Relational Indexing of Vectorial Primitives for Symbol Spotting in Line-Drawing Images

Marçal Rusiñol[*], Agnés Borràs, Josep Lladós

Computer Vision Center, Dept. Ciències de la Computació
Edifici O, Univ. Autònoma de Barcelona, 08193 Bellaterra (Barcelona), Spain

Abstract

This paper presents a symbol spotting approach for indexing by content a database of line-drawing images. As line-drawings are digital-born documents designed by vectorial softwares, instead of using a pixel-based approach, we present a spotting method based on vector primitives. Graphical symbols are represented by a set of vectorial primitives which are described by an off-the-shelf shape descriptor. A relational indexing strategy aims to retrieve symbol locations into the target documents by using a combined numerical-relational description of 2D structures. The zones which are likely to contain the queried symbol are validated by a Hough-like voting scheme. In addition, a performance evaluation framework for symbol spotting in graphical documents is proposed. The presented methodology has been evaluated with a benchmarking set of architectural documents achieving good performance results.

Key words: Document image analysis and recognition, Graphics recognition, Symbol spotting, Vectorial representations, Line-drawings.

1. Introduction

Nowadays, a lot of information still resides in paper format. The process of digitizing these document collections is justified for space saving and preservation issues. However, as pointed in (Tombre and Lamiroy, 2008), the design of efficient and reliable methods for browsing and querying these image collections is still a challenge. Indexing mechanisms which organize the information extracted by the analysis of the document images are essential in order to improve the accessibility to these large collections.

Digital libraries containing mostly text documents require a first step converting printed text into ASCII characters. The conversion into ASCII character encoding allows to retrieve contents from the collection by the use of textual queries. However, there is an emerging interest in extracting content information without the need of fully recognizing all the text with an OCR either for complexity issues or because the document information is not represented by typewritten characters. For instance, in (Rath and Manmatha, 2003) a word detection method is presented aiming to localize several keywords in an historical document image database. In (van Beusekom et al., 2006) a document categorization system is presented on the basis of analyzing the document layout. While in (Sun et al., 2008) a collection of electronic documents are categorized in terms of the presence of a set of watermarks. Finally, in (Journet et al., 2008) the analysis of texture features aim to categorize historical documents. However, since most of these works mainly focus on textual document, they are not applicable when we have to deal with documents containing a large amount of graphical information. In this paper, we have focused our research on a framework dealing with technical line-drawings such as architectural floor-plans.

Line-drawings are digitally-born documents which are generated with computer-aided design software. These software use vectorial primitive entities such as points, lines, polylines, circles, arcs, etc. instead of the pixels as in raster images. The vectorial representation has several advantages. The main interest of vectorial images is that they can be geometrically transformed without loss of detail while bitmap images degrade with these transforms. This makes possible to apply indefinitely zooms or rotations without any resolution losses. In addition, the use of vectorial primitives offers a compact data representation and the documents can be easily edited and modified.

In this paper we propose a method aiming to efficiently query line-drawings in terms of the graphical symbols they contain. Let us further describe in detail the problem to tackle.

1.1. Problem definition

Generally speaking, the *Symbol Spotting* problem can be defined as the location of a set of regions of interest from a document image which are likely to contain an instance of a certain queried symbol without explicitly recognizing it (Tombre and Lamiroy, 2003). One of the main applications for symbol spotting methods is its use in large collections of documents. This particular application can be seen as a Content Based Image Retrieval (CBIR) application, but it involves some particularities. The main difference is that standard document retrieval approaches find atomic documents leaving to the user the task of locating the real relevant information within the provided results. Whereas symbol spotting provides the user a more direct

[*]Corresponding author. Tel.: +34-93-581-40-90; Fax: +34-93-581-16-70.
Email addresses: marcal@cvc.uab.es (Marçal Rusiñol), agnesba@cvc.uab.es (Agnés Borràs), josep@cvc.uab.es (Josep Lladós)

access to relevant information by returning the set of regions of interest which are sub-parts of the documents in the collection that contains the desired information. Such applications which return passages of interest within documents instead of complete documents, are known as *Focused Retrieval* systems. The interested reader is referred to the review on the topic of focused retrieval presented in (Joty and Sadid-Al-Hasan, 2007). This particularity provokes that spotting architectures are meant to recognize and segment the objects at the same time. In addition, spotting systems are usually queried by example. That is, the user segments an object he wants to retrieve from the document database and this cropped image acts as input of the system. This particularity reinforces the fact that spotting methods should not work for a specific set of model symbols nor have a learning stage where the relevant features describing a certain symbol are trained. The retrieval of the relevant zones should be done on-the-fly as in (Wenyin, 2009). Nevertheless, in the acquisition step, i.e. when a given document is added to the collection (which is a process that is done off-line) several steps of primitive extraction and description are computed. The desired output of the spotting methods is a ranked list of zones of interest likely to contain similar symbols to the queried one. That is, each result should have an associated confidence value depending on a certain similarity function between the query and the result.

In order to be efficient, spotting processes require a querying mechanism based on an indexing strategy over the primitive description space. Descriptors require to be simple and compact in order to be able to be efficiently organized in the indexing structure. The simplicity of the description technique can entail a precision loss. If more precise recognition rates are required, a more sophisticated recognition approach could afterwards focus on each of these zones of interest.

We can find in the literature several approaches to face the symbol spotting problem. However, most of them work with raster images instead of using the vectorial format. The analysis and recognition of vectorial primitives entail some specific difficulties. Line-drawings in paper format need to be digitized and then a raster-to-vector conversion process has to be applied to obtain the drawings in vectorial format. The obtained vectorial representations from the conversion step are usually quite unstable in terms of artifact appearances, segment fragmentation, errors in junctions, etc. which introduce a lot of noise. On the other hand, since the shapes are compactly represented by vectorial primitives, the amount of features we can use to describe the shapes is much lower than the features we can extract from a pixel-based representation.

1.2. Related work

Among the *Graphics Recognition* community, a lot of efforts have been devoted in the last years to the problem of locating elements in graphics-rich document images. The first attempts to build systems able to recognize and locate graphical symbols like (Barbu et al., 2005; Lladós et al., 2001; Messmer and Bunke, 1996), relied on a graph based representation of the document images. These methods focused on a structural definition of the graphical symbols. Subgraph isomorphism techniques were then proposed to locate and recognize graphical symbols with a single step. However these approaches do not seem suitable when facing large data collections since graph matching schemes are computationally expensive.

Realizing that the computational cost has to be taken into account, several works like (Dosch and Lladós, 2004; Rusiñol and Lladós, 2006; Wenyin et al., 2007) were centred on computing symbol signatures in some regions of interest of the document image. Obviously, these methods are quicker than graph matching but make the strong assumption that the symbols always fall into a region of interest. In most of the works, the regions of interest are defined by a sliding window. However we can find in the literature some works using other kind of heuristics to extract regions of interest. In (Zuwala and Tabbone, 2006), a dendrogram of junction points is able to determine the regions of interest providing a fast way to spot graphical symbols by the use of signatures describing density. However one of the main drawbacks of the use of signatures is that they are highly affected by noise or occlusions.

Other techniques work with a previous ad-hoc coarse segmentation (Tabbone et al., 2003) between text and graphics, or thick and thin lines to separate symbols from background. Global numerical shape descriptors are then computed at each location and compared against the training set of pixel features extracted from model symbols. These descriptors are much more accurate and provide good results. However, these methods assume that the symbols or objects have been previously segmented as the case of (Tabbone et al., 2001). In order to avoid the use of ad-hoc segmentation strategies, works like (Adam et al., 2000) have proposed the use of digital filters applied for spotting purposes. In this paper the Fourier-Mellin transform is able to extract symbols and characters appearing in complete engineering drawings without segmentation. Nevertheless, both signatures and global descriptors are meant to be computed over all the regions of interest in a sequential way. In the existing literature there is a lack of use of indexing mechanisms when designing spotting systems.

In order to avoid sequential search we have presented in (Rusiñol et al., 2009) a symbol spotting method allowing to avoid the computation of the similarity measure for all the primitives extracted from the collection by means of a lookup table. The use of such indexing structures aims to efficiently access and to retrieve graphic elements by similarity, and becomes a must when dealing with applications which have to face large collections of documents. In the particular use case presented in that paper, we achieved to reduce the amount of distance computations by almost a factor of 45 without missing an important 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 inconvenient when we use symbolic description of primitives as the attributed strings used in our previous method. We propose in this paper to enhance 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 allow in ideal conditions to retrieve items by similarity with a complexity $O(1)$. Moreover, there is another drawback in our previous method. Since graphical symbols are composed of several primitives, querying a symbol consisted in separately querying each of its primitives. The locations showing a higher accumulation of primitives were taken as the most plausible hypotheses 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. We propose in this paper an indexing methodology aiming add structural information in the primitive queries.

Finally, in some domains, graphical objects can be annotated by text labels. In these cases, the spotting mechanism could manage textual queries to provide graphical results as presented in (Lorenz and Monagan, 1995; Syeda-Mahmood, 1999). Another example of the use of textual information is the work presented in (Najman et al., 2001) where technical line-drawings are indexed by the information extracted from the legend. In our work we do not consider textual annotations and thus the spotting method only manages graphical entities.

1.3. Solution outline and contributions

The proposed framework mainly consists of four different steps:

- Preprocessing and primitive description,
- primitive hashing,
- relational querying,
- voting scheme.

Since some documents may be stored in paper format, a scanning process is necessary as the first step. A raster-to-vector algorithm is applied to these images as a preprocessing step to obtain a vectorized representation of the line-drawings. After that, we need to retrieve features from the document in order to compactly represent high level entities. We propose in Section 2 a primitive decomposition and we briefly review a set of off-the-shelf shape descriptors which can be formulated to describe these primitives. In Section 3, these compact representations of symbols are organized in an indexing structure aiming to efficiently retrieve primitives by similarity in order to avoid sequential searches. A relational querying technique is presented in Section 4 together with a voting scheme. The relational indexing strategy aims to combine numerical description of primitives with the spatial relationship among them. The voting scheme aims to validate the hypothesis where a symbols is likely to be found.

The main contribution of this work is twofold. First, the use of indexing structures for symbol spotting in vectorial line-drawings. The proposed segmentation-free recognition allows to query by shape document images, which is useful to browse, categorize and to provide efficient access to large collections of documents. Second, from a methodological point of view, we propose a novel structural approach for indexing vectorial data. In our approach, vectorial primitives are coarsely described by an off-the-shelf shape descriptor. A relational indexing methodology is presented to efficiently recall regions of interest in the document database that have similar relational descriptions to the queried element. In addition, a performance evaluation framework is proposed in Section 5 in order to evaluate both recognition and localization capabilities of the presented method. We can see in Section 6, that the presented spotting methodology achieves good performance results.

2. Description of graphical symbols in terms of vectorial primitives

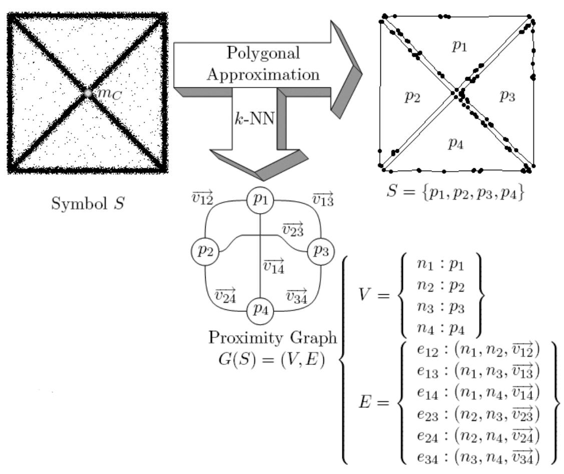
Recognition schemes rely on two basic steps namely primitive extraction and description. First, the primitive extraction step has to transform the image drawings arising from the scanning process to a vector domain. Then, in the second step, such primitives have to be represented by a shape descriptor.

2.1. Vectorial primitives

Graphical symbols are usually composed by the union of several simple sub-shapes. According to that, a symbol can be described in terms of the assembly of sub-shapes which composes 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. The documents are scanned and de-noised by some simple morphological operations. The raster-to-vector algorithm proposed in (Rosin and West, 1989) is then applied to these line-drawing images to obtain a vectorial representation of the documents. However, vectors as it, are not suitable to be used as primitives due to its instability in terms of artifacts, fragmentation, errors in junctions, etc. A higher level entity has to be used as primitive. Adjacent vectors are merged together into polyline instances. These polylines represent then the sub-shapes conforming a given graphical symbol. In our method, we use the contour of the closed loops conforming a symbol as the primitives to polygonally approximate and to merge as single polylines. Note that these primitives are only valid if the symbols appearing in the line-drawings are composed by closed loops. In our case scenario, most of the symbols we can find in floor-plan documents fit the assumption that they are formed by several loops. In other kinds of documents, other primitives as key-points, contours, skeletons, etc. should be used to describe the symbols. Anyway, our spotting architecture is independent of the chosen primitives and can be applied no matter which primitive is taken to represent the symbols.

Formally, let $p = \{s_1 ... s_n\}$ be a polyline consisting of n segments s_i. A symbol is represented in terms of its polylines representing loops and denoted as $S = \{p_1 ... p_m\}$. The gravity



Polygonal Approximation
m_C
k-NN
Symbol S
p_1
p_2
p_3
p_4
$S = \{p_1, p_2, p_3, p_4\}$
$\overrightarrow{v_{12}}$
$\overrightarrow{v_{13}}$
$\overrightarrow{v_{23}}$
$\overrightarrow{v_{14}}$
$\overrightarrow{v_{24}}$
$\overrightarrow{v_{34}}$
Proximity Graph $G(S) = (V, E)$
$V = \{ n_1 : p_1, n_2 : p_2, n_3 : p_3, n_4 : p_4 \}$
$E = \{ e_{12} : (n_1, n_2, \overrightarrow{v_{12}}), e_{13} : (n_1, n_3, \overrightarrow{v_{13}}), e_{14} : (n_1, n_4, \overrightarrow{v_{14}}), e_{23} : (n_2, n_3, \overrightarrow{v_{23}}), e_{24} : (n_2, n_4, \overrightarrow{v_{24}}), e_{34} : (n_3, n_4, \overrightarrow{v_{34}}) \}$

Figure 1: Primitive symbol decomposition. A graphical symbol is decomposed in sub-shapes which are polygonally approximated. An attributed proximity graph is the basis for the relational indexing.

center of the symbol is computed as the average of the gravity centers of each polyline, and it is denoted as 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 compounds 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.

We can appreciate in Fig. 1 how the different parts of a symbol are detached making the loops meaningful primitives, and how their spatial organization is also an important cue to describe the symbol under analysis.

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

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

2.2. *Symbol description*

Formally speaking, given a symbol $S = \{p_1...p_m\}$ 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 conforming primitives. Let us briefly review in the next section the used shape descriptors.

2.3. *Global numerical shape descriptors*

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 the review of shape representation and description techniques by (Zhang and Lu, 2004). 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 different shape descriptors. Depending on the user's needs, no matter which descriptor can be chosen and plugged into the system instead of the ones we use here. The selection of one or another shape descriptor is application dependent. For example, if we are interested in retrieving just 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 really giving importance to the presence of false positives, one must choose a simpler shape descriptor. Three 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. Firstly we introduce some basic notation. We consider an image $I(x, y)$ containing the object shapes 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 $f(O)$. In our case, the objects O will be the primitives composing a symbol. Let us briefly introduce the well-known shape descriptors we use.

2.3.1. *Moment invariants*

The set of seven invariants proposed in (Hu, 1962) involving moments up to third order, are widely used as shape descriptors. Let us see how these invariants can be computed for a vectorial primitive. The central $(p+q)$th order moment for a digital image $I(x, y)$ is expressed by

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q I(x, y) \tag{1}$$

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

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

The geometric moments can also be computed on the contour of the object as introduced in (Chen, 1993; Sardana et al., 1994) by using eq. 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 eq. 2 with

$\gamma = p + q + 1$. When the contours of the objects are polygonally approximated, the geometric moments can be formulated for line segments as introduced in (Lambert and Gao, 1995; Lambert and Noll, 1996). Finally, the invariance to rotation is achieved by using the set of seven functions proposed by Hu. The feature vector $f(O)$ is composed by these seven invariants after applying the set of normalization functions presented in (Hupkens and de Clippeleir, 1995) with monotonic rescaling to get each value into similar numerical ranges and achieve a better robustness to noise.

2.3.2. Simple shape description ratios

Shapes are also commonly coarsely described by the use of some simple ratios. The *eccentricity* 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 eq. (1) as

$$ecc = \frac{\mu_{20} + \mu_{02} + \sqrt{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}}{\mu_{20} + \mu_{02} - \sqrt{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}} \tag{3}$$

The *circularity* 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 lesser than 1. It is computed as

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

Obviously, there are many other shape ratios describing certain geometrical properties. The interested reader is referred to (Russ, 2002; Stoyan and Stoyan, 1994). In our case, we only use these two ratios as the feature vector describing a shape.

2.3.3. Fourier descriptors

Finally, the use of the Fourier transform is also a well-known method to describe shapes and can be easily adapted to vectorial primitives. Given a polyline p_O which is the polygonal approximation of the boundary of a shape O, we use as a vectorial shape signature the centrical distance function computed as

$$r_i = \sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2} \ \forall \ (x_i, y_i) \in p_O. \tag{5}$$

In (Zahn and Roskies, 1972), a Fourier descriptor of a shape is obtained by applying the Fourier transform on the signature representing the shape boundary. Sampling r_i to $N = 2^n$ samples so the use of the FFT is possible, the feature vector of the Fourier descriptor is given by

$$f(O) = \left[\frac{|F_1|}{|F_0|} \cdots \frac{|F_{N/2}|}{|F_0|}\right] \tag{6}$$

where F_i corresponds to the ith component of the Fourier spectrum. Other shape signatures as curvature or complex coordinates can be used to compute the Fourier descriptor. The interested reader is referred to (Kauppinen et al., 1995).

Let us study in the next section how to adapt classical indexing structures used in the databases field to index graphical symbols in a document database.

3. Multidimensional hashing to index primitives

Shape recognition methods suffer from a huge constraint. As the number of considered shape models is increased, the computational cost of the matching step can be unaffordable. As pointed in (Califano and Mohan, 1994), in order to avoid a brute-force matching step, the use of indexing paradigms becomes necessary. In the data mining field, the study and research of efficient indexing structures is a quite prolific topic. Large databases need indexing structures to support efficient data search and retrieval. However one of the main characteristics of shape recognition stems from the use of large feature vectors describing the shapes. Whereas the performance of a shape descriptor usually is improved by the use of larger feature vectors, most indexing algorithms do not work effectively and efficiently in high-dimensional spaces. This is due to the so-called curse of dimensionality described in (Bellman, 1961). In our work, primitive shapes are coarsely described by small feature vectors which can be efficiently retrieved by similarity from an indexing structure.

From the wide taxonomy of indexing structures (cf. (Gaede and Günther, 1998)), the *point access methods* are the ones which are more 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 have to be traversed and in addition, tree balancing algorithms are needed to maintain an effective search performance. As 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 (Nievergelt et al., 1984), in order to index the vectorial primitives. Let us overview with 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 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 which it belongs to.

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 into the two closest buckets in each dimension.

Usually, the main drawback of hashing techniques are the collisions. Given two different items to store in the database, we have to guarantee that the hash function used to index such items do 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 neighbouring buckets) in order to store in a single entry all the similar primitives. This technique allows to have an efficient retrieval by similarity.

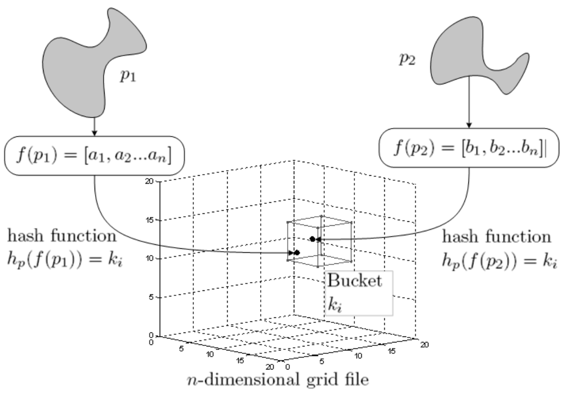


Figure 2: The use of a grid file to index vectorial primitives. The hash function projects the feature vectors into key-indices. Two similar primitives are stored into the same bucket.

In Fig. 2 we can appreciate an overview of how the indexing mechanism works. Formally speaking, a symbol $S = \{p_1 ... p_m\}$ is described by a set of feature vectors $f(p_i) = [x_1 ... x_n]$ for all $i \in [1, m]$ arising from one of the descriptors presented above in section 2.3. A hash function $h_p([x_1 ... x_n]) = k_i$ establish a quantization criterion to apply to each dimension of the feature vector and returns a key-index identifying a certain bucket in the n-dimensional indexing space. The hash function $h_p(f(p_i))$ rounds off each value x of $f(p_i)$ defined over a continuous range to the nearest value in a set of predefined discrete integer values by means of a threshold q.

$$h([x_1 ... x_n]) = [x'_1 ... x'_n] \text{ where } x'_j = \lfloor q \times x_j \rceil \forall j \in [1, n] \quad (7)$$

The feature space is thus simplified to a discrete number (finite) of possible features, each one of those identified by a unique integer number k_i. We have experimentally determined the value of the threshold q by a classification experiment over a set of shapes from the MPEG experiment (Latecki et al., 2000). In this experiment we tried to maximize the clustering of similar shapes under the same bucket and tried to reduce the quantization error given by

$$e = \sum_{i=1}^{n} x_i - \lfloor q \times x_i \rceil \quad (8)$$

by analyzing the results over a ROC space (Fawcett, 2006).

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 $\{k_1 ... k_k\}$ 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 k_i is stored. To summarize this section, the proposed indexing methodology allows to retrieve all the spatial locations where similar primitives than the queried one are likely to be found.

4. Relational indexing and hypothesis 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. We present in this section a relational indexing algorithm to furnish the indexation methodology with spatial information. A voting scheme aiming to validate the spotted locations is also presented.

4.1. Relational indexing

When considering large databases, many symbols may share a substantial part of primitives with many other. Bag-of-words models describe objects in terms of the presence of the primitives which compounds them, ignoring their spatial structure. Recently, a method to locate objects in images using a bag-of-words model has been proposed in (Sivic et al., 2005). The large amount of features taken from interest points aim to discard spatial information. However, in our case, the presence in a given location of a set of primitives do not guarantee the presence of the searched symbol, since symbols are not usually composed by too many primitives. The geometrical configuration of these primitives is a crucial information to refine the zones of interest. Inspired by the work presented in (Costa and Shapiro, 2000), spatial relationships among primitives are also considered when indexing in order to obtain much more valid hypothesis.

Given a symbol represented by a set of primitives $S = \{p_1 ... p_m\}$, the similar primitives appearing in a document can be retrieved by using the set of key-indices $\{k_1 ... k_k\}$. 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. An example on the use of relational indexing is shown in Fig. 3.

In order to efficiently retrieve all the edges of a query symbol, a hash table H_R is used to store in memory the adjacency matrix of the proximity graphs. 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 presented in (Smith et al., 1972). The entry of the table $H_R[k_a, k_b]$ stores all the possible edges e_{ij} where the primitive stored in 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

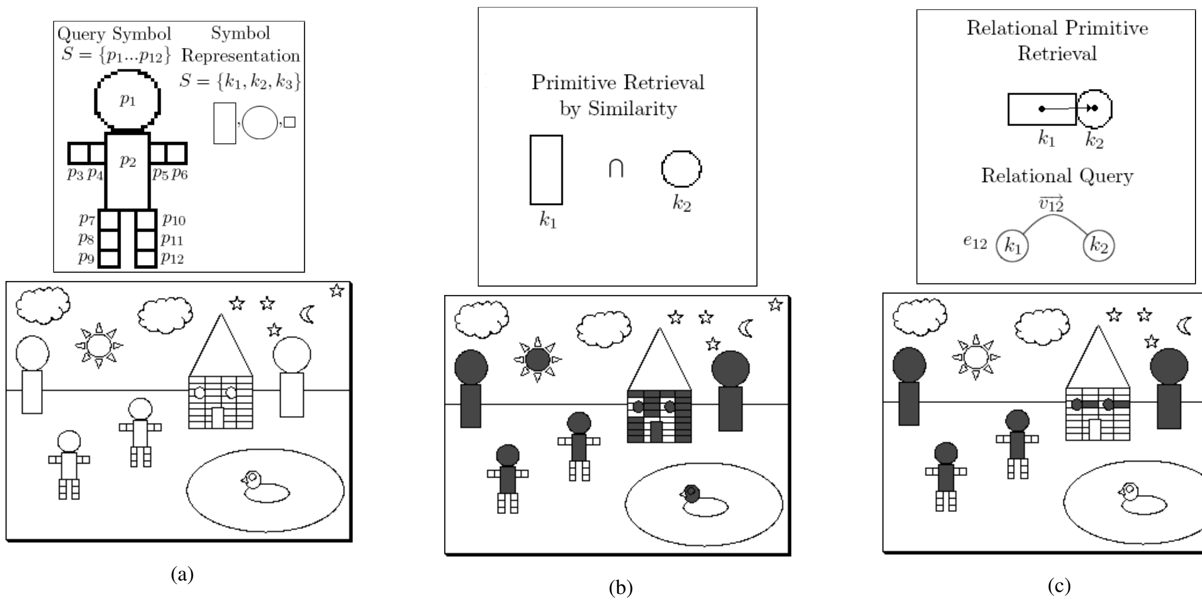


Figure 3: Relational indexing. For the sake of visibility, only two primitives p_1 and p_2 are queried. (a) Sample line-drawing and the query symbol; (b) results of retrieving a couple of primitives by similarity without taking into account the spatial information, the resulting primitives are highlighted in gray; (c) retrieving the same two primitives by using the relational indexing mechanism.

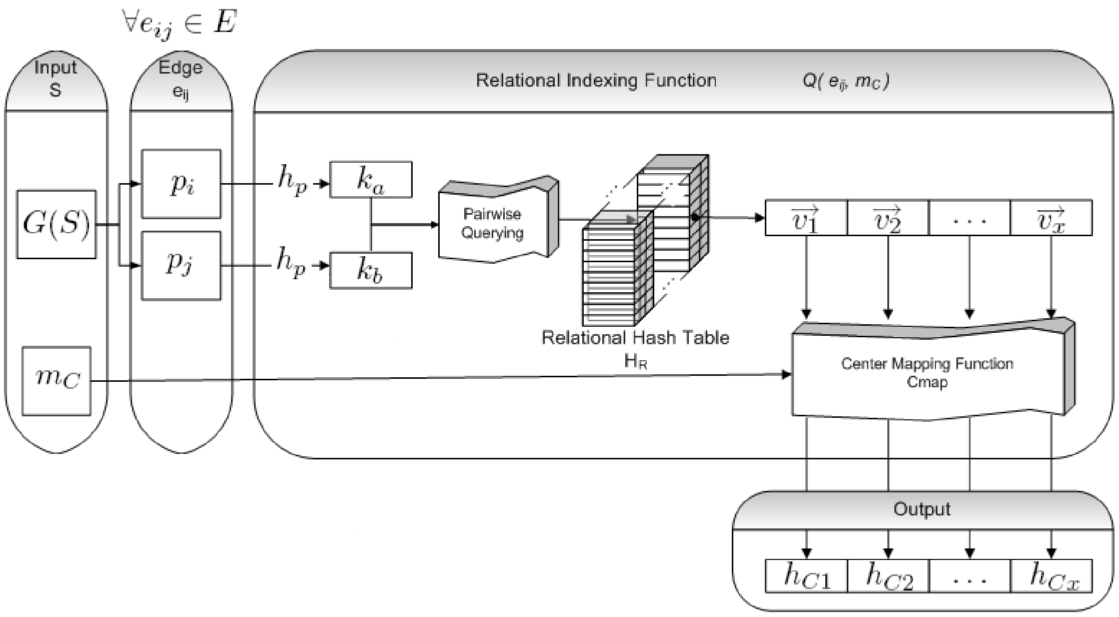

$\forall e_{ij} \in E$
Input S
Edge e_{ij}
Relational Indexing Function
$Q(e_{ij}, m_C)$
$G(S)$
p_i
p_j
h_p
k_a
k_b
Pairwise Querying
Relational Hash Table H_R
$\overrightarrow{v_1}$ $\overrightarrow{v_2}$ ... $\overrightarrow{v_x}$
Center Mapping Function Cmap
m_C
Output
h_{C1} h_{C2} ... h_{Cx}

Figure 4: Relational indexing framework. 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.

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 $Lh_C = [h_{C1}...h_{Cx}]$ where to find the two primitives with a given pose. We can see in Fig. 4 how this function proceeds. The key-indices representing the primitives stored in 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 e_{ij}, and most importantly its associated 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 the vector $\overrightarrow{v_i}$. We can see an example of the hypothetic center location in Fig. 5. Note that the center mapping process align the query edge to the retrieved edges in the line-drawing database, thus being invariant to scale and rotation transforms.

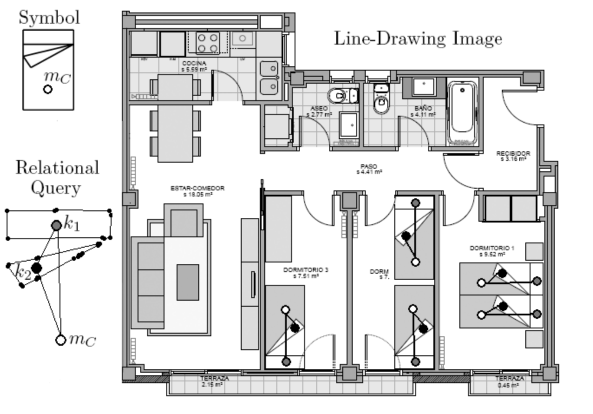


Figure 5: Center mapping function to find the pose of the hypothetic centers given an edge of the relational query and the gravity center of the query symbol.

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.

4.2. *Voting scheme*

Following the idea of the Generalized Hough Transform (GHT) (Ballard, 1981), each of these centers accumulate 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 than the query. In the voting space, the coherent votes tend to form salient peaks, the rest of votes will be scattered in different locations. A simple ranking of the vote clusters result in the positions of the documents where it is more feasible to find the queried symbol.

The querying process leads to consider each pair of primitives of the queried symbol $S = \{p_1 ... p_m\}$, 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 amount of position vectors are stored at each table entry. Obviously, the x value is directly related to the number of documents stored in the library. That results that for each query symbol we have

$$x \cdot C_2^m = x \cdot \binom{m}{2} = x \cdot \frac{m!}{2(m-2)!} \tag{9}$$

centers where to accumulate votes. The locations where the votes are casted are sorted and returned as the retrieved regions of interest. Note that no threshold to decide whether a symbol is present or not is used. Let us further describe in the next Section how we proceed to evaluate a symbol spotting method.

5. Performance evaluation

Symbol spotting can be seen as a particular case of the *Information Retrieval* problem. We base our performance measures on those used in this field. In the retrieval problems, most measures to evaluate the effectiveness are based on a binary labelling of relevance of the items, namely that every item is considered as relevant (*rel*) or non-relevant ($\overline{rel}$), and a binary retrieval notion, either an item is retrieved (*ret*) or not ($\overline{ret}$). To evaluate the spotting system in terms of its abilities to segment and recognize symbols, we reformulate the typical measures of precision, recall, fall-out and generality (see (van Rijsbergen, 1981) for more details) in terms of the amount of overlapping areas between results and the ground-truth. These ratios are computed as follows

$$\begin{array}{llll} \text{Recall} & R = \frac{A(rel \cap ret)}{A(rel)}, & \text{Precision} & P = \frac{A(rel \cap ret)}{A(ret)} \\ \text{Fall-Out} & Fo = \frac{A(\overline{rel} \cap ret)}{A(\overline{rel})}, & \text{Generality} & G = \frac{A(rel)}{A(tot)} \end{array} \tag{10}$$

We can see in Fig. 6 an example of how to obtain and interpret the precision and recall ratios. In this case, the system only retrieves a single zone of interest, merging the two symbols. We detail in Table 1 the overlapping areas and the computed ratios from this retrieval matrix. As some non-relevant area is retrieved, the precision value does not reach the hundred percent. On the other hand, a portion of a symbol has been missed harming the total recall value. Obviously, the fall-out and the generality ratios are highly dependent on the size of the documents and we will have much more lower values than the shown in the example as the non-relevant area will be much higher.

Precision versus recall and fall-out versus recall plots give information of the correctness of the recognition and the refine level of the segmentation. However, sometimes it is hard to compare different methods by a couple of numbers and some composite measures have been used to rank the methods under study. The average precision *Ave*P uses each precision value after truncating the result list after each relevant item, it is computed as follows

$$Ave\text{P} = \frac{\sum_{n=1}^{N}(P@n \times \text{r}(n))}{|rel|} \tag{11}$$

Being N the number of retrieved areas, $P@n$ the precisions at a certain cut-off rank n, and r(n) a binary function on the relevance of a given rank n. Another classical composite measure is the F-score which is the weighted harmonic mean of precision and recall, computed as follows

$$F_\beta = \frac{(1+\beta^2) \times P \times R}{(\beta^2 \times P) + R} \tag{12}$$

We finally complement these measures by giving the recognition rate at symbol level and the amount of false positives. These measures are only meant to evaluate the symbol recognition task despite the localization ability. We will only consider a binary concept of retrieval, either a symbol is found or not. If the overlapping between a resulting polygon and the ground-truthed representation of a symbol is more than a given

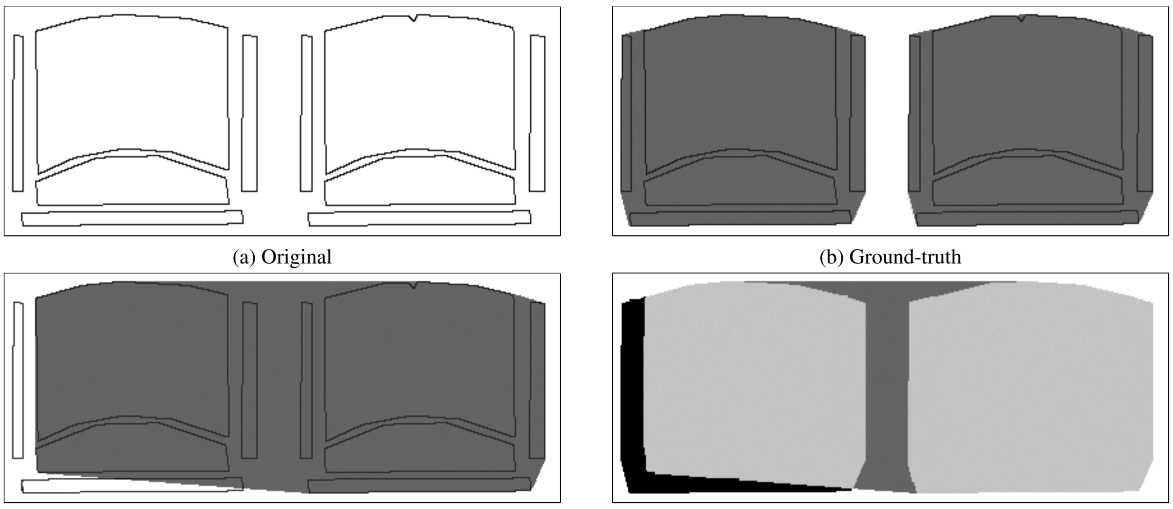


Figure 6: Original image (a), its ground-truth (b) and the result (c) of a spotting system. The overlapping between results and ground-truth (d) is labelled whether $ret \cap rel$ (light gray), $ret \cap \overline{rel}$ (dark gray) or $\overline{ret} \cap rel$ (black).

Table 1: Retrieval matrix for example in Fig. 6

	Relevant	Non-Relevant	TOTAL
Retrieved	55449	6858	62307
Not Retrieved	5111	10447	15558
TOTAL	60560	17305	77865

Precision = 88.99%
Recall = 91.56%
Fall-Out = 39.63%
Generality = 77.77%

threshold we will consider the symbol as recognized. On the other hand, if the resulting polygon has less overlapping with the ground-truth than the threshold, the result is considered as a false positive response. More details about the proposed protocol for evaluating the performance of symbol spotting systems can be found in (Rusiñol and Lladós, 2009).

6. Experimental results and discussion

Let us first introduce the dataset we use for the spotting experiments.

6.1. Dataset

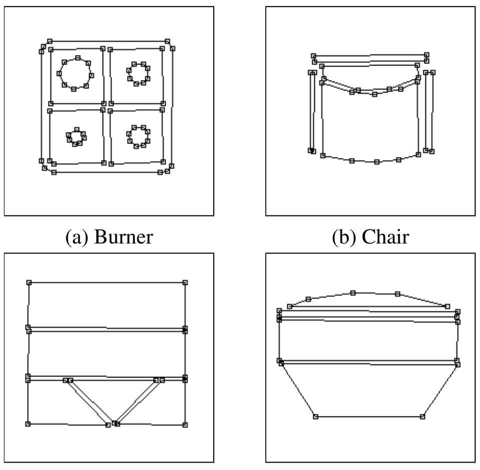
To perform the experimental results we worked with a collection of architectural floorplans consisting of 42 images (of 3215×2064 pixels in average) arising from four different projects. These images are polygonally approximated resulting in a collection of vectorial documents. The symbols taken into account for these experiments are divided in 38 classes and we have in total 344 instances in the document images. In a single document image the average of symbols is around 8 and it goes from 0 to 28 symbols. The models to query the document database are cropped from the document images, so they also contain vectorial distortions. We can see in Fig. 7 some of the symbols taken as models.

To build the ground-truth, an annotation tool has been developed. The user can select symbols and label them. The convex hull (Barber et al., 1996) containing all the points of all the polylines composing the selected symbol is taken as the minimum area of interest containing the symbol. The convex hull coordinates and the symbol category as well as other information about the document are stored in a XML file (following the same file structure used for page layout ground-truth by (Antonacopoulos et al., 2006) containing the information about the whole library[1].

6.2. Spotting Results

Let us first see some qualitative results. When querying a model symbol against the database, the convex hull of the activated polylines in the documents conform a set of regions of interest which are sorted by confidence value depending on the number of received votes. We can see in Fig. 8 the first five results of querying several symbols in a given document. As we can appreciate, all symbols are found and obviously some areas of false positives appear as we requested more results than appearances of symbols in the document. However we observe two phenomena, usually, two close symbols (i.e. burners in

[1]The vectorial image dataset as well as its ground-truth is public available and can be downloaded through the following website http://www.cvc.uab.cat/~marcal/FPLAN/



(a) Burner (b) Chair (c) Stairs (d) Television

Figure 7: Symbol models.

Fig.8a or chairs in Fig. 8b) are grouped in 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 stairs in Fig.8c). These two phenomenons will of course decrease the precision in terms of retrieved area. Figs. 9 and 10 show the first twenty results when querying several symbols in the whole document collection using the Fourier shape descriptor. As we can appreciate, most of the results correspond to the correct queried symbol, even if some false positives areas appear.

Regarding the quantitative evaluation, we can see in Fig. 11a the precision and recall plot corresponding to the average of querying all the models in the whole collection using the three different primitive description techniques. The starting precision differs significantly from a descriptor to another. Fourier descriptors are much more accurate than the simple ratios. On the other hand, the more accurate is a shape descriptor, the more symbols are missed due to slight shape changes. Simple ratios reach best recall values than accurate shape descriptors. This tradeoff between precision and recall is an indicator of the performance of the system and the user should select a descriptor or another depending on the application needs. The interesting point here is to notice that the tend of the three curves is almost the same despite its precision variances and the final recall value.

Similar responses can be appreciated from the fall-out and recall plot of Fig. 11b. The use of coarser description techniques entails a larger amount of false positives but yields to best recall values. Again, the tend of the three curves is maintained without any sudden changes.

In order to give a better idea of the performance of the system some measures of quality are given in Table 2. Fourier descriptors yield the best average precision as they provide relevant results ranked in the first positions. Boundary moments have the best F-score since they show the most moderate tradeoff between precision and recall. The simple ratios yield the best recognition rates at symbol level since they are able to retrieve the major number of symbols. At symbol level, the Fourier descriptors are the ones which provide less amount of false alarms. In order to evaluate the efficiency of the indexing method, we provide the time taken to perform a query. Note 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 expend more time in retrieving zones of interest since the table entries are more populated and the amount of false positives is also increased. On the other hand, the methods which have less recognition rate but also less false positives, are usually less computationally expensive. In order to check the efficiency gain when using the proposed indexed methodology we performed the same experiment by storing the primitives in a list that implies a sequential access to perform the search of primitives by similarity instead of the presented hashing technique. The results show that in average the use of the hashing technique provides the results near 1200 faster than a sequential access to the primitives and their spatial relationships. Finally, note that the low generality of the dataset explains the low precision values reached by the three methods.

We have performed another experiment aiming to test if the number of primitives of the queried symbol has impact on the final performance of the method. The possible queries have been partitioned into four different group depending on the amount of composing polylines and we can see in Table 3 the recognition rates (RR) and the amount of false positives (FP) for all the different descriptors. We can see that usually, the greater the number of primitives is, the better the system responds. However, there is a side effect on increasing the number of primitives. As we consider more complex symbols the amount of false positives which are returned is also increased.

Finally, note that our work do not focus on the evaluation of the descriptors, the interesting point is to see that the system behaves in the expected way depending on the selected shape descriptors. The choice of one or other descriptor will only affect in the maximum precision and maximum recall leaving the behavior of the system intact. The shape descriptor can be seen as a black box which can be plugged into the system depending on the application needs. Retrieval applications may need better precision values whereas categorization applications are interested in higher recall values.

7. Conclusions

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 aiming 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 compounds a symbol but also the spatial relationship among them. Finally a Hough-like voting scheme aims to validate the hypothesis where a symbol is likely to be found. In addition, a set of measures to evaluate the performance of spotting systems in terms of recognition and localization abilities has been presented.

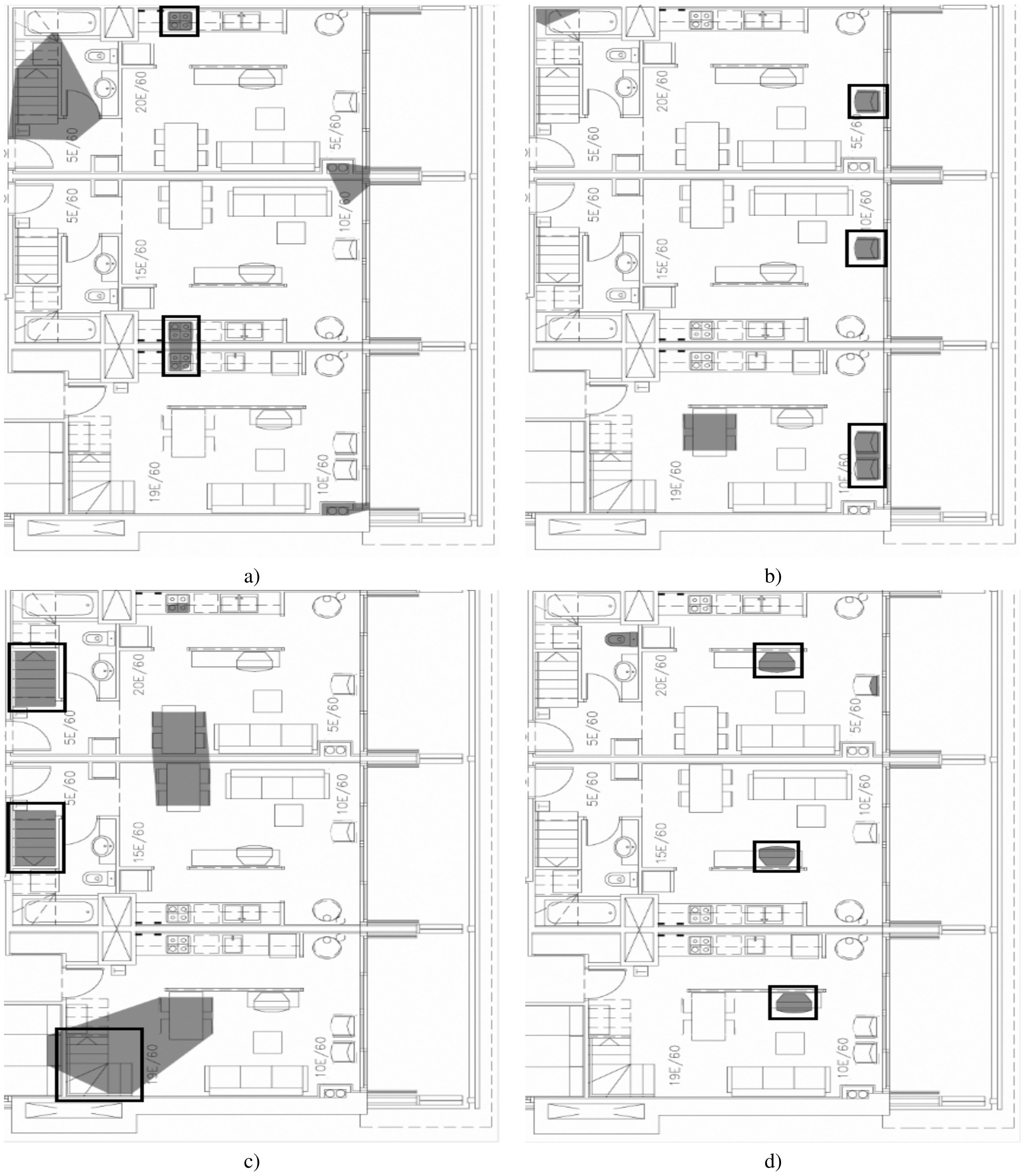


Figure 8: Top five indexed regions of interest when querying the burners (a), the chairs (b), the stairs (c), and the television set (d). The correct results are framed by a rectangle, remaining areas are false positives.

Our feeling is that one of the right directions to follow in spotting-related problems for the next years is the use of coarser descriptors rather than accurate descriptions techniques. We have shown that the combination of coarse description and relational validation, i.e. combining numeric and structural description techniques, yields very good results. In particular, we have proven in this paper that there is no need for high-dimensional descriptors for spotting purposes, and with really simple shape descriptors, we can reach acceptable performances when combining those descriptions with relational information. Obvi-

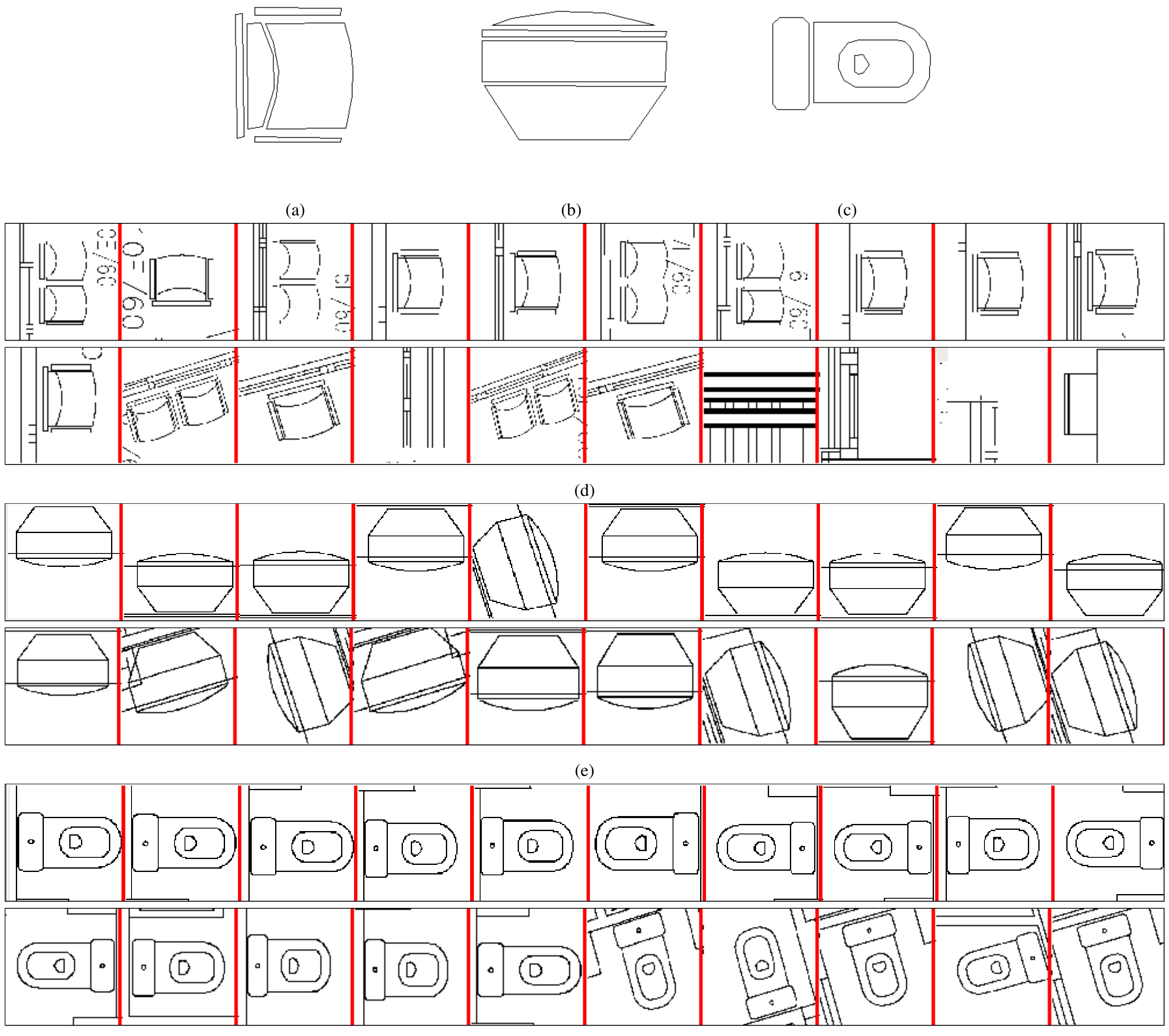


Figure 9: Qualitative results of the relational indexing method (1). (a) Query symbol *chair*; (b) query symbol *TV set*; (c) query symbol *toilet*; (d),(e) and (f) first 20 retrieved regions when querying the symbols (a), (b) and (c) respectively.

Table 2: Measures of quality.

Method	Composite measures		Symbol level measures		Other measures	
	*Ave*P	F_1-score	Rec. rate (%)	False Pos.	Time (secs.)	G (%)
Ecc.	20.08	6.87	**93.62**	153.42	3.44	
Moments	39.77	**23.34**	91.3	76.76	**0.71**	0.16
Fourier desc.	**41.99**	21.45	73.33	**58.76**	0.78	

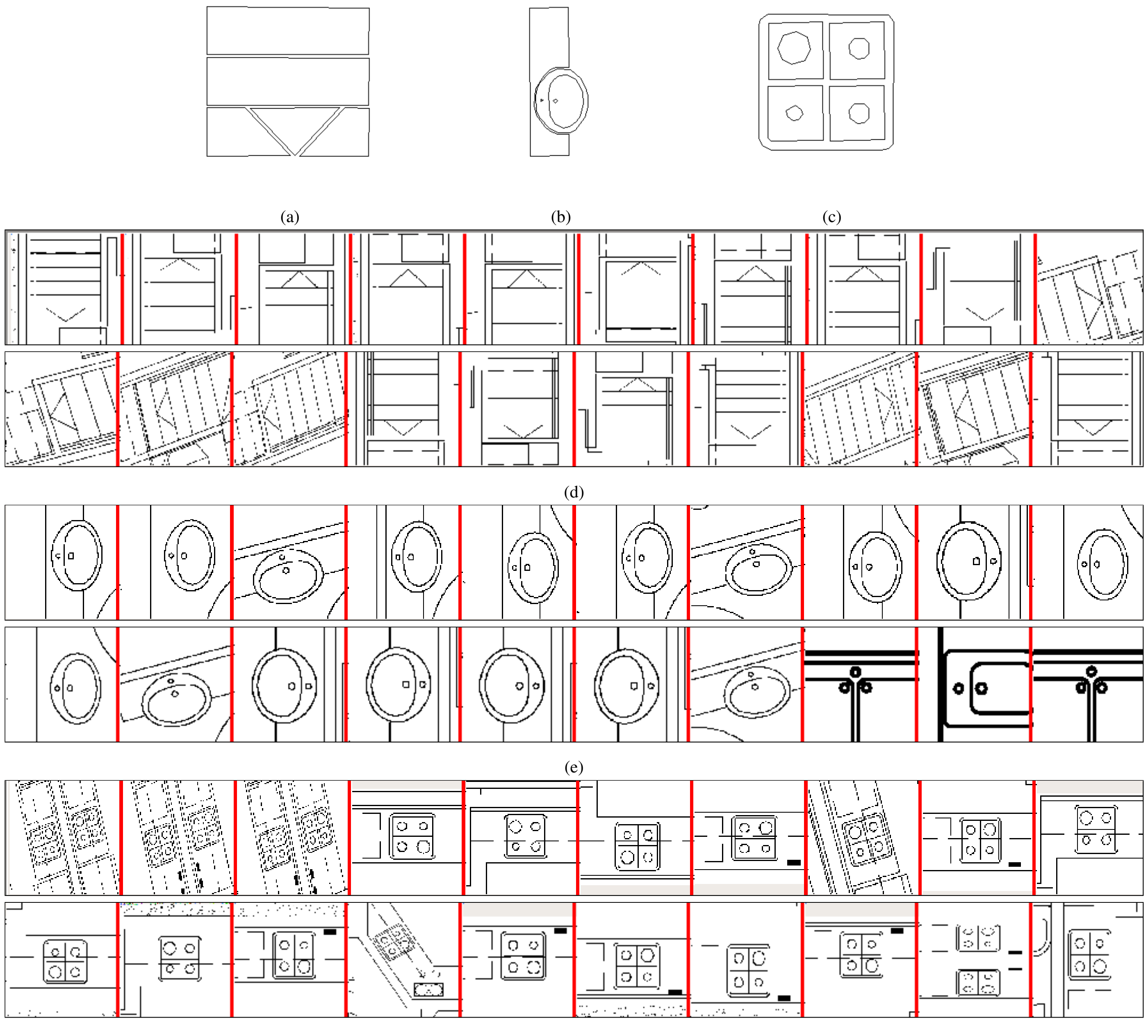
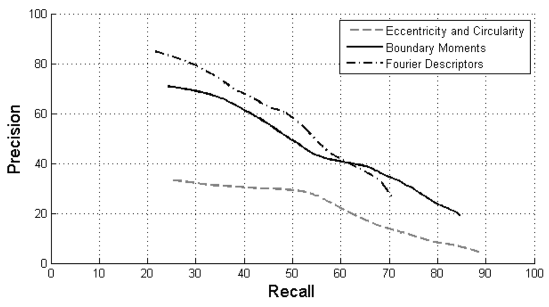
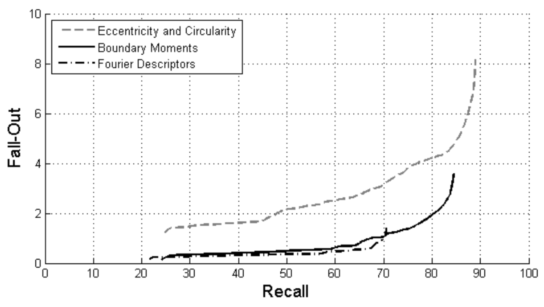


Figure 10: Qualitative results of the relational indexing method (2). (a) Query symbol *stairs*; (b) query symbol *sink*; (c) query symbol *burners*; (d),(e) and (f) first 20 retrieved regions when querying the symbols (a), (b) and (c) respectively.

ously, depending on the intended final application, the word "acceptable" may adopt several meanings. As we have seen in the experimental part, if the user of the final application is interested in retrieving the most of the relevant portions of images from the collection, no matter the number of false alarms, a simpler description should be used. If the user is more interested in a better precision without caring the fact the system misses symbols, then we should start using more and more complex and fine description techniques. However, we strongly believe that for most of applications, the use of low-dimensional descriptors is enough. The choice of such low-dimensional feature vectors avoids the so-called curse of dimensionality and provides an efficient access to the data. The good results obtained by such simple description techniques are also favored by the inclusion of relational and structural information of the graphical symbols. The use of a joint local numerical description and structural analysis contributes to obtain an important discriminative power. However structural information should be added carefully since the analysis of complex structural relationships (entailing comparisons in the graph domain) can not be managed on the context of symbol spotting due to its huge complexity.



(a) Precision and recall



(b) Fall-out and recall

Figure 11: Precision and recall plot (a) and fall-out and recall plot (b) for all symbols in the whole collection.

Table 3: Effect of the number of primitives.

Number of primitives	Method							
	Ecc.		Moments		Fourier desc.		Average	
	RR(%)	FP	RR(%)	FP	RR(%)	FP	RR(%)	FP
$m \leq 4$	90.2	**136.2**	90.2	**54.6**	54.2	**22.3**	78.2	**71**
$5 \leq m \leq 6$	93	158.2	89.6	64.8	68.9	56	83.5	93
$7 \leq m \leq 9$	97.2	173.2	**97.2**	111.8	91.6	101.8	95.3	128.9
$9 \leq m$	**100**	141.5	92	98.6	**94.8**	88.8	**95.6**	109.6

Finally, one of the critical assumptions that we made along this paper is that the graphical symbols can be well represented by a particular primitives, the region contours. Obviously, not in all the cases the symbols are formed by closed loops, and such proposed primitives can not be used. The scalability of the proposed method has to be further investigated by analyzing other kind of documents such as electronic diagrams or mechanical schemes.

Acknowledgments

This work has been partially supported by the Spanish projects TIN2006-15694-C02-02, TIN2009-14633-C03-03 and CONSOLIDER-INGENIO 2010 (CSD2007-00018). The authors would also like to thank the anonymous reviewers for their helpful and constructive comments, as well as the architect E. Farrerons for providing the floor-plan images and vectorial files.

References

Adam, S., Ogier, J., Cariou, C., Mullot, R., Labiche, J., Gardes, J., 2000. Symbol and character recognition: Application to engineering drawings. International Journal on Document Analysis and Recognition 3 (2), 89–101.

Antonacopoulos, A., Karatzas, D., Bridson, D., 2006. Ground truth for layout analysis performance evaluation. In: Document Analysis Systems, DAS. Vol. 3872 of LNCS. Springer Verlag, pp. 302–311.

Ballard, D., 1981. Generalizing the Hough transform to detect arbitrary shapes. Pattern Recognition 13 (2), 111–122.

Barber, C., Dobkin, D., Huhdanpaa, H., 1996. The quickhull algorithm for convex hulls. ACM Transactions on Mathematical Software 22 (4), 469–483.

Barbu, E., Hérroux, P., Adam, S., Trupin, S., 2005. Frequent graph discovery: Application to line drawing document images. Electronic Letters on Computer Vision and Image Analysis 5 (2), 47–57.

Bellman, R., 1961. Adaptive Control Processes. Princeton University Press.

Califano, A., Mohan, R., 1994. Multidimensional indexing for recognizing visual shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (4), 373–392.

Chen, C., 1993. Improved moment invariants for shape discrimination. Pattern Recognition 26 (5), 683–686.

Costa, M., Shapiro, L., 2000. 3d object recognition and pose with relational indexing. Computer Vision and Image Understanding 79 (3), 364–407.

Dosch, P., Lladós, J., 2004. Vectorial signatures for symbol discrimination. In: Graphics Recognition: Recent Advances and Perspectives. Vol. 3088 of LNCS. Springer Verlag, pp. 154–165.

Fawcett, T., 2006. An introduction to ROC analysis. Pattern Recognition Letters 27 (8), 861–874.

Gaede, V., Günther, O., 1998. Multidimensional access methods. ACM Computing Surveys 30 (2), 170–231.

Hu, M., 1962. Visual pattern recognition by moment invariants. IRE Transactions on Information Theory 8, 179–187.

Hupkens, T., de Clippeleir, J., 1995. Noise and intensity invariant moments. Pattern Recognition Letters 16 (4), 371–376.

Joty, S., Sadid-Al-Hasan, S., 2007. Advances in focused retrieval: A general review. In: Proceedings of the Tenth International Conference on Computer and Information Technology, ICCIT07. pp. 1–5.

Journet, N., Ramel, J., Mullot, R., Eglin, V., 2008. Document image characterization using multiresolution analysis of the texture: application to old documents. International Journal on Document Analysis and Recognition 11 (1), 9–18.

Kauppinen, H., Seppänen, T., Pietikäinen, M., 1995. An experimental comparison of autoregressive and Fourier-based descriptors in 2D shape classification. IEEE Transactions on Pattern Analysis and Machine Intelligence 17, 201–207.

Lambert, G., Gao, H., 1995. Line moments and invariants for real tie processing of vectorized contour data. In: Proceedings of the 8th International Conference on Image Analysis and Processing, ICIAP95. pp. 347–352.

Lambert, G., Noll, J., 1996. Discrimination properties of invariants using the line moments of vectorized contours. In: Proceedings of the 13th International Conference on Pattern Recognition, ICPR96. pp. 735–739.

Latecki, L., Lakämper, R., Eckhardt, T., 2000. Shape descriptors for non-rigid shapes with a single closed contour. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR00. pp. 424–429.

Lladós, J., Martí, E., Villanueva, J., 2001. Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (10), 1137–1143.

Lorenz, O., Monagan, G., 1995. A retrieval system for graphical documents. In: Proceedings of the 4th Symposium on Document Analysis and Information Retrieval, SDAIR95. pp. 291–300.

Messmer, B., Bunke, H., 1996. Automatic learning and recognition of graphical symbols in engineering drawings. In: Graphics Recognition Methods and Applications.

Najman, L., Gibot, O., Berche, S., 2001. Indexing technical drawings using title block structure recognition. In: Proceedings of the 6th International Conference on Document Analysis and Recognition, ICDAR01. pp. 587–591.

Nievergelt, J., Hinterberger, H., Sevcik, K., 1984. The grid file: An adaptable, symmetric multikey file structure. ACM Transactions on Database Systems 9 (1), 38–71.

Rath, T., Manmatha, R., 2003. Word image matching using dynamic time warping. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR03. pp. 521–527.

Rosin, P., West, G., 1989. Segmentation of edges into lines and arcs. Image and Vision Computing 7 (2), 109–114.

Rusiñol, M., Lladós, J., 2006. Symbol spotting in technical drawings using vectorial signatures. In: Graphics Recognition. Ten Years Review and Future Perspectives. Vol. 3926 of LNCS. Springer Verlag.

Rusiñol, M., Lladós, J., 2009. A performance evaluation protocol for symbol spotting systems in terms of recognition and location indices. International Journal on Document Analysis and Recognition 12 (2), 83–96.

Rusiñol, M., Lladós, J., Sánchez, G., 2009. Symbol spotting in vectorized technical drawings through a lookup table of region strings. Pattern Analysis and Applications.

Russ, J., 2002. The Image Processing Handbook. CRC Press; 4th edition, Boca Raton.

Sardana, H., Daemi, M., Ibrahim, M., 1994. Global description of edge patterns using moments. Pattern Recognition 27 (1), 109–118.

Sivic, J., Russell, B., Efros, A., Zisserman, A., Freeman, W., 2005. Discovering objects and their localization in images. In: International Conference on Computer Vision, ICCV05. pp. 370–377.

Smith, O., Makani, K., Krishna, L., 1972. Sparse solutions using hash storage. IEEE Transactions on Power Apparatus and Systems 91 (4), 1396–1404.

Stoyan, D., Stoyan, H., 1994. Fractals, Random Shapes and Point Fields (Methods of Geometrical Statistics). John Wiley & Sons, Chichester.

Sun, J., Naoi, S., Fujii, Y., Takebe, H., Fujimoto, K., 2008. An image based watermark string detection system for document security checking. In: Proceedings of the 8th IAPR Workshop on Document Analysis Systems, DAS08. pp. 43–50.

Syeda-Mahmood, T., 1999. Indexing of technical line drawing databases. IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (8), 737–751.

Tabbone, S., Wendling, L., Tombre, K., 2001. Indexing of technical line drawings based on F-signatures. In: Proceedings of the 6th International Conference on Document Analysis and Recognition, ICDAR01. pp. 1220–1224.

Tabbone, S., Wendling, L., Tombre, K., 2003. Matching of graphical symbols in line-drawing images using angular signature information. International Journal on Document Analysis and Recognition 6 (2), 115–125.

Tombre, K., Lamiroy, B., 2003. Graphics recognition - from re-engineering to retrieval. In: Proceedings of the 7th International Conference on Document Analysis and Recognition, ICDAR03. pp. 148–155.

Tombre, K., Lamiroy, B., 2008. Pattern recognition methods for querying and browsing technical documentation. In: Proceedings of 13th Iberoamerican Congress on Pattern Recognition, CIARP08. Vol. 5197 of LNCS. Springer Verlag.

van Beusekom, J., Keysers, D., Shafait, F., Breuel, T., 2006. Distance measures for layout-based document image retrieval. In: Proceedings of the 2nd International Conference on Document Image Analysis for Libraries, DIAL06. pp. 232–242.

van Rijsbergen, C., 1981. Information Retrieval. Butterworth-Heinemann Newton, 2nd edition, MA, USA.

Wenyin, L., 2009. Example-driven graphics recognition. In: Structural, Syntactic, and Statistical Pattern Recognition. Vol. 2396 of LNCS. Springer Verlag.

Wenyin, L., Zhang, W., Yan, L., 2007. An interactive example-driven approach to grahpics recognition in engineering drawings. International Journal on Document Analysis and Recognition 9 (1), 13–29.

Zahn, C., Roskies, R., 1972. Fourier descriptors for plane closed curves. IEEE Transactions On Computer 21 (3), 269–281.

Zhang, D., Lu, G., 2004. Review of shape representation and description techniques. Pattern Recognition 37, 1–19.

Zuwala, D., Tabbone, S., 2006. A method for symbol spotting in graphical documents. In: Document Analysis Systems VII. Vol. 3872 of LNCS. Springer Verlag.