An Algorithm to Recognize Graphical Textured Symbols using String Representations *

Gemma Sánchez†§, Josep Lladós†, Karl Tombre§
†Computer Vision Center, Dept. Informàtica.
Universitat Autònoma de Barcelona, 08193 Bellaterra (Barcelona), Spain
§Loria, Campus scientifique, B.P. 239, 54506 Vandœuvre-lès-Nancy CEDEX, France

Abstract

This paper describes an algorithm to segment and recognize textured symbols in architectural drawings. These symbols are formed by structural textures, i.e. they are characterized by a repetitive pattern and a regular placement rule. We propose a string based representation of both, texels and rules. Thus, texture recognition is formulated in terms of two-step clustering approach: shape and rule clustering performed on a graph structure. In both cases the string edit distance is the basis to define the similarity criterion used to group features.

Keywords: graphics recognition, symbol recognition, texture segmentation, structured textures, clustering.

1 Introduction

Graphic symbol recognition is an important issue in graphics recognition. In the Collins Cobuild English Language Dictionary the word *symbol* is defined as "a shape or design that is used to represent something such as an idea." Although in graphics recognition this concept should be more restricted taking only into account the graphical images with a particular meaning, the kind of symbols that are often recognized belongs to a subset of this wide definition. Usually these symbols are fixed structures that represent a concept, as a window or a door in architectural plans, a dimension in engineering drawings, etc. We call them *classical symbols*. But what about the *textured symbols*? There are some graphics with a particular meaning which are not defined by a fixed structure but by a textured pattern filling them. For example, in architectural drawings there are symbols as the roof or the stairs which are defined by a texture. In the first case a set of tiles arranged following

Work partially supported by projects 2FD97-1800 and TIC2000-0382 of Spanish CICYT

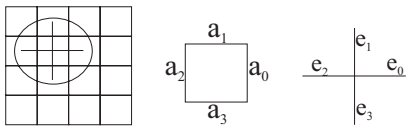

Figure 1: (a)Texture. (b)Polygon($P = a_0a_1a_2a_3$). (c)Placement rule($R = e_0e_1e_2e_3$)

a certain structure defines the roof, but each roof can have a different number of tiles. In the second, a set of rectangles arranged in rows and columns defines the stairs, but they can have different number of steps. While classical symbols are characterized by prototype patterns and the recognition follows a pattern matching approach, textured symbols are characterized by a repetition and recognized by a texture segmentation. So the algorithms to recognize the classical symbols are not useful for these textured symbols.

This paper presents an algorithm to recognize textured symbols within the domain of architectural drawings. In this domain classical and textured symbols may appear, being the former recognized using a graph matching algorithm presented in [1], while the latter are characterized by structured textures. Structured texture detection involves the detection of two repetitive features: the pattern, or texel, and the structure. The detection of a repetitive pattern usually conveys a clustering procedure in a certain feature space, see [2], while the detection of a regular structure has been solved in the literature using, projection information or a distance matching function, both with a Coocurrence Matrix (CM), see [3] and [4] respectively, a Binary CM, see [5], or a clustering, see [2]. We propose a common formulation for both, pattern and structure repetition detection, in terms of clustering string-based representations. In architectural drawings structured textured symbols can be characterized by repetitions of points, lines, or polygons, but we consider polygons, being the other possibilities easily adapted to the algorithm by simplifying the way to compare them. Thus given a set of polygons in the drawing we group them in terms of their shapes and placement rules, obtaining the textured symbols in the clusters with a certain number of shapes. Each cluster is represented by the mean shape and its mean placement rule computed from the set of shapes. The recognition process determines the correspondence between a given textured symbol and the segmented ones by comparing their texels and placement rules.

This paper is organized as follows. §2 presents the textured symbol recognition algorithm, §3 the results, ending with §4 where the conclusions are presented.

2 Textured symbol recognition

In our work, architectural drawings are first vectorized by the algorithm described in [6]. Then a RAG (*Region Adjacency Graph*) $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{L}_\mathcal{V}, \mathcal{L}'_\mathcal{V}, \mathcal{L}_\mathcal{E})$ describing the regions of the diagram and their adjacency relations is constructed as in [1], denoted as $\mathcal{G}(\mathcal{V}, \mathcal{E})$. Thus, $\mathcal{V}$ is the set of nodes corresponding to regions and $\mathcal{E}$ is the set of edges representing the region adjacencies. We denote as $(v, v') \in \mathcal{E}$ the edge connecting two regions v, $v' \in \mathcal{V}$. Every graph vertex, $v \in \mathcal{V}$, is represented by two attributed cyclic strings. One, given by the labeling function $\mathcal{L}_\mathcal{V} : \mathcal{V} \rightarrow E^*$, represents the sequence of lines forming the shape contour, out of the set E of segments in the vectorized drawing. The other, given by $\mathcal{L}'_\mathcal{V} : \mathcal{V} \rightarrow E'^*$, is a neighbourhood string, i.e. it represents the sequence of edges joining the centre of v and the respective centers of its neighbouring shapes, from the set E' of segments joining the shape centers. Both string-based attributes are illustrated in Fig.1. Finally, every graph edge (v, v') is attributed by the straight segment connecting the centers of the shapes v and v'. This is given by the labeling function $\mathcal{L}_\mathcal{E} : \mathcal{E} \rightarrow E'$.

The recognition process compares the representation of a given textured symbol with each representation of the symbols segmented from a RAG $\mathcal{G}(\mathcal{V}, \mathcal{E})$. This symbol segmentation is computed by a double hierarchical graph clustering on $\mathcal{G}(\mathcal{V}, \mathcal{E})$. First we cluster similar shapes and from each resultant cluster we group similar placement rules.

Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be a RAG. Let $\mathcal{P}$ be a partitioning of $\mathcal{V}$. A *clustering* on $\mathcal{G}$ is a partitioning $\mathcal{P}$ in which every cluster is a connected graph. This partition defines a set of subgraphs $\mathcal{C}_1(\mathcal{V}_1, \mathcal{E}_1), \ldots, \mathcal{C}_m(\mathcal{V}_m, \mathcal{E}_m)$. Let us call *clusters* such subgraphs. We call the *bridge* between $\mathcal{C}_i$ and $\mathcal{C}_j$ the set of edges connecting one vertex in $\mathcal{C}_i$ with one vertex in $\mathcal{C}_j$: $(\mathcal{C}_i, \mathcal{C}_j) = \{(v, v') \in \mathcal{E} | v \in \mathcal{V}_i \wedge v' \in \mathcal{V}_j\}$ for $i, j = 1, \ldots, m, i \neq j$. The image of $\mathcal{G}$ through $\mathcal{P}$ is the graph $\mathcal{G}_\mathcal{P}(\mathcal{V}_\mathcal{P}, \mathcal{E}_\mathcal{P})$, whose vertexs and edges are respectively the clusters and non-empty bridges induced by $\mathcal{P}$, $\mathcal{V}_\mathcal{P} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$, $\mathcal{E}_\mathcal{P} = \{(\mathcal{C}_i, \mathcal{C}_j) | (\mathcal{C}_i, \mathcal{C}_j) \neq \emptyset, i, j = 1, \ldots, m, i \neq j\}$. A *hierarchical clustering of height* n is a sequence of n clusterings, $\mathcal{G}^0(\mathcal{V}^0, \mathcal{E}^0), \ldots, \mathcal{G}^n(\mathcal{V}^n, \mathcal{E}^n)$, each applied to the image graph generated by the preceding clustering, which produces a hierarchy of $n + 1$ graphs including the original one. Where each $\mathcal{G}^k$ is the image of $\mathcal{G}^{k-1}$ through the partition $\mathcal{P}^k$, $\forall k = 1, \ldots, n$, and $\mathcal{G}^0$ is the original graph.

Textured symbol segmentation is solved by a graph clustering procedure grouping shapes or placement rules by redefining the similarity measure between vertexs in the graph. The common clustering engine starts giving a weight, $\frac{1}{n} \sum_{i=1}^{n} (dist(v, v_i))$, to each $v \in \mathcal{V}^0$, where $(v, v_i) \in \mathcal{E}^0$, and $dist(v, v_i)$ is a distance between vertexs. The vertexs with higher weight among their neighbours are selected as survivors and the rest, v_j, are associated to the nearest neighbour survivor, v_s, if $d(v_j, v_s)$ is less than

a given threshold, otherwise they are converted to survivors. Each survivor with its associated neighbours is one cluster $\mathcal{C}_i^j(\mathcal{V}_i^j, \mathcal{E}_i^j)$ in the current graph $\mathcal{G}^j$ and one vertex $v_i^{j+1} \in \mathcal{V}^{j+1}$ in the next graph $\mathcal{G}^{j+1}$. The attributes of each v_i^{j+1} are computed as a mean of the attributes of $v_i^j \in \mathcal{V}_i^j$. The process stops when $\mathcal{G}^n = \mathcal{G}^{n+1}$.

2.1 Shape Clustering

Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be the input RAG. A graph region $v \in \mathcal{V}$ is represented by the attributed cyclic string $\mathcal{L}_\mathcal{V}(v)$. Then the distance function to compare two shapes described by the strings $\mathcal{L}_\mathcal{V}(v)$ and $\mathcal{L}_\mathcal{V}(v')$, $dist(v, v') = w_1 d(\mathcal{L}_\mathcal{V}(v), \mathcal{L}_\mathcal{V}(v')) + w_2 d_a(\mathcal{L}_\mathcal{V}(v), \mathcal{L}_\mathcal{V}(v'))$, where d_a is the arithmetic difference between the areas of the strings, w_1 and w_2 are two weighting constants, and d is the string edit distance explained in [1]. Given two strings X and Y, $d(X, Y)$ is, informally speaking, the minimum cost to transform X into Y using certain edit operations. As in our case shapes can have their edges cut in several parts, we need the *merging edit operation* that allows to transform a whole sequence of symbols by another. The attribute of one vertex $v_i^{j+1} \in \mathcal{G}^{j+1}$, $v_i^{j+1} = \mathcal{C}_i^j(\mathcal{V}_i^j, \mathcal{E}_i^j)$, is a string representing all shapes in $\mathcal{V}_i^j$, and it is computed as the mean shape among the group of shapes $v' \in \mathcal{V}_i^j$, using the algorithm described in [7]. The clustering process explained above starts with $\mathcal{G}^0(\mathcal{V}^0, \mathcal{E}^0) = \mathcal{G}(\mathcal{V}, \mathcal{E})$. At the end clusters in $\mathcal{G}^n$ define sets of regions in $\mathcal{G}^0$ with a similar shape and area.

2.2 Neighbourhood Clustering

After the shape clustering, each vertex in $\mathcal{G}^n(\mathcal{V}^n, \mathcal{E}^n)$ is one cluster representing polygons with the same shape and area. Now the problem is to group shapes having also the same placement rule. For that purpose a second clustering is computed using the same algorithm. Initially we have one starting graph $\mathcal{G}_i^0(\mathcal{V}_i^0, \mathcal{E}_i^0)$ for each $v_i \in \mathcal{V}^n$. Each vertex $v \in \mathcal{V}_i^0$ is attributed with a string representing its neighbourhoods $\mathcal{L}'_\mathcal{V}(v)$, explained in §1, and for each graph $\mathcal{G}_i^0$, one string $S_i = \mathcal{L}_\mathcal{V}(v_i)$ represents its mean shape. The distance between two vertexs v and v' is computed as $dist(v, w) = d(\mathcal{L}'_\mathcal{V}(v), \mathcal{L}'_\mathcal{V}(v'))$, where d is the string edit distance defined in [1]. As the shape representing a placement rule is radial, with a set of lines starting in the same point, and not continuous as before, the merge edit operation explained in the previous subsection has no sense, and then the three edit operations: *substitution*, *insertion* and *deletion* of string elements are used. The attribute of one cluster $\mathcal{C}_i^j(\mathcal{V}_i^j, \mathcal{E}_i^j)$ is a string representing the placement rule, and it is also computed as the mean among the placement rules of all $v \in \mathcal{V}_i^j$, using the mean string algorithm described in [7].

At the end of the process we have several graphs $\mathcal{G}_i^{n_i}(\mathcal{V}_i^{n_i}, \mathcal{E}_i^{n_i})$, and each node $v \in \mathcal{V}_i^{n_i}$ grouping a certain number of polygons in $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is one textured symbol.

This textured symbol is represented by two strings: the mean texel S_i repeated in the texture, and the mean placement rule represented by the mean string ${\mathcal{L}'}_{\mathcal{V}_i^{n_i}}(v)$.

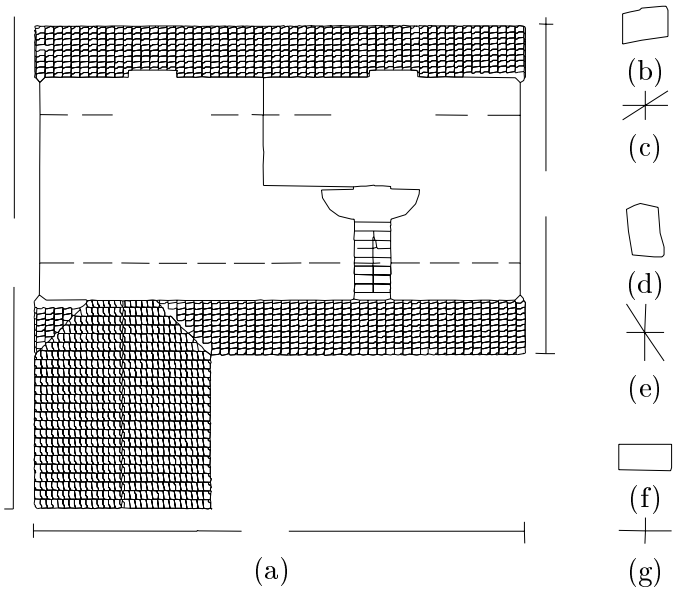

Figure 2: (a)Vectorized image (textured symbols in bold lines). (b)Texel of horizontal roof. (c)Placement rule of (b). (d)Texel of vertical roof. (e)Placement rule of (d). (f)Texel of the stairs. (g)Placement rule of (f).

3 Results

The algorithm has been tested in some architectural drawings, Fig. 2 illustrates one of them. Figure 2(a) presents the vectorization with the recognized textured symbols marked in bold lines. Two main textured symbols are observed, both roofs. In the top and the middle of the image they have their tiles in an horizontal position, while in the left bottom they are vertically placed. The texel and placement rule representing the roof symbols with horizontal tiles are in Fig. 2(b) and Fig. 2(c) respectively, while the representation, texel and placement rule, of the roof with vertical tiles are in Fig. 2(d) and Fig. 2(e) respectively. We can notice that in both cases they are the visually expected representations although some tiles are not segmented due to distortions. Another textured symbol is the stairs. The structured texture inferred is shown in Fig. 2(f) and Fig. 2(g). Since the symbol appears distorted due to gaps and additional lines the clustering process is unable to get a complete segmentation.

4 Conclusions

This paper has presented an algorithm to recognize textured symbols in architectural drawings by clustering similar shapes with similar placement rules. Both are represented by attributed cyclic strings and their similarity computed using the string edit distance. From each cluster the texel and placement rule are computed as the mean string among all the shapes and the mean string among all their placement rules. Since the algorithm tolerates distortion it fails in the presence of gaps and added lines. This suggest a relaxation process to find this distorted, cut or joined texels by trying to adapt the computed texel and placement rule on them. Once we have the representation of a segmented textured symbol we can recognize if it is a given one by comparing their representations. This process can be easily implemented by using a string edit distance algorithm invariant to rotation.

References

[1] J. Lladós, G. Sánchez and E. Martí. A string-based method to recognize symbols and structural textures in architectural plans. *Graphics Recognition, Algorithms and Systems. K. Tombre and A.K.Chhabra (eds). Lecture Notes in Computer Science, Springer-Verlag*, 1389:91–103, 1998.

[2] F. Tomita, Y. Sjirai and S. Tsuji. Description of Textures by a Structural Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(2):183–191, March 1982.

[3] H. Kim and R. Park. Extracting Spatial Arrangement of Structural Textures using Projection Information. *Pattern Recognition*, 25(3):237–245, 1992.

[4] G. Oh, S. Lee and S.Y. Shin. Fast determination of textural periodicity using distance matching function. *Pattern Recognition Letters*, 20:191–197, 1999.

[5] V.V. Starovoitov, S.Y. Jeong and R.H. Park. Texture Periodicity Detection: Features, Properties and Comparisons. *IEEE Transactions on Systems Man and Cybernetics, part A: Systems and Humans*,28(6):839–849, 1998.

[6] K. Tombre et al. Stable and Robust Vectorization: How to Make the Right Choices. *Graphics Recognition—Recent Advances. A. K. Chhabra and D. Dori (eds). Lecture Notes in Computer Science, Springer Verlag*, 1941:3–18, 2000.

[7] G. Sánchez, J. Lladós and K. Tombre. A mean string algorithm to compute the average among a set of 2D shapes. *CVC Technical Report 38*, February 2000.