Texture Segmentation On a Local Binary Pattern Space

Gemma S. Parra-Dominguez, Raul E. Sanchez-Yanez, and
Victor Ayala-Ramirez

Universidad de Guanajuato DICIS Carretera Salamanca-Valle de Santiago Km 3.5+1.8 Km Comunidad de Palo Blanco, C.P. 36885 Salamanca, Gto.
gsparra@laviria.org, sanchezy@salamanca.ugto.mx, ayalav@salamanca.ugto.mx
(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract. A visual texture segmentation approach is proposed. Here, an input image is transformed into a local binary pattern space, making it a local indicator of texture units. Those texture units are then used to calculate local binary pattern difference histograms and local homogeneity values. Such values, computed from texture units, are a better indicator of texture presence than homogeneity values computed directly from gray levels. The local histograms improve the creation of classes and allow the possibility to discriminate between different textures. Then, our system performs both tasks, texture detection and texture discrimination trough a refinement of the detected textured regions. Experimental work shows good texture segmentation over Brodazt textures, along with good identification of non-textured regions. Texture segmentation is also performed on natural scenes with satisfactory results.

Key words: Texture detection, Texture segmentation, Local binary patterns, Difference histograms, Local homogeneity values.

1 Introduction

Visual texture is an important property of most objects, since it helps us to differentiate between elements in an image. Usually, texture is considered as an indicative of the gray level spatial distribution; and it is possible to describe it as the repetition of a certain pattern (texture element or texel) over a large region, larger than the pattern itself [1].

Texture analysis plays an important role in low level image analysis and understanding [2]. Usually, texture detection represents a processing prior to image classification or segmentation. Many texture analysis methods are based on gray level co-occurrence matrices, difference histograms, lineal transformations, Gabor filters, and Markov random fields [1]. Even though, most methods in texture analysis are orientated towards texture classification or segmentation, Karu *et al.* [3] argue that before applying any texture analysis to an image, it is necessary to identify if the input image has some textural characteristics, and if so, to locate where the texture is.

(C) C. Zepeda, R. Marcial, A. Sánchez
J. L. Zechinelli and M. Osorio (Eds)
Advances in Computer Science and Applications
Research in Computing Science 53, 2011, pp. 103-113

A texture characterization approach is the spatial gray level co-occurrence matrices (GLCM) introduced by Haralick [4]. Texture features such as energy, entropy and homogeneity can be computed from GLCM [4]. Each texture feature represents a specific property or phenomenon. In particular, homogeneity refers to how similar distributions of gray levels are. It is a second order statistic, with values between 0 and 1, where 0 stands for heterogeneous distributions, and 1 is for identical ones. Unser [5] proposed the use of first order statistics, computing sum and difference histograms, to approximate those features. Then, computation time is reduced and memory storage reminds manageable [5].

Different from GLCM approaches, there are non parametric methods such as the local binary patterns (LBP). A number of works about LBP is found nowadays. The LBP operator introduced by Ojala *et al.* [6] provides a robust way for describing pure local binary patterns in a texture. It is relatively invariant with respect to changes in illumination and image rotation, and it is computationally simple [7]. The use of LBP histograms as texture feature descriptors, has been extensively explored to achieve texture segmentation and texture classification [2, 7, 8, 9, 10]. However, none of those works have used an image in the LBP domain to compute textural features.

In this paper, an approach for texture segmentation is presented using an image transformation into an LBP space. Two textural features are then composed from it, local homogeneity values and LBP difference histograms. These allow the detection of texture presence and the discrimination between different types of texture. Although the most relevant activity is the transformation into an LBP space, four similar issues are repeated in the two stages of the system (detection and refinement): an splitting of the input image, a computation of feature vectors, a clustering process and a conformation of the output image.

The rest of the paper is organized as follows. In Section II, an introduction to the LBP space transformation is given. Also a brief review about difference histograms and homogeneity values computation is included. Section III describes the method explaining every step and assumed considerations. In Section IV, a performance evaluation is shown, also results and a brief discussion are presented. Finally, the conclusions of this study are given in Section V.

2 Image Transformation and Textural Features

The LBP transformation is a very simple process, just consisting in sums and comparisons. Having an input image I_{in} of size $K \times L$, it is scanned by a window of 3×3 pixels, and pixels inside the window are thresholded by the center pixel value. Values bigger or equals to the threshold are marked as 1; otherwise, they are marked as 0. As result, a binary neighborhood is obtained. Later, this binary neighborhood is multiplied by the weights given to the corresponding pixels. Finally, the values of the eight pixels are summed up to obtain a texture unit value associated to this neighborhood, as shown in Fig. 1 [6]. The transformed image will be of $(K-2) \times (L-2)$ size.

LBP texture units characterize a region, depending on its frequency over a determined zone. A distribution of texture units shows which of them are present and the number of times they appear in the region under inspection. Moreover, a distribution of texture unit differences, shows a relationship between them. When those texture units have similar values, the distribution of differences will tend to 0. Then, this distribution of differences, also called difference histogram, can characterize a region.

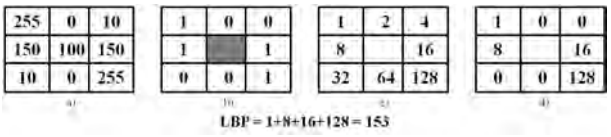

Fig. 1. An example of how LBP operator is applied to a pixel neighborhood[6].

Difference histograms and homogeneity values are calculated according to Unser's formulation, as explained in [5]. Then, having a normalized difference histogram $\hat{P}$ the local homogeneity in specific neighborhoods is computed using

$$h = \sum_{j} \frac{1}{1 + j2} \hat{P}(j) \tag{1}$$

where j is the difference value between two pixels according to a particular displacement and orientation [5].

As a feature, texture may have preferential directions over the image, and the inspection formulated should include most of them. Then, two displacement vectors $d_x = \{1,2\}$ and $d_y = \{1,2\}$ pixels, together with an angle vector $\Theta = \{0,45,90,135\}$ are used, seeking to detect texture in different directions. The average value of the 8 calculations in the Cartesian product $d \times \Theta$ is used in this work to calculate $\hat{P}$ and h.

3 Texture Segmentation Approach

The texture segmentation approach includes two stages, a detection of textured regions and a refinement of detected regions. In the first one, an image indicating the presence of texture is obtained; while in the second one, an image indicating the presence of different types of texture is reached. The main difference between both stages, is that more rigorous similarity measures between regions are used for refinement.

3.1 Detection of Textured Regions

A texture detection process is performed in order to determine if the image under test has some textured regions or not. If texture is identified on the input

image, a second process is then performed to achieve texture discrimination. The detection process performs four activities: an splitting of the input image, a computation of feature vectors, a clustering procedure, and finally, a conformation of the output image.

Splitting of the Input Image The texture detection system receives as input data an image I_{in} of $K \times L$ size in gray levels. This I_{in} is split into smaller images I_{a} using a partition window of size $M \times K$, where $M < K$ and $N < L$. This partition window is displaced according to a particular distance in each direction, horizontal and vertical. Then, a number of I_{a} is obtained from I_{in} depending on the partition window size and its displacement distance. This displacement distance determines the resolution of the system.

Computation of the Feature Vectors In general, the feature vectors are composed of a value of homogeneity, an LBP differences histogram and a key code of the region position. They are computed as follows. Every I_{a} is transformed into an LBP space, where a transformed I_{Ti} is obtained. Later, an examination over I_{Ti} is done in order to determine if I_{Ti} is an uniform region. I_{Ti} is divided in four different images I_{Tj} of the equal size. For every I_{Tj} an histogram W_{j} is computed. Then, the four W_{j} are compared using the cosine amplitude metric

$$r_{a,b} = \frac{\sum_{k=1}^{m} W_{ak} W_{bk}}{\sqrt{(\sum_{k=1}^{m} W_{ak}^2)(\sum_{k=1}^{m} W_{bk}^2)}} \tag{2}$$

where $r_{\text{a,b}}$ is the similarity value between two different histograms, W_{ak} and W_{bk} are the histograms with m values. The cosine metric amplitude is used to compare two histograms and to determine, in a range from 0 to 1, how similar they are. A value of 1 means identical histograms, while a value of 0 means dissimilar histograms. The smallest $r_{\text{a,b}}$ is divided by the biggest $r_{\text{a,b}}$ of the four W_{j}. From this division, a value bigger than 0.8 is expected; if it is reached, the four I_{Tj} are joined as I_{Ti}, if not, they are handled separately as I_{Tj}.

Unser's formulation is applied to obtain the normalized LBP difference histogram $\hat{P}$ and the local homogeneity value h of the region. $\hat{P}$ is an LBP difference histogram because it is computed from a transformed image. Then, there are two types of feature vectors, X_{i} computed from I_{Ti} and X_{j} computed from I_{Tj}

$$X_i = [Position_i, h_i, \hat{P}_i]$$
$$X_j = [Position_j, h_j, \hat{P}_j]$$

where, sub-indexes i and j stand for crops at different scale.

Clustering Procedure Once the feature vectors are computed, there are three possible cases to be performed by the clustering procedure. In case A only X_{i} is found, in case B both X_{i} and X_{j} exist, and in case C only X_{j} is available. Two revisions are made during clustering of the feature vectors. The first one groups

all vectors into different classes, while the second one, ensures the vectors are grouped into the best possible class.

The first revision begins for cases A and B using X_{i}, and for case C using X_{j}. The homogeneity value h_1 of the first vector is compared with the other homogeneity values h_n of the set. The difference between them is taken according to

$$d(n) = h_1 - h_n; (n = 1, ..., N) \tag{3}$$

and where the smallest $d(n)$ is found, both $\hat{P}$ are compared using (2). If a certain similarity percentage is reached, both vectors are grouped into one class.

In the next iteration, a different vector is inspected using the same method:

1. Comparison of homogeneity values: the vector under test is compared using (3) against the rest of vectors that have not been grouped, and against to all the groups previously created.
2. Similarity revision of histograms: the vector under test is compared using (2) against the vector, and against the group where the smallest $d(n)$ is found.
3. Merging of feature vectors: if the desired similarity value is reached, the inspected vector is grouped where the similarity is maximum. If it is not reached for any group or any vector, a new cluster containing only the inspected vector is created.

Before creating a new class, a similarity test is made using (2) and the average $\hat{P}$ of the clusters. If this new class is similar in certain percentage to an existing one, then both classes are grouped, if not, a new cluster is made. The goal is to avoid having repeated clusters. Each time a new vector is merged into a certain group, the average values of the group are updated. The process is repeated until all vectors are grouped into one and only one cluster.

The second revision occurs after all the vectors have been grouped. Here, the goal is to make sure that every vector is classified into the best possible cluster. This is achieved by comparing every $\hat{P}$ of each feature vector with the average $\hat{P}$ of each cluster, using (2). Then, every vector is stored in the group where the similarity is maximum. For case B, is here where X_{j} is assigned to one class. Since the average values of the clusters are updated during this revision, another similarity inspection is performed in order to avoid having repeated classes. Using the average $\hat{P}$ of the clusters and (2), the groups are considered as belonging to the same class if they are similar up to a certain percentage; higher values indicate it is necessary to merge the groups. When they are not similar enough, both groups remain.

During the clustering process, a threshold for homogeneity values is set. If any h is equal or bigger than the threshold, the region with this vector is considered as non-texture and the vector is not considered for a cluster creation. It is stored in an extra class. Also, during this process, a matrix containing which vector belongs to each class is created, using the original position key codes.

Conformation of the Output Image To create the output image I_{D} of the detection stage, the average h of the groups are sorted in descendent order;

leaving the most textured groups first, and the more homogeneous at the end. Then, every vector gets as gray level value x the number of its cluster, where $x = 1, 2, \ldots, X$. I_{D} will have the most textured region in black color, the textured regions in gray intensities, and the more homogeneous in lighted color. The regions considered as non-texture are painted in white.

3.2 Refinement of the Detected Regions

During the refinement, every textured region is inspected individually, following the methodology described before. Thus, if N classes were created, N inspections are performed. Each inspection includes an splitting of the input image, a computation of feature vectors and a clustering procedure. At the end, if more classes are created, a general revision is performed. Information about the final clusters is used to conform the output image. The cluster containing the non-texture regions is excluded from this inspection.

1. Splitting of I_{in} using I_{D} as reference: a number of images I_{b} is obtained, according to the size of the textured region under inspection, by using the same partition window and the same displacement distance, as before. The direction of this displacement is now determined by I_{D}.
2. Computation of the features vectors: after I_{b} is transformed into an LBP space image I_{Tk}, an uniformity examination over this region is not necessary. Then, feature vectors X_{k} are composed as described in Section 3.1, and, every X_{k} has the same scale.
3. Clustering procedure: as explained in Section 3.1, the clustering procedure is performed using X_{k}. The similarity percentages are more restrictive than those in the detection stage, seeking to discriminate among different types of texture.

In order to avoid having repeated classes, a general revision is performed after the N inspections. This is made by comparing every $\hat{P}_{\mathrm{k}}$ with the average $\hat{P}$ of every cluster. Then, X_{k} is stored in the group whose similarity is maximum. Again, the cluster containing the non-textured regions is excluded in this revision. A matrix containing which region belongs to each class is created, using the original position key codes of X_{k}.

For the creation of the output image I_{R} of the refinement stage, the same procedure described in Section 3.1 is followed. Later, to create the output image I_{out} of the system, two majority filters of size 3×3 pixels are applied to I_{R}, in order to eliminate any spurious regions created in the process. A majority filter assigns the most repeated values in the neighborhood to the pixel under inspection.

4 Experimental Work and Discussion

Before evaluation, four parameters were determined for the detection process: 1) Similarity between vectors of 0.850 (85%), 2) Similarity between clusters of 0.80

(80%), 3) Similarity between final clusters of 0.950 (95%), and 4) Homogeneity threshold of 0.875. For the refinement stage, a similarity between vectors of 0.950 (95%) and a similarity between clusters of 0.930 (93%) were used. Other considerations were assumed: partition windows of size 32×32 pixels to collect data with a displacement distance of 16 pixels, cosine amplitude as a resemblance metric between two histograms, and Unser's estimation to calculate local homogeneity values and local difference histograms.

A texture mosaic containing textured and non-textured regions was composed in order to evaluate the detection of homogeneous zones in an image. From Brodazt photographic album [11], texture D68 and texture D16 were selected. In Fig. 2 we can see the original image, its ground truth reference, and the results for texture detection, and for texture refinement before and after filtering. As we can see, the result is improved on each step of the approach, as expected. The homogeneous region is well detected, and both textured regions are segmented. However, texture 4 and texture 5 are mistakenly created. The errors in the frontiers between textures are due to the mix of information from different textured zones. The region can be assigned to the most present texture in the crop, or, it can be identified as another type of texture. In Table 1, a confusion matrix of segmentation results upon Texture Mosaic 1 is shown. Here, values are given in percentage values. A 91.70% of correct assignation was reached when comparing pixel correspondence between the output image and its ground truth reference.

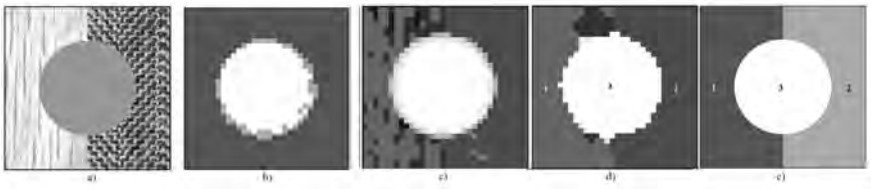

Fig. 2. Texture mosaic 1: a) I_{in}, b) I_{D}, c) I_{R}, d) I_{out}, and e) ground truth reference.

Table 1. Confusion matrix of texture mosaic 1

Class	1	2	3	TOTAL
1	29.20	0.0	0.0	29.20
2	0.29	35.89	0.34	36.52
3	2.38	0.69	26.62	29.69
4	3.91	0.0	0.0	3.91
5	0.68	0.0	0.0	0.68
TOTAL	36.46	36.58	26.97	100

Two more texture mosaics [12] were evaluated using the texture segmentation system. In Fig. 3, a texture mosaic consisting of two different types of texture is shown, together with its ground truth reference and the results of each step. Two textured regions were obtained as expected, and a 94.5% of correct assignation when comparing pixels correspondence was reached, see Table 2. In Fig. 4 a texture mosaic consisting of five different textures is shown. All textured regions were segmented, however, texture 3 is being confused with texture 2. Even tough a 69.30% of global performance was reached, more than 91.0% of texture 3 and texture 4 was correctly assigned when comparing pixels correspondence, see Table 3. If a different resolution is used, a displacement distance of 8 pixels, an improved result is obtained (refer to Fig. 5). Here, well defined elements are shown; a global performance of 66.45% was reached, and texture 4 is 99.20% correct assigned when comparing pixels correspondence.

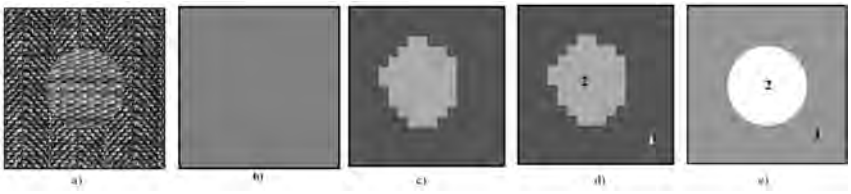

Fig. 3. Texture mosaic 2: a) I_{in}, b) I_{D}, c) I_{R}, d) I_{out}, and e) ground truth reference.

Table 2. Confusion matrix of texture mosaic 2

Class	1	2	TOTAL
1	77.29	2.40	79.69
2	3.10	17.21	20.31
TOTAL	80.39	19.61	100

A last evaluation was performed over a natural scene, using the image 108073 from the Berkeley database [13]. In Fig. 6 the original image and the evaluation results are shown. Three textured regions were detected (see Fig. 6b) and seven textured regions were segmented.

5 Conclusions

A texture segmentation approach is proposed. The methodology is divided in two tasks: a texture detection and a refinement of detected textured regions. Texture segmentation is efficiently performed using an input image transformed

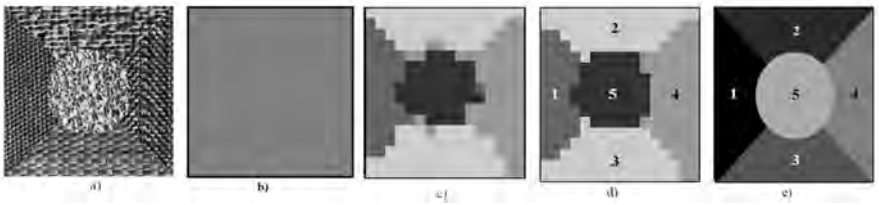

Fig. 4. Texture mosaic 3: a) I_{in}, b) I_{D}, c) I_{R}, d) I_{out}, and e) ground truth reference.

Table 3. Confusion matrix of texture mosaic 3

Class	1	2	3	4	5	TOTAL
1	15.63	0.0	0.0	0.0	0.0	15.63
2	3.19	18.69	18.46	0.71	1.53	42.58
3	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	1.05	1.31	19.19	2.28	23.83
5	1.48	0.69	0.01	0.0	15.80	17.97
TOTAL	20.29	20.42	19.78	19.90	19.61	100

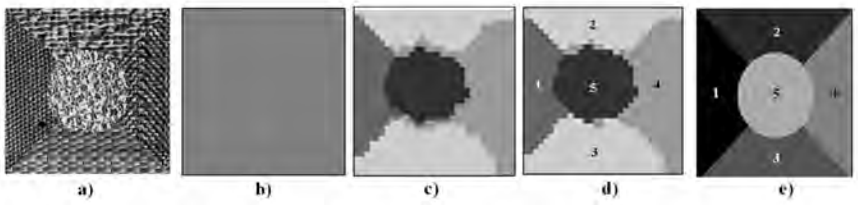

Fig. 5. Texture mosaic 2:a) I_{in}, b) I_{D}, c) I_{R}, d) I_{out}, and e) ground truth reference, using a displacement distance of 8 pixels.

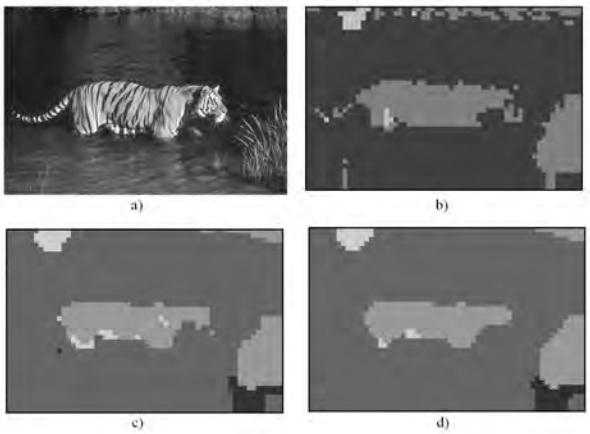

Fig. 6. Image 108073 [13]: a) I_{in}, b) I_{D}, c) I_{P}, and d) I_{out}.

into an LBP space; having then local indicators of texture units. Then, local homogeneity values indicate the presence of textured and non-textured regions; while LBP difference histograms improve the creation of classes and allow the discrimination between different textures. Texture segmentation is a refinement process after texture detection.

Experimental work shows good texture segmentation over Brodatz textures, along with good identification of non-textured regions. Texture segmentation is also performed on natural scenes with satisfactory results. Texture segmentation can be improved by decreasing the displacement distance of the partition window.

Even though the creation of prototypes and the clustering procedure are unsupervised processes, some parameters and considerations have to be determined empirically before texture segmentation. Automatic determination of them would be desirable in future works for performance improving and generalization of the system. Texture classification can be achieved by designing a learning process accordingly.

Acknowledgements

Gemma S. Parra-Dominguez gratefully acknowledges the Mexican National Council for Science and Technology (CONACyT) for the financial support through the scholarship grant number 302076/290564.

References

[1] M. Tuceryan and A. K. Jain, "Texture analysis," in *The Handbook of Pattern Recognition and Computer Vision*, L. F. P. C. H. Chen and P. S. P. Wang, Eds.

World Scientific Publishing, 1993, ch. 11, pp. 235–276.

[2] X. Qing, Y. Jie, and D. Siyi, "Texture segmentation using LBP embedded region competition," *Electronic Letters on Computer Vision and Image Analysis*, pp. 41–47, 2005.

[3] K. Karu, A. K. Jain, and R. M. Bolle, "Is there any texture in the image?" *Pattern Recognition*, vol. 29, no. 9, pp. 1437 – 1446, 1996.

[4] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 3, no. 6, pp. 610 –621, nov. 1973.

[5] M. Unser, "Sum and difference histograms for texture classification," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-8, no. 1, pp. 118 –125, jan. 1986.

[6] T. Ojala, M. Pietikinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51 – 59, 1996.

[7] T. Maenp and M. Pietikinen, "Texture analysis with local binary patterns," in *The Handbook of Pattern Recognition and Computer Vision*, 3rd ed., C. H. Chen and P. S. P. Wang, Eds. World Scientific Publishing, 2005, ch. 1, pp. 197–216.

[8] X. Liu and D. Wang, "Image and texture segmentation using local spectral histograms," *Image Processing, IEEE Transactions on*, vol. 15, no. 10, pp. 3066 –3077, oct. 2006.

[9] M. Savelonas, D. Iakovidis, and D. Maroulis, "An lbp-based active contour algorithm for unsupervised texture segmentation," *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 2, pp. 279 –282, 2006.

[10] E. Tekeli, M. Cetin, and A. Ercil, "A local binary patterns and shape priors based texture segmentation method," *Signal Processing and Communications Applications, 2007. SIU 2007. IEEE 15th*, pp. 1 –4, jun. 2007.

[11] U. of Southern California. USC-SIPI image database. Http://sipi.usc.edu/database/database.cgi?volume=textures.

[12] R. Trigve. Trigve Randen. Http://www.ux.uis.no/ tranden/.

[13] P. Arbelaez, C. Fowlkes, and D. Martin. The Berkeley segmentation dataset and benchmark. Http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/.