An Overview of Symbol Recognition

16

Salvatore Tabbone and Oriol Ramos Terrades

Contents

Abstract

According to the Cambridge Dictionaries Online, a symbol is a sign, shape, or object that is used to represent something else. Symbol recognition is a subfield of general pattern recognition problems that focuses on identifying, detecting,

S. Tabbone (✉)
Université de Lorraine – LORIA, Vandœuvre-lès-Nancy Cedex, France
e-mail: tabbone@loria.fr

O.R. Terrades
Universitat Autònoma de Barcelona – Computer Vision Center, Bellaterra, Spain
e-mail: oriolrt@cvc.uab.es

D. Doermann, K. Tombre (eds.), *Handbook of Document Image Processing and Recognition*, DOI 10.1007/978-0-85729-859-1_17,

and recognizing symbols in technical drawings, maps, or miscellaneous documents such as logos and musical scores. This chapter aims at providing the reader an overview of the different existing ways of describing and recognizing symbols and how the field has evolved to attain a certain degree of maturity.

Keywords

Pattern recognition • Shape descriptors • Structural descriptors • Symbol recognition • Symbol spotting

Introduction

In any symbol recognition process, the following operations are usually performed (not necessarily in the same order): segmentation, feature extraction, invariance to similarities, and comparison. Traditionally, these operations are seen as a two-step process consisting of feature extraction and symbol recognition. Most of the features are extracted on segmented symbols. However, symbols are often embedded in technical documents, connected to other symbols, and associated with text. Moreover, it is usually very difficult to perform symbol recognition by simply assuming that they have been cleanly extracted. Therefore, "symbol spotting" methods have been proposed to localize symbols without the need of a full segmentation step.

Some of the methodologies proposed in the forthcoming sections have been developed to deal with particular problems coming from technical documents. The evolution of symbol recognition has a close link with text recognition because characters can also be considered as symbols. Several methods in symbol recognition have been inspired from character recognition. Specialized techniques for OCR are described in details in ▶Part C (Text Recognition) of this book, and this chapter will focus on symbol recognition only.

This chapter starts with a brief history of pattern recognition methods related to symbol recognition from the 1950s until today. This historical review will be used as a basis for the structure of the forthcoming sections.

History

The first works on pattern recognition appeared in the late 1950s and early 1960s (Fig. 16.1). These studies were modest in terms of objectives and data used in experimentation, but at that time authors of these works felt the potential of applications behind the pattern recognition field. Forty years later, much progress has been made but there is still room for researches.

From today's perspective where everyone has a camera and can transmit images worldwide instantly, one of the main difficulties that researchers had 40 years ago was image acquisition and manipulation. For instance, the Lincoln Laboratory was

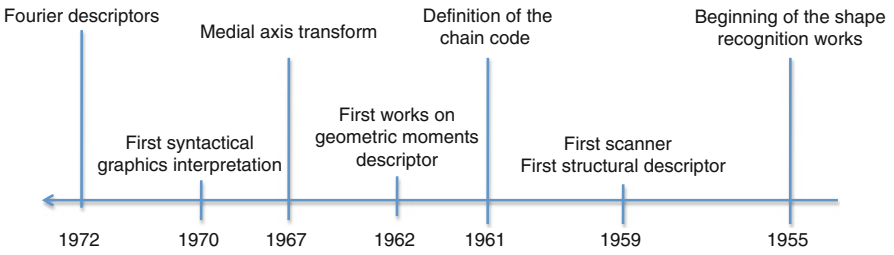

Fig. 16.1 Chronological history related to shape recognition purposes

one of the few laboratories in the world that could read and process digital images through the "Computer Memory Test" (CMT) in 1955. At that time, Dinneen [25] studied whether the averaging and edge operators could be used for simple shape recognition purposes (images of A's, O's, triangles, and squares). In 1959, Bomba [10] proposed the first structural descriptor for character recognition and based on a set of features (straight lines at different orientations, four orientations of T- and L-junctions, and some selected V-junctions). To extract those features, the averaging operator studied by Dinneen was applied to reduce the noisy pattern and to normalize line width. These features were computed from *sliding windows* where local features were computed by aggregating pixels at different orientations. As a result of this process, characters were decomposed (segmented) into different layers and recognition was performed with a decision tree. In 1961, Freeman [34] proposed the *Chain Code*, an encoding method for the representation of arbitrary planar curves, by a sequence of integers ranging from 0 to 7, largely used in many applications and until now.

These previous works by Dinneen, Bomba, and Freeman assumed that patterns (squares and triangles) and characters were isolated inside the image, perfectly oriented, and of the same scale. In 1962, Hu [40] defined a first set of invariant features, namely, geometric moments, to deal with shapes at different positions, scales, and orientations. The theory of moment invariants was based on the algebraic invariant theory developed during the second half of the nineteenth century by Cayley, Sylvester, and Boole. In these works, Hu observed that the more the number of moments were used, the higher the discrimination capacity of his method would be.

All these methods proposed a set of features for describing shapes based on authors intuition about which kind of shape information was relevant for recognition purposes. At that time, a set of features inspired by psychological and psychophysical works [4] was proposed in [9, 82]. Based on those psychophysical knowledge, Blum [9] proposed in 1967 the Medial Axis Transform (MAT). The MAT is defined as the locus of points that are equidistant to the shape contour, and it has largely been used in many other recognition systems, sometimes under the name *Shape Skeleton*. Blum used the MAT to represent pattern's symmetrical

lines and observed some properties that make this transform useful for recognition purposes.

Zahn and Roskies [82] proposed a set of descriptors based on shape contours in 1972. By construction, the amplitude of the Fourier coefficients are invariant to shape's scale, position, and rotation. However, the phase of the Fourier coefficients lacks invariance properties because it depends on the starting point used to parameterize the curve. Thereby, Zahn and Roskies proposed a family of invariant functions of the phase, regardless the starting point.

The aforementioned contributions are typical examples of methods in which authors supposed that symbols had previously been segmented from the document. On the contrary, some works assumed that symbols were connected to other parts of the document. For instance in 1969, Shaw [71] proposed the Picture Description Language (PDL) to describe graphics for the interpretation of electrical circuits. In this case the symbols representing each of the circuit elements were not segmented from the document, and their recognition was performed during the interpretation of the document through a grammar. Shaw represented circuits by directed graphs where each node was a segment of the circuit and the nodes were connected based on adjacency segment relations [35]. This chapter will not go into details about the use of grammar as an interpretation method since it is discussed in ▸Chap. 17 (Analysis and Interpretation of Graphical Documents) and it only focuses on the representation way.

In these early works, from a statistical point of view, it was necessary that symbols were segmented and the number of classes was quite limited (rather printed capital letters), and from structural one, the described relationships were narrowed to adjacent relationships (regardless of inclusion or overlap). It should also be noted that, at that time, different algorithms were not available in common programming languages as debugging tools or image processing libraries like today (e.g., C language was developed in the late 1960). Therefore, the development of any new method had a high cost in time and resources. Later on, with the advent of the IBM 704 and the construction of the first scanner, researchers were able to process images in a more similar manner to what is done today. The majority of the most competitive state-of-the-art methods nowadays rely on concepts and ideas that could be found in these early works. The next section will show how different contributions by adding more processing layers will refine and improve these early works.

Evolution of the Research Field

Symbol recognition has become a fertile field where several methods have been proposed in many directions. That methods introduced in the 1960s and 1970s are still relevant today, despite some improvements. For instance, the Angular Radial Transform (ART) descriptor [53], which is included in the MPEG-7 standard, is an evolution of descriptors based on Zernike moments, which in turn evolved from the Hu moments [40]. Likewise, based on the works of Zahn and Roskies, the univariate [63, 79] and bivariate Fourier descriptors [47, 84] have been widely used

and reused to describe shapes for different applications. The current technology makes that methods that were impossible to apply 50 years ago, due to their computation complexity, become possible. Early works in these years sowed the seeds for the growth of many of today's methods.

For the following, to illustrate the evolution of symbol recognition methods, two axes of changes are selected. The first axe will show how structural methods have evolved into representations of complex data, which has led to increasingly efficient matching algorithms. The second one will show how the problem of deformation of symbols was addressed whatever the used method.

Structural methods focus on describing relations among elementary parts of symbols, namely, *primitives*, that make up symbols. Primitives are, in most of the cases, vectors and arcs, and the relations between them could be their adjacency or inclusion/overlap. Trees and graphs are data structures that best represent this type of information, and the recognition process is to find patterns of symbols in these structures. Therefore, structural approaches focus on developing matching methods that are efficient enough to find substructure within graphs or trees. It is important however to use suitable data structures so that matching algorithms give the expected results. For instance, directed graphs have been used for the representation of electronic circuits [71]. In addition, some attempts to use associative graphs to describe electrical diagrams [22] and shapes [62] will also be mentioned later in this chapter. In 1982, Bunke [11] used attributed graphs for the interpretation of electrical diagrams. This method was then extended as a basis for the proposal of region adjacency graphs (RAG) for similar purposes in 2001 [48].

Despite many improvements, matching algorithms are still so computationally expensive that they cannot be applied to large data sets. For this reason, the interest in kernel embedding methods has increased recently. The goal of such methods is to transform graphs in feature vectors to apply machine learning and statistical pattern recognition techniques. Further details on these methods are discussed in section "Embedding Methods."

These structures are able to represent any kind of technical documents by establishing simple relations among primitives. However, symbols can also be deformed or show local distortions near boundaries, and existing methods at that time, such as the *Chain Code*, were not able to deal with them. In the late 1970s, these difficulties were known and relaxation techniques were used to find approximative correspondences between shapes with, or without, incomplete information [22]. The edit distance is another technique used for the same kind of problem [48, 55].

A slightly different approach to the previous ones are elastic, or deformable, models. These methods consider symbols as objects that can be deformed by adding enough energy to warp one shape into another. Thus, a vectorial field based on curve equations was used by Burr to transform one symbol into another [14]. In the same vein, deformable models based on active contours were created to calculate similarities between handwritten symbols [77].

In the 1980s and 1990s, the use of graphics tablets with CAD software allowed the acquisition of images of diagrams and freehand sketches of architectural

symbols directly in addition to digitized documents. These new devices allowed to explore new symbol recognition applications such as the recognition of mathematical expressions [52]. In addition, online acquisition added dynamic information of symbol drawing, which was used for segmentation and recognition tasks as well. However, the use of this dynamic information introduced new challenges to the symbol recognition field. For instance, the time of response of the system became an important issue to be taken into account, and the order for which the strokes of a symbol were drawn affected the recognition results. In 2010, an adjacency grammar, which can be generated either on- or off-line, was proposed in [54]. Depending on the source of data, a set of primitives were taken and used as terminal elements. The derived adjacency grammar is built from region adjacency graphs [48], but unlike the grammars proposed by Shaw [71] or Bunke [11], inclusion and neighborhood relations between terminal elements and between terminal and nonterminals elements were defined.

These methods are just a few examples of how the problem of nonrigid deformations of the symbols has been tackled. In general, structural descriptors are more flexible than statistical ones because the adopted metrics and matching algorithms allow greater variability between symbols of the same class. For structural descriptors, relationships between primitives are usually independent of position, scale, and rotation since they are defined locally. The matching algorithms are, as with nonrigid transformations, in charge of finding symbols regardless of the position, scale, and rotation. Statistical descriptors work differently from the structural ones to be invariant to similarities. In statistical methods, relationships between primitives are not made explicitly. Images of symbols are seen as surfaces or contours defined in the plane, and features are obtained as the result of applying mathematical transformations. Thus, to be invariant to these deformations, the possible deformations of symbols are taken into account [5, 6] in the comparison stage and not in the statistical descriptor. Generally speaking, statistical descriptors can achieve invariance just before feature extraction by using some kind of normalizations. Ideally, the similarity invariance should be achieved during the process of feature extraction. More details on descriptors invariance will be found in the next section.

Features

This section focuses on feature extraction methods for the construction of symbol descriptors. Several surveys have been proposed in the literature to summarize advances in shape descriptors [61, 85]. However, due to the large number of methods, and since many of them are combinations of the previous ones which could be of different types, it is difficult to establish a clear categorization of symbol recognition methods. Broadly speaking, descriptors are divided into four main groups. On the one hand, descriptors are categorized according to their structural or statistical properties. On the other hand, descriptors are classified depending on whether they are extracted from the regions or the contours. A slightly different terminology had been used by Pavlidis [61] in 1978. Pavlidis divided *algorithms*

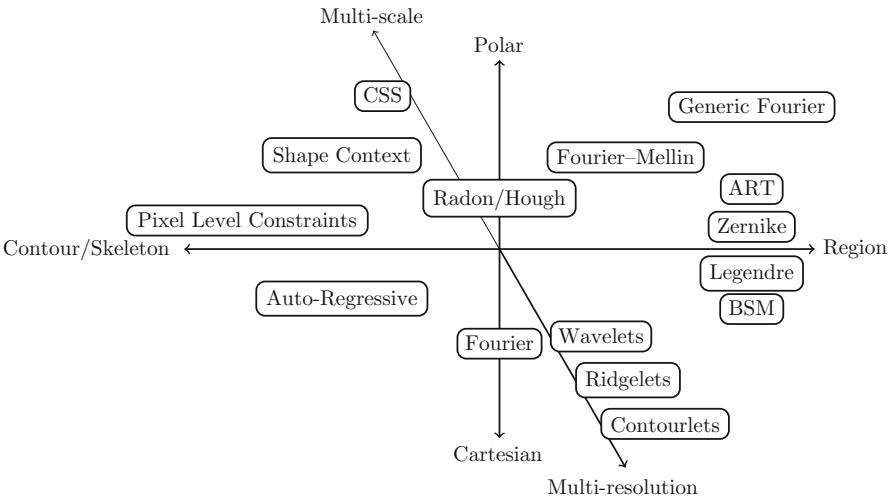

Fig. 16.2 A taxonomy of pixel feature extraction methods

for shape analysis in several binary classes: *external* methods refer to methods defined over the local boundary, whereas *internal* methods are defined over the whole shape. Pavlidis also made another distinction between *scalar* and *domain* transforms. Domain methods transform one image to another, whereas scalar methods compute scalar features from input images. According to these criteria, he defines the following four classes of algorithms: *external scalar* transforms, *internal scalar* transforms, *external space domain* techniques, and *internal space domain* techniques. Scalar features are simple descriptors like area, compactness, rectangularity, and ellipticity.

Nevertheless, the distinction between contours and regions is somehow confused. Indeed, many transforms can be applied to both *contours* and *regions* (e.g., Fourier transform, wavelet transform, Radon transform, regression methods), and many of their properties do not depend on whether they are applied on one-dimensional or two-dimensional data. Moreover, the notion of contour also depends on the topology of a symbol. Full symbols such as logos are different from the ones like wire diagrams. Extracting contours in the latter case means that the skeleton, or the MAT (see ▸Chap. 15 (Graphics Recognition Techniques)), is considered, whereas for full symbols, the contour means the external shape. Moreover, as symbols are often represented in black and white, for noiseless full symbols, describing a symbol by its region or external contour is equivalent. On contrary, for noisy full symbols, region descriptors are more robust than contour ones. Following these considerations, Fig. 16.2 proposes a taxonomy of the most representative pixel feature methods following their representation (polar or Cartesian), their decomposition in a multi-scale/resolution space, and the domain of applicability (contour/skeleton vs. region).

Polar Representation

Shapes are usually expressed in Cartesian coordinates but sometimes descriptors are based on polar coordinates. In the polar representation, the description of a shape is more concise and therefore less sensitive to noise and shape variations. However, the main drawback of this representation is the definition of the coordinate origin. The change of Cartesian to polar, and also polar to Cartesian, is based on the distance of points from the origin. The same shape can be represented in a very different manner depending on the definition of the origin, leading to instability when the shape is noisy. Examples of methods to determine the origin coordinates are the center of gravity, the center of the bounding box, or the center of the minimal enclosing circle. Each of these methods will result in a different polar description of the shape. Another drawback is the effect of shift in a polar description. While a shift in Cartesian coordinates follows a linear map, a shift in polar coordinates follows a sinusoidal function. This fact causes difficulty in getting invariance to shape translation. The change of Cartesian-to-polar coordinates is also a time-consuming process, and methods on pseudo-polar transform have been proposed using concentric squares instead of concentric circles to represent shapes [64] to speed up the change. However, this transform introduces geometric distortions due to the approximation of circles by squares.

Although most of the descriptors using this kind of representation are built from pixel images, some examples of structural descriptors coded in polar coordinates are also found [42]. Hough and Radon transforms describe straight lines in terms of slope angle and distance to the origin which provides a polar description that has been used in structural and statistical descriptors. The $\mathcal{R}$-transform, which is an integral function of the Radon transform in the radial parameter, gives rise to a descriptor called $\mathcal{R}$-signature [38, 74]. Ridgelets descriptor [64] is defined by performing a wavelet transform on the Radon space or combination of several transforms (Radon, Fourier, and wavelets [16]).

Polar Fourier transform simply consists in computing 2D Fourier transform in polar coordinates. The projection in the polar space provides the rotation invariance. An example is the generic Fourier descriptor [85]. The Fourier–Mellin transform is defined by using the Fourier and Mellin transforms, respectively, on the angular and radial parameters [1, 39]. Trace transform generalizes the Radon transform by applying other functionals on a set of lines [43].

ART [53] decomposes a shape in a basis defined by the multiplication of a radial and an angular function. Both functions, angular and radial, are defined by a parameter that determines the ART coefficients. Finally, Zernike moments [17] are defined by the same angular function as ART descriptors, but the radial function is a real-valued polynomial.

Invariance to Similarities

Usually, in invoice documents, trademark logos are not rotated and thus, descriptors do not need to be rotation invariant. On the contrary, in architectural or electronic

documents, symbols can be found in almost any orientation. Such documents could be scanned at different resolutions or, in case of architectural maps, drawn at different scales. In such cases, descriptors have to be invariant to scale. For instance, the descriptor in Fig. 16.3 is invariant to scale and translation but not rotation. In the latter case, the descriptor is shifted horizontally and circularly. Also, when the document is nonplanar such as thick-bound book pages or for documents captured by mobile devices (see ▸Chaps. 2 (Document Creation, Image Acquisition and Document Quality) and ▸4 (Imaging Techniques in Document Analysis Processes)), images are often warped and this deformation can be approximated by an affine transform.

There are different ways of achieving invariance to similarities. The first one is to extract directly from symbols an invariant descriptor. The next section will show a set of different pixel descriptors that are intrinsically invariant to similarities. In other cases, symbol is normalized by estimating its center, scale, and orientation. Having done this, these changes can be reversed to obtain the *original* one. Symbol normalization is sensitive to wrong segmentations, document noise, and partial occlusions or distortions. The estimation of these parameters is done before applying any feature extraction method. When symbols are completely segmented, this normalization is achieved as it is done for polar representation. Thus, the symbol center is defined either as the center of gravity of the symbol or as the center of bounding boxes, convex hull, or minimal enclosing circle. Therefore, invariance to translation and scaling is achieved by shifting the symbol center to the coordinate center and rescaling the symbol to a fixed value. Achieving invariance to symbol rotation by normalization is more critical because sometimes symbols do not have a clear orientation, as it is the case for characters. A typical technique to recover symbol rotation is to use the angle of the main axis defined by the second-order moment, but this technique is sensitive to noise and distortions and not robust when the eccentricity value is near to 1, meaning that the symbol is quite circular. Consequently, when possible, it is better to achieve similarity invariance by means of an invariant feature extraction method since they do not require the estimation of symbol position, scale, and rotation.

The last way of taking into account the invariance is to incorporate the invariance directly into the measure of similarity. For instance, to recover warping deformation, several works use the property of elasticity of the dynamic time warping (DTW) distance [5, 32]. Until now, affine invariance has only been slightly addressed in the symbol recognition community, but with the popularity of mobile devices, the interest on invariance to affine transformation will increase substantially in the future.

Pixel Descriptors

Pixel descriptors are features directly computed from raw images (Fig. 16.4). These type of descriptors usually have been named *statistical* since traditionally they have been used as input of statistical classifiers. Some examples of the most simple pixel features are the Euler number, the number of connected components, the area,

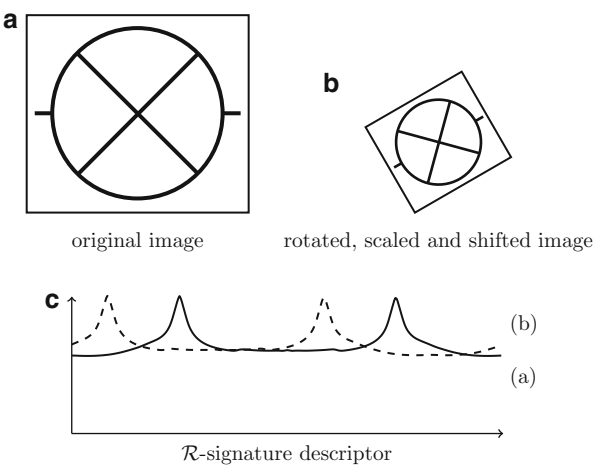

Fig. 16.3 Example of similarities invariance

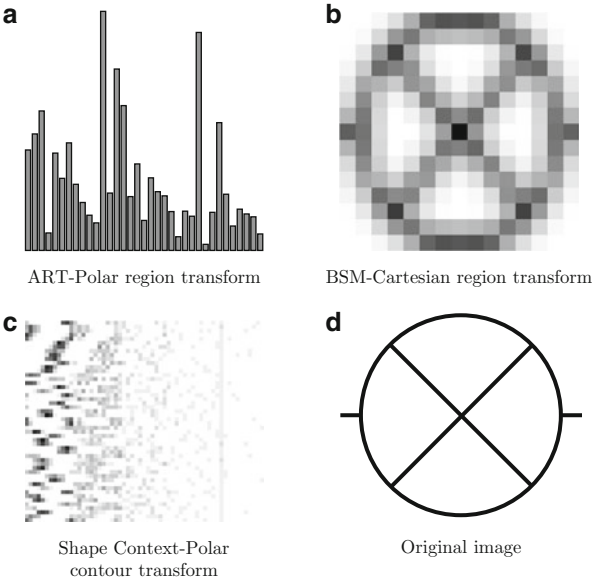

Fig. 16.4 Example of pixel descriptors

the perimeter, the compactness, the rectangularity, and the symbol ellipticity. These scalar features can be enough to solve very simple recognition problems, or they can also be used as attributes in graph descriptors. For more complex tasks, more complex transforms are needed like those reviewed below.

The Fourier transform is probably the most popular extraction method in pattern recognition problems, both in one and two dimensions. There are several

ways of applying the Fourier transform to a planar curve. As explained in the section "History," the first time the Fourier transform was applied to recognition problems was in 1972 [82]. The phase of the Fourier coefficients computed from shape contours was used to compute similarity between shapes [5]. Moreover, the Fourier transform was applied to the $\mathcal{R}$-signature, obtained from the Radon transform of the image, to get invariance to rotation [74]. The strength of Fourier descriptors is that they permit to get a global description of curves without requiring a large number of coefficients. However, they lose their discriminant capability when the similarity between shapes is important because slight shape differences are confused with noise. Bivariate Fourier transform is also used for symbol recognition after applying other transforms like contourlets [15] or Radon transform [16, 39]. The generic Fourier descriptor computed as bivariate Fourier transform in polar coordinates has proved to be more robust to symbol distortions [85].

Geometric moments were introduced in 1962 by Hu [40] to build descriptors invariant to similarity transforms. If, instead of using monomials of order p, x^p, orthogonal polynomials like Zernike and Legendre are used over the unitary disk we obtain the Zernike and Legendre moments. Moreover, the moment order is defined as the polynomial degree. The related mathematical theory proves that we can express any bivariate function defined over the unitary disk as a Legendre or Zernike polynomial of infinite degree. Then, an approximation of the original symbol is obtained by truncating these infinite polynomials. Zernike moments have proved to be more discriminant and robust to noise than geometric and Legendre moments. However, they are more computationally expensive. Some comparative studies have been carried out in this direction in [17]. Some other works concerning Legendre and Zernike moments are found in [75, 81], just to mention a few. Based on the geometric invariant theory [57], several wavelet invariant descriptors have been proposed in [28, 76], but most of the proposed invariant functions require contour-based description of symbols and their extension to regions is not straightforward [33, 45].

Local norm methods compute the norm over a set of features. This type of descriptors was formalized in [19]. Zoning descriptors are the most basic kind of local norm descriptors where an image is divided into cells and the area is computed on each cell. In general, such descriptors are not invariant to similarities unless symbols are centered and resized before computing local norms. These descriptors are useful since shape description is compact and the size of descriptor is reduced. However, the discrimination capability is less, especially for tasks with a lot of different classes of symbols. The blurred shape model (BSM) is a sophisticated zoning descriptor which is robust to symbol deformations [29]. The $\mathcal{R}$-signature is invariant to shift and scale because the signature is computed along the radial parameter of the Radon transform [74].

Auto-regressive (AR) methods consist of computing the parameters of closed curves using regression techniques like least squares. These methods are usually applied to contour curves. Coefficients fitting contour curves are then used to derive descriptors invariant to similarity transforms. Bivariate AR models were proposed in [21, 70] to overcome some shape representation problems instead of univariate

function representing the shape boundary [44]. With a bivariate function, convex and non-convex shapes are treated in the same way. One of the drawbacks of stochastic methods is that the number of coefficients required to describe the shape is high for complex shapes and is usually chosen empirically.

Curvature function is based on the second derivative, describes a planar curve (except its position and orientation), and is invariant to shift and rotation in shape. Changes of curvature in shapes are considered to be dominant features, and they have been the focus of detailed studies since the beginning of the 1980s. It has been shown that this kind of descriptor usually has good performance for general shape description purpose. However, the required computation of the second derivative makes this descriptor sensitive to noise. Curvature function has largely been used as symbol descriptor. The concept of *curvature primal sketch*, which is in fact a Curvature Scale Space, was introduced in [3]. Besides, maxima curvature points were proposed in [7]. The curvature function was computed and local maxima points were extracted to construct a structural descriptor. Finally, the Curvature Scale Space (CSS) descriptor [56] is based on a multi-resolution description of the curvature function and was included in the MPEG-7 standard [53].

Directional methods compute shape gradients in several directions [49]. They are essentially based on the implementation of discrete derivatives, and, as a result, their strengths and weaknesses are strongly influenced by this fact. In general, these descriptors are extremely sensitive to contour distortions and local occlusions of shapes, but they can be easily applied and adapted. Therefore, these descriptors have been used for both machine-printed and handwritten non-Latin characters with different degrees of success since the end of 1970s. In these works, directional information is extracted by means of different masks like Kirsh or Sobel masks. A different approach has been proposed by Kimura et al. in 1997 based on *Chain Codes* for handwritten Japanese character recognition [46] and successfully applied for mathematical symbols recognition [52]. An advantage of this type of descriptors in comparison with the gradient-based approach is the computation time. However, the Chain Code-based descriptor usually shows lower accuracy compared to gradient-based descriptors.

Histogram descriptors are empirical approximations of probability density functions of features. These descriptors are useful because they allow us to reduce the feature space into a small set of bins where feature information is accumulated, like directions or variations in gradient modulus. Their use is not restricted to shape description. For instance, histograms based on color features [73], computed from global, or local, shape features [36, 80], have been proposed in the literature (Fig. 16.4).

Multi-scale/Resolution Decomposition

Some of the previous descriptors are defined in multi-scale or multi-resolution frameworks. The underlying hypothesis is that the most relevant features are

preserved at rough scales. The multi-scale decomposition of a shape basically consists in smoothing by convolving it with a *scale* function, which is most of the time the Gaussian function. The scale function depends on a parameter, σ, which is usually referred to as the *scale parameter*. An inherent drawback of multi-scale decomposition is that the size of the shape is always the same, in spite of a reduction in the number of features. This means that the size of the descriptor is the same, regardless of the scale used. For this reason, descriptors based on this method usually extract features from a pyramidal decomposition of shape from the roughest scale to the finest one to complete the shape description. Contour-based scale space descriptors are obtained by convolving the contours of the shape by a Gaussian kernel (first and second derivative of a Gaussian filter). The Curvature Scale Space [56] and the "primal sketch curvature" [3] are defined in a multi-scale context where local maximal curvatures are extracted, leading to a reduction in the size of the descriptor. Region-based scale space descriptors are obtained by applying Gaussian kernels over the whole image. For instance, the SIFT descriptor [49] is defined by selecting key points at extrema of the difference of Gaussian function in a scale space where at each scale and at each selected point a directional descriptor is computed.

Different from multi-scale decomposition, multi-resolution decomposition is derived from the Multi-Resolution Analysis (MRA) theory [51]. Wavelets were the first MRA methods used as symbol descriptors and applied to shape contours in combination with affine invariants [28]. Moreover, wavelet transform has also been used directly on images to detect horizontal or vertical lines and corners. As symbols are composed of lines oriented in any direction, the performance of bivariate wavelets descriptors in symbol recognition tasks is not high. Therefore, other MRA methods like Gabor wavelets [78], ridgelets [64], and contourlets [15] have been used to overcome this problem of line orientation (Fig. 16.5).

Structural Descriptors

Structural descriptors consider the shape structure in their definition. Shape structures are the logical relations (perpendicularity, adjacency, crossing, and so on) between the *primitive* elements composing the shape. These types of descriptors are usually stored in *graph* or *grammar* structures.

This section is dedicated in the descriptors themselves or, more specifically, in the existing structures to represent the relations between shape entities. The next section "Structural Classification" will consider main techniques for matching algorithms and methods to reduce their high complexity. Structural descriptors for graphics recognition can be broadly divided into two classes: syntactic and prototype-based descriptors.

Syntactic descriptors are determined by a grammar, which is based on the formal language theory introduced by Chomsky in the middle of the 1950s [37], for graphic document interpretation (see ▸Chap. 17 (Analysis and Interpretation of Graphical Documents) for further details). A grammar is a condensed representation of a

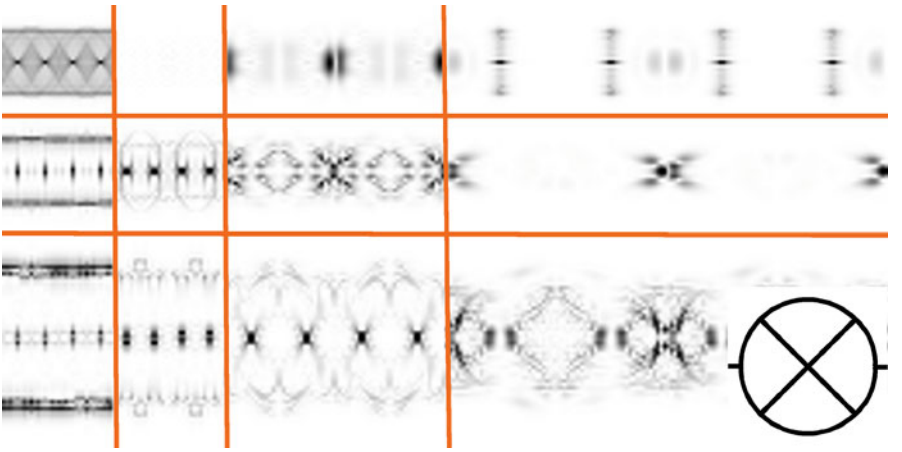

Fig. 16.5 Example of a multi-resolution descriptor based on ridgelet decomposition

large set of prototypes. From a finite set of elements and a set of rules, a large set of prototypes are produced in a similar way as in human language, in which alphabets and language grammar rules allow us to produce words. This kind of representation is suitable when the number of prototype patterns is big, when common substructures among patterns are large, and when the available knowledge about the structure facilitates the grammar inference. When any of these factors is not held, it will be better to use a prototype-based descriptor.

A prototype is a class representative usually represented by using strings and graphs. Using the graph theory (graph and subgraphs isomorphisms), it is possible to compare and to classify shapes that can even be partially occluded. The use of prototypes, instead of all graph descriptors, reduces the complexity of matching algorithms, but the computation of graph prototypes may be computationally expensive also. However, graph prototypes are computed only once during the learning phase, and matching is performed each time at the query level. There are several definitions of prototypes and researches in learning graph prototypes are still a subject of interest. For instance, a recent work [65] has proposed to use a genetic algorithm for learning graph prototypes (generalized median set, generalized discriminative set).

In addition to the above, the use of embedding techniques (see section "Embedding Methods") allows to replace the computation of the graph-edit distance by the computation of a L_p distance in a n-dimensional space, thus reducing the computational complexity. The most representative graph structures used in symbol recognition are *labeled graphs*, *attributed graphs*, and *associative graphs*.

A **labeled graph** is a set of nodes, edges, and labeling functions. It is one of the simplest graph structures still used which can be constructed from a graphic

document after a vectorization process. The formal definition of a labeled graph can be found in any structural pattern recognition textbook with slight differences in notation and names. However, the set of nodes is usually composed of vectors, which play the role of *primitives*, and the set of edges is composed of pairs of nodes representing touching vectors. The definition of the labeling function depends on each particular method. It has been shown before that a grammar was used in 1969 to create the PDL. Later on, labeled graphs were used as a basis to build more sophisticated representations like, for instance, attributed graphs. In 1999, a labeled graph, called "shock graph," was proposed in [72]. Here, the set of nodes is composed of terminal and junction points obtained after applying the MAT to symbols.

An **attributed graph** is a labeled graph (Fig. 16.6) with two more functions that assign a set of attributes to nodes and edges [11]. Graphs without attributes represent the structural information of symbols, while attributes add semantic information for the interpretation of schematic diagrams. In addition to symbol recognition, attributed graphs have also been used in other related applications such as handwriting recognition (see ▸Chaps. 11 (Handprinted Character and Word Recognition) and ▸12 (Continuous Handwritten Script Recognition)) and general structural pattern recognition (see ▸Chap. 15 (Graphics Recognition Techniques)). A particular case of attributed graphs is the RAG where the minimal closed loops (regions) are extracted within their adjacency relations [48].

An associative graph is a different graph-based representation. Each node is an association between local descriptions of a symbol model and partial descriptions of the document being processed. Then, each association is evaluated by a matching function which is defined locally. Therefore, in this representation, two nodes are connected if the matching value is below a preset threshold. With this representation, graphs do not represent simple primitive connections but incorporate information related to the sought primitive symbol models. This representation, which provided a definition of associated graph for trees, was used [22, 62] for comparing shock graphs.

Regardless of the graph structure used for symbol description, structural relations can be grouped taking into account the number of primitives:

- **Unitary** relations are commonly used as node attributes in graphs. For instance, attributed graphs based on the MAT have been proposed in [24, 72]. Another set of structural information that is classically used is the area, perimeter, eccentricity, length, and the number of holes of a region.
- **Binary** relations are the most used set to capture structural information from shapes including for instance, parallelism, angle of intersection, or inclusion between lines [59]. These relations are classically represented by means of edge attributes in graph structures or in a constraint set for adjacency grammars [54].
- **Ternary** relations are less often used than binary ones but may be more important in some applications like text detection. For instance, a ternary descriptor in a Markov random field was proposed in [83] to measure the text alignment using adjacent regions as primitives.

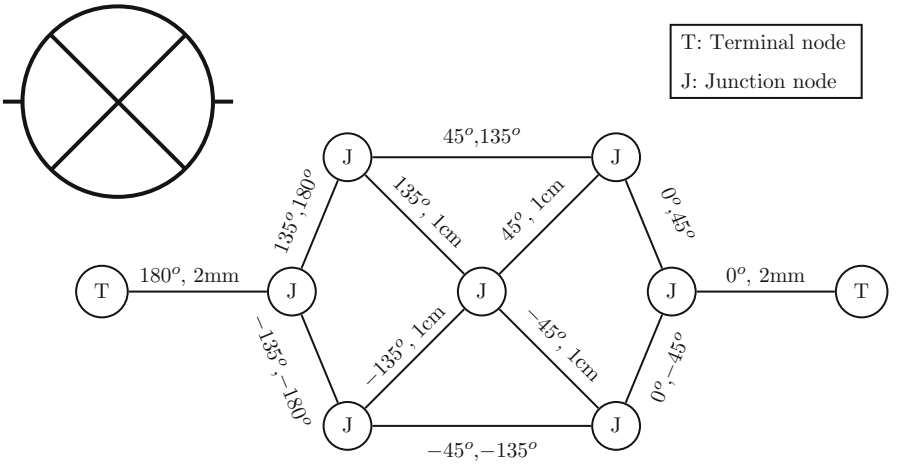

Fig. 16.6 Example of structural descriptor based on an attributed graph representation

Recognition Methods

In 50 years of publications in the field of pattern recognition, many textbooks [8,27] have been proposed. While some of them are dedicated to specific techniques [31], many others have been applied to symbol recognition. Therefore, a thorough review of them is not only unworkable but is also beyond the scope of this chapter. In this perspective, only the most used and well-known methods in the field are discussed in this section.

Pattern recognition process is divided into two stages: feature extraction and classification. As seen, feature extraction involves the construction of symbol descriptors which can be pixel or structural, invariant or not to transformations. Classification indicates the set of methods that will allow to recognize symbols. In general, the majority of symbol recognition methods fit within supervised learning methods (i.e., classification), and therefore, a set of learning symbols is given.

When the training set is lacking or there is no need for supervised learning, the classification step reduces to mere calculation of distance or similarity measure between symbols to be recognized and a set of models. Therefore, the first part of this section will be first dedicated to similarity and distance measures for both structural and statistical methods. The computational cost is one of the motivation for the introduction of embedding and kernel techniques that allow to move from a graph description to a feature one. The second part gives an overview of the respective techniques in structural and statistical classification that have been used to recognize symbols.

Distance and Similarity Measures

The easiest way to recognize a symbol is to compare it to a reference set, namely, models or prototypes, and assign to it the label of the most similar model. If the set of models is not very large, this comparison can be done sequentially using a *similarity* measure. A variety of similarity measures such as distances, correlation, inner product, trigonometric functions, and integral operators, all of which can be applied to symbols recognition, can be found in the literature.

Generally speaking, any functional defined on two elements that returns a scalar value can be interpreted as a similarity measure. Of course, it is preferable that this value has a meaning. An example of a similarity measure used in some symbol recognition methods is the Kullback–Leibler (KL) divergence, which is used as a measure for comparing two probability function densities q and p [86].

A special case of similarity measure is the distance, or metric. A set X with a distance d is called a metric space. From a formal viewpoint, if d is a distance, a set of metric spaces properties can be directly applied. The definition of the distance that one can find in any book of elementary geometry verifies the three well-known properties: positivity, symmetry, and triangle inequality. Examples of the most common distances are:

1. Real vectorial space $\mathbb{R}^n$ with any of the L_p distance: $d(x, y) = \left(\sum_{i=1}^{n} (x_i - y_i)^p\right)^{1/p}$. For $p = 1$, 2, and ∞, it is, respectively, the *Manhattan*, the *Euclidean*, and the *supremum* distance. The sum operator is replaced by the max operator for the supremum distance.
2. The space of real functions $L^p(\mathbb{R})$ composed of p-integrable functions: $d(f, g) = \left(\int_{\mathbb{R}} (f(x) - g(x))^p dx\right)^{1/p}$. For $p = 1$, it is the Banach space and for $p = 2$ the Hilbert space.

In general, for any vectorial space with norm N, a distance is defined as $d(x, y) = N(x - y)$. When the chosen symbol descriptor is a feature vector, it can be useful to consider one of the distances in the previous examples. In such case, the feature vector is embedded into a vectorial space of finite dimension $n < \infty$ where all distances are topologically equivalent. That is, given two distances d_1 and d_2 defined in the metric space X, for all values x and y, two real positive values A and B exist such that

$$A d_1(x, y) \leq d_2(x, y) \leq B d_1(x, y).$$

In practice, this means that given a symbol described by means of feature vectors, regardless of the distances from the Example 1 used, differences in classification rates are insignificant. In other words, the performance of a given feature vector will not change much if the Manhattan distance is used instead of the Euclidean, but the complexity will decrease.

If the L_p distances are the most widely used in finite-dimensional vector spaces, the edit distance, which is in a broad sense a similarity measure, is most often used to compare structural descriptors. It was initially defined to compare strings

and, later, extended to trees and graphs. The distance calculation is obtained by adding the costs of edit operations: insertion, deletion, and substitution needed to transform one string to another. The costs associated with each operation depend on the application, and if they are chosen properly, the edit distance is a true distance which satisfies the three properties required for a distance.

The edit distance is a measure that is robust to errors obtained during the extraction of primitives, but it is computationally expensive to be calculated accurately. To overcome this problem, an algorithm allowing an estimation of this distance by means of a bipartite graph was proposed in [66].

Embedding Methods

The goal of kernel and embedding methods is to apply statistical methods to structural descriptors. The reason here is twofold. On the one hand, they take benefit from structural descriptors, which allow a richer representation of symbols by using graphs and trees than feature vectors. On the other hand, they extend the range of classification methods that can be used in classification problems and reduce the order of complexity of some operations, for example, the calculation of the *generalized median graph* [30].

Embedding methods are categorized formally as implicit or explicit. Explicit methods transform a graph into a feature vector. Thus, we can apply any statistical method: dimension reduction such as PCA by Fisher's discriminant analysis and classifiers such as KNN, boosting, neural network, and SVM. In all cases, the difficulty of embedding methods is to find suitable embedding functions.

Rather than seeking explicit transforms, implicit embedding is based on graph kernels. A kernel is a bivariate function that performs two operations at once. It first embeds structural descriptors into a vectorial space and then performs the dot product in such a space. The advantage of using kernel functions is that the embedding transformation is not needed to be known and it is much easier to define kernel functions than embedding functions. Further details on how to apply this framework for graphs are found in [12, 13, 58].

Structural Classification

Classification methods with structural descriptors consist mostly in finding substructures in global representations of documents. One advantage of these methods, in contrast to statistical ones, is that they do not require a learning phase. However, an expert is needed to set the parameters and the heuristics to have good performance, either in execution time or recognition rate. Basic programming techniques such as dynamic programming [5, 14] or branch-and-bound techniques [48] were applied for searching subgraphs. For an overview of these algorithms, not only for symbol recognition but also for any field of applications in structural pattern recognition, some overviews can be found in [13, 18].

Ideally, matching algorithms look for exact matches between the object to be recognized and a list of known patterns and models. This means that, if data are represented by graphs, both structures must have at least the same number of nodes. On the contrary, in statistical methods distances between two feature vectors are required to have the same dimension. The first component of the first vector has to be compared to the first component of the second vector and so on until last component. In contrast, in structural approaches, there is no canonical order of nodes, and a priori all nodes are compared between them, making sure that all relations between nodes of the first graph are the same as the defined relations between nodes in the second graph. The complexity of these algorithms, in the worst case, grows exponentially with the number of nodes. This, in practice, makes these approaches intractable, especially when graphs have many nodes. Matching algorithms seek to perform these searches in a more intelligent way, with heuristics to reduce the search space. Structural descriptors obtained after a feature extraction process are not free of errors. Thus, two descriptors extracted from different images of the same object can have different representations in the number of nodes and edges. Matching algorithms have to be able to deal with this source of errors in descriptors, and therefore, the graph matching problem in symbol recognition is rather a problem of subgraph matching. Furthermore, a technique to provide error tolerance due to the primitive extraction process is achieved through the edit distance. Other possibilities are relaxation or elastic techniques [22] and active contours [77].

Finally, relations between nodes of trees and graphs can also be represented by adjacency matrices. The values of these matrices depend on the type of trees or graphs, but in any case they are real or even complex numbers. Thus, eigenvalues and eigenvectors of these matrices are computed to compare two different graphs [72]. One advantage is that the order of complexity of the matching algorithm is similar to the complexity of any given distance but graphs should have the same number of nodes and two similar eigenvectors do not necessarily correspond to similar graphs.

Statistical Classification

Statistical methods use information from labeled data to learn classifiers for symbol recognition purposes. The aim of these methods is to learn decision rules to classify with the minimum possible risk of being wrong. In its simplest formulation, the decision rule is defined from the a posteriori probabilities of classes. That is, it maximizes the a posteriori probability (MAP rule) that the class is ω_m given a descriptor x:

$$P(y = \omega_m|x) > P(y = \omega_p|x) \text{ for all } p \neq m. \tag{16.1}$$

This formulation is useful for a classifier, the response of which approximates the a posteriori probabilities of classes. In contrast, for other classifiers it is necessary to introduce the concept of loss function and risk of error:

$$R(y = \omega_p|x) = \sum_q \mathcal{L}(\omega_p, \omega_q) P(\omega_q|x) \tag{16.2}$$

A loss function, $\mathcal{L}$ evaluates the *loss* suffered when a mistake in classification is made. When the loss function is 0/1, MAP rule is recovered, Eq. (16.1). The *risk error* $R(y|x)$ is an operator that indicates the risk of error given a descriptor x, and it is obtained from the loss function and the a posteriori probability of the class. In symbol recognition, the classifier traditionally used is the nearest neighbors. Other classifiers such as the k-th nearest neighbors (KNN), support vector machine (SVM), boosting methods, or genetic algorithms become to be used since benchmarks are available.

Thus, the KNN is the simplest and most used classifier in symbol recognition. The distance between the unknown symbol and all the elements of the data set is computed, and the labels of the closest elements are counted. Then, the label of the majority class is assigned to the unknown symbol. When k increases the ratio obtained by dividing the number of votes with k estimates the posteriori probability [41].

Other state-of-the-art classifiers, like boosting and SVM methods, are two-class classifiers that have been extended to multi-class classifiers to perform symbol recognition. That extension will depend on the strategy used to learn. In classical recognition problems, one possibility is to follow a strategy *1 vs. 1* in which we take $m-1$ classifiers for each of the m classes. The predictions of these $m-1$ classifiers can be combined using any of the techniques described in [2]. Another possibility is to use a *1 vs. all* strategy in which m classifiers, one per class, are trained. In this case, one class is composed of all the elements of the same class (*positive* class), and the other class (*negative* class) is composed of elements taken randomly from the other classes.

SVM are a family of classifiers looking for the optimal hyperplane separating two classes. When this hyperplane does not exist (because classes overlap), the least bad hyperplane that best separates two classes is sought. To find the optimal solution, the problem is formulated as a quadratic optimization problem with boundary constraints. The method of Lagrange multipliers is used to determine the vector w, perpendicular to the separation hyperplane, and a scalar b which is the offset of the hyperplane from the origin. Hence, for classifying a new element x, the sign operator is applied:

$$y = \langle w, x \rangle + b \tag{16.3}$$

To better fit data, instead of looking for planes, surfaces are sought. In other words, data are transformed so that the optimal surface which separates two classes becomes flat. That's the idea behind the *kernel trick* and which has inspired kernel methods for structural descriptors. In this case, the dot product of Eq. (16.3) is replaced by a kernel function k which is in charge of performing the dot product of x and w, transformed by the embedding function ϕ, in another vectorial space, usually of higher dimension than the original data:

$$y = k(w, x) + b = \langle \phi(w), \phi(x) \rangle + b \tag{16.4}$$

Choosing the appropriate kernel function is one of the difficulties to solve. Examples of kernel functions are polynomial of order p, $(\langle x, y\rangle + 1)^p$; radial basis functions, $\exp\left(-\frac{||x-y||^2}{2\sigma^2}\right)$; and the hyperbolic tangent, $\tanh(\kappa\langle x, y\rangle - \delta)$. Once the kernel function is chosen, optimal parameters can be obtained by a cross validation on the training set. However, learning SVM for large data sets (millions of support vectors) is still a challenging problem [26].

Boosting methods seek to build up a good classifier reinforcing the learning of what is called a *weak classifier*. A weak classifier makes slightly better than the object class chosen randomly. That is to say, for a two-class classification problems, it is a classifier with a recognition rate slightly more than 50 %. Boosting methods assigns a weight to each learning sample, which is used to learn a new classifier. This new classifier is also evaluated so that the weights of samples well classified are reduced while the weight of misclassified samples is increased. The resulting classifier is an additive model consisting of the sum of all weak classifiers $\{f_p\}$ that have been trained so far, and c_p is a weight that is obtained from the error of classification for f_p on the training set:

$$y = \sum_p c_p f_p(x) \tag{16.5}$$

Both SVM and boosting methods have proven to be good classifiers in many applications of pattern recognition and are often used and proposed as reference classifiers too. For instance, directional features were used in [52] for SVM classification. A descriptor based on a combination of contourlets and Fourier coefficients, which were trained using an AdaBoost classifier, was proposed in [15]. However, recent experiments using these two kinds of classifiers on zoning-like descriptors show some degree of success whose differences are not statistically significant [29].

Symbol Spotting

All symbol descriptors seen so far assume that symbols are cleanly segmented. However, when symbols are embedded in documents, the well-known paradox rises: to correctly recognize the symbols, one should be able to segment the input data, but to correctly segment them, one need to recognize the symbols! Generally speaking, symbol spotting is a kind of "strategy" used to break down this paradox since one does not try to recognize a symbol in a document as a whole but as a set of primitives.

In a spotting system, the user submits a query he or she wants to retrieve from the document database, and the system retrieves zones in the documents that are likely to contain the query (see Fig. 16.7). The query defined by the user is usually a cropped image of a document or a hand-sketched one belonging to the database. The retrieval stage is done online and, of course, short delay of response is expected.

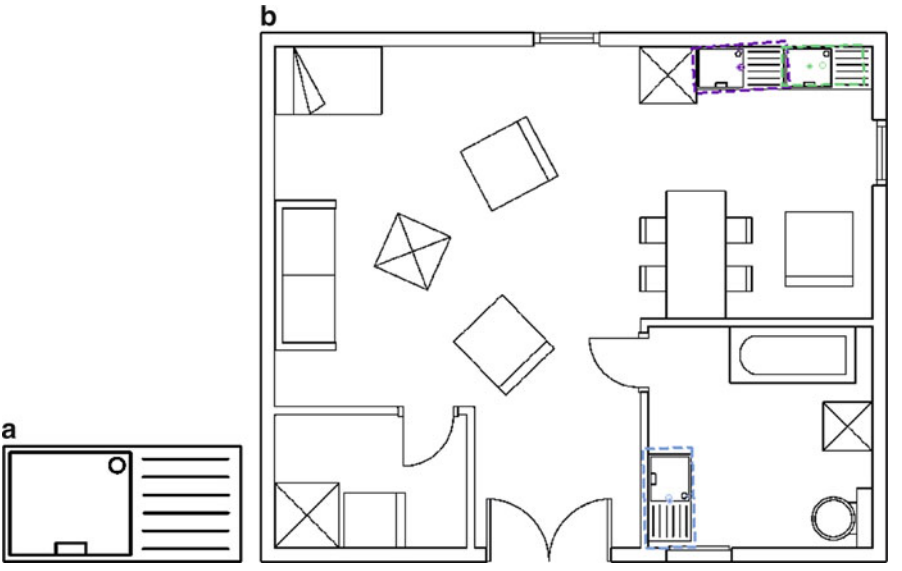

Fig. 16.7 (**a**) The query. (**b**) Retrieved zones in the document that are likely to contain the query

The analogy to the computer vision community is that a spotting system looks like a *content-based image retrieval* (CBIR) application that focuses on subpart retrieval of the image. The main difference here is that technical documents have different photometrical conditions. Documents are usually in black and white and descriptors based on color or texture features cannot be used. Thus, although the underlying strategies are somewhat similar, the features are different since the structural representation is very important for a document.

Symbol spotting is an emerging topic and few works have been proposed so far. As a manuscript on symbol spotting in digital libraries [68] has been published recently, the state of the art will be short in this section. Broadly speaking, symbol spotting methods are decomposed into two main steps. The first one describes the document and is related to the feature extraction method (pixel or structural). This description could be done locally with focus on points or regions of interest, or globally through a structural representation. The second one is the decision step that occurs in the retrieval stage. Since the query is decomposed into a set of primitives like the document, a strategy is needed to organize these primitives to generate a list of location hypotheses in the document that are likely to contain the queried symbol. If the number of symbols and documents is high, a sequential search is not realistic. In this perspective, hashing structures have shown their efficiency in quickly generating hypotheses. In [68], a relational indexing scheme is proposed where a hash table stores the adjacency matrix of proximity graphs and a hash function is designed for feature vectors allowing, in fact, to combine numerical and structural descriptions of symbols. Other structures have been proposed based on

hierarchical organization or inverted file combined with the vectorial model. These structures, however, require more time complexity. As in CBIR applications, the list of hypotheses is ranked from the most to the least likely based on similarity measures or voting strategies.

When the number of documents is small and the same symbols have a low variability in size and orientation, a brute force solution could be applied. In this case the spotting works like a correlation filter. The document is decomposed into regions defined on either the connected components, a grid partition, or a sliding window, and then the similarity (like a correlation function) with the query is measured. For instance, a method to localize mechanical objects in grayscale images was proposed in [50]. The approach does not require any preprocessing step and is based on a pyramidal decomposition. The first step is to search for potential positions of the query symbol in the document as maxima of a normalized correlation surface. These positions are then propagated level by level towards the lowest level of the pyramid in order to precisely locate the corresponding objects. The approach defined in [29] is based on a sliding window strategy and a descriptor (circular blurred shape model) describing the spatial arrangement around a centroid point of the object region in a correlogram structure. These methods based on correlation principle locate accurately positions of a query symbol and are robust enough to be applied on real applications, but the invariance to symbol variations (size and rotation) remains a bottleneck. In this perspective, to avoid the scanning of the whole document, a variant is to focus on interest points, usually corner points, and to use these points to locally describe the document by means of visual words, as it is done in information retrieval where documents are indexed by textual words, or to organize these points spatially to validate hypotheses in order to find a symbol.

Lines remain one of the most used and simplest primitives [60]. An extension to polyline primitives was proposed to take into account circular shapes and segment fragmentation due to the sensitivity to noise of raster-to-vector algorithms. In [68, 69], regions are considered as being circular polylines and symbols are supposed to be defined by closed contours but it is not necessary the case in electrical drawings. Some structural approaches encode primitives in terms of strings [69] or dendrograms [87] representation; however, graphs remain the most popular data structure since it offers a more powerful representation to encode structural relations between different parts of symbols. In some approaches, graph is a means to extract a vectorial signature [68] using a windowing or bucket decomposition of the document for correlation filter. The main limitation of these approaches is that they are not robust to line fragmentation. They could, however, be used to pre-segment the document into zones of interest, which offer a richer description than interest points. Others [48] consider graph as a whole and the spotting is seen as a problem of subgraph matching between the query and document, both represented by graphs. However, these approaches suffer from the classical drawbacks of subgraph isomorphism in pattern recognition such as tolerance to noise and high time complexity, as discussed in section "Structural Classification."

Spotting methods are strongly related to the adopted symbol descriptors as well as the subsequent treatments, since the heart of the methods relies on the

description of document. Usually the used primitives are either pixel or structural. Pixel descriptors are less dependent on the quality of prior segmentations but are less robust to partial occlusions and provide more false responses because they do not take into account the structural configuration of a symbol. In contrast, structural descriptors offer a more powerful description of the document (less false alarms are generated), and they are more flexible to occlusions but they are dependent on the quality of the preprocessing step needed to feed the structural descriptor. Therefore, the compactness and precision of the representation are very important because they have an impact on the performance of the spotting method, on the indexing efficiency of the system and on the delay of response.

Conclusion

The field of symbol recognition has reached its maturity with many descriptors which have been proposed so far. Some descriptors are variants of the previous ones, and several descriptors have reached very high recognition rate even when symbols are noisy, partially occluded, and transformed under similarities. However, the problem of obtaining descriptors robust to severe deformations remains a topic of interest. To achieve a high level of performance, several assumptions on symbols related to their number of models, variability, and segmentation quality are required. Thus, some specific problems still remain for future research.

When the number of symbol models or the complexity of the symbols increases, the confusion between different classes increases. For instance, in an aircraft electrical wiring diagrams, symbols are numerous and are complex entities that associate a graphical representation, a number of connection points, and associated text annotations. The main challenge here is to discriminate symbols not on their global shape but on small details (e.g., the number of connections). Combining outputs of classifiers, descriptors, or selecting features is one of the strategies used to improve the recognition rate. Although it is relatively easy to adapt these strategies to statistical descriptors, the extension to structural ones remains difficult. Recent works on embedding methods [12, 30] have been proposed with promising results.

Performance evaluation campaigns on massive data collection to test the scalability of the proposed approaches are lacking (see ▸Chap. 30 (Tools and Metrics for Document Analysis Systems Evaluation)). However, contrary to a huge number of documents managed, technical documents are rare and few data are available as benchmarks to the image analysis community. Several efforts have been made to create database in graphical contest, but these databases are often small and synthetic, in spite of the effort to generate synthetic documents that look like real ones [23]. The probable reason is the difficulty in getting real documents and the cost to associated the ground truth to real documents since the amount of time to generate it manually would require a huge effort.

With the growing popularity of digital input devices like tablet PCs and smartphones, there is an increasing interest in designing systems that can recognize automatically hand-sketched or camera-captured symbols

(see ▸Chap. 28 (Sketching Interfaces)). These types of symbols are warped and thus have a high variation and distortion. Even if general de-warping document methods can be applied to the particular domain of symbols, new descriptors robust to shape deformations or affine transformations are needed.

The spotting methods are still in their early years, the proposed methods so far are only applied on synthetic documents. Spotting methods need to reach a high level of maturity to consider applications to real documents. Methods combining hash tables and voting strategies seem to be efficient in terms of time and indexing complexity when we face large collections of documents. Moreover, although performance measures [67] have been proposed for symbol spotting methods, these measures are more dedicated to synthetic documents, and their applicability on real documents has not been really verified. In addition, we believe that even if symbol spotting is closely related to symbol recognition, it can be seen as a particular case of document mining and the next step to symbol spotting would be new methods for symbol mining, and there are still many things to do to reach this goal. In this vein, recent works [20] use an original adaptation of a Galois lattice which has shown good performances in the field of data mining. The use of Galois lattices is a means to get a "symbolic representation" from a numeric one in the perspective to narrow the semantic gap.

Cross-References

▸Analysis and Interpretation of Graphical Documents
▸Asian Character Recognition
▸Continuous Handwritten Script Recognition
▸Document Creation, Image Acquisition and Document Quality
▸Graphics Recognition Techniques
▸Imaging Techniques in Document Analysis Processes
▸Language, Script, and Font Recognition
▸Middle Eastern Character Recognition
▸Sketching Interfaces
▸Text Segmentation for Document Recognition
▸Tools and Metrics for Document Analysis Systems Evaluation

References

1. Adam S, Ogier MJ, Cariou C, Mullot R, Labiche J, Gardes J (2000) Symbol and character recognition: application to engineering drawings. Int J Doc Anal Recognit 3:89–101
2. Allwein EL, Schapire RE, Singer Y (2000) Reducing multiclass to binary: a unifying approach for margin classifiers. J Mach Learn Res 1:113–141
3. Asada H, Brady M (1986) The curvature primal sketch. IEEE Trans PAMI 8(1):2–14
4. Attneave F, Arnoult MD (1956) The quantitative study of shape and pattern perception. Psychol Bull 53(6):452–471

5. Bartolini L, Ciaccia P, Patella M (2005) Warp: accurate retrieval of shapes using phase of fourier descriptors and time warping distance. IEEE Trans PAMI 27(1):142–147
6. Belongie S, Malik J, Puzicha J (2002) Shape matching and object recognition using shape contexts. IEEE Trans PAMI 24(24):509–522
7. Berreti S, del Bimbo A, Pala P (2000) Retrieval by shape similarity with perceptual distance and effective indexing. IEEE Trans Multimed 2(4):225–239
8. Bishop C (2006) Pattern recognition and machine learning. Springer, New York
9. Blum H (1967) A transformation for extracting new descriptors of shape. In: Wathen-Dunn W (ed) Models for the perception of speech and visual form. MIT, Cambridge, pp 362–380
10. Bomba JS (1959) Alpha-numeric character recognition using local operations. Papers presented at the eastern joint IRE-AIEE-ACM (Eastern) computer conference IRE, 1–3 Dec 1959. ACM, New York, pp 218–224
11. Bunke H (1982) Attributed programmed graph grammars and their application to schematic diagram interpretation. IEEE Trans PAMI 4:574–582
12. Bunke H, Riesen K (2011) Improving vector space embedding of graphs through feature selection algorithms. Pattern Recognit 44(9):1928–1940. Computer analysis of images and patterns
13. Bunke H, Riesen K (2011) Recent advances in graph-based pattern recognition with applications in document analysis. Pattern Recognit 44(5):1057–1067
14. Burr DJ (1981) Elastic matching of line drawings. IEEE Trans PAMI 3(6):708–713
15. Chen GY, Kégl B (2010) Invariant pattern recognition using contourlets and adaboost. Pattern Recognit 43(3):579–583
16. Chen GY, Bui TD, Krzyzak A (2009) Invariant pattern recognition using radon, dual-tree complex wavelet and fourier transforms. Pattern Recognit 42(9):2013–2019
17. Chong C-W, Raveendran P, Mukundan R (2003) A comparative analysis of algorithms for fast computation of zernike moments. Pattern Recognit 36:731–742
18. Conte D, Foggia P, Sansone C, Vento M (2004) Thirty years of graph matching in pattern recognition. Int J Pattern Recognit Artif Intell 18(3):265–298
19. Costa LdFR, Cesar RM Jr (2009) Shape analysis and classification, 2nd edn. CRC Press, Boca Raton, Florida
20. Coustaty M, Bertet K, Visani M, Ogier J-M (2011) A new adaptive structural signature for symbol recognition by using a galois lattice as a classifier. IEEE Trans Syst Man Cybern 41(4):1136–1148
21. Das M, Paulik MJ, Loh NK (1990) A bivariate autoregressive technique for analysis and classification of planar shapes. IEEE Trans PAMI 12(1):97–103
22. Davis LS (1979) Shape matching using relaxation techniques. IEEE Trans PAMI 1(1):60–72
23. Delalandre M, Valveny E, Pridmore T, Karatzas D (2010) Generation of synthetic documents for performance evaluation of symbol recognition and spotting systems. Int J Doc Anal Recognit 13(3):187–207
24. Di Ruberto C (2004) Recognition of shapes by attributed skeletal graphs. Pattern Recognit 37:21–31
25. Dinneen GP (1955) Programming pattern recognition. In: Proceedings of the western joint computer conference, AFIPS'55 (Western), Los Angeles, 1–3 Mar 1955. ACM, New York, pp 94–100
26. Dong J-X, Krzyzak A, Suen CY (2005) Fast svm training algorithm with decomposition on very large data sets. IEEE Trans PAMI 27(4):603–618
27. Duda RO, Hart PE, Stork DG (2000) Pattern classification. New York
28. El Rube M, Ahmed I, Kamel M (2006) Wavelet approximation-based affine invariant shape representation functions. IEEE Trans PAMI 28(2):323–327
29. Escalera S, Fornés A, Pujol O, Lladós J, Radeva P (2011) Circular blurred shape model for multiclass symbol recognition. IEEE Trans Syst Man Cybern B Cybern 41(2):497–506
30. Ferrer M, Valveny E, Serratosa F, Riesen K, Bunke H (2010) Generalized median graph computation by means of graph embedding in vector spaces. Pattern Recognit 43(4): 1642–1655

31. Flusser J, Zitova B, Suk T (2009) Moments and moment invariants in pattern recognition. Wiley, Chichester
32. Fornés A, Lladós J, Sánchez G, Karatzas D (2010) Rotation invariant hand-drawn symbol recognition based on a dynamic time warping model. Int J Doc Anal Recognit 13:229–241
33. Forsyth D, Mundy JL, Zisseman A, Coelho C, Heller A, Rothwell C (1991) Invariant descriptors for 3d object recognition and pose. IEEE Trans PAMI 13(10):971–991
34. Freeman H (1961) On the encoding of arbitrary geometric configurations. IRE Trans Electron Comput 10(2):260–268
35. Fu K-S, Bhargava BK (1973) Tree systems for syntactic pattern recognition. IEEE Trans Comput 22(12):1087–1099
36. Fujisawa H, Liu C-L (2003) Directional pattern matching for character recognition revisited. In: 7th international conference on document analysis and recognition, Washington, DC. IEEE Computer Society, p 794
37. Groen F, Sanderson A, Schalag F (1985) Symbol recognition in electrical diagrams using probabilistic graph matching. Pattern Recognit Lett 3:343–350
38. Hoang T-V, Tabbone S (2012) The generalization of the R-transform for invariant pattern recognition. Pattern Recognit 45(6):2145–2163
39. Hoang T-V, Tabbone S (2012) Invariant pattern recognition using the rfm descriptor. Pattern Recognit 45(1):271–284
40. Hu M-K (1962) Visual pattern recognition by moments invariants. IRE Trans Inf Theory 8:179–187
41. Jain AK, Duin RP, Mao J (2000) Statistical pattern recognition: a review. IEEE Trans PAMI 22(1):4–37
42. Josep L, Gemma S (2004) Graph Matching Versus Graph Parsing In Graphics Recognition - A Combined Approach. IJPRAI 18(3):455–473
43. Kadyrov A, Petrou M (2001) The trace transform and its applications. IEEE Trans PAMI 23(8):811–828
44. Kashyap R, Chellappa R (1981) Stochastic models for closed boundary analysis: representation and reconstruction. IEEE Trans Inf Theory 27(5):627–637
45. Khalil MI, Bayoumi MM (2001) A dyadic wavelet affine invariant function for 2d shape recognition. IEEE Trans PAMI 23(10):1152–1164
46. Kimura F, Wakabayashi T, Tsuruoka S, Miyake Y (1997) Improvement of handwritten japanese character recognition using weighted direction code histogram. Pattern Recognit 30(8): 1329–1337. Oriental character recognition
47. Lin C-S, Hwang C-L (2010) New forms of shape invariants from elliptic fourier descriptors. Pattern Recognit 20(5):535–545
48. Lladós J, Martí E, Villanueva JJ (2001) Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. IEEE Trans PAMI 23(10):1137–1143
49. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60(2):91–110
50. MacLean W, Tsotsos K (2008) Fast pattern recognition using normalized grey-scale correlation in a pyramid image representation. Mach Vis Appl 19(3):163–179
51. Mallat S (1999) A wavelet tour of signal processing. Academic Press, San Diego, California
52. Malon C, Uchida S, Suzuki M (2008) Mathematical symbol recognition with support vector machines. Pattern Recognit Lett 29(9):1326–1332
53. Manjunath B, Salembier P, Sikora T (2002) Introduction to MPEG-7: multimedia content description interface.
54. Mas J, Lladós J, Sánchez G, Jorge JAP (2010) A syntactic approach based on distortion-tolerant adjacency grammars and a spatial-directed parser to interpret sketched diagrams. Pattern Recognit 43(12):4148–4164
55. Messmer BT, Bunke H (1998) A new algorithm for error-tolerant subgraph isomorphism detection. IEEE Trans PAMI 20(5):493–504
56. Mokhtarian F, Abbasi S (2002) Shape similarity retrieval under affine transforms. Pattern Recognit 35(1):31–41

57. Mundy JL, Zisserman A (1992) Geometric invariance in computer vision. MIT, Cambridge, pp 02142
58. Neuhaus M, Bunke H (2007) Bridging the gap between graph edit distance and kernel machines. World Scientific Publishing, River Edge
59. Park JH, Umm BS (1999) A new approach to similarity retrieval of 2-d graphic objects based on dominant shapes. Pattern Recognit Lett 20(6):591–616
60. Park GB, Lee MK, Lee US, Lee HJ (2003) Recognition of partially occluded objects using probabilistic arg (attributed relational graph)-based matching. Comput Vis Image Underst 90(3):217–241
61. Pavlidis T (1978) Survey: a review of algorithms for shape analysis. Comput Graph Image Process 7(7):243–258
62. Pelillo M, Siddiqi K, Zucker SW (1999) Matching hierarchical structures using association graphs. IEEE Trans PAMI 21(11):1105–1120
63. Pratt I (1999) Shape representation using fourier coefficients of the sinusoidal transform. J Math Imaging Vis 10:221–235
64. Terrades OR, Valveny E (2006) A new use of the ridgelets transform for describing linear singularities in images. Pattern Recognit Lett 27(6):587–596
65. Raveaux R, Adam S, Héroux P, Trupin E (2011) Learning graph prototypes for shape recognition. Comput Vis Image Underst 115(7):905–918. Special issue on graph-based representations in computer vision
66. Riesen K, Bunke H (2009) Approximate graph edit distance computation by means of bipartite graph matching. Image Vis Comput 27(7):950–959
67. Rusiñol M, Lladós J (2009) A performance evaluation protocol for symbol spotting systems in terms of recognition and location indices. Int J Doc Anal Recognit 12(2):83–96
68. Rusiñol M, Lladós J (2010) Symbol spotting in digital libraries. Springer, London
69. Rusiñol M, Lladós J, Sánchez G (2010) Symbol spotting in vectorized technical drawings through a lookup table of region strings. Pattern Anal Appl 13(3):321–331
70. Sekita I, Kurita T, Otsu N (1992) Complex autoregressive model for shape recognition. IEEE Trans PAMI 14(4):489–496
71. Shaw AC (1969) A formal picture description scheme as a basis for picture processing systems. Inf Control 14(1):9–52
72. Siddiqi K, Shokoufandeh A, Dickinson SJ, Zucker SW (1999) Shock graphs and shape matching. Int J Comput Vis 35(1):13–22
73. Swain MJ, Ballard DH (1991) Color indexing. Int J Comput Vis 7(1):11–32
74. Tabbone S, Wendling L, Salmon JP (2006) A new shape descriptor defined on the radon transform. Comput Vis Image Underst 102(1):42–51
75. Teh C-H, Chin RT (1988) On image analysis by methods of moments. IEEE Trans PAMI 10(4):496–513
76. Tieng QM, Boles WW (1997) Wavelet-based affine invariant representation: a tool for recognizing planar objects in 3d space. IEEE Trans PAMI 19(8):846–857
77. Valveny E, Martí E (2003) A model for image generation and symbol recognition through the deformation of lineal shapes. Pattern Recognit Lett 24(15):2857–2867
78. Wu X, Bhani B (1997) Gabor wavelet representation for 3-d object recognition. IEEE Trans Image Process 6:47–64
79. Yadava RB, Nishchala NK, Gupta AK, Rastogi VK (2007) Retrieval and classification of shape-based objects using fourier, generic fourier, and wavelet-fourier descriptors technique: a comparative study. Opt Lasers Eng 45(6):695–708
80. Yang S (2005) Symbol recognition via statistical integration of pixel-level constraint histograms: a new descriptor. IEEE Trans PAMI 27(2):278–281
81. Yap P-T, Paramesran R (2005) An efficient method for the computation of legendre moments. IEEE Trans PAMI 27(12):1996–2002
82. Zahn CT, Roskies RZ (1972) Fourier descriptors for plane closed curves. IEEE Trans Comput 21(3):269–281

83. Zhang DQ, Chang SF (2004) Learning to detect scene text using a higher-order MRF with belief propagation. In: Proceedings of the 2004 conference on computer vision and pattern recognition workshop (CVPRW'04). IEEE Comput Soc. Washington, DC, USA **6**:101–108
84. Zhang D, Lu G (2003) A comparative study of curvature scale space and fourier descriptors for shape-based image retrieval. J Vis Commun Image Represent 14(1):41–60
85. Zhang D, Lu G (2004) Review of shape representation and description techniques. Pattern Recognit 37:1–19
86. Zhang W, Wenyin L, Zhang K (2006) Symbol recognition with kernel density matching. IEEE Trans PAMI 28(12):2020–2024. Senior Member-Liu Wenyin
87. Zuwala D, Tabbone S (2006) A method for symbol spotting in graphical documents. In: Bunke H, Spitz A (eds) Document analysis systems VII. Volume 3872 of lecture notes in computer science. Springer, Berlin/Heidelberg, pp 518–528

Further Reading

This chapter does not claim to be exhaustive, and therefore, interested readers are referred to broaden their knowledge with the following books [19, 68] for shape descriptions and symbol spotting considerations and [8, 27] for more global skills on shape classification and machine learning.