceleraciones ($A$) contra la cantidad de procesadores para todas las cantidades de objetos de 8 rasgos. Como vemos la aceleración tiene un comportamiento cuasilineal cuando el tamaño del problema es suficientemente grande. En este caso se observa un comportamiento cercano al ideal.



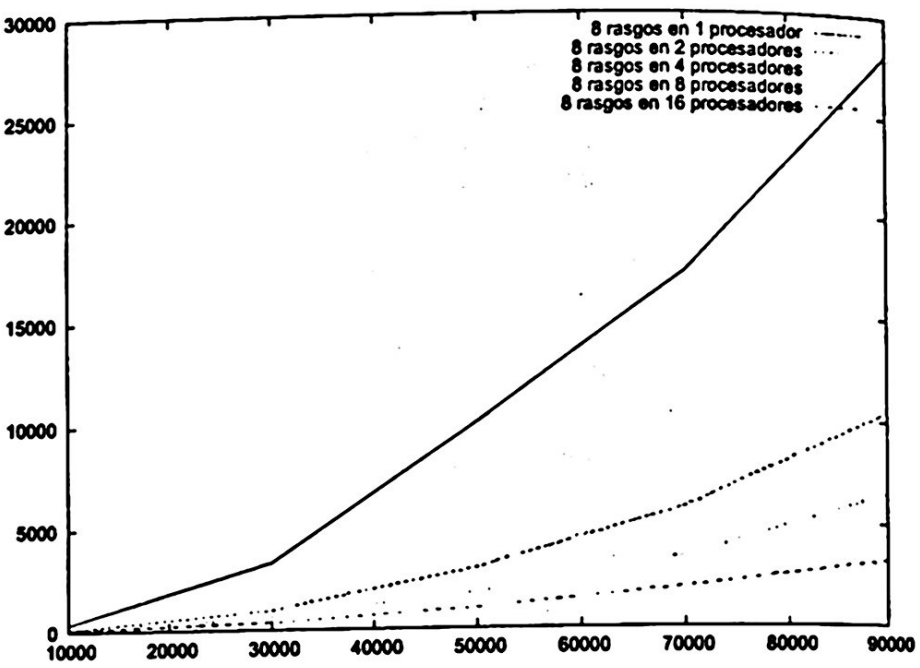



Figura 1: Tiempos obtenidos con objetos de 8 rasgos.

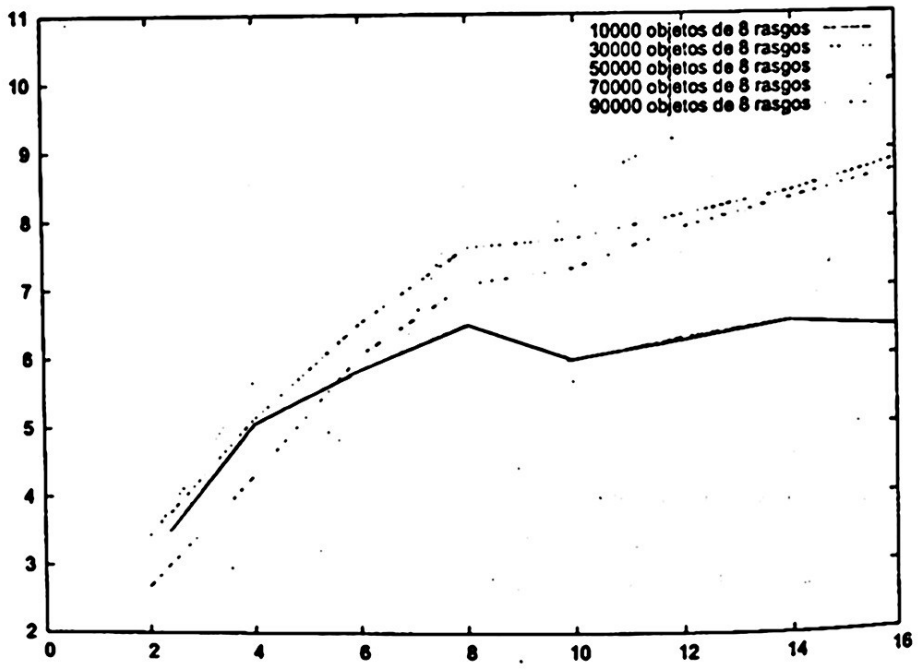



Figura 2: Aceleraciones para algunos tamaños del problema.

También puede observarse superaceleraciones cuando la cantidad de procesadores es pequeña y los conjuntos de datos son de pequeño o medio tamaño, lo que consideramos se debe a la influencia del caché del procesador. Esto puede ocurrir debido a que si al dividir los datos entre los procesadores, todos los datos del problema caben en la cache de los procesadores se aceleran los cálculos. También se observa un comportamiento anómalo del agrupamiento con 10000 objetos lo que consideramos se deba a algún factor externo que haya pertubado el comportamiento de la máquina paralela durante la ejecución de ese conjunto de datos.

# 5. Conclusiones

En el trabajo se expone la paralelización del algoritmo de agrupamiento incremental GLC y el análisis de su complejidad computacional. Como resultado se obtuvo una paralelización que aprovecha efi cientemente los recursos de una computadora paralela, pues se logra un buen equilibrio de carga entre los procesadores y una efi ciencia teórica ideal cuando la cantidad de objetos es mucho mayor que la cantidad de procesadores.

Los resultados experimentales obtenidos muestran las bondades del algoritmo paralelo propuesto al lograrse disminuir los tiempos de ejecución del algoritmo secuencial según aumenta el número de procesadores empleados. Estos resultados se corresponden a los resultados teóricos obtenidos, pues se logran aceleraciones cercanas a la cantidad de procesadores y éstas se incrementan en la medida en que aumentan la cantidad de objetos y la cantidad de rasgos que los describen.

Como trabajo futuro se preve profundizar en el estudio experimental del algoritmo propuesto sobre otras arquitecturas paralelas y con una mayor cantidad de procesadores. También pretendemos emplear el algoritmo paralelo diseñado para la detección de sucesos en grandes volúmenes de noticias periodísticas.

# Referencias

[1] Greengrass, E. (2000). Information Retrieval: A Survey, *Technical Report.*

[2] Karypis, G. & Han, E, & Kumar, V. (1999). Chameleon: A hierarchical clustering algorithm using dynamic modeling, *IEEE Computer*: Special Issue on Data Analysis and Mining, Vol. 32, No. 8, 68-75.

[3] Allan, J. & Carbonell, J. & Doddington, G. &Yamron, J. & Yang, Y. (1998) Topic Detection and Tracking Pilot Study: Final Report, *Proceeding of DARPA Broadcast News Transcription and Understanding Workshop*, 194-218.

[4] Nagesh, H. & Goil, S. & Choudhary, A. (1999). MAFIA: Effi cient and Scalable Subspace Clustering for Very Large Data Sets, *Technical Report No. CPDC-TR-9906-010*, Center for Parallel and Distributed Computing.

[5] Gil-García, R. & Pons-Porrata, A. (2000). Parallel Agglomerative Hierarchical Clustering Algorithm in bus based networks, *Proceeding of V Ibero American Symposium of Pattern Recognition*, 691-696.

[6] Olson, C. (1995). Parallel algorithms for hierarchical clustering , *Parallel Computing* 21, 1313-1325.

[7] Dhillon, I &Modha, B. A. (2000). Data Clustering Algorithm On Distributed Memory Multiprocesor,Workshop on Large-scale Parallel KDD Systems, 1999, 245-260.

[8] Hill, D.R. (1968). A vector clustering technique, *Samuelson (ed.), Mechanized Information Storage, Retrieval and Dissemination*, North-Holland, Amsterdam.

[9] Aslam, J. & Pelekhov, K. & Rus, D. (1998). Static and Dynamic Information Organization with Star Clusters, *Proceedings of the 1998 Conference on Information Knowledge Management, CIKM 98, Baltimore, MD.*

[10] Sánchez, G. (2001). *Desarrollo de algoritmos para el agrupamiento de grandes volúmenes de datos mezclados*, Tesis Doctoral, Centro de Investigación en Computación, IPN, México.

[11] Shulcloper, J.R. & Martínez, J.F. (1995). *Clasificación Sin Aprendizaje y Con Aprendizaje Parcial (Enfoque Lógico Combinatorio)*, Centro de Investigación de Estudios Avanzados del I.P.N, Dpto. de Ingeniera Eléctrica, México.

[12] Wilkinson, B. & Allen, M. (1999). *Parallel programing: techniques and applications using networked workstations and parallel computers,* Prentice Hall.

[13] Foster, I. (1995). *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*, Addison-Wesley Publishing Company.

# Análisis Comparativo de Tres Sistemas de la Programación Lógica Inductiva

Hilda Escobedo Hernández

Instituto de Investigaciones Eléctricas
escobedo@iie.org.mx

**Resumen**

En este artículo se presenta un análisis comparativo entre tres sistemas representativos de la Programación Lógica Inductiva(PLI): GOLEM, FOIL y PROGOL aplicados a la Minería de Datos. A la fecha se han reportado aplicaciones exitosas de la PLI, sin embargo, en la literatura especializada no aparece un análisis comparativo formal desde el punto de vista del usuario. En este sentido, en el presente trabajo se comparan dichos sistemas bajo tres dimensiones: la facilidad de uso del sistema, la estrategia de búsqueda y la facilidad de expresar un conocimiento previo. Para ejemplificar este estudio se utilizó una base de datos real. Los resultados experimentales mostraron, por una parte, que sistemas cumplen mejor en cada una de las dimensiones, y por otra, que no hay una dominación de un sistema sobre los otros. Finalmente, consideramos que la caracterización de los sistemas puede ser usada para la selección de un sistema, dada una aplicación especifica.

## 1. Introducción.

La mayoría de los sistemas de aprendizaje que se han aplicado a Minería de Datos están basados en la lógica proposicional y han demostrado su validez al aplicarlos sobre conjuntos de datos reales[16], sin embargo, presentan tres limitaciones importantes que restringen su campo de aplicación [4] a) usan limitado formalismo para la representación del conocimiento, b) tienen dificultad en incorporar en su proceso de aprendizaje un conocimiento previo, y c) tienen restricciones en su vocabulario. La primera restricción dificulta la representación de ciertos dominios que no pueden ser expresados de manera efectiva a nivel proposicional, la segunda es importante porque lo ideal en cualquier sistema de aprendizaje es que el conocimiento descubierto pueda ser almacenado y utilizado para descubrimientos posteriores, la tercera restricción se debe a que sería muy deseable que los sistemas realicen invención de nuevos predicados que ayudarán a definir mejor el conjunto de entrenamiento. Los sistemas de la PLI intentan vencer este tipo de limitaciones [11], utilizando un lenguaje de representación más potente: la lógica de predicados de

primer orden, y hacen uso de un conocimiento previo en su proceso de aprendizaje, además algunos sistemas tienen incorporada la facilidad para realizar invención nuevos predicados.

El Objetivo central de la PLI es construir un programa lógico que junto con conocimiento que se tenga del dominio, obtenga como consecuencia lógica conjunto de entrenamiento. Generalmente, el programa lógico que se construye forma de varias cláusulas hipótesis, por lo que de ahora en adelante le llamaremos teoría que denotaremos como $\Sigma$. Para encontrar a $\Sigma$, es necesario recorrer un espacio de búsqueda[12]. La dificultad de encontrar una $\Sigma$ correcta esta determinado por tamaño del espacio de búsqueda y las propiedades de sus relaciones de generalidad que son consideradas en los sistemas de la PLI.

En este trabajo analizaremos las técnicas de generalización y las heurísticas utilizan los sistemas: GOLEM, FOIL y PROGOL para evitar tratar con espacios búsqueda complejos, lo cual nos permitirá mostrar las ventajas y limitaciones entre estos sistemas para encontrar a una $\Sigma$ correcta. Según sea el método con el que recorre el espacio de posibles cláusulas, podemos distinguir varias técnicas: generalización relativa menos general bajo subsunción que es la que utiliza el sistema GOLEM [6], la búsqueda de grafos refinados que utiliza FOIL y la deducción inversa (en inglés, *inverse entailment*) que utiliza PROGOL.

En la sección 2, se presenta una descripción de cada sistema donde se explica como se obtiene a $\Sigma$ y se explora la estrategia de búsqueda que ofrece mejores resultados entre estos sistemas. En la sección 3, se presenta una aplicación a Minería de Datos utilizando los tres sistemas bajo estudio en una Base de Datos real, la representa una muestra estadística de pacientes que han sufrido un infarto, en aplicación se pretende encontrar reglas que nos permitan identificar de acuerdo ciertas características de los pacientes cuales tienen probabilidades de sobrevivir.

## 2 Descripción de los Sistemas.

En la terminología de Base de Datos, los sistemas de PLI construyen definicione para una relación de la base de datos, fijada como relación objetivo o relación por el usuario. y para ello se utiliza el resto de predicados de la Base de Datos En esta sección se presentan las principales características de los sistemas estudio comparando los mecanismos de búsqueda que utilizan para encontrar teoría ($\Sigma$), que mejor explique al conjunto de entrenamiento, concluyendo con el ofrece mejores resultados.

## 2.1 GOLEM.

El sistema GOLEM utiliza una búsqueda *bottom-up* (de lo específico a lo general), sigue el método de generalización llamado Generalización Relativa Menos General bajo Subsunción (por sus siglas en ingles, RLGGs) [1]. Para evitar el problema de encontrar una RLGGs infinita el conocimiento previo es definido extensionalmente, es decir; por medio de un conjunto de hechos (cláusulas unitarias) [4]. La RLGGs construida por GOLEM, fue forzada a tener un número tratable de literales, requiriendo para esto que la posible cláusula hipótesis contenga solamente cláusulas definidas que sean *ij*-determinadas. Estas cláusulas tienen una restricción en la conexión de variables acotadas por los parámetros *i* y *j*. GOLEM acepta como entrada un conjunto de ejemplos positivos, ejemplos negativos y el conocimiento previo, todos estos están representados como hechos.

Para la inducción de la Σ [3], como primer paso elige aleatoriamente un par de ejemplos positivos y computa la RLGGs, usando el conocimiento previo. En el segundo paso, GOLEM vuelve a seleccionar aleatoriamente un ejemplo positivo y construye la RLGGs junto con la cláusula resultante del primer paso, eliminando los ejemplos positivos que satisfaga la hipótesis. El proceso del segundo paso continua mientras incrementalmente sean satisfechos los ejemplos positivos, en caso contrario, se repite el proceso desde el primer paso hasta que se cubran todos los ejemplos positivos o el criterio de detenerlo sea satisfecho. La cláusula de la RLGGs calculada es frecuentemente demasiado especifica y por lo tanto se generaliza eliminando literales del cuerpo de la cláusula hasta que se pueda ejecutar sin satisfacer ningún ejemplo negativo[12]. En GOLEM se introdujo el concepto de cláusulas *ij*-determinadas, que después se usaron en FOIL y PROGOL.

El sistema GOLEM usa un algoritmo de reducción *bottom-up*. Pero dicho algoritmo no garantiza una optimización en la Σ final, produciendo cláusulas dentro de esta que cubre ejemplos negativos. Una ventaja de este sistema es su facilidad de uso, solo se le deberá indicar el conjunto de ejemplos positivos, ejemplos negativos y el conocimiento previo en archivos por separado. La desventaja de utilizar este sistema es que solo puede usarse en dominios pequeños acotado por el número de cláusulas que pueden ser almacenadas (aproximadamente 10000 [3]). También resulta un proceso tedioso el de generar todos los hechos que representan al conocimiento previo, o quizá el usuario puede no conocerlo. Este sistema no tiene la facilidad de inventar nuevos predicados para una mejor definición de la relación objetivo, además este sistema es susceptible al ruido.

## 2.2 FOIL.

El sistema Foil (versión 1992), es considerado una extensión natural de algunas ideas de los sistemas proposicionales, para ser utilizado con la lógica de primer orden. FOIL es un sistema del tipo de búsqueda *top-down* (de lo general a lo especifico) usa la técnica de grafos refinados [7,12] (la principal técnica de especialización en la

PLI), para la cual emplea varias heurísticas entre ella la más usada es la llamada ganancia de información, la cual apoya una especialización bajo la capacidad de discriminar entre los ejemplos positivos y negativos. Similar al GOLEM requiere que el conocimiento previo sea definido extensionalmente. La forma en que son definidos los ejemplos positivos, ejemplos negativos y el conocimiento previo es por medio de tuplas. El proceso inductivo lo empieza con una cláusula muy general, la cual es gradualmente especializada agregando literales al cuerpo. La cabeza de la cláusula es siempre la literal de la relación objetivo con variables únicas para cada posición de sus argumentos. Por ejemplo: padre(X,Y):- ?, en está cláusula se representa relación objetivo "padre" la tarea de FOIL es buscar las literales del cuerpo de cláusula, eligiendo el nombre del predicado y las variables a usar. Si se introduce una nueva variable en una nueva literal, las tuplas se tienen que extender para incluir los valores de esa variable, esto se hace de manera automática por FOIL. El añadir una literal en el cuerpo de una cláusula sirve para dos propósitos [10], el primero, incrementar la proporción de ejemplos positivos cubiertos y el segundo servir para introducir nuevas variables necesarias para construir la $\Sigma$ final. Las literales del primer tipo, se refieren a literales que son evaluadas usando la heurística información (la que obtenga mayor ganancia de información es la que se agrega), y segunda clase de literales son llamadas literales *ij*-determinadas que fueron utilizadas motivados por el éxito de GOLEM.

Los ejemplos negativos para FOIL, pueden ser proporcionados por el usuario indicarse que sean construidos automáticamente, de acuerdo a la suposición mundo cerrado, donde toda la información que no este presente explícitamente en ejemplos positivos son negativos. FOIL utiliza como lenguaje para expresar teorías un dialecto de Prolog e incluye algunas características adicionales [10] como son: heurística para la poda del espacio de búsqueda de literales, métodos para incluir igualdad y negación de fallo.

Una ventaja de usar este sistema es que es de fácil uso, pero el usuario deberá definir la siguiente información: todas las posibles constantes del conjunto entrenamiento, indicar cuales predicados representan a la relación objetivo y los representan al conocimiento previo, además deberá acotar los argumentos de predicados. Este sistema al igual que GOLEM, sólo puede utilizarse en dominios pequeños limitado por la capacidad de almacenamiento de las tuplas.

## 2.3 PROGOL.

El sistema Progol, es un sistema de tipo de búsqueda *top-down* [5]. Usa aproximación al problema general de la PLI llamado deducción inversa (en ingles, *inverse entailment*), cuenta con un interprete de Prolog estándar argumentado capacidades inductivas. El conocimiento previo puede ser definido de dos formas: extensionalmente e intensionalmente, es decir, por medio de un conjunto de hechos reglas respectivamente, permitiendo por medio de las reglas expresar un meta-

conocimiento del dominio, está es una ventaja muy importante comparada con los sistemas que aquí analizamos. Para usar este sistema, el usuario debe utilizar un lenguaje tipificado de expresiones de primer orden (cláusulas definidas) las cuales serán usadas como el "espacio de las hipótesis" [2]. Las expresiones son definidas usando "declaraciones de modo", en las cuales el usuario especifica que predicados serán usados en la cabeza y cuáles en el cuerpo de las cláusulas hipótesis.

Para la inducción de la $\Sigma$ [5], el proceso consiste en seleccionar un ejemplo positivo e+ para ser generalizado, construyendo la cláusula más especifica llamada la *cláusula bottom* representada por "$\perp$", que debe implicar lógicamente al ejemplo seleccionado según las restricciones del lenguaje, tal que $B \wedge \perp \vdash e+$, donde B representa el conocimiento previo. Sobre esta cláusula PROGOL realiza una búsqueda *top-down* acotada por: "cláusula-vacía" $\leq H \leq \perp$ (donde H es la hipótesis a búscar). Para esta búsqueda, PROGOL lista todas las cláusulas que subsumen a la *cláusula bottom*, a partir de esta lista se realiza la búsqueda de la cláusula más general. En su estrategia de búsqueda utiliza el algoritmo A*, el cual es guiado por una medida de predicción/comprensión, la cláusula que ofrezca mejor puntuación es la que garantiza la máxima comprensión de los datos y es agregada a la base de conocimiento, todos los ejemplos que cubra esta cláusula son eliminados del conjunto de entrenamiento. El proceso continua desde el primer paso hasta que todos los ejemplos positivos sean satisfechos.

Las declaraciones de modos no solamente definen el lenguaje de las hipótesis sino también define la separación entre ejemplos (positivos y negativos) y el conocimiento previo. PROGOL, ofrece una serie de parámetros para controlar el proceso de generalización [14]. Este sistema tiene la capacidad de introducir nuevos predicados cuando sea necesario, es decir, puede proponer un predicado útil que no se encuentre ni en los ejemplos ni en el conocimiento previo y que describa mejor la relación objetivo.

En PROGOL, la técnica que utiliza para construir la cláusula bottom "$\perp$" para reducir el espacio de búsqueda ya existía en GOLEM, sin embargo con GOLEM se tiene que definir el conocimiento del dominio de manera extensionalmente mientras que en PROGOL puede ser definido intensionalmente. Aunque FOIL y PROGOL usan la misma dirección de búsqueda, PROGOL realiza una búsqueda sistemática acotada por: "cláusula-vacía" $\leq H \leq \perp$.

## 3 Resultados Experimentales.

En esta sección se presentan los resultados obtenidos al aplicar los sistemas: GOLEM, FOIL y PROGOL, utilizando una Base de Datos real, que contiene el

historial clínico de una muestra estadística de 120 pacientes que han sufrido un infarto en seis diferente lugares del corazón[1].

Para esta aplicación se pretende encontrar reglas que nos permitan identificar acuerdo a ciertas características de los pacientes cuales tienen probabilidades sobrevivir o no. Los resultados obtenidos se presentan en una tabla que permite comparar los resultados obtenidos con base al porcentaje de los ejemplos que lograron clasificarse dentro de las reglas encontradas y cuales de estos están clasificados correctamente.

### 3.1 Descripción de los Datos.



Objetivo: Encontrar las características de los pacientes que tienen riesgo de muerte pueden sobrevivir a causa de un infarto con base a las variables que se consideran, cuales son: edad, sexo, diabetes y localización del infarto.

Los atributos relevantes elegidos de la base de datos fueron:

1.- Nombre: Edad
Tipo : Nominal (existen tres grupos de edades: uno, dos, tres)
Descripción: Indica la edad del paciente. Grupo uno (1 a 24 años), grupo dos (25 55 años) grupo tres (56 en adelante)

2.- Nombre: Sexo
Tipo :Nominal (mas y fem)
Descripción: indica el sexo del paciente: masculino y femenino.

3.- Nombre: Diabetes
Tipo :Nominal (valores posibles: si, no)
Descripción: El valor si indica que el paciente tiene diabetes y no lo contrario.

4.- Nombre: locinf
Tipo :Nominal (existen seis lugares posibles: antsep, diaf, antlat, posinf, antext y sub)
Descripción: indica seis posibles lugares del corazón donde se puede localizar infarto.

Para esta aplicación se utilizaron los tres sistemas, los cuales se aplicaron con en el mismo objetivo planteado. Para la representación del conocimiento se utilizaron predicados que representaron a los ejemplos positivos, ejemplos negativos conocimiento previo. Para representar a los ejemplos, se utilizó el predicado llamado "evento", el cual representa las características que tiene el paciente que ha sufrido

[1] Esta muestra representativa se obtuvo en el historial clínico de pacientes del Hospital Civil de la ciudad de Xalapa, Veracruz.

infarto y el lugar del corazón donde este se produjo. Por ejemplo: evento(tres, mas, no, antext), significa que el paciente esta en el grupo de edad tres, es de sexo masculino, no tiene diabetes y sufrió el infarto en la parte del corazón llamada "antext".

Ejemplos positivos:

```
evento(tres, mas, no, antext).
evento(tres,fem,no,antsep).
evento(dos,fem,si,diaf).
evento(uno,mas,no,antlat).
evento(dos,mas,no,si,sub).
....
```

Para los ejemplos negativos se consideraron a todos los ejemplos que no son positivos y se utilizaron en cada tipo de sistema bajo estudio. Para representar el conocimiento previo o conocimiento del dominio se analizó la información contenida en la base de datos, donde se observó que se podían formar tres conjuntos de pacientes, representados por tres tipos de predicados. El primero representaría a los pacientes que no tienen riesgo de muerte a causa de un infarto, el segundo a pacientes que si lo tienen y el tercer conjunto representaría a los pacientes que con las mismas características algunas veces lograban sobrevivir y otras veces no. Para representar este conocimiento se utilizaron tres predicados a los que llamamos: clase_no, clase_si, y clase_nosesabe, los cuales nos representaban a cada conjunto respectivamente. Es importante señalar que en nuestra Base de Datos se tenían más características de los pacientes que pertenecían a la clase_no y clase_nosabe, por lo que los resultados obtenidos lograron mejor clasificación para estas. A continuación damos algunos ejemplos del conocimiento previo.

Conocimiento previo:

```
clase_no(tres,mas,si,diaf).
clase_no(dos, fem,no,sub).
clase_no(uno,fem,no, diaf).
....
clase_nosesabe(dos,fem,si,antext).
clase_nosesabe(dos,fem,no,antsep).
...
clase_si(dos,fem,si,antlat).
...
```

### 3.2 Resultados Obtenidos

A continuación presentamos las teorías Σ que se obtuvieron al utilizar los sistemas: GOLEM, FOIL y PROGOL.

**GOLEM**

```
evento(tres,mas,A,antlat).
evento(A,B,si,antsep).
evento(uno,mas,no,antext).
evento(dos,B,C,diaf).
evento(A,mas,no,diaf).
evento(tres,B,no,D):- clase_no(tres,B,no,D).
evento(uno,B,no,D):- clase_no(uno,B,no,D).
evento(A,B,no,antsep):- clase_nosesabe(A,B,no,antsep).
evento(dos,fem,si,D):- clase_si(dos,fem,si,D).
```

**FOIL**

```
evento(A,B,C):- C= antsep, A= uno.
evento(A,B,C):- C <> postinf, C<> sub, C<> antlat, C<> antext, B=mas.
evento(A,B,C):- C= antext, A<> uno, B=fem.
evento(A,B,C):- A<>dos, B=mas, C=antlat.
evento(A,B,C):- C= antsep, clase_nosesabe(A,B,D,C).
evento(A,B,C):- C=antlat, clase_no(A,B,D,C).
evento(A,B,C):- clase_no(A,B,D,C), B=mas.
evento(A,B,C):- clase_nosesabe(A,B,si,C).
evento(A,B,C):- clase_nosesabte(A,B,si,C)
```

**PROGOL**

```
evento(uno,mas,no,antext).
evento(dos,fem,sii,diaf).
evento(dos,fem,no,antext).
evento(tres,mas,si,antlat).
evento(tres,fem,si,antext).
evento(uno,mas,A,B):- clase_no(uno,mas,A,B).
evento(uno,fem,A,B):- clase_no(uno,fem,A,B).
evento(dos,mas,A,B):- clase_no(dos,mas,A,B).
evento(dos,fem,A,B):- clase_nosesabe(dos,fem,A,B).
evento(tres,mas,A,B):- clase_no(tres,mas,A,B).
evento(tres,fem,A,B):- clase_nosesabe(tres,fem,A,B).
```

En los resultados obtenidos se presenta reglas y excepciones. Las reglas nos permitirán la clasificación de los ejemplos y las excepciones representan a los ejemplos para los cuales no se logro una generalización. En la Tabla 1, se comparan los resultados obtenidos con base al porcentaje de los ejemplos que lograron clasificarse y de estos se representa el porcentaje de ejemplos clasificados correctamente.

**Tabla 1.** Comparación de resultados con base a ejemplos cubiertos y cubiertos correctamente.

| Sistemas de la PLI | % Ejemplos Cubiertos | % Ejemplos Cubiertos Correctamente |
|---|---|---|
| GOLEM | 52 | 73 |
| FOIL | 85 | 82 |
| PROGOL | 98 | 84 |

En la Tabla 1, los resultados demuestran que los sistemas FOIL y PROGOL ofrecen mejores resultados de clasificación (por el número de ejemplos clasificados), sin embargo, GOLEM obtiene resultados sin necesidad de que el usuario especifique un control en su proceso de generalización, aunque la teoría obtenida es más especifica. La teoría del sistema FOIL presenta reglas más generales que logran una mejor clasificación comparada con GOLEM, sin embargo, se requirió hacer varias pruebas acotando sus diferentes argumentos hasta lograr obtener resultados. En la aplicación sólo se obtuvieron resultados acotando la variable: localización de infarto. El sistema PROGOL encontró una teoría que permite clasificar mejor los ejemplos con base a las variables: edad y sexo, para obtener esta teoría se guío la búsqueda especificando en las "declaraciones de modo" utilizadas por este sistema, las variables de entrada fueron: edad y sexo, y las variables de salida: diabetes y localización del infarto, además se utilizaron varios parámetros para controlar el proceso de generalización, indicando: el número máximo de literales en la cláusula, profundidad de sus variables, utilización de ejemplos negativos y manejo de ruido, entre otros.

## 4 Conclusiones.

Con base en el análisis y la experimentación realizada se observó que el sistema PROGOL obtiene teorías con mejores resultados en términos del porcentaje de ejemplos clasificados y el porcentaje de ejemplos clasificados correctamente, esto se debe a que realiza una búsqueda sistemática acotada. Las facilidades de este sistema son por un lado, que el usuario puede proporcionarle por medio de reglas un meta conocimiento del dominio, favoreciendo con esto, que puede ser aplicado en dominios grandes (sin limitaciones) y por otro lado, que este sistema permite realizar invención de nuevos predicados para una mejor definición de la relación objetivo. Sin embargo, tiene como desventaja la dificultad de su uso, en el sentido de que el usuario debe conocer el tipo de patrones que pueden ser descubiertos en la base de datos para guiar la búsqueda y controlar el proceso de generalización por medio de varios parámetros que deberá proporcionar.

Por otra parte, los sistemas GOLEM y FOIL tienen más dificultad de encontrar teoría debido a su estrategia de búsqueda. En particular, las reglas encontradas FOIL son más generales por lo que logran una mejor clasificación. La ventaja utilizar estos sistemas es su facilidad de uso. La desventaja es que solo puede usarse en dominios pequeños, que se encuentran limitados a la capacidad almacenamientos de sus datos. Otra desventaja de estos sistemas es que requieren postprocesamiento en la teoría final por parte del usuario, ya que puede contener cláusulas que satisfacen ejemplos negativos o que no cubran ningún ejemplo.

## Agradecimientos.

Agradezco al Dr. Luciano García Garrido de la Universidad de Matemáticas Habana, Cuba. Al Dr. Joaquin Peréz Ortega y Dr. Guillermo Rodriguez Ortiz, Instituto de Investigaciones Eléctricas por sus valiosos comentarios en el desarrollo este trabajo.

## Referencias.

[1] Muggleton,S., Feng, C.: Efficient induction of logic program, the Turing Institute(1992).
[2] Muggleton,S., Firth, J.: Cprogol4.4 a tutorial introduction, Department of computer science, University of York. 2 (1994)
[3] Morales, E.:Learning chess patterns in Inductive Logic Programming, London, Academ Press, tesis ph (1992)
[4] Muggleton,S., De Raedt, L.:Inductive Logic Programming: Theory and Methods. Journal of Logic Programming (1994) 19/20:629-679.
[5] Muggleton, S.: Inverse Entailment and PROGOL, Oxford University computation Laboratory,United kingdom (1995).
[6] Nienhuys-Cheng,S.H., De Wolf, R.: Least Generalization and Greatest Specializations of sets of clauses pp.341-363 (1996).
[7] Quinlan,J.R.,Cameron-Jones,R.M: Induction of Logic Programs: FOIL and related System (1992).
[8] Piatetsky-Shapiro,G.,William, F.J.: Knowledge Discovery in Databases (1991) pp. 1-27.
[9] Jorg-Uwe, K., Wrobel, S: Controlling the complexity of Learning in Logic through syntactic and task-oriented models (1992).
[10] Cameron_Jones, R.,M., Quinlan R.,J.: Efficient top-down Induction of Logic Programs, University of Sydney,Australia. (1996).
[11] Brantko,I.,King,R.: Applications of inductive Logic Programming, Bulletin Sigart, volumen 5, number 1, (1994).
[12] Nienhuys-Cheng,S.H., De Wolf: Foundations of Inductive Logic Programming, LNAI-tutorial, 1228. Springer-Verlag, Berlin (1997).
[13] Muggleton, S.: Inductive Logic Programming, derivations, successes and shortcomings. Bulletin Sigar, volumen 5, number 1, (1994).
[14] Sam,R.: An Introduction to Progol, (1997).
[15] Blockeel, H., De Raedt, L.: Relational Knowdge Discovery in Databases, Department Computer Science. (1996).
[16] Fayyad,M.,U,Piatetsky-Shapiro,G.,Smyth,P.,Uthurusamu,R.: Advances in Knowledge Discovery and Data Mining, AAAI Press/The Mit Press (1997).

# Un algoritmo incremental para la obtención de cubrimientos con datos mezclados

Aurora Pons-Porrata[1], José Ruiz-Shulcloper[2], Rafael Berlanga-Llavori[3], Yovanis Santiesteban Alganza[1]
[1] Universidad de Oriente, Santiago de Cuba (Cuba)
aurora@app.uo.edu.cu, yosva@csd.uo.edu.cu
[2] Instituto de Cibernética, Matemática y Física (Cuba)
recpat@cidet.icmf.inf.cu
[3]Universitat Jaume I, Castellón (España)
berlanga@lsi.uji.es

**Resumen**

En este trabajo se introduce un algoritmo incremental eficiente para estructurar un conjunto de datos en conjuntos fuertemente compactos. Este algoritmo se apoya en algunas propiedades de los conjuntos fuertemente compactos que son demostradas en el trabajo. El algoritmo propuesto crea un cubrimiento único del conjunto de datos, por lo que no depende del orden de presentación de los objetos. La complejidad computacional del algoritmo incremental presentado sigue siendo la misma que la de su variante no incremental y, por tanto, mucho más eficiente que la aplicación reiterada de la variante no incremental. El algoritmo propuesto puede ser utilizado en todas las tareas que requieran del agrupamiento de objetos en fuertemente compactos y del procesamiento dinámico de la información, tales como, por ejemplo, la identificación y seguimiento de noticias en un flujo de documentos; el estudio de la morbilidad de una población; la estructuración de la información para estudios socio-económicos; entre otras posibles aplicaciones. Es bueno resaltar que este algoritmo puede utilizarse, además, en problemas donde se deseen agrupar objetos de cualquier naturaleza, descritos por rasgos cuantitativos y cualitativos mezclados, incluso con ausencia de información.

**Palabras clave:** algoritmos incrementales, algoritmos de agrupamiento, conjuntos fuertemente compactos.

## 1. Introducción.

El agrupamiento de datos es uno de los problemas centrales en la Minería de Datos. Entre las técnicas de agrupamiento existentes, el criterio de agrupamiento en conjuntos fuertemente compactos tiene la propiedad de que la semejanza entre un par de objetos de un mismo grupo es máxima. Los grupos obtenidos por este criterio son solapados y relativamente pequeños y densos. Existen muchas aplicaciones donde se necesita formar grupos de objetos de los que sabemos que por su naturaleza son solapados, por lo que este criterio resulta de utilidad. Existen,

además, otros problemas en los que el conjunto de datos se incrementa en el tiempo. Por ejemplo, la identificación y seguimiento de noticias en un flujo de documentos. Un documento que trate acerca de la mortalidad por el virus VIH en un país dado, aparecerá junto a otros que aborden el tema de la medicina, pero también formará parte de otros conjuntos de documentos, digamos por caso, que hablen de problemas sociales tales como la esperanza de vida al nacer o los que traten sobre los derechos humanos, entre otros. Estas estructuraciones pueden ser de utilidad para analistas información de diferentes perfiles. Para este tipo de aplicaciones se requiere de algoritmo incremental que obtenga los conjuntos fuertemente compactos.

En este trabajo se introduce un algoritmo incremental eficiente para estructurar un conjunto de datos en conjuntos fuertemente compactos. Este algoritmo se apoya en algunas propiedades de los conjuntos fuertemente compactos que demostradas en el trabajo.

## 2. Conceptos y resultados necesarios.

Sea $\zeta$ una colección de objetos y $S$ una función de semejanza entre objetos simétrica. Aquí sólo consideraremos este tipo de función de semejanza. además, $\beta_0$ un umbral de semejanza definido por el usuario.

**Definición 1.** *Se dice que dos objetos $O_i$ y $O_j$ son $\beta_0$-semejantes si $S(O_i,O_j)\geq\beta_0$. para todo $O_j$ de la colección de objetos $\zeta$ se cumple que $S(O_i,O_j)<\beta_0$ entonces $O_i$ denomina $\beta_0$-aislado.*

**Definición 2.** *Diremos que $NU\subseteq\zeta$, $NU\neq\emptyset$ es una componente conexa si se cumple que* [1]:

1. $\forall O_i,O_j\in NU,\ O_i\neq O_j,\ \exists O_{i_1},\ldots,O_{i_q}\in NU[O_i=O_{i_1}\wedge O_j=O_{i_q}\wedge\forall p=1,\ldots,q\text{-}1\ S(O_p,O_{p+1})\geq\beta_0]$.
2. $\forall O_i\in\zeta[O_j\in NU\wedge S(O_i,O_j)\geq\beta_0]\Rightarrow O_i\in NU$.
3. *Todo elemento $\beta_0$-aislado es una componente conexa (degenerada).*

**Definición 3.** *Diremos que $NU\subseteq\zeta$, $NU\neq\emptyset$, es un conjunto compacto si* [1]:

1. $\forall O_j\in\zeta[O_i\in NU\wedge \max\limits_{\substack{O_t\in\zeta\\ O_t\neq O_i}}\{S(O_i,O_t)\}=S(O_i,O_j)\geq\beta_0]\Rightarrow O_j\in NU$.
2. $[\max\limits_{\substack{O_i\in\zeta\\ O_i\neq O_p}}\{S(O_p,O_i)\}=S(O_p,O_t)\geq\beta_0\wedge O_t\in NU]\Rightarrow O_p\in NU$.
3. $|NU|$ *es mínimo.*

*Todo elemento $\beta_0$-aislado constituye un conjunto compacto (degenerado).*

El criterio de agrupamiento basado en los conjuntos compactos forma, al que el de las componentes conexas, una partición.

**Definición 4.** Diremos que $NU\subseteq\zeta$, $NU\neq\emptyset$, es un *conjunto fuertemente compacto* si:

1. $\forall O_j \in \zeta\ [O_i \in NU \wedge \max_{\substack{O_t \in \zeta \\ O_t \neq O_i}} \{S(O_i, O_t)\} = S(O_i, O_j) \geq \beta_0] \Rightarrow O_j \in NU.$

2. $\exists O_i \in NU\ \forall O_j \in NU\ \exists O_{i_1}, \ldots, O_{i_q} \in NU\ [O_i = O_{i_1} \wedge O_j = O_{i_q} \wedge \forall p<q$
$[\max_{\substack{O_t \in \zeta \\ O_t \neq O_{i_p}}} \{S(O_{i_p}, O_t)\} = S(O_{i_p}, O_{i_{p+1}}) \geq \beta_0]].$

3. No existe $NU'$ que cumpla 1. y 2. y $NU \subset NU'$.

Todo elemento $\beta_0$-aislado constituye un conjunto fuertemente compacto (degenerado) [1].

La condición 1 dice que todo objeto de *NU* tiene en *NU* al objeto màs $\beta_0$-semejante. La condición 2 significa que en *NU* existe un objeto tal que para cualquier otro que pertenezca a *NU* existe una sucesión de objetos de *NU* tales que uno es el más $\beta_0$-semejante al siguiente. La condición 3 significa que *NU* es el conjunto más grande que cumple las condiciones 1 y 2. Este criterio forma grupos no disjuntos.

**Ejemplo 1.** Sean $\zeta=\{O_1,O_2,O_3,O_4,O_5,O_6,O_7,O_8,O_9,O_{10},O_{11},O_{12}\}$ y la matriz de semejanza siguiente, obtenida a partir de una función de semejanza *S*:

$$MS = \begin{array}{c} \\ O_1 \\ O_2 \\ O_3 \\ O_4 \\ O_5 \\ O_6 \\ O_7 \\ O_8 \\ O_9 \\ O_{10} \\ O_{11} \\ O_{12} \end{array} \begin{pmatrix} 1.0 & 0.85 & 0.75 & 0.78 & 0.63 & 0.70 & 0.51 & 0.38 & 0.43 & 0.27 & 0.22 & 0.13 \\ & 1.0 & 0.90 & 0.90 & 0.76 & 0.62 & 0.58 & 0.45 & 0.56 & 0.38 & 0.36 & 0.27 \\ & & 1.0 & 0.88 & 0.83 & 0.50 & 0.60 & 0.47 & 0.65 & 0.47 & 0.47 & 0.33 \\ & & & 1.0 & 0.85 & 0.58 & 0.68 & 0.55 & 0.63 & 0.45 & 0.42 & 0.30 \\ & & & & 1.0 & 0.48 & 0.73 & 0.65 & 0.80 & 0.65 & 0.58 & 0.50 \\ & & & & & 1.0 & 0.51 & 0.36 & 0.27 & 0.11 & 0.05 & 0.00 \\ & & & & & & 1.0 & 0.83 & 0.66 & 0.58 & 0.48 & 0.51 \\ & & & & & & & 1.0 & 0.66 & 0.63 & 0.53 & 0.61 \\ & & & & & & & & 1.0 & 0.83 & 0.80 & 0.70 \\ & & & & & & & & & 1.0 & 0.88 & 0.85 \\ & & & & & & & & & & 1.0 & 0.80 \\ & & & & & & & & & & & 1.0 \end{pmatrix}$$

Si consideramos $\beta_0$=0.8, entonces los conjuntos compactos son $NU_1=\{O_1,O_2,O_3,O_4,O_5\}$, $NU_2=\{O_9,O_{10},O_{11},O_{12}\}$, $NU_3=\{O_7,O_8\}$ y $NU_4=\{O_6\}$ y los fuertemente compactos son: $NU_1=\{O_1,O_2,O_3,O_4\}$, $NU_2=\{O_2,O_3,O_4,O_5\}$, $NU_3=\{O_9,O_{10}, O_{11}\}$, $NU_4=\{O_{10},O_{11}, O_{12}\}$, $NU_5=\{O_7,O_8\}$ y $NU_6=\{O_6\}$

**Definición 5.** *Llamaremos grafo basado en la máxima $\beta_0$-semejanza según S, y lo denotaremos máx-S, al grafo orientado $G=(\zeta,E)$ cuyos vértices son los objetos de la colección $\zeta$ y existe un arco del vértice $O_i$ al $O_j$ si se cumple que $O_j$ es el objeto más $\beta_0$-semejante a $O_i$.*

**Proposición 1.** *El conjunto de todos los conjuntos compactos de $\zeta$ coincide con el conjunto de todas las componentes conexas del grafo máx-S asociado a $\zeta$ sin tener en cuenta la orientación.*

Entre las componentes conexas, los conjuntos compactos y fuertemente compactos definidos anteriormente se satisfacen las siguientes relaciones [1]:

1. Toda componente conexa es la unión finita de conjuntos compactos.
2. Todo conjunto compacto es la unión finita de conjuntos fuertemente compactos.
3. Toda componente conexa es la unión finita de conjuntos fuertemente compactos.

**Definición 6.** *Una componente fuertemente conexa de un grafo orientado es un conjunto maximal de vértices en el cual para todo vértice en el conjunto existe un camino a cualquier otro vértice del conjunto* [2]. *Todo objeto $\beta_0$-aislado es una componente fuertemente conexa (degenerada).*

## 3. Nuevas propiedades de los conjuntos fuertemente compactos.

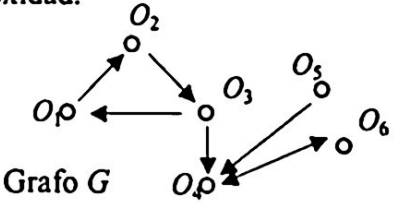
**Definición 7.** *Sea $G=(X,E)$ un grafo orientado y $B \subset X$. B es una base del grafo G si se cumplen las dos condiciones siguientes* [3]:

1. *Todo vértice del conjunto X es descendiente de al menos un vértice de B.*
2. *No existe en el conjunto X un subconjunto de menor número de elementos con dicha propiedad.*

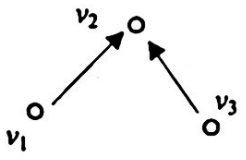
**Definición 8:** *Sean $G=(X,E)$ un grafo orientado y P una partición del conjunto de vértices X, donde $V_i$ denota a cada elemento de P. Se llama grafo reducido de G, y lo denotaremos como $G_r=(V,E_r)$, al grafo cuyos vértices son los subgrafos generados por los elementos $V_i$ de P y existe un arco del vértice $v_i$ al $v_j$ si existe un vértice x en $V_i$ y un vértice y en $V_j$ tal que existe el arco de x a y en G* [3].

**Definición 9.** *Sea G un grafo orientado. Llamaremos grafo reducido según la fuerte-conexidad de G, y lo denotaremos como G/f-C, al grafo reducido de G asociado a la partición inducida por las componentes fuertemente conexas de G. En el grafo G/f-C los vértices son las componentes fuertemente conexas de G y existe un arco de un vértice v a otro v' en el grafo reducido, $v \neq v'$, si existe al menos un arco en G de un vértice en la componente fuertemente conexa que representa v a algún vértice en la componente fuertemente conexa que representa v'.*

**Ejemplo 2.** En la figura siguiente se muestra un grafo *G* y su grafo reducido según la fuerte-conexidad.




Grafo *G*




Grafo reducido según la fuerte-conexidad de *G*

Las componentes fuertemente conexas del grafo $G$ son $v_1=\{O_1,O_2,O_3\}$, $v_2=\{O_4,O_6\}$ y $v_3=\{O_5\}$. Las bases del grafo $G$ son $\{O_1,O_5\}$, $\{O_2,O_5\}$ y $\{O_3,O_5\}$. La base del grafo reducido según la fuerte-conexidad de $G$ es $\{v_1,v_3\}$.

**Proposición 2.** *Sea $G=(\zeta,E)$ el grafo máx-S. El grafo $G_r/f\text{-}C=(V,U)$ tiene una única base formada sólo por los vértices de grado interior nulo.*

Sean $G=(\zeta, E)$ el grafo *máx-S*, $G_r/f\text{-}C=(V,U)$ su grafo reducido, $B=\{b_1,...,b_r\}$ la base de $G_r/f\text{-}C$ y $D(b_i)$ el conjunto de vértices descendientes de $b_i$ en $G_r/f\text{-}C$. Sea, además, $F$: $V\rightarrow\zeta$ tal que $F(v_i)$ es una componente fuertemente conexa en $G$. $F$ es sobreyectiva porque $G_r/f\text{-}C$ es el grafo reducido de $G$.

**Lema 1.** $K=\bigcup_{v_j\in D(b_i)} F(v_j)$ *es un conjunto fuertemente compacto en* $\zeta$.

***Demostración.*** $K\neq\phi$, ya que $D(b_i)\neq\phi$. Probemos que $K$ cumple las siguientes condiciones:

1. $\forall x\in\zeta\,[x'\in K\wedge \max_{\substack{x_j\in\zeta\\ x_j\neq x}}\{S(x',x_j)\}=S(x',x)\geq\beta_0]\Rightarrow x\in K.$

Como $x'\in K$ existe al menos un $j$ tal que $x'\in F(v_j)$, $v_j\in D(b_i)$. Luego, un elemento cualquiera $x$, más $\beta_0$-semejante a $x'$, pertenece a $F(v_j)$ o pertenece a $F(v_t)$, $t\neq j$, $v_t\in V$.

Si $x\in F(v_j)$, entonces $x\in K$. Por el contrario, si $x\in F(v_t)$, implica que en el grafo reducido $G_r/f\text{-}C$ tiene que existir un arco de $v_j$ a $v_t$, por lo que $v_t$ sería descendiente de $v_j$ y, por tanto, de $b_i$. Luego, $v_t\in D(b_i)$ y $x\in K$.

2. $\exists x'\in K\ \forall x\in K\ \exists x_{i_1},...,x_{i_q}\in K\,[x'=x_{i_1}\wedge x=x_{i_q}\wedge\forall p<q$

$$[\max_{\substack{x_t\in\zeta\\ x_t\neq x_{i_p}}}\{S(x_{i_p},x_t)\}=S(x_{i_p},x_{i_{p+1}})\geq\beta_0]].$$

Sea $x'\in K$ tal que $x'\in F(b_i)$. Entonces, para todo $x\in K$ se cumple que $x\in F(b_i)$ ó $x\in F(v_j)$, $v_j\neq b_i$, $v_j\in D(b_i)$. En el primer caso, existen $x_{i_1},...,x_{i_q}\in F(b_i)$ tal que $x'=x_{i_1}\wedge x=x_{i_q}\wedge\forall p<q\,[\max_{\substack{x_t\in\zeta\\ x_t\neq x_{i_p}}}\{S(x_{i_p},x_t)\}=S(x_{i_p},x_{i_{p+1}})\geq\beta_0]]$ porque $F(b_i)$ es una componente fuertemente conexa en el grafo $G$ y, por tanto, $x_{i_1},...,x_{i_q}\in K$. En el segundo caso, como $v_j\in D(b_i)$, existen $v_{i_1},...,v_{i_r}\in D(b_i)$ tal que $b_i=v_{i_1}\wedge v_j=v_{i_r}\wedge\forall p<r\ \exists<v_{i_p},v_{i_{p+1}}>\in U$. Basta con escoger los elementos $x_{i_t}$ $t=1,...,q$ de la manera siguiente: $x_{i_1}=x'$, $x_{i_q}=x$. Como $<v_{i_p},v_{i_{p+1}}>$ y $<v_{i_{p+1}},v_{i_{p+2}}>\in U$ $\exists w,y,z,t$ $w\in F(v_{i_p})$, $y,z\in F(v_{i_{p+1}})$ y $t\in F(v_{i_{p+2}})$ tal que $y$ es el más $\beta_0$-semejante a $w$, $t$ es el más $\beta_0$-semejante a $z$ y como $F(v_{i_{p+1}})$ es

una componente fuertemente conexa en $G$ existe un camino en $G$ entre $y$ y $z$. Luego, existe un camino entre $w$ y $t$ en $G$. Por tanto, existe un camino entre $x'$ y $x$ en $G$.

3. No existe $K' \supset K$ que cumple las condiciones 1 y 2.

Supongamos que $\exists K' \supset K$ que cumple las condiciones 1 y 2. En $K' \setminus K$ no puede existir $x$ que sea el más $\beta_0$-semejante a un $x' \in K$, pues tendría que estar en $K$. Luego, cualquier elemento de $K' \setminus K$ es el más $\beta_0$-semejante de uno de $K' \setminus K$ y/o tiene su más $\beta_0$-semejante en $K$. No puede ocurrir que todos los elementos de $K' \setminus K$ tengan su más $\beta_0$-semejante en $K' \setminus K$, ya que no se cumpliría la condición 2 (no habría manera de llegar de un elemento de $K' \setminus K$ a $K$ ni tampoco habría manera de llegar de un elemento de $K$ a uno de $K' \setminus K$). Luego, tiene que existir $x \in K' \setminus K$ que tenga su más $\beta_0$-semejante en $K$ y no sea el más $\beta_0$-semejante de un $y \in K$. En virtud de lo anterior y de la condición 2 que también cumple $K'$, tiene que existir un $x \in K' \setminus K$ tal que se pueda llegar a cualquier elemento de $K$. Luego, existe $v_j$ tal que $x \in F(v_j)$ y $b_i \in D(v_j)$ y $b_i$ no sería un elemento de la base del grafo reducido $G_r/f\text{-}C$. Por tanto, no puede existir $K'$. ∎

**Teorema.** *El conjunto de todos los conjuntos fuertemente compactos de* $\zeta$ *es:*

$$\bigcup_{i=1}^{r} \left\{ \bigcup_{v_j \in D(b_i)} F(v_j) \right\}$$

***Demostración.*** Supongamos que existe $K'$ conjunto fuertemente compacto en $\zeta$ tal que no existe $b_j \in B$ que cumpla que $K' = \bigcup_{v_t \in D(b_j)} F(v_t)$. Sean $\{P_1,...,P_n\}$ la partición en componentes fuertemente conexas de $G$. Consideremos los $P_{i_1},...,P_{i_m}$ tal que $P_{i_j} \cap K' \neq \phi \ \forall j = 1,...,m$. Demostremos que $P_{i_j} \subset K' \ \forall j = 1,...,m$. Sea $P_{i_j}$ una componente fuertemente conexa de $G$ con $j \in \{1,...,m\}$. Supongamos que $\exists x \in P_{i_j}$ tal que $x \notin K'$. Como $x \in P_{i_j}$, el grado interior de $x$ debe ser diferente de cero. Pero si esto se cumple significa que $\exists x' \in K'$ tal que $x$ es el más $\beta_0$-semejante a $x$ y, en ese caso, $K'$ no sería un conjunto fuertemente compacto. Por lo tanto, $\bigcup_{j=1}^{m} P_{i_j} = K'$. Sean $v_{i_j}, j = 1,...,m$ los vértices del grafo $G_r/f\text{-}C$ correspondientes a $P_{i_j}, j = 1,...,m$. Por condición 2 de la definición de conjunto fuertemente compacto $\exists x' \in K'$ tal que existe un camino entre $x'$ y cualquier elemento de $K'$. Por tanto, entre el $v_{i_t}$ tal que $x' \in F(v_{i_t})$ y cualquier otro $v_{i_t}$, $j \neq t$ existe un camino en $G_r/f\text{-}C$. Luego, $v_{i_j}, j = 1,...,m$, $j \neq t$ son descendientes de $v_{i_t}$ en $G_r/f\text{-}C$. Pero, por la suposición, $v_{i_t} \notin B$, lo que significa que $v_{i_t}$ tiene que ser descendiente de un $b_i \in B$ por

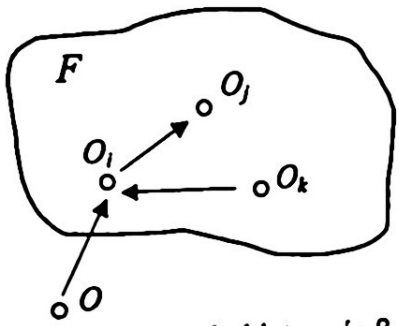
definición de base. Luego, $K' \subset K = \bigcup_{v_j \in D(b_i)} F(v_j)$ y $K'$ no sería fuertemente compacto, pues incumple la tercera condición de la definición. Luego, $v_{i_t} \in B$. ∎

**Corolario.** *El número de conjuntos fuertemente compactos de ζ es igual al cardinal de la base de G_F/f-C.*

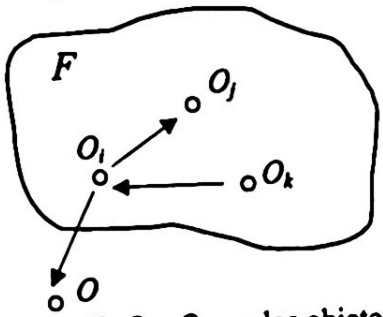
**Definición 10.** *Sea F un conjunto fuertemente compacto (compacto) de una colección de objetos ζ. Decimos que un objeto O está conectado con F si existe al menos un O'∈ F que cumpla una de las dos condiciones siguientes:*

1. *O' es el objeto más $\beta_0$-semejante a O según S.*
2. *O es el objeto más $\beta_0$-semejante a O' según S.*

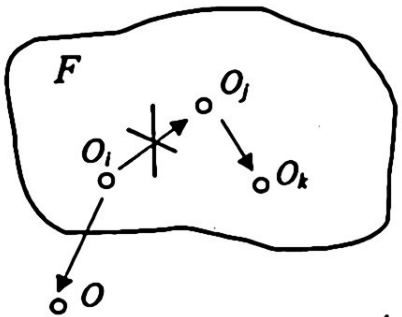
Un objeto $O$ puede conectarse con un conjunto fuertemente compacto (compacto) $F$ de varias formas. La siguiente figura muestra todos los casos que pueden presentarse. En los casos I y II el objeto $O$ no rompe ningún arco del grafo *máx-S* asociado a $F$. En los casos III y IV sí se rompen arcos.



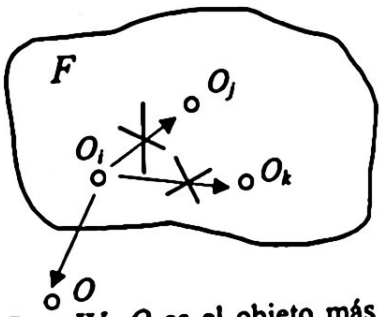

Caso I: $O_i$ es el objeto más $\beta_0$-semejante a $O$.




Caso II: $O$ y $O_j$ son los objetos más $\beta_0$-semejantes a $O_i$.




Caso III: $O$ es el objeto más $\beta_0$-semejante a $O_i$ y $O_j$ deja de serlo.




Caso IV: $O$ es el objeto más $\beta_0$-semejante a $O_i$ y $O_j$ y $O_k$ dejan de serlo.

Sean $\wp$ el conjunto de todos los conjuntos fuertemente compactos de ζ, $F \in \wp$ un conjunto fuertemente compacto y $G_F=(F,U)$ el grafo *máx-S* asociado a $F$. Sea, además, $O$ un objeto tal que $O \notin \zeta$.

**Proposición 3.** *Si O no está conectado con F, entonces F será un conjunto fuertemente compacto de $\zeta \cup \{O\}$.*

**Corolario.** *Si $O$ es $\beta_0$-aislado, entonces el conjunto de todos los conjuntos fuertemente compactos de $\zeta\cup\{O\}$ es $\wp\cup\{\{O\}\}$.*

**Proposición 4.** *Si $O$ es el objeto más $\beta_0$-semejante a un objeto de $F$ y $O$ no rompió ningún arco de $G_F$, entonces $O$ y todos los objetos de $F$ pertenecerán a un mismo conjunto fuertemente compacto de $\zeta\cup\{O\}$.*

***Demostración.*** Como $F$ es un conjunto fuertemente compacto $\exists x\in F\ \forall y\in F$ $\exists x_{i_1},\ldots,x_{i_q}\in F$ [ $x=x_{i1}\wedge\ y=x_{i_q}\wedge\ \forall p<q$ [ $\max\limits_{\substack{x_l\in\zeta\\ x_l\neq x_{i_p}}}\{S(x_{i_p},x_l)\}=S(x_{i_p},x_{i_{p+1}})$ $\geq\beta_0$]] . Luego, en particular, existe un camino en $G_F$ de $x$ a $O'$ y, por tanto, de $x$ a $O$. Por tanto, $F\cup\{O\}$ satisface la condición 2 de la definición de conjunto fuertemente compacto.

Como $O$ es el más $\beta_0$-semejante a un objeto $O'\in F$ pueden ocurrir dos cosas:

a) Existe al menos $O''\in F$ tal que $O''$ es el más $\beta_0$-semejante a $O$ y, en ese caso, $F\cup\{O\}$ cumple la condición 1 y también la condición 3 de la definición de conjunto fuertemente compacto. Por tanto, $F\cup\{O\}$ sería un conjunto fuertemente compacto y $O$ y todos los objetos de $F$ estarían en el mismo conjunto fuertemente compacto.

b) Existe al menos $O'''\in\zeta\backslash F$ tal que $O'''$ es el más $\beta_0$-semejante a $O$ y, en ese caso, existe $F'$ conjunto fuertemente compacto tal que $F'\supset F\cup\{O\}$. Luego, $O$ y todos los objetos de $F$ estarían en el mismo conjunto fuertemente compacto. ∎

**Definición 11.** *Un punto de articulación de un grafo no orientado es un vértice v tal que cuando removemos v y todos los arcos incidentes en él, la componente conexa de v se divide en dos o más partes* [2].

Sean $G'$ el grafo *máx-S* asociado a $\zeta\cup\{O\}$, $C$ un conjunto compacto, tal que $C=\bigcup\limits_{F_i\in\wp}F_i$ , $G_c=(C,U)$ el grafo *máx-S* asociado a $C$, $NG_c$ el grafo $G_c$ sin tener en cuenta la orientación y $H$, el subconjunto de objetos de $C$ que pertenecen a la componente conexa de $O$ en $G'$ sin tener en cuenta la orientación.

**Proposición 5.** *Si $O$ está conectado con $C$ y $O$ rompe uno y sólo un arco de $G_c$ (caso III), entonces si el conjunto $C\backslash H\neq\emptyset$, $C\backslash H$ es un conjunto compacto de $\zeta\cup\{O\}$.*

***Demostración.*** Sea $O_i\in C$ el objeto al cual $O$ es el más $\beta_0$-semejante y tal que se rompe su único arco $a$ que incide al exterior. Si $O_i$ no es un punto de articulación en $NG_c$, entonces al eliminar $a$ el grafo sigue siendo conexo y, por tanto, $H=C$. Si $O_i$ es un punto de articulación en $NG_c$, entonces el grafo $G''=<C,U\backslash\{a\}>$ sin tener en cuenta la orientación no es conexo. Si $C\backslash H\neq\emptyset$, entre cualesquiera dos objetos de $C\backslash H$ existe un camino en $NG_c$ porque, de lo contrario, $C$ no sería un conjunto compacto. Como $O_i$ es un punto de articulación en $NG_c$, no existe en $H$ ningún otro objeto conectado con $C\backslash H$. Por tanto, $C\backslash H$ es una componente conexa en $G'$ sin tener en cuenta la orientación, es decir, $C\backslash H$ es un conjunto compacto en $\zeta\cup\{O\}$. ∎

Note que el único arco que se rompe en el conjunto compacto $C$ se romperá también en los conjuntos fuertemente compactos $F_i$ donde pertenece el objeto $O_l$.

# 4. Algoritmo fuertemente compacto incremental

Paso 1. Cada vez que se presenta un nuevo objeto $O$ se calcula la similaridad con todos los objetos de los conjuntos fuertemente compactos existentes.

Paso 2. Se seleccionan los objetos que son más $\beta_0$-semejantes a $O$ y aquellos a los que $O$ es el más $\beta_0$-semejante. Si no existen tales objetos, en virtud del corolario de la proposición 3, el conjunto $\{O\}$ es un nuevo conjunto fuertemente compacto y terminar.

Paso 3. a) Se construye el conjunto compacto $C(O)$ al que pertenece $O$, con todos los objetos que pertenecen a la componente conexa de $O$ en el grafo *máx-S* de $\zeta \cup \{O\}$ sin tener en cuenta la orientación (proposición 1). Se coloca el conjunto compacto construido en la lista de compactos a procesar.

b) Estos objetos son eliminados de todos los conjuntos fuertemente compactos a los que pertenecían. Sean estos conjuntos $FC_1,...,FC_m$ los cuales son eliminados de $\wp$.

Paso 4. Para cada objeto del conjunto $K = \left( \bigcup_{i=1}^{n} FC_i \right) \setminus C(O)$:

a) Se construye el conjunto compacto al que él pertenece y se coloca en la lista de compactos a procesar.

b) Se eliminan esos objetos del conjunto $K$.

Paso 5. Para cada compacto $C$ en la lista de compactos a procesar se construye su cubrimiento en conjuntos fuertemente compactos:

a) Hallar las componentes fuertemente conexas del grafo *máx-S* asociado a $C$, $G_c$.

b) Construir el grafo reducido de $G_c$.

c) Determinar la base $B$ del grafo reducido de $G_c$, que estará formada por todos los vértices del grafo reducido cuyo grado interior sea nulo.

d) Para cada elemento $b_l$ de $B$:

i. Construir el conjunto fuertemente compacto $F$ a partir de $b_l$, esto es, a $F$ pertenecerán todos los vértices de $G_c$ cuyos vértices correspondientes en el grafo reducido sean descendientes de $b_l$.

ii. Agregar $F$ al conjunto de conjuntos fuertemente compactos $\wp$.

Observación: Conservar la información de cuáles son los conjuntos fuertemente compactos que conforman un compacto resulta de mucha utilidad en el paso 4 dado que en virtud de la proposición 5 no sería necesario procesar a los objetos que pertenecen a conjuntos fuertemente compactos que conforman un compacto donde sólo se perdió un arco. Bastaría con colocar directamente a dicho compacto en la lista de compactos a procesar.

Note que en este algoritmo se aprovecha la propiedad que cumplen los conjuntos fuertemente compactos de ser un cubrimiento de conjuntos compactos.

Por eso, en lugar de hallar los conjuntos fuertemente compactos en todo el conjunto de objetos que modificaron sus arcos producto de la llegada del nuevo objeto se construye el grafo reducido asociado a cada conjunto compacto modificado por separado y se aplica el teorema demostrado anteriormente.

## 5. Análisis de la complejidad computacional

La complejidad computacional de este algoritmo es $O(n^2)$, pues en el paso 1 cada objeto se compara con todos los existentes. El paso 3 conforma el conjunto compacto (componente conexa) al que el nuevo objeto pertenece. Este paso tiene complejidad $O(e)$, pues se trabaja con listas de adyacencia [4]. Aquí $e$ es la cantidad de aristas del grafo. Aunque desde el punto de vista teórico $e$ es del orden $n^2$, en la práctica los grafos de máxima semejanza no son grafos completos y lo que ocurre con mucha frecuencia es que la cantidad de objetos más semejantes a uno dado es 1. Por eso, en nuestro caso, hemos estimado experimentalmente que $e=cn$, donde $c$ es la cantidad máxima de objetos más semejantes a uno dado. En el paso 4 se reconstruyen las componentes conexas de los grupos que perdieron arcos, lo cual, es $O(n+e)$. El paso 5 se encarga de construir las componentes fuertemente conexas para, a partir de ellas, conformar los conjuntos fuertemente compactos. Este paso es $O(n+e)$, por lo que no sube la complejidad total del algoritmo. La complejidad espacial (teniendo en cuenta lo anterior) es $O(n)$, pues para cada objeto sólo tendríamos que almacenar el valor de su máxima semejanza y la lista de los objetos más $\beta_0$-semejantes, cuyo cardinal nuevamente podría aproximarse por una constante.



## 6. Experimentación

La efectividad del algoritmo incremental propuesto ha sido evaluada usando una colección de 526 artículos de periódicos publicados en la sección de noticias internacionales del periódico español "El País" durante el mes de junio de 1999. En este tipo de problema, es claro el carácter dinámico de la colección de datos (en este caso, de documentos) y, por tanto, imprescindible la utilización de un algoritmo incremental que identifique los tópicos que son abordados en las noticias. Los documentos, denotados aquí por su código y el título del artículo, fueron representados en función de la frecuencia relativa de los términos que ocurren en dichos documentos y como medida de semejanza se utilizó la medida del coseno, ampliamente usada en este tipo de aplicaciones. Al aplicar el algoritmo se obtuvieron 297 grupos. A continuación mostramos algunos de los grupos formados utilizando como umbral de semejanza 0.25:

Grupo#1: {i101-Un observador ruso se incorpora a la reunión, i100-Los generales yugoslavos se resisten a firmar el plan para la retirada de sus tropas, i50-Calendario del conflicto, i181-Los serbios tienen 11 días para dejar Kosovo}.

Grupo#2: {i103-EE UU teme que se produzca un éxodo de población serbia, i24-Clinton dice que Europa asumirá el peso de la intervención y la reconstrucción de Kosovo, i50-Calendario del conflicto, i46-Clinton deja abiertas todas las opciones,

i181-Los serbios tienen 11 días para dejar Kosovo, i290-Francia se opone a que Europa pague la reconstrucción de los Balcanes}
Grupo#3: {i318-El anuncio oficial de la candidatura de Gore abre la campaña presidencial en Estados Unidos, i103-EE UU teme que se produzca un éxodo de población serbia, i50-Calendario del conflicto, i181-Los serbios tienen 11 días para dejar Kosovo}
Grupo#4: {i50-Calendario del conflicto, i258-El mando de la OTAN deja a los soldados rusos el control del aeropuerto de Pristina, i239-La OTAN confirma la retirada de 11.000 soldados serbios, i305-Los últimos soldados serbios dejan Pristina bajo control de la Kfor, i181-Los serbios tienen 11 días para dejar Kosovo}
Grupo#5: {i181-Los serbios tienen 11 días para dejar Kosovo, i184-Clinton se alegra del avance hacia los 'objetivos' aliados en Kosovo, i50-Calendario del conflicto}
Grupo#6: {i181-Los serbios tienen 11 días para dejar Kosovo, i4-Calendario del conflicto, i50-Calendario del conflicto}
Grupo#7: {i181-Los serbios tienen 11 días para dejar Kosovo, i141-Calendario del conflicto, i50-Calendario del conflicto}
Grupo#8: {i181-Los serbios tienen 11 días para dejar Kosovo, i214-Las tropas aliadas prevén entrar hoy o mañana en Kosovo, i239-La OTAN confirma la retirada de 11.000 soldados serbios, i50-Calendario del conflicto}
Grupo#9: {i181-Los serbios tienen 11 días para dejar Kosovo, i135-Los puntos de la discordia, i50-Calendario del conflicto}
Grupo#10: {i134-Silencio en Kumanovo tras el parón en las conversaciones, i50-Calendario del conflicto, i181-Los serbios tienen 11 días para dejar Kosovo}
Grupo#11: {i181-Los serbios tienen 11 días para dejar Kosovo, i500-España se integrará en el Grupo de países Amigos de Kosovo, i225-Calendarido del conflicto, i50-Calendario del conflicto}

Como puede notarse a estos 11 conjuntos fuertemente compactos pertenecen las noticias i181 e i50. Precisamente estas noticias constituyen holotipos de este conjunto, es decir, son los elementos a los cuales más se parecen el resto de los objetos en el conjunto dado. La base de este grafo es: {i500, i101, i290, i318, i305, i184, i141, i4, i214, i135, i134}. Note cómo cada elemento de la base genera un conjunto fuertemente compacto. Cada elemento de la base pudiera interpretarse como la noticia "primaria" más particular y cada conjunto fuertemente compacto generado por éste pudiera verse como el "desarrollo" de dicha noticia. Hemos observado también que el grado de similaridad aumenta monótonamente de la noticia "primaria" al holotipo.

# 7. Conclusiones

En este trabajo se presenta un algoritmo incremental de agrupamiento que crea un cubrimiento en conjuntos fuertemente compactos de la colección de objetos. La variante no incremental de este criterio de agrupamiento necesitaba almacenar la matriz de similaridad (de orden $n^2$) de los objetos de la colección lo que, sin dudas, constituye una gran limitación cuando se desea trabajar con colecciones dinámicas o

grandes de objetos. La variante incremental desarrollada no necesita almacenar esta matriz.

Otra ventaja del algoritmo propuesto es que permite encontrar grupos con formas arbitrarias. Además el agrupamiento obtenido por el algoritmo propuesto es único, es decir, no depende del orden de presentación de los objetos, a diferencia del *algoritmo de las estrellas* [5] que pueden producir diferentes agrupamientos al cambiar el orden de entrada de los objetos. Esta unicidad hay que entenderla de la siguiente manera: dados $m$ objetos la estructuración en conjuntos fuertemente compactos es única, independientemente del orden en que los $m$ objetos sean considerados. Es claro que en una población dinámica, la llegada de un nuevo objeto puede producir nuevas estructuraciones, pero de poblaciones diferentes. Sin embargo, en el caso de la estructuración en fuertemente compactos, una vez que ha sido considerada una población determinada, los objetos que la conforman pudieron haber arribado en cualquier orden produciendo la misma estructuración. El algoritmo propuesto no necesita fijar a priori el número de grupos a obtener, es decir, es aplicable a problemas de clasificación no supervisada libre. La complejidad computacional del algoritmo incremental presentado sigue siendo la misma que la de su variante no incremental, lo que es muy superior a la aplicación reiterada del algoritmo no incremental.

El algoritmo propuesto puede ser utilizado en todas las tareas que requieran del agrupamiento de objetos en fuertemente compactos y del procesamiento dinámico de la información, tales como, por ejemplo, navegación, filtrado, ruteo, detección y seguimiento de noticias en un flujo de documentos; el estudio de la morbilidad de una población; la estructuración de la información para estudios socio-económicos; entre otras aplicaciones. Elementos como la base y los conjuntos fuertemente compactos pueden resultar de mucha utilidad para el analista de información. Es bueno resaltar que este algoritmo puede utilizarse, además, en problemas donde se deseen agrupar objetos de cualquier naturaleza, descritos por rasgos cuantitativos y cualitativos mezclados, incluso con ausencia de información.

## Referencias


[1] Martínez-Trinidad, J. F., Ruiz-Shulcloper J., Lazo-Cortés, M. (2000). Structuralization of Universes, *Fuzzy Sets and Systems*, Vol. 112 (3), 485-500.

[2] Aho, A. V. , Hopcroft, J. E., Ullman, J. D. (1983). *Data Structures and Algorithms*, Addison-Wesley Publishing Company.

[3] Kakes, A., Casas, O. (1987). *Teoría de grafos*, Editorial Pueblo y Educación.

[4] Horowitz, E., Sahni, S. (1975). *Fundamentals of Data Structures*, Computer Science Press, Woodland Hills, California.

[5] Aslam, J., Pelekhov, K., Rus, D. (1998). Static and Dynamic Information Organization with Star Clusters, *in Proceedings of the 1998 Conference on Information Knowledge Management*, CIKM'98, Baltimore, MD.

# Desarrollo de un framework de clasificación de patrones

Ignacio Ramírez, Andrés Alcarraz, Alicia Fernández y André Fonseca de Oliveira
Instituto de Ingeniería Eléctrica - Facultad de Ingeniería
Universidad de la República Oriental del Uruguay
{ nacho, alcarraz, alicia, andre } @fing.edu.uy

24 de octubre de 2002

**Resumen**

El presente trabajo describe la arquitectura de un paquete desarrollado para sistemas de reconocimiento de patrones.

Se detalla el estudio de los elementos comunes y las variabilidades existenten en diversos sistemas de reconocimiento de patrones propuestos en la literatura, utilizados en la determinación de la arquitectura del paquete.

De este análisis se ha diseñado un paquete que define en forma fija la lógica, estructura e implementación de las partes identificadas como comunes a los sistemas de reconocimiento de patrones analizados, dejando incompletas las particularidades de estos.

De este modo el producto permite desarrollar aplicaciones que se adapten a cada caso, posibilitando al desarrollador concentrarse solamente en las variaciones (modelos, algoritmos, parámetros).

Al final del trabajo se describe la prueba del paquete desarrollado, utilizándolo como base para el desarrollo de un sistema de clasificación de patrones sencillo cuyo fin es el agrupamiento de maderas por colores para su uso en mueblería.

## 1. Introducción

Este trabajo surge como respuesta a la necesidad de centralizar los conceptos y conocimientos generales a toda aplicación de reconocimiento de patrones (RP) en un paquete de software reutilizable, surgida en el contexto académico del IIE[1] y en particular dentro del GTI[2], para complementar al resto de los paquetes que hasta el momento se han desarrollado en dicha institución.

### 1.1. Herramientas existentes

En el contexto actual no se tiene conocimiento de que existan herramientas con características similares a las deseadas para este producto, y que se adapten a las necesidades del IIE.

### 1.2. Objetivos

En resumen, las metas que este paquete pretende alcanzar son:

- Facilitar el desarrollo de aplicaciones de clasificación de patrones, suministrando una arquitectura prefabricada con sus conceptos y algoritmos más comunes.
- Definir un mecanismo único, pronto para utilizar, mediante el cual configurar y almacenar sistemas de clasificación de patrones.
- Proveer una capa de abstracción superior sobre los paquetes y conjuntos de algoritmos ya implementados en el IIE para el tratamiento de señales, en particular imágenes.

[1] Instituto de Ingeniería Eléctrica
[2] Grupo de Tratamiento de Imágenes

## 1.3. Vista general

En terminología de software, el paquete define un *framework* de clasificación de patrones, es decir, un marco de trabajo genérico cuyo fin es facilitar el desarrollo de aplicaciones concretas de clasificación de patrones, centralizando los conceptos y algoritmos comunes a la familia definida por este tipo de aplicaciones.

Además de la implementación de los algoritmos y conceptos generales inherentes a la Teoría de Reconocimiento de Patrones (en particular la Clasificación de Patrones), el paquete define un conjunto de facilidades para uniformizar el mecanismo de *persistencia*, codificación y transmisión de los datos dentro de un sistema de clasificación; y para consultar, definir y modificar los parámetros de los algoritmos involucrados.

Junto con las clases básicas del paquete se incluyen implementaciones de referencia para algunos algoritmos de clasificación populares como ser el Algoritmo de Bayes o el K-Means.

Finalmente, como resultado de las especificaciones del proyecto, se incluye especialmente un subframework para el desarrollo de sistemas basados en *redes neuronales artificiales*. Este ultimo es mucho mas que un agregado al paquete, constituyendo parte importante de el y pudiendo ser utilizado en otros contextos no relacionados con el reconocimiento de patrones, como por ejemplo compresión, control automático, etc.

## 1.4. Modo de uso

A diferencia de otro tipo de paquetes de software, el trabajo aquí presentado no provee una *aplicación funcional* lista para utilizar. Los usuarios de un *framework* son aquellos desarrolladores que deseen aprovechar los algoritmos, estructuras y conceptos de este paquete para crear aplicaciones concretas y funcionales diseñadas a medida para un problema particular.

El *framework* se compone de un conjunto de clases comunes a todo sistema, funcionales y que usualmente no deben ser especializadas, denominadas *frozen spots*, junto con una o mas clases abstractas que *deben* ser especializadas para cada caso particular, denominadas *hotspots*.

Quien desarrolle una aplicación basada en este paquete deberá entonces definir una o mas implementaciones concretas de los *hotspots* que ajusten a sus necesidades.

## 1.5. Ventajas adicionales

Además de facilitar y catalizar el desarrollo de nuevas aplicaciones, mediante su uso y propia evolución el framework sirve como contenedor de la base de conocimientos sobre la aplicación de la teoría del reconocimiento de patrones.

## 1.6. Estructura del documento

En la sección 2 se describe el proceso de análisis previo al diseño del paquete, que comprende al estudio de las principales características de los sistemas de reconocimiento estudiados. La sección 3 detalla el diseño del sistema. Luego en la sección 4 se describen algunos detalles de la implementación (lenguaje de programación, etc.).

En la segunda parte del documento, comenzando en la sección 5 , se habla específicamente del subframework de redes neuronales artificiales que forma parte del framework general.

En la sección 6 se describe una aplicación sencilla de clasificación, creada para probar los distintos aspectos del paquete desarrollado. Finalmente en la sección 7 se comentan los resultados obtenidos, las conclusiones en la 8 y finalmente algunos lineamientos sobre trabajos futuros en la sección 10.

# 2. Análisis

El principal objetivo de este trabajo fue lograr un producto flexible y reutilizable, que sirviera para desarrollar diversos tipos de sistemas.

Para esto, se pasó por una etapa de análisis en la cual se estudiaron distintos tipos de sistemas de reconocimiento de patrones. El foco del análisis fue identificar los aspectos comunes a todos ellos, y las particularidades que los distinguen entre sí.

Los tipos de sistemas estudiados se clasificaron según los paradigma (o modelos) teóricos sobre los cuales se basan, algoritmos que utilizan, modo de aprendizaje y operación, etc.

Dentro de los paradigmas vistos se incluyen los estadísticos, los estructurales, los KBS y los lógico-combinatorios, prestando especial atención a los primeros ya que constituyen la mayoría de los sistemas referidos en la literatura.

En cuanto a algoritmos se estudiaron sistemas paramétricos como el clasificador de Bayes o el K-Means, y no paramétricos, en especial la familia de las redes neuronales artificiales.

Por último se buscó, en cada familia, ejemplos de sistemas no entrenables, entrenables y adaptivos.

**Referencias**

Durante la etapa de análisis se estudiaron algoritmos como el de *Bayes*, tratado en libros como el de Duda y Hart [1] y el de Tou y Gonzales [2]. Algoritmos como el *K-Means*, *K-NEarest Neighbour*, así como los lógico combinatorios como el de *Testores Típicos* o el *Class* fueron tomados del libro de Shulcloper, Arenas y Trinidad [3] durante el curso dictado por el profesor Manuel Lazo Cortés del ICMF (Cuba). Las referencias utilizadas para los algoritmos basados en redes neuronales artificiales son descritas más adelante en la sección 5.

Además de los algoritmos específicos, el estudio general sobre sistemas de reconocimiento fue completado con trabajos como el de Cortijo [4], Crida [5], Pandya [6]. De este último fueron tomadas también las nociones básicas sobre sistemas de reconocimento estructurales. La información sobre los KBS (Knowledge Based Systems), no tenidos en cuenta originalmente, fue tomada a partir del sitio de Richard O. Duda [7] y recolectada de diversos lugares de Internet, recomendando al sitio anteriormente mencionado como punto de partida. Este sitio posee además información teórica y referencias a otros lugares relacionados con el RP.

## 2.1. Conceptos comunes (invariantes)

El primer aspecto común a todos los sistemas vistos es que admiten una representación de su funcionamiento en etapas bien distinguidas: adquisición (o sensado) de datos, extracción de patrones (o caracterización) y actuación (en particular, clasificación).

También surjen como invariantes los tipos de datos que fluyen de una etapa hacia la siguiente.

En este trabajo se focaliza en la etapa de clasificación, los datos de entrada a dicha etapa y los resultados obtenidos a la salida.

En este contexto se vió que, independientemente del significado que cada paradigma da a los patrones, su función y contenido pueden ser codificados en una misma estructura. Es decir, un patrón siempre podrá ser visto como un conjunto de características, sean símbolos o magnitudes. También, dependiendo del paradigma, los patrones obtenidos en un mismo proceso pueden ser de tamaño variable o fijo. Esto también puede unificarse, asumiendo el primer caso por defecto.

Durante el entrenamiento de un sistema, mas allá del paradigma o algoritmo utilizado, la información es suministrada en forma de ejemplos que representan prototipos de los tipos de objetos o entidades a reconocer. La función de un ejemplo como unidad de información de entrenamiento es algo que también se identificó como independiente del sistema en cuestión. Un modelo suficientemente útil al que se llegó es el de un ejemplo conformado por un patrón representando al prototipo mas un conjunto de información adicional.

Si el sistema es de clasificación, y el algoritmo de clasificación utilizado es supervisado, la información adicional toma la forma de la clase correcta a la que el prototipo pertenece según un experto (artificial o humano).

Asociado a un ejemplo también aparece en algunos sistemas el concepto de peso informacional del ejemplo, valor que pondera qué tanta información aporta cada ejemplo al entrenamiento, o bien con qué certeza el resultado de referencia es obtenido, para sistemas supervisados.

Tanto la información adicional como el peso informacional no aparecen en todos los sistemas vistos, pero con su inclusión no se pierde generalidad, siendo ignorada esta información por aquellos sistemas que no la utilicen.

La salida de un sistema de clasificación puede ser expresada en una forma lo suficientemente general como para abarcar los sistemas vistos. Concretamente, si un sistema de clasificación conoce N clases, a lo sumo podrá inferir el grado de pertenencia del patrón clasificado a cada una de ellas. Por ejemplo, si el sistema utiliza lógica borrosa, los grados de pertenencia pueden ser continuos en el rango [0,1], mientras que en sistemas basados en teoría de conjuntos clásica, los valores admisibles son 0 y 1.

Las clases también son modeladas en forma única, asignando "etiquetas" numéricas a cada una de ellas. La posible descripción de cada clase se deja en manos de la aplicación concreta a implementar.

Por último, la principal invariante es la que refiere a la *interfaz* de la etapa de clasificación, es decir, su modelo de "caja negra". En este aspecto, todos los clasificadores vistos cumplen con una operación de *clasificación* que recibe un patrón y produce un resultado. También se puede definir una operación de *entrenamiento*

La operación de entrenamiento pudo ser unificada para todos los sistemas vistos al dividirse en dos suboperaciones consecutivas: *recolección de la información* suministrada por los ejemplos, y *adaptación del sistema* en base a la información almacenada. A estas operaciones se las identificó como *recolección* y *adaptación*.

Estas operaciones pueden ser definidas sea el sistema entrenable, adaptivo o incluso no entrenable. Por ejemplo, para este último caso, ambas operaciones simplemente se ignoran.

## 2.2. Variabilidades

La principal variabilidad del sistema reside en el *algoritmo utilizado para clasificar*, la información necesaria para ello y el algoritmo de entrenamiento para sintetizar esa información.

# 3. Diseño

El diseño parte de los conceptos centrales identificados en la mayoría de los sistemas de clasificación: *clasificador, patrón, cluster, resultado, clase, ejemplo.* El primer criterio adoptado fue la representación de cada uno de estos conceptos centrales por clases.

## 3.1. Clases principales

### 3.1.1. Patrones

La clase `Pattern` define el concepto general de patrón como un vector real de características de tamaño variable. Esta representación se acomoda especialmente a los sistemas de clasificación estadísticos, que admiten una representación *vectorial* en un *espacio de características*. En contraparte no es del todo adecuada para sistemas cuyas características son *etiquetas* o *símbolos* como podría ser el caso de clasificadores *sintácticos*[3], *lógico combinatorios* [3] o *KBS* [9], cita tomada del sitio web de Richard O. Duda [7], mas que nada en lo que respecta a eficiencia en el almacenamiento. Esta decisión se fundamenta en el hecho de que la mayoría de los sistemas de clasificación se enmarcan en el paradigma estadístico y en la necesidad de evitar sobrecargas excesivas en tiempos de ejecución para clases de uso intensivo como *Pattern*. El resultado es una clase *Pattern* de poco peso y sobre todo concreta, evitando así la sobrecarga debida al mecanismo de *polimorfismo dinámico* de lenguajes como C++.

### 3.1.2. Ejemplos de entrenamiento

`Example` encapsula un ejemplo de entrenamiento formado por un *Pattern* proveniente de un prototipo de entrenamiento y eventualmente un *resultado de clasificación de referencia* provisto por un *experto* para el uso en algoritmos de clasificación supervisados.

[3] Ver Chomsky [8], cita tomada del libro de Pandya [6]

Esta implementación define además un *peso informacional* para cada ejemplo, que puede tomar valores entre 0 (no aporta información) y 1 (máxima información).

### 3.1.3. Resultados

`Result` representa un resultado genérico de clasificación. En el diseño realizado, éste se expresa como una lista de pares *clase-grado de pertenencia*, con un par para cada clase definida.

El clasificador que produce un resultado, lo hace creando una instancia de esta clase y asignando al patrón clasificado un grado de pertenencia a cada clase conocida.

La forma en que esta clase fue definida permite abarcar resultados de clasificación de diversos tipos de sistemas desde Bayes hasta sistemas de *lógica borrosa* como el [10].

Las clases son representadas por *identificadores numéricos*, reservando el "ID" de valor 0 para la *clase de rechazo*, utilizada por muchos algoritmos como aquella a la que se asignan los patrones de no haber clasificación satisfactoria dentro de las otras clases existentes.

## 3.2. Classifier

El principal *hotspot* es la clase abstracta `Classifier` y la interfaz que define sus operaciones.

La principal operación a definir es el propio algoritmo de clasificación, dada por método `classify()`.

Además, si el clasificador a diseñar requiere *entrenamiento* deben implementarse o bien los métodos `train()` y `batchTrain()` o bien el par de métodos `adapt()` y `learn()`.

`train()` y `batchTrain()` se utilizan para *entrenar* o *reentrenar* a un sistema de clasificación a partir de uno (`train()`) o varios (`batchTrain()`) ejemplos de entrenamiento. Este es el enfoque mas clásico de entrenamiento de sistemas de clasificación.

Por otro lado se define también el par `learn()`-`adapt()`, previsto para algoritmos no sólo *entrenables* sino *adaptivos*, pudiéndose utilizar en cualquiera de las dos situaciones. La idea es que `learn()` colecta información de aprendizaje, de a un ejemplo por vez. `adapt()` sintetiza toda la información recolectada por `learn()` y efectúa la adaptación del sistema según ella.

### 3.2.1. train() y batchTrain() vs. learn() y adapt()

En este contexto se considera como *entrenables* a aquellos sistemas que requieran un conjunto de ejemplos (o prototipos) de entrenamiento para definir sus parámetros, sin implicar esto que dichos sistemas sean capaces de adaptarse a cambios en la naturaleza del problema durante su utilización. Llegado este último caso, estos sistemas pueden ser a lo sumo *reentrenados* con un nuevo conjunto de datos actualizado.

Por su parte, los sistemas *adaptivos* son aquellos cuya estructura admite una modificación *incremental* de su estado interno para adaptarse al problema, incluso en tiempo real como podría ser en una línea de producción.

`learn()` y `adapt()` permiten este comportamiento y dan lugar a optimizaciones del tipo que puede verse en algoritmos tipo *batch*, permitiendo a la vez su uso en tiempo real. Es posible también definir el proceso de entrenamiento en función de estos métodos, aunque en algunos casos sea poco práctico o conceptualmente forzado. De hecho, `train()` y `batchTrain()` poseen implementaciones por defecto basadas en `learn()` y `adapt()` de forma de reutilizar código al máximo.

### 3.2.2. Interfaz de persistencia y configuración

`Classifier` implementa por defecto las interfaces `Encodable` y `Configurable` para poder ser propiamente almacenado y configurado. Ambas interfaces serán descritas más adelante en este documento.

### 3.2.3. Matriz de confusión

Como mecanismo estándar de evaluación de desempeño para algoritmos supervisados se provee el método `getConfusionMatrix()` que permite, en base a un conjunto de ejemplos de prueba, analizar el

desempeño de un clasificador dado en base a la comparación de los resultados producidos por él y los resultados esperados, almacenados en los ejemplos de prueba.

La matriz de confusión se confecciona de la siguiente manera:

1. Se define el tamaño de la matriz como $N \times N$, siendo $N$ la cantidad de clases definidas, y se asocia un índice a cada clase existente, secuencialmente de 1 a $N$.

2. Para cada ejemplo de prueba:

   a) se determina $i$ como el índice de la clase *correcta* contenida en el ejemplo.

   b) se determina $j$ como el índice de la clase obtenida como resultado de la clasificación del ejemplo.

   c) se incrementa en 1 la casilla $i, j$.

De esta forma, todos los elementos de una fila corresponden a una misma clase *correcta*, mientras que los de una columna son los que el clasificador ubicó en la misma clase.

En la representación utilizada en las tablas, la matriz es de $N + 1 \times N + 1$. La última fila de cada tabla contiene el *índice de confiabilidad* del clasificador, que indica que tan probable es que "cuando el clasificador decida la clase dada, el resultado verdadero sea efectivamente ese". El cálculo para este índice es:

$$confiabilidad_j = \frac{c_{jj}}{\sum_{i=1}^{N} c_{ij}}$$

La última columna, por su parte, indica el *índice de confusión* e indica "que tan bien clasifica el sistema a los elementos de la clase". Su cálculo viene dado por la expresión

$$confusion_j = \frac{c_{ii}}{\sum_{j=1}^{N} c_{ij}}$$

Por último, el elemento $[N + 1, N + 1]$ contiene un índice de desempeño global, que se calcula como

$$perf = \frac{\sum_{i=1}^{N} c_{ii}}{\sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij}}$$

Naturalmente, otros mecanismos de evaluación del desempeño pueden ser definidos por los algoritmos particulares que implementen la interfaz de `Classifier`.

## 3.3. Configuración

Hay otras funciones que un sistema de clasificación real debe poseer además de clasificar y ser entrenado. En particular los algoritmos de clasificación, sobre todo para la etapa de entrenamiento, suelen definir una serie de parámetros que los gobiernan (por ejemplo el *factor de aprendizaje* presente en muchas redes neuronales adaptivas).

Para estas dos operaciones esenciales se definen las clases `Configurable` y `Configuration` y las interfaces `Encodable` e `IOManager`. Las primeras dos definen un mecanismo común y una estructura para el almacenamiento, consulta y modificación de parámetros definidos por un clasificador, sus algoritmos y subalgoritmos. `Encodable` e `IOManager` por su parte definen las interfaces que se deben implementar para que un sistema de clasificación y todas las clases que definen su estado y comportamiento puedan exportar su información interna de modo que pueda ser almacenada. La interfaz `Encodable` define cómo los distintos objetos deben exportar su información, en cambio `IOManager` define la interfaz que se debe implementar para definir un formato de exportación de la información, por ejemplo a un archivo XML.

El punto saliente del mecanismo de configuración es que permite consultar en tiempo de ejecución todos los parámetros existentes para un sistema de clasificación sin conocerse las clases, algoritmos y/o *estrategias* involucrados. Para ello se definió un esquema similar a *DNS* en el cual cada algoritmo tiene un *dominio* asociado y sus nodos son parámetros o nombres de *subdominios* definidos por *subalgoritmos*.

Para clarificar este concepto, supóngase un sistema de clasificación que utiliza el algoritmo *LMS* (Least Medium Square) para su adaptación. El *LMS* a su vez define una serie de parámetros propios como el *factor de aprendizaje* o *factor de momentum*. Por último el sistema define sus propios parámetros, por ejemplo, *error mínimo* o una opción para *mezclar los patrones de entrenamiento*. Al consultarse la lista de parámetros configurables para el sistema se obtendría la siguiente lista:

- classifier.minerr
- classifier.mezclar
- classifier.lms.lr
- classifier.lms.momentum

Los "dominios" definidos son en este caso "classifier" para el sistema de clasificación general y "lms" para el algoritmo lms. Notar que si el algoritmo de adaptación fuera otro (por ejemplo Kalman), la consulta devolvería la lista de parámetros correspondiente a ese caso.

## 3.4. Persistencia

Además, es esencial poder almacenar el estado de un sistema de clasificación entrenado y listo para ser utilizado.

Las interfaces `Encodable` e `IOManager` definen las operaciones que se deben implementar para que un sistema de clasificación y todas las clases que definen su estado y comportamiento puedan exportar su información interna de modo que pueda ser almacenada.

La interfaz `Encodable` define cómo los distintos objetos deben exportar su información.

`IOManager` define la interfaz que se debe implementar para definir un formato de exportación de la información, por ejemplo a un archivo XML.

`Encodable` utiliza el estándar *DOM* [11] como base para la codificación del estado y comportamiento de un sistema dado. Implementando esta interfaz es posible traducir esta información a diversos formatos, por defecto *XML* [12]. Tanto la codificación a *DOM* como el almacenamiento en formato *XML* dependen de la utilización de la biblioteca *Xerces* de uso público desarrollada y mantenida por el proyecto *Apache* [13].

Como nota final cabe notar que la complejidad de este diseño se debe a la necesidad, considerada importante, de independizar la forma en que se codifican internamente los datos y la forma en que se guardan y/o transmiten. Como aplicación directa de esto puede decirse que es posible, sin tocar ninguna línea del código existente, almacenar todo un sistema de clasificación en un formato totalmente nuevo (binario, ASCII, base64, etc).

## 3.5. Mensajes del sistema

La última de las utilidades disponibles en el framework es la clase *Logger* que permite controlar la emisión de mensajes de error, advertencias, trazado o depuración durante la ejecución según un nivel configurable en tiempo de ejecución.

## 3.6. Implementaciones listas para usar

Para cada *hotspot* se definieron algunas implementaciones de referencia para probar el sistema y darle cierta funcionalidad por defecto. Tal es el caso de `Classifier`, que actualmente dispone de tres implementaciones: Bayes, K-Means y NeuralNetClassifier, este último haciendo de *adaptador* para definir clasificadores basados en las redes neuronales artificiales tal como son provistas por el *subframework de redes neuronales* descrito más adelante.

# 4. Implementación

La implementación fue realizada en C++ que es un lenguaje orientado a objetos y a la vez permite lograr un buen desempeño en tiempo de ejecución, a costo de ser un lenguaje complejo y difícil de dominar y utilizar eficientemente.

ClasiLiptus no es un paquete autocontenido ya que depende fuertemente de otros dos paquetes de libre distribución: el *Xerces* [12] para la codificación y almancenamiento de la información inherente a los sistemas de clasificación, y la *Matrix Template Library* [14] para la representación de vectores y matrices así como para las operaciones algebraicas mas comunes (producto, producto interno, norma, matriz inversa, determinante, etc.).

# 5. Subframework de redes neuronales

Los algoritmos basados en *redes neuronales artificiales* son una importante rama de la familia de algoritmos de clasificación y reconocimiento de patrones y abarcan una amplia gama de variedades pensadas para resolver ciertos problemas complejos o difíciles de resolver mediante otros métodos.

Debido a esto es que el framework de clasificación dispone de un *subframework* para la implementación de redes neuronales artificiales. Este *subframework* posee además la virtud de poder ser utilizado en forma independiente de la parte de clasificación de patrones, por ejemplo en aplicaciones que no tienen que ver con el reconocimiento de patrones como la codificación de datos[4]

## 5.1. Análisis

Debido a la importancia atribuída a esta rama de algoritmos de reconocimiento de patrones es que se detalla aquí el análisis realizado previo al diseño de este módulo.

Al igual que para los clasificadores en general, para identificar las partes comunes y las variabilidades que caracterizan a la familia de redes neuronales artificiales (*RNA* de aquí en más), se analizó la estructura y funcionamiento de una variedad de tipos de ellas.

### Referencias

Algoritmos como *Perceptron*, *Adaline*, *Backpropagation*, *Kohonen SOM*, *Hopfield*, *BAM*, *LAM*, *Counterpropagation* fueron tomados principalmente del libro de Freeman [15]. En particular, una gran cantidad de variedades de *Backpropagation* son descritas en "The Backpropagation Family Album" [16]. También se consultó el libro de Pandya y Macy [6], que describe entre otros al *LVQ* (Learning Vector Quantization). Para el estudio del *ANFIS* se consultó el trabajo de Jang [10]. Por último, fue tenida en cuenta la implementación del SNNS (Stuttgart Neural Network Simulator) [17] como referencia tanto a nivel conceptual como técnico durante el diseño y desarrollo de este módulo.

## 5.2. Invariantes

Las invariantes fueron tomadas en cuenta a nivel de estructura, para modelar los datos, y a nivel de interfaz, para modelar la funcionalidad.

En la primera categoría se encuentra la estructura general de una red neuronal artificial. En todos los modelos vistos, incluso en algunas definiciones "genéricas" dadas por autores como Hetch [18] (cita tomada de [6]), las redes neuronales artificiales son modeladas a nivel estructural como un *grafo dirigido* cuyos *nodos* son las unidades de procesamiento (PE) o *neuronas* y cuyos enlaces son *unidireccionales*. Cada enlace recibe un *peso* que pondera el grado de influencia de la salida unidad de origen sobre la entrada la unidad destino.

De este modo se definieron como invariantes a los elementos básicos de esta estructura, es decir, *enlace* (o conexión), y *unidad de procesamiento* (o neurona artificial), desde el punto de vista de la *conectividad*,

[4] Un ejemplo son las *LVQ - Linear Vector Quantification*, brevemente descritas en el libro de Pandya [6]

siendo el modelo computacional y operación de cada unidad una variabilidad muy importante a tener en cuenta. De este mismo análisis se definió como invariante la *estructura* de una red neuronal para cualquier tipo, y la interfaz utilizada para construir y modificar dicha estructura desde el exterior.

La interfaz principal, es decir, entrenamiento, adaptación y aplicación de la RNA, también fue fijada como invariante. También fueron fijadas las interfaces de entrenamiento, adaptación y aplicación a nivel de unidad de procesamiento. Cabe remarcar que el *fijar* una interfaz no implica *fijar* una implementación, sino restringir. En especial, las RNA como fueron modeladas reciben entradas y producen salidas de tipo *vectorial*, asumiendo en todos los casos que estas son *señales*, no *símbolos* (etiquetas, etc.) de acuerdo con la definición general vista en los textos consultados.

## 5.3. Variabilidades

Se distinguieron como variabilidades principales a los mecanismos de *inicialización* (o *calibración inicial* en este contexto), *entrenamiento/adaptación* y *propagación*. A cada uno de estos se los independizó de los otros, teniendo en cuenta que muchos de los modelos vistos comparten algunos de ellos y varían en otros.

Por dar un ejemplo, tanto una red *backpropagation*, como el *ANFIS* comparten el mismo algoritmo de *propagación* (algoritmo *feedforward*). El propio algoritmo de entrenamiento del *ANFIS* puede también verse como una variante del *Backpropagation*, pero la forma de inicializarse es completamente distinta entre ellos. *Kohonen* también puede ser propagado mediante el algoritmo *feedforward* y su entrenamiento es totalmente distinto, y sin embargo el algoritmo *Nguyen-Widrow* de *inicialización* puede ser usado tanto en *Backpropagation* como en *Kohonen*.

Además de distinguir las operaciones principales como diferentes variabilidades, también se separó cada una de ellas en *algoritmo general/global* y *algoritmo local*. El primero se refiere a aquel que controla el proceso a nivel de la red en su globalidad (por ejemplo presentación de datos de entrada, propagación, comparación con salida de referencia, orden de adaptación, etc.) mientras que el segundo es efectuado localmente por cada *unidad de procesamiento*. Por ejemplo, la propagación global para una red feedforward sería "presentar entrada a la capa de entrada, propagar hacia las capas superiores, obtener salida en capa de salida". Localmente, cada unidad de procesamiento procesa la operación *propagar* como "recibir entrada, calcular valor de función de red, calcular activación, calcular salida, publicar salida".

Para las redes cuyos algoritmos son supervisados se identificó como variabilidad a la *función de costo* utilizada para evaluar el desempeño de la red y tomar acciones correctivas (por ejemplo, en , *Backpropagation*). Si bien el ejemplo más común es el *error cuadrático medio*, la decisión tomada fue que no se debería fijar esto ya que quizá alguien desee utilizar algún otro tipo de función.

Por último, se identificaron en forma separada cada una de las funciones que intervienen en la propagación local de una unidad de procesamiento, es decir, las funciones de *red (o membrana)*, *activación* y *salida*. La primera es una función de $\mathbf{R}^{2N} \rightarrow \mathbf{R}$ para una unidad con $N$ entradas con sus $N$ señales de entrada y $N$ pesos asociados. Las otras dos son modeladas como funciones $\mathbf{R} \rightarrow \mathbf{R}$, suponiendo que la *activación* puede no ser algebraica (por ejemplo una ecuación de recurrencia o función algorítmica, etc.), pero no forzando este concepto por lo que queda libre al desarrollador.

## 5.4. Diseño

El principal énfasis en esta etapa fue puesto en facilitar y fomentar el desarrollo *ortogonal* de los distintos algoritmos que gobiernan el funcionamiento de una red neuronal artificial. Por *desarrollo ortogonal* se identifica a aquel que permite variar en forma independiente una parte del funcionamiento del sistema sin afectar al resto en lo que refiere a implementación e interfaz.

En particular, para una red neuronal artificial, fueron identificadas una serie de operaciones fundamentales comunes a todo sistema de este tipo en significado y cuyas distintas implementaciones caracterizan a cada tipo de red. Estas operaciones son *inicialización*, *calibración*, *propagación*, *entrenamiento* y *adaptación*.

En el diseño esto se traduce en una clase `NeuralNet` concreta cuya funcionalidad es extendida mediante *manejadores* (denominados *managers* en el framework) para cada tipo de operación. El *manejador de*

*adaptación* se denomina `AdaptationManager`, el de *propagación* `PropagationManager`, y el de *calibración* `CalibrationManager`.

El diseño obtenido lleva la ortogonalidad un poco más al extremo, permitiendo la variación independiente de los aspectos *globales* y *locales* de un algoritmo. Todos estos conceptos serán detallados en las siguientes subsecciones.

### 5.4.1. Clases Principales

La principal clase es `NeuralNet`. Esta no es una clase heredable sino que las distintas implementaciones de redes neuronales deben definir *managers* adecuados para lograr el efecto deseado. Esto hace que `NeuralNet` no sea un *hotspot*. En todo caso, `NeuralNet` puede ser vista como una *fachada* para todo el *subframework*, ya que define el comportamiento general de una red neuronal artificial a través de su interfaz.

Además, `NeuralNet` almacena la estructura y estado de la red en la forma de *unidades de procesamiento* representadas por la clase `NetNeuron` y conexiones entre unidades, modeladas por la clase `NetConnection`.

La clase `NetNeuron` es la segunda más importante, además de ser uno de los *hotspots* del subframework. Esta clase define el comportamiento básico de una unidad de procesamiento y provee una interfaz para la *propagación*, *calibración* y *adaptación* local de la unidad. Esta no es una clase *abstracta* ya que define un comportamiento concreto básico que consta de un algoritmo general de *propagación* de las señales, otro de calibración básica y una implementacion *no operante* para la *adaptación*.

En lo que respecta al modelo *computacional* de cada unidad, se define un mecanismo mediante el cual cada unidad puede ser configurada con su propia *función de red*, *función de activación* y/o *función de salida* en forma independiente mediante instancias de las clases `NetFunction` y `UnitFunction`.

`NetFunction` define la forma general para las *funciones de red*, es decir, funciones de $\mathbf{R}^N \rightarrow \mathbf{R}$, siendo $N$ la cantidad de entradas de una unidad particular. Estas funciones toman además un conjunto de $N$ parámetros correspondientes a los $N$ pesos de las conexiones de entrada.

`UnitFunction` defune funciones de $\mathbf{R} \rightarrow \mathbf{R}$ y es utilizada para modelar las distintas funciones de *activación* y *salida* posibles de las unidades.

`NetConnection` no es mas que un enlace entre dos unidades de procesamiento con un *peso* asociado que indica la *intensidad* con que la salida de la unidad de origen afecta a la entrada de la de destino. La única particularidad de esta implementación es que admite un *learning rate* propio y la posibilidad de definir el *peso* como parámetro *interno* a la instancia o *externo*.

### 5.4.2. Hotspots

Hasta ahora no se ha aclarado el significado de las operaciones anteriormente nombradas: *inicialización*, *calibración*, *propagación*, *entrenamiento* y *adaptación*. Esto se debe a su correspondencia directa con los hotspots del subframework que serán detallados en este apartado. La posible excepción la hace la *inicialización*, que refiere a una cuestión *estructural* del sistema y que cada algoritmo particular define de acuerdo a su estructura. La necesidad de una operación como la *inicialización* depende de la posibilidad de tener diversas estructuras de unidades y conexiones coexistiendo con diversos algoritmos para lograr el funcionamiento de un sistema complejo como lo es la red neuronal artificial.

La primera operación fundamental es la *propagación de las señales* dentro de una red. Una red neuronal artificial recibe siempre una entrada en forma de vector numérico y la propaga dentro de su estructura de alguna forma hasta obtener una salida *estable*. Para distintos tipos de red la forma en que esto se lleva a cabo varía mucho. Por ejemplo, una red tipo *feed forward* toma la entrada en su *capa de entrada* y la propaga según un *orden topológico* hacia su capa de salida. Por otro lado, una red *recurrente* como *Hopfield* requiere de varias iteraciones para producir una salida desde que se carga una entrada en sus unidades.

Continuando con el ejemplo de *Hopfield* existen variantes *síncronas* y *asíncronas* para el mecanismo de propagación de dichas redes recurrentes. La variante *síncrona* implica el disparo simultáneo de todas las unidades de la red en base a un mismo estado previo, mientras que en el modelo *asíncrono* las unidades disparan en forma aleatoria una por vez y sin importar el orden del sistema.

Todos estos comportamientos pueden ser definidos implementando especializaciones de `PropagationManager`. El framework, para facilitar la tarea de estas implementaciones en lo que refiere al caso de *modelo de sincronismo* anteriormente mencionado, define un modelo de propagación de *unidades de procesamiento* en dos etapas: *propagación* y *disparo*.

La *propagación*, a nivel de *unidad de procesamiento* implica calcular una *salida potencial* de la unidad en base a su estado interno y entrada actuales. Esta salida no es *disparada* mientras no se le indique a la unidad mediante un método especial `fire()`.

De esta manera las distintas implementaciones de este *manager* pueden administrar el orden de propagación y disparo de las unidades para lograr el efecto deseado en lo que refiere a sincronía.

La *calibración* se refiere a la obtención de un estado propicio para lograr una rápida y efectiva adaptación durante el entrenamiento de la red, en base o no a un conjunto de vectores de calibración especiales. En general esto se traducirá a descubrir los *rangos de variación* de las distintas entradas y ajustar los parámetros iniciales del sistema para situarlo en una posición que minimice la probabilidad de caer en mínimos locales o generar *unidades muertas* durante el entrenamiento.

Por último el *entrenamiento* y *la adaptación* son ambas operaciones administradas a nivel global por `AdaptationManager`, dejando los detalles de nivel local a cada unidad para las distintas implementaciones de `NetNeuron`. Esto permite *ortogonalizar* el desarrollo de algoritmos generales como *backpropagation* (lógica general del algoritmo, secuencia de adaptación, etc.) y sus variantes a nivel global, de sus variaciones a nivel local (es decir, en la adaptación de cada unidad). Una implementación de este esquema está presente en el framework y es justamente la del algoritmo *backpropagation* por `BpnAdaptationManager` con dos variantes `BpnNeuron` y `QpNeuron` para *backpropagation con momentum* y *quickpropagation* (ver [16] por descripciones de estas variantes).

Mas aún, no solo se aumenta el reuso de código con este esquema sino que es posible definir redes *híbridas* que combinen unidades de distintas variantes de un mismo algoritmo, comandadas a su vez por un mismo algoritmo general, lo que agrega flexibilidad al sistema y lo hace atractivo para el prototipado de nuevos tipos de redes.

Por último, para el caso particular de redes con entrenamiento *supervisado* se define una subclase especial de `AdaptationManager`, `SupervisedAdaptationManager`, que a su vez utiliza una instancia de `ObjectiveFunction` para evaluar el *error* en la salida de la red respecto a una referencia dada, al propagar una entrada asociada a esa referencia. Como ejemplo, `BpnAdaptationManager` es una subclase de `SupervisedAdaptationManager` que utiliza la función `MseObjectiveFunction`[5] para evaluar el error en la salida de la red.

### 5.4.3. Funcionalidad adicional

Aparte de la funcionalidad adicional "heredada" del framework general, el subframework de redes neuronales artificiales define un mecanismo propio para el almacenamiento y carga de sus datos. En particular, fue definido un "filtro de exportación/importación" que puede ser utilizado para transferir redes neuronales tipo *backpropagation* desde y hacia el simulador de redes neuronales artificiales de la Universidad de Stuttgart, el SNNS [17].

## 5.5. Implementación

Los detalles de la implementación de este subframework se remiten a los criterios utilizados a nivel general en el framework. Sin embargo, cabe destacar el mayor énfasis puesto en el *desempeño* del sistema en tiempo de ejecución, dado que las redes neuronales artificiales suelen requerir un gran número de cálculos para su entrenamiento. El uso excesivo del *polimorfismo dinámico*, que suele ser tentador por lo elegante que resultan los diseños resultantes, fue evitado en la medida de lo posible debido a la degradación del desempeño que esto suele tener como consecuencia.

[5] MSE significa *Mean Square Error*.

### 5.6. Implementaciones disponibles para usar del subframework de redes neuronales

Se proveen implementaciones de administradores de entenamiento y neuronas para los algoritmos Backpropagation (con momentum), Quickpropagation, Kohonen SOM y ANFIS. Como función objetivo está disponible el *error cuadrático medio*.

Para la propagación se dispone de un algoritmo *feed forward* que descubre la topología automáticamente. Este algoritmo es adecuado para todas las redes implementadas nombradas anteriormente.

Para la *calibración* se implementó un algoritmo inspirado en el método de inicialización de pesos de Nguyen-Widrow [19], que minimiza la probabilidad de que se produzacan "neuronas muertas"durante el entrenamiento.

Por último se definieron funciones de *red* tipo *sumatoria*, *productoria*, *normalización de entradas*, *distancia*, y funciones de activación/salida tipo *sigmoide*, *inversa normalizada*, *modulo*, *gaussiana*, *campana (bell)* y *lineal*.

## 6. Aplicación de prueba

En esta parte del trabajo se plantea el diseño de una aplicación que ponga a prueba al framework de clasificación frente a un problema real. El problema planteado es un problema sencillo, ya que el objetivo aquí no se centra en las visiscitudes de la doctrina de reconocimiento de patrones en sí, sino en probar la funcionalidad del framework como herramienta de diseño.

El problema en cuestión es la *clasificación de maderas de eucalyptus* en base a un criterio de *similitud*. La idea es obtener un conjunto de unas pocas clases de madera de eucalyptus de forma de que en cada grupo las maderas sean lo suficientemente similares entre ellas en aspecto.

La necesidad de una clasificación de este tipo surge de la intención de introducir la madera de eucalyptus al mercado de la carpintería, especialmente en aquellas áreas en las cuales importa la uniformidad de color y aspecto de la madera utilizada en un mismo producto.

Para esta aplicación los requerimientos previstos implican la adquisición de las imágenes de las maderas en tiempo real mediante una cámara, y la comunicación de los resultados de clasificación a un sistema SCADA[6] para un eventual control de un dispositivo de accionamiento que permita dirigir los listones de madera hacia uno de los hipotéticos canastos.

### 6.1. Diseño del sistema de clasificación

Dado que el problema planteado se basa en la percepción subjetiva del ojo humano del color de las maderas, las características elegidas para conformar los patrones fueron basadas en las componentes HSV de las imágenes obtenidas de la superficie de las maderas.

La primera decisión importante fue entonces utilizar como patrón el vector *HSV promedio* de la imagen. Esta decisión surge de las propiedades del modelo de color *HSV* [2], que pretende modelar los parámetros más relevantes para el ojo humano en la detección del color[7]. La elección del *promedio* HSV también deriva del hecho de que el ojo humano *promedia* la imagen cuando la resolución natural no le alcanza para percibir los detalles, en este caso, la veta de la madera a partir de cierta distancia.

Ya que no se trata aquí de ubicar a las maderas en un conjunto de clases predefinidas (y estandarizadas) sino que la agrupación obtenida puede ser arbitraria, siempre y cuando se cumpla el requisito básico de *similitud* entre maderas de la misma clase, se optó por un algoritmo de clasificación *no supervisado*.

El algoritmo en cuestión es el *K-Means*, debido a su sencilla implementación y aparente idoneidad para el problema a resolver. El *K-Means* es un algoritmo de *agrupación no supervisado* que toma como único parámetro a la cantidad de grupos a crear: $K$. El valor óptimo de $K$ es el mínimo necesario para lograr el cumplimiento del requisito de similitud entre maderas del mismo grupo, debido a que un mayor número de

[6] Supervisory Control And Data Acquisition.

[7] Tener en cuenta que la relación de similitud buscada dentro de los grupos formados es respecto al ojo humano.

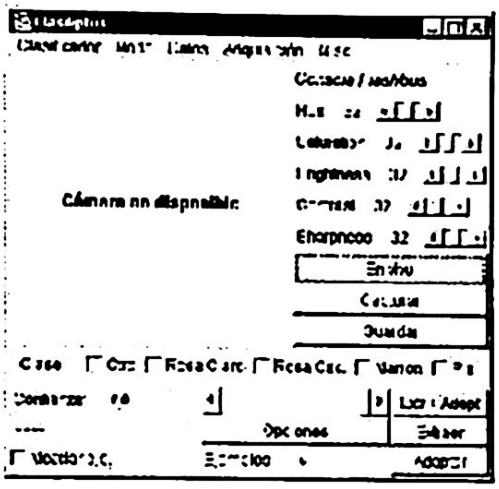


Figura 1: Interfaz de usuario

grupos aumenta los costos de administración del stock (derivado de un mayor número de compartimientos, más categorías y menor disponibilidad de madera en cada categoría).

## 6.2. Implementación

La aplicación de prueba fue implementada con el *framework* como base para toda la parte de clasificación, mas módulos para adquisición y preprocesamiento (no abarcados por el framework en su estado actual) y otros tantos para la comunicación con el SCADA utilizado para "publicar" los resultados.

El resultado final permite la creación de varios tipos de sistemas de clasificación (todos los implementados como parte integral del framework), a saber, *Backpropagation*, *Bayes* y *K-Means*. De esta forma pudo ser utilizada como plataforma de prueba de los algoritmos supervisados implementados como parte del framework, utilizando K-Means para definir las clases y producir en base a ellas un conjunto de entrenamiento supervisado.

En las figuras 1 y 2 puede verse una muestra del aspecto que tiene la interfaz gráfica diseñada. Esta interfaz permite crear, configurar, calibrar, entrenar y almacenar un clasificador dado. También permite crear, cargar y almacenar ejemplos de entrenamiento, seleccionar el origen de las imágenes (cámara archivo), seleccionar el método de calibración de la cámara, ajustar los parámetros de adquisición (brillo, contraste, saturación, etc.).

# 7. Resultados

## 7.1. El Framework como herramienta de desarrollo

Algo que vale la pena destacar es la naturalidad con que los conceptos y las interfaces definidas fueron utilizadas a la hora de desarrollar aplicaciones concretas. Cabe notar que bajo el mismo esquema implementaron algoritmos supervisados y no supervisados, entrenados "por lotes" (*batch*) o "en línea" (*online*), paramétricos o no. Por supuesto, esta naturalidad no fue conseguida sin antes pasar por más de un ciclo de desarrollo, desde el planteo del trabajo. La posible excepción, por no haber sido probada aun, es la implementación de algoritmos no estadísticos sino *estructurales* o KBS, por nombrar algunos paradigmas.

También puede nombrarse el gran aporte que el sistema de persistencia basado en XML hizo a aplicación. El almacenamiento de clasificadores y ejemplos probó ser de mucha utilidad, así como lo fue

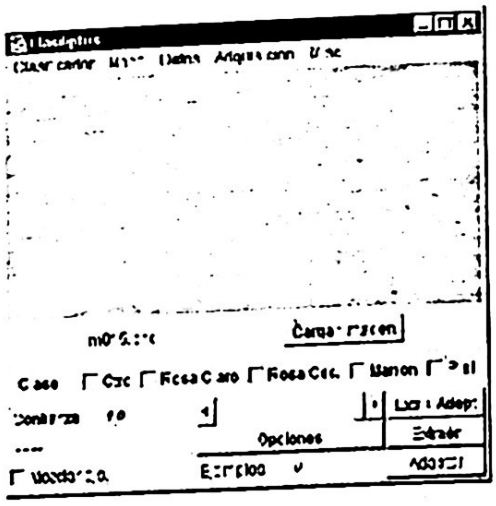



Figura 2: Interfaz de usuario

el poder modificar o generar dichos archivos manualmente o desde otras aplicaciones y cargarlos desde el programa para algunos casos especiales.

Otra de las ventajas del framework es su interfaz genérica para la configuración, entrenamiento, clasificación y evaluación de desempeño, permitiendo con la misma lógica utilizar todos los clasificadores soportados/implementados. La *matriz de confusión* provee, para sistemas supervisados, una buena codificación que sintetiza el comportamiento del sistema para una aplicación general. En particular para la *configuración* mostró ser de mucha utilidad la interfaz *Configurable* definida para tales propósitos en el framework.

## 7.2. La clasificación de maderas

El algoritmo utilizado finalmente fue entonces el *K-Means*, obteniéndose resultados para diversas cantidades de *grupos* o *clusters*.

Este algoritmo, de muy sencilla implementación, probó ser una opción suficiente para los requerimientos impuestos sobre el problema y dentro de las opciones que se tenían para elegir.

No se descartan otras alternativas de algoritmos no supervisados como *K-nearest-neighbors* o *Kohonen*. K-Means fue una opción lógica por su sencillez de implementación y depuración.

En todo caso se utilizó una *métrica* para el *espacio de características* que ponderó de igual forma a las tres componentes $H$, $S$ y $V$, no habiéndose realizado análisis específicos respecto a la sensibilidad relativa que cada componente tiene a la vista humana.

De todas formas, el propio formato *HSV* está definido en base a estos criterios [20], lo que ayudó sin duda a obtener un resultado aceptable con un algoritmo fuertemente dependiente de la métrica utilizada como el K-Means.

En la figura 3 pueden verse cuatro ejemplares correspondientes a cada uno de los cuatro *grupos* definidos por el clasificador *K-Means* con $K = 4$.

Se observó que una separación de 4 o 5 grupos de maderas resulta en grupos cuyas maderas son aceptablemente similares, al menos según los criterios de los *expertos* que observaron los resultados. Cabe notar entre estos expertos se encuentra al menos un especialista en el tema de las maderas: el Ing. Roberto Bavosi de la empresa *Eufores*, y un especialista en tratamiento de imágenes: el Dr. Guillermo Sapiro, profesor de la Universidad de Minnesota, EE.UU. .

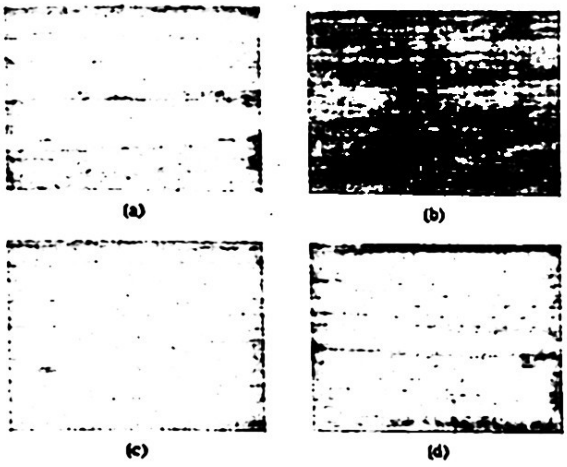



Figura 3: Ejemplos de maderas pertenecientes a los cuatro grupos creados.

## 7.3. Bayes y Backpropagation

El otro punto pendiente es la prueba de los algoritmos de clasificación implementados. Para probar dichos algoritmos, en particular *Backpropagation* y *Bayes*, se utilizó como *experto* al propio algoritmo *K-Means*.

Con la agrupación realizada por K-Means se crearon ejemplos de entrenamiento supervisado y con ellos fueron entrenados y probados los algoritmos Backpropagation y Bayes. Las pruebas fueron realizadas para diversas cantidades de grupos ($K = 2,3,4,5,6,7,8,9,10$). Los resultados mostrados a continuación corresponden a $K = 4$, cantidad que fue encontrada como las más adecuada para el problema.

Los clasificadores basados en *Backpropagation* y *Bayes* fueron entrenados con la mitad de los ejemplos generados (que suman 282 ejemplos) y probados con la otra mitad, observándose los resultados de las matrices de confusión obtenidas. A continuación se listan tablas y figuras que muestran los resultados obtenidos.

| | | | | |
|---|---|---|---|---|
| 44 | 7 | 0 | 6 | 0.77 |
| 0 | 97 | 0 | 0 | 1.00 |
| 0 | 2 | 47 | 0 | 0.96 |
| 2 | 4 | 0 | 70 | 0.92 |
| 0.96 | 0.88 | 1.00 | 0.92 | 0.92 |

Cuadro 1: Matriz de confusión para una red Backpropagation para K=4. Los parámetros utilizados: $epocas = 500$, $neuronasocultas = 25$, $learning_rate = 0{,}001$, $momentum = 0{,}8$.

| | | | | |
|---|---|---|---|---|
| 54 | 2 | 0 | 1 | 0.95 |
| 1 | 90 | 6 | 0 | 0.93 |
| 0 | 1 | 48 | 0 | 0.98 |
| 1 | 0 | 0 | 75 | 0.99 |
| 0.98 | 0.97 | 0.89 | 0.99 | 0.96 |

Cuadro 2: Clasificador de Bayes para K=4

# 8. Conclusiones

Obtuvimos como resultado de este trabajo un framework de clasificación de patrones que en su primera versión permite la creación de aplicaciones en forma razonablemente rápida y con un buen nivel de flexibilidad. Esto se cumple especialmente dentro de la familia de clasificadores estadísticos y con algún pequeño inconveniente, pero no impedimento, para los otros dos tipos más conocidos de clasificadores: KBS y estructurales, como puede ser la utilización de elementos de tipo *double* para representar etiquetas o símbolos.

La posible excepción son quizá los clasificadores lógico-combinatorios, cuya estructura en algunos casos puede requerir tipos especiales no previstos actualmente.

En cuanto al paquete como herramienta de software se pueden destacar:

- En sí, el paquete es extensible, pudiéndose mejorar su infraestructura para, por ejemplo, abarcar paradigmas como el lógico-combinatorio.
- Está ampliamente documentado, a través de su manual de usuario y su documentación de API (interfaz de programación).
- Es suficientemente genérico como para abarcar una buena parte de los algoritmos de clasificación existentes.
- La interfaz de programación resulta suficientemente intuitiva, permitiendo un rápido desarrollo de nuevos prototipos de clasificadores.
- La arquitectura del paquete no agrega recargas al tiempo de ejecución.

Entre las decisiones más acertadas está sin duda la elección de los estándares *DOM* y *XML* para la codificación, almacenamiento y transmisión de los datos que definen a un sistema de clasificación, su configuración y estado. Esto probó ser de enorme utilidad en la aplicación final, minimizando la lógica de entrada/salida para prácticamente cualquier operación de este tipo para patrones, ejemplos, clasificadores, resultados, etc.

Otra decisión acertada fue sin duda la incorporación de un mecanismo genérico de *configuración*, facilitando enormemente el desarrollo de la aplicación final.

El mecanismo de control de *nivel de logging o verborragia* también ayuda a monitorear los algoritmos implementados con detalle o reducir los mensajes al mínimo según se desee o necesite.

## 8.1. Subframework de redes neuronales

Este módulo terminó ocupando gran parte del paquete total, quizá debido a la mayor especificidad de los algoritmos involucrados. El resultado es una base para el desarrollo de redes neuronales que bien puede utilizarse aparte del framework general para otros fines que no sean clasificación.

El diseño obtenido brinda un buen nivel de ortogonalidad en base a la formulación de una red neuronal como una clase compleja, formada por unidades, conexiones y *managers* que controlan cada aspecto de su funcionalidad.

La estrategia de dividir los algoritmos en una parte *global* (dentro de los managers) y *local* (dentro de las neuronas) permite definir redes neuronales *híbridas* que utilicen distintas variantes de un mismo algoritmo (por ejemplo *Backpropagation*) en neuronas distintas (ver [16]). Creemos que esto es un punto fuerte del framework que permitirá, para quien lo utilice, experimentar con nuevos tipos de redes neuronales en base a este paradigma.

Una posible limitación es la no utilización de *managers* a nivel *local* (dentro de una neurona) como se hizo en `NeuralNet` para los algoritmos globales.

Otra limitación es la falta de un mecanismo de adaptación *general* para *todos* los parámetros de una unidad, no sólo los *pesos de las conexiones*. Si bien para la gran mayoría de las redes existentes esto no es problema, sistemas como el *ANFIS* [10] necesitan esta cualidad para su correcta adaptación. De hecho,

la actual implementación del ANFIS *no* modifica uno de los parámetros importantes debido a este diseño (ver [21] por detalles).

El problema de la falta de un mecanismo *amigable* de construcción de sistemas de clasificación se nota fuertemente en este módulo. Si bien por esta razón se creó la clase `NetBuilder` desde sus comienzos, su diseño no ha llegado a una forma aceptable aún y su uso es un tanto engorroso.

Por último, de la experiencia en el *uso* de las redes neuronales se concluye que sería muy útil tener algún mecanismo de *reportes* o *auditoría* de la evolución interna de los parámetros y las señales durante el entrenamiento. Se pensó en un mecanismo de *auditoría*, implementable mediante otro *manager* (por ejemplo "NetAuditor"), pero no se pudo llegar a implementar dicho mecanismo dentro del tiempo admisible.

## 8.2. El framework como base para la aplicación final

Del uso del framework en la aplicación es que surgieron la mayoría de las conclusiones respecto a él citadas anteriormente. En general, una buena prueba de la utilidad del framework es el tiempo relativo de desarrollo de la aplicación que tomó la parte de clasificación.

La aplicación comenzó a ser desarrollada a mediados de Enero del 2002 y fue culminada en la primera semana de Abril del mismo año. De los casi 3 meses que llevó el desarrollo y puesta a punto, solo una semana de implementación y depuración estuvo directamente ligada a la parte de clasificación. El resto del tiempo fue dedicado a los módulos de *adquisición*, *preprocesamiento y calibración* y *comunicación con el SCADA* y a la interfaz gráfica (junto con la interfaz de comunicación Java/C++: JNI).

# 9. Conclusiones finales

Finalmente se obtuvo un framework que en su primera versión permite desarrollar sistemas de clasificación de patrones con relativa facilidad, a la vez resolviendo algunos problemas comunes a los que un desarrollador se enfrenta al crear este tipo de aplicaciones.

## 9.1. Ventajas de utilizar el framework

Al tener resueltos la estructura lógica y el diseño general del sistema de clasificación, el desarrollador puede concentrar sus esfuerzos en el perfeccionamiento de las etapas de extracción de características clasificación.

Además se ahorrará tiempo en resolver problemas comunes como la codificación interna, el almacenamiento y la recuperación de la información almacenada por un sistema arbitrario, tanto de entrenamiento como de configuración.

El paquete está desarrollado en C++, lenguaje muy difundido en el área de la computación científica y que combina eficiencia con un alto nivel de abstracción. El desarrollador podrá beneficiarse de estas características al trabajar sobre este framework.

## 9.2. Desventajas

El grado de generalidad alcanzado por en Framework tiene como consecuencia negativa el inevitable enlentecimiento general del sistema debido a la lógica adicional requerida para manejar cierta diversidad de situaciones. Parte de esta lógica está implícita en el mecanismo de *polimorfismo dinámico*, que forma parte importante del paradigma de Orientación a Objetos bajo su implementación en el lenguaje C++. Claramente, una aplicación realizada desde cero para una aplicación puntual será seguramente más eficiente en tiempos de ejecucion que una realizada para el Framework con idéntica funcionalidad.

También habrá un tiempo de adaptación por parte del desarrollador, necesario para que éste pueda captar la filosofía general que gobierna el diseño del framework y así implementar correctamente aplicación en base a este último.

## 10. Trabajo Futuro

Quedan pendientes todas las posibles mejoras mencionadas: facilitar la construcción de sistemas de clasificación, extender el framework para comprender a *todo el sistema de clasificación* (es decir, agregando *adquisición* y *extracción*).

También sería muy interesante definir una interfaz *asíncrona* para el sistema, para así permitir un proceso en *pipeline* en el cual cada etapa trabaje continuamente y la comunicación entre etapas sea mediante *eventos*. Esta interfaz requeriría el uso de técnicas de *programación concurrente/multiproceso* y *señalización* e implicaría un importante tiempo de desarrollo, con impacto global en todo el framework. Cabe notar que este objetivo fue tenido en mente desde el comienzo del desarrollo, pero no hubo lugar para su realización dentro del tiempo disponible.

Por la parte de las redes neuronales, es necesario focalizar en aumentar la velocidad de cálculo. Si bien la velocidad actual no es mala, el entrenamiento de redes neuronales suele ser largo y costoso en tiempos de ejecución. Varias alternativas de diseño, de carácter demasiado técnico para incluir aquí, quedan planteadas como posibles mejoras.

Siguiendo con las redes neuronales, un importante avance en el diseño es la generalización del mecanismo de adaptación para poder modificar *todos* los parámetros de una unidad y no solo los pesos de las conexiones, con vistas a mejorar las implementaciones de redes como el ANFIS nombradas anteriormente.

## Referencias


[1] R. O. Duda and P. E. Hart, *Pattern Classification And Scene Analysis.* Wiley/Interscience, 1974.

[2] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles*, ch. 4. Addison-Wesley, 1974.

[3] J. R. Shulcloper, A. G. Arenas, and J. F. M. Trinidad, *Enfoque Lógico Combinatorio al Reconocimiento de Patrones.* Instituto Politécnico Nacional - Mexico, 1999.

[4] F. J. Cortijo, "Reconocimiento de patrones y análisis de imágenes," tech. rep., Universidad de Granada, 1998.

[5] R. C. Crida, *A Machine Vision Approach to Rock Fragmentation Analysis.* PhD thesis, University of Cape Town, 1995.

[6] A. S. Pandya and R. B. Macy, *Pattern Recognition with Neural Networks in C++.* CRC Press, 1996.

[7] R. O. Duda, *Pattern Recognition Tutorial - http://www-engr.sjsu.edu/~knapp/HCIRODPR/PR_model/PR_model.htm.*

[8] N. Chomsky, "Three models for the description of language," in *Proc. of the Group of Information Theory*, vol. 2, pp. 113–124, 1956.

[9] M. Stefik, *Introduction to Knowledge Systems.* Morgan Kaufmann, 1995.

[10] J.-S. R. Jang, *Neuro-Fuzzy Modeling: Architectures, Analyses and Applications.* PhD thesis, University of California, 1992.

[11] W. W. W. Consortium, *DOM Reference - http://www.w3.org/TR/REC-DOM-Level-1/.*

[12] W. W. W. Consortium, *XML Reference - http://www.w3.org/TR/REC-xml1/.*

[13] T. A. X. Project, *Xerces C++ XML Parser - http://xml.apache.org/xerces-c/.* The Apache Group.

[14] A. Lumsdaine, J. Siek, and L.-Q. Lee, *The Matrix Template Library http://www.osl.iu.edu/research/mtl/.* Open Systems Laboratory - Indiana University.

[15] D. M. S. James A. Freeman, *Neural Networks: Algorithms, Applications, and Programming Techniques.* Addison-Wesley, 1991.

[16] J. Gibb, "Back propagation family album," tech. rep., Department of Computing - Macquarie University, August 1996.

[17] University of Stuttgart - University of Tübingen, *SNNS User Manual, Version 4.2 - http://www-ra.informatik.uni-tuebingen.de/downloads/SNNS/SNNSv4.2.Manual.pdf.*

[18] R. Hech-Nielsen, *Neurocomputing.* Addison-Wesley, 1990.

[19] Matlab(TM), *Neural Network Tookbox Documentation - http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/* MatLab(TM).

[20] R. C. Gonzalez and R. E. Woods, *Digital Image Processing.* Addison-Wesley, 1992.

[21] I. Ramírez and A. Alcarraz, *Manual del Usuario de ClasiLiptus - http://www.iie.edu.uy/~nacho/clasiliptus/manual.pdf.* Instituto de Ingeniería Eléctrica - Facultad de Ingeniería, Julio Herrera y Reissig 565, Montevideo (Uruguay), 1997.

# Lernmatrix de Steinbuch: Condiciones necesarias y suficientes para recuperación perfecta

F. A. Sánchez-Garfias, J. L. Díaz-de-León-S. & C. Yáñez-Márquez
Centro de Investigación en Computación
del Instituto Politécnico Nacional

**Resumen**

En este artículo se presenta una manera alternativa de expresar las fases de aprendizaje y recuperación de patrones para la *Lernmatrix*, modelo de memoria asociativa creado y descrito por Karl Steinbuch en 1961, en uno de los trabajos que se han convertido en pioneros en esta área, y que constituye un antecedente crucial en el desarrollo de los modelos actuales de memorias asociativas. Además se presentan los resultados de un estudio sistemático que se realizó, por primera vez en la historia de la *Lernmatrix*, con el propósito de evidenciar las condiciones necesarias y suficientes para recuperación perfecta de patrones.

Los resultados obtenidos permiten colocar a la *Lernmatrix*, después cuatro décadas de haber surgido, como una buena alternativa para clasificación y reconocimiento de patrones.

## 1. Introducción.

Las memorias asociativas han merecido la atención de numerosos investigadores internacionales desde hace más de cuatro décadas. Uno de los pioneros fue el científico alemán Karl Steinbuch quien, a principios de la década de los sesenta, ideó, desarrolló y aplicó la *Lernmatrix* (Steinbuch, 1961; Steinbuch & Frank, 1961).

La *Lernmatrix* constituye un antecedente crucial en el desarrollo de los modelos actuales de memorias asociativas, y constituye uno de los primeros intentos exitosos de codificar información en arreglos cuadriculados conocidos como *crossbar* (Simpson, 1990).

Una memoria asociativa tiene como propósito fundamental: recuperar correctamente patrones completos a partir de patrones de entrada, los cuales pueden estar alterados con ruido aditivo, sustractivo o combinado.

El problema inherente al funcionamiento de las memorias asociativas se escinde en dos fases claramente distinguibles:

1. Fase de *aprendizaje* (generación de la memoria asociativa)
2. Fase de *recuperación* (operación de la memoria asociativa)

En ambas fases, una *memoria asociativa* M puede formularse como un sistema de entrada y salida, idea que se esquematiza a continuación:

$$\mathbf{x} \longrightarrow \boxed{\mathrm{M}} \longrightarrow \mathbf{y}$$

El *patrón de entrada* está representado por un vector columna denotado por $\mathbf{x}$ y el *patrón de salida*, por el vector columna denotado por $\mathbf{y}$. Cada uno de los patrones de entrada forma una *asociación* con el correspondiente patrón de salida. La notación para una asociación es similar a la de una pareja ordenada; por ejemplo, los patrones $\mathbf{x}$ y $\mathbf{y}$ del esquema forman la asociación $(\mathbf{x}, \mathbf{y})$.

Dado un número entero positivo $k$ específico, la asociación correspondiente será $(\mathbf{x}^k, \mathbf{y}^k)$.

La memoria asociativa M se representa mediante una matriz cuya componente $ij$-ésima es $m_{ij}$ (Palm, Schwenker, Sommer & Strey, 1997); la matriz M se genera a partir de un conjunto finito de asociaciones conocidas de antemano: éste es el *conjunto fundamental de asociaciones*, o simplemente *conjunto fundamental*. Se denota por $p$ la cardinalidad del conjunto fundamental ($p$ es un número entero positivo).

Si $\mu$ es un índice, el conjunto fundamental se representa de la siguiente manera:

$$\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu = 1, 2, ..., p\}$$

**Definición 1.1** *A los patrones que conforman las asociaciones del conjunto fundamental, se les llama patrones fundamentales.*

La naturaleza del conjunto fundamental proporciona un importante criterio para clasificar las memorias asociativas (Kohonen,1972).

**Definición 1.2** *Si se cumple que* $\mathbf{x}^\mu = \mathbf{y}^\mu \; \forall \mu \in \{1, 2, ..., p\}$, *se dice que la memoria es autoasociativa; de otro modo, la memoria es heteroasociativa. Para una memoria heteroasociativa se puede afirmar lo siguiente:* $\exists \mu \in \{1, 2, ..., p\}$ *para el que se cumple que* $\mathbf{x}^\mu \neq \mathbf{y}^\mu$.

**Definición 1.3** *Si al presentarle a la memoria* M *un patrón alterado* $\tilde{\mathbf{x}}^\omega$ *como entrada* ($\omega \in \{1, 2, ..., p\}$), M *responde con el correspondiente patrón fundamental de salida* $\mathbf{y}^\omega$, *se dice que la recuperación es perfecta.*

**Definición 1.4** *Una memoria perfecta es aquella que realiza recuperaciones perfectas para todos los patrones fundamentales.*

$A$ y $B$ son dos conjuntos que cumplen con lo siguiente: las componentes de los vectores columna que representan a los patrones, tanto de entrada como de salida, son elementos del conjunto $A$, y las entradas de la matriz M son elementos del conjunto $B$.

No hay requisitos previos ni limitaciones respecto de la elección de estos dos conjuntos, por lo que no necesariamente deben ser diferentes o poseer características especiales. Sean $m, n$ números enteros positivos. Se denota por $n$ la

dimensión de los patrones de entrada, y por $m$ la dimensión de los patrones de salida; claramente, nada impide que los valores de $m$ y de $n$ sean iguales. Aún más, uno de los requisitos que debe cumplir una memoria autoasociativa es que la dimensión de los patrones de entrada sea igual a la dimensión de los patrones de salida; por otro lado, si en una memoria sucede que $m \neq n$, es evidente que la memoria debe ser heteroasociativa.

Cada vector columna que representa a un patrón de entrada tiene $n$ componentes cuyos valores pertenecen al conjunto $A$, y cada vector columna que representa a un patrón de salida posee $m$ componentes cuyos valores pertenecen al conjunto $A$. Es decir:

$$\mathbf{x}^\mu \in A^n \text{ y } \mathbf{y}^\mu \in A^m \; \forall \mu \in \{1,2,...,p\}$$

La $j$-ésima componente de un vector columna se indica con la misma letra del vector, pero sin negrilla, colocando a $j$ como subíndice ($j \in \{1,2,...,n\}$ o $j \in \{1,2,...,m\}$ según corresponda). La $j$-ésima componente de un vector columna $\mathbf{x}^\mu$ se representa por

$$x_j^\mu$$

Los vectores columna que representan a los patrones fundamentales de entrada y de salida son, respectivamente:

$$\mathbf{x}^\mu = \begin{pmatrix} x_1^\mu \\ x_2^\mu \\ \vdots \\ x_n^\mu \end{pmatrix} \in A^n \qquad \mathbf{y}^\mu = \begin{pmatrix} y_1^\mu \\ y_2^\mu \\ \vdots \\ y_m^\mu \end{pmatrix} \in A^m$$

No obstante que Steinbuch presentó la *Lernmatrix* hace más de cuatro décadas, ningún investigador, incluyendo al propio Steinbuch, se ha dado a la tarea de estudiar con rigor científico las condiciones necesarias y suficientes para recuperación perfecta del conjunto fundamental y de patrones que no pertenezcan a éste. En este trabajo se presenta, **por primera vez**, un estudio sistemático de esta naturaleza.

## 2. El trabajo de Steinbuch.

La *Lernmatrix* es una memoria heteroasociativa que puede funcionar como un clasificador de patrones binarios si se escogen adecuadamente los patrones de salida; es un sistema de entrada y salida que al operar acepta como entrada un patrón binario $\mathbf{x}^\mu \in A^n$, $A = \{0,1\}$ y produce como salida la clase $\mathbf{y}^\mu \in A^m$ que le corresponde (de entre $m$ clases diferentes), codificada ésta con un método simple, a saber: para representar la clase $k \in \{1,2,...,m\}$, se asignan a las componentes del vector de salida $\mathbf{y}^\mu$ los siguientes valores: $y_k^\mu = 1$, y $y_j^\mu = 0$ para $j = 1,2...,k-1,k+1,...m$

En la tabla se esquematiza la fase de aprendizaje para la *Lernmatrix* Steinbuch, con la pareja de patrones fundamentales $(\mathbf{x}^\mu, \mathbf{y}^\mu) \in A^n \times A^m$.

| | $x_1^\mu$ | $x_2^\mu$ | $\cdots$ | $x_j^\mu$ | $\cdots$ | $x_n^\mu$ |
|---|---|---|---|---|---|---|
| $y_1^\mu$ | $m_{11}$ | $m_{12}$ | $\cdots$ | $m_{1j}$ | $\cdots$ | $m_{1n}$ |
| $y_2^\mu$ | $m_{21}$ | $m_{22}$ | $\cdots$ | $m_{2j}$ | $\cdots$ | $m_{2n}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | | $\vdots$ | | $\vdots$ |
| $y_i^\mu$ | $m_{i1}$ | $m_{i2}$ | $\cdots$ | $m_{ij}$ | $\cdots$ | $m_{in}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | | $\vdots$ | | $\vdots$ |
| $y_m^\mu$ | $m_{m1}$ | $m_{m2}$ | $\cdots$ | $m_{mj}$ | $\cdots$ | $m_{mn}$ |

Cada uno de los componentes $m_{ij}$ de M, la *Lernmatrix* de Steinbuch, tiene valor cero al inicio, y se actualiza de acuerdo con la regla $m_{ij} + \Delta m_{ij}$, donde:

$$\Delta m_{ij} = \begin{cases} +\varepsilon \text{ si } y_i^\mu = 1 = x_j^\mu \\ -\varepsilon \text{ si } y_i^\mu = 1 \text{ y } x_j^\mu = 0 \\ 0 \text{ en otro caso} \end{cases}$$

siendo $\varepsilon$ una constante positiva escogida previamente.

La fase de recuperación consiste en encontrar la clase a la que pertenece un vector de entrada $\mathbf{x}^\omega \in A^n$ dado. Encontrar la clase significa obtener coordenadas del vector $\mathbf{y}^\omega \in A^m$ que le corresponde al patrón $\mathbf{x}^\omega$; en virtud del método de construcción de los vectores $\mathbf{y}^\mu$ la clase debería obtenerse ambigüedad.

La $i$-ésima coordenada $y_i^\omega$ del vector de clase $\mathbf{y}^\omega \in A^m$ se obtiene como indica la siguiente expresión, donde $\bigvee$ es el operador *máximo*:

$$y_i^\omega = \begin{cases} 1 \text{ si } \sum_{j=1}^n m_{ij}.x_j^\omega = \bigvee_{h=1}^m \left[\sum_{j=1}^n m_{hj}.x_j^\omega\right] \\ 0 \text{ en otro caso} \end{cases}$$

# 3. Caracterización alterna de las fases de aprendizaje y recuperación

En esta sección definiremos una función que nos permita desarrollar forma alterna de caracterizar las fases de aprendizaje y recuperación.

**Definición 3.1** *Sean* $A = \{0, 1\}$ *y* $C = \{-1, 1\}$. *Llamaremos función de Steinbuch a una función* $f : A \to C$ *que cumpla con la siguiente propiedad:*

$$f(0) = -1$$
$$f(1) = 1$$

Claramente, existe un número infinito de funciones de Steinbuch, algunas las cuales se ejemplifican a continuación:

**Ejemplo 3.1** $f(x) = \begin{cases} -1, \textit{ si } x = 0 \\ 1, \textit{ si } x = 1 \end{cases}$

**Ejemplo 3.2** $f(x) = 2x - 1$

**Ejemplo 3.3** $f(x) = -(-1)^x$

Una vez definida una función de Steinbuch $f$, podemos reescribir la expresión 2 de la siguiente forma:

$$\Delta m_{ij} = \epsilon y_i^\mu f(x_j^\mu) \tag{5}$$

La tabla muestra que la expresión 5 es equivalente a la 2

| $y_i^\mu$ | $x_j^\mu$ | $f(x_j^\mu)$ | $\Delta m_{ij}$ |
|---|---|---|---|
| 1 | 1 | 1 | $+\epsilon$ |
| 1 | 0 | $-1$ | $-\epsilon$ |
| 0 | 1 | 1 | 0 |
| 0 | 0 | $-1$ | 0 |

Para lograr caracterizar la regla de aprendizaje de la Lernmatrix 5 en forma matricial, se requiere la siguiente definición.

**Definición 3.2** *Sean* $A = \{0,1\}$ *y* $C = \{-1,1\}$ *y sea* $f : A \to C$ *una función de Steinbuch. Llamaremos función vectorial de Steinbuch con respecto a* $f$ *a una función* $\mathbf{F} : A^n \to C^n$, *tal que:*

$$\mathbf{F(x)} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \dots \\ f(x_n) \end{pmatrix}$$

Ahora, sea $\Delta\mathbf{M}$ es una matriz cuya componente $ij$-ésima es $\Delta m_{ij}$, y $\mathbf{F}$ una función vectorial de Steinbuch con respecto a $f$.

$$\Delta\mathbf{M} = \begin{pmatrix} \Delta m_{11} & \Delta m_{12} & \dots & \Delta m_{1j} & \dots & \Delta m_{1n} \\ \Delta m_{21} & \Delta m_{22} & \dots & \Delta m_{2j} & \dots & \Delta m_{2n} \\ \dots & \dots & & \dots & & \dots \\ \Delta m_{i1} & \Delta m_{i2} & \dots & \Delta m_{ij} & \dots & \Delta m_{in} \\ \dots & \dots & & \dots & & \dots \\ \Delta m_{m1} & \Delta m_{m2} & \dots & \Delta m_{mj} & \dots & \Delta m_{mn} \end{pmatrix}$$

Al usar la expresión 5 en la matriz anterior, se tiene:

$$\Delta\mathbf{M} = \begin{pmatrix} \epsilon y_1^\mu f(x_1^\mu) & \epsilon y_1^\mu f(x_2^\mu) & \dots & \epsilon y_1^\mu f(x_j^\mu) & \dots & \epsilon y_1^\mu f(x_n^\mu) \\ \epsilon y_2^\mu f(x_1^\mu) & \epsilon y_1^\mu f(x_2^\mu) & \dots & \epsilon y_2^\mu f(x_j^\mu) & \dots & \epsilon y_2^\mu f(x_n^\mu) \\ \dots & \dots & & \dots & & \dots \\ \epsilon y_i^\mu f(x_1^\mu) & \epsilon y_i^\mu f(x_2^\mu) & \dots & \epsilon y_i^\mu f(x_j^\mu) & \dots & \epsilon y_i^\mu f(x_n^\mu) \\ \dots & \dots & & \dots & & \dots \\ \epsilon y_m^\mu f(x_1^\mu) & \epsilon y_m^\mu f(x_2^\mu) & \dots & \epsilon y_m^\mu f(x_j^\mu) & \dots & \epsilon y_m^\mu f(x_n^\mu) \end{pmatrix}$$

Factorizando $\varepsilon$:

$$\Delta \mathrm{M} = \varepsilon \begin{pmatrix} y_1^\mu f(x_1^\mu) & y_1^\mu f(x_2^\mu) & \cdots & y_1^\mu f(x_j^\mu) & \cdots & y_1^\mu f(x_n^\mu) \\ y_2^\mu f(x_1^\mu) & y_1^\mu f(x_2^\mu) & \cdots & y_2^\mu f(x_j^\mu) & \cdots & y_2^\mu f(x_n^\mu) \\ \cdots & \cdots & & \cdots & & \cdots \\ y_i^\mu f(x_1^\mu) & y_i^\mu f(x_2^\mu) & \cdots & y_i^\mu f(x_j^\mu) & \cdots & y_i^\mu f(x_n^\mu) \\ \cdots & \cdots & & \cdots & & \cdots \\ y_m^\mu f(x_1^\mu) & y_m^\mu f(x_2^\mu) & \cdots & y_m^\mu f(x_j^\mu) & \cdots & y_m^\mu f(x_n^\mu) \end{pmatrix}$$

Es decir:

$$\Delta \mathrm{M} = \varepsilon \begin{pmatrix} y_1^\mu \\ y_2^\mu \\ \cdots \\ y_i^\mu \\ \cdots \\ y_m^\mu \end{pmatrix} \cdot \begin{pmatrix} f(x_1^\mu) & f(x_2^\mu) & \cdots & f(x_j^\mu) & \cdots & f(x_n^\mu) \end{pmatrix}$$

de donde se concluye que:

$$\Delta \mathrm{M} = \varepsilon \mathbf{y}^\mu \cdot (\mathbf{F}(\mathbf{x}^\mu))^T$$

Finalmente, y tomando en cuenta que para formar la Lernmatrix se suma todas las $\Delta$M para cada valor de $\mu$, podemos, formular las fases de aprendizaje y de recuperación con base en la nueva caracterización que proponemos.

*Fase de Aprendizaje de la LernMatrix*

Sea $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu \in \{1, 2, .., m\}\}$ un conjunto fundamental y F una función vectorial de Steinbuch con respecto a $f$. La Lernmatrix M para el conjunto fundamental se construye de acuerdo con la siguiente regla:

$$\mathbf{M} = \varepsilon \sum_{\mu=1}^{m} \mathbf{y}^\mu \cdot (\mathbf{F}(\mathbf{x}^\mu))^T$$

*Fase de recuperación de la LernMatrix*

Sea M una Lernmatrix y $\mathbf{x}^\omega$ un patrón de dimensión $n$. El patrón $\tilde{\mathbf{y}}^\omega$ perado a partir de $\mathbf{x}^\omega$ y M se determina de la siguiente forma:

$$\mathbf{z}^\omega = \mathbf{M} \cdot \mathbf{x}^\omega$$

$$\tilde{y}_i^\omega = \begin{cases} 1 \text{ si } z_i^\omega = \bigvee_{h=1}^{m} z_h^\omega \\ 0 \text{ en otro caso} \end{cases}$$

Donde $\tilde{\mathbf{y}}^{\omega}$ no es necesariamente igual a $\mathbf{y}^{\omega}$. En particular, si $\tilde{\mathbf{y}}^{\omega} = \mathbf{y}^{\omega}$, entonces la recuperación es perfecta, de acuerdo con la definición 1.3.

# 4. Condiciones para recuperación perfecta

Con las expresiones 7 y 8, hemos caracterizado, en forma matricial, las reglas de aprendizaje y de recuperación de la Lernmatrix. Echando mano de esta caracterización y algunas definiciones adicionales, iniciaremos la investigación de las condiciones necesarias y suficientes para que la Lernmatrix recupere todo su conjunto fundamental de manera perfecta, de acuerdo con la definición 1.3.

**Definición 4.1** *Sea* $A = \{0,1\}$ *y sean* $\mathbf{x}^{\alpha}, \mathbf{x}^{\beta} \in A^n$ *dos patrones. Se dice que* $\mathbf{x}^{\alpha}$ *es igual que* $\mathbf{x}^{\beta}$ *(simbolizado* $\mathbf{x}^{\alpha} = \mathbf{x}^{\beta}$*) si y sólo si* $\forall i \in \{1,2,..,n\}$ *se cumple que* $x_i^{\beta} = x_i^{\alpha}$.

**Definición 4.2** *Sea* $A = \{0,1\}$ *y sean* $\mathbf{x}^{\alpha}, \mathbf{x}^{\beta} \in A^n$ *dos patrones. Se dice que* $\mathbf{x}^{\alpha}$ *es menor o igual que* $\mathbf{x}^{\beta}$ *(simbolizado* $\mathbf{x}^{\alpha} \leq \mathbf{x}^{\beta}$*) si y sólo si* $\forall i \in \{1,2,..,n\}$ *se cumple que* $x_i^{\beta} = 1$ *siempre que* $x_i^{\alpha} = 1$.

**Ejemplo 4.1** *Sean* $\mathbf{x}^{\alpha} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$ *y* $\mathbf{x}^{\beta} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}$, *entonces* $\mathbf{x}^{\alpha} \leq \mathbf{x}^{\beta}$, *ya que* $\forall i \in \{1,2,3,4,5,6\}$ *se cumple que* $x_i^{\beta} = 1$ *cada vez que* $x_i^{\alpha} = 1$

**Ejemplo 4.2** *Sean* $\mathbf{x}^{\alpha} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$ *y* $\mathbf{x}^{\beta} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$, *entonces es falso que* $\mathbf{x}^{\alpha} \leq \mathbf{x}^{\beta}$, *porque* $x_1^{\alpha} = 1$ *y* $x_1^{\beta} = 0$.

Después de haber descrito las herramientas necesarias, se presentan un lema, un teorema y un corolario, los cuales forman la parte central del presente trabajo.

**Lema 4.1** *Sea* M *una Lernmatrix, entonces no es posible recuperar de manera perfecta el patrón* $\mathbf{x} = 0$, *donde* 0 *es el vector con ceros en todas sus entradas, es decir, que* $x_i = 0\ \forall i \in \{1,2,..,n\}$.

**Demostración.** Sea M una Lernmatrix y sea $\mathbf{x}$ un patrón de dimension $n$ tal que $x_i = 0\ \forall i \in \{1,2,..,n\}$.

Para la fase de recuperación, se necesita primero calcular el vector $\mathbf{z} = \mathbf{M} \cdot \mathbf{x}$. Puesto que $x_i = 0\ \forall i \in \{1,2,..,n\}$, entonces $z_i = 0\ \forall i \in \{1,2,..,m\}$. Luego, entonces, $\bigvee_{h=1}^{m} [z_h] = 0$, de aqui que $y_i = 1\ \forall i \in \{1,2,..,m\}$ .Dado que todos los $\mathbf{y}^{\mu}$ del conjunto fundamental se crearon de forma que $y_{\mu}^{\mu} = 1$ y $y_{\alpha}^{\mu} = 0\ \forall \alpha \neq \mu$,

el vector recuperado $\mathbf{y}$ no coincide con ningún $\mathbf{y}^\mu$, por tanto, $\mathbf{x}$ no puede recuperado de manera perfecta y el lema queda demostrado. ■

La importancia del lema anterior radica en que, sin importar las características del conjunto fundamental, el patrón 0 nunca podrá ser parte de una Lernmatrix, tanto como parte de su conjunto fundamental, como en el caso de sea un patrón alterado de otro fundamental.

El siguiente teorema, y el corolario que se desprende a partir de él, muestran las condiciones necesarias y suficientes que deben cumplir los vectores del conjunto fundamental para que la recuperación sea perfecta, de acuerdo la definición 1.3.

**Teorema 4.2** *Sea* $\mathbf{M}$ *una Lernmatrix y sea* $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu \in \{1,2,..,m\}\}$ *el conjunto fundamental de* $\mathbf{M}$. *Es posible recuperar de manera perfecta el conjunto fundamental de* $\mathbf{M}$ *si y sólo si* $\forall\ \alpha, \beta \in \{1,2,..,m\}$ *y* $\alpha \neq \beta$, *la proposición* $\leq \mathbf{x}^\beta$ *es falsa.*

**Demostración.** Caso 1: $\exists\ \alpha \in \{1,2,..,m\}$ tal que $\mathbf{x}^\alpha = \mathbf{0}$.

Claramente, se cumple que $\mathbf{x}^\alpha \leq \mathbf{x}^\beta\ \forall \beta \in \{1,2,..,m\}$ y por ello la proposición $\mathbf{x}^\alpha \leq \mathbf{x}^\beta$ es verdadera al menos para un caso. Por otro lado, por lema 4.1 se afirma que el patrón $\mathbf{x}^\alpha$ no se recupera de manera perfecta; es decir, es posible recuperar el conjunto fundamental de manera perfecta. Con ello teorema queda demostrado.

Caso2: $\mathbf{x}^\mu \neq 0\ \forall \mu \in \{1,2,..,m\}$.

Supongamos que $\forall\ \alpha, \beta \in \{1,2,..,m\}$ y $\alpha \neq \beta$ la proposición $\mathbf{x}^\alpha \leq \mathbf{x}^\beta$ falsa. De acuerdo con la expresión 7 para la fase de aprendizaje y considerando un valor $\alpha \in \{1,2,..,m\}$ cualquiera, la matriz $\mathbf{M}$ se construye como sigue:

$$\mathbf{M} = \varepsilon \sum_{\mu=1}^{m} \mathbf{y}^\mu \cdot (\mathbf{F}(\mathbf{x}^\mu))^T$$

$$\mathbf{M} = \varepsilon \left[ \begin{pmatrix} 1 \\ \cdots \\ 0 \\ \cdots \\ 0 \end{pmatrix} \cdot (\mathbf{F}(\mathbf{x}^1))^T + \ldots + \begin{pmatrix} 0 \\ \cdots \\ 1 \\ \cdots \\ 0 \end{pmatrix} \cdot (\mathbf{F}(\mathbf{x}^\alpha))^T + \right.$$
$$\left. \ldots + \begin{pmatrix} 0 \\ \cdots \\ 0 \\ \cdots \\ 1 \end{pmatrix} \cdot (\mathbf{F}(\mathbf{x}^m))^T \right]$$

$$\mathbf{M} = \varepsilon \begin{pmatrix} f(x_1^1) & f(x_2^1) & \ldots & f(x_j^1) & \ldots & f(x_n^1) \\ \cdots & \cdots & & \cdots & & \cdots \\ f(x_1^\alpha) & f(x_2^\alpha) & \ldots & f(x_j^\alpha) & \ldots & f(x_n^\alpha) \\ \cdots & \cdots & & \cdots & & \cdots \\ f(x_1^m) & f(x_2^m) & \ldots & f(x_j^m) & \ldots & f(x_n^m) \end{pmatrix}$$

Ahora, si queremos recuperar el patrón $\mathbf{y}^\alpha$ que corresponde a la salida del patrón $\mathbf{x}^\alpha$, tenemos que calcular el vector $\mathbf{z}^\alpha$ aplicando la expresión 8 que corresponde a la primera parte de la fase de recuperación:

$$\mathbf{z}^\alpha = \mathbf{M} \cdot \mathbf{x}^\alpha$$

Aplicando la expresión 9 en la expresión anterior, se tiene:

$$\mathbf{z}^\alpha = \varepsilon \begin{pmatrix} f(x_1^1) & f(x_2^1) & \dots & f(x_j^1) & \dots & f(x_n^1) \\ \dots & \dots & & \dots & & \dots \\ f(x_1^\alpha) & f(x_2^\alpha) & \dots & f(x_j^\alpha) & \dots & f(x_n^\alpha) \\ \dots & \dots & & \dots & & \dots \\ f(x_1^m) & f(x_2^m) & \dots & f(x_j^m) & \dots & f(x_n^m) \end{pmatrix} \cdot \mathbf{x}^\alpha$$

Es decir:

$$\mathbf{z}^\alpha = \varepsilon \begin{pmatrix} \sum_{j=1}^{n} f(x_j^1) x_j^\alpha \\ \dots \\ \sum_{j=1}^{n} f(x_j^\alpha) x_j^\alpha \\ \dots \\ \sum_{j=1}^{n} f(x_j^m) x_j^\alpha \end{pmatrix} \tag{10}$$

Esto significa que para algún valor $k \in \{1, 2, .., m\}$, la componente $k$-ésima de $\mathbf{z}^\alpha$ se expresa así:

$$z_k^\alpha = \varepsilon \sum_{j=1}^{n} f(x_j^k) x_j^\alpha \tag{11}$$

Específicamente las componentes $\alpha$-ésima y $\beta$-ésima, con $\beta \in \{1, 2, .., m\}$ un valor cualquiera tal que $\alpha \neq \beta$, se expresan de la siguiente manera:

$$z_\alpha^\alpha = \varepsilon \sum_{j=1}^{n} f(x_j^\alpha) x_j^\alpha \quad \text{y} \quad z_\beta^\alpha = \varepsilon \sum_{j=1}^{n} f(x_j^\beta) x_j^\alpha \tag{12}$$

Siguiendo con la expresión 8, de acuerdo con la segunda parte donde se obtienen las coordenadas $\tilde{y}_i^\omega$ y considando la manera en que se forma cada uno de los $\mathbf{y}^\mu$ del conjunto fundamental, podemos inferir que la condición en la expresión 10 para que se recupere el patrón $\mathbf{y}^\alpha$ de manera perfecta es que

$$z_\alpha^\alpha > z_\beta^\alpha \tag{13}$$

La desigualdad es estricta porque debe haber sólo un máximo para recuperación perfecta del patrón $\mathbf{y}^\alpha$, según la expresión 8.

Aplicando la expresión 12 en 13 se obtienen las siguientes desigualdades:

$$\sum_{j=1}^{n} f(x_j^\alpha)x_j^\alpha > \sum_{j=1}^{n} f(x_j^\beta)x_j^\alpha$$
$$\sum_{j=1}^{n} f(x_j^\alpha)x_j^\alpha - \sum_{j=1}^{n} f(x_j^\beta)x_j^\alpha > 0 \tag{14}$$

Definamos ahora el conjunto $T = \{j \mid x_j^\alpha = 1\}$, donde llamaremos $t$ a cardinalidad de $T$, y tomando en cuenta que no influyen en la suma los términos donde $x_j^\alpha = 0$, la expresión anterior se reduce a:

$$\sum_{j\in T} f(x_j^\alpha) - \sum_{j\in T} f(x_j^\beta) > 0$$
$$t - \sum_{j\in T} f(x_j^\beta) > 0 \tag{15}$$

Luego, es claro que $-t \leq \sum_{j\in T} f(x_j^\beta) \leq t$, ya que, de acuerdo con la definición 4, sucede que $f(x_j^\beta) = 1$ ó $f(x_j^\beta) = -1$ y además existen $t$ términos en sumatoria.

Por otro lado, $\sum_{j\in T} f(x_j^\beta) \neq t$, como se muestra a continuacion. Supongamos que: $\sum_{j\in T} f(x_j^\beta) = t$, entonces:

$$f(x_j^\beta) = 1 \; \forall j \in T$$
$$x_j^\beta = 1 \; \forall j \in T$$
$$x_j^\beta = 1 \; \forall j \text{ tal que } x_j^\alpha = 1$$
$$\mathbf{x}^\alpha \leq \mathbf{x}^\beta \text{ contradicción.}$$

Por tanto, $-t \leq \sum_{j\in T} f(x_j^\beta) < t$ y la desigualdad 15 siempre se cumple si sólo si la recuperación de $\mathbf{y}^\alpha$ es perfecta.

Finalmente, en virtud de que $\alpha$ y $\beta$ se escogieron de manera arbitraria, podemos concluir que es posible recuperar de manera perfecta el el conjunto fundamental de M si y sólo si $\forall\, \alpha, \beta \in \{1, 2, .., m\}$ y $\alpha \neq \beta$, la proposición $\leq \mathbf{x}^\beta$ es falsa. ■

**Corolario 4.2.1** *Sea* M *una Lernmatrix y sea* $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu \in \{1, 2, .., m\}\}$ *conjunto fundamental de* M, *con* $\mathbf{x}^\mu \neq 0 \; \forall \mu \in \{1, 2, .., p\}$. *Si* $\exists\, \alpha \in \{1, 2, .., p\}$ *de modo que* $\mathbf{x}^\alpha \in \{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu \in \{1, 2, .., m\}\}$ *con* $x_i^\alpha = 1 \; \forall i \in \{1, 2, .., n\}$, *entonces no es posible recuperar de manera perfecta el conjunto fundamental de* M.

**Demostración.** Dado que $\mathbf{x}^\alpha$ es un patrón del conjunto fundamental M cuyas entradas son sólo unos, es decir $x_i^\alpha = 1 \; \forall i \in \{1, 2, .., n\}$, entonces $\mathbf{x}^\mu \leq \mathbf{x}^\alpha$, $\forall \mu \in \{1, 2, .., p\}$ y $\mu \neq \alpha$, de acuerdo con la definición 4.2. Dado que

existen $\alpha, \mu \in \{1, 2, .., p\}$ tales que $\mu \neq \alpha$ y $\mathbf{x}^{\mu} \leq \mathbf{x}^{\alpha}$, al aplicar el teorema 4.2 se demuestra de manera inmediata que no es posible recuperar de manera perfecta todo el conjunto fundamental de la Lernmatrix $\mathbf{M}$. ■

# 5. Conclusiones y trabajo futuro.

En el presente trabajo se ha planteado una forma alternativa de representar la Lernmatrix de Steinbuch, así como las demostraciones de un lema, un teorema y un corolario que describen las condiciones necesarias y suficientes para que la Lernmatrix de Steinbuch recupere de manera perfecta todo su conjunto fundamental.

Es preciso enfatizar que este tipo de estudio se ha realizado en este artículo por primera vez, desde la aparición de la *Lernmatrix* en 1961.

La relevancia de los resultados presentados en este artículo nos permite afirmar que este trabajo abre caminos claros para quienes se interesen en realizar investigaciones futuras sobre el tema, a saber:

- investigar algunas otras propiedades que pueda exhibir esta memoria asociativa
- las condiciones de respuesta ante ruido aditivo y sustractivo
- las condiciones bajo las cuales se da la saturación al tener varios patrones que formen parte de una misma clase
- la posible creación de una nueva versión de la Lernmatrix, élla que no sólo trabaje sobre patrones binarios, sino en el dominio de los números enteros, o más aún, en los reales.

Los resultados de estas investigaciones podrían dar paso a la creación de nuevas memorias asociativas que podrían facilitar la resolución de algunos tipos de problemas, principalmente en el área de la clasificación de patrones.

# 6. Agradecimientos.

Los autores agradecen el apoyo que recibieron de las siguientes instituciones, para la realización de este trabajo: Instituto Politécnico Nacional, COFAA y Secretaría Académica del IPN, CONACyT y Sistema Nacional de Investigadores.

# Referencias

[1] Amari, S. (1977). Neural theory of association and concept-formation, *Biological Cybernetics, 26*, 175-185.

[2] Anderson, J. A. & Rosenfeld, E. (Eds.) (1990). *Neurocomputing: Foundations of Research*, Cambridge: MIT Press.

[3] Anderson, J. R. & Bower, G. (1977). *Memoria Asociativa*, México: Limusa.

[4] Bosch, H. & Kurfess, F. J. (1998). Information storage capacity of incompletely connected associative memories, *Neural Networks (11)*, 5, 869-876.

[5] Díaz-de-León, J. L. & Yáñez, C. (1999). Memorias asociativas con respuesta perfecta y capacidad infinita, *Memoria del TAINA'99, México, D.F.*, 23-38.

[6] Hassoun, M. H. (Ed.) (1993). *Associative Neural Memories*, New York: Oxford University Press.

[7] Kohonen, T. (1972). Correlation matrix memories, *IEEE Transactions Computers, C-21, 4*, 353-359.

[8] Palm, G., Schwenker, F., Sommer F. T. & Strey, A. (1997). Neural associative memories, In A. Krikelis & C. C. Weems (Eds.), *Associative Processing and Processors*, (pp. 307-326). Los Alamitos: IEEE Computer Society.

[9] Simpson, P. K. (1990). *Artificial Neural Systems*, New York: Pergamon Press.

[10] Steinbuch, K. (1961). Die Lernmatrix, *Kybernetik, 1, 1*, 36-45.

[11] Steinbuch, K. & Frank, H. (1961). Nichtdigitale Lernmatrizen als Perzeptoren, *Kybernetik, 1, 3*, 117-124.

[12] Yáñez-Márquez , C. (2002). *Memorias Asociativas basadas en Relaciones de Orden y Operadores Binarios*. Tesis doctoral, CIC-IPN, México.

[13] Yáñez-Márquez, C. & Díaz-de-León Santiago, J.L. (2001). *Lernmatrix Steinbuch*, IT-48, Serie Verde, ISBN 970-18-6688-6, CIC-IPN, México.

# Clasificador híbrido de patrones basado en la *Lernmatrix* de Steinbuch y el *Linear Associator* de Anderson-Kohonen

R. Santiago-Montero, J. L. Díaz-de-León-S. & C. Yáñez-Márquez
Centro de Investigación en Computación
del Instituto Politécnico Nacional

**Resumen**

En este trabajo se presenta un nuevo clasificador basado en los principios de álgebra de matrices que utiliza el *Linear Associator* y el criterio de clasificación de la *Lernmatrix* de Steinbuch, creándose así un método híbrido que tiene la ventaja de que puede trabajar patrones de entrada con valores reales, y además no requiere de la restricción de ortonormalidad, impuesta por el *Linear Associator*.

## 1. Introducción.

Las memorias asociativas han merecido la atención de numerosos investigadores internacionales desde hace más de cuatro décadas.

Uno de los pioneros en esta área de investigación fue el científico alemán Karl Steinbuch quien, a principios de la década de los sesenta, ideó, desarrolló y aplicó la *Lernmatrix* (Steinbuch, 1961; Steinbuch & Frank, 1961). La *Lernmatrix* constituye un antecedente crucial en el desarrollo de los modelos actuales de memorias asociativas, y es reconocido como uno de los primeros intentos exitosos de codificar información en arreglos cuadriculados conocidos como *crossbar* (Simpson, 1990).

Once años después de que Steinbuch dio a conocer *Lernmatrix*, dos investigadores concluyeron y presentaron ante al comunidad científica internacional sus trabajos de investigación. Apoyado por la *UCLA*, a principios de 1972 James A. Anderson desarrolló y presentó su *Interactive Memory* (Anderson, 1972), y meses más tarde, Teuvo Kohonen, a la sazón profesor de la *Helsinki University of Technology*, dio a conocer ante el mundo sus *Correlation Matrix Memories* (Kohonen, 1972).

Los trabajos de Anderson y Kohonen dieron lugar al modelo que actualmente se conoce con el nombre genérico de *Linear Associator*.

Es pertinente mencionar un hecho curioso, que se ha presentado en personajes dedicados a otras ramas de la ciencia: James A. Anderson y Teuvo Kohonen

obtuvieron resultados *asombrosamente similares* a pesar de que trabajaron independientemente, alejados, y sin tener noticia uno del otro, hasta tiempo después de que aparecieron los artículos; además, estos autores tienen formaciones profesionales totalmente diferentes: Anderson es neurofisiólogo (estadunidense) y Kohonen es físico e ingeniero eléctrico (finlandés) (Anderson & Rosenfeld, 1990; Kohonen, 1989).

Cada uno de los modelos mencionados tiene ventajas y desventajas. En este artículo se presenta un modelo híbrido de memoria asociativa que trabaja eficientemente como un clasificador de patrones racionales, y en cuyo diseño se han tomado las ventajas de ambos modelos evitando, al mismo tiempo, las desventajas. Los resultados de este trabajo ilustran la importancia potencial de las memorias asociativas en el área de reconocimiento de patrones.

Una memoria asociativa tiene como propósito fundamental: recuperar correctamente patrones completos a partir de patrones de entrada, los cuales pueden estar alterados con ruido aditivo, sustractivo o combinado.

El problema inherente al funcionamiento de las memorias asociativas se escinde en dos fases claramente distinguibles:

1. Fase de *aprendizaje* (generación de la memoria asociativa)
2. Fase de *recuperación* (operación de la memoria asociativa)

En ambas fases, una *memoria asociativa* M puede formularse como un sistema de entrada y salida, idea que se esquematiza a continuación:

$$\mathbf{x} \longrightarrow \boxed{\mathbf{M}} \longrightarrow \mathbf{y}$$

El *patrón de entrada* está representado por un vector columna denotado por $\mathbf{x}$ y el *patrón de salida*, por el vector columna denotado por $\mathbf{y}$. Cada uno los patrones de entrada forma una *asociación* con el correspondiente patrón salida. La notación para una asociación es similar a la de una pareja ordenada; por ejemplo, los patrones $\mathbf{x}$ y $\mathbf{y}$ del esquema forman la asociación $(\mathbf{x}, \mathbf{y})$.

Dado un número entero positivo $k$ específico, la asociación correspondiente será $(\mathbf{x}^k, \mathbf{y}^k)$.

La memoria asociativa M se representa mediante una matriz cuya componente $ij$-ésima es $m_{ij}$ (Palm, Schwenker, Sommer & Strey, 1997); la matriz se genera a partir de un conjunto finito de asociaciones conocidas de antemano: éste es el *conjunto fundamental de asociaciones*, o simplemente *conjunto fundamental*. Se denota por $p$ la cardinalidad del conjunto fundamental ($p$ un número entero positivo).

Si $\mu$ es un índice, el conjunto fundamental se representa de la siguiente manera:

$$\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu = 1, 2, ..., p\}$$

**Definición 1.1** *A los patrones que conforman las asociaciones del conjunto fundamental, se les llama patrones fundamentales.*

La naturaleza del conjunto fundamental proporciona un importante criterio para clasificar las memorias asociativas (Kohonen,1972).

**Definición 1.2** *Si se cumple que* $\mathbf{x}^\mu = \mathbf{y}^\mu \; \forall \mu \in \{1,2,...,p\}$, *se dice que la memoria es autoasociativa; de otro modo, la memoria es heteroasociativa. Para una memoria heteroasociativa se puede afirmar lo siguiente:* $\exists \mu \in \{1,2,...,p\}$ *para el que se cumple que* $\mathbf{x}^\mu \neq \mathbf{y}^\mu$.

**Definición 1.3** *Si al presentarle a la memoria* M *un patrón alterado* $\tilde{\mathbf{x}}^\omega$ *como entrada* ($\omega \in \{1,2,...,p\}$), M *responde con el correspondiente patrón fundamental de salida* $\mathbf{y}^\omega$, *se dice que la recuperación es perfecta.*

**Definición 1.4** *Una memoria perfecta es aquella que realiza recuperaciones perfectas para todos los patrones fundamentales.*

$A$ y $B$ son dos conjuntos que cumplen con lo siguiente: las componentes de los vectores columna que representan a los patrones, tanto de entrada como de salida, son elementos del conjunto $A$, y las entradas de la matriz M son elementos del conjunto $B$.

No hay requisitos previos ni limitaciones respecto de la elección de estos dos conjuntos, por lo que no necesariamente deben ser diferentes o poseer características especiales. Sean $m,n$ números enteros positivos. Se denota por $n$ la dimensión de los patrones de entrada, y por $m$ la dimensión de los patrones de salida; claramente, nada impide que los valores de $m$ y de $n$ sean iguales. Aún más, uno de los requisitos que debe cumplir una memoria autoasociativa es que la dimensión de los patrones de entrada sea igual a la dimensión de los patrones de salida; por otro lado, si en una memoria sucede que $m \neq n$, es evidente que la memoria debe ser heteroasociativa.

Cada vector columna que representa a un patrón de entrada tiene $n$ componentes cuyos valores pertenecen al conjunto $A$, y cada vector columna que representa a un patrón de salida posee $m$ componentes cuyos valores pertenecen al conjunto $A$. Es decir:

$$\mathbf{x}^\mu \in A^n \text{ y } \mathbf{y}^\mu \in A^m \; \forall \mu \in \{1,2,...,p\}$$

La $j$-ésima componente de un vector columna se indica con la misma letra del vector, pero sin negrilla, colocando a $j$ como subíndice ($j \in \{1,2,...,n\}$ o $j \in \{1,2,...,m\}$ según corresponda). La $j$-ésima componente de un vector columna $\mathbf{x}^\mu$ se representa por

$$x_j^\mu$$

Los vectores columna que representan a los patrones fundamentales de entrada y de salida son, respectivamente:

$$\mathbf{x}^\mu = \begin{pmatrix} x_1^\mu \\ x_2^\mu \\ \vdots \\ x_n^\mu \end{pmatrix} \in A^n \qquad \mathbf{y}^\mu = \begin{pmatrix} y_1^\mu \\ y_2^\mu \\ \vdots \\ y_m^\mu \end{pmatrix} \in A^m$$

## 2. El trabajo de Steinbuch.

La *Lernmatrix* es una memoria heteroasociativa que puede funcionar como un clasificador de patrones binarios si se escogen adecuadamente los patrones de salida; es un sistema de entrada y salida que al operar acepta como entrada un patrón binario $\mathbf{x}^\mu \in A^n$, $A = \{0, 1\}$ y produce como salida la clase $\mathbf{y}^\mu \in A^m$ que le corresponde (de entre $m$ clases diferentes), codificada ésta con un método simple, a saber: para representar la clase $k \in \{1, 2, ..., m\}$, se asignan a las componentes del vector de salida $\mathbf{y}^\mu$ los siguientes valores: $y_k^\mu = 1$, y $y_j^\mu = 0$ para $j = 1, 2..., k-1, k+1, ...m$

En la tabla se esquematiza la fase de aprendizaje para la *Lernmatrix* de Steinbuch, con la pareja de patrones fundamentales $(\mathbf{x}^\mu, \mathbf{y}^\mu) \in A^n \times A^m$.

| | $x_1^\mu$ | $x_2^\mu$ | $\cdots$ | $x_j^\mu$ | $\cdots$ | $x_n^\mu$ |
|---|---|---|---|---|---|---|
| $y_1^\mu$ | $m_{11}$ | $m_{12}$ | $\cdots$ | $m_{1j}$ | $\cdots$ | $m_{1n}$ |
| $y_2^\mu$ | $m_{21}$ | $m_{22}$ | $\cdots$ | $m_{2j}$ | $\cdots$ | $m_{2n}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | | $\vdots$ | | $\vdots$ |
| $y_i^\mu$ | $m_{i1}$ | $m_{i2}$ | $\cdots$ | $m_{ij}$ | $\cdots$ | $m_{in}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | | $\vdots$ | | $\vdots$ |
| $y_m^\mu$ | $m_{m1}$ | $m_{m2}$ | $\cdots$ | $m_{mj}$ | $\cdots$ | $m_{mn}$ |

(1)

Cada uno de los componentes $m_{ij}$ de M, la *Lernmatrix* de Steinbuch, tiene valor cero al inicio, y se actualiza de acuerdo con la regla $m_{ij} + \Delta m_{ij}$, donde:

$$\Delta m_{ij} = \begin{cases} +\varepsilon \text{ si } y_i^\mu = 1 = x_j^\mu \\ -\varepsilon \text{ si } y_i^\mu = 1 \text{ y } x_j^\mu = 0 \\ 0 \text{ en otro caso} \end{cases} \tag{2}$$

siendo $\varepsilon$ una constante positiva escogida previamente.

La fase de recuperación consiste en encontrar la clase a la que pertenece un vector de entrada $\mathbf{x}^\omega \in A^n$ dado. Encontrar la clase significa obtener las coordenadas del vector $\mathbf{y}^\omega \in A^m$ que le corresponde al patrón $\mathbf{x}^\omega$; en virtud del método de construcción de los vectores $\mathbf{y}^\mu$ la clase debería obtenerse sin ambigüedad.

La $i$-ésima coordenada $y_i^\omega$ del vector de clase $\mathbf{y}^\omega \in A^m$ se obtiene como lo indica la siguiente expresión, donde $\bigvee$ es el operador *máximo*:

$$y_i^\omega = \begin{cases} 1 \text{ si } \sum_{j=1}^n m_{ij}.x_j^\omega = \bigvee_{h=1}^m \left[\sum_{j=1}^n m_{hj}.x_j^\omega\right] \\ 0 \text{ en otro caso} \end{cases} \tag{3}$$

**Ejemplo 2.1** *3 patrones de dimensión 5, donde cada uno se asigna a una clase diferente:*

$$\mathbf{x}^1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \mathbf{y}^1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}; \quad \mathbf{x}^2 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \mathbf{y}^2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix};$$

$$\mathbf{x}^3 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad \mathbf{y}^3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

La fase de aprendizaje se inicia asignando el valor cero a todos los elementos $m_{ij}$ y a continuación se realizan las operaciones de la crossbar 1 y la expresión 2 con las tres asociaciones, para obtener la *Lernmatrix*:

$$\mathbf{M} = \begin{pmatrix} \varepsilon & -\varepsilon & \varepsilon & -\varepsilon & \varepsilon \\ \varepsilon & \varepsilon & -\varepsilon & -\varepsilon & \varepsilon \\ \varepsilon & -\varepsilon & \varepsilon & \varepsilon & -\varepsilon \end{pmatrix} \tag{4}$$

La fase de recuperación consiste en presentarle a la matriz M uno de los patrones de entrada y realizar las operaciones indicadas en la expresión 3; a la salida se espera obtener la clase a la que pertenece el vector de entrada.

$$\mathbf{M} \cdot \mathbf{x}^1 = \begin{pmatrix} \varepsilon & -\varepsilon & \varepsilon & -\varepsilon & \varepsilon \\ \varepsilon & \varepsilon & -\varepsilon & -\varepsilon & \varepsilon \\ \varepsilon & -\varepsilon & \varepsilon & \varepsilon & -\varepsilon \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 3\varepsilon \\ \varepsilon \\ \varepsilon \end{pmatrix}$$

Se observa que: $\sum_{j=1}^{5} m_{1j}x_j^1 = 3\varepsilon$; $\sum_{j=1}^{5} m_{2j}x_j^1 = \varepsilon$ y $\sum_{j=1}^{5} m_{3j}x_j^1 = \varepsilon$. Por ello, $\sum_{j=1}^{5} m_{1j}x_j^1 = \bigvee_{h=1}^{p} \left[\sum_{j=1}^{5} m_{hj}x_j^1\right]$ y de acuerdo con la expresión 3, se tiene que $y_1^1 = 1$ y $y_2^1 = 0 = y_3^1$. Por lo tanto, el vector que representa a la clase es $\mathbf{y}^1$.

De igual manera se recuperan las clases a la que pertenecen los patrones de entrada $\mathbf{x}^2$.y $\mathbf{x}^3$:

$$\mathbf{M} \cdot \mathbf{x}^2 = \begin{pmatrix} \varepsilon & -\varepsilon & \varepsilon & -\varepsilon & \varepsilon \\ \varepsilon & \varepsilon & -\varepsilon & -\varepsilon & \varepsilon \\ \varepsilon & -\varepsilon & \varepsilon & \varepsilon & -\varepsilon \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \varepsilon \\ 3\varepsilon \\ -\varepsilon \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \mathbf{y}^2$$

$$\mathbf{M} \cdot \mathbf{x}^3 = \begin{pmatrix} \varepsilon & -\varepsilon & \varepsilon & -\varepsilon & \varepsilon \\ \varepsilon & \varepsilon & -\varepsilon & -\varepsilon & \varepsilon \\ \varepsilon & -\varepsilon & \varepsilon & \varepsilon & -\varepsilon \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \varepsilon \\ -\varepsilon \\ 3\varepsilon \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \mathbf{y}^3$$

**Ejemplo 2.2** *Agreguemos las siguientes dos asociaciones:*

$$x^4 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} \; y^1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}; \quad x^5 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \quad y^3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

La nueva *Lernmatrix* es:

$$M = \begin{pmatrix} 0 & 0 & 0 & 0 & 2\varepsilon \\ \varepsilon & \varepsilon & -\varepsilon & -\varepsilon & \varepsilon \\ 0 & -2\varepsilon & 2\varepsilon & 0 & 0 \end{pmatrix}$$

Al intentar recuperar la clase para cada uno de los cinco patrones de entrada, se observa cómo la *Lernmatrix* llega a la **saturación**:

$$M \cdot x^1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 2\varepsilon \\ \varepsilon & \varepsilon & -\varepsilon & -\varepsilon & \varepsilon \\ 0 & -2\varepsilon & 2\varepsilon & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \text{ ¿clase } y^1 \text{ o } y^3?$$

$$M \cdot x^2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 2\varepsilon \\ \varepsilon & \varepsilon & -\varepsilon & -\varepsilon & \varepsilon \\ 0 & -2\varepsilon & 2\varepsilon & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \text{ clase } y^2$$

$$M \cdot x^3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 2\varepsilon \\ \varepsilon & \varepsilon & -\varepsilon & -\varepsilon & \varepsilon \\ 0 & -2\varepsilon & 2\varepsilon & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \text{ clase } y^3$$

$$M \cdot x^4 = \begin{pmatrix} 0 & 0 & 0 & 0 & 2\varepsilon \\ \varepsilon & \varepsilon & -\varepsilon & -\varepsilon & \varepsilon \\ 0 & -2\varepsilon & 2\varepsilon & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \text{ clase } y^1$$

$$M \cdot x^5 = \begin{pmatrix} 0 & 0 & 0 & 0 & 2\varepsilon \\ \varepsilon & \varepsilon & -\varepsilon & -\varepsilon & \varepsilon \\ 0 & -2\varepsilon & 2\varepsilon & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \text{ ¿clase } y^1 \text{ o } y^3?$$

## 3. El trabajo de Anderson y Kohonen.

Para presentar el *Linear Associator*, consideremos el conjunto fundamental $\{(x^\mu, y^\mu) \mid \mu = 1, 2, ..., p\}$ con

$$\mathbf{x}^{\mu} = \begin{pmatrix} x_1^{\mu} \\ x_2^{\mu} \\ \vdots \\ x_n^{\mu} \end{pmatrix} \in A^n \quad \mathbf{y} \quad \mathbf{y}^{\mu} = \begin{pmatrix} y_1^{\mu} \\ y_2^{\mu} \\ \vdots \\ y_m^{\mu} \end{pmatrix} \in A^m$$

La *fase de aprendizaje* consiste de dos etapas:

Etapa 1.- Para cada una de las $p$ asociaciones $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu})$ se encuentra la matriz $\mathbf{y}^{\mu} \cdot (\mathbf{x}^{\mu})^t$ de dimensiones $m \times n$

$$\mathbf{y}^{\mu} \cdot (\mathbf{x}^{\mu})^t = \begin{pmatrix} y_1^{\mu} \\ y_2^{\mu} \\ \vdots \\ y_m^{\mu} \end{pmatrix} \cdot (x_1^{\mu}, x_2^{\mu}, ..., x_n^{\mu}) = \begin{pmatrix} y_1^{\mu}x_1^{\mu} & y_1^{\mu}x_2^{\mu} & \cdots & y_1^{\mu}x_j^{\mu} & \cdots & y_1^{\mu}x_n^{\mu} \\ y_2^{\mu}x_1^{\mu} & y_2^{\mu}x_2^{\mu} & \cdots & y_2^{\mu}x_j^{\mu} & \cdots & y_2^{\mu}x_n^{\mu} \\ \vdots & \vdots & & \vdots & & \vdots \\ y_i^{\mu}x_1^{\mu} & y_i^{\mu}x_2^{\mu} & \cdots & y_i^{\mu}x_j^{\mu} & \cdots & y_i^{\mu}x_n^{\mu} \\ \vdots & \vdots & & \vdots & & \vdots \\ y_m^{\mu}x_1^{\mu} & y_m^{\mu}x_2^{\mu} & \cdots & y_m^{\mu}x_j^{\mu} & \cdots & y_m^{\mu}x_n^{\mu} \end{pmatrix} \tag{5}$$

Etapa 2.- Se suman la $p$ matrices para obtener la memoria

$$\mathbf{M} = \sum_{\mu=1}^{p} \mathbf{y}^{\mu} \cdot (\mathbf{x}^{\mu})^t = [m_{ij}]_{m \times n} \tag{6}$$

de manera que la $ij$-ésima componente de la memoria $\mathbf{M}$ se expresa así:

$$m_{ij} = \sum_{\mu=1}^{p} y_i^{\mu} x_j^{\mu} \tag{7}$$

La *fase de recuperación* consiste en presentarle a la memoria un patrón de entrada $\mathbf{x}^{\omega}$, donde $\omega \in \{1, 2, ..., p\}$ y realizar la operación

$$\mathbf{M} \cdot \mathbf{x}^{\omega} = \left[ \sum_{\mu=1}^{p} \mathbf{y}^{\mu} \cdot (\mathbf{x}^{\mu})^t \right] \cdot \mathbf{x}^{\omega} \tag{8}$$

Al desarrollar la sumatoria de la expresión 8, se tiene:

$$\begin{aligned} \mathbf{M} \cdot \mathbf{x}^{\omega} &= \left[ \mathbf{y}^1 \cdot (\mathbf{x}^1)^t + \mathbf{y}^2 \cdot (\mathbf{x}^2)^t + \cdots + \mathbf{y}^{\omega} \cdot (\mathbf{x}^{\omega})^t + \cdots + \mathbf{y}^p \cdot (\mathbf{x}^p)^t \right] \cdot \mathbf{x}^{\omega} \\ \mathbf{M} \cdot \mathbf{x}^{\omega} &= \left[ \mathbf{y}^1 \cdot (\mathbf{x}^1)^t \right] \cdot \mathbf{x}^{\omega} + \left[ \mathbf{y}^2 \cdot (\mathbf{x}^2)^t \right] \cdot \mathbf{x}^{\omega} + \cdots + \left[ \mathbf{y}^{\omega} \cdot (\mathbf{x}^{\omega})^t \right] \cdot \mathbf{x}^{\omega} + \cdots + \left[ \mathbf{y}^p \cdot (\mathbf{x}^p)^t \right] \\ \mathbf{M} \cdot \mathbf{x}^{\omega} &= \mathbf{y}^1 \cdot \left[ (\mathbf{x}^1)^t \cdot \mathbf{x}^{\omega} \right] + \mathbf{y}^2 \cdot \left[ (\mathbf{x}^2)^t \cdot \mathbf{x}^{\omega} \right] + \cdots + \mathbf{y}^{\omega} \cdot \left[ (\mathbf{x}^{\omega})^t \cdot \mathbf{x}^{\omega} \right] + \cdots + \mathbf{y}^p \cdot \left[ (\mathbf{x}^p)^t \right. \end{aligned}$$

Y esto último se puede escribir de la siguiente manera, usando de nuevo la notación sumatoria:

$$\mathbf{M} \cdot \mathbf{x}^{\omega} = \mathbf{y}^{\omega} \cdot \left[ (\mathbf{x}^{\omega})^{t} \cdot \mathbf{x}^{\omega} \right] + \sum_{\mu \neq \omega} \mathbf{y}^{\mu} \cdot \left[ (\mathbf{x}^{\mu})^{t} \cdot \mathbf{x}^{\omega} \right] \tag{9}$$

La forma de la expresión 9 nos permite investigar las condiciones que se deben cumplir para que el método de recuperación propuesto dé como resultado salidas perfectas. Para que la expresión anterior arroje como resultado al patrón $\mathbf{y}^{\omega}$, es preciso que se cumplan dos igualdades:

a) $\left[ (\mathbf{x}^{\omega})^{t} \cdot \mathbf{x}^{\omega} \right] = 1$

b) $\left[ (\mathbf{x}^{\mu})^{t} \cdot \mathbf{x}^{\omega} \right] = 0$ siempre que $\mu \neq \omega$

Dado que $\omega$ se escogió arbitrariamente, las dos igualdades se deben cumplir $\forall \omega \in \{1, 2, ..., p\}$, lo cual indica que los vectores de entrada $\mathbf{x}^{\mu}$ deben ser ortonormales. Esta condición de ortonormalidad se puede resumir en la siguiente expresión:

$$(\mathbf{x}^{\mu})^{t} \cdot \mathbf{x}^{\omega} = \begin{cases} 1 \text{ si } \mu = \omega \\ \\ 0 \text{ si } \mu \neq \omega \end{cases} \tag{10}$$

Si se cumple la condición que se manifiesta en la expresión 10, entonces la recuperación es perfecta; es decir, la expresión 9 toma la forma $\mathbf{M} \cdot \mathbf{x}^{\omega} = \mathbf{y}^{\omega}$.

Sin embargo, si los vectores de entrada no son ortonormales, suceden dos cosas:

· el factor $\left[ (\mathbf{x}^{\omega})^{t} \cdot \mathbf{x}^{\omega} \right]$ no es 1

· el término $\sum_{\mu \neq \omega} \mathbf{y}^{\mu} \cdot \left[ (\mathbf{x}^{\mu})^{t} \cdot \mathbf{x}^{\omega} \right]$ no es 0

Este último término, llamado *cross-talk*, representa el ruido producido por la interacción entre los patrones de entrada, y tiene como consecuencia inmediata que la recuperación no es perfecta, excepto si el número de patrones almacenados es pequeño comparado con la dimensión $n$ de los vectores de entrada. Algunos investigadores afirman que ese número pequeño de patrones debe estar entre $0{,}1n$ y $0{,}2n$ (Anderson & Rosenfeld, 1990; Hassoun, 1995; Ritter, Sussner & Díaz-de-León, 1998).

**Ejemplo 3.1** *Cuatro parejas de patrones ($p = 4$), con $n = 3$ y $m = 5$.*

$$\mathbf{x}^1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \mathbf{y}^1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}; \quad \mathbf{x}^2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \mathbf{y}^2 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix};$$

$$\mathbf{x}^3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \mathbf{y}^3 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}; \quad \mathbf{x}^4 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \quad \mathbf{y}^4 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

Calculemos los términos $\mathbf{y}^{\mu} \cdot (\mathbf{x}^{\mu})^t$ usando la expresión 5:

$$\mathbf{y}^1 \cdot (\mathbf{x}^1)^t = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}; \quad \mathbf{y}^2 \cdot (\mathbf{x}^2)^t = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\mathbf{y}^3 \cdot (\mathbf{x}^3)^t = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}; \quad \mathbf{y}^4 \cdot (\mathbf{x}^4)^t = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

Se obtiene la memoria M a partir de las expresiones 6 y 7:

$$\mathbf{M} = \sum_{\mu=1}^{4} \mathbf{y}^{\mu} \cdot (\mathbf{x}^{\mu})^t = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 2 \\ 2 & 1 & 1 \end{pmatrix} \tag{11}$$

Se usa la expresión 8 y la memoria 11 para recuperación:

$$\mathbf{M} \cdot \mathbf{x}^1 = \left[ \sum_{\mu=1}^{4} \mathbf{y}^{\mu} \cdot (\mathbf{x}^{\mu})^t \right] \cdot \mathbf{x}^1 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 2 \\ 2 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 2 \end{pmatrix} \neq \mathbf{y}^1$$

Análogamente con los otros tres patrones:

$$\mathbf{M} \cdot \mathbf{x}^2 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \mathbf{y}^2; \quad \mathbf{M} \cdot \mathbf{x}^3 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 2 \\ 1 \end{pmatrix} \neq \mathbf{y}^3 \quad \mathbf{M} \cdot \mathbf{x}^4 = \begin{pmatrix} 2 \\ 2 \\ 2 \\ 3 \\ 3 \end{pmatrix} \neq \mathbf{y}^4$$

# 4. El clasificador híbrido de patrones

Los ejemplos 2.1 y 3.1 ilustran el funcionamiento y las desventajas de los dos modelos de memorias asociativas, la *Lernmatrix* y el *Linear Associator*.

Por un lado, la *Lernmatrix* puede aceptar sólo patrones binarios como entradas, y el ejemplo 2.1 muestra cómo se llega rápidamente a la saturación, fenómeno que impide la clasificación correcta. Por otro lado, no obstante que el *Linear Associator* elimina la restricción de patrones binarios a la entrada,

puesto que puede aceptar patrones con valores reales en sus componentes, surge una restricción bastante fuerte: la ortonormalidad de los patrones de entrada.

En el ejemplo 3.1 sólo se recuperó un patrón de salida de manera perfecta, debido a los devastadores efectos que tiene, en la fase de recuperación, el término de *cross-talk* diferente de cero, en virtud de que los patrones de entrada no son ortonormales, como lo exige la expresión 10.

El clasificador híbrido que se propone en este trabajo elimina las desventajas de ambos modelos, pero incluye algunos principios de diseño y operación de éstos. El algoritmo del nuevo clasificador es como sigue:

1. Sea un conjunto fundamental de patrones de entrada de dimensión $n$ con valores reales en sus componentes (a la manera del *Linear Associator*), que se aglutinan en $m$ clases diferentes.

2. A cada uno de los patrones de entrada que pertenece a la clase $k$ se le asigna el vector formado por ceros, excepto en la coordenada $k$-ésima, donde el valor es uno (a la manera de la *Lernmatrix*).

3. La *fase de aprendizaje* es similar a la del *Linear Associator*, de acuerdo con las expresiones 5, 6 y 7.

4. La fase de recuperación es similar a la que usa la *Lernmatrix*, de acuerdo con la expresión 3

Este algoritmo simple nos permite diseñar un poderoso clasificador de patrones, como se ilustra en el siguiente ejemplo:

**Ejemplo 4.1** *Cinco patrones de dimensión 2 con valores reales, aglutinados en dos clases diferentes:*

A la primera clase pertenecen dos patrones:

$$\mathbf{x}^1 = \begin{pmatrix} -4{,}1 \\ 3{,}8 \end{pmatrix} \quad \text{y} \quad \mathbf{x}^2 = \begin{pmatrix} -4{,}8 \\ 4{,}2 \end{pmatrix}$$

Tres patrones pertenecen a la segunda clase:

$$\mathbf{x}^3 = \begin{pmatrix} -6{,}3 \\ -3{,}8 \end{pmatrix}, \mathbf{x}^3 = \begin{pmatrix} -6{,}2 \\ -3{,}1 \end{pmatrix} \quad \text{y} \quad \mathbf{x}^4 = \begin{pmatrix} -7{,}0 \\ -3{,}0 \end{pmatrix}$$

Lo anterior significa, de acuerdo con el inciso 2 del algoritmo, que los patrones de salida son los siguientes:

$$\mathbf{y}^1 = \mathbf{y}^2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{y} \quad \mathbf{y}^3 = \mathbf{y}^4 = \mathbf{y}^5 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Para la fase de aprendizaje, de acuerdo con el inciso 3 del algoritmo, se calculan los términos $\mathbf{y}^\mu \cdot (\mathbf{x}^\mu)^t$ usando la expresión 5, y el clasificador $C$ se obtiene a partir de las expresiones 6 y 7:

$$C = \sum_{\mu=1}^{5} y^{\mu} \cdot (x^{\mu})^{t} = \begin{pmatrix} -8{,}9 & 8{,}0 \\ -19{,}5 & -9{,}9 \end{pmatrix} \tag{12}$$

El inciso 4 del algoritmo indica que la fase de recuperación se lleva acabo de de acuerdo con la expresión 3.

$$\begin{pmatrix} -8{,}9 & 8{,}0 \\ -19{,}5 & -9{,}9 \end{pmatrix} \cdot \begin{pmatrix} -4{,}1 \\ 3{,}8 \end{pmatrix} = \begin{pmatrix} 66{,}89 \\ 42{,}33 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix} \rightarrow \text{clase 1}$$

$$\begin{pmatrix} -8{,}9 & 8{,}0 \\ -19{,}5 & -9{,}9 \end{pmatrix} \cdot \begin{pmatrix} -4{,}8 \\ 4{,}2 \end{pmatrix} = \begin{pmatrix} 76{,}32 \\ 52{,}02 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix} \rightarrow \text{clase 1}$$

$$\begin{pmatrix} -8{,}9 & 8{,}0 \\ -19{,}5 & -9{,}9 \end{pmatrix} \cdot \begin{pmatrix} -6{,}3 \\ -3{,}8 \end{pmatrix} = \begin{pmatrix} 25{,}67 \\ 160{,}47 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix} \rightarrow \text{clase 2}$$

$$\begin{pmatrix} -8{,}9 & 8{,}0 \\ -19{,}5 & -9{,}9 \end{pmatrix} \cdot \begin{pmatrix} -6{,}2 \\ -3{,}1 \end{pmatrix} = \begin{pmatrix} 30{,}38 \\ 151{,}59 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix} \rightarrow \text{clase 2}$$

$$\begin{pmatrix} -8{,}9 & 8{,}0 \\ -19{,}5 & -9{,}9 \end{pmatrix} \cdot \begin{pmatrix} -7{,}0 \\ -3{,}0 \end{pmatrix} = \begin{pmatrix} 38{,}3 \\ 166{,}2 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix} \rightarrow \text{clase 2}$$

**Ejemplo 4.2** *Ahora probemos el clasificador con dos patrones que no pertencen al conjunto fundamental:*

$$\begin{pmatrix} -8{,}9 & 8{,}0 \\ -19{,}5 & -9{,}9 \end{pmatrix} \cdot \begin{pmatrix} -4{,}0 \\ 6{,}0 \end{pmatrix} = \begin{pmatrix} 83{,}6 \\ 18{,}6 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix} \rightarrow \text{clase 1}$$

$$\begin{pmatrix} -8{,}9 & 8{,}0 \\ -19{,}5 & -9{,}9 \end{pmatrix} \cdot \begin{pmatrix} -6{,}0 \\ -4{,}0 \end{pmatrix} = \begin{pmatrix} 21{,}2 \\ 117{,}6 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix} \rightarrow \text{clase 2}$$

**Nota 1** *Resulta evidente que este problema de clasificación no puede ser resuelto por la Lernmatrix de Steinbuch, ya que los patrones de entrada no son binarios, como lo requiere ese modelo. Por otro lado, tampoco es posible resolverlo por el Linear Associator, debido a que los patrones de entrada no son ortonormales, como lo exige la expresión 10.*

# 5. Conclusiones y trabajo futuro.

En este trabajo se ha presentado un nuevo clasificador el cual utiliza como base dos modelos conocidos de memorias asociativas: la *Lernmatrix* de Steinbuch y el *Linear Associator*.

El algoritmo del nuevo clasificador recoge lo mejor de ambos modelos y elimina sus desventajas, y con ello se ha logrado crear un método híbrido que tiene la ventaja de que puede trabajar patrones de entrada con valores reales, y además no requiere de la restricción de ortonormalidad, impuesta por el *Linear Associator*.

El trabajo futuro inmediato es encontrar la base teórica que sustenta el buen funcionamiento del algoritmo propuesto.

## 6. Agradecimientos.

Los autores agradecen el apoyo que recibieron de las siguientes instituciones, para la realización de este trabajo: Instituto Politécnico Nacional, COFAA y Secretaría Académica del IPN, CONACyT y Sistema Nacional de Investigadores.

## Referencias

[1] Amari, S. (1977). Neural theory of association and concept-formation, *Biological Cybernetics, 26*, 175-185.

[2] Anderson, J. A. & Rosenfeld, E. (Eds.) (1990). *Neurocomputing: Foundations of Research*, Cambridge: MIT Press.

[3] Anderson, J. R. & Bower, G. (1977). *Memoria Asociativa*, México: Limusa.

[4] Bosch, H. & Kurfess, F. J. (1998). Information storage capacity of incompletely connected associative memories, *Neural Networks (11), 5*, 869-876.

[5] Díaz-de-León, J. L. & Yáñez, C. (1999). Memorias asociativas con respuesta perfecta y capacidad infinita, *Memoria del TAINA'99, México, D.F.*, 23-38.

[6] Hassoun, M. H. (Ed.) (1993). *Associative Neural Memories*, New York: Oxford University Press.

[7] Kohonen, T. (1972). Correlation matrix memories, *IEEE Transactions on Computers, C-21, 4*, 353-359.

[8] Palm, G., Schwenker, F., Sommer F. T. & Strey, A. (1997). Neural associative memories, In A. Krikelis & C. C. Weems (Eds.), *Associative Processing and Processors*, (pp. 307-326). Los Alamitos: IEEE Computer Society.

[9] Simpson, P. K. (1990). *Artificial Neural Systems*, New York: Pergamon Press.

[10] Steinbuch, K. (1961). Die Lernmatrix, *Kybernetik, 1, 1*, 36-45.

[11] Steinbuch, K. & Frank, H. (1961). Nichtdigitale Lernmatrizen als Perzeptoren, *Kybernetik, 1, 3*, 117-124.

[12] Yáñez-Márquez , C. (2002). *Memorias Asociativas basadas en Relaciones de Orden y Operadores Binarios.* Tesis doctoral, CIC-IPN, México.

[13] Yáñez-Márquez, C. & Díaz-de-León Santiago, J.L. (2001). *Lernmatrix de Steinbuch*, IT-48, Serie Verde, ISBN 970-18-6688-6, CIC-IPN, México.

# Automated Quality Control in Sound Speaker Manufacturing using Soft Computing Techniques and Fractal Theory for Pattern Recognition

Patricia Melin and Oscar Castillo
Department of Computer Science
Tijuana Institute of Technology
Tijuana, Mexico
pmelin@tectijuana.mx ocastillo@tectijuana.mx

**Abstract**

We describe in this paper the application of a hybrid neuro-fuzzy-fractal approach to the problem of automated quality control in sound speaker manufacturing. Traditional quality control has been done by manually checking the quality of sound after production. This manual checking of the speakers is time consuming and occasionally was the cause of error in quality evaluation. For this reason, we developed an intelligent system for automated quality control in sound speaker manufacturing. The intelligent system has a fuzzy rule base containing the knowledge of human experts in quality control. The parameters of the fuzzy system are tuned by applying the ANFIS methodology using, as training data, a real time series of measured sounds as given by good sound speakers. We also use the fractal dimension as a measure of the complexity of the sound signal. The intelligent system has been tested in a real plant with very good results.

**Key words:** Fuzzy Logic, Neural Networks, Neuro-Fuzzy, Manufacturing, Fractal Theory

## 1. Introduction

We describe in this paper the application of a neuro-fuzzy-fractal approach to the problem of quality control in the manufacturing of sound speakers in a real plant. The quality control of the speakers was done before by manually checking the quality of sound achieved after production [4]. A human expert evaluates the quality of sound of the speakers to decide if production quality was achieved. Of course, this manual checking of the speakers is time consuming and occasionally was the cause of error in quality evaluation [7]. For this reason, it was necessary to consider automating the quality control of the sound speakers. The problem of measuring the quality of the sound speakers is as follows:

1) First, we need to extract the real sound signal of the speaker during the testing period after production
2) Second, we need to compare the real sound signal to the desired sound signal of the speaker, and measure the difference in some way
3) Third, we need to decide on the quality of the speaker based on the difference found in step 2. If the difference is small enough then the speaker can be considered of good quality, if not then is bad quality.

The first part of the problem was solved by using a multimedia kit that enable us to extract the sound signal as a file, which basically contains 108000 points over a period of time of 3 seconds (this is the time required for testing). We can say that the sound signal is measured as a time series of data points [3], which has the basic characteristics of the speaker. The second part of the problem was solved by using a neuro-fuzzy approach to train a fuzzy model with the data from the good quality speakers [9]. We used the ANFIS approach [6] to obtain a Sugeno fuzzy system [13] with the time series of the ideal speakers. In the ANFIS approach a neural network [5, 10, 12] is used to adapt the parameters of the fuzzy system with real data of the problem. With this fuzzy model, the time series of other speakers can be used as checking data to evaluate the total error between the real speaker and the desired one. The third part of the problem was solved by using another set of fuzzy rules [14], which basically are fuzzy expert rules to decide on the quality of the speakers based on the total checking error obtained in the previous step. Of course, in this case we needed to define membership functions for the error and quality of the product, and the Mamdani reasoning approach was used. We also use as input variable of the fuzzy system the fractal dimension of the sound signal. The fractal dimension [8] is a measure of the geometrical complexity of an object (in this case, the time series). We tested our neuro-fuzzy-fractal approach for automated quality control during production with real sound speakers with excellent results. Of course, to measure the efficiency of our intelligent system we compared the results of the neuro-fuzzy-fractal approach to the ones by real human experts. The results clearly show that the neuro-fuzzy-fractal approach was better than the manual method because it reduced the time required for testing and also the accuracy was improved slightly. We think that our neuro-fuzzy-fractal approach for quality control can be used for similar problems, with only some minor changes to the structure of the fuzzy system.

## 2. Basic Concepts of Sound Speakers

In any sound system, ultimate quality depends on the speakers [4]. The best recording, encoded on the most advanced storage device and played by a top-of-the-line deck and amplifier, will sound awful if the system is hooked up to poor speakers. A system's speaker is the

component that takes the electronic signal stored on things like CDs, tapes and DVD's and turns it back into actual sound that we can hear.

## 2.1 Sound Basics

To understand how speakers work, the first thing you need to do is understand how sound works. Inside your ear is a very thin piece of skin called the eardrum. When your eardrum vibrates, your brain interprets the vibrations as sound. Rapid changes in air pressure are the most common thing to vibrate your eardrum.

An object produces sound when it vibrates in air (sound can also travel through liquids and solids, but air is the transmission medium when we listen to speakers). When something vibrates, it moves the air particles around it. Those air particles in turn move the air particles around them, carrying the pulse of the vibration through the air as more and more particles are pushed farther from the source of the vibration.

To see how this works, let's look at a simple vibrating object -- a bell. When you ring a bell, the metal vibrates -- flexes in and out -- rapidly. When it flexes out on one side, it pushes out on the surrounding air particles on that side. These air particles then collide with the particles in front of them, which collide with the particles in front of them, and so on. When the bell flexes away, it pulls in on these surrounding air particles, creating a drop in pressure that pulls in on more surrounding air particles, which creates another drop in pressure that pulls in particles that are even farther out, and so on. This decreasing of pressure is called rarefaction.

In this way, a vibrating object sends a wave of pressure fluctuation through the atmosphere. When the fluctuation wave reaches your ear, it vibrates the eardrum back and forth. Our brain interprets this motion as sound. We hear different sounds from different vibrating objects because of variations in:

- sound wave frequency -- A higher wave frequency simply means that the air pressure fluctuates faster. We hear this as a higher pitch. When there are fewer fluctuations in a period of time, the pitch is lower.
- air pressure level -- the wave's amplitude -- determines how loud the sound is. Sound waves with greater amplitudes move our ear drums more, and we register this sensation as a higher volume.

*A speaker is a device that is optimized to produce accurate fluctuations in air pressure.*

A microphone works something like our ears. It has a diaphragm that is vibrated by sound waves in an area. The signal from a microphone gets encoded on a tape or CD as an electrical signal. When you play this signal back on your stereo, the amplifier sends it to the speaker, which re-interprets it into physical vibrations. In the next section, we'll see how the speaker accomplishes this task.

## 2.2 Making Sound

In the last section we saw that sound travels in waves of air pressure fluctuation, and that we hear sounds differently depending on the frequency and amplitude of these waves. We also learned that microphones translate sound waves into electrical signals, which can be encoded onto CDs, tapes, LPs, etc. Players convert this stored information back into an electric current for use in the stereo system.

A speaker is essentially the final translation machine -- the reverse of the microphone. It takes the electrical signal and translates it back into physical vibrations to create sound waves. When everything is working as it should, the speaker produces nearly the same vibrations that the microphone originally recorded and encoded on a tape, CD, LP, etc. Traditional speakers do this with one or more drivers. A driver produces sound waves by rapidly vibrating a flexible cone, or diaphragm. Figure 1 shows a typical speaker driver.

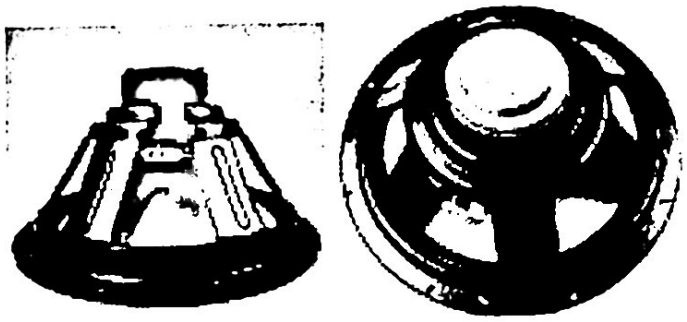


Figure 1 A typical speaker driver, with a metal basket, heavy permanent magnet and paper diaphragm

- The cone, usually made of paper, plastic or metal, is attached on the wide end to the suspension, or surround.
- This rim of flexible material allows the cone to move, and is attached to the driver's metal frame, called the basket.
- The narrow end of the cone is connected to the voice coil.
- The coil is attached to the basket by the spider, a ring of flexible material. The spider holds the coil in position, but allows it to move freely back and forth.

Some drivers have a dome instead of a cone. A dome is just a diaphragm that extends out instead of tapering in.

The voice coil is a basic electromagnet. An electromagnet is a coil of wire, usually wrapped around a piece of magnetic metal, such as iron. Running electrical current through the wire creates a magnetic field around the coil, magnetizing the metal it is wrapped around. The field acts just like the magnetic field around a permanent magnet: It has a polar orientation -- a "north" end and a "south" end -- and it is attracted to iron objects. But unlike a permanent magnet, in an

electromagnet you can alter the orientation of the poles. If you reverse the flow of the current, the north and south ends of the electromagnet switch.

This is exactly what a stereo signal does -- it constantly reverses the flow of electricity. If you've ever hooked up a stereo system, then you know that there are two output wires for each speaker -- typically a black one and a red one. Figure 2 shows the wire that runs through the speaker system.

Essentially, the amplifier is constantly switching the electrical signal, fluctuating between a positive charge and a negative charge on the red wire. Since electrons always flow in the same direction between positively charged particles and negatively charged particles, the current going through the speaker moves one way and then reverses and flows the other way. This alternating current causes the polar orientation of the electromagnet to reverse itself many times a second.

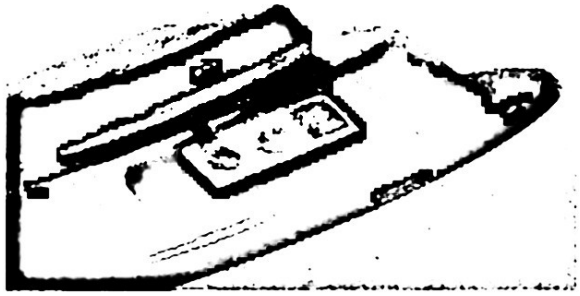


Figure 2 The wire that runs through the speaker system connects to two hook-up jacks on the driver.

So how does this fluctuation make the speaker coil move back and forth? The electromagnet is positioned in a constant magnetic field created by a permanent magnet. These two magnets -- the electromagnet and the permanent magnet -- interact with each other as any two magnets do. The positive end of the electromagnet is attracted to the negative pole of the permanent magnetic field, and the negative pole of the electromagnet is repelled by the permanent magnet's negative pole.

When the electromagnet's polar orientation switches, so does the direction of repulsion and attraction. In this way, the alternating current constantly reverses the magnet forces between the voice coil and the permanent magnet. This pushes the coil back and forth rapidly, like a piston. When the electrical current flowing through the voice coil changes direction, the coil's polar orientation reverses. This changes the magnetic forces between the voice coil and the permanent magnet, moving the coil and attached diaphragm back and forth.

When the coil moves, it pushes and pulls on the speaker cone. This vibrates the air in front of the speaker, creating sound waves. The electrical audio signal can also be interpreted as a wave. The frequency and amplitude of this wave, which represent the original sound wave, dictates the rate and distance that the voice coil moves. This, in turn, determines the frequency and amplitude of the sound waves produced by the diaphragm. Different driver sizes are better suited for certain frequency ranges. For this reason, loudspeaker units typically divide a wide frequency range among multiple drivers. In the next section, we'll find out how speakers divide up the frequency range, and we'll look at the main driver types used in loudspeakers.

## 2.3 Chunks of the Frequency Range

In the last section we saw that traditional speakers produce sound by pushing and pulling an electromagnet attached to a flexible cone. Although drivers all work on the same concept, there is actually a wide variety in driver size and power. The basic driver types are: 1) Woofers 2) Tweeters 3) Midrange (Figure 3).

Woofers are the biggest drivers, and are designed to produce low frequency sounds. Tweeters are much smaller units, designed to produce the highest frequencies. Midrange speakers produce a range of frequencies in the middle of the sound spectrum.

And if you think about it, this makes perfect sense. To create higher frequency waves -- waves in which the points of high pressure and low pressure are closer together -- the driver diaphragm must vibrate more quickly. This is harder to do with a large cone because of the mass of the cone. Conversely, it's harder to get a small driver to vibrate slowly enough to produce very low frequency sounds. It's more suited to rapid movement.

To produce quality sound over a wide frequency range more effectively, you can break the entire range into smaller chunks that are handled by specialized drivers. Quality loudspeakers will typically have a woofer, a tweeter and sometimes a midrange driver, all included in one enclosure.

Of course, to dedicate each driver to a particular frequency range, the speaker system first needs to break the audio signal into different pieces -- low frequency, high frequency and sometimes mid-range frequencies. This is the job of the speaker crossover.



Woofer Tweeter Midrange

Figure 3 Three basic driver types: woofer, tweeter, and midrange.

The most common type of crossover is passive, meaning it doesn't need an external power source because it is activated by the audio signal passing through it. This sort of crossover uses inductors, capacitors, and sometimes, other circuitry components. Capacitors and inductors only become good conductors under certain conditions. A crossover capacitor will conduct the current very well when the frequency exceeds a certain level, but will conduct poorly when the frequency is below that level. A crossover inductor acts in the reverse manner -- it is only a good conductor when the frequency is below a certain level (Figure 4).

When the electrical audio signal travels through the speaker wire to the speaker, it passes through the crossover units for each driver. To flow to the tweeter, the current will have to pass through a capacitor. So for the most part, the high frequency part of the signal will flow on to the tweeter voice coil. To flow to the woofer, the current passes through an inductor, so the driver will mainly respond to low frequencies. A crossover for the mid-range driver will conduct the current through a capacitor and an inductor, to set an upper and lower cutoff point.

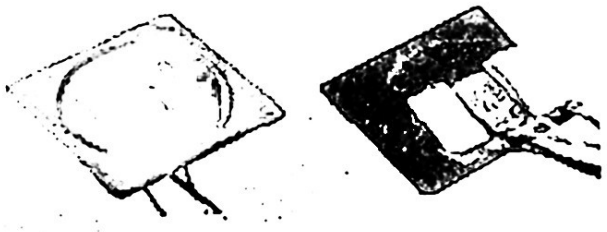


Figure 4 Typical crossover unit from a loudspeaker. The frequency is divided up by inductors and capacitors and then sent on to the woofer, tweeter and mid-range driver.

There are also active crossovers. Active crossovers are electronic devices that pick out the different frequency ranges in an audio signal before it goes on to the amplifier (you use an amplifier circuit for each driver). They have several advantages over passive crossovers, the main

one being that you can easily adjust the frequency ranges. Passive crossover ranges are determined by the individual circuitry components -- to change them, you need to install new capacitors and inductors. Active crossovers aren't as widely used as passive crossovers, however, because the equipment is much more expensive and you need multiple amplifier outputs for your speakers.

Crossovers and drivers can be installed as separate components in a sound system, but most people end up buying speaker units that house the crossover and multiple drivers in one box. In the next section, we'll find out what these speaker enclosures do and how they affect the speaker's sound quality.

# 3. Description of the problem

The basic problem consists in the identification of sound signal quality. Of course, this requires a comparison between the real measured sound signal and the ideal good sound signal. We need to be able to accept speakers, which have a sound signal that do not differ much from the ideal signals. We show in Figure 5 (a) the form of the sound signal for a good speaker (of a specific type). The measured signal contains about 108 000 points in about 3 seconds. We need to compare any other measured signal with the good one and calculate the total difference between both of them, and if the difference is small then we can accept the speaker as a good one. On the other hand, if the difference is large then we reject the speaker as a bad one.

We show in Figure 5 (b) the sound signal for a speaker of bad quality. We can clearly see the difference in the geometrical form of this signal and the one shown in Figure 5 (a). In this case, the difference between the figures is sufficiently large and we easily determine that the speaker is of bad quality. We also show in Figure 5 (c) another sound signal for a bad quality speaker. Again, we can see clearly the difference in the form of the signal with respect to the good speaker.



(a) (b) (c)

Figure 5 Sound signal (a) Good Speaker (b) Bad Speaker (Case 1), (c) Bad Speaker (Case 2)

# 4. Fractal Dimension of a Geometrical Object

Recently, considerable progress has been made in understanding the complexity of an object through the application of fractal concepts [8] and dynamic scaling theory [11]. For example, financial time series show scaled properties suggesting a fractal structure [1, 2, 3]. The fractal dimension of a geometrical object can be defined as follows:

$$d = \lim_{r \to 0} [\ln N(r)] / [\ln(1/r)] \qquad (1)$$

where N(r) is the number of boxes covering the object and r is the size of the box. An approximation to the fractal dimension can be obtained by counting the number of boxes covering the boundary of the object for different r sizes and then performing a logarithmic regression to obtain d (box counting algorithm). In Figure 6 (a), we illustrate the box counting algorithm for a hypothetical curve C. Counting the number of boxes for different sizes of r and performing a logarithmic linear regression, we can estimate the box dimension of a geometrical object with the following equation:

$$\ln N(r) = \ln\beta - d \ln r \qquad (2)$$

this algorithm is illustrated in Figure 6 (b).

The fractal dimension can be used to characterize an arbitrary object. The reason for this is that the fractal dimension measures the geometrical complexity of objects. In this case, a time series can be classified by using the numeric value of the fractal dimension (d is between 1 and 2 because we are on the plane xy). The reasoning behind this classification scheme is that when the boundary is smooth the fractal dimension of the object will be close to one. On the other hand, when the boundary is rougher the fractal dimension will be close to a value of two.

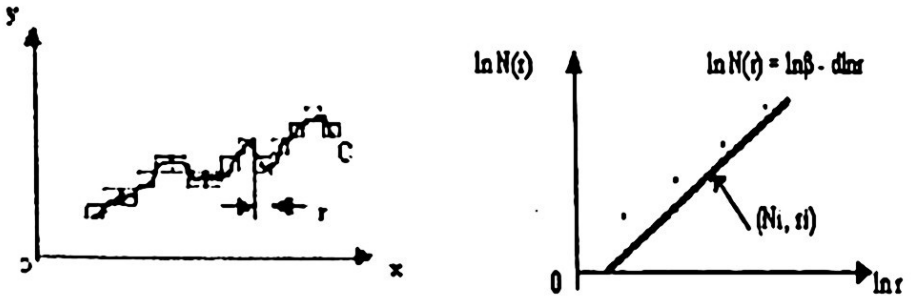



Figure 6 (a) Box counting algorithm for a curve C,
(b) Logarithmic regression to find dimension

We developed a computer program in MATLAB for calculating the fractal dimension of a sound signal. The computer program uses as input the figure of the signal and counts the number of boxes covering

the object for different grid sizes. The fractal dimension for the sound signal of Figure 5(a) is of 1.6479, which is a low value because it corresponds to a good speaker. On the other hand, the fractal dimension for Figure 5 (b) is 1.7843, which is a high value (bad speaker). Also, for the case of Figure 5 (c) the dimension is 1.8030, which is even higher (again, a bad speaker).

# 5. Experimental Results

We describe in this section the experimental results obtained with the intelligent system for automated quality control. The intelligent system uses a fuzzy rule base to determine automatically the quality of sound in speakers. We used a neuro-fuzzy approach to adapt the parameters of the fuzzy system using real data from the problem. We used the time series of 108000 points measured from a good sound speaker (in a period of 3 seconds) as training data in the ANFIS approach. We then use the measured data of any other speaker as checking data, to compare the form of the sound signals. We show in Figures 7 and 8 two cases where the ANFIS approach is used to adapt a fuzzy system with training data of good sound speakers. The approximation is very good considering the complexity of the problem. Once the training was done, we used the fuzzy system for measuring the total difference between a given signal and the good ones. This difference is used to decide on the final quality of the speaker using another set of fuzzy rules with the Mamdani approach.

We show in Figure 9 (a) the implementation of the fuzzy rule base, in the Fuzzy Logic Toolbox of the MATLAB programming language. We also show in Figure 9 (b) the non-linear surface for modelling the quality control problem. Finally, we show in Figure 9 (c) the fuzzy rule viewer of MATLAB, in which the fuzzy rule base is used for specific values of the input variables.

The fuzzy rules are as follows:

IF Difference is small AND Fractal Dimension is small
THEN Quality is Excellent

IF Difference is regular AND Fractal Dimension is small
THEN Quality is Good

IF Difference is regular AND Fractal Dimension is high
THEN Quality is Medium

IF Difference is medium AND Fractal Dimension is small
THEN Quality is Medium

IF Difference is medium AND Fractal Dimension is high
THEN Quality is Bad

IF Difference is large AND Fractal Dimension is small
THEN Quality is Medium

IF Difference is large AND Fractal Dimension is high
THEN Quality is Bad

IF Difference is small AND Fractal Dimension is high
THEN Quality is Medium

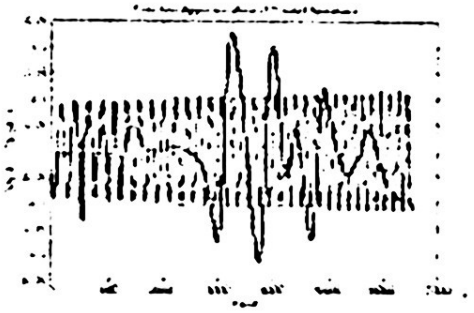


Figure 7 Function approximation of the sound signal using ANFIS (Case 1).

In the fuzzy rule base, the "Difference", "Fractal Dimension" and "Quality" are considered as linguistic variables in the fuzzy rules. We used gaussian membership functions for the linguistic values of all the variables. The membership functions for the linguistic values of these variables were tuned by using a simple genetic algorithm. Of course, the human experts on quality evaluation gave initial values for the fuzzy system, but these were optimized using real data for the problem and by applying the simple genetic algorithm.

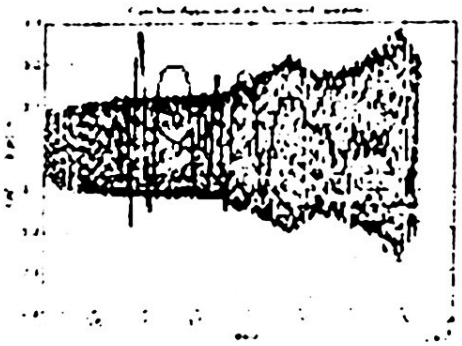


Figure 8 Function approx. of the sound signal using ANFIS (Case 2).

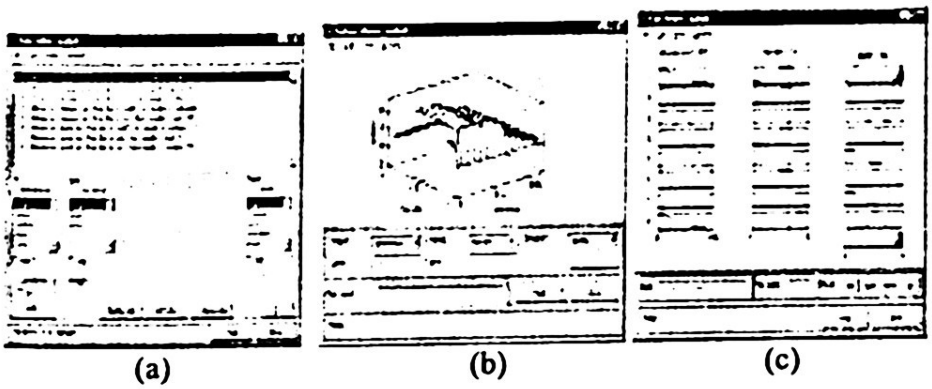



Figure 9 (a)Fuzzy rule base for quality control, (b) Non-linear surface for quality control, (c) Rule viewer for specific input values.

## 6. Conclusions

We described in this paper the application of a neuro-fuzzy-fractal approach to the problem of automating the quality control of sound speakers during manufacturing in a real plant. We have implemented an intelligent system for quality control in the MATLAB programming language using the ANFIS approach. We also use the fractal dimension as a measure of geometrical complexity of the sound signals. The intelligent system performs rather well considering the complexity of the problem. The intelligent system has been tested in a real manufacturing plant with very good results. We think that our approach for automating quality evaluation can used for similar problems with only minor changes in the membership functions and rules.

## Acknowledgments

We would like to express our gratitude to the Research Grant Committee of COSNET, for their financial support under grant 487.02P.

## References:

[1] Castillo, O. and Melin P. (1998). "A New Fuzzy-Genetic Approach for the Simulation and Forecasting of International Trade Non-Linear Dynamics", Proc. of CIFEr'98, IEEE Press, New York, USA, pp. 189-196.

[2] Castillo, O. and Melin, P. (1999). "Automated Mathematical Modelling for Financial Time Series Prediction Combining Fuzzy Logic and Fractal Theory", Edited Book " Soft Computing for Financial Engineering", Springer-Verlag, Heidelberg, Germany, pp. 93-106.

[3] Castillo, O. and Melin, P. (2000). A New Method for Fuzzy Estimation of the Fractal Dimension and its Applications to Time Series Analysis and Pattern Recognition, Proc. of NAFIPS'2000, Atlanta, GA, USA. pp. 451-455.

[4] Dickason, V. (1997). The Loudspeaker Cookbook, McGraw Hill.

[5] Haykin, S. (1996). "Adaptive Filter Theory", Third Edition, Prentice Hall.

[6] Jang, J.R., Sun, C.T. and Mizutani E. (1997). Neuro-Fuzzy and Soft Computing, Prentice Hall.

[7] Loctite Co. (1999). Speaker Assembly Adhesives Guide.

[8] Mandelbrot, B. (1987). "The Fractal Geometry of Nature", W.H. Freeman.

[9] Melin, P. and Castillo, O. (2000). Controlling Complex Electrochemical Manufacturing Processes with Hybrid Intelligent Systems, Proc. of NAFIPS'2000, Atlanta, GA, USA. pp. 490-494.

[10] Parker, D.B. (1982). " Learning Logic", Invention Report 581-64, Stanford University.

[11] Rasband, S.N. (1990). "Chaotic Dynamics of Non-Linear Systems", Wiley Interscience.

[12] Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986). "Learning Internal Representations by Error Propagation", in "Parallel Distributed Processing: Explorations in the Microstructures of Cognition", MIT Press, Cambridge, MA, USA, Vol. 1, pp. 318-362.

[13] Sugeno, M. and Kang, G.T. (1988). Structure Identification of Fuzzy Model, *Fuzzy Sets and Systems*, Vol. 28, pp.15-33.

[14] Zadeh, L. A. (1975). "The Concept of a Linguistic Variable and its Application to Approximate Reasoning", *Information Sciences*, 8, pp. 43-80.

# A Hybrid Intelligent Approach for Automated Quality Control Combining Learning Vector Quantization Neural Networks and Fuzzy Logic

O. Castillo[1], R. Sepulveda[2], O. Montiel[2], P. Melin[1]
[1] Dept. of Computer Science
Tijuana Institute of Technology, Mexico
[2] CITEDI-IPN, Tijuana, Mexico
ocastillo@tectijuana.mx, pmelin@tectijuana.mx

## Abstract

We describe in this paper a new hybrid intelligent approach for automated quality control combining Learning Vector Quantization (LVQ) and fuzzy logic. In our approach, LVQ neural networks are used for image processing and classification. Also, a set of fuzzy rules are used for solving the problem of automating the decision making for quality control. The fuzzy system contains the expert knowledge for quality evaluation. The new approach has been tested with the specific case of automating the quality control of tomato in a food processing plant with excellent results.

## 1. Introduction

Automating quality control in manufacturing applications is very important because there is always a need of reducing times and costs of production [2, 3]. Also, traditional quality control relies on human experts and as a consequence has the following problems [7]:

a) Lack of objectivity due to several factors as illumination, mud changes, humans being tired, etc.
b) Lack of homogeneity due to similar factors as before.
c) Lack of accuracy as a consequence of the previous problems.

The use of artificial vision in classification processes, not only helps in reducing the time of identification, also helps in giving

greater consistency to the classification, and this implies more homogeneity in quality evaluation [7]. Neural networks [9] and fuzzy logic are soft computing techniques that have been used for pattern recognition and decision making problems with good results [1, 4, 11]. For this reason, these techniques are a good alternative in manufacturing applications [3, 8].

The approach proposed in this paper is to use LVQ neural networks [6] to perform image processing and classification, and then use a set of fuzzy if-then rules [12] to decide on the quality based on the results of the LVQ network. The LVQ network enables the classification of an arbitrary geometrical object based on its digital image. However, the classification outputs discrete values only, and therefore fuzzy logic [11, 12] is needed to model the uncertainty associated with this classification process. In this case, a digital video camera is used to capture the images of the objects subject of the classification. The images are then processed to compute the area or volume. These images are then classified using the LVQ network. Finally, a set of fuzzy rules are used to decide on the quality of the product.

Neural networks with local activation functions, for example Radial Basis Functions (RBF) Networks, have the merit of excellent generalization capabilities [5]. So they have been widely used in function approximation [4]. Learning Vector Quantization (LVQ) networks also have the characteristic of local activation functions. Each output node is given a prototype label, which is its associated output value. For each output node, through LVQ learning, an appropriate local region is found where the output is approximated by the label value. Thus the partition of the input space is done based on the output information. The LVQ network, however, outputs discrete values only, and therefore can not approximate real-valued functions. For this reason, fuzzy sets are used to cope with the uncertainty in the classification done by LVQ.

## 2. LVQ Networks and Function Approximation

Kohonen's LVQ network is a supervised learning algorithm associated with the competitive network shown in Figure1. The network consists of an input layer and an output layer. A weight vector wi is associated to the i-th node in the output layer. In learning phase, the LVQ network selects a weight vector closest to a given input vector and then compares the output of LVQ network with the output of training data [6]. If they match, the selected weight vector is updated so that it approaches the input vector. Otherwise, the selected weight vector is updated so that it moves away from the input vector. After the learning, the LVQ network chooses the nearest weight vector to a given input vector, and outputs its 'label' as the network output. Thus, a weight vector can be regarded as the center of a local region in the input space.

In function approximation problems, a certain fixed real value must be assigned to each weight vector as its 'label'. So, if there are a sufficient number of weight vectors with various 'label' values, they found their proper locations by their learning. This realizes the partition of the input space based on the output values of the training data.

However, when the LVQ algorithm is used in function approximation, it has some problems. The LVQ training algorithm drives a weight vector closer to a training input vector when the 'label' value of the weight vector matches the training output value. But, there are only a finite number of 'labels' while the training output data can take any real number. So, there are an infinite number of data that do not match any weight vector 'label', and thus the LVQ learning algorithm sometimes does not work well in function approximation problems. Another problem occurs when the output is calculated after learning. The LVQ network output can take a value among the finite set of 'label' values only.

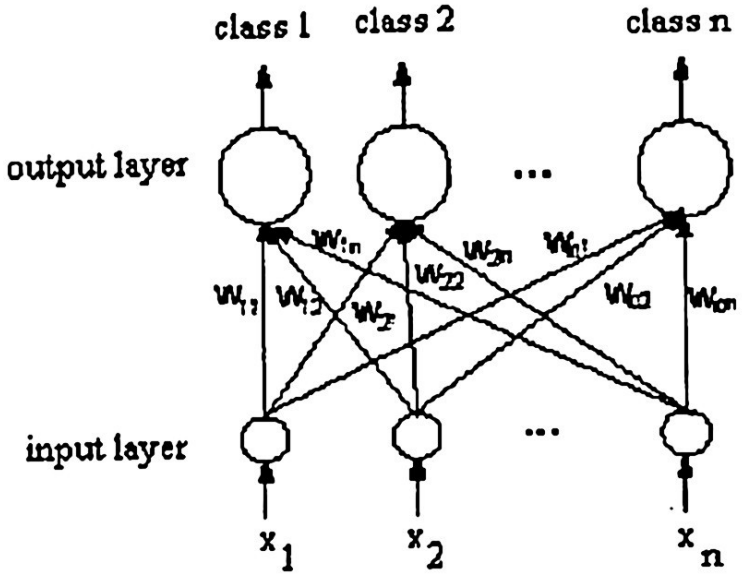



Figure 1 Structure of LVQ network

## 3. Problem Description

The San Quintin valley is a region of Baja California, Mexico in which tomato is the main export product. The tomato is taken from the fields to the food processing plants, in which according to their quality a classification is performed. Tomato is classified in to four categories: export quality, national quality, regional, and salsa quality. This classification is very important because the net profit from tomato production depends on the quality evaluation been done appropriately.

Traditionally quality control of tomato production has been done by manually selecting the good quality tomatoes from production lines. Human workers visually classify the tomatoes from the production line. The tomatoes move on a belt, as shown in Figure 2, and are classified according to their size and to their appearance. Human experts know how to do this job very well. However, they usually have the problems of objectivity, homogeneity, and accuracy, mentioned before. Also, there is a need of performing quality control in less time.

For this reason, automating the quality control process was decided to reduce costs and times of production.

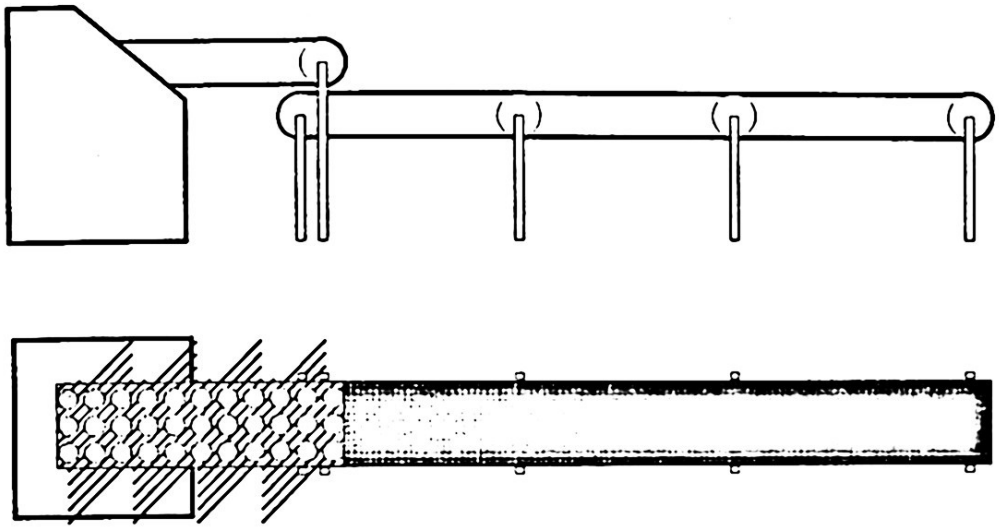


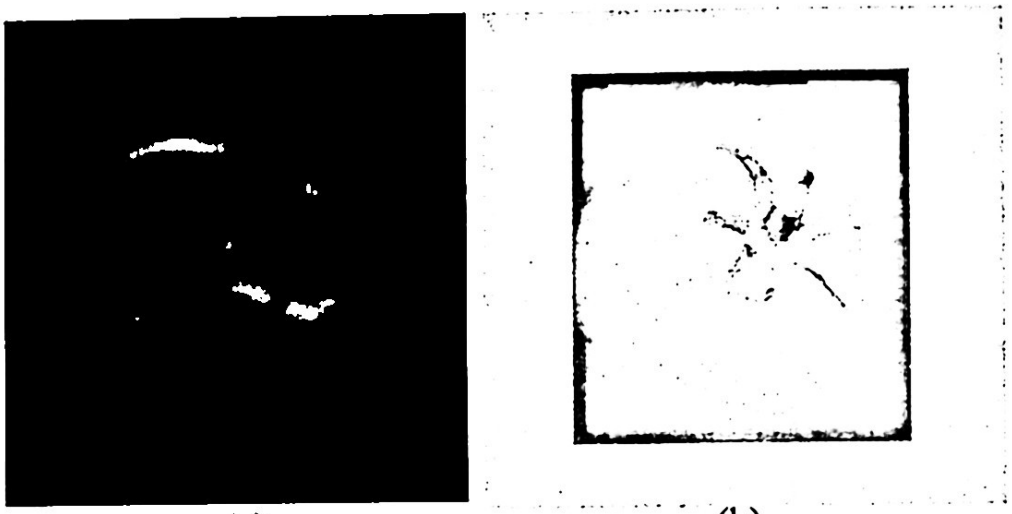
Figure 2 Belt of a tomato production line.

A window of opportunity was recognized for the application of soft computing techniques for achieving automated quality control of tomato production. A LVQ neural network was selected to perform image processing and classification. Also, fuzzy logic techniques were used to deal with inherent uncertainty in deciding the final quality of the product.

## 4. Image Acquisition and Processing

There are several techniques for processing digital images. For example, in the MATLAB language there are several functions for image analysis and processing [10]. To begin the process, we first need to define the method for image acquisition. This method has to acquire the digital image with less noise as possible. Of course, any acquisition method adds noise to the image, but we need to use a method that minimizes this noise.

For a real time system, like for tomato classification, image acquisition has to be done with a digital camera, but the scene has to be reflection free and with uniform illumination. We show in Figure 3(a) the image of a tomato acquired with a digital camera. Figure 3(b) shows the same image, but with intensity values adjusted for better processing. After the image is acquired, then we perform processing operations to calculate the area of the image and estimate the size of the tomato.



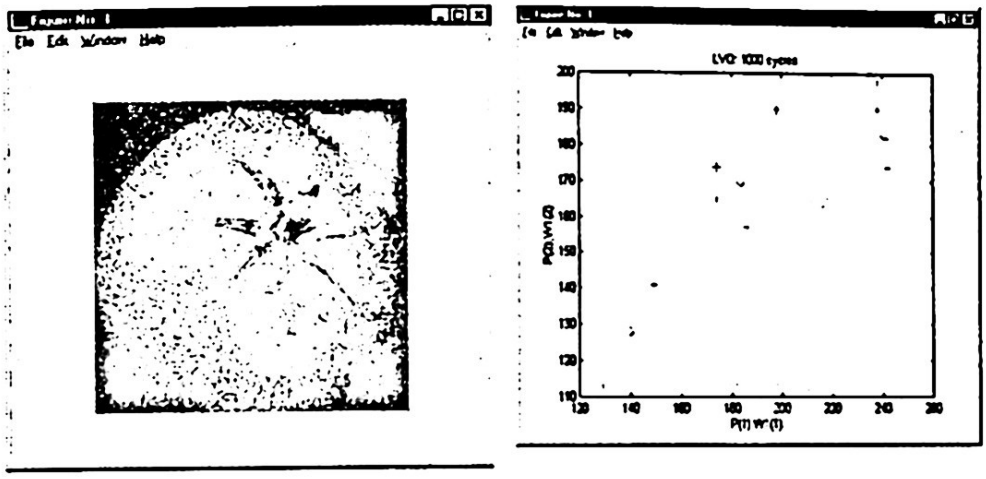
(a) (b)

Figure 3 (a) Image acquired by digital camera, (b) Image with intensity values adjusted.

## 5. Application of LVQ Neural Network for Image Classification

In this section, we describe how the LVQ network was applied to achieve image classification. First, we did make a choice about which pixels of the image to use. For tomato image we used ten pixels as input to the network. We show in Figure 4(a) the tomato image with the pixels used as input. Of course, the final choice of pixels was achieved after experimentation, and comparison of results. We consider that the final choice of pixels is representative of the image, or at least this was true

for the purpose of this work. We also show in Figure 4(b) the final training of the LVQ network after 1000 cycles of learning.



(a) (b)

Figure 4 (a) Image with the choice of pixels, (b) Final training of the LVQ network.

The LVQ neural network basically classifies the tomato images according to their similarities. However, there exists uncertainty in this classification. For this reason, the use of fuzzy logic is justified to complement the LVQ network.

## 6. Fuzzy System for Automated Quality Evaluation

We formulated a set of fuzzy if then rules to automate quality evaluation of the tomato according to the results of the LVQ network. We used as input linguistic variables the area of the image (size of the tomato), maturity of the tomato, and defects of the tomato. We used as output linguistic variable the class of the tomato. We used a Mamdani type fuzzy system with seven fuzzy rules, which were formulated to model experts in this task. We show in Figure 5 the general architecture of the fuzzy system for automated quality control. In this figure it is clearly shown the use of the three inputs and one output.

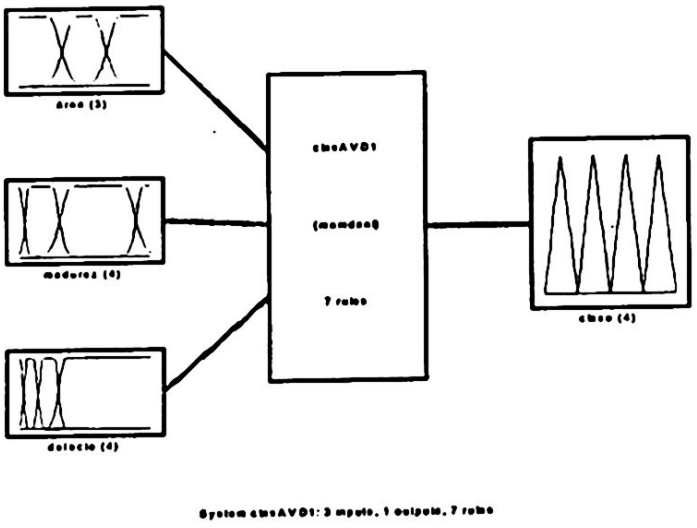


Figure 5 General architecture of the fuzzy system for quality control.

Now we describe the choice of the linguistic terms and membership functions for each of the linguistic variables in the fuzzy system. First, for the area input variable we decided to use three linguistic terms: small, medium, and large. The numerical value of this area is calculated from the tomato image (in millimeters) and this is used to calculate the membership degree to these linguistic terms. We show in Figure 6 the membership functions for the area input variable. On the other hand, for the maturity input variable we used four linguistic terms: very mature, mature, less mature, and, green. In this case, the numerical value of maturity is obtained from the LVQ network. We show in Figure 6 the membership functions of the maturity input variable. Finally, for the defect input variable we used four linguistic terms: few, regular, many, too many. In all the cases, we used generalized bell membership functions to obtain more approximation power.

For the output linguistic variable of the fuzzy system, we decided to use four linguistic terms to classify the quality of the product. The quality linguistic variable was classified as: export, national, local and salsa quality. In this case, we considered sufficient to use triangular membership functions for the output variable because we only needed to do the classification. We show in Figure 7 the membership functions for the quality linguistic variable.

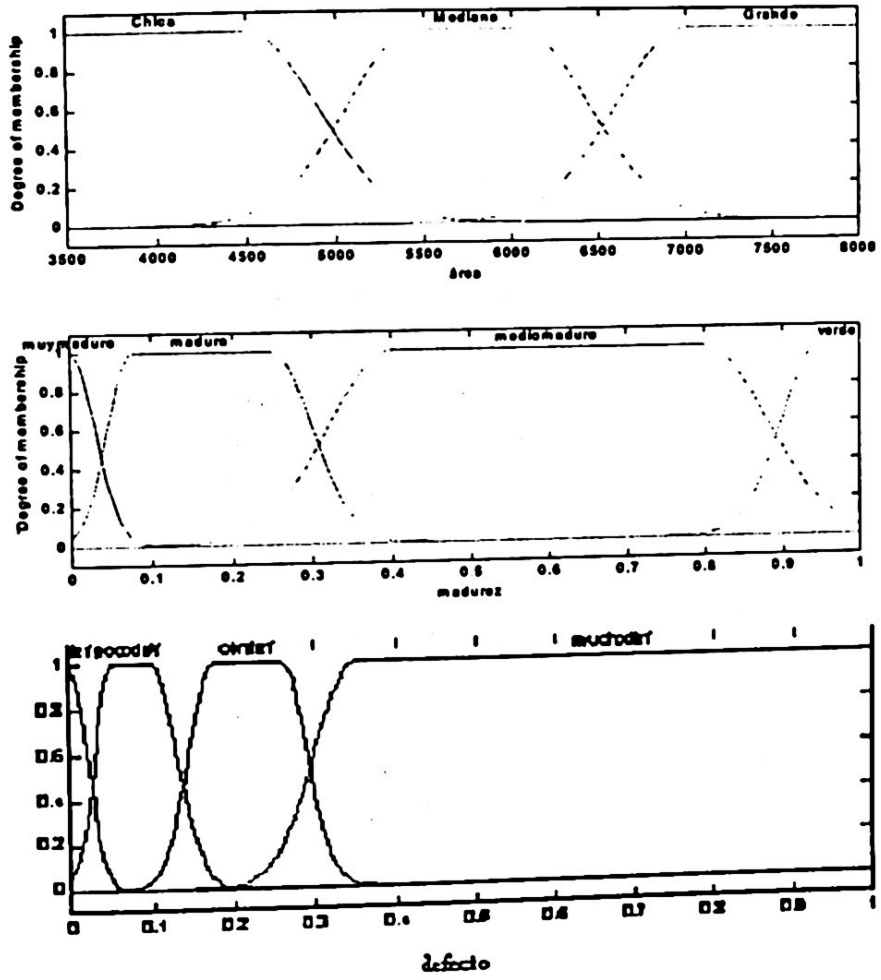



Figure 6 Membership functions for the input linguistic variables of the fuzzy system.

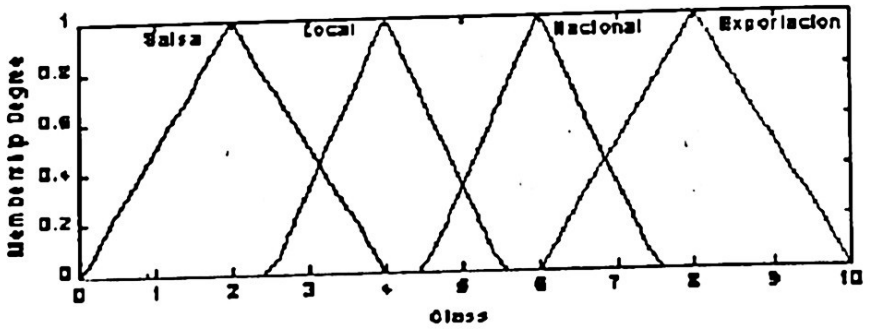



Figure 7 Membership functions for the output linguistic variable of the fuzzy system.

## 7. Conclusions

We described in this paper a new hybrid intelligent approach for automated quality control combining LVQ neural networks and fuzzy logic. In our approach, LVQ networks are used for image processing and classification. Also, a fuzzy system is used for solving the problem of automating the decision making process for quality control. The fuzzy system contains the expert knowledge for quality evaluation of a specific product. The new approach has been tested with the specific case of automating the quality control of tomato in a food processing plant with excellent results. The accuracy of the classification has been improved, using the hybrid approach, from 85% to approximately 95%. On the other hand, the time required to perform this classification process has been reduced more than 15% with the automation of the process. Finally, we have to say that this hybrid approach can be used for similar problems in quality control.

## References

[1] Castillo, O. & Melin, P. (1998). "Modelling, Simulation and Behavior Identification of Non-Linear Dynamical Systems with a New Fuzzy-Fractal-Genetic Approach",

Proceedings of IPMU'98, EDK Publishers, Vol. 1, pp. 467-474.

[2] Castillo, O. & Melin, P. (1999). "Intelligent Quality Control for the Food Industry using a Fuzzy-Fractal Approach", Proceedings IPMM'99, IEEE Press, Vol. 1, pp. 151-156.

[3] Castillo, O. & Melin, P. (2001). "Soft Computing for Control of Non-Linear Dynamical Systems", Springer-Verlag, Heidelberg, Germany.

[4] Jang, J.-S. R., Sun, C.-T. & Mizutani, E. (1997). "Neuro-fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence", Prentice-Hall.

[5] Kohonen, T. (1995). "Self-Organizing Maps", Springer Series in Information Sciences.

[6] Kohonen, T. (1990) "Learning Vector Quantization for Pattern Recognition", Technical Report TKK-F-A602, Helsinki University of Technology, Finland.

[7] Melin, P., Castillo, O. & Sotelo, F. (2001). "A Hybrid Neuro-Fuzzy-Fractal Approach for Automated Quality Control in the Manufacturing of Sound Speakers", Proceedings of ANNIE'2001, ASME Press, pp. 539-544.

[8] Melin, P. & Castillo, O. (2001). "Intelligent Control of Complex Electrochemical Systems with a Neuro-Fuzzy-Genetic Approach", IEEE Transactions on Industrial Electronics, Vol. 48, No. 5, pp. 951-955.

[9] Miller, W. T., Sutton, R.S., & Werbos, P.J. (1995). "Neural Networks for Control", MIT Press.

[10] Nakamura, S. (1997). "Numerical Analysis and Graphic Visualization with MATLAB", Prentice Hall.

[11] Ruan, D. (1996). "Fuzzy Logic Foundations and Industrial Applications", Kluwer Academic Publishers.

[12] Zadeh, L. A. (1975). The Concept of a Linguistic Variable and its Application to Approximate Reasoning, *Information Sciences*, **8**, 43-80.

# Índice de Autores


Aguirre, P., 185
Alcarraz, A., 417
Alonso, M., 313,325
Alquézar, R., 171
Altamirano, L., 201
Badía, J., 383
Balsi, M., 229
Barreto, A., 201
Bautista, E., 337,349
Berlanga, R., 265,405
Botello, S., 99
Calderón, J., 159
Campoy, P., 159
Cardelino, J., 185
Carrasco, J., 277
Castellanos, I., 125
Castillo, O., 461,473
Corona, J., 213
Cruz, R., 63
De la Cruz, A., 313,325
De la Rivera, L., 147
De Oliveira, 417
Decelis, J., 87
Díaz de León, J., 437,449
Escobedo, H., 395
Felipe, E., 111,125
Fernández, A., 417
Fernández, M., 257
Figueroa, J., 337,349,359,371
Gallegos, F., 147
García, S., 359
Gil, R., 383
Godoy, S., 289
Gómez, E. , 1
Gómez, F., 15
Gómez, H., 99
Henríquez, N., 257
Hernández, A., 99
Jiménez, J., 257
Lazo, M., 289
Levachkine, S., 135
Loeff, N., 185
López, I., 213
Marroquín, J., 99
Martínez, J., 277,289
Mavilio, A., 257
Mazzanti, F. , 1
Medina, J., 359,371
Melín, P., 461,473
Mendiola, J., 51
Molinar, J., 15
Montiel, O, 473
Moreno, J., 15
Moreno, M., 135
Pérez, O., 201
Pérez, R., 301
Pogrebnyak, O., 147
Ponomaryov, V., 147
Pons, A., 265,405
Ramírez, I., 417
Ramos, E., 111
Ribeiro, S., 171
Ritter, G., 27
Sánchez, F., 437
Sánchez, H., 75
Sánchez, L., 125
Santiago, R., 449
Santiesteban, Y., 265,405
Sepúlveda, R., 473
Serra, J., 39
Shulcloper, J., 245,265,405
Simini, F., 185
Terol, I., 51
Torres, M., 135
Velázquez, A., 135
Vilasís, X., 229
Villanueva, A., 349
Yáñez, C., 437,449

# Reconocimiento de Patrones
## Avances y Perspectivas

Juan Luis Díaz de León Santiago y Cornelio Yáñez Márquez (Eds.)

Este libro contiene una selección, con arbitraje internacional, de los artículos que fueron presentados en el *VII Congreso Iberoamericano de Reconocimiento de Patrones*, CIARP 2002, llevado a cabo en las instalaciones del Centro de Investigación en Computación del Instituto Politécnico Nacional de México, del 19 al 22 de noviembre de 2002.

El propósito fundamental de este trabajo es presentar ante la comunidad internacional involucrada en Reconocimiento de Patrones, un diagnóstico preciso y bien documentado de los avances recientes en el área y de las perspectivas de desarrollo a corto y mediano plazo en los temas afines.

Entre las áreas de interés del CIARP 2002 se encuentran: reconocimiento de patrones, procesamiento y análisis de imágenes, álgebra de imágenes, memorias asociativas, extracción de características e invariantes, procesamiento y análisis de señales, morfología matemática, visión por computadora, visión activa y en tiempo real, redes neuronales, algoritmos genéticos sistemas híbridos e implementaciones en hardware.

Este libro será de interés para investigadores y estudiantes de maestría y doctorado cuyo trabajo involucre alguna o algunas de las áreas afines al reconocimiento de patrones.