Hand Drawn Document Understanding Using the Straight Line Hough Transform and Graph Matching

Josep Lladós[†], Jaime López-Krahe[††], Enric Martí[†]
[†] CVC - Departament Informatica. Universitat Autonoma de Barcelona, 08193 Bellaterra, Spain
[††] Departement Informatique, Université Paris 8, Saint Denis, 93526 Paris CEDEX 02, France
e-mail: josep@upisun1.uab.es, lopez@ai.univ-paris8.fr, enric@upisun1.uab.es

Abstract

This paper presents a system to understand hand drawn architectural drawings in a CAD environment. The procedure is to identify in a floor plan the building elements, stored in a library of patterns, and their spatial relationships. The vectorized input document and the patterns to recognize are represented by attributed graphs. To recognize the patterns a such, we apply a structural approach based on subgraph isomorphism techniques. In spite of their value, graph matching techniques do not recognize adequately those building elements characterized by hatching patterns, i.e. walls. Here we focus on the recognition of hatching patterns and we develop a straight line Hough transform (SLHT) based method in order to detect regions filled in with parallel straight lines. This allows not only to recognize filling patterns, but it actually reduces the computational load associated to the subgraph isomorphism computation. The result is then that the document can be redrawn by editing all the patterns recognized.

1. Introduction

CAD systems are a tool of great help to create and modify technical documents efficiently. The field of *document image analysis* allows to solve the reverse problem: converting paper-based drawings for their integration into a CAD environment. Here we focus on hand drawn floor plans for which we propose an alternative CAD system input technique that allows to create new CAD documents by means of hand drawn sketches.

Attributed relational graphs have lately become a powerful technique to represent and recognize line drawings. Recognition is performed using graph matching procedures that find a morphism between a model graph and an input graph representing the input document. Some outstanding examples can be found in [2, 5, 7]. Traditionally, graph matching uses backtracking tree search procedures that require an exponential time of computation in the worst case. To speed up these procedures and prune the search space some heuristic techniques have been proposed. Discrete relaxation [3] is a constraint propagation technique that allows the removal of inconsistent hypotheses before tree search expansion. Another obstacle is to deal with disturbed graphs obtained from noisy data. In hand drawn documents this problem is clearly noticeable because of the uncertainty induced by hand drawn strokes. In this case, inexact graph matching [1] must be performed.

In the current work we present a system to understand hand drawn floor plans recognizing the building elements (doors, walls, etc) and their topological properties. The input document is scanned and vectorized [9] giving an attributed graph. This graph is matched against model graphs representing the building elements. The matching process uses *AC4* algorithm [11] based on discrete relaxation techniques. To speed up the interpretation process, we use some structural properties of the document elements as heuristic criterion. As for walls, one building element characterized by a hatching pattern, we assume some structural properties and we search their features in the input graph. Walls' edges graph recognition can be used as a previous filter for the graph isomorphism process because their edges removal reduces search space meaningfully and thus it speeds up model matching.

SLHT has been often used to understand linear images. Some characteristic configurations in the original image (parallel edges, cross points, etc.) can be easily detected in the Hough space. In [10] SLHT is used to detect parallel straight lines in small scale images. In [13] cluster patterns are detected in Hough space to carry out an interpretation of 3-D polyhedral scenes. [12] uses a SLHT-based method to match continuous closed smooth curves. In our case, we use a SLHT-based method to detect graph edges belonging to the textured areas filling walls. The detection starts by transforming each straight graph edge to a parameter space. The

1015-4651/96 $5.00 © 1996 IEEE
Proceedings of ICPR '96

peaks in this parameter space, detected by means of a clustering process, give information about the textured areas.

Next you will see in section 2 a summary of graph matching process. Section 3 explains how to speed up the recognition by a SLHT-based method that finds textured areas. Section 4 shows the synthetized image after parameter estimation. The last section is devoted to the conclusion.

2. Model recognition by graph matching

The input line drawing and the patterns to recognize are represented using an attributed graph. The graph nodes represent the characteristic points (junctions or end points of lines). The attributes are their position, degree (number of lines joining in the node) and the angles between these lines. The graph edges represent the segments joining at characteristic points. The attributes are the lenght and, depending on whether the segment is adjusted by a straight line or a circumference arc, the parameters that characterize the respective equation. We will represent our graphs, using a standarized notation, by $G(V, E)$ where V is a set of nodes and E is a set of edges. Fig. 1a shows an input image to be recognized and Fig. 1b shows its graph representation. Starting from this representation, matching is carried out using inexact subgraph isomorphism techniques, that is, by finding the model graph representing the pattern to be recognized in terms of a subgraph of the input graph that approximates the input line drawing at best. The subgraph isomorphism problem is solved using *AC4*. To allow inexact graph matching, the matching algorithm takes into account accuracy errors involved in hand drawn design. Figs. 2c, 2d are respectively the results of subgraph matching with the patterns of Figs. 2a, 2b. See [8] for further details on isomorphism algorithm.

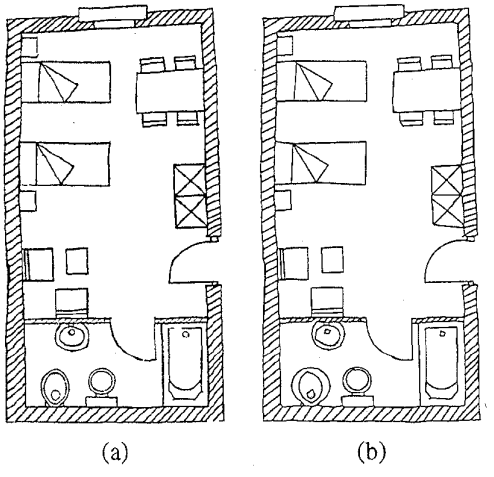


Figure 1. (a) Input image. (b) Graph.

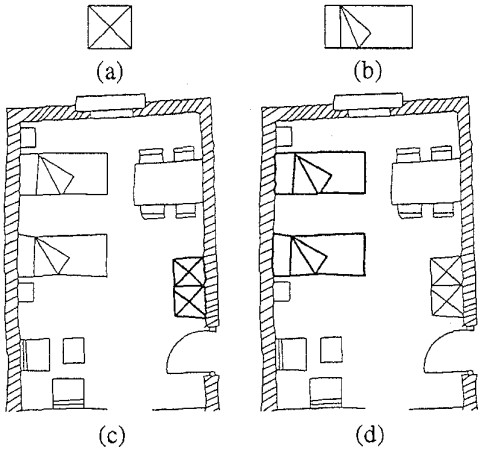


Figure 2. (a)(b) Some model graphs. (c) (d) Results after isomorphism.

3. Hatching patterns detection

Wall detection involves two purposes at the same time: recognising a building element which does not have a pattern graph and speed up the subgraph isomorphism process. Walls recognition will be carried out by detection of textured areas filled in by parallel straight lines even spaced. We make two additional assumptions for these regions: there are two directions allowed for the walls and these two directions must be orthogonal. It is possible to allow more than two directions. In this case, since the detection of dominant directions involves a supervised learning process, the number of directions should be an input parameter to the system.

Classical SLHT transforms each image point (x, y) into a set of points (θ, ρ) in a parameter space that fulfill the equation $\rho = xcos\theta + ysin\theta$. Detection of peaks in the parameter space constitutes a powerful method to detect straight lines in the input image. Several applications have been developped from this idea. A complete survey can be found in [4, 6]. With the purpose to detect textured regions forming walls we use the idea of SLHT in the following definition:

Definition 3.1 Given an attributed graph $G(V, E)$, we define its *Graph Based Hough Transform* with parameters θ and ρ as a function $\text{GBHT}_{\theta\rho} : E \rightarrow [0, \pi] \times \mathbb{R}$ that, for each straight edge $e \in E$ with attribute values θ_e and ρ_e, transforms e into a point (θ_e, ρ_e) in the θ-ρ parameter space. If the considered attributes of e are θ_e and its module m_e, we define equivalently $\text{GBHT}_{\theta m}$ in θ-m parameter space.

Fig. 3 shows $\text{GBHT}_{\theta\rho}$ and $\text{GBHT}_{\theta m}$ in an ideal case. $\text{GBHT}_{\theta\rho}$ gives, for each wall, two peaks with the same θ and a difference in ρ equal to the wall's width. $\text{GBHT}_{\theta\rho}$ gives also a sequence of points even spaced with the same

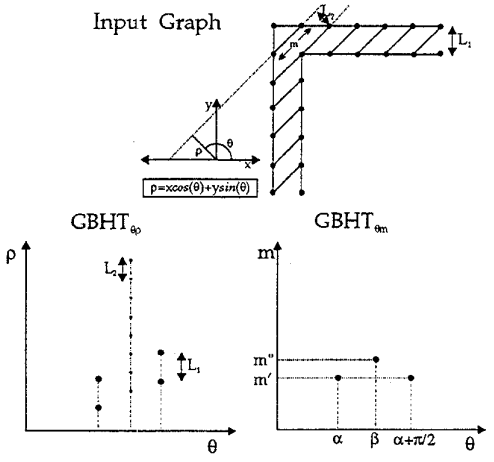
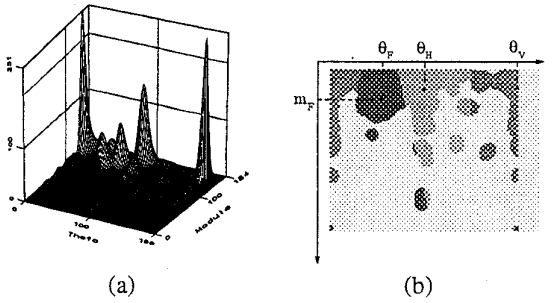


Figure 3. GBHT$_{\theta\rho}$ and GBHT$_{\theta m}$.

θ. GBHT$_{\theta m}$ gives three peaks in the parameter space. Two of them have the same m and a difference of $\pi/2$ in θ. These two peaks correspond to the orthogonal dominant directions. The third peak corresponds to the filling edges and the wall's width (L_1 in the figure) is calculable starting from this peak as it can be seen below. Both transformations allow to calculate the orthogonal dominant directions, the filling edges direction and the wall's width. The inaccuracy of hand drawn design causes that graph edges with the same a priori attribute values will be slightly different after vectorization. This fact provokes scattered points in Hough space and an added difficulty in peak detection. For this reason we have used GBHT$_{\theta m}$ to extract the properties of the walls since the information required is concentrated only in three peaks. Then, the size of the input graph is reduced and the subgraph isomorphism against the models is done with the remainder of input graph. You can see below more details about the algorithm of document interpretation and reconstruction.

3.1. Parameter computing

A clustering process is applied over θ-m space, given by GBHT$_{\theta m}$, with the aim to detect the three peaks mentioned above. Two of these three peaks should have a difference of $\pi/2$ and correspond to orthogonal dominant directions. The third peak will be used to calculate the filling direction and the wall's width. The well-known *k-means* algorithm is used to classify the points of θ-m space in three classes C_V, C_H and C_F. The centres of these three classes will be taken as the three peaks shown in an ideal case in Fig. 3. Fig. 4a shows a 3D view of the θ-m space after GBHT$_{\theta m}$ of Fig. 1b. Notice that, since the θ axis is cyclic, the distance used in the clustering process must be defined in terms of a cyclic distance. Fig. 4b shows the three classes found with their corresponding dominant directions θ_V, θ_H and θ_F and the average length m_F of filling edges. Let be (θ_1, m_1), (θ_2, m_2)



(a) (b)

Figure 4. (a) 3D view of the θ-m space. (b) Clustering using k-means algorithm.

and (θ_3, m_3) the three peaks detected in θ-m space. We can calculate the following information from these peaks:

- **Dominant directions.** θ_1, θ_2 and θ_3 are the angles corresponding to the three dominant directions of the input edges. The first step is to find, out of θ_1, θ_2 and θ_3, which two directions are orthogonal. Let denote θ_{o1} and θ_{o2} the angles corresponding to these two directions. The third direction will be the filling direction and its angle will be denoted as θ_F.

- **Graph rotation value.** Let be V_α and H_α the minimal rotation angle which must be applied to a direction with angle α to align it respectively with vertical and horizontal directions. The rotation θ_{rot} that must be applied to the input graph to align its dominant directions with vertical and horizontal directions can be calculated as follows:

$$\theta_{rot} = min(\frac{1}{2}(V_{\theta_{o1}} + H_{\theta_{o2}}), \frac{1}{2}(H_{\theta_{o1}} + V_{\theta_{o2}})) \quad (1)$$

  If we assume $|\theta_{o1} - \theta_{o2}| = \pi/2 \pm \delta$, then the following equalities must also be satisfied: $|V_{\theta_{o1}} - H_{\theta_{o2}}| = \delta$ and $|H_{\theta_{o1}} - V_{\theta_{o2}}| = \delta$. After finding θ_{rot} we can know which direction θ_{o1} or θ_{o2} corresponds to vertical direction and which one corresponds to horizontal direction. Let be θ_V and θ_H respectively the dominant directions closest to vertical and horizontal directions.

- **Directions variation.** Let be Δ_H, Δ_V and Δ_F the range of variation allowed for the dominant directions.

Definition 3.2 Given an input graph $G_I(V_I, E_I)$, we define the *set of vertical edges* $E_V \subseteq E_I$ as $E_V = \{e \in E_I, \theta_e = \theta_V \pm \Delta_V\}$. Where θ_e is the orientation of input edge e. Similarly we can define E_H as the *set of horizontal edges* and E_F as the *set of filling edges*.

Given a dominant direction θ_i and its class C_i, Δ_V, Δ_H and Δ_F are calculated as follows:

$$\Delta_i = \frac{1}{card(C_i)} \sum_{p \in C_i} |\theta_p - \theta_i| \qquad (2)$$

- **Walls width.** Let be (θ_F, m_F) the class centre corresponding to filling edges. Given an ideal filling edge with length m_F and orientation θ_F (see Fig. 5), the width of the wall containing this edge will be a or b depending on whether it respectively joins vertical or horizontal edges. These values can be estimated according to the following equations:

$$a = m_F \cos(\theta_H - \theta_F) = m_F \sin(\theta_F - \theta_V) \qquad (3)$$

$$b = m_F \sin(\theta_H - \theta_F) = m_F \cos(\theta_F - \theta_V) \qquad (4)$$

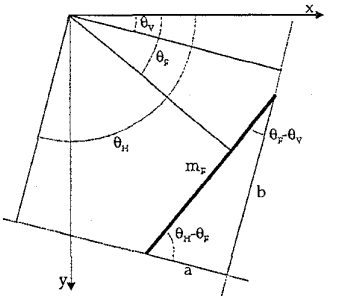


Figure 5. Average walls width calculation.

A filling edge, after GBHT$_{\theta m}$, is transformed into a point (θ_F, m_F) in the parameter space. In the estimation of average walls width we can not know which value a or b must be taken because both types of filling edges, vertical and horizontal, contribute in the accumulation point (θ_F, m_F). We will estimate an interval $[W_{min}, W_{max}]$ for walls width depending on a and b values and the standard deviation of filling edges length. The interval is calculated as follows:

$$W_{max} = (m_F + \Delta_F)\, max(\cos\beta, \sin\beta) \qquad (5)$$

$$W_{min} = (m_F - \Delta_F)\, min(\cos\beta, \sin\beta) \qquad (6)$$

where $\beta = \theta_H - \theta_F$ and Δ_F is calculated within the filling edges class as follows:

$$\Delta_F = \frac{1}{card(C_F)} \sum_{p \in C_F} |m_p - m_F| \qquad (7)$$

3.2. Connected component extraction

After walls' parameter estimation, we filter the input graph to obtain E_V, E_H and E_F. These edge sets are used to define three subgraph of input graph $G_I(V_I, E_I)$:

Definition 3.3 We define $G_V(V_V, E_V)$ as the *subgraph of vertical edges of* G_I. E_V was defined in definition 3.2 and V_V is defined as $V_V = \{v \in V_I, \exists e = (v_1, v_2) \in E_V, v = v_1 \text{ or } v = v_2\}$. In the same manner we define the subgraphs $G_H(V_H, E_H)$ and $G_F(V_F, E_F)$ using horizontal edges and filling edges respectively.

G_V and G_H do not contain enough information to extract vertical and horizontal walls because other picture elements contain vertical and horizontal edges too. It is necessary another step which searchs for pairs of vertical or horizontal edges joined by a filling edge. According to this idea we define vertical and horizontal connected components as follows:

Definition 3.4 A *vertical chain* $L_{VC}(V_{VC}, E_{VC})$ is defined as a connected subgraph of G_V such that its vertex set V_{VC} can be ordered in a sequence $[v_{i_1}, v_{i_2}, \ldots, v_{i_I}]$ such that for all $j = 2 \ldots I$, $(v_{i_{j-1}}, v_{i_j}) \in E_{VC}$.

Definition 3.5 Given $G_V(V_V, E_V)$, a *vertical connected component* is defined as a 3-tuple $C_V =< L_1, L_F, L_2 >$ where L_1 and L_2 are vertical chains and L_F is a subgraph of G_F whose edges join L_1 and L_2. An *horizontal connected component* is equivalently defined starting from G_H.

According to definition 3.5, we will denote C_V (C_H) as the set of vertical (horizontal) connected components extracted from G_V (G_H). Two vertical (horizontal) components sharing one of their vertical chains are merged in one vertical (horizontal) component. This merging process, shown if Fig. 6, is repeated until stability.

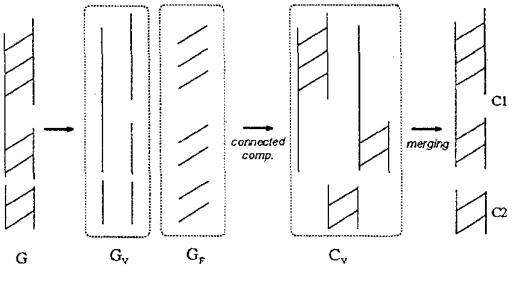


Figure 6. Vertical connected components extraction and merging.

Figs. 7a and 7b show vertical and horizontal connected componets and their bounding boxes. These bounding boxes are a first approximation of walls. For each box we store the following features which characterize its corresponding wall: width, filling edges orientation and frequence. Some of these attributes had already been calculated, but now we adjust better them using only the edges inside the wall.

4. Document redrawing

After wall recognition, a matching between the remainder of input graph and the set of model graphs is carried out. After finding the models in the input graph, we store only their position, scaling factor and orientation. These parameters are calculated by means of an alignment between the matched input vertices and the model vertices. The document can be reconstructed with wall information obtained above and instantiating each model according to its attributes (Fig. 7c shows this result). Notice that some elements appear overlapped. The last step is a correction of element position to avoid overlapping (see Fig. 7d).

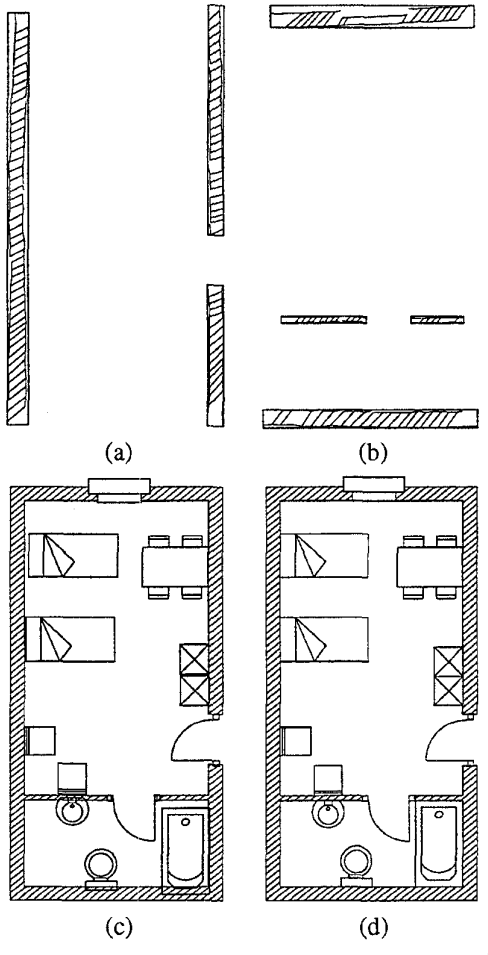


Figure 7. (a)(b) Vertical and horizontal connected components and their respective bounding boxes. (c) Document reconstruction. (d) Result after overlapping correction.

5. Conclusion

Hand drawn document understanding has been discussed and, particularly, the problem of hand drawn floor plan documents analysis. An attributed graph has been proposed to represent the structural information of the document after its vectorization. Recognition has been stated in terms of an inexact subgraph isomorphism between a graph representing the input document and a set of graphs representing some models to recognize. To speed up the matching and recognize some elements that doesn't have a fixed pattern, a Hough based process has been introduced. This process carries out a recognition of walls, characterized by a filling texture based on parallel straight lines even spaced. This step allows to reduce the size of the input graph and also to reduce the computational load for the subgraph isomorphism.

References

[1] H. Bunke and A. Sanfeliu. *Syntactic and Structural Pattern Recognition. Theory and Applications*. World Scientific Publishing Company, 1990.

[2] A. Habacha. Structural recognition of disturbed symbols using discrete relaxation. In *1st. ICDAR*, pages 170–178, September-October 1991. Saint Malo, France.

[3] T. C. Henderson. *Discrete Relaxation Techniques*. Oxford University Press, 1990.

[4] J. Illingworth and J. Kitler. A survey of the hough transform. *CVGIP*, (44):87–116, 1988.

[5] P. Kuner and B. Ueberreiter. Pattern recognition by graph matching. combinatorial versus continuous optimization. *IJPRAI*, 2(3):527–542, September 1988.

[6] V. Leavers. Which hough transform? *CVGIP*, 58(2):250–264, September 1993.

[7] S. Lee, J. Kim, and F. Groen. Translation-, rotation-, and scale-invariant recognition of hand-drawn symbols in schematic diagrams. *IJPRAI*, 4(1):1–25, 1990.

[8] J. Lladós and E. Martí. Structural recognition of hand drawn floor plans. In *VI Spanish Simposium on Pattern Recognition and Image Analysis*, pages 27–34, April 1995. Cordoba.

[9] J. Lladós, J. Regincós, and E. Martí. Interpretación de diseños a mano alzada como técnica de entrada a un sistema cad en un ámbito de arquitectura. In *III Congreso Español de Informática Gráfica*, pages 33–46, June 1993. Granada.

[10] J. López Krahe and P. Pousset. The detection of parallel straight lines with the application of the hough transform. In *9th. ICPR*, pages 939–941, 1988. Rome, Italy.

[11] R. Mohr and T. Henderson. Arc and path consistency revisited. *Artificial Intelligence*, (28):225–233, 1986.

[12] D. Pao, H. Li, and R. Jayakumar. Shapes recognition using the straight line hough transform: Theory and generalization. *IEEE Trans. on PAMI*, 14(11):1076–1089, November 1992.

[13] F. Wahl. *Deriving Features from Hough Space for Object Recognition and Configuration Estimation. In Simon J.C. (ed) From Pixels to Features*. Elsevier Science Publishers B.V. (North-Holland), 1989.