Chapter 6
A Relational Indexing Method for Symbol Spotting

Abstract In this chapter, we present a method to retrieve from a collection of document images the regions of interest where a query symbol is likely to be found. In order to foster the querying speed, a hashing technique is proposed which is able to retrieve very efficiently primitives by similarity. Vectorial primitives are coarsely encoded by well-known shape description methods providing a numerical description of the primitives. A relational indexing approach is presented in order to introduce some structural information of the symbols and provide an accurate hypotheses validation. Experimental results show the performance of the proposed approach.

6.1 Introduction and Related Work

The use of a lookup table providing a prototype-based search of similar primitives, as presented in the last chapter, allows avoiding the computation of the similarity measure for all the primitives extracted from the collection. The use of such indexing structures aims at efficiently accessing and retrieving graphic elements by similarity, and becomes a must when dealing with applications which have to face large collections of documents. In the particular usage case presented in the last chapter, we achieved reducing the number of distance computations by almost a factor of 45 without missing a significant number of symbols. However, there is still need to compute several hundreds of distances between descriptors. Even if this is not an important burden when working with numeric descriptors, it may be an important inconvenience when we use symbolic description of primitives such as the attributed strings. In this chapter, we propose enhancing the accessibility to the stored descriptors by two means. First, we will coarsely describe primitives by the use of well-known descriptors with low dimensionality. These descriptors result in a numeric feature vector. The distance among those descriptors is easily computed as the distance between two points in the n-dimensional description space. Second, this description space is efficiently organized and accessed by the use of a hashing technique. The use of hashing techniques in ideal conditions allows retrieving items by similarity with a complexity $\mathcal{O}(1)$. We can find many works which use such efficient indexing structures to organize and retrieve the primitive descriptors in the

M. Rusiñol, J. Lladós, *Symbol Spotting in Digital Libraries*,
DOI 10.1007/978-1-84996-208-7_6,

literature. Califano and Mohan [2] used a hash table indexed by four-dimensional indices describing the geometric configuration of triplets of points extracted from a contour image in order to efficiently locate the location of query objects in an image. Stein and Medioni [19] also used a hash table in order to provide an efficient retrieval of similar portions of a contour described by a set of features extracted from a super-segment. Recently, Lladós and Sánchez [12] proposed a binary codification of the shape context descriptor which is stored in an indexing structure aiming at efficiently retrieving the locations within a document image where a given typewritten word is likely to appear.

Moreover, there is another drawback in the previously presented method. Since graphical symbols are composed of several primitives, querying a symbol used to involve separately querying each of its primitives. The locations showing a larger accumulation of primitives were taken as the most plausible places to contain the queried symbol. This technique may lead to several false alarms since we are not checking which primitives appear in those zones and whether their spatial organization and their structural configuration is consistent with the query symbol design. In this chapter, we propose an indexing methodology to add structural information in the primitive queries. In the literature, we can find several works such as by Chang and Lee [3] or by Costa and Shapiro [5] which are focused on the addition of structural information to the primitive querying process. We can call such approaches relational indexing since, besides indexing primitive objects, these works try to index also their spatial relationships. An enhanced voting scheme aiming at a better validation of the spotted locations is also presented in this chapter.

The remainder of this chapter is structured as follows. We start by detailing how the symbols are represented in terms of a polygonal approximation of contours and a relational graph. Subsequently, in Sect. 6.3, we present the off-the-shelf shape descriptors we have used in our experiments to coarsely describe and index the primitives by similarity. Even if some of the descriptors were conceived to describe images, they are reformulated to be applied to a set of polygonal primitives. Section 6.4 presents the indexing structure to efficiently retrieve primitives, and Sect. 6.5 outlines how the relational indexing methodology works. In Sect. 6.6, we present some qualitative results of using the proposed spotting architecture to retrieve locations of interest from a collection of line-drawing images. Finally, the conclusions and a short discussion can be found in Sect. 6.7.

6.2 Description of Graphical Symbols in Terms of Vectorial Primitives

Recognition schemes rely on two basic steps, namely primitive extraction and description. First, the primitive extraction step has to transform the image drawings arising from the scanning process to a vector domain. Then, in the second step, such primitives have to be represented by a shape descriptor.

6.2.1 Vectorial Primitives

Graphical symbols usually comprise a union of several simple sub-shapes. Therefore, a symbol can be described in terms of the assembly of sub-shapes which comprises it. The basic primitives we want to extract to represent a graphical symbol are these simple sub-shapes.

As our work is focused on the management of graphical data in vectorial format, the documents which are in paper format need a digitalization process. In this chapter, we use the same raster-to-vector process as in the previous chapter with just one particularity. Since we want to add relational information between primitives to the indexing framework, a graph representation of the symbols is also needed. The documents are scanned and de-noised by some simple morphological operations. The raster-to-vector algorithm proposed in [14] is then applied to these line-drawing images to obtain a vectorial representation of the documents. However, such vectors are not suitable to be used as primitives due to their instability in terms of artifacts, fragmentation, errors in junctions, etc. A higher level entity has to be used as a primitive. Adjacent vectors are merged together into a polyline instance. These polylines represent then the sub-shapes forming a given graphical symbol. In our method, we use the contour of the closed loops corresponding to a symbol as the primitives to polygonally approximate and to merge as single polylines.

Formally, let $p = \{s_1 \dots s_n\}$ be a polyline consisting of n segments s_i. A symbol is represented in terms of its polylines representing loops and denoted as $S = \{p_1 \dots p_m\}$. The gravity center of the symbol is computed as the average of the gravity centers of each polyline, and it is denoted by m_C. The gravity center of the symbol will be used in the subsequent process of localization of the query symbol inside the line-drawing images. To represent the spatial organization of primitives which comprise a symbol, a proximity graph is constructed. Using the k-NN algorithm, each primitive is linked to its k nearest primitives by an edge of the graph $G(S) = (V, E)$. A node $n_i \in V$ is attributed with the primitive p_i. An edge $e \in E$ is denoted as $e = (n_i, n_j, \overrightarrow{v_{ij}})$, where n_i and n_j are nodes of V and $\overrightarrow{v_{ij}}$ is a vector representing the spatial relationship between the primitives p_i and p_j. This proximity graph is the basis of the proposed relational indexing technique.

In Fig. 6.1, we can observe how the different parts of a symbol are detached, making the regions meaningful primitives, and how their spatial organization can describe a symbol.

Note that the same primitive representation and extraction is used for the complete documents in the acquisition step. A given document D is composed of a large number of polylines. A proximity graph $G(D)$ is also computed to link nearby primitives and to store their spatial relationship. Obviously, in this case we do not know which polylines comprise a symbol; the graph just represents neighboring primitives.

The polygonally approximated sub-shapes are used as the local components of a given symbol. To describe them, at each primitive separately we apply one of the off-the-shelf global numerical shape descriptors existing in the literature.

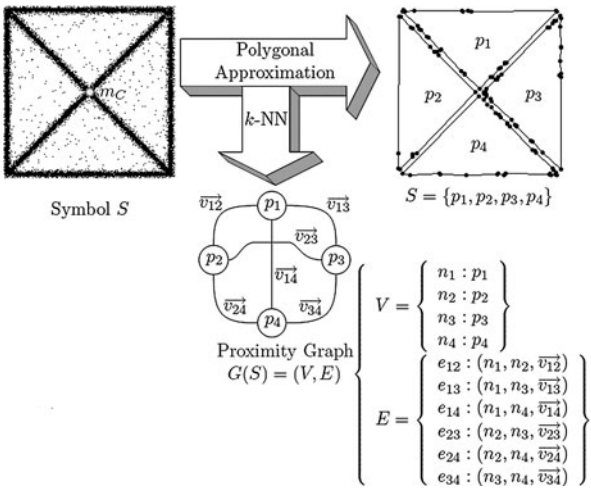

Fig. 6.1 Primitive symbol decomposition. A graphical symbol is decomposed into sub-shapes which are polygonally approximated. An attributed proximity graph is the basis for the relational indexing

6.3 Off-the-Shelf Shape Descriptors Applied to Vectorial Data

Formally speaking, given a symbol $S = \{p_1 \dots p_m\}$ and a shape descriptor f defined over the space of primitives, after applying f to each primitive we will have in return a set of feature vectors $f(p_i)$ for all $i \in [1, m]$. A symbol is then expressed by a set of feature vectors describing its primitives. Let us briefly review the used shape descriptors in the next section.

Global numerical shape descriptors are formulated in terms of a compact representation of expressive invariant features describing a shape as a whole. The interested reader is referred to Zhang and Lu's [22] review of shape representation and description techniques. In this section, we will summarize the global shape descriptors used in our experiments. We make no claims about robustness of the chosen descriptors. Depending on the nature of the data, better descriptors can be used. The point here is only to test several shape descriptors seen as black-boxes which one can plug-in into the system. The selection of one or another shape descriptor is application dependent. For example, if we are interested in retrieving just the correct symbols despite missing some positives, an accurate shape descriptor has to be chosen. On the other hand, if the user wants to retrieve all the instances of a given symbol without giving real importance to the presence of false positives, one must choose a simpler shape descriptor. Four shape descriptors with different accuracy are chosen here to test the behavior of the system.

Let us further overview the numerical shape descriptors used in our work. First, we introduce some basic notation. We consider an image $I(x, y)$ containing an object shape O with area A and perimeter P. Its centroid is the point $c = (\bar{x}, \bar{y})$. The boundary B of the shape is polygonally approximated by a polyline p_O composed by a set of n adjacent segments $s_i = \{(x_i, y_i), (x_{i+1}, y_{i+1})\}$. A shape descriptor will result in a compact representation of the shape formulated in terms of a feature vector $f(O)$. Let us briefly introduce the well-known shape descriptors we use.

6.3.1 Geometric Moments

The central $(p+q)$th order moment for a digital image $I(x, y)$ is expressed by

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q I(x, y). \tag{6.1}$$

The use of the centroid $c = (\bar{x}, \bar{y})$ allows for the invariance to translation. A normalization by the object area is used to achieve invariance to scale.

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \quad \text{where } \gamma = \frac{p+q}{2} + 1. \tag{6.2}$$

6.3.1.1 Boundary Moments

The geometric moments can also be computed for the contour of the object as described by Chen [4] and by Sardana et al. [16] by using (6.1) only for the pixels of the boundary of the object. In that case, a normalization by the object perimeter is used to achieve invariance to scale by using (6.2) with $\gamma = p + q + 1$. By sampling the polygonal approximation, we can use the boundary moments as geometric descriptors of the primitives.

6.3.1.2 Geometric Moments for Line Segments

When the contours of the objects are polygonally approximated, the geometric moments can be formulated for line segments as introduced by Lambert and Gao in [10, 11]. Given a polygonally approximated shape composed of n segments, let us take $a_i = (y_{i+1} - y_i)/(x_{i+1} - x_i)$ as the slope of the segment s_i. The line moments are then computed by

$$\mu_{pq} = \sum_{i=1}^{n} D_i,$$

$$D_i = \sqrt{1 + (a_i)^2} \cdot \sum_{k=0}^{q} \left\{ \binom{q}{k} a_i^k (y_i - a_i x_i)^{q-k} \cdot \frac{x_{i+1}^{p+k+1} - x_i^{p+k+1}}{p+k+1} \right\}. \tag{6.3}$$

And if the segment s_i is vertical, we use

$$D_i = x_i^p \cdot \frac{y_{i+1}^{q+1} - y_i^{q+1}}{q+1}. \tag{6.4}$$

6.3.1.3 Hu's Moment's Invariants

To obtain invariance with respect to translation, the centroid is used as in (6.1). The normalization by the polyline length is used to obtain scaling invariance. Finally,

invariance to rotation is achieved by using the set of seven functions proposed in [7] involving moments up to the third order.

$$
\begin{aligned}
\phi_1 &= \eta_{20} + \eta_{02},\\
\phi_2 &= (\eta_{20} - \eta_{02})^2 + (2\eta_{11})^2,\\
\phi_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2,\\
\phi_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2,\\
\phi_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})\big[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\big]\\
&\quad + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})\big[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\big],\\
\phi_6 &= (\eta_{20} - \eta_{02})\big[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\big] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}),\\
\phi_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})\big[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\big]\\
&\quad - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})\big[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\big].
\end{aligned}
\tag{6.5}
$$

Moment invariants can be normalized to get the different invariants fall into similar numerical ranges. Usually, we can use the logarithm as a coarse normalization:

$$\psi_1 = \log|\phi_i|, \quad i \in \{0, \ldots, 7\}. \tag{6.6}$$

Hupkens and de Clippeleir [8] proposed the following normalization of invariants to achieve a better robustness to noise:

$$
\begin{aligned}
\phi'_1 &= \phi_1 = \eta_{20} + \eta_{02},\\
\phi'_2 &= \phi_2/\phi_1^2,\\
\phi'_3 &= \phi_3/\phi_1^3,\\
\phi'_4 &= \phi_4/\phi_1^3,\\
\phi'_5 &= \phi_5/\phi_1^6,\\
\phi'_6 &= \phi_6/\phi_1^4,\\
\phi'_7 &= \phi_7/\phi_1^6.
\end{aligned}
\tag{6.7}
$$

6.3.2 Simple Shape Description Ratios

The eccentricity, aspect-ratio or Feret's ratio, of a given shape is the ratio of the length of the longest chord of the shape to the longest chord perpendicular to it. It can be computed by using the moments described in (6.3) as

$$ecc = \frac{\mu_{20} + \mu_{02} + \sqrt{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}}{\mu_{20} + \mu_{02} - \sqrt{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}}. \tag{6.8}$$

The circularity, or area-perimeter ratio of a shape, is defined as how closely-packed the shape is. For a circle it is equal to 1, all other shapes have a circularity smaller than 1. It is computed as

$$circ = \frac{4\pi A}{P^2}. \tag{6.9}$$

Obviously, there are many other shape ratios describing certain geometrical properties. The interested reader is referred to [15, 20]. In our case, we only use these two ratios as the feature vector describing a shape.

6.3.3 Fourier Descriptors

Given a polyline p_O which is the polygonal approximation of the boundary of a shape O, as a vectorial shape signature we use the centrical distance function computed as

$$r_i = \sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2} \quad \text{for } (x_i, y_i) \in p_O. \tag{6.10}$$

Zahn [21] obtained a Fourier descriptor of a shape, applying the Fourier transform on the signature representing the shape boundary. Sampling r_i to $N = 2^n$ samples so that the use of the FFT is possible, the feature vector of the Fourier descriptor is given by

$$f(O) = \left[\frac{|F_1|}{|F_0|} \dots \frac{|F_{N/2}|}{|F_0|} \right], \tag{6.11}$$

where F_i corresponds to the ith component of the Fourier spectrum. Other shape signatures such as curvature or complex coordinates can be used to compute the Fourier descriptor. The interested reader is referred to [9].

In the case of graphical symbols, the shape descriptors presented above can be applied to each of the primitives of the symbol extracted as mentioned in Sect. 6.2.1. Formally speaking, given a symbol $S = \{p_1 \dots p_p\}$, applying one of the presented descriptors will return a set of feature vectors $f(p_i)$ for all $i \in [1, p]$. In the next section, we will study how to adapt classical indexing structures used in the databases field to index graphical symbols in a document database.

6.4 Multidimensional Hashing to Index Primitives

The previously described methods for spotting symbols from a document database present an important constraint. As the number of considered shape models is increased, the computational cost of the matching step can be unaffordable. As pointed in [2], in order to avoid a brute-force matching step, the use of indexing paradigms becomes necessary.

Among the wide taxonomy of indexing structures (cf. [6]), the *point access methods* are the ones which are the most suitable for our purposes. Tree-based structures are frequently used in indexing mechanisms. Nevertheless, they suffer from several drawbacks. The querying process can be computationally expensive since the tree has to be traversed, and in addition, tree balancing algorithms are needed to maintain an effective search performance. Since in our case we want to foster the querying speed and we want a system where the data could be easily added at any moment, a multidimensional hashing technique has been selected instead of a tree-based one. In particular, we use a grid file structure, described in [13], in order to index the vectorial primitives. Let us overview in more detail how multidimensional hashing methods work.

Multidimensional hashing methods partition the space into hypercubes of known size and group all the records contained in the same hypercube into a bucket. The buckets are uniquely identified by a *key-index* which aims at a fast retrieval of all the data contained in the bucket. A hash function performing one-dimensional partitions automatically computes the key-index of a given query to identify the bucket to which it belongs.

In our case, given a polyline, a feature vector is computed using one of the presented descriptors and then a hash function obtains the key-index. This hash function establish a quantization criterion to apply to each dimension of the feature vector to limit the key-index parameters to a finite number of discrete values. To avoid boundary effects, each primitive is stored at the two closest buckets in each dimension.

Usually, the main drawback of hashing techniques is the collisions. Given two different items to store in the database, we have to guarantee that the hash function used to index such items does not assign the same key-index to them. To overcome this problem expensive re-hashing algorithms are applied once a collision is detected. In our case, collisions are not a problem but the basis of our indexing strategy. Given two similar (but not equal) primitives, they are represented by a compact feature vector. Hopefully, if the two primitives have a similar shape, the two feature vectors will be two nearby points in the description n-dimensional space. The partition of this space by the grid file has to guarantee that both points fall into the same bucket (or at least to neighboring buckets) to have all the similar primitives stored in a single entry. This technique allows having an efficient retrieval by similarity.

In Fig. 6.2, we can see an overview of how the indexing mechanism works. Formally speaking, a symbol $S = \{p_1 \dots p_m\}$ is described by a set of feature vectors $f(p_i)$ for all $i \in [1, m]$ arising from one of the descriptors presented above in Sect. 6.3. A hash function $h_p(f(p_i)) = k_i$ returns a key-index identifying a certain bucket in the n-dimensional indexing space. As the shape descriptors are invariant to similarity transformations and robust to noise, even if the input primitives are not completely equal, the whole procedure leads to the same bucket. The symbol S is then represented by the set of key-indices $\{k_1 \dots k_k\}$ with $k \leq m$ since all the similar primitives are represented by the same key-index.

In each bucket, the information of the position in a three-dimensional space (i.e., (x, y) coordinates of the primitive gravity center appearing in a certain document d of the collection) of all the primitives in the document database having key-index

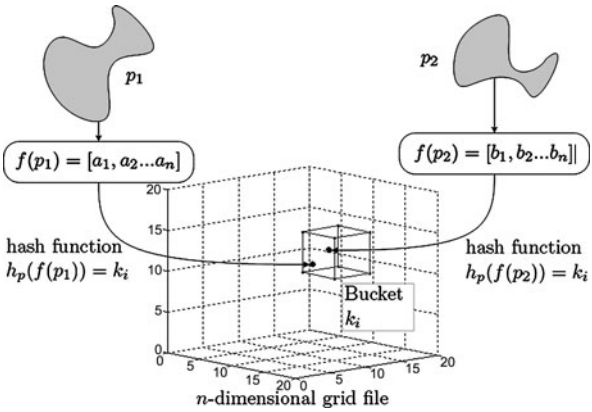

Fig. 6.2 The use of a grid file to index vectorial primitives. The hash function projects the feature vectors into key-indices. Two similar primitives are stored at the same bucket

k_i is stored. Summarizing this section, the proposed indexing methodology allows retrieving all the spatial locations where similar primitives as the queried one are likely to be found.

6.5 Relational Indexing and Hypotheses Validation

Since graphical symbols are composed of several primitives, indexing a symbol consists in separately indexing each of its primitives. This approach has a big drawback since the spatial coherence of the retrieved primitives is not taken into account. In this section, we present a relational indexing algorithm to furnish the indexation methodology with spatial information. A voting scheme to validate the spotted locations is also presented.

6.5.1 Relational Indexing

When considering large databases, many symbols may share a substantial part of primitives with each other. Bag-of-words models describe objects in terms of the presence of the primitives which compound them, ignoring their spatial structure. Recently, a method to locate objects in images using a bag-of-words model has been proposed in [17]. The large number of features taken from interest points aim to discard spatial information. However, in our case, the presence of a set of primitives in a given location does not guarantee the presence of the searched symbol since symbols are not usually composed of too many primitives. The geometrical configuration of these primitives is crucial information to refine the zones of interest. Inspired by the work presented in [5], spatial relationships among primitives are also considered when indexing in order to obtain much more valid hypotheses.

Given a symbol represented by a set of primitives $S = \{p_1 \dots p_m\}$, the similar primitives appearing in a document can be retrieved by using the set of key-indices

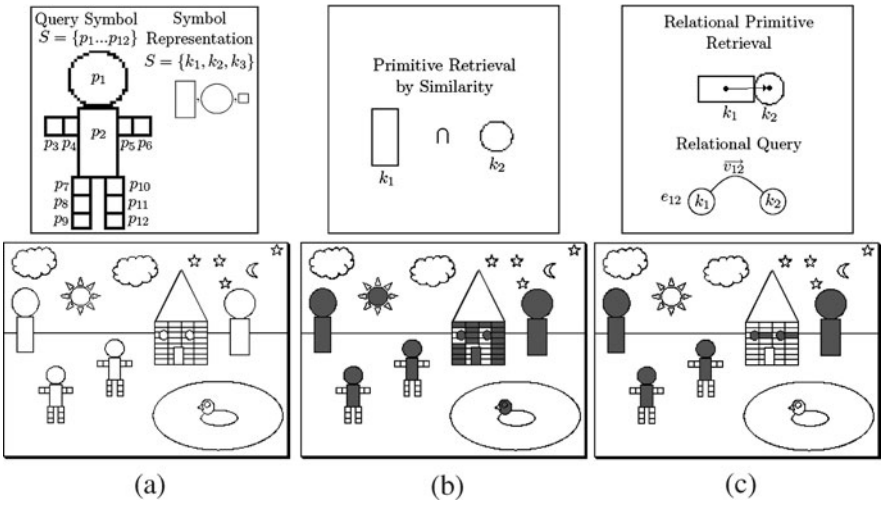

Fig. 6.3 Relational indexing. For the sake of visibility, only two primitives p_1 and p_2 are queried. (**a**) Sample line-drawing and the query symbol; (**b**) results of retrieving a couple of primitives by similarity without taking into account the spatial information, the resulting primitives are highlighted in *gray*; (**c**) retrieving the same two primitives by using the relational indexing mechanism

$\{k_1 \ldots k_k\}$. To take into account the spatial configuration of those primitives, the proximity graph $G(S)$ has to be used. The edges $e_{ij} \in E$ represent the relationship between two primitives stored in the nodes n_i and n_j. These edges can be used to retrieve by similarity pairs of primitives agreeing with a certain spatial distribution. We can find an example of the use of relational indexing in Fig. 6.3.

To efficiently retrieve all the edges of a query symbol, a hash table H_R is used to store the adjacency matrix of the proximity graphs in the memory. This hash table is indexed by pairs of primitives. The use of hash tables with multiple indices has been used over the years to store and guarantee an efficient access to sparse matrices like in [18]. The entry of the table $H_R[k_a, k_b]$ stores all the possible edges e_{ij} where the primitive stored at the node n_i is indexed by k_a and the primitive of the node n_j is indexed by k_b. In the acquisition step, for all the documents D in the collection, each graph $G(D)$ is added to the table H_R so a spatial relationship between two given primitives can be efficiently retrieved from all the document collection.

When querying a given symbol, each edge of the graph is considered. A querying function $Q(e_{ij}, m_C)$, taking an edge and the center of the query symbol m_C, results in a list of hypothetic centers $Lh_C = [h_{C1} \ldots h_{Cx}]$ where the two primitives with a given pose are to be found. We can see how this function proceeds in Fig. 6.4. The key-indices representing the primitives stored at the nodes are computed by using the hash function h_p. Both indices identify an entry of the hash table H_R storing a list of edges, and most importantly the corresponding vectors $\overrightarrow{v_{ij}}$. These vectors are the spatial distributions of the primitives appearing in the document database. A center mapping function $Cmap(\overrightarrow{v_i}, m_C) = h_{Ci}$ applies a scale and rotation transform to the center m_C in order to find the pose of the hypothetic center h_{Ci} depending on

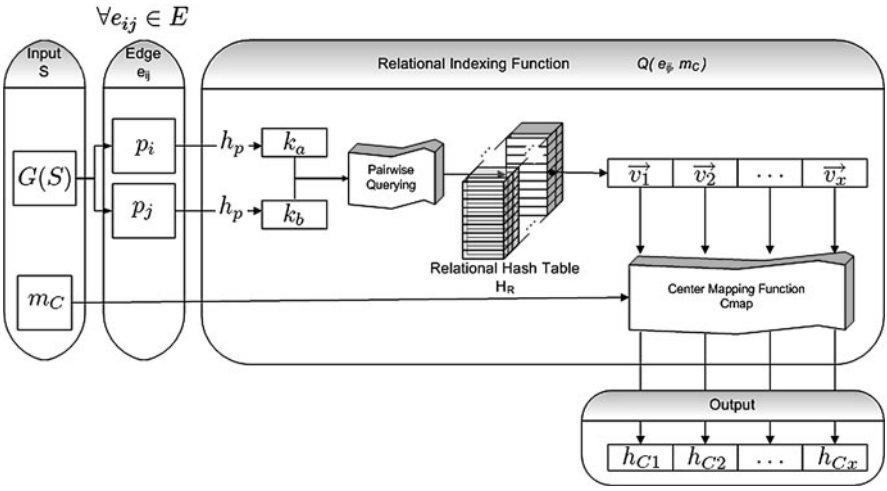

Fig. 6.4 Relational indexing architecture. Starting from the proximity graph, each edge performs a relational query based on the indices representing the primitives stored in the nodes. A list of vectors is retrieved corresponding to spatial relationships between primitives in target documents. A center mapping function transform these vectors into hypothetic centers where the symbol should be found

the vector $\overrightarrow{v_i}$. We can see an example of the hypothetic center location in Fig. 6.5. Note that the center mapping process aligns the query edge with the retrieved edges in the line-drawing database, thus being invariant to scale and rotation transforms.

By applying the relational indexing function to each edge of the proximity graph of the query, the locations in the documents where we can really find the queried symbol should appear several times in the hypothetic centers list. The use of a voting scheme reinforces these hypotheses and validates the possible locations.

6.5.2 Voting Scheme

Following the idea of the Generalized Hough Transform (GHT) [1], each of these centers accumulates votes. Applying the querying function to each edge of the graph from the query symbol, we accumulate evidences in the hypothetic centers in the stored documents where it is probable to find similar primitives with the same spatial organization as the query. In the voting space, the coherent votes tend to form salient peaks, the rest of the votes will be scattered in different locations but not forming clusters. A simple ranking of these clusters results in the positions of the documents where it is more feasible to find the queried symbol.

The querying process leads to considering each pair of primitives of the queried symbol $S = \{p_1 \dots p_m\}$, implying C_2^m accesses to the hash table H_R. The number x of hypothetic centers where to cast votes is the same as the number of how many position vectors are stored at each table entry. Obviously, the x value is directly

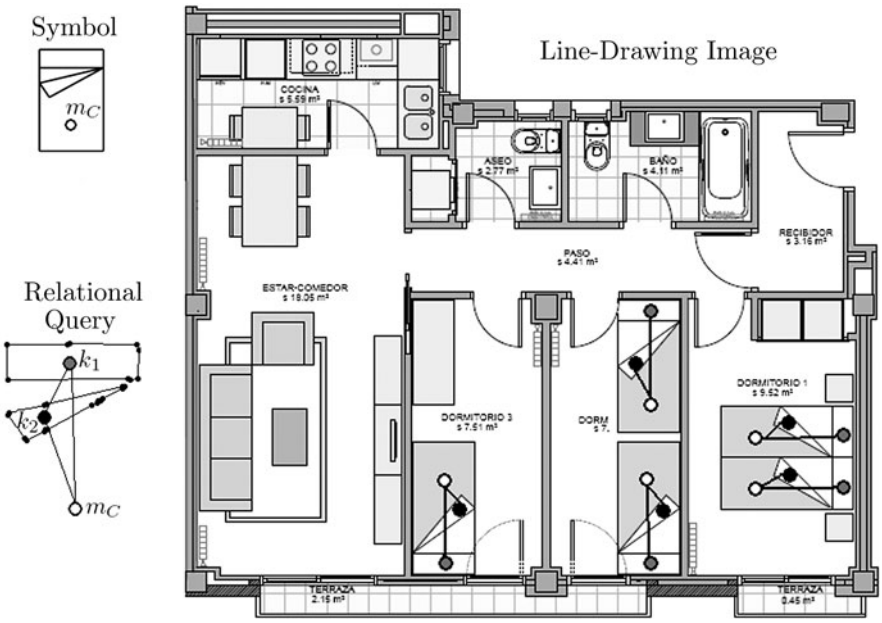

Fig. 6.5 Center mapping function to find the pose of the hypothetic centers given an edge of the relational query and the gravity center of the query symbol

related to the number of documents stored in the library. The result is that for each query symbol we have

$$x \cdot C_2^m = x \cdot \binom{m}{2} = x \cdot \frac{m!}{2(m-2)!} \tag{6.12}$$

centers where to accumulate votes. The locations where the votes are cast are sorted and returned as the retrieved regions of interest. Note that no threshold is used to decide whether a symbol is present or not. In the next section, we present some qualitative results of applying the presented relational indexing method.

6.6 Experimental Results

To obtain the experimental results, we worked with a collection of architectural floor-plans consisting of 42 images (of 3,215 × 2,064 pixels on average) arising from four different projects. This dataset is the FPLAN-POLY database,[1] detailed in Appendix A. These images are polygonally approximated, resulting in a collection of vectorial documents. The symbols taken into account for these experiments are divided into 38 classes, and we have a total of 344 instances in the document

[1]The FPLAN-POLY database is available at http://www.cvc.uab.cat/~marcal/FPLAN-POLY/.

images. In a single document image, the average number of symbols is around 8, and it ranges from 0 to 28 symbols. The models to query the document database are cropped from the document images, so they also contain vectorial distortions.

When querying a model symbol against the database, the convex hull of the activated polylines in the documents forms a set of regions of interest which are sorted by confidence value depending on the number of received votes. We can see the first 20 results of querying several symbols in the whole document collection when using the Fourier shape descriptor in Figs. 6.6 and 6.7. As we can observe, most of the results correspond to the correct queried symbol, but obviously some areas of false positives appear. We observe two interesting phenomena, usually, two close symbols (i.e., burners in Fig. 6.7f or chairs in Fig. 6.6d) are grouped into a single region of interest; on the other hand, it is common to find that a symbol is well spotted but the returned region of interest is bigger than expected (i.e., the burners in Fig. 6.7f).

We consider that if the resulting polygons are able to overlap at least a certain percentage of the ground-truthed representation of a symbol, they can be considered as recognized. On the other hand, if the resulting polygons do not cover the ground-truth, the symbol should be considered as missed. Of course, as with all decisions implying a certain threshold, its value can be critical, and the system's evaluation can depend on it. The definition of this threshold is completely subjective as it depends on what the user considers a symbol as being detected or not. In our case, we consider a symbol as detected if it overlaps at least 75% of the ground-truth area. In Table 6.1, we can see the total True Positive Rate (*TPR*) when applying the different shape descriptors and the average of False Positives (*FP*) regions obtained by all these methods. Notice that the time to retrieve a symbol from a document is highly related to the accuracy of the selected method. Methods having higher recognition rates spend more time in retrieving zones of interest since the table entries are more populated and the number of false positives is also increased. On the other hand, the methods which have smaller recognition rate but also fewer false positives are usually less computationally expensive.

However, in focused retrieval applications, there are some cases where performance evaluation is not straightforward. Let us consider the example shown in Fig. 6.8. Given a document in the collection, we query one symbol which can be found twice within the document. Instead of obtaining two different regions of interest framing the occurrences of this symbol, the system outputs a single region framing both instances of the symbol. The two symbols were relatively close in space in the document, so it is understandable that the system just retrieved one big region of interest where the probability to find the query object was high enough. However, the question of how to evaluate this result is not easy to answer. Both symbols were retrieved, but the system failed to identify that there were two different instances. By returning just one region, its area is big enough to contain other graphic objects which are not parts of the symbol, but it is hard to consider this result as a false alarm. In the last part of this book, we propose a protocol for performance evaluation for symbol spotting and focused retrieval systems. In this part, we will present the quantitative evaluation of the relational indexing method presented in this chapter.

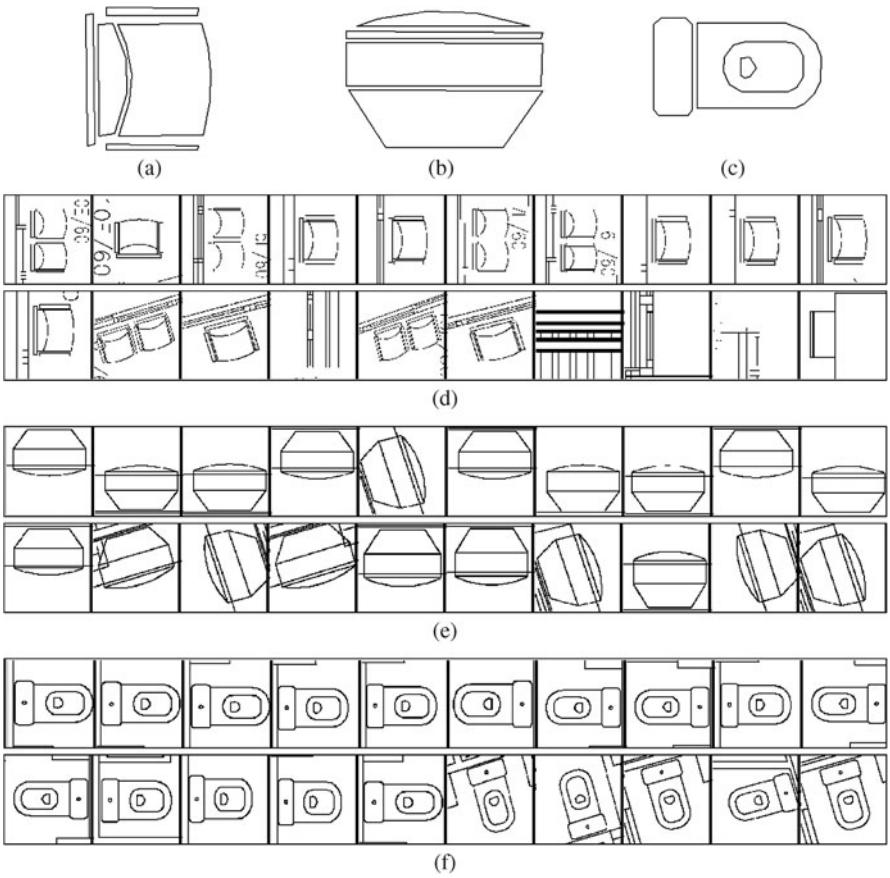

Fig. 6.6 Qualitative results of the relational indexing method (1). (**a**) Query symbol *chair*; (**b**) query symbol *TV set*; (**c**) query symbol *toilet*; (**d**), (**e**) and (**f**) first 20 retrieved regions when querying the symbols of (**a**), (**b**) and (**c**), respectively

6.7 Conclusions and Discussion

A relational indexing mechanism to spot symbols in a collection of line-drawing images in vectorial format has been presented. A first step of primitive extraction and description has been introduced in order to have a compact representation of the graphical symbols. These primitives are organized in an indexing structure to retrieve by similarity all the primitives in the collection. A relational indexing mechanism has been presented in order to take into account not only the similarity of the primitives which compound a symbol but also the spatial relationship among them. Finally, a Hough-like voting scheme aims at validating the hypotheses where a symbol is likely to be found.

The qualitative results show good performance results. Most of the approaches in the literature always make a choice of using only structural information about

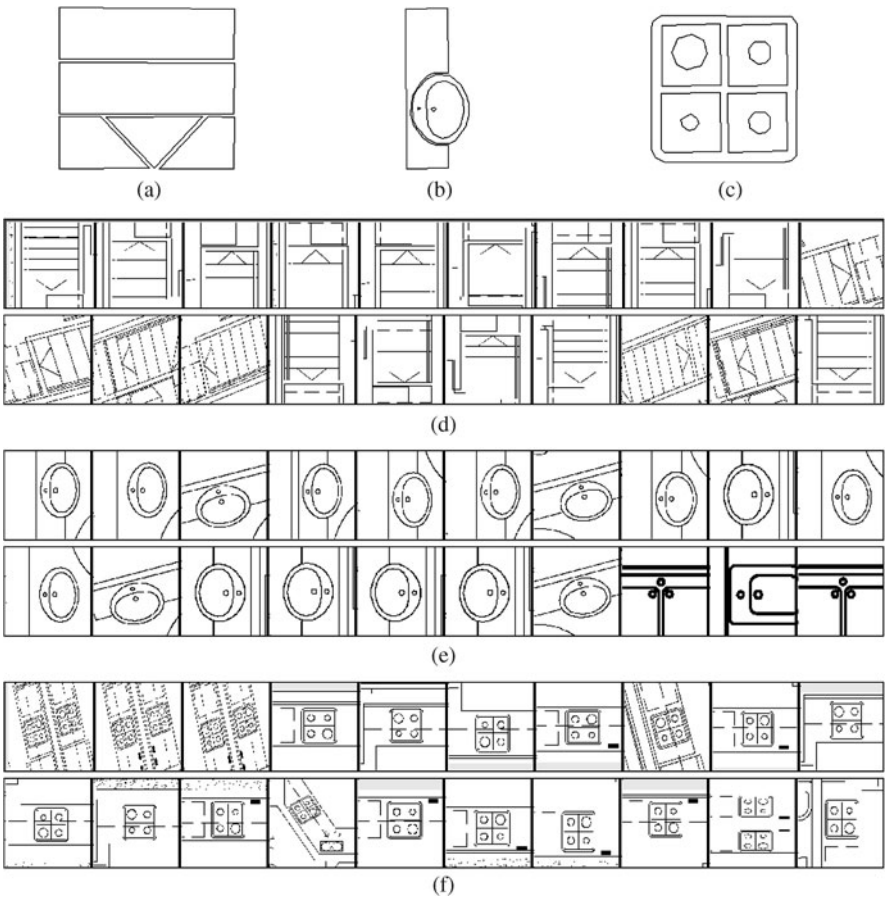

Fig. 6.7 Qualitative results of the relational indexing method (2). (**a**) Query symbol *stairs*; (**b**) query symbol *sink*; (**c**) query symbol *burners*; (**d**), (**e**) and (**f**) first 20 retrieved regions when querying the symbols of (**a**), (**b**) and (**c**), respectively

Table 6.1 Recognition results of the relational indexing method

Description	*TPR* (%)	*FP*	Time (s/plan)
Simple ratios	**93.62**	153.42	3.44
Hu's boundary moments	91.3	76.76	0.71
Line segment moments	55.62	63.89	**0.55**
Fourier descriptor	73.33	**58.76**	0.78

the symbols or just numerical descriptions of a symbol. The presented approach uses both structural and numerical information. The use of both information sources increases the robustness of the method. It also aims at using very simple descriptors with good results according to the user needs.

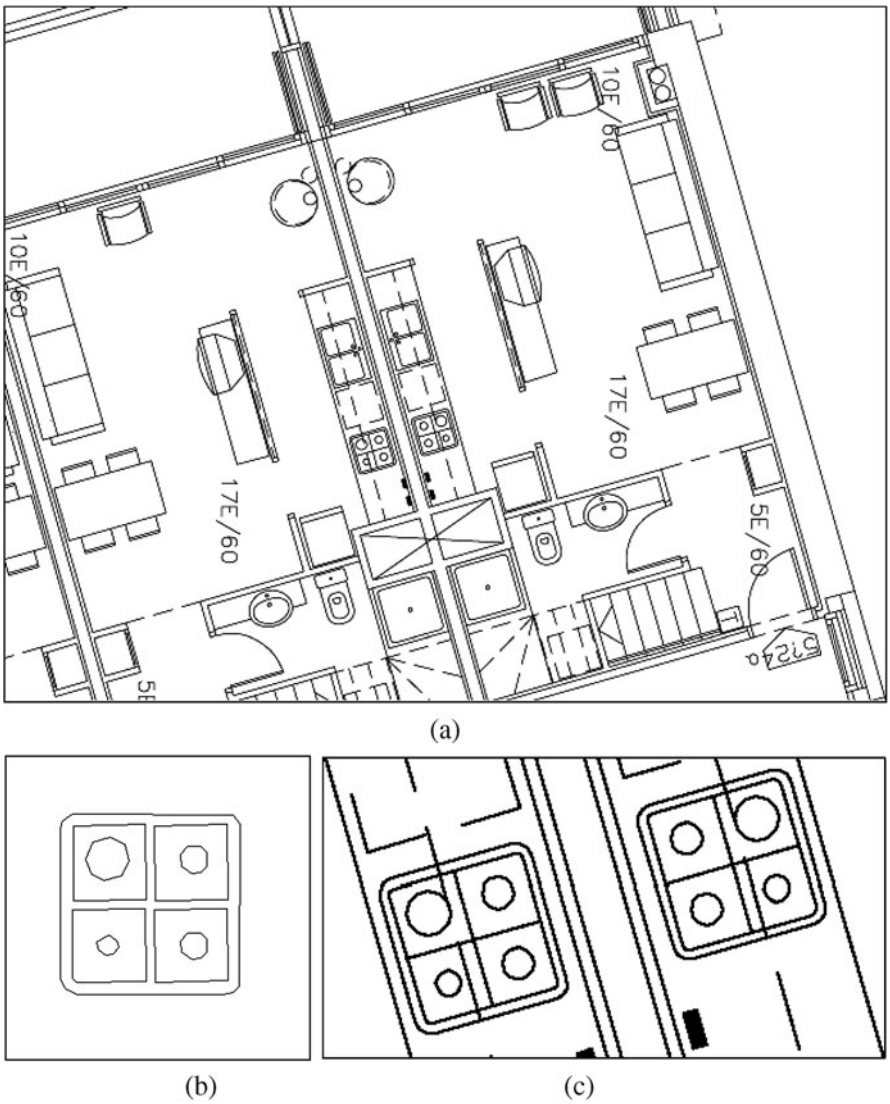

Fig. 6.8 Illustration of a result which is difficult to evaluate. (**a**) Floor-plan image in the collection; (**b**) queried symbol; (**c**) retrieved region

There is obviously still some room for improvements. By describing symbols with closed regions, we make the assumption that the symbols are composed of several loops. This may not be the case in certain graphic-rich documents. In such cases, another primitive extraction process should be considered.

In some application domains, as, for instance, in the case of complex electronic diagrams, some symbols share a substantial part of their design and only differ by slight details. Symbols may also be composed of other known and significant sym-

bols. In this context, the proposed focused retrieval methodology might result in an important number of false alarms.

References

1. Ballard, D.: Generalizing the Hough transform to detect arbitrary shapes. Pattern Recognition **13**(2), 111–122 (1981)
2. Califano, A., Mohan, R.: Multidimensional indexing for recognizing visual shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence **16**(4), 373–392 (1994)
3. Chang, C., Lee, S.: Retrieval of similar pictures on pictorial databases. Pattern Recognition **24**(7), 675–681 (1991)
4. Chen, C.: Improved moment invariants for shape discrimination. Pattern Recognition **26**(5), 683–686 (1993)
5. Costa, M., Shapiro, L.: 3D object recognition and pose with relational indexing. Computer Vision and Image Understanding **79**(3), 364–407 (2000)
6. Gaede, V., Günther, O.: Multidimensional access methods. ACM Computing Surveys **30**(2), 170–231 (1998)
7. Hu, M.: Visual pattern recognition by moment invariants. IRE Transactions on Information Theory **8**, 179–187 (1962)
8. Hupkens, T., de Clippeleir, J.: Noise and intensity invariant moments. Pattern Recognition Letters **16**(4), 371–376 (1995)
9. Kauppinen, H., Seppänen, T., Pietikäinen, M.: An experimental comparison of autoregressive and Fourier-based descriptors in 2D shape classification. IEEE Transactions on Pattern Analysis and Machine Intelligence **17**, 201–207 (1995)
10. Lambert, G., Gao, H.: Line moments and invariants for real time processing of vectorized contour data. In: Image Analysis and Processing, *Lecture Notes on Computer Science*, vol. 974, pp. 347–352. Springer, Berlin (1995)
11. Lambert, G., Gao, H.: Discrimination properties of invariants using the line moments of vectorized contours. In: Proceedings of the Thirteenth International Conference on Pattern Recognition, pp. 735–739. IEEE Computer Society, Los Alamitos (1996)
12. Lladós, J., Sánchez, G.: Indexing historical documents by word shape signatures. In: Proceedings of the Ninth International Conference on Document Analysis and Recognition, pp. 362–366. IEEE Computer Society, Los Alamitos (2007)
13. Nievergelt, J., Hinterberger, H., Sevcik, K.: The grid file: An adaptable, symmetric multikey file structure. ACM Transactions on Database Systems **9**(1), 38–71 (1984)
14. Rosin, P., West, G.: Segmentation of edges into lines and arcs. Image and Vision Computing **7**(2), 109–114 (1989)
15. Russ, J.: The Image Processing Handbook. CRC Press, Boca Raton (1995)
16. Sardana, H., Daemi, M., Ibrahim, M.: Global description of edge patterns using moments. Pattern Recognition **27**(1), 109–118 (1994)
17. Sivic, J., Russell, B., Efros, A., Zisserman, A., Freeman, W.: Discovering objects and their localization in images. In: Proceedings of the Tenth IEEE International Conference on Computer Vision, pp. 370–377. IEEE Computer Society, Los Alamitos (2005)
18. Smith, O., Makani, K., Krishna, L.: Sparse solutions using hash storage. IEEE Transactions on Power Apparatus and Systems **91**(4), 1396–1404 (1972)
19. Stein, F., Medioni, G.: Structural indexing: Efficient 2D object recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **14**(12), 1198–1204 (1992)
20. Stoyan, D., Stoyan, H.: Fractals, random shapes and point fields (methods of geometrical statistics). Wiley, Chichester (1994)
21. Zahn, C., Roskies, R.: Fourier descriptors for plane closed curves. IEEE Transactions On Computer **21**(3), 269–281 (1972)
22. Zhang, D., Lu, G.: Review of shape representation and description techniques. Pattern Recognition **37**, 1–19 (2004)