Three-dimensional reconstruction of textured parallelepipeds from digital video

Israel Vite Silva and Luis Gerardo de la Fraga

Computer Science Section
Department of Electrical Engineering. CINVESTAV
Av. Instituto Politécnico Nacional 2508. 07300 México, D.F.
E-mail: fraga@cs.cinvestav.mx

Abstract. In this article, a three-dimensional automatic reconstruction system is presented. The proposed approach acquires digital video with perspective projection of any textured parallelepiped, processes the video frame, reconstructs and visualizes the three-dimensional object. The system is divided in three modules: (1) The digital processing module extracts a video frame from a digital video camera. Next, it processes a video frame by means of a smoothing filter and an edge detector. After that, it identifies the vertices of the parallelepiped. (2) The vision module uses the vanishing points properties and the vertices previously detected to estimate the intrinsic and extrinsic parameters of the camera. Then, using an algorithm proposed here, the dimensions of the parallelepiped are calculated. Finally, the texture is extracted. For us, the texture is a short phrase of text, and then must be corrected geometrically. (3) The visualization module shows the recovered three-dimensional object using OpenGL. The proposed approach was tested using different parallelepipeds with excellent results.

Keywords: Video processing, vanishing points, 3D reconstruction from a single image.

1 Introduction

In the real world many objects exist that have a parallelepiped form, for example: buildings, containers, boxes, among others. These parallelepipeds present straight lines and right angles that are used to obtain vanishing points. Such points are the intersection of the parallelepiped lines projected to the infinite.

Caprile and Torre [1] use the vanishing points to calculate some intrinsic and extrinsic parameters of a camera. When calculating all the parameters say that the camera has been calibrated, therefore, it knows the position and direction in a three dimension for each object points allowing reconstruct the object in a three dimension.

© *A. Gelbukh, C. Yáñez Márquez, O. Camacho Nieto (Eds.)*
Advances in Artificial Intelligence and Computer Science
Research on Computing Science 14, 2005, pp. 107-115

In order to find all the camera parameters, Cipolla [2] uses multiple images and epipolar geometry, the problem with this approach is that it needs at least two views for finding all camera parameters, in addition, the user must define lines to find the vanishing points. Liebowitz and Zisserman [3] use stratified reconstruction. First, they determine the projective reconstruction by means of vanishing points, then, estimates an affine projection and finally they obtain the metric projection and use it to reconstruct the three-dimensional object. Nevertheless, this approach also needs at least two views and it is frequently unstable when performing all these numerical calculations.

Jelinek [4] models the object with a polyhedron, doing a correspondence between the characteristics of the object and the image. With this information, estimates the camera projection model and the object dimensions. However, the user must interact with this algorithm to define the correspondence between the edge of the model and the edge of the image object, in addition, it does not completely exploit the vanishing points properties, by this reason they must refine the object dimensions by means of optimization nonlinear techniques.

Several systems have been revised to reconstruct objects. However they need certain parameters to be specified by the user (p. e. the user manually selects set of lines or points) to perform the reconstruction.

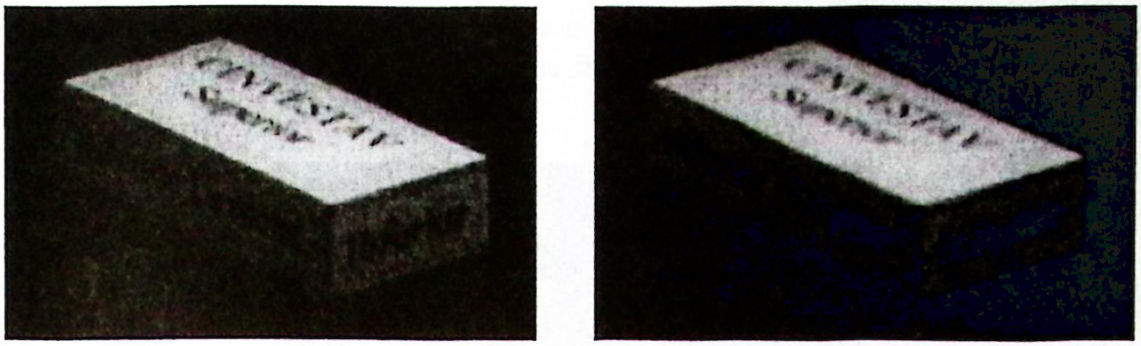
The proposed system does not need any interaction with user. It only needs that he/she provides a video stream of the parallelepiped in a perspective projection, to reconstruct the object automatically. First, the system extracts video frame and process the image to obtain the parallelepiped's edges, then, it calculates the parallelepiped's vertices using a variation of the Hough transform. After that, it determines the vanishing points and calculates most of the camera parameters. With these camera parameters and the proposed algorithm based on [4], the system recovers the parallelepiped dimensions and the rest of the camera parameters. To recover the texture, the system transforms each three-dimensional point that composed each one of the visible faces of the parallelepiped, to the video frame. Finally, the reconstructed object and its texture are visualized using OpenGL.

2 System description

The three-dimensional system reconstruction of a parallelepiped is performed in three modules: the digital processing module, the vision, and the visualization modules. They will be described in the following subsections.

2.1 Digital processing module

The system acquires a video stream of a parallelepiped in perspective projection from a digital video camera. Then a single video frame is extracted with size 720 × 576 pixels, using *libdv* library [5]. The frame is converted to a gray scale image using the formula of Craig. An example of this kind of image is show in Fig. 1(a).



(a) (b)

Fig. 1. (a) A gray scale image of a video frame of a parallelepiped A, (b) corresponding noiseless video frame.

It is very probable that the video frame presents noise and it affects the subsequent steps of the process, therefore it is necessary to use a smoothing technique [6]. This technique reduces the abrupt changes in the image: a new image $g(x, y)$ is generated, where the gray level in each point (x, y) is the average of the pixels values in a neighborhood of size 3×3 pixels.

The next step is the segmentation of the smoothing image. We use an edge detection to extract the lines of the parallelepiped (see Fig. 2. This step uses the Sobel operators [6].

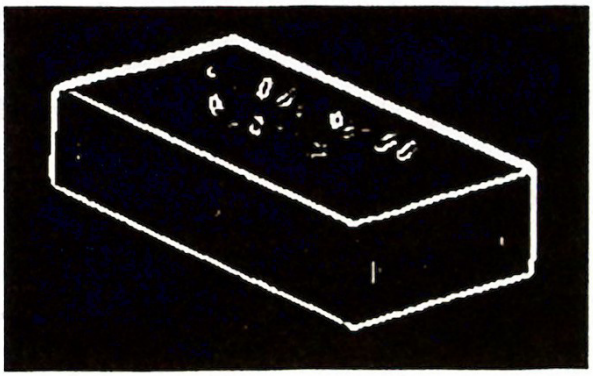


Fig. 2. Segmented video frame of parallelepiped A.

In order to recognize the parallelepiped vertices, the algorithm proposed by Fei Shen and Wang [7] was used. It is based on the following definitions: a vertex must belong to an edge and must have at least two line intersections. This algorithm uses a simplified Hough transform, and by this reason, it is possible to be applied in real-time systems. Also, the simplified Hough transform detect lines, so the texture over the parallelepiped's faces does not must composed by lines larger than the shorter parallelepiped's edge in order to extract the texture correctly. To obtain satisfactory results and to reduce the searched area for each vertex, the system applies to the previously segmented image, a skeleton algorithm that reduces the contour thickness to one pixel. Here we used the

skeleton algorithm proposed in [8], based on a 8×8 neighborhood, which identifies essential pixels that connect the structure of the parallelepiped.

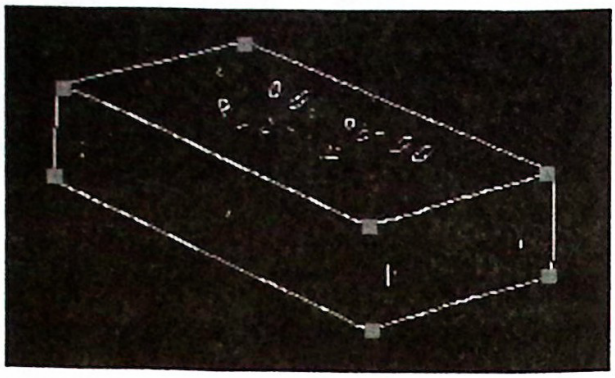


Fig. 3. Skeleton of parallelepiped A. The small squares show the recognized vertices.

The vanishing points are estimated using a sorted version of the vertices previously found. The sorting allows combine pairs of vertices that form parallel lines and they are intersected in a vanishing point. First we find all the joined vertices: we test if a line between a vertex and all the other vertices exists. Second, for each found line (a pair if joined vertices), its slope is calculated. All slopes can be sorted in three groups, each one identifying a coordinate axis. Finally, when the lines inside a group are projected to the infinite, these tend to be united but not in the same point; by this reason, to find the three vanishing points a weighed average is calculated.

2.2 Vision module

According to the camera pinhole model, the relation between a point in an image with perspective projection and the three-dimensional world is described by the equation:

$$\lambda_i \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} \tag{1}$$

where $(x_i, y_i, z_i, 1)$ is the i-point in homogeneous coordinates in the three-dimensional world, $(u_i, v_i, 1)$ is the i-point in homogeneous coordinates in the image, λ_i is an arbitrary scale, and $\mathbf{P}$ is the projection matrix,

$$\mathbf{P} = \mathbf{A}[\mathbf{RT}] \tag{2}$$

where $\mathbf{R}$ is a 3×3 rotation matrix, $\mathbf{T}$ is 3×1 translation vector (known as camera extrinsic parameters) and $\mathbf{A}$ is a 3×3 matrix that relates the coordinates of the image to the coordinates of the camera (known as intrinsic parameters):

$$\mathbf{A} = \begin{bmatrix} f_x & \alpha & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

where f_x, f_y are scale factors, α is the skew parameter, and o_x, o_y is the pixel where is the intersection of the optical axis with the image plane.

Based on [2] and [1], the three vanishing points have the following restriction,

$$\begin{bmatrix} \lambda_1 u_1 & \lambda_2 u_2 & \lambda_3 u_3 \\ \lambda_1 v_1 & \lambda_2 v_2 & \lambda_3 v_3 \\ \lambda_1 & \lambda_2 & \lambda_3 \end{bmatrix} = \mathbf{P} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{4}$$

where λ_i is the scale that initially is unknown, and u_i, v_i are the vanishing points coordinates.

Under assumption of known aspect ratio ($f = f_x = f_y$) and zero skew ($\alpha = 0$) in the matrix **A**, the Eq. (4) can be expressed in six linear equations [2] that recovered scale factor f, projection center o_x, o_y and lambdas λ_i. With these parameters and using Eqs. 2 and 4, the system calculates the rotation matrix.

To recover the parallelepiped dimensions, the system uses an algorithm proposed in [4], which models the object by a polyhedron, where coordinates of the vertices are expressed by linear functions.

It means that exists a set of $3 \times n$ matrices $(K_1, K_2..., K_n)$, where the position of the i-th vertex is given by $K_i\gamma$.

In the proposed algorithm, a vertex is translated to origin. The rest of the vertices are transformed, using the new origin, to linear functions of the vector dimension $\gamma = (LHD)^t$, for example, in figure 4,

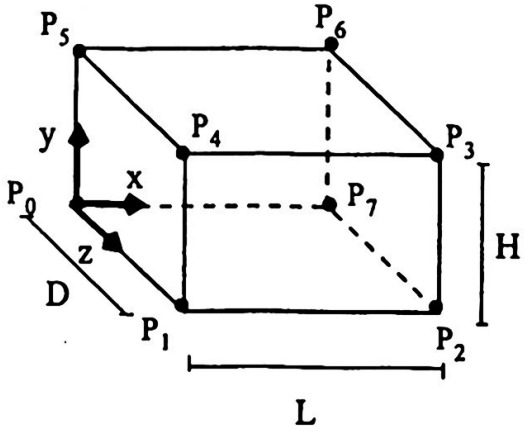


Fig. 4. Parameterized linear model

the points P_1 y P_3 are represented by:

$$P_1 = \begin{pmatrix} 0 \\ 0 \\ D \end{pmatrix} = \begin{pmatrix} 0\,0\,0 \\ 0\,0\,0 \\ 0\,0\,1 \end{pmatrix} \gamma;$$

$$P_3 = \begin{pmatrix} L \\ H \\ D \end{pmatrix} = \begin{pmatrix} 1\,0\,0 \\ 0\,1\,0 \\ 0\,0\,1 \end{pmatrix} \gamma.$$

where L is width, H is height and D is depth of the parallelepiped. The projection of each vertex in the image should lie along the lines and can be expressed as:

$$l'^{t}_{jk} A(RK_j\gamma + T) = 0 \tag{5}$$

where l'^{t}_{jk} is a line in homogeneous coordinates that goes from vertex j to vertex k. The line in homogeneous coordinates is described by,

$$ax + by + c = 0; \qquad l = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \tag{6}$$

The cross product between the vertices is used to calculate these homogeneous lines. This operation performs some multiplications that could result in a slowly process. By this reason, we propose to generate the lines in homogeneous coordinates, from the origin $(0,0)$ to the vertex (x_j, y_j). This gives us a line $(-y_j, x_j, 0)^t$ which does not calculate the cross product, eliminates the c term in Eq. (6), and reduces the number of operations to solve the homogeneous linear system of equations.

When we apply all the vertices to the equation 5, a linear system of equations is obtained. This linear system is solved by SVD [9], obtaining the parallelepiped dimensions. Also, the translation vector t is recovered.

Finally, using Eq. 1 with intrinsic and extrinsic parameters already found, the system extracts the visible textures of the parallelepiped. This is performed by linking each three-dimensional point with its corresponding pixel in the image on each parallelepiped face. Therefore the texture is obtained without any rotation, and it can be used to recover some kind of information that the face has (in our case, it is text).

2.3 Visualization module

This module visualizes the recovered three-dimensional object, and is a graphical interface developed with OpenGL. The user can interact with it changing the position, direction, scale and illumination. Also, the GUI does texture mapping, showing the object in a realistic form.

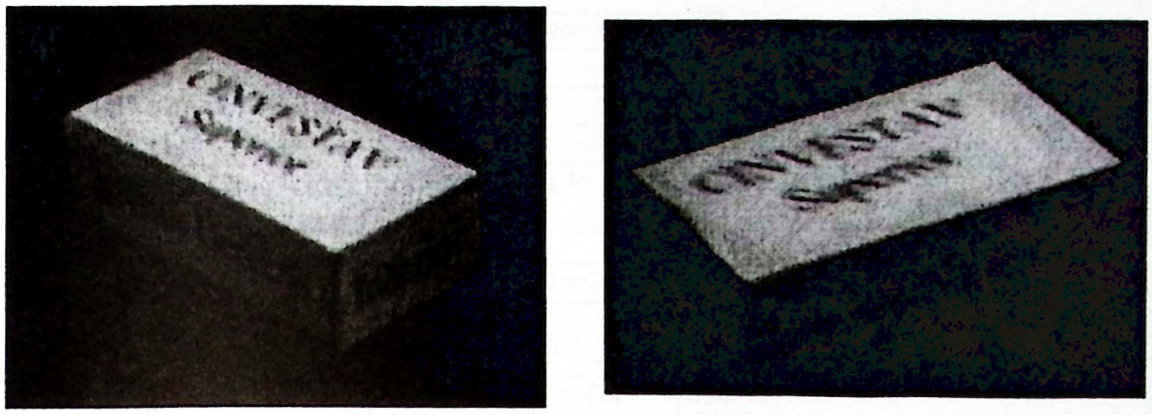
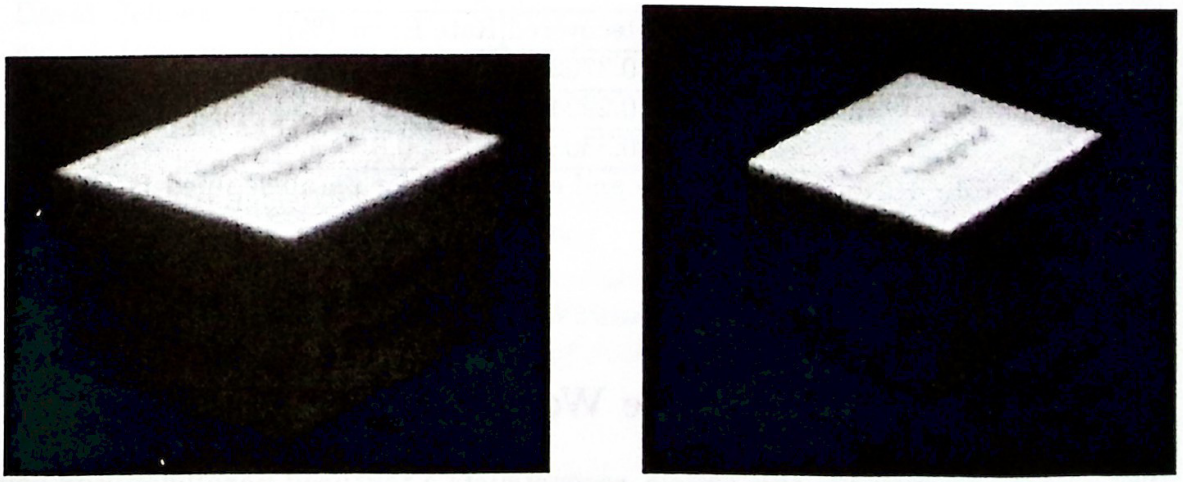


Fig. 5. Model visualization of parallelepiped A. The right image shows a hidden face of the reconstructed parallelepiped. Of course, for that hidden face none texture was assigned.

3 Results

In Fig. 5 we can see two views of the three-dimensional reconstruction, of a parallelepiped labeled 'A', obtained from the single video frame showed in Fig. 1. The right image in Fig. 5 shows a side of the parallelepiped that can not be seen in the frame in Fig. 1.

In figure 6(a) is shown the video frame of other parallelepiped, labeled 'B', and in figure 6(b) appears the object reconstructed with our system.



(a) (b)

Fig. 6. (a) Video frame of parallelepiped B. (b) Visualization of the object B.

In the tables I and II are shown the real measurement and the recovered value in an arbitrary scale of parallelepipeds A and B presented in figure 1 and figure 6 respectively.

Side	Real (cm)	Recovered
Width	4	0.280808
Height	2	0.139793
Depth	8	0.607545

Table 1. Values of parallelepiped A.

Side	Real (cm)	Recovered
Width	9	0.24289
Height	6	0.196395
Depth	7.5	0.216665

Table 2. Values of parallelepiped B

In order to compare real dimensions and recovered dimensions, both are normalized and the error ratio is shown in the tables III and IV.

Side	Real	Recovered	Rate Error (%)
Width	0.285714	0.27312	1.25
Height	0.142857	0.135966	0.68
Depth	0.571428	0.590913	1.94

Table 3. Normalized values and error ratio for parallelepiped A.

Side	Real	Recovered	Rate Error (%)
Width	0.4	0.370287	2.97
Height	0.266666	0.299405	3.27
Depth	0.333333	0.330307	0.3

Table 4. Normalized values and error ratio for parallelepiped B.

4 Conclusions and Future Work

The system described in this article, reconstructs a textured parallelepiped using a single video frame acquired by a digital video camera (using a Cannon DM-GL2). The camera does not need previously calibration.

All modules do not need any fine-tune by the user. The user only takes a video stream of the parallelepiped using the video camera and the system extracts a single video frame to recover and visualize the three-dimensional object.

The system can reconstruct another type of objects, but it should replace the parallelepiped model by the object model wished; the restriction is the object must be a polyhedron.

The total execution time of the system is smaller than a second, in a computer Pentium 4 to 2.0 GHz under GNU/Linux.

The obtained results that were presented have a high accurate because of the small error ratio, between the real dimensions of the object and the dimensions obtained; also, the extracted texture of the image allows to visualize it as it is in the real world, even though the texture is not totally visible in the processed frame.

Our proposed system can be applied to recover text short phrase regions and correct its orientation in any parallelepiped or an object that it can be modeled by a polyhedron, thus the characters are straightened and these could be recognized. This can be used to recognize automatically cars plates or the containers legends in a shipping port.

Acknowledgments

This work has been partly supported by grant 45306 from CONACyT, México.

References

1. B. Caprile and V. Torre. Using vanishing points for camera calibration. *International Journal of Computer Vision*, (4):127–140, 1990.
2. R. Cipolla, T. Drummond, and D. Robertson. Calibration from vanishing points in image of architectural scenes. In *The 10th British Machine vision conference*, 1999.
3. D. Liebowitz and A. Zisserman. Metric rectification for perspective images of planes. *CVPR*, pages 482–488, 1998.
4. David Jelinek and Camillo J. Taylor. Reconstruction of linearly parameterized models from single images with camera of unknown focal length. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(7):767–773, Julio 2001.
5. Charles Krasic and Erik Walthinsen. http://libdv.sourceforge.net/.
6. Rafael C. Gonzalez. *Digital Image Processing.* Addison Wesley, 1996.
7. Han Wang Fei Shen. Corner detection based on modified hough transform. *Pattern Recognition Letters*, (23):1039–1049, 2002.
8. J. R. Parker. *Practical Computer Vision using C.* Wiley, 1994.
9. William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C++: The Art of Scientific Computing.* Presss Syndicate of the University of Cambridge, 2002.