A Hough-based method for hatched pattern detection in maps and diagrams*

Josep Lladós, Enric Martí
Computer Vision Center, Dept. Informàtica.
Universitat Autònoma de Barcelona,
08193 Bellaterra (Barcelona), Spain
{josep,enric}@cvc.uab.es

Jaime López-Krahe
Dép. Informatique, GRAII, Lab. ai/mime.
Universit Paris 8,
Saint Denis, 93526 Paris CEDEX 02, France
lopez@ai.univ-paris8.fr

Abstract

A hatched area is characterized by a set of parallel straight lines placed at regular intervals. In this paper, a Hough-based schema is introduced to recognize hatched areas in technical documents from attributed graph structures representing the document once it has been vectorized. Defining a Hough-based transform from a graph instead of the raster image allows to drastically reduce the processing time and, second, to obtain more reliable results because straight lines have already been detected in the vectorization step. A second advantage of the proposed method is that no assumptions must be made a priori *about the slope and frequency of hatching patterns, but they are computed in run time for each hatched area.*

1 Introduction

The problem of hatched pattern detection is an important concern in the field of document analysis [1, 2, 4]. Documents belonging to different domains often contain hatched regions with particular meanings. An efficient extraction of hatched areas allows to drastically reduce the information and to focus further interpretation on the remaining lines. Two attributes characterize a hatched pattern, namely, the orientation of the filling straight lines, and the distance between two consecutive ones. This suggests that an analysis method able to transform the input image to a parameter space where these two attributes are outlined would be convenient. This is just the Hough idea. Although the Hough transform was initially designed to detect straight lines, a lot of interesting extensions have been proposed to detect parametrized shapes. The key idea of the Hough transform is that spatially extended patterns are transformed into a parameter space where they can be represented in a spatially compact way. Thus, a difficult global detection problem in the image space is reduced to an easier problem of peak detection in a parameter space. The reader is referred to references [3] and [5] for a comprehensive review of the available Hough transform methods.

In a previous work [7] we proposed a graph-based Hough transform (GBHT) to detect hatched patterns representing walls in architectural drawings. The GBHT was designed to detect hatched areas with a particular layout, making an assumption of orthogonality between rectangular hatched regions and also assuming that the orientation and frequency of hatched patterns were constant in a document. In this work, we generalize the idea of the GBHT in order to detect hatched regions of any orientation, frequency and boundary. The proposed method has been illustrated with its application to cadastral city maps (see an example of a French city map in Fig. 1). The automated conversion of city maps to CAD has received a strong interest in the literature. Cadastral city maps usually have a standardized notation which makes suitable the use of an *a priori* knowledge to analyze them. One of the major components of city maps is hatched areas representing buildings. Thus, a central part of the city map recognition systems is devoted to extract all hatched areas from the document in order to make easier the recognition of the other elements (streets, parcels, text, etc.).

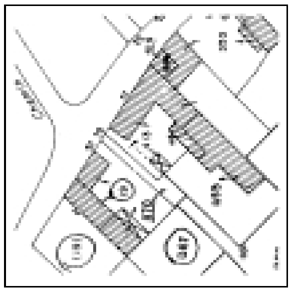

Figure 1. An example of a cadastral city map.

In the following section we describe a graph-based Hough transform designed to detect hatched patterns. Af-

*Work partially supported by project TAP98-0618 of Spanish CICYT

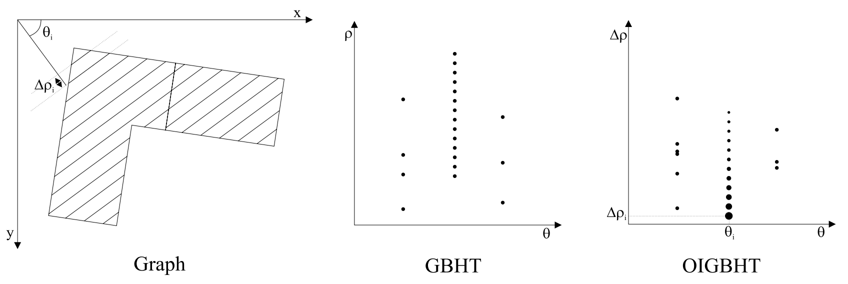

Figure 2. An example of the OIGBHT applied to detect hatched patterns.

terwards, in Section 3 some results of the proposed method are shown. Finally, Section 4 is devoted to the conclusions.

2 Hatched pattern detection by a graph-based Hough transform

A hatched pattern can be defined as a sequence of I straight lines in the image space $\{l_i = (\theta_i, \rho_i) : \theta_i = \theta_{i+1}, \rho_i + \Delta_\rho = \rho_{i+1}, i = 1, \ldots, I-1\}$ where Δ_ρ is a constant value which characterizes the spatial frequency of the hatched pattern. Since a hatched pattern consists of a set of parallel straight lines placed at regular intervals, when the SLHT is applied, a sequence of peaks which are vertically aligned and regularly placed is obtained. Thus, parallel straight lines could be detected by a double SLHT: first a SLHT applied to the image space, and a second SLHT from the parameter space in order to detect collinear peaks. However, a more intelligent processing can be done on the parameter space by computing the distances between pairs of points having the same θ values. This is what Pao et al. [8] called *signature* of the shape. A peak (θ, Δ_ρ) in the signature space will represent a hatched pattern characterized by the orientation and increment defined by the peak coordinates. A more detailed study about the effect of translation, rotation and scaling operations on the Hough space is reported in reference [6].

One of the drawbacks of the Hough transform is its hight computational load. Since the document is first vectorized, we use the information of the vectorial structure instead of transforming the original image. Thus, after a vectorization stage, the document is represented by an *attributed graph* whose nodes represent characteristic points (junction, inflexion and end points), and edges approximate segments by straight lines or circumference arcs. This graph is transformed to a Hough space where peaks represent hatched areas. Afterwards, graph edges belonging to the same hatched area are grouped and the boundary of the region is detected.

Taking into account that straight graph edges are attributed by the parameters θ and ρ that characterize the equation $\rho = x\cos\theta + y\sin\theta$ of the straight line passing through this edge, in [7] we defined the GBHT as follows:

Definition 2.1 Given an attributed graph $G = (V, E)$, we define its *Graph Based Hough Transform* as a function GBHT: $E \to [0, \pi] \times \mathbf{R}$ such that, for each straight edge $e \in E$ with attributes values θ_e and ρ_e, transforms e into a point (θ_e, ρ_e) in the θ-ρ parameter space.

If the GBHT is applied to the input graph, vertical alignments are obtained at orientations corresponding to hatched patterns. But we are interested in finding not only the direction of hatching segments, but also the difference in ρ between two consecutive ones. Based on the idea of the *signature* of a shape defined by Pao et al. [8] we define the *orientation-invariant graph-based Hough transform* (OIGBHT) as that which computes the distances between pairs of points of the parameter space having the same θ value. Formally it is defined as follows:

Definition 2.2 Let $G = (V, E)$ and GBHT be an attributed graph and its corresponding graph-based Hough transform, respectively. We define its *orientation-invariant graph-based Hough transform* (OIGBHT) as a function which maps each pair of vertically aligned points of the θ-ρ space to a point of a new parameter space θ-$\Delta\rho$, i.e OIGBHT: $[0, \pi] \times \mathbf{R} \times [0, \pi] \times \mathbf{R} \to [0, \pi] \times \mathbf{R}$ is defined as

$$\text{OIHBHT}(\theta_1, \rho_1, \theta_2, \rho_2) = \begin{cases} (\theta_1, |\rho_1 - \rho_2|) & \text{if } \theta_1 = \theta_2 \\ \text{undefined} & \text{otherwise} \end{cases}$$

Since the GBHT produces vertically aligned and regularly placed points in the parameter space when a hatched pattern is transformed from the image space, the OIGBHT will produce peaks in the θ-$\Delta\rho$ space characterizing these hatched patterns. The coordinates of a peak in the θ-$\Delta\rho$ space define the orientation and the increment of ρ between two consecutive edges of the hatched pattern. An illustrative example of an ideal case is displayed in Fig. 2. The edge orientation and frequency of each hatched pattern is detected by computing the OIGBHT from the input graph representing the document. Notice that hatched patterns can have different edge orientations and frequencies, it depends on the number of peaks detected in the θ-$\Delta\rho$ space.

Let $(\theta_H, \Delta\rho_H)$ be the coordinates of a peak, in the θ-$\Delta\rho$ space, which defines a hatched area. Those edges having

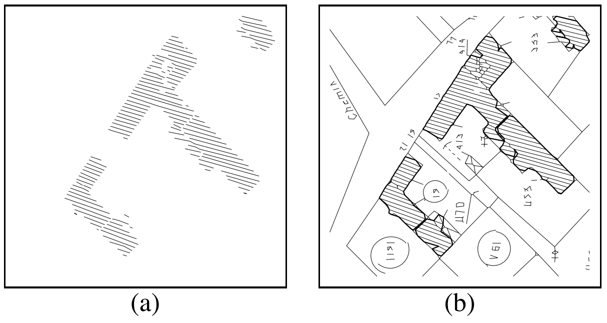

Figure 3. (a) Extraction of graph edges candidate to belong to hatched areas after performing the OIGBHT to the input graph, (b) boundaries of hatched areas.

a θ value equal to θ_H, with a certain tolerance range, and another parallel graph edge at distance $\Delta\rho_H$, also with a certain tolerance range, are selected from the input graph. Figure 3a shows the subgraph G_H obtained after applying this filtering procedure.

The final step consists in segment each hatched area, i.e. detecting its boundary defined as a sequence of graph vertices. First, the edges of the graph G_H of hatching edges are sorted according to their ρ value. An overlapping relation between edges is taken into account in the sorting procedure. Following an idea given in [1], a single-scan algorithm in the direction of the filling edges defines an order relation for each vertex of each segment:

Definition 2.3 Let θ_H be the orientation which defines a set of hatching edges. Following a scan direction perpendicular to θ_H, an edge is said to be *open* when the process reaches its origin vertex, and it is said to be *closed* when the process reaches its end vertex. A vertex which is open, in the scan direction, between the vertices of another segment is said to *overlap* the latter.

Considering this relation, given an edge e, the corresponding following node in the sorted list contains those edges which have a similar distance in ρ from e and overlap it. See at Fig. 4 for an illustrative example of sorted edges of a hatched pattern. In this example, two lists are built starting from the topmost edges, i.e. those edges that don't have any overlapped edge before.

The boundary of a hatched region is represented by a sequence of graph vertices $[v_1, \ldots, v_n]$; $v_i \in V$ which outline the region in a counter-clockwise sense. This boundary can be computed by a straightforward recursive procedure. Let L be a sorted list of hatching edges of a graph $G = (V, E)$, the function *boundary*(G, L, e) returns the sequence of vertices outlining a sorted list L of hatching edges starting at the edge $e \in E$. The idea is to concatenate at each recursive call $e_{v_1} \circ boundary(G, L, next(e, L)) \circ e_{v_2}$, where $\circ$ is the concatenation operator, e_{v_1} and e_{v_2} are the leftmost and rightmost vertices of e in the scan direction, respectively, and *next*(e, L) returns the following edges to e in the list L. See Fig. 5 for a trace of the algorithm in an ideal case.

A postprocessing procedure must be applied to the string of vertices given by the boundary detection algorithm. This step solves two possible problems: first, a region which results in two lists of edges and, second, the presence of gaps inside the hatched region. The first problem is solved by merging boundary strings having common substring. In Fig. 5 we can see that two sequences are obtained after boundary detection, AXB and CXD, which are merged in the string $AXDCB$. The second condition, the presence of gaps, is detected when the boundary string contains a re-

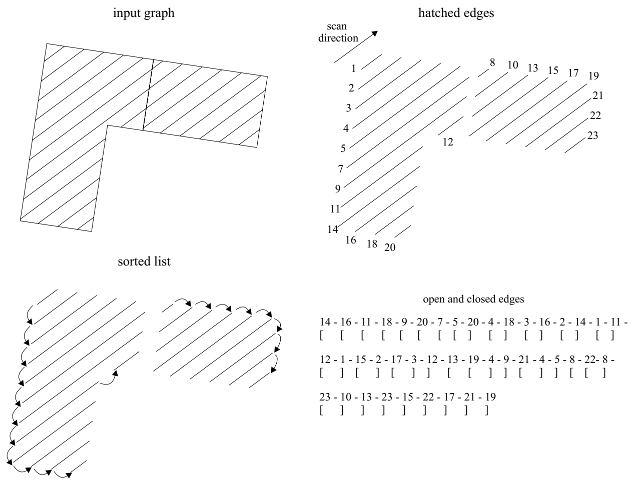

Figure 4. Sorting the edges of a hatched area according to their overlapping relation.

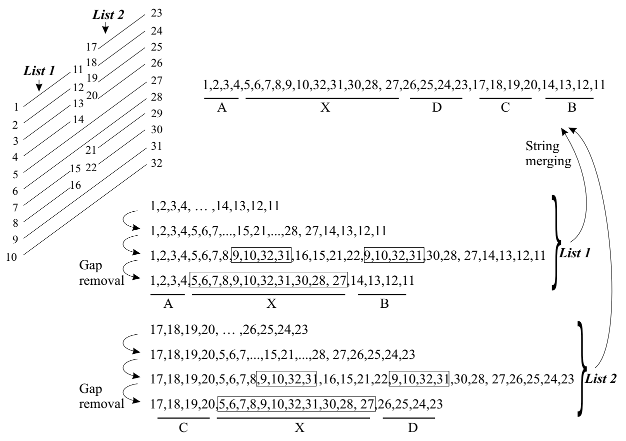

Figure 5. The boundary detection procedure.

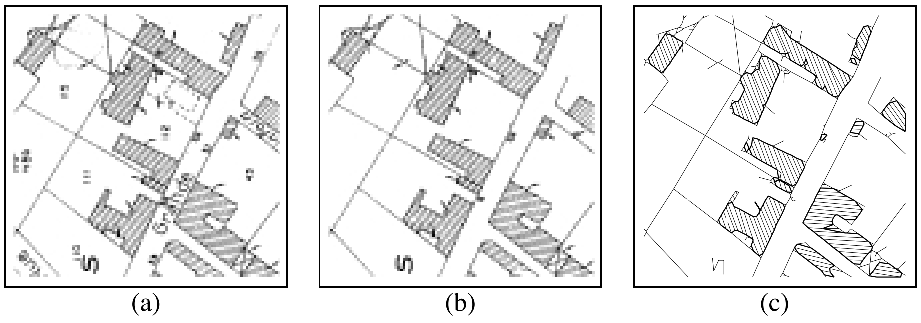

Figure 6. Example of cadastral city map analysis: (a) Input image, (b) Text-graphics separation, (c) Graph approximation and hatched areas discrimination.

peated substring (see Fig. 5), i.e. a string $AXBXC$ must be rewritten as AXC to remove the gap. A real example of boundary detection is shown in Fig. 3b.

3 Results

An additional example is presented in Fig. 6. Figure 6b illustrates a text-graphics separation step which has been applied to the original image: the connected components of the image of Fig. 6a have been labeled and separated in two classes accorting to their size. Thus, the small components (text and annotations) have been removed. The hatched pattern detection procedure has been applied to the image containing the large components. Figure 6c shows the graph approximation of the image and the boundaries of the hatched areas. Notice that the algorithm is able to discriminate hatched areas with different orientations. This illustrates that the orientation of the filling edges is not required to be previously known, but it is inferred in run time.

4 Conclusions

In this paper we have described a method to detect hatched patterns, one of the most extended graphical entities in maps and technical drawings. Instead of performing the recognition from the input raster image, hatched patterns are searched in an attributed graph that represents the document after being it vectorized. Considering that a set of parallel straight lines produce a set of vertically aligned points in the $\theta-\rho$ Hough space, we have defined a transform inspired in the Hough idea from the input graph to a $\theta - \Delta\rho$ parameter space where peaks represent hatched patterns.

Two advantages arise from the use of a graph-based transform instead of a raster image based transform. First, the processing time of the Hough transform is drastically reduced. Second, since straight lines have already been detected, and their attributes stored in the graph structure, the results are more reliable because potential distorsions due to isolated pixels are avoided. Moreover, the hatching detection module is a part of a map-to-CAD conversion system where raster-to-graph conversion is performed anyway. On the other hand, hatching detection algorithms make often some *a priori* assumptions about slope and frequency of hatching lines. In our case, these two attributes are computed from the Hough space, allowing to detect areas with different hatching values in the same document.

References

[1] D. Antoine, S. Collin, and K. Tombre. Analysis of technical documents: The REDRAW system. In H. Baird, H. Bunke, and K. Yamamoto, editors, *Structured document image analysis*, pages 385–402. Springer Verlag, 1992.

[2] L. Boatto, V. Consorti, M. Del Buono, S. Di Zenzo, V. Eramo, A. Espossito, F. Melcarne, M. Meucci, A. Morelli, M. Mosciatti, S. Scarci, and M. Tucci. An interpretation system for land register maps. *Computer*, 25(7):25–33, July 1992.

[3] J. Illingworth and J. Kittler. A survey of the Hough transform. *CVGIP: Image Understanding*, 44:87–116, 1988.

[4] R. Kasturi, S. Bow, W. El-Masri, J. Shah, G. J.R., and M. U.B. A system for interpretation of line drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):978–992, october 1990.

[5] V. Leavers. Which Hough transform? *CVGIP: Image Understanding*, 58(2):250–264, Sep 1993.

[6] J. Lladós. *Combining Graph Matching and Hough Transform for Hand-Drawn Graphical Document Analysis. Application to Architectural Drawings*. PhD thesis, Universitat Autònoma de Barcelona and Université de Paris 8, 1997.

[7] J. Lladós, J. López-Krahe, and E. Martí. A system to understand hand-drawn floor plans using subgraph isomorphism and Hough transform. *Machine Vision and Applications*, 10(3):150–158, 1997.

[8] D. Pao, H. Li, and R. Jayakumar. Shapes recognition using the straight line Hough transform: Theory and generalization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(11):1076–1089, Nov 1992.