

## Chapter 2

# State-of-the-Art in Symbol Spotting

**Abstract** In this chapter, we will review the related work on symbol spotting which has been done in the last years. We first present a review of the contributions from the Graphics Recognition community to the spotting problem. In the second part, we focus our attention on the different symbol description techniques and the families we can find in the literature. Then, the existing data structures which aim to store the extracted descriptors and provide efficient access to them will be analyzed. We finally review the existing methods for hypotheses validation which can be used for spotting purposes.

### 2.1 Introduction

Generally speaking, the architecture of a symbol spotting system consists of the three main levels outlined in the previous chapter in Fig. 1.2. In the first level, the documents are decomposed into a set of primitives which are characterized by a descriptor capturing the most important cues. The second level is focused on how these descriptors are organized to be posteriorly consulted. Finally, the third level is in charge of validating the hypotheses arising from the matching between model and stored data. This third level shall provide the resulting list of locations where a queried symbol is likely to be found.

We organize this state-of-the-art into four different parts. First, in Sect. 2.2, we briefly review the recent contributions of the Graphics Recognition community to the spotting problem. The subsequent three parts refer to each level of the general symbol spotting architecture. In Sect. 2.3, we focus on the symbol descriptor categorization. Section 2.4 describes the organization and access to the stored descriptors, and in Sect. 2.5 we present the existing approaches for hypotheses validation. We finally summarize the suitable approaches for spotting graphics in Sect. 2.6.

### 2.2 Spotting Graphical Elements

Among the Graphics Recognition community, a lot of efforts have been devoted in the last years to the problem of locating elements in document images. However,

two different applications can be identified, namely locating words in textual image documents in the image domain and identifying regions likely to contain a certain symbol within graphics-rich documents. Although the problem is the same, the proposed methods are very different whether the focus of the application is centered on text or in graphics. Let us briefly review in the next sections the existing work on both word and symbol spotting.

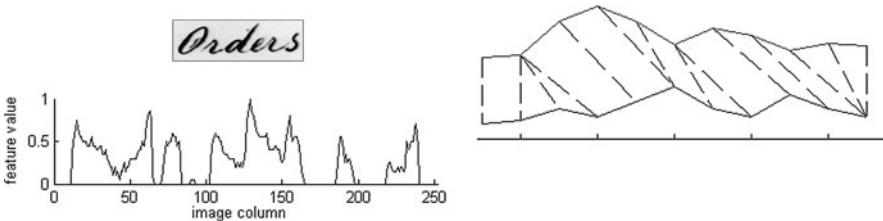
### 2.2.1 Word Spotting

OCR engines benefit from the nature of alphanumeric information, i.e., text strings which are one-dimensional structures with underlying language models that facilitate the construction of dictionaries and indexing structures. Word spotting techniques also take advantage of this aspect and usually represent words as one-dimensional signals which will further be matched against the query word image. The main idea of these approaches is to represent keywords with shape signatures in terms of image features. The detection of the keyword in a document image is usually done by a cross-correlation approach between the prototype signature and signatures extracted from the target document image.

Using image features without word recognition though, the information is still one-dimensional and it facilitates the use of some classical techniques used in speech recognition. Rath and Manmatha [71, 72] presented a method to spot handwritten words. They use the normalized projection profiles of segmented words as word signatures. These word signatures are seen as time series and are aligned using the dynamic time warping (DTW) distance. We can see an example of such approach in Fig. 2.1.

Kuo and Agazzi [45] used another classical technique from the speech processing field. A hidden Markov model (HMM) is applied to spot words in poorly printed documents. In this case, a learning step to train the HMM is needed. In addition, the features describing each word the user wants to query have to be learned previously. By also using HMMs, Rodríguez [74] presents a framework to spot handwritten words.

Lladós and Sánchez [50] proposed a keyword spotting method based on the shape context descriptor. Words are represented by a signature formulated in terms of the



**Fig. 2.1** Word image signature for word spotting using the DTW distance (this image is based on Figs. 2 and 4 appearing in [72])

shape context descriptor and are encoded as bit vectors codewords. A voting strategy is presented to perform the retrieval of the zones of a given document containing a certain keyword.

Leydier et al. [48] presented a word spotting method in order to perform text searches in medieval manuscripts. The orientation of the gradients in a given zone of interest are taken as features describing the local structure of the strokes and the orientation of the characters' contours. A matching process is then proposed to identify and retrieve the locations where a given word is likely to be found. The experimental results show that the proposed method is tolerant to several kinds of noises as well as geometric distortions.

In [42], Konidaris et al. presented another strategy for word spotting. In this strategy, the query is not an image but is an ASCII string typed by the user. Thus, the features which represent a word should be invariant enough to appear in both synthetic characters and the character extracted from the ancient documents. The authors propose a hybrid approach by using the density of pixels in a given zone of the character and the projections of the upper and lower profile of the character. In order to improve the retrieval performance, a user feedback procedure is also proposed.

Recently, Lu and Tan [57] proposed a very simple typewritten word coding which is useful enough to characterize documents. The proposed word code is based on character extremum points and horizontal cuts. Words are represented by simple digit sequences. Several similarity measures based on the frequency of the codes are defined to retrieve documents written in the same language or describing similar topics.

Terasawa and Tanaka [90] presented a word spotting method based on sliding windows. In each window, a histogram of gradients (HOG) feature is computed in order to locally describe parts of a word. The word matching step is done with a dynamic programming algorithm very similar to dynamic time warping.

Finally, Kise et al. [41] addressed another interesting aspect of the word spotting problem. Since they focus their approach on Japanese documents, they found that a word can be formed by several Kanji characters. Locating a query word within a document is then done by analyzing the character density distribution within a document image. The same idea can be applied to other languages when we not only want to spot a single word, but also to perform what is known as passage retrieval.

One of the weak points we find in almost all the methods presented in the existing literature is that most of the approaches take advantage of the layout knowledge. By assuming that the entities of the document images follow a certain spatial structure, they are able to segment words and take them as atomic elements. To the best of our knowledge, there are very few methods which can deal with the document image as a whole without the specific word segmentation step. This is a strong limitation of these approaches since the performance of these methods will always be strongly dependent on the performance of the previously done word segmentation. We believe that rather than using cross-correlation approaches, the use of some indexing structure pointing to the locations where the queried word is likely to appear

would be much more interesting for spotting purposes. As we will see, some symbol spotting methods are based on this idea.

### 2.2.2 *Symbol Spotting*

The main idea of symbol spotting is to describe symbols by a very coarse descriptor to foster the querying speed rather than the recognition rates. Even if symbol spotting is still an emerging topic, several works facing the problem can be found.

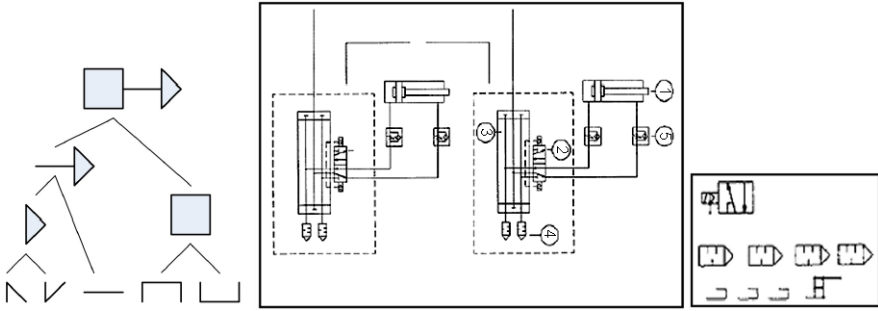
Müller and Rigoll [64] proposed one of the first approaches we can identify as symbol spotting. By using a grid of a fixed size, technical drawing images are partitioned. Each small cell acts then as an input in a two-dimensional HMM trained to identify the locations where a symbol from the model database is likely to be found. The main advantage the system presents is that symbols can be spotted even if they appear in a cluttered environment. However, the fact that the recognizer must be trained with the model symbols entails a loss of flexibility of the presented method.

On the other hand, some techniques work with a previously done ad-hoc rough segmentation, as presented in [83]. In that case, an algorithm of text/graphics separation is applied in order to separate symbols from the text and the background. In [82, 84], the symbols which are linked to a network are segmented by analyzing the junction points of the skeleton image by a loop extraction process. After these ad-hoc segmentations, global numeric shape descriptors are computed at each location and compared against the training set of pixel features extracted from model symbols. Like most of the word spotting methods, in this case, when querying a certain object, a set of segmentations are proposed. A descriptor is computed sequentially for each sub-image and a distance metric decides whether it is the searched item or not. The one-to-one matching is a clear limitation of such approaches which will not be a feasible solution to adopt when facing large collections. In addition, the ad-hoc segmentations are only useful for a restricted set of documents, which makes the method not scalable to other application domains.

Other techniques, as in [6, 49, 52, 60, 70], rely on a graph based representation of the document images. These methods focus on a structural definition of the graphical symbols. Subgraph isomorphism techniques are then proposed to locate and recognize graphical symbols with a single step. However, these approaches do not seem suitable when facing large collections of data since graph matching schemes are computationally expensive.

Realizing that the computational cost has to be taken into account, several works (see, e.g., [20, 93, 102]) were centered on computing symbol signatures in some regions of interest of the document image. These regions of interest can come from a sliding window or be defined in terms of interest points. Obviously, these methods are quicker than graph matching or sequential search, but they make the assumption that the symbols always fall into a region of interest. In addition, symbol signatures are usually highly affected by noise or occlusions.

Zuwala and Tabbone [103, 104] presented an approach to find symbols in graphical documents which is based on a hierarchical definition of the symbols. They



**Fig. 2.2** Dendrogram representation for spotting symbols in technical drawings (this image is based on Fig. 3.5 appearing in [103])

propose the use of a dendrogram structure to hierarchically decompose a symbol. A symbol is represented by its subparts split at the junction points. These subparts are merged according to a measure of density building the dendrogram structure. Each subpart is described by an off-the-shelf shape descriptor. The dendrogram can be subsequently traversed in order to retrieve the regions of interest of a line drawing where the queried symbol is likely to appear. We can see an example on the use of a dendrogram representation for symbols appearing in technical drawings and the obtained spotting results in Fig. 2.2. In [85], the authors proposed an enhancement of the traversal step, which, by the use of indexing strategies, allowed reducing the retrieval time.

Finally, in some domains, graphical objects can be annotated by text labels. In these cases, the spotting mechanism can manage textual queries to provide graphical results as presented by Lorenz and Monagan in [53]. Najman et al. [65] present a method to locate the legend in technical documents. The text contained in the legend can be posteriorly used to extract graphical areas annotated by these text strings, as presented by Syeda-Mahmood in [81]. In this study, we do not consider textual annotations, and thus the spotting method only manages graphical entities.

We can find a summary of the state-of-the-art symbol spotting approaches in Table 2.1. As in the case of word spotting, our feeling is that indexing mechanisms and voting schemes are very useful when trying to not only recognize a graphical object but also locate and recognize at the same time. Spotting methods which do not use indexing structures may discriminate zones of interest from a document image, but can hardly be transferred to a real focused retrieval application dealing with large collections of document images. Let us focus on the problem of describing graphical symbols in the next section.

## 2.3 Symbol Description

Symbol Recognition is at the heart of many of the Graphics Recognition applications. As pointed out in the state-of-the-art review of Lladós et al. [51], due to the

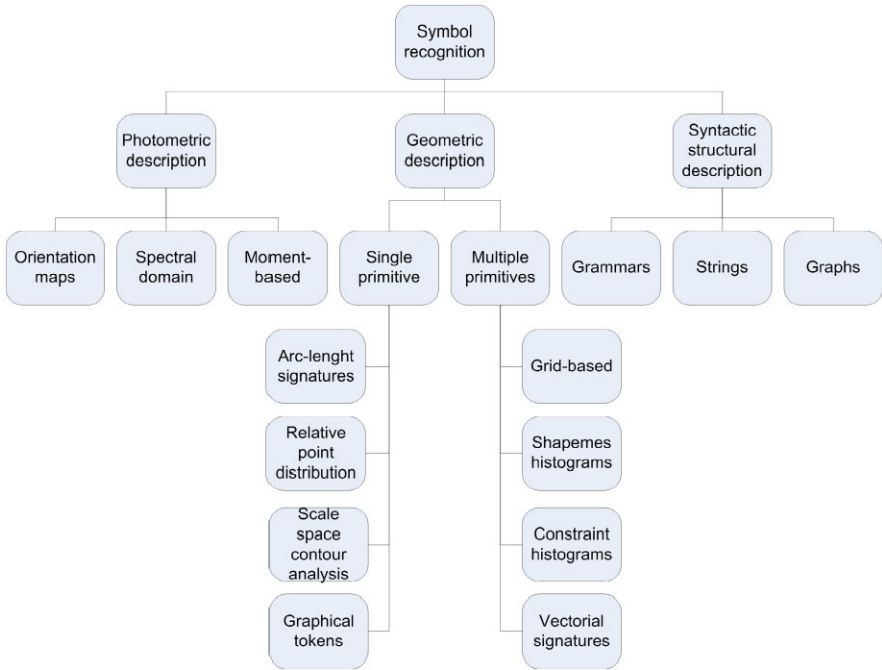
**Table 2.1** State-of-the-art symbol spotting approaches

Family	Method	Pros.	Cons.
2D HMM	[64]	Segmentation-free	Needs training
Pixel features	[83] [82] [84]	Robust symbol description	Ad-hoc previous segmentation
Graph-based	[60] [49] [6] [52] [70]	Simultaneous symbol segmentation and recognition	Computationally expensive
Symbol signatures	[93] [20] [102]	Compact and simple symbol description	Performance decreases if the symbol could not be perfectly isolated
Hierarchical symbol representation	[104] [103] [85]	Linear matching is avoided by using an indexing technique	Dendrogram structure is strongly dependent on the merging criterion
Textual queries	[53] [81] [65]	More robust since it is easier to recognize characters than symbols	Only applicable when textual information is present

wide range of different graphic documents, each of them containing its particular symbols, it is not easy to find a precise definition of what a symbol is. In the context of graphic-rich documents, symbols can be defined as the graphical entities which are meaningful in a specific domain and which are the minimum constituents that convey the information.

From this definition, we can see that there is a large variety of entities that can be considered symbols. Symbols can range from simple 2-D binary shapes composed of line segments as in the case of the entities found in engineering or architectural documents to complex sets of gray-level or even color sub-shapes as in the case of trademarks or logos.

This vast and heterogenous nature of symbols provokes that, when facing the problem of describing and recognizing symbols, the proposed methods found in the literature can rely on different primitives and visual cues to describe a symbol depending on the application at hand. In the different reviews of description techniques, each author proposes a different taxonomy to cluster the methods following different criteria. For instance, Mehtre et al. [59] base their classification of shape description techniques on whether the methods describe the shapes from the previously extracted boundary of the objects, or if they are region-based and use all the internal pixels to describe a shape. A more recent review of shape description was proposed by Zhang and Lu in [101]. In that case, the authors add another criterion to cluster the existing methods. Besides the contour-based or region-based nature of the systems, they propose to check if they are structural or global. This sub-class is based on whether the shape is represented as a whole or represented by seg-



**Fig. 2.3** Classification of symbol representation and description techniques

ments/sections. Lladós et al. [51] presented a state-of-the-art on symbol recognition techniques, clustering the existing methods not only by the nature of techniques but also by their intended applications. In this book, we propose to cluster the description techniques into three different categories depending on the visual cues which the different methods aim to encode. In the first category, the *photometric description* of symbols describes the graphic objects in terms of the intensity of its pixels. At the same time, it thus encodes several visual cues as the shape of the object, its color, texture, etc. On the other hand, a *geometric description* of symbols is only centered on the analysis of the shape as a basic visual cue. Finally, the *syntactic and structural description* of symbols aims at representing the structure of a set of geometric primitives by defining relationships among them. Obviously, some methods in the literature are difficult to classify following this taxonomy since they use a combined strategy, or because they may be understood as belonging to different categories at the same time. We can find the whole hierarchy of the classification in the diagram shown in Fig. 2.3.

### 2.3.1 Photometric Description

The main interest of this kind of approaches to describe graphical symbols is that the photometric description encodes several visual cues at the same time. This kind

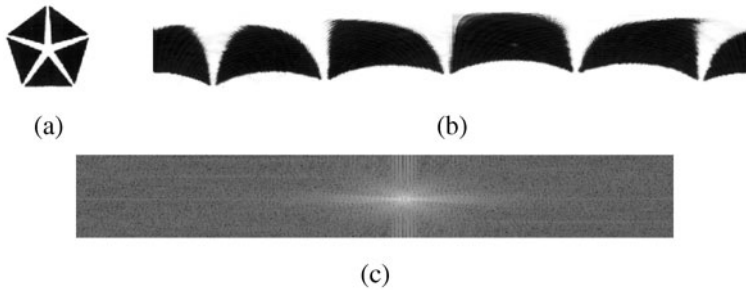
of description is suitable when we face complex symbols that could hardly be described by the shape information only. The problem of logo recognition is one of the application examples where a photometric description is suitable.

Bagdanov et al. [4] present a method focused on the detection of trademarks appearing in real images. In order to describe those symbols, in that work, the authors use the SIFT descriptor to match the trademark models against video frames. The SIFT descriptor, presented by Lowe in [54, 55], basically characterizes the local edge distribution around a given interest point by analyzing the intensity gradients in a patch surrounding the previously extracted key point having a certain scale and orientation. The feature descriptor is computed as a set of orientation histograms on a grid of  $4 \times 4$  neighborhoods. These histograms are computed relative to the key point orientation in order to achieve invariance to rotations. In addition, the magnitude and orientation of the gradients are computed from the Gaussian scale space image closest in scale to the key point's scale. The contribution of each pixel is weighted by the gradient magnitude, and by a Gaussian with a  $\sigma$  value proportional to the scale of the key point. Histograms contain 8 bins each, and each descriptor contains an array of 16 histograms around the key point. This leads to a SIFT feature vector with  $4 \times 4 \times 8 = 128$  elements. The SIFT descriptor has been widely used in Computer Vision for several applications as object recognition or robotics related problems such as SLAM (simultaneous localization and mapping). It could be very useful to describe complex graphic symbols as logos, but it loses effectiveness when representing simpler symbol designs.

From another point of view, there is a family of photometric descriptors which base the symbol representation in the spectral domain. The analysis of images in the spectral domain overcomes the problem of noise sensitivity. Within this family, we can find some works focused on the application of such descriptors for symbol recognition. For instance, the generic Fourier descriptor (GFD) presented by Zhang and Lu in [100] is used to recognize a set of trademarks. In this work, the raster images of logos (as the one shown in Fig. 2.4) are transformed from the Cartesian to the polar space and then a two-dimensional Fourier transform is applied to obtain the symbol description. Another example of spectral descriptors is the Fourier–Mellin transform. After a polar representation of the image, the angular parameter is expressed by the coefficients of the Fourier transform, whereas the Mellin transform is applied to the radial parameter. In [1], Adam et al. present a method allowing the classification of multi-oriented and multi-scaled characters appearing in technical documents. They base their set of invariants on the Fourier–Mellin transform, and are able to deal even with connected characters without a prior segmentation step.

Another family of photometric descriptors are those based on moments. As presented in Teh and Chin's review [89], moments have been utilized as pattern features in a number of applications to achieve invariant recognition of two-dimensional image patterns. Let us briefly review some of the moment-based descriptors which can be applied to the description of graphical symbols. Hu [33] first introduced a set of moment invariants by using nonlinear combinations of geometric moments. Those invariants have the properties of being invariant under image translation, scaling,





**Fig. 2.4** Example of a generic Fourier descriptor. (a) An original logo image; (b) polar-raster sampled image plotted in Cartesian space; (c) Fourier spectra of (b); (this image is based on Figs. 4 and 5 appearing in [100])

and rotation. All these properties make Hu's invariants a suitable symbol descriptor. For instance, in [17], Cheng et al. presented a symbol recognition system focused on the recognition of previously segmented electrical symbols. After a normalization, a symbol is described by a feature vector representing the six geometric moment invariants computed with respect to the symbol centroid. From the theory of orthogonal polynomials, Zernike moments have been introduced in [88]. By projecting the symbol image to a vectorial space defined by a set of orthogonal polynomials named Zernike polynomials, the Zernike moments are obtained. Independent moment invariants are then easily constructed of an arbitrarily high order. As an application example, Khotanzad and Hong [39] use the Zernike moments to describe a small set of upper case letters affected by several transformations and distortions. They show that Zernike features compare favorably with Hu's geometric moment invariants. In addition, the Zernike moments have the ability to reconstruct the graphical symbol from its description, which in some applications may result useful. Finally, another kind of orthogonal moments are the Legendre moments which make use of the Legendre polynomials. In [18], a descriptor based on an enhancement of the Legendre moments is presented to recognize Chinese characters ongoing several transformations. Usually, moment invariants are good descriptors since they are easy to compute, and besides describing the intensity of the pixels, they also have a relation with geometric properties as the center of gravity or axes of inertia.

We can find a summary of the state-of-the-art photometric symbol descriptors in Table 2.2. In the next section, let us focus on the geometric descriptors.

### 2.3.2 Geometric Description

Geometric description techniques are primitive-based methods encoding basically the shape as the most important visual cue to describe graphical symbols. Symbols are broken down into lower level graphical primitives and are then described

**Table 2.2** State-of-the-art photometric symbol descriptors

Name	Application to symbol description	Notes
SIFT	[4]	Invariance to affine transforms and illumination changes; set of orientation histograms computed over previously extracted key points
GFD	[100]	Applied to segmented symbols; 2-D Fourier transform of the polar image
Fourier–Mellin	[1]	Can be applied to non-segmented symbols
Hu’s invariants	[17]	Nonlinear combination of the lower order moments; invariant to similarity transforms but not too much discriminative
Zernike	[39]	Orthogonal moments; allow a reconstruction of the shape; robust to noise
Legendre	[18]	Orthogonal moments; allow a reconstruction of the shape; less robust than Zernike’s

in terms of these primitives. The usual extracted primitives from the symbols are: contours, closed regions (loops), connected components, skeletons, etc. Within this family, we will differentiate between methods which are only able to describe one primitive, or methods which can be used to describe the whole symbol in terms of all these composing primitives.

### 2.3.2.1 Single Primitive Description

A great variety of simple shape descriptors coping with geometric characteristics exist in the literature. Those descriptors are very easy to compute but are usually poorly discriminant. They can be used as a first stage of the selection process among the shapes likely to be good candidates. Most of these simple descriptors are computed over a contour primitive, but can also be used to describe the skeleton or a region. Among the whole variety of simple shape descriptors, we can cite a few. The area and the perimeter of the shape under analysis can be used as a coarse filter in applications where invariance to scale is not needed. The diameters of the circles with the same area or perimeter as the considered shape can also be used as simple descriptors, but again the scale invariance is not achieved. Usually, for segmentation purposes, the orthogonal projections of the shape following the  $x$  and  $y$  axes are used to describe whether a shape is present or not. To have invariance to rotation, Feret’s diameters are used. Feret’s diameters are the maximal and minimal orthogonal projections of the shape on a line. In order to achieve invariance to scale, some ratios among simple features are usually used. The eccentricity, also

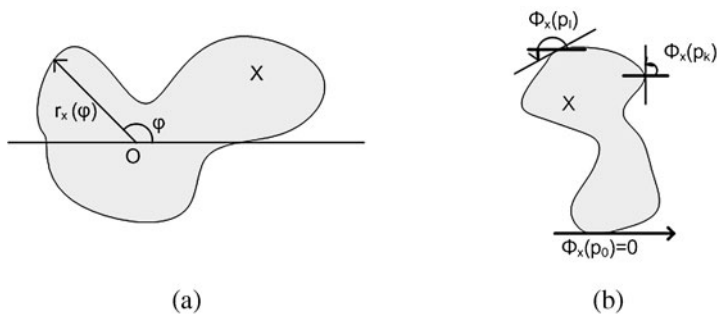
called aspect-ratio or Feret's ratio, characterizes the dimensionality of the shape and is computed as the ratio between the maximum and minimum Feret's diameters. The area-perimeter ratio computed as  $4\pi A(X)/(P(X)^2)$  (where  $A(X)$  and  $P(X)$  are the area and the perimeter of the shape  $X$ , respectively) characterizes deviations of the shape from a circular form. For a disc it is equal to 1, while for all other shapes it is less than 1. The convexity ratio is defined as the ratio between the area of the shape and the area of its convex-hull. It characterizes deviations from convexity.

Obviously, not all of these simple descriptors have the desired invariance to rotation or scale, but most of them can be easily normalized to achieve such invariance. In addition to these simple descriptors, we can find several arc-length-based signatures in the literature. These signatures represent a shape by a one-dimensional function derived from its contour. From the variety of arc-length signatures, we can cite:

- The radius-vector function  $r_x(\varphi)$  which is the distance from a reference point  $O$  in the interior of the shape  $X$  to the contour in the direction of the  $\varphi$ -ray, where  $0 \leq \varphi \leq 2\pi$ .
- The tangent-angle function  $\phi_x(p)$  which characterizes the changes of direction of the points of the contour. The tangent angle at some point is measured relative to the tangent angle at the initial point.

We can find an illustration on how these contour signatures are computed in Fig. 2.5. These signatures are also normalized to achieve invariance to translation and scale. Invariance to rotation is obtained by considering the function as periodic and analyzing a circular permutation of the signature. In addition to the high matching cost, contour signatures are sensitive to noise, and slight changes in the boundary can cause large errors in matching.

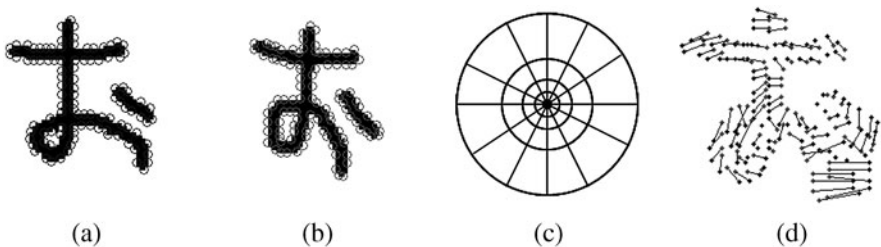
Another family of descriptors are those working at different scales. The scale space representation of a given shape is created by tracking the positions of interest points (protrusions and inflections) in a shape boundary filtered by a Gaussian filter of different width  $\sigma$ . As the  $\sigma$  value increases, little inflections are eliminated from the contour and the shape becomes more and more smooth. The inflection points



**Fig. 2.5** Computation of the arc-length-based signatures. (a) The radius-vector function; (b) the tangent-angle function; (this image is based on Figs. 2.1.3 and 2.1.9 appearing in [40])

of the shape under analysis that remain present in the representation are expected to be significant object characteristics. Mokhtarian et al. [62] present the curvature scale space (CSS) signature. The peaks from the curvature scale space contour map are extracted and used as to match two shapes under analysis. The CSS signature is tolerant to noise and changes in the boundary since it bases its representation at different detail scales. However, since the matching process tries to find the best match between the contour branches of the CSS signature by applying shifts and different scales to achieve invariance to scale and rotation, the matching process proves to be very expensive.

As another example of geometric descriptor for single primitives, we can cite the shape context (SC) descriptor presented by Belongie et al. in [10]. The shape context descriptors allow measuring shape similarity by recovering point correspondences between the two shapes. Given a set of points from a symbol (e.g., interest points extracted from a set of detected edge elements), the shape context captures the relative distribution of points in the plane relative to each point on the shape. Specifically, a histogram using log-polar coordinates which counts the number of points inside each bin is constructed. The descriptor offers a compact representation of the distribution of points relative to each selected point. An example of the shape context descriptor to match shapes can be seen in Fig. 2.6. Translational invariance comes naturally to the shape context. Scale invariance is obtained by normalizing all radial distances by the mean distance between all the point pairs in the shape. In order to provide rotation invariance in shape contexts, angles at each point are measured relative to the direction of the tangent at that point. Shape contexts are empirically demonstrated to be robust to deformations and noise. The shape context descriptor has been tested on different datasets. It has been used to recognize handwritten digits, to retrieve silhouettes by similarity and even to retrieve logos. In [61], Mikolajczyk and Schmid proposed enhancing the shape context descriptor by weighting the point contribution to the histogram with the gradient magnitude and adding orientation information to the histogram besides point locations. This enhancement allows the shape context descriptor to be also classified as a photometric description technique.



**Fig. 2.6** Example of the shape context descriptor for shape matching. (a)–(b) Original shapes to match with sampled edge points; (c) diagram of the log-polar histogram bins used in computing the shape contexts; (d) correspondences found using bipartite matching for the two shapes (a) and (b); (this image is based on Fig. 3 appearing in [10])

Describing graphical symbols by geometric descriptors coping with a single primitive can be very useful when the symbols are non-isolated and other entities than the symbol may appear. These methods may also be very useful when the symbols can be affected by occlusions. As the last example of geometric description for single primitives, we can mention what we call a description based on graphical tokens. Given a primitive (usually a contour or a skeleton of a symbol), it is partitioned into small graphical entities which can be described by very simple attributes. For example, Berretti et al. [12] present a system which partitions a contour into a set of tokens by partitioning the shape at minima of the curvature function. Each token  $\tau_i$  is described through the features  $(m_i, \theta_i)$  representing the curvature of the token and its orientation with respect to a reference system. Stein and Medioni [79] propose to polygonally approximate a shape and then to partition this representation into sets of adjacent segments named super-segments. Those chains of consecutive segments are then represented by several attributes as the lengths, angles, orientation and eccentricity of the token. Nishida [68] presents a simpler, yet effective approach. He proposes applying quantized-directional codes to the approximated contour and characterizing the tokens by a tuple representing the angular span and the direction of the segment. Lorenz and Monagan [53] propose another set of simple tokens to represent graphical entities. To describe regular structures, parallel segments and junctions are taken as tokens and represented by attributes such as length ratios and angles. To cope with irregular structures, chains of adjacent segments are taken as more complex tokens and are encoded by using the first six harmonics of the Fourier approximation of the segments chain. Those simple descriptions of symbols allow coping with distorted symbols and with occlusions. However, as the symbol to be recognized is composed of several tokens, usually, in order to match two different symbols, an algorithm of bipartite graph matching has to be used.

### 2.3.2.2 Description of Several Primitives

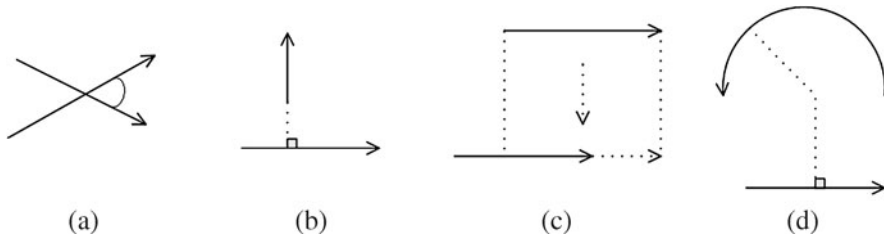
The first example of geometric description of symbols which can handle several primitives at the same time can be the grid-based method proposed by Lu and Sajjanhar in [56]. This descriptor is inspired by the classic photometric descriptor known as zoning [14], but adapted to work with primitives, thus encoding geometric constraints among them. After a primitive extraction step, e.g., of contours or skeletons, the symbol is normalized for rotation and scale. The symbol is scaled into a fixed size rectangle, shifted to the upper left of this rectangle and rotated so the major axis  $F_{\max}(X)$  of the symbol is horizontal. Then, the symbol is mapped on a grid of fixed cell size. Subsequently, the grid is scanned, and a binary value is assigned to the cells depending on whether the number of points in the cell is greater than or less than a predetermined threshold. A unique binary number is obtained as the symbol descriptor. Despite its simplicity, this kind of simple description is very dependent on the normalization step, and may not tolerate well slight distortions.

Inspired by the shape context descriptor described above, Mori et al. [63] present the shapeme histogram descriptor. This approach computes the shape context de-

descriptor for all the interest points extracted from a symbol and uses vector quantization in the space of shape contexts. Vector quantization involves a clustering step of the shape context feature vectors and then represents these feature vectors by the index of the cluster that it belongs to. These clusters are called shapemes. By such means, we obtain a single descriptor for a symbol, no matter how many primitives it has. Each symbol is represented by a collection of shapemes in a histogram. The matching of two symbols is done by finding the nearest neighbors in the space of histograms of shapemes.

Yang [99] presents a symbol descriptor which as the shape context descriptor also captures the relative distribution of points from a symbol. However, the pixel level constraint histogram (PLCH) descriptor encodes the complete symbol no matter how many primitives (skeleton branches in that case) comprise the symbol. The descriptor represents geometric constraints between every pair of points from the skeleton in reference to a third point. At each point from the symbol, we compute a histogram depicting how the other points lie surrounding this point. Length ratios and angles are computed for each pair of points by using one point as a reference. Using an equal bin partition and an accumulation space, two matrices (one for the length information and the other for the angle information) of fixed dimensions are obtained. These matrices are used as the shape descriptor. The distance between two symbols is then defined as the sum of differences between the model and the test matrices. The tests, using the data from the symbol recognition contest held in the Fifth IAPR International Workshop on Graphics Recognition (*GREC 2003*) [92], give good recognition rates under diverse drawbacks such as degradation, distortion, rotation and scaling. As the descriptor focuses on geometric constraints among points, the rotation and scale-invariance are guaranteed. However, this method can only work with segmented symbols and its computational complexity may become very high ( $\mathcal{O}(n^3)$ ), since all the pixel triplets of the skeleton image are considered.

Another family of methods to describe graphical symbols using geometric information are the approaches based on vectorial signatures. This description technique is best suited for applications dealing with symbols arising from line-drawings such as electronic diagrams or architectural floor plans where the primitives are of vectorial nature. The primitives representing the symbols are the segments extracted from a polygonal approximation of the contour or the skeleton of the symbol. The signatures are defined as a set of elementary features, containing intrinsically a discrimination potential. Huet and Hancock [34] present a simple and compact histogram representation which combines geometrical and structural information for line-patterns. The attributes which are taken into account to build the signature are computed between pairs of line segments. These pairwise geometric attributes are the relative orientation between pairs of line segments, length ratios, distances and projections. This representation can be effectively used to index a large database according to shape similarity. Based on the work by Etemadi et al. [21], Dosch and Lladós [20] present a method for symbol discrimination by using vectorial signatures. The method starts with a study of basic relationship between pairs of lines. Several main relations are thus enumerated: collinearity, parallelism and intersections. For each of these relations, some extensions are considered, like overlapping



**Fig. 2.7** Geometric constraints taken as features to build a vectorial signature. (a) Intersection of segments; (b) parallelism; (c) perpendicularity; (d) constraints regarding arcs and circles; (this image is based on Figs. 3, 4, 5, 6, and 7 appearing in [96])

for parallelism or the kind of intersection point. The number and the type of the relations found in a particular zone will form the signature. In [96], Liu et al. present a similar approach. In that case, symbols are also represented by the occurrences of intersections among segments, parallelism and perpendicularities. Each of those features is attributed to certain parameters such as angles, length ratios or directions. We can see an illustration of the considered geometric constraints which built the proposed signature in Fig. 2.7. These approaches present a very compact representation of graphical symbols having enough discriminative power to be used as a basis for spotting systems. However, the main drawback of such signatures is that they may be very sensitive to slight changes in the primitives.

We can find a summary of the state-of-the-art geometric symbol descriptors in Table 2.3. In the next section, let us focus on the syntactic and structural description family.

### 2.3.3 Syntactic and Structural Description

Finally, the syntactic and structural description approaches are focused on the structure of the analyzed symbol. Symbols are first decomposed into basic primitives which may be represented by any description presented above. The syntactic and structural descriptors aim then at defining the relationships among those primitives. Whereas in syntactic description we offer a rule based description of the symbols, in the structural description the recognition of a given symbol is performed by comparing its symbolic representation against a predefined model of the symbol under analysis.

As introduced by Fu in [26], the syntactic approach to pattern recognition provides a capability for describing a large set of complex patterns by using small sets of simple pattern primitives and of grammatical rules. The application of grammars to the problem of symbol description has been widely used over the years since this application is especially well suited to model description through syntactic rules. Terminal elements of the grammars will correspond to the basic primitives comprising a graphical symbol, and the non-terminal elements will describe the production

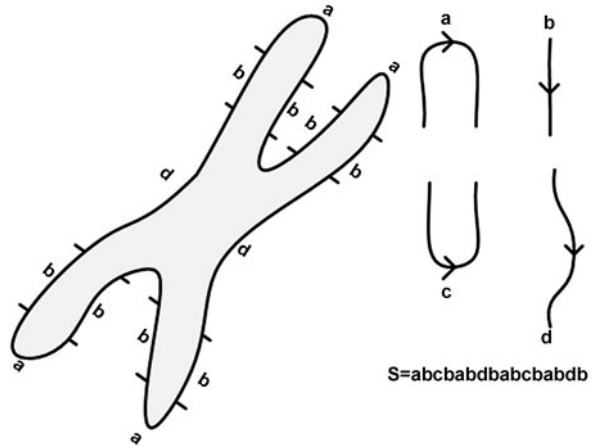
**Table 2.3** State-of-the-art geometric symbol descriptors

Name	Application to symbol description	Primitives	Notes
Simple ratios	[66]	Single	Very simple to compute, low discriminant power
Arc-length signatures	[86]	Single	Sensitive to slight boundary deformations
CSS	[62]	Single	Scale space analysis of a single contour; tracking of the inflection points; robust to boundary noise
Shape context	[10]	Single	Compact representation of distribution of points relative to a reference point; invariant to similarity transforms
Graphical tokens	[79], [53], [12], [68]	Single	Partition of graphical primitives into simpler entities; tokens are described by simple geometric attributes; very useful in case of occlusions
Grid-based	[56]	Multiple	Provide a binary description of the symbol's shape; very simple representation, but dependent on a prior normalization step to achieve invariance to similarity transforms
Shapeme	[63]	Multiple	Vector quantization on the shape contexts of a symbol; obtains a single feature vector for a symbol
PLCH	[99]	Multiple	Represents geometric constraints between every pair of points from the skeleton in reference to a third point; invariant to similarity transforms and robust to noise
Vector signatures	[34], [20], [96]	Multiple	Compact representation of vectorial symbols; invariant to similarity transforms, but very sensitive to noise at the vector level

rules. Syntactic analyzers are built from these grammars in order to group the basic primitives following the production rules and in order to finally recognize graphical symbols. In the literature, we can find many kinds of grammars from the linear ones as the PDL-grammars presented in [77] or the adjacency grammars used by Mas et al. in [58] to more complex grammatical structures as the graph grammars presented by Bunke in [15]. However, the syntactic approaches present the problem that the rule based description schemes are very affected by noisy data. Since the recognition of the symbols is done in terms of the rules of composition of primitives,



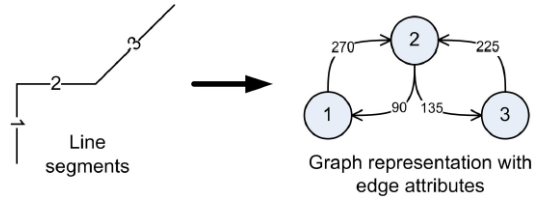
**Fig. 2.8** String representation of closed contours (this image is based on Fig. A.1 appearing in [26])



slight perturbations on the terminal elements may provoke the production rules that cannot be applied, and then the symbol cannot be recognized.

As the first example of structural descriptors, we focus on the string representation of symbols. Symbols are represented by an ordered set of primitives which are encoded as a one-dimensional string. These descriptors codify in which order we expect to find the primitives that comprise a given symbol. For instance, Fu [26] proposed to represent chromosome shapes by a chain of boundary segments forming codewords (see Fig. 2.8). In this kind of approaches, the similarity measure between two string representations of a symbol will be computed by using the string edit operations proposed by Wagner and Fischer in [94]. Tsay and Tsai [91] and Wolfson [98] use the string edit operations applied to the recognition of polygons. Another commonly used method for transforming shapes into one-dimensional strings is the use of the chain codes presented by Freeman in [25]. In that case, shapes are described by a sequence of unit-size line segments with a given set of orientations.

The most common structural representation of symbols is the attribute relational graph (ARG). Graphical primitives are extracted from the symbols and a graph is built representing the structural relationships among those primitives. Messmer and Bunke [60] proposed a symbol recognition framework based on an ARG representation. The primitives taken into account are the line segments that arise from a polygonal approximation of an engineering drawing. An ARG is constructed by taking the segments as the nodes of the graph, and the edges represent that two segments are adjacent. Both nodes and edges have associated attributes. The length of the segment is stored in the nodes, whereas the angle between two segments is stored in the corresponding edge. We can see an example of such graphs in Fig. 2.9. As a second example, Lladós et al. [49] present a different approach of using an ARG. In this case, the authors use higher level primitives than segments. Closed regions are identified from the symbol prototype and are used as the nodes of the graph. The nodes of two adjacent regions of the symbol are linked through an edge of the graph. The nodes of the region graph are attributed by the string representation of the boundary of the region. The edges are attributed by the shared string of the



**Fig. 2.9** Attribute relational graph for symbol description. Nodes represent line segments and are attributed by their length, while edges represent an adjacency relationship and are attributed by the formed angle between the two segments (this image is based on Fig. 3 appearing in [60])

two adjacent regions. Length and orientation attributes are also added to this graph representation. The use of attributed graphs for representing symbols has the main advantage that we can have a very complete description of the symbols. Primitives can be described by photometric or geometric descriptors and these descriptions can be the attributes of the nodes. In addition, the relationships among those primitives are represented by the edges, codifying structural information. In the general case of the graph-based symbol description, the recognition of the symbols has to be done by the use of a sub-graph isomorphism. For each symbol, a prototype of its ideal shape is build as an attributed graph. An input symbol is recognized by means of the matching between the representation of the symbol and the symbol prototype. The main drawback of such powerful representation is that the sub-graph isomorphism algorithms are extremely expensive with respect to computation time since they are tackling an NP-complete problem as stated in [28].

In order to avoid the complex step of matching two graph representations, several approaches can be found. For instance, Franco et al. [24] use the minimum spanning tree as a simplification of a graph. By connecting all the pixels constituting the object under the constraint to define the shortest path, the shape topology is captured. A template matching algorithm which uses minimum spanning trees as symbol representation is presented.

We can find a summary of the state-of-the-art syntactic and structural symbol description approaches in Table 2.4. In the next section, let us focus on the problem of organizing the descriptors to provide an efficient access to the information.

## 2.4 Descriptors Organization and Access

In the problem of recognizing graphics appearing within document images, the basic paradigm involves a matching step between the features extracted from the model graphical symbol and the features extracted from the document images. To be able to recognize and locate elements in documents, the descriptors should be stored in a data structure and clustered by similarity. Once the user formulates a query in terms of a symbol, we have to retrieve by an efficient mechanism the locations within the document images where similar symbols to the queried one appear. As pointed out by Califano and Mohan in [16], since all feature combinations may have to be

**Table 2.4** State-of-the-art syntactic and structural symbol descriptors

Name	Application to symbol description	Notes
Grammars	[77], [58], [15]	Rule-based description; performance highly affected by noise in primitives
Strings	[25], [98]	One-dimensional representation of symbols; structural information is the order followed by the primitives; similarity measure defined in terms of edit operations
Graph-based	[60], [49], [24]	Prototype-based description; very powerful tool to describe symbols; extremely expensive with respect to computation time

explored, brute-force matching is equivalent to an exponential search. In focused retrieval applications with large databases, these costs may become unaffordable. The choice of a data structure providing efficient access to the descriptors is crucial to the final performance of the system. In this section, let us briefly review the approaches that can be found in the literature.

### 2.4.1 Sequential Access

First, we can find a family of spotting methods which work with a sequential access to the symbol descriptors, i.e., the descriptors are stored in a list or a similar sequential data structure. In those methods, regions of interest where the symbols are likely to be found are extracted by some means. A symbol descriptor for each of these regions of interest is computed afterwards. When the user wants to retrieve the zones of the image collection having similar description as the queried symbol, the one-to-one matching has to be computed in a sequential way. The complexity of searching similar descriptors by using data structures with sequential access is  $\mathcal{O}(N)$ , with  $N$  being the number of segmented regions of interest for the entire collection. Even if the use of sequential structures has several important drawbacks, it is the most used approach in the literature dealing with symbol spotting.

As application examples we can mention the work by Tabbone et al. [83] where an algorithm of text/graphics separation is applied in order to separate symbols from the text and the background of technical documents. Each connected component is represented by a photometric descriptor computed over the area defined by the bounding box of the connected component. Subsequently, each descriptor is sequentially compared against all the model descriptors and if the distance between two descriptors is small enough, the interest region is labeled as containing a certain symbol. Such approaches are obviously very dependent on the segmentation phase.

To avoid the dependence on a prior segmentation method, some approaches as [64] or [20] use a grid partition of the document or a sliding window approach to compute the descriptors all over the documents. As in the previous method, each

descriptor arising from a window is sequentially compared against all the model descriptors. If the descriptor matches one of the model's description, the window is labeled as containing a certain symbol. In those cases, the number of descriptors to compute and the number of distances among descriptors are dramatically increased.

The spotting methodologies which use a sequential access to the descriptors present several drawbacks. On the one hand, the access to the descriptors is not efficient, leading to an exponential search when all the combinations of descriptors have to be tested. On the other hand, they are hardly scalable to a large number of documents to consider or a larger number of symbol models.

### 2.4.2 Hierarchical Organization

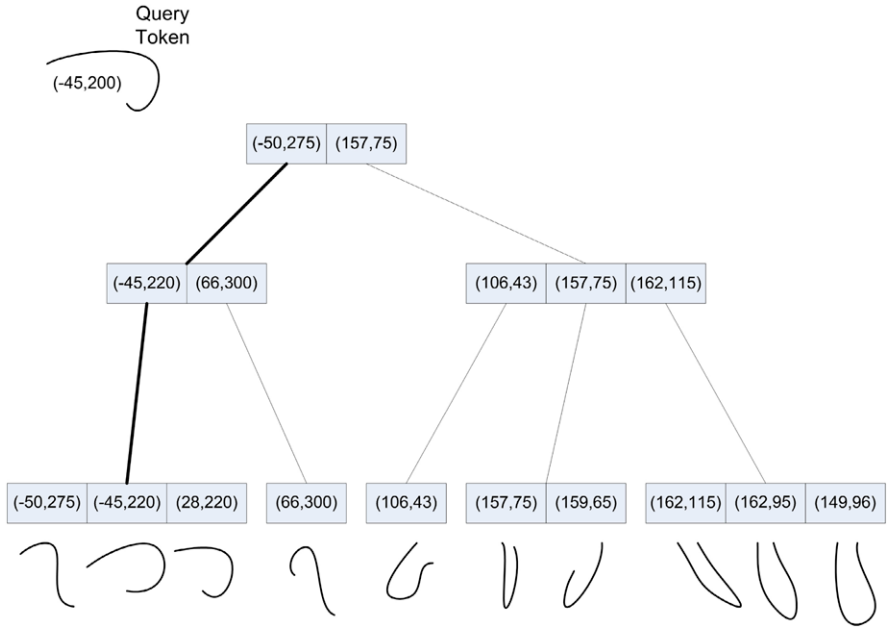
In order to provide a more efficient search by similarity in the description space, we can find a number of methods which work with a hierarchical representation of descriptors. These methods use data structures such as trees, dendrograms, lattices or graphs. Hierarchical data structures are based on the principle of recursive decomposition. They are attractive because they are compact and depending on the nature of the data they save space as well as time and also facilitate operations such as search. The search by similarity is done by a traversal of the data structure that usually can be done in logarithmic time with respect of the number of clusters involved in the structure.

Decades of research in the data mining field have resulted in a great variety of hierarchical data structures that are suitable for retrieval by similarity. The interested reader is referred to Gaede and Günther's [27] comprehensive survey on multidimensional access methods. As examples, we can, for instance, cite the  $K-D-B$ -trees [73] which partition the universe and associate disjoint subspaces with tree nodes in the same level. The  $LSD$ -trees [31] guarantee that the produced data structure is in addition a balanced tree, being much more efficient in the traversal step.

As application examples we can mention Lowe's work [54] which uses a  $k$ -D tree structure [11] to organize the instances of the SIFT descriptor. In their work, Berretti et al. [12] divide contour primitives into diverse tokens which are subsequently stored in an  $M$ -tree indexing structure [19]. We can see an example of the obtained  $M$ -tree structure in Fig. 2.10. From another point of view, Punitha and Guru [69] use a  $B$ -tree [7] to represent the spatial organization of previously recognized primitives.

Within the symbol recognition field, we can, for instance, cite the work of Zuwala and Tabbone [104] who use a dendrogram to hierarchically represent the primitives which compound the graphical symbols present in technical documents. The dendrograms are tree structures which are used to illustrate the arrangement of the clusters produced by a clustering algorithm. Following a similar idea, Guillas et al. [29] use a concept lattice to hierarchically organize symbol descriptors. The navigation of the concept lattice is done in a similar way as for a decision tree.

Finally, graphs can also be used to store information and provide some kind of hierarchical organization of data. For example, in [96] graphs representing symbols



**Fig. 2.10** Hierarchical organization of descriptors by using an *M*-tree structure. Traversing the structure allows a nearest neighbor search in logarithmic time (this image is based on Fig. 11 appearing in [12])

are reduced to a spanning tree which is posteriorly traversed to identify symbols within technical documents. Messmer and Bunke [60] propose to build a network of common subgraph patterns. This network is then traversed by applying graph isomorphisms. Ah-Soon and Tombre [2] propose building a graph of geometric constraints which are then used to recognize symbols appearing in line-drawings.

Structures for hierarchical organization of information based on the principle of recursive decomposition can be very useful for the specific application of symbol spotting. Thousands of feature vectors may arise from the documents, and in the querying step a similarity search has to be done in an efficient way. However, trees can grow arbitrarily deep or wide, and usually the efficiency of the traversal step is very dependent on the tree topology. Balancing algorithms can be applied to maintain the structure usability, but they are very costly to apply.

**2.4.3 Prototype-Based Search**

Another kind of techniques conceived to avoid brute-force matching are based on a prototype-based search. Although this kind of approach is not very common in the data mining field, in the particular case of pattern recognition it has been used in several applications to provide efficient access to clusters of patterns by similarity.

In prototype-based search, we are given a set of distorted samples of the same pattern and want to infer a representative model. In this context, the median concept turns out to be very useful. Given a set of similar patterns, a representative of this set having the smallest sum of distances to all the patterns in the set can be computed. If we want to retrieve similar patterns to a certain query, by using a prototype-based search the retrieval by similarity is done efficiently since only the distances between the query pattern and the representative of a cluster of similar patterns have to be computed. Avoiding a brute-force distance computation allows a fast pattern retrieval by similarity.

The computation of the median of a given set is straightforward when the feature vectors used as descriptors are numeric; however, it is more complex to extend this concept to the symbolic domain. In the literature, we can find several approaches to compute the median of a set of symbolic representations. Recently, Ferrer et al. [22, 23] presented a method to compute the median of a set of graphs. Jiang et al. [37] review the possible applications of the median graphs. Obviously, due to the extreme cost of computing the matching between graphs, prototype-based search implementations are very efficient methods to provide access to the information.

#### 2.4.4 Hashing Approaches

Finally, another widely used data structure is the lookup table to access the information immediately without any structure traversal step. For instance, grid files [67] are a bucket method which superposes a  $n$ -dimensional grid on the universe, and a directory (built with the definition of a hash function) associates cells with bucket's indices. When using hashing techniques, the search operations can theoretically reach  $\mathcal{O}(1)$  time with well chosen values and hashes. To perform a search by similarity by using such structures, the hash function must be seen as a clustering function which can assign the same index to similar shapes.

Multidimensional hashing methods partition the space into hypercubes of known size and group all the records contained in the same hypercube into a bucket. To identify the bucket to which a certain query belongs, the index of the query is automatically computed using a hash function (performing one-dimensional partitions) and the resulting bucket is obtained. In the specific case of shape retrieval, given a primitive, a feature vector is computed using one of the presented descriptors. A hash function establishes quantization criteria to apply to each dimension of the feature vector to limit the index parameters to a finite number of discrete values.

Califano and Mohan [16] use a lookup table mechanism to replace the runtime computation of one-to-one matching with a simpler lookup operation. The speed gain can be significant since retrieving a value from this structure is faster than traversing a tree structure, and much faster than a sequential comparison. Stein and Medioni [79] propose a similar approach to retrieve by similarity subparts of a shape. The subparts comprising a shape are encoded by a hash function resulting in the bucket index of the indexing structure. The same hash function is applied in

the querying step, and all the instances of similar primitives stored in the bucket identified by the resulting index are retrieved.

Kumar et al. [44] present a word spotting method based on a locality sensitive hashing indexing structure. This particular indexing structure aims at efficiently performing an approximated nearest neighbor search by computing multiple hashes. Word images are efficiently retrieved from a large database.

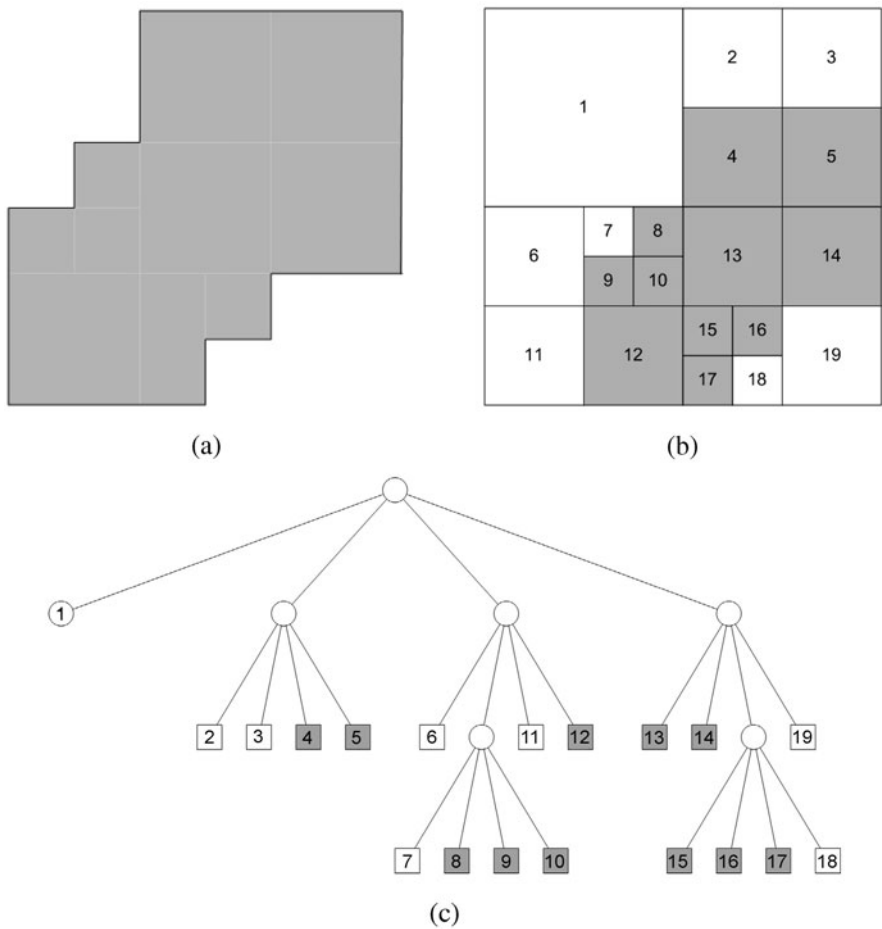
Another classical example of the use of such structures is the geometric hashing method introduced by Lamdan and Wolfson in [46]. The geometric hashing approach is applied to match geometric features against a database of such features. Geometric hashing encodes the model information in a pre-processing step and stores it in a hash table. During the recognition phase, the method accesses the previously constructed hash table, indexing the geometric features extracted from the scene for matching with candidate models. By using the geometric hashing, the search of all models and their features is obviated. The simplest features one can use are the coordinates of points forming a shape. Normalizing the shape scaling, rotating and translating it, taking as basis a reference vector, will give the position of hash table bins to store the shape entry. The major disadvantage of the method is that the same subset has to be chosen for the model image as for the previously acquired images.

In the data mining field, the main drawback of hashing techniques are the collisions. Given two different entries to store in the database, the database system has to guarantee that the hash functions used to index such entries do not assign the same key-index to them. If such thing happens, it would provoke the data to be lost. To overcome this problem, expensive re-hashing algorithms have to be applied once a collision is detected. In the specific case of shape retrieval by similarity, collisions are not a problem but rather the basis of the indexing strategy. Given two similar (but not equal) primitives, they are represented by a compact feature vector. Hopefully, if the two primitives have similar shapes, the two feature vectors will be two nearby points in an  $n$ -dimensional space. The hash function has to guarantee that both points fall into the same bucket to have all the similar primitives stored in a single entry.

### 2.4.5 Spatial Access Methods

So far, we have only focused on the *point access methods*, i.e., indexing structures which can only handle information represented by  $n$ -dimensional points. However, another family of indexing structures exists, designed to manage polygons and high dimensional polyhedra, which is named the *spatial access methods*.

Point access methods cannot be directly applicable to databases containing objects with spatial extension. Spatial data consists of objects made up of points, lines, regions, rectangles, surfaces, volumes, etc. The spatial access methods can be seen as a joint shape description and feature organization. The spatial access methods can handle such data and are finding increasing use in applications in urban planning,



**Fig. 2.11** Quadtree representation of a shape. (a) Sample region; (b) its maximal blocks from the array representation; (c) its quadtree representation (this image is based on Fig. 1 appearing in [75])

geographic information systems (GIS), etc. The interested reader is referred to the book on spatial access methods by Samet [76].

In the spatial access methods class, we can find the quadtree structures, illustrated in Fig. 2.11, which divide a two-dimensional space by recursively partitioning it into four quadrants or regions. The *R*-trees [30] which represents a hierarchy of nested *n*-dimensional intervals store minimum bounding boxes in leaf nodes. An improved structure are the *R*<sup>\*</sup>-trees [8], which try to minimize the overlap between bucket regions, minimize the perimeter of the leaf regions and maximize the storage utilization. *SS*-trees [97] use spheres instead of rectangular regions, *SR*-trees [38] combine both *R*<sup>\*</sup>-trees and *SS*-trees and store the intersection between spheres and rectangles. Finally, *P*-trees [36] manage polygon-shape containers instead of intervals.



Even if such data structures may be very helpful in the context of symbol spotting, our study is focused on the use of symbol descriptors as a basis for data representation. In this book, we will only focus on the use of point access methods to organize feature vectors describing graphic objects.

### ***2.4.6 Curse of Dimensionality***

The curse of dimensionality is a term coined by Bellman [9] to describe the problem caused by the addition of extra dimensions to a space which provokes an exponential increase in volume. This volume increase usually results in a performance degradation. Weber et al. [95] argue that indexing techniques reduce to sequential search for ten or more dimensions. However, in the case of symbol spotting, it is difficult that we suffer from this problem.

The study by Korn et al. [43] showed that the feeling that the nearest neighbor search in high dimensional spaces is hopeless, due to the curse of dimensionality, may be overpessimistic. Real data sets disobey the assumption that the data is uniformly distributed since they typically are skewed and exhibit intrinsic dimensionalities that are much lower than their embedding dimension due to subtle dependencies between attributes.

In addition, for spotting purposes high-dimensional descriptors are not the best suited. Usually, high-dimensional descriptors are more robust to noise and transforms and are more reliable than simpler ones. Obviously, in the case of isolated symbol recognition, it is desirable to have this robustness and accuracy despite the volume explosion. However, in the case of symbol spotting, we are more interested in the efficiency of the retrieval by similarity step than the final accuracy of the recognition task. Usually compact representations are best suited for spotting purposes despite the discriminative power loss. Therefore, for spotting applications low-dimensional descriptors are usually chosen.

## **2.5 Hypotheses Validation**

In the retrieval stage, the result of the traversal of the data structure is a set of primitives similar to the ones which compound the searched symbol. The document locations accumulating several primitives are hypothetic locations where it is likely to find the symbol under a certain pose. These hypotheses have to be validated in a final phase of the retrieval process.

We can find two different approaches to face the hypotheses validation problem. The first one focuses on the feature vectors arising from the description phase and whether these descriptors can or cannot be of the correct symbol. On the other hand, there are several other approaches which focus on geometric and spatial relationships among primitives to finally validate if a zone is likely to contain a certain symbol.

The first family is usually focused on a statistical analysis of the features, whereas the second family is based on a voting strategy and on the accumulation of little evidences to solve the pose estimation problem.

### ***2.5.1 Statistical Validation***

One of the most common approaches to validate whether or not a zone contains a symbol is based on the study of statistical measures with the help of a probabilistic classifier. The features obtained by a symbol descriptor are seen as points in an  $n$ -dimensional space, and the classifiers which are previously trained with a supervised learning step are able to identify the class which the symbols belong to. Within this family of approaches, a lot of different classifiers are used for the problem of classifying symbol instances. One of the most common approaches are the Bayesian classifiers which, for instance, are used in [87] to recognize handwritten symbols. The support vector machines (SVMs) are used in [32] to recognize sketched symbols described by Zernike moments. The modeling of neural networks as classifiers is another option; it was the method applied to musical symbol recognition presented in [80].

From another point of view, there are some other validation methods which are based on the bag-of-words (BoW) model. These approaches use a frequency vector of features to decide whether or not a region can contain a symbol. The principle of the bag-of-words model relies on a document representation as a vector of features where each feature has an assigned frequency. Bag-of-words approaches have been used over the years for text document classification as, for instance, in [3], but the analogy to the bag-of-visual-words can be derived to classify images as in [78]. The approaches based on bag-of-words models have the advantage that the hypotheses validation is done without any spatial information, being very simple to implement and quick to use. In [6], Barbu et al. present a method which applies the bag-of-words model to the symbol recognition problem. However, instead of building the vocabulary from a photometric description of the symbols, they propose a bag-of-graphs model where structural descriptors act as words.

Although high recognition rates can be obtained with statistical validation, the main drawback these approaches present is that they are dependent on a learning stage. In order to have good performance, we need a lot of training samples to feed the classifier. In the particular case of symbol spotting, we cannot use such a priori knowledge nor have an immense sample set of every symbol we want to query. Since spotting approaches are intended to be queried by example, a statistical validation for symbol spotting approaches is usually out of the question.

### ***2.5.2 Voting Strategies and Alignment***

On the other hand, there exist other validation approaches which do not focus on the features arising from the description phase, but rather on testing if the spatial organization of features in a certain location agrees with the expected topology.

Usually, these approaches are focused on some kind of voting strategy as the generalized Hough transform (GHT) presented by Ballard in [5]. The problem of finding a query object inside an image is transformed into the problem of identifying accumulation points in a parameter space. A transformation function maps spatially sparse shapes in the image space to compact regions in the parameter space. The parameter space is divided into buckets. Then, every query descriptor votes in this space according to transformations provided from the matchings with the database descriptors. A high density of votes in a bucket indicates a high probability of detecting the object with its corresponding transformations. For example, in [47], Lamiroy and Gros extended the geometric hashing method with a Hough-like voting strategy to validate the hypotheses in an object recognition application.

Depending on the nature of the symbols to retrieve, the hypotheses validation can be seen as a registration problem. Some approaches validate the coherence of the symbol retrieval using geometric alignment techniques that put in correspondence the original information of the query symbol with the information of the retrieved zones of interest. Some affine transformations can be inferred to align the information of the model object and the retrieved results. Classical techniques use spatial distance between the contours of both images, but other characteristics such as the gradient information can be used as shown in [35]. Other techniques such as B-splines or snakes can also be used for elastic shape matching as in the approach presented by Del Bimbo and Pala in [13]. However, these alignment techniques based on deformable template matching are hardly applicable to symbols where the extracted primitives are not their contour.

## 2.6 Conclusions and Discussion

In order to summarize this state-of-the-art chapter, let us recall the most suitable methods to apply to the symbol spotting problem in each of the three levels.

Regarding the description phase, we should select one of the photometric descriptors if our application has to deal with complex symbols such as logos which may have information in several visual cues such as color, shape, texture, etc. since the descriptors from this family have the ability to encode all this information. For simpler symbol designs, as the ones appearing in line-drawings, geometric descriptors are the most suitable methods to compactly represent the primitives' shape. We should carefully select the appropriate descriptor depending on whether the symbols can be represented in an accurate fashion by a single primitive as the contour or, on the other hand, if we need several primitives to describe a single symbol instance. Finally, syntactic and structural descriptors are a powerful tool in the context of symbol description, but present some drawbacks in the context of symbol spotting. Syntactic approaches are very sensitive to noise and need a definition of the rule set, which is a strong burden for the approach flexibility. Structural techniques should be carefully applied due to the strong time constraints which spotting approaches have to face.

Another factor which is important to take into account is that for spotting purposes it is not essential to look for the descriptor which provides the more accurate description and the best recognition results. Usually, a simpler description able to coarsely discriminate symbols would be a better choice than a complex descriptor with high accuracy.

Once a suitable description technique has been chosen, we have to think how we should organize all the information arising from the document collection to provide an efficient search access. Obviously, the approaches which follow a sequential access of the descriptors are simple to design, but their application to large databases is not realistic. Indexing mechanisms, whether from the hierarchical category or the hashing techniques, should be adopted to access the data. We believe that in the particular case of spotting, hashing techniques are a better choice since we avoid traversal steps and costly balancing algorithms. However, a comparative study should be further described in order to really determine which are the strengths and the weaknesses of both approaches in this application.

We strongly believe that the use of low-dimensional descriptors is highly recommended in spotting applications in order to avoid the curse of dimensionality. As we previously mentioned, the use of simpler descriptors will cause an accuracy loss and an increase of false positives; however, these two phenomena are not a limitation in the case of spotting since high recognition rates are not needed.

Finally, regarding the hypotheses validation step, our feeling is that voting strategies are more recommendable than statistical validation schemes for a spotting application. Voting strategies do not need a learning stage which is an advantage for scalability reasons. In addition, some voting schemes such as the Hough transform or some works inspired by the geometric hashing are formulated in terms of spatial organization of primitives. The use of a geometric descriptor and such voting schemes provides a combined geometrical definition and structural validation.

## References

1. Adam, S., Ogier, J., Cariou, C., Mullot, R., Labiche, J., Gardes, J.: Symbol and character recognition: Application to engineering drawings. *International Journal on Document Analysis and Recognition* **3**(2), 89–101 (2000)
2. Ah-Soon, C., Tombre, K.: Architectural symbol recognition using a network of constraints. *Pattern Recognition Letters* **22**(2), 231–248 (2001)
3. Apté, C., Damerau, F., Weiss, S.: Towards language independent automated learning of text categorization models. In: *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 23–30. ACM, New York (1994)
4. Bagdanov, A., Ballan, L., Bertini, M., Bimbo, A.D.: Trademark matching and retrieval in sports video databases. In: *Proceedings of the International Workshop on Multimedia Information Retrieval*, pp. 79–86. ACM, New York (2007)
5. Ballard, D.: Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition* **13**(2), 111–122 (1981)
6. Barbu, E., Hérroux, P., Adam, S., Trupin, E.: Using bags of symbols for automatic indexing of graphical document image databases. In: *Graphics Recognition. Ten Years Review and*

- Future Perspectives, *Lecture Notes on Computer Science*, vol. 3926, pp. 195–205. Springer, Berlin (2005)
7. Bayer, R., Metzger, J.: On the encipherment of search trees and random access files. *ACM Transactions on Database Systems* **1**(1), 37–52 (1976)
  8. Beckmann, N., Kriegel, H., Schneider, R., Seeger, B.: The  $R^*$ -tree: An efficient and robust access method for points and rectangles. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 322–331. ACM, New York (1990)
  9. Bellman, R.: *Dynamic Programming*. Princeton University Press, Princeton (1957)
  10. Belongie, S., Malik, J., Puzicha, J.: Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**(4), 509–522 (2002)
  11. Bentley, J.: Multidimensional binary search trees used for associative searching. *Communications of the ACM* **18**(9), 509–517 (1975)
  12. Berretti, S., Bimbo, A.D., Pala, P.: Retrieval by shape similarity with perceptual distance and effective indexing. *IEEE Transactions on Multimedia* **2**(4), 225–239 (2000)
  13. Bimbo, A.D., Pala, P.: Visual image retrieval by elastic matching of user sketches. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**(2), 121–132 (1997)
  14. Bokser, M.: Omnidocument technologies. In: *Proceedings of the IEEE*, vol. 80, pp. 1066–1078. IEEE Computer Society, Los Alamitos (1992)
  15. Bunke, H.: Attributed programmed graph grammars and their application to schematic diagram interpretation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **4**(6), 574–582 (1982)
  16. Califano, A., Mohan, R.: Multidimensional indexing for recognizing visual shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **16**(4), 373–392 (1994)
  17. Cheng, T., Khan, J., Liu, H., Yun, D.: A symbol recognition system. In: *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, pp. 918–921. IEEE Computer Society, Los Alamitos (1993)
  18. Chong, C., Raveendran, P., Mukundan, R.: Translation and scale invariants of Legendre moments. *Pattern Recognition* **37**(1), 119–129 (2004)
  19. Ciaccia, P., Patella, M., Zezula, P.:  $M$ -tree: An efficient access method for similarity search in metric spaces. In: *Proceedings of the Twenty-Third International Conference on Very Large Data Bases*, pp. 426–435. Morgan Kaufmann, San Mateo (1997)
  20. Dosch, P., Lladós, J.: Vectorial signatures for symbol discrimination. In: *Graphics Recognition: Recent Advances and Perspectives, Lecture Notes on Computer Science*, vol. 3088, pp. 154–165. Springer, Berlin (2004)
  21. Etemadi, A., Schmidt, J., Matas, G., Illinworth, J., Kittler, J.: Low-level grouping of straight line segments. In: *Proceedings of the British Machine Vision Conference*, pp. 119–126. The British Machine Vision Association and Society for Pattern Recognition (1991)
  22. Ferrer, M., Valveny, E., Serratos, F.: Median graph: A new exact algorithm using a distance based on the maximum common subgraph. *Pattern Recognition Letters* **30**(5), 579–588 (2009)
  23. Ferrer, M., Valveny, E., Serratos, F., Riesen, K., Bunke, H.: Generalized median graph computation by means of graph embedding in vector spaces. *Pattern Recognition* **43**(4), 1642–1655 (2010)
  24. Franco, P., Ogier, J., Loonis, P., Mullet, R.: A topological measure for image object recognition. In: *Graphics Recognition: Recent Advances and Perspectives, Lecture Notes on Computer Science*, vol. 3088, pp. 279–290. Springer, Berlin (2004)
  25. Freeman, H.: On the encoding of arbitrary geometric configurations. *IRE Transactions on Electronic Computers* **2**, 260–268 (1961)
  26. Fu, K.: *Syntactic Methods in Pattern Recognition*. Academic Press, New York (1974)
  27. Gaede, V., Günther, O.: Multidimensional access methods. *ACM Computing Surveys* **30**(2), 170–231 (1998)
  28. Garey, M., Johnson, D.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York (1979)

29. Guillas, S., Bertet, K., Ogier, J.: A generic description of the concept lattices classifier: Application to symbol recognition. In: Graphics Recognition. Ten Years Review and Future Perspectives, *Lecture Notes in Computer Science*, vol. 3926, pp. 47–60. Springer, Berlin (2006)
30. Guttman, A.: *R*-trees: A dynamic index structure for spatial searching. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 47–57. ACM, New York (1984)
31. Henrich, A., Six, H., Widmayer, P.: The LSD-tree: Spatial access to multidimensional point and non point objects. In: Proceedings of the Fifteenth International Conference on Very Large Databases, pp. 45–53. Morgan Kaufmann, San Mateo (1989)
32. Hse, H., Newton, A.: Sketched symbol recognition using Zernike moments. In: Proceedings of the Seventeenth International Conference on Pattern Recognition, pp. 367–370. IEEE Computer Society, Los Alamitos (2004)
33. Hu, M.: Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory* **8**, 179–187 (1962)
34. Huet, B., Hancock, E.: Line pattern retrieval using relational histograms. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **21**(12), 1363–1370 (1999)
35. Huttenlocher, D., Ullman, S.: Object recognition using alignment. In: Proceedings of the First IEEE International Conference on Computer Vision, pp. 102–111. IEEE Computer Society, Los Alamitos (1987)
36. Jagadish, H.: Spatial search with polyhedra. In: Proceedings of the Sixth International Conference on Data Engineering, pp. 311–319. IEEE Computer Society, Los Alamitos (1990)
37. Jiang, X., Munger, A., Bunke, H.: On median graphs: Properties, algorithms, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **23**(10), 1144–1151 (2001)
38. Katayama, N., Satoh, S.: The SR-tree: An indexing structure for high-dimensional nearest neighbor queries. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 369–380. ACM, New York (1997)
39. Khotanzad, A., Hong, Y.: Invariant image recognition by Zernike moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **12**(5), 489–497 (1990)
40. Kindratenko, V.: Development and application of image analysis techniques for identification and classification of microscopic particles. Ph.D. Thesis, University of Antwerp, Belgium (1997)
41. Kise, K., Tsujino, M., Matsumoto, K.: Spotting where to read on pages—retrieval of relevant parts from page images. In: Document Analysis Systems V, *Lecture Notes on Computer Science*, vol. 2423, pp. 388–399. Springer, Berlin (2002)
42. Konidaris, T., Gatos, B., Ntzios, K., Pratikakis, I., Theodoridis, S., Perantonis, S.: Keyword-guided word spotting in historical printed documents using synthetic data and user feedback. *International Journal on Document Analysis and Recognition* **9**(24), 167–177 (2007)
43. Korn, F., Pagel, B., Faloustos, C.: On the “dimensionality curse” and the “self-similarity blessing”. *IEEE Transactions on Knowledge and Data Engineering* **13**(1), 96–111 (2001)
44. Kumar, A., Jawahar, C.V., Manmatha, R.: Efficient search in document image collections. In: Computer Vision—ACCV2007, *Lecture Notes on Computer Science*, vol. 4843, pp. 586–595. Springer, Berlin (2007)
45. Kuo, S., Agazzi, O.: Keyword spotting in poorly printed documents using pseudo 2D hidden Markov models. *IEEE Transactions Pattern Analysis and Machine Intelligence* **16**(8), 842–848 (1994)
46. Lamdan, Y., Wolfson, H.: Geometric hashing: A general and efficient model-based recognition scheme. In: Proceedings of the Second IEEE International Conference on Computer Vision, pp. 238–249. IEEE Computer Society, Los Alamitos (1988)
47. Lamiroy, B., Gros, P.: Rapid object indexing and recognition using enhanced geometric hashing. In: Computer Vision, *Lecture Notes on Computer Science*, vol. 1064, pp. 59–70. Springer, Berlin (1996)
48. Leydier, Y., Lebourgeois, F., Emptoz, H.: Text search for medieval manuscript images. *Pattern Recognition* **40**(12), 3552–3567 (2007)

49. Lladós, J., Martí, E., Villanueva, J.: Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **23**(10), 1137–1143 (2001)
50. 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)
51. Lladós, J., Valveny, E., Sánchez, G., Martí, E.: Symbol recognition: Current advances and perspectives. In: Graphics Recognition Algorithms and Applications, *Lecture Notes on Computer Science*, vol. 2390, pp. 104–127. Springer, Berlin (2002)
52. Locteau, H., Adam, S., Trupin, E., Labiche, J., Héroux, P.: Symbol spotting using full visibility graph representation. In: Proceedings of the Seventh International Workshop on Graphics Recognition (2007)
53. Lorenz, O., Monagan, G.: A retrieval system for graphical documents. In: Proceedings of the Fourth Symposium on Document Analysis and Information Retrieval, pp. 291–300 (1995)
54. Lowe, D.: Object recognition from local scale-invariant features. In: Proceedings of the Seventh IEEE International Conference on Computer Vision, pp. 1150–1157. IEEE Computer Society, Los Alamitos (1999)
55. Lowe, D.: Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* **60**(2), 91–110 (2004)
56. Lu, G., Sajjanhar, A.: Region-based shape representation and similarity measure suitable for content-based image retrieval. *Multimedia Systems* **7**(2), 165–174 (1999)
57. Lu, S., Tan, C.: Retrieval of machine-printed Latin documents through word shape coding. *Pattern Recognition* **41**(5), 1816–1826 (2008)
58. Mas, J., Jorge, J., Sánchez, G., Lladós, J.: Representing and parsing sketched symbols using adjacency grammars and a grid-directed parser. In: Graphics Recognition. Recent Advances and New Opportunities, *Lecture Notes on Computer Science*, vol. 5046, pp. 169–180. Springer, Berlin (2008)
59. Mehtre, B., Kankanhalli, M., Lee, W.: Shape measures for content based image retrieval: A comparison. *Information Processing & Management* **33**(3), 319–337 (1997)
60. Messmer, B., Bunke, H.: Automatic learning and recognition of graphical symbols in engineering drawings. In: Graphics Recognition Methods and Applications, *Lecture Notes on Computer Science*, vol. 1072, pp. 123–134. Springer, Berlin (1996)
61. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **27**(10), 1615–1630 (2005)
62. Mokhtarian, F., Abbasi, S., Kittler, J.: Robust and efficient shape indexing through curvature scale space. In: Proceedings of the British Machine Vision Conference, pp. 53–62. The British Machine Vision Association and Society for Pattern Recognition (1996)
63. Mori, G., Belongie, S., Malik, J.: Shape contexts enable efficient retrieval of similar shapes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 723–730. IEEE Computer Society, Los Alamitos (2001)
64. Müller, S., Rigoll, G.: Engineering drawing database retrieval using statistical pattern spotting techniques. In: Graphics Recognition Recent Advances, *Lecture Notes on Computer Science*, vol. 1941, pp. 246–255. Springer, Berlin (2000)
65. Najman, L., Gibot, O., Barbey, M.: Automatic title block location in technical drawings. In: Proceedings of the Fourth International Workshop on Graphics Recognition, pp. 19–26 (2001)
66. Neumann, J., Samet, H., Soffer, A.: Integration of local and global shape analysis for logo classification. *Pattern Recognition Letters* **23**(12), 1449–1457 (2002)
67. 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)
68. Nishida, H.: Structural feature indexing for retrieval of partially visible shapes. *Pattern Recognition* **35**, 55–67 (2002)
69. Punitha, P., Guru, D.: Symbolic image indexing and retrieval by spatial similarity: An approach based on *B*-tree. *Pattern Recognition* **41**, 2068–2085 (2008)

70. Qureshi, R., Ramel, J., Barret, D., Cardot, H.: Spotting symbols in line drawing images using graph representations. In: Graphics Recognition. Recent Advances and New Opportunities, *Lecture Notes on Computer Science*, vol. 5046, pp. 91–103. Springer, Berlin (2008)
71. Rath, T., Manmatha, R.: Features for word spotting in historical manuscripts. In: Proceedings of the Seventh International Conference on Document Analysis and Recognition, pp. 218–222. IEEE Computer Society, Los Alamitos (2003)
72. Rath, T., Manmatha, R.: Word image matching using dynamic time warping. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 521–527. IEEE Computer Society, Los Alamitos (2003)
73. Robinson, J.: The K-D-B-tree: A search structure for large multidimensional dynamic indexes. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 10–18. ACM, New York (1981)
74. Rodríguez, J.A.: Statistical framework and prior information modeling in handwritten word-spotting. Ph.D. Thesis, Computer Vision Center/Universitat Autònoma de Barcelona, CVC/UAB (2009)
75. Samet, H.: The quadtree and related hierarchical data structures. *ACM Computing Surveys* **16**(1), 187–260 (1984)
76. Samet, H.: The Design and Analysis of Spatial Data Structures. Addison-Wesley, Reading (1990)
77. Shaw, A.: A formal picture description scheme as a basis for picture processing systems. *InfoControl* **14**(1), 9–52 (1969)
78. 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)
79. Stein, F., Medioni, G.: Structural indexing: Efficient 2D object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **14**(12), 1198–1204 (1992)
80. Su, M., Chen, H., Cheng, W.: A neural-network-based approach to optical symbol recognition. *Neural Processing Letters* **15**(2), 117–135 (2002)
81. Syeda-Mahmood, T.: Indexing of technical line drawing databases. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **21**(8), 737–751 (1999)
82. Tabbone, S., Wendling, L.: Recognition of symbols in grey level line-drawings from an adaptation of the Radon transform. In: Proceedings of the Seventeenth International Conference on Pattern Recognition, pp. 570–573. IEEE Computer Society, Los Alamitos (2004)
83. Tabbone, S., Wendling, L., Tombre, K.: Matching of graphical symbols in line-drawing images using angular signature information. *International Journal on Document Analysis and Recognition* **6**(2), 115–125 (2003)
84. Tabbone, S., Wendling, L., Zuwala, D.: A hybrid approach to detect graphical symbols in documents. In: Document Analysis Systems VI, *Lecture Notes on Computer Science*, vol. 3163, pp. 342–353. Springer, Berlin (2004)
85. Tabbone, S., Zuwala, D.: An indexing method for graphical documents. In: Proceedings of the Ninth International Conference on Document Analysis and Recognition, pp. 789–793. IEEE Computer Society, Los Alamitos (2007)
86. Tapia, E., Rojas, R.: Recognition of on-line handwritten mathematical formulas in the E-chalk system. In: Proceedings of the Seventh International Conference on Document Analysis and Recognition, pp. 980–984. IEEE Computer Society, Los Alamitos (2003)
87. Taxt, T., Ólafsdóttir, J., Dæhlenshort, M.: Recognition of handwritten symbols. *Pattern Recognition* **23**(11), 1155–1166 (1990)
88. Teague, M.: Image analysis via the general theory of moments. *Journal of the Optical Society of America* **70**(8), 920–930 (1980)
89. Teh, C., Chin, R.: On image analysis by the methods of moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **10**(4), 496–513 (1988)
90. Terasawa, K., Tanaka, Y.: Slit style HOG feature for document image word spotting. In: Proceedings of the Tenth International Conference on Document Analysis and Recognition, pp. 116–120. IEEE Computer Society, Los Alamitos (2009)



91. Tsay, Y., Tsai, W.: Model-guided attributed string matching by split-and-merge for shape recognition. *International Journal on Pattern Recognition and Artificial Intelligence* **3**(2), 159–179 (1989)
92. Valveny, E., Dosch, P.: Symbol recognition contest: A synthesis. In: *Graphics Recognition, Recent Advances and Perspectives, Lecture Notes on Computer Science*, vol. 3088, pp. 368–385. Springer, Berlin (2004)
93. della Ventura, A., Schettini, R.: Graphic symbol recognition using a signature technique. In: *Proceedings of the Twelfth International Conference on Pattern Recognition*, pp. 533–535. IEEE Computer Society, Los Alamitos (1994)
94. Wagner, R., Fischer, M.: The string-to-string correction problem. *Journal of the Association for Computing Machinery* **21**(1), 168–173 (1974)
95. Weber, R., Schek, H., Blott, S.: A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: *Proceedings of the Twentyfourth International Conference on Very Large Data Bases*, pp. 194–205. Morgan Kaufmann, San Mateo (1998)
96. Wenyin, L., Zhang, W., Yan, L.: An interactive example-driven approach to graphics recognition in engineering drawings. *International Journal on Document Analysis and Recognition* **9**(1), 13–29 (2007)
97. White, D., Jain, R.: Similarity indexing with the SS-tree. In: *Proceedings of the Twelfth International Conference on Data Engineering*, pp. 516–523. IEEE Computer Society, Los Alamitos (1996)
98. Wolfson, H.: On curve matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **12**(5), 483–489 (1990)
99. Yang, S.: Symbol recognition via statistical integration of pixel-level constraint histograms: A new descriptor. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **27**(2), 278–281 (2005)
100. Zhang, D., Lu, G.: Shape-based image retrieval using generic Fourier descriptor. *Signal Processing* **17**, 825–848 (2002)
101. Zhang, D., Lu, G.: Review of shape representation and description techniques. *Pattern Recognition* **37**, 1–19 (2004)
102. Zhang, W., Wenyin, L.: A new vectorial signature for quick symbol indexing, filtering and recognition. In: *Proceedings of the Ninth International Conference on Document Analysis and Recognition*, pp. 536–540. IEEE Computer Society, Los Alamitos (2007)
103. Zuwala, D.: Reconnaissance de symboles sans connaissance a priori. Ph.D. Thesis, Laboratoire Lorrain de Recherche en Informatique et ses Applications, LORIA (2006)
104. Zuwala, D., Tabbone, S.: A method for symbol spotting in graphical documents. In: *Document Analysis Systems VII, Lecture Notes on Computer Science*, vol. 3872, pp. 518–528. Springer, Berlin (2006)

Symbol Spotting in Digital Libraries  
Focused Retrieval over Graphic-rich Document  
Collections

Rusiñol, M.; Lladós, J.

2010, XIV, 180 p., Hardcover

ISBN: 978-1-84996-207-0