Marçal Rusiñol
Josep Lladós

Symbol Spotting in Digital Libraries

Focused Retrieval over Graphic-rich Document Collections

Springer

Symbol Spotting in Digital Libraries

Marçal Rusiñol • Josep Lladós

Symbol Spotting in Digital Libraries

Focused Retrieval over
Graphic-rich Document Collections

Foreword by Karl Tombre

Springer

Marçal Rusiñol
Departament de Ciències de la Computació
Centre de Visió per Computador
Universitat Autònoma de Barcelona
Edifici O, Campus UAB
08193 Bellaterra, Spain
marcal@cvc.uab.cat

Josep Lladós
Departament de Ciències de la Computació
Centre de Visió per Computador
Universitat Autònoma de Barcelona
Edifici O, Campus UAB
08193 Bellaterra, Spain
josep@cvc.uab.es

ISBN 978-1-84996-207-0 e-ISBN 978-1-84996-208-7
DOI 10.1007/978-1-84996-208-7
Springer London Dordrecht Heidelberg New York

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2010927492

Cover design: KünkelLopka GmbH, Heidelberg

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Foreword

Pattern recognition basically deals with the recognition of patterns, shapes, objects, things in images. Document image analysis was one of the very first applications of pattern recognition and even of computing. But until the 1980s, research in this field was mainly dealing with text-based documents, including OCR (Optical Character Recognition) and page layout analysis. Only a few people were looking at more specific documents such as music sheet, bank cheques or forms.

The community of *graphics recognition* became visible in the late 1980s. Their specific interest was to recognize high-level objects represented by line drawings and graphics. The specific pattern recognition problems they had to deal with was raster-to-graphics conversion (i.e., recognizing graphical primitives in a cluttered pixel image), text-graphics separation, and symbol recognition.

The specific problem of symbol recognition in graphical documents has received a lot of attention. The symbols to be recognized can be musical notation, electrical symbols, architectural objects, pictograms in maps, etc. At first glance, the symbol recognition problems seems to be very similar to that of character recognition; after all, characters are basically a subset of symbols. Therefore, the large know-how in OCR has been extensively used in graphical symbol recognition: starting with segmenting the document to extract the symbols, extracting features from the symbols, and then recognizing them through classification or matching, with respect to a training/learning set.

However, this approach has its limitations for various reasons, one of the most important being that the segmentation methods which are available do not always provide complete enough information for the recognition task to be completed. On the other hand, in order to get a better segmentation, one often needs some contextual information provided by the recognition process. This is the well-known Sayre paradox, inspired by Kenneth Sayre's early work on handwriting recognition: in order to correctly recognize, you need to segment, but in order to segment you need to recognize!

There is no perfect solution to this dilemma. However, there are actually a number of applications where the need is *not* for full-scale recognition, but rather for localization of some useful information without any claim of being able to analyze the whole content. This is sometimes called *information spotting* and has been addressed in various application contexts. At the beginning of the twenty first century, the graphics recognition community therefore acknowledged the need for *symbol spotting* methods, i.e., ways of detecting symbols in maps or technical drawings without having to actually fully segment or fully recognize everything.

It is a great pleasure for me to introduce the present work by Marçal Rusiñol and Josep Lladós; it is probably the first complete, integrated and large-scale solution to the challenge of designing a robust symbol spotting method without dedicating it to a very specific application. Drs. Rusiñol and Lladós have carefully explored the methods which can be used for that purpose. They start with basic photometric descriptors from usual computer vision techniques and identify their limitation for the symbol spotting problem. They then focus on the use of a number of features and descriptors which are specific to graphical shapes: vectorial signatures expressing the geometric and structural constraints between basic graphical entities, a prototype-based search using the decomposition into closed regions which are represented by attributed strings and organized in lookup tables, and a relational indexing approach to retrieve locations of interest.

The authors take also a very insightful look into the problem of performance evaluation of such spotting methods, so as to avoid having only subjective assessments of a method's strengths and weaknesses. This methodology can be used in a number of contexts, and I am convinced that it will be of great use to the whole graphics recognition community.

All in all, this work gives us the first general and complete framework for symbol spotting in graphical documents. It is recommended reading for any researcher wanting to contribute to this challenging problem. Of course, there are still a number of open problems, as Drs. Rusiñol and Lladós mention themselves, but I would advise to start using this framework, and then address the open problems from that point.

Nancy, France Karl Tombre

Preface

Pattern recognition systems usually consist of two main parts. On the one hand, the data acquisition and learning stage and, on the other hand, the classification of this data to a certain category. In order to recognize which category a certain query element belongs to, a set of pattern models must be provided beforehand. An off-line learning stage is needed to train the classifier and to offer a robust classification of the patterns. Within the pattern recognition field, we are interested in the document image analysis topic and, in particular, in the recognition of graphics appearing within documents rich in graphical information. In the particular case of graphical symbol recognition, descriptors are extracted from the symbol to recognize and are subsequently matched with the set symbol models. In this context, one of the main concerns is to see if the proposed systems remain scalable with respect to the data volume to be able to handle growing number of symbol models. In order to avoid working with a database of reference symbols, symbol spotting and on-the-fly symbol recognition methods have been introduced in the past years.

Generally speaking, the symbol spotting problem can be defined as the identification of a set of regions of interest from a document image, which are likely to contain an instance of a certain queried symbol without explicitly applying the whole pattern recognition scheme. Our application framework consists in indexing a collection of graphic-rich document images. This collection is queried by example with a single instance of the symbol to look for and, by means of symbol spotting methods, to retrieve the regions of interest where the symbol is likely to appear within the documents. This kind of applications are known as focused retrieval methods.

In order that the focused retrieval application can handle large collections of documents, there is a need to provide an efficient access to the large volume of information that might be stored. Indexing strategies are used in order to efficiently retrieve by similarity the locations where a certain part of the symbol appears. In that scenario, graphical patterns should be used as indices for accessing and navigating the collection of documents. These indexing mechanisms allow the user to search for similar elements using graphical information rather than textual queries.

In this book, we present a spotting architecture and different methods aimed at building a complete focused retrieval application dealing with a graphic-rich document collections.

Different symbol descriptors encoding geometric and structural information are proposed. These descriptors aim at describing parts of the symbols in a very compact and efficient way. Vectorial signatures, attributed strings and off-the-shelf shape descriptors are used to cluster parts of the symbols by similarity.

Several strategies to search for graphical information by similarity are used in this book. In order to retrieve locations from the document collection where parts of the symbols appear, we use lookup tables and grid files indexed by graphical patterns. A final validation phase is introduced to validate the hypothetic locations where a symbol is likely to be found. This validation stage is formulated in terms of spatial and relational information.

In addition, a protocol to evaluate the performance of symbol spotting systems in terms of recognition abilities, location accuracy and scalability is also studied. Evaluation measures allowing to determine the weaknesses and strengths of the methods under analysis are presented. All the methods under analysis have been tested on an experimental scenario consisting of a collection of architectural drawings with its corresponding ground-truth.

Structure

This book is divided into four parts. Part I is of introductory nature consisting of two chapters. Chapter 1 presents the symbol spotting and focused retrieval problems and outlines the proposed architecture. Chapter 2 reviews the related work to symbol spotting which has been proposed in the last years.

Part II is centered on the application of well-known methods of Computer Vision for recognizing objects in scenes to the specific problem of spotting graphical symbols in documents. Chapter 3 presents, as a running example, an application of logo spotting for a document categorization application. The method processes incoming document images such as invoices or receipts. The categorization of these document images is done in terms of the presence of a certain graphical entity detected without segmentation.

Part III is centered on the use of geometrical and structural constraints as symbol description techniques. Chapter 4 presents a method to determine which symbols are probable to be found in technical drawings by the use of vectorial signatures as symbol descriptors. Chapter 5 presents a spotting method which uses a prototype-based search as the basis for the focused retrieval task. Finally, Chapter 6 presents an indexing method to retrieve locations of interest where a query symbol is likely to be found. In order to foster the querying speed, a hashing technique is used in order to retrieve primitives by similarity very efficiently.

Part IV including just Chapter 7 is centered on the performance evaluation of spotting systems. Since symbol spotting systems and focused retrieval applications shall have the ability to recognize and locate graphical symbols in a single step, the measures to evaluate the performance of a symbol spotting system are defined in terms of recognition abilities, location accuracy and scalability.

Finally, Chapter 8 gives some concluding remarks about this study, and specifies some possible future research lines on symbol spotting techniques. Throughout this book, different symbolic databases have been used to perform the experiments. All these databases are explained in Appendix A.

Audience

This book is intended for researchers and practitioners from the field of graphics recognition who are interested in the problem of symbol spotting and focused retrieval applications in the context of digital libraries. Some basic knowledge of pattern recognition, document image analysis and graphics recognition is assumed.

Acknowledgments

We would like to thank the friends and colleagues from the Computer Vision Center. It is a pleasure to share research and teaching with all of them. Special thanks go to Prof. Juanjo Villanueva for making all this possible. We also thank the people from the Document Analysis Group for their insightful comments on our work. In particular, the authors would like to thank Dr. Ernest Valveny, Dr. Dimosthenis Karatzas, Dr. Alicia Fornés, Dr. Agnés Borràs, and Joan Mas, to name just a few.

The authors would also like to thank Prof. Karl Tombre and Dr. Philippe Dosch from the LORIA laboratory in Nancy, France, and Prof. Jean-Marc Ogier and Dr. Karell Bertet from the Université de La Rochelle, France, for their collaboration in many aspects of this work, and for hosting us during many research visits.

We would also like to acknowledge the Spanish Ministry of Research for funding this work through the grants TIN2006-15694-C02-02, TIN2009-14633-C03-03 and CONSOLIDER – INGENIO 2010 (CSD2007-00018).

Barcelona, Spain

Marçal Rusiñol
Josep Lladós

Contents

Part I
Introduction

Chapter 1
Introduction

Abstract This first chapter puts in context the symbol spotting problem. By giving a general overview of the Document Image Analysis and Recognition field and, in particular, of the Graphics Recognition research topic, we present the motivations for the present study. We summarize the objectives and contributions of this book as well as the contents of each chapter.

1.1 Document Image Analysis and Recognition Context

Document image analysis and recognition (DIAR) is one of the most important subfields of Pattern Recognition. In its early years, the research efforts were mainly focused on the processing of textual documents. In particular, most research efforts were centered on the development of automatic reader systems which entailed the design of effective page layout analysis (PLA) methods and optical character recognition (OCR) techniques. However, nowadays, commercial OCR software achieving good recognition results in type-written documents can be purchased, and we can say that OCR in type-written documents is a mature problem from the scientific point of view. Today, the interests of the Document Image Analysis and Recognition community cover a wide spectrum of open challenges. Let us enumerate a few of them. For instance, the processing of hand-written documents, for both off-line [1] and on-line [11] inputs, is still an important research topic. The huge variability of the character shapes among different writers make hand-written character recognition a much more complex and interesting problem that type-written OCR. Another research topic which has attracted the attention of researchers in the last years is the problem of processing documents acquired with low-resolution digital cameras [8]. This problem has emerged due to the presence of such cameras in ordinary devices like PDAs or cell-phones and a big number of interesting applications that can be envisaged with the inclusion of recognition tasks in portable devices. As an example, nowadays, several cell-phone models have built-in OCRs able to process business cards by finding names, phone numbers, addresses and automatically import them to the phone-book. Another actual and interesting problem is the analysis of web documents, as presented in [4, 17]. Although the processing of web documents might seem quite easy since they are digitally-born documents, they may still be a great challenge since they can contain a great amount of artwork, great variability of font types, different font sizes, non-standard layouts, a large variety of colors, etc. which present difficult recognition tasks. Finally, another example of an open problem that nowadays is receiving a lot of interest is the management of digital libraries of cultural heritage documents [2, 3]. Usually, the main problem to tackle such applications is the management of historic documents which may be very old and degraded. In addition, these document collections are quite large and the methods to analyze these documents should be conceived to provide an efficient access to such amounts of information.

1.1.1 Accessibility to Large Document Collections

Nowadays, there is still a huge amount of information stored in paper format. Libraries are the main example. For instance, the Spanish National Library[1] has about eight million paper documents (besides books) of different kinds, such as musical scores, maps, plans, engravings, etc. Great efforts are made to digitize such information mainly for space saving and preservation issues, but also in order to avoid physical boundaries and to facilitate the information retrieval. For example, *Gallica*[2] is the digital library for on-line users of the French National Library. It provides free access to 90,000 scanned and OCRed books and has made available more than 80,000 document images. The interest of providing access to books through the web has also gained importance with big initiatives such as *Google Books*.[3] However, the need of digitizing paper documents is not just a specific problem of libraries, and it is not just focused on old and rare documents which need preservation. Hundreds or even thousands of invoices, receipts, faxes, etc. can be managed per day by big companies. Obviously, the cost of storing and consulting this information in paper format becomes unaffordable, and the use of a digital collection becomes a must.

However, these huge amounts of digitized information are usually stored in poor formats making access to the contained information difficult. On the one hand, type-written documents are scanned and then transcribed by an OCR software to provide access to the text. The fact of storing these collections using the ASCII character encoding allows retrieving desired contents from the collection by using textual queries. In this particular scenario, the main challenge nowadays is to add semantic information to these digital documents in order to permit a higher level information extraction process. On the other hand, there are a lot of documents which cannot be processed by an OCR software since they are hand-written or contain non-textual information. In those cases, digital libraries use facsimile representations of these documents, i.e., the image arising from the scanning process, to store them. Even

[1]See http://www.bne.es/.

[2]See http://gallica.bnf.fr/.

[3]See http://books.google.com/.

if the use of facsimile representation is useful for storing and preservation issues, it still presents a great drawback which is the lack of accessibility. Nowadays, recognition methods for hand-written documents or non-textual elements do not reach such reliable recognition rates as OCR systems. In addition, the computational cost of recognizing type-written characters is very low in comparison with hand-written character recognition or graphic recognition schemes. These constraints provoke that usually facsimile documents are just manually annotated with a set of previously harvested metadata. This means that the only information we have about these documents is a set of predefined keywords, these documents cannot be queried in terms of their contents, but they can be retrieved just by querying the predefined keywords. This problem is common to any search tool that has to face non-textual information. For example, *Google Image Search*[4] service bases its search engine on the image filenames and text adjacent to the images. In this context, there is a need of creating tools aiming to provide efficient categorization, indexation, browsing, information retrieval in terms of visual contents, etc. for non-textual documents, without any human inspection of each document. In particular, one of the main motivations of this book is the adaptation of the idea of text mining techniques to non-textual elements. In that scenario, graphical patterns should be used as indices for accessing and navigating large collections of documents.

1.1.2 Information Spotting

The use of graphic indices to access non-textual documents is not straightforward. One of the strategies proposed to enhance the accessibility of large data collections that may result suitable in the case of non-textual documents is the *Information Spotting* technique. We can define the term *spotting* as the task of locating and retrieving specific information from large datasets without explicitly recognizing it. That means that if we want to provide a retrieval tool for non-textual documents, with a spotting approach there is no need to fully recognize all the objects conforming to a document in the database but to coarsely locate some regions of interest where the queried object is likely to be found. These spotting approaches were already proposed some years ago within the speech recognition field in order to spot spoken words from a sound recording. In [5, 6, 10], the use of hidden Markov models (HMM) allowed to process the speech signal and to focus the attention on a set of time intervals where a certain keyword is likely to be pronounced. This problem is known as *phonetic word spotting*.

Spotting techniques have also been applied to textual document images in the recent years by following the same ideas of the word spotters used in the speech recognition field. Even if images are two-dimensional structures, the text lines appearing on those documents can be segmented and subsequently taken as one-dimensional signals. These signals are then processed by a HMM or a neural network as presented in [16]. The locations where the output of the network has higher responses

[4]See http://images.google.com/.

are the ones likely to contain the queried word. These techniques can be applied to both type-written documents, as in the case of automatic fax routing applications [15], and hand-written documents, as in the historical word spotting method presented in [9]. The use of spotting methodologies to treat textual documents allows processing large amounts of documents without the need to apply an OCR software to get the ASCII characters. These methods are of particular interest in applications dealing with documents where an OCR would not produce a reliable result.

Discussed methods are, however, hardly useful when we want to treat graphic-rich documents. All word spotting methods make the strong assumption that all the objects in the document can be segmented and transformed into a one-dimensional signal in order to be processed in a linear way. This assumption is no longer valid when we want to spot graphic elements instead of textual ones. In the last years, the problem of spotting symbols within graphical documents has been an emerging research topic.

1.2 Symbol Spotting

Among the Graphics Recognition community, a lot of efforts have been devoted over the years to the problem of recognizing symbols. Several contests of Symbol Recognition have been held during the last editions of the *Graphics Recognition Workshop* (GREC). These contests are an excellent way to track the progress of the research on this specific problem and aim at determining the challenges and the future research directions. In the GREC 2007 edition,[5] an important challenge to be addressed has been identified. For many years, researchers of the Symbol Recognition community centered their methods on recognizing isolated symbols undergoing several transforms and degradations. Nowadays, state-of-the-art recognition schemes yield performances far above 90% recognition rates, but the real challenge is not achieving the 100% rate but rather it should be centered on three different aspects. Firstly, we should see if the proposed methods are really scalable in terms of the number of symbols to recognize. Secondly, we should test if the proposed methods could be applied to any symbol design or whether they are ad-hoc conceived to recognize a specific dataset and tackle a specific source of noise. Finally and most importantly, we should consider if these methods are able to recognize symbols present in complete drawings without previous segmentation. In this direction, the concept of spotting graphical symbols within graphic-rich documents has been introduced by Tombre and Lamiroy in [13]. Five years later, the authors presented some achievements in this field by pointing several open challenges in [14].

Generally speaking, the *Symbol Spotting* problem is defined as the location of a set of regions of interest from a document image which are likely to contain an

[5] Seventh IAPR International Workshop on Graphics Recognition, GREC07. Curitiba, Brazil, 20–21 September 2007.

instance of a certain queried symbol without explicitly recognizing it. One of the main applications for symbol spotting methods is its use in large collections of documents. This particular application can be seen as a content based image retrieval (CBIR) application but having some particularities. The main difference is that standard document retrieval approaches find atomic documents, leaving to the user the tasks of locating relevant information within the provided results. Whereas symbol spotting provides the user a more direct access to relevant information by returning a set of regions of interest which are sub-parts of the documents in the collection. Such applications which return passages of interest within documents instead of complete documents, are known as *Focused Retrieval* systems. The interested reader is referred to the recent review by Joty and Sadid-Al-Hasan [7] on the topic of focused retrieval.

To the best of our knowledge, in the workshops organized by the focused retrieval community,[6] no works dealing with graphics have ever been proposed, and all the works have been centered on the retrieval of textual passages from ASCII documents. Going back to the image documents, there is an important difference between the spotting systems dealing with graphics and those dealing with word images. In the case of word spotting, usually a learning step is required and only a small subset of keyword queries is allowed. In the symbol spotting problem, the amount of items comprising the symbol alphabet can increase indefinitely. In addition, the input of a spotting system is the user's query symbol which he wants to retrieve from the whole collection. Therefore, usually the spotting systems are queried by example. That is, the user segments a symbol he wants to retrieve from the document database and this cropped image acts as the input. This particularity reinforces the fact that spotting methods should not work for a specific set of model symbols nor have a learning stage where the relevant features describing a certain symbol are trained. The retrieval of the relevant zones should be done on-the-fly. Nevertheless, in the acquisition step, i.e., when a given document is added to the collection (which is a process that could be done off-line) several steps of primitive extraction and description can be computed. The desired output of the spotting methods is a ranked list of zones of interest likely to contain similar symbols to the queried one. That is, each result should have an associated confidence value depending on a certain similarity function between the query and the result. We can see an overview of the symbol spotting methodology applied to a focused retrieval application in Fig. 1.1.

In Document Image Analysis and Recognition and in Computer Vision in general, the relationship between recognition results and segmentation performance presents a common problem known as the Sayre paradox [12]. In order to achieve good recognition results, the objects should be previously segmented, but to get a reliable segmentation, the objects should be previously recognized. To avoid such a paradox, symbol spotting architectures do not use a preliminary segmentation step

[6]The first Workshop on Focused Retrieval http://www.cs.otago.ac.nz/sigirfocus/ held in Amsterdam in 2007, and the second Workshop on Focused Retrieval http://www.cs.otago.ac.nz/sigirfocus2008/ held in Singapore in 2008.

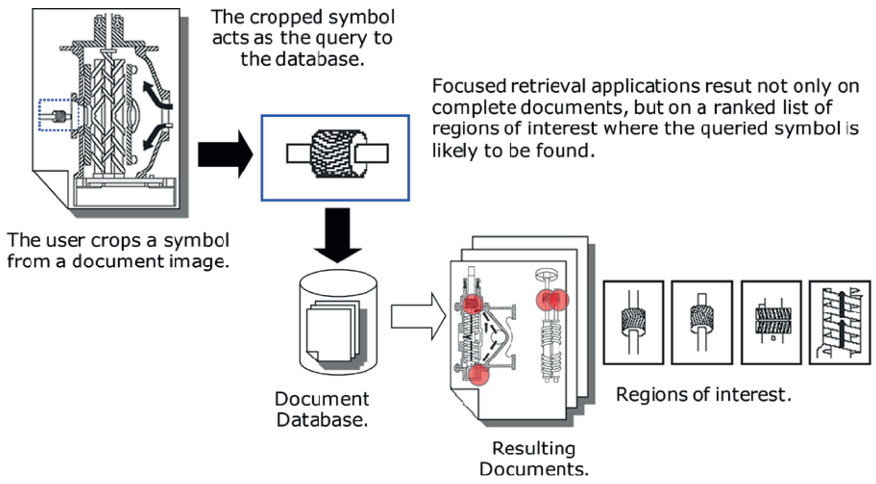

The cropped symbol acts as the query to the database.
Focused retrieval applications resut not only on complete documents, but on a ranked list of regions of interest where the queried symbol is likely to be found.
The user crops a symbol from a document image.
Document Database.
Regions of interest.
Resulting Documents.

Fig. 1.1 Symbol spotting applied to a focused retrieval application overview

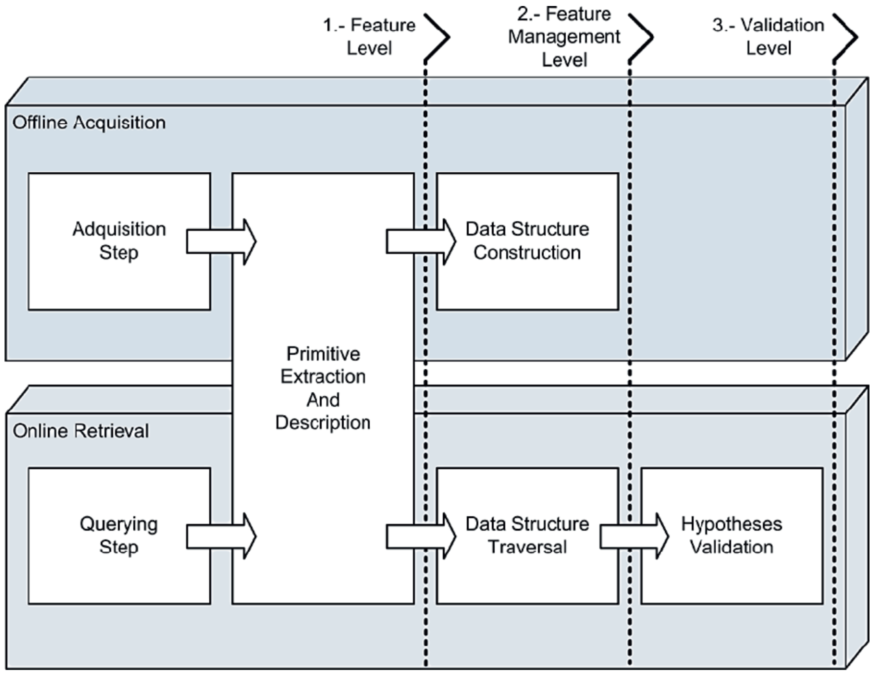


Fig. 1.2 General architecture of a symbol spotting system

followed by a proper recognition method, but are usually conceived to coarsely recognize and segment in a single step. We can appreciate our proposal of a general architecture for symbol spotting systems in Fig. 1.2. Basically, three different levels can be identified. The first level aims at representing and compactly describing the primitives that compound the graphical symbols. These features describing graphical symbols are then stored in a particular data structure. This data structure should be chosen carefully in order to provide efficient access to the symbol descriptors. During the querying process, this data structure is traversed and the locations within the document images where to find similar primitives as the queried ones are retrieved. A final validation stage determines the valid hypotheses where the queried symbol is likely to be found.

Summarizing, the present study has been motivated by the specific problem of proposing a spotting methodology applied to a focused retrieval problem. The proposed methods should be able to locate and retrieve graphical content within a database of complete document images. From a methodological point of view, the main challenges stem from the nature of the queries, which are iconic queries instead of the ASCII strings used in the keyword-based searches. The fact of working with graphical entities raises several problems to tackle. The first important problem is how to compactly represent and describe symbols without a preliminary segmentation stage. Another important issue is the choice of the data structures allowing to efficiently retrieve graphical patterns by similarity.

1.3 Outline of this Book

The main objective of this study is to propose a symbol spotting methodology for locating graphic symbols within a collection of complete documents. The spotting method is formulated in terms of a search by similarity of all the primitives which comprise the queried graphical symbol. Among the wide variety of possible symbols and graphic documents, we have basically focused our research on a framework dealing with technical line-drawings such as architectural floor-plans or electronic schemes.

To this end, the problem will be tackled from different points of view and this main objective can be detailed into the following points:

1. **Testing Well-known Methods from the Computer Vision Field**
   Although symbol spotting has its own particularities, the problem of locating symbols in documents can be seen as a particular case of the object recognition problem from the Computer Vision field. Our first objective is to test if such well-known techniques can be applied to the problem of spotting graphic symbols. In this part of the book, we describe graphical symbols by means of well-known photometric descriptors. We identify the limitations of those approaches in the particular scenario of spotting symbols in line-drawing collections.

   The main contribution of this part does not correspond to the recognition methodology, since we use off-the-shelf recognition methods, but shows an application of this kind of techniques to the graphics recognition domain.

2. **Geometric and Structural Symbol Description Techniques**
   Since this book is mainly focused on technical line-drawings, our specific framework is mostly centered on the use of geometric and structural constraints to describe graphical symbols and graphic-rich documents. The primitives to extract and the description techniques to represent a graphical symbol are expressed and defined in the vectorial domain instead of working with the raw image format. The objective of this part is to find a methodology to describe symbols to cope with the different noise sources that we face in our framework. We present three different proposals of vectorial primitives and the subsequent symbol description techniques:

   - **Vectorial Signatures** The use of signatures as a coarse description technique is usually used on spotting systems. Taking vectors as the primitives which compound a graphical symbol, a model of vectorial signature is proposed. The symbols are described by the occurrences of simple geometric configurations among segments.
   - **String Representation of Polygons** The second proposal to represent graphical symbols is the use of a higher-level entity than segments. In this case, graphical symbols are described by a set of chains of adjacent segments grouped into polygon instances. These polygons are described as one-dimensional attributed strings, and the distance between two similar polygons is computed by using string edit operations.
   - **Off-the-shelf Shape Descriptors Applied to Vectorial Primitives** Finally, we will study the use of several well-known shape descriptors applied to the vectorial primitives which comprise a symbol. In this case, the contribution is not the descriptors themselves but its use to represent vectorial symbols.

3. **The Descriptors Organization**
   The second main research axis of this study is centered on how the primitives' descriptors can be organized in a data structure for posterior efficient access. The main objective of this part is to find mechanisms allowing graphical patterns to be used as indices so as to provide an efficient access to graphic information contained in large data corpora.

   Throughout this book, we present three different approaches, each one related to the previous description of symbols. These structures aim at organizing by similarity all the extracted vectorial primitives from the documents in the collection. In focused retrieval applications, it is indispensable to avoid one-to-one matching when querying a certain graphical primitive by providing mechanisms which allow searching for graphical primitives by similarity.
4. **The Hypotheses Formulation**
   The last step of spotting architectures is the hypotheses formulation. Regions of interest where the queried symbol is likely to appear have to be generated with their associated confidence values. The main objective of this part is to present validation schemes to reduce the false alarms that may appear from the retrieval of primitives by similarity.

   Inspired by the classical voting schemes where the hypotheses' validation is done in terms of an accumulation of evidence, we present a validation scheme to discard false alarms. In addition of the accumulation of evidence in terms of locations within a document where the query symbol can be found, we also propose a relational validation method which also takes into account the spatial configuration and the structural relationships among the primitives which comprise a graphical symbol.

5. **Performance Evaluation**
   Finally, one of the main concerns of the Graphics Recognition community is the generation of evaluation studies to assess and compare the accuracy and robustness of the proposed methods. To the best of our knowledge, there have been very few attempts to describe a performance evaluation protocol for symbol spotting architectures applied to focused retrieval tasks.

   Inspired by several works on the performance evaluation of Graphics Recognition methods and algorithms, and on the evaluation measures used in the Information Retrieval field, we propose a set of measures to evaluate the performance of symbol spotting systems in terms of their localization and recognition abilities.

1.4 Organization

The rest of this book is organized into eight chapters and one appendix, structured as four main parts.

Part I

In Chapter 2, the state-of-the-art in symbol spotting is reviewed. Since symbol spotting is quite an emerging topic, the literature dealing with this problem is not vast. Some other works which are not directly related to the problem of spotting symbols, but which may be related to one of the three levels of the spotting architecture, are presented. After a brief overview of the literature of symbol spotting, we organize this chapter into three differentiated parts, namely, the state-of-the-art in symbol description techniques, in feature organization and in hypotheses validation.

Part II

The second part of this book is centered on the application of well-known methods of Computer Vision for recognizing objects in scenes to the specific problem of spotting graphical symbols in documents.

- In Chapter 3, we present a method for spotting symbols by using techniques from the Computer Vision field. As a running example, we present an application of logo spotting for a document categorization application. The method processes incoming document images such as invoices or receipts. The categorization of these document images is done in terms of the presence of a certain graphical entity detected without segmentation. The symbols are described by a set of local features computed using the well-known methods of SIFT and shape context descriptors. The categorization of the documents is performed by the use of a bag-of-visual-words model. Spatial coherence is introduced by a voting scheme in order to reinforce the correct category hypotheses, aiming also to spot the logo inside the document image. Experiments which demonstrate the effectiveness of this system on a large set of real data are presented.

Part III

The third part of this book is devoted to proposing spotting methods in a framework of line-drawing images. Therefore, it is centered on the use of geometrical and structural constraints as symbol description techniques.

- In Chapter 4, we present a method to determine which symbols are probable to be found in technical drawings by the use of vectorial signatures as symbol descriptors. The proposed signature model is formulated in terms of geometric and structural constraints among segments, as parallelisms, straight angles, etc. After representing vectorized line drawings by attributed graphs, our approach works with a multi-scale representation of these graphs, retrieving the features that are expressive enough to create the signature. A window-based system aims at computing these signatures within complete documents, identifying the zones of interest where a symbol is likely to appear.
- In Chapter 5, we present a spotting method which uses a prototype-based search as the basis for the focused retrieval task. First, symbols are decomposed into primitives representing closed regions. These primitives are then encoded in terms of attributed strings. Second, the strings are organized in a lookup table so that the set median strings act as representative prototype of the clusters of similar primitives. This indexing data structure aims at efficiently retrieving the locations from the document collection where similar primitives as the queried ones can be found. Finally, a voting scheme formulates hypotheses about the locations of the line drawing image where there is a high presence of regions similar to the queried ones, and therefore a high probability to find the queried graphical symbol. The proposed approach has been proved to work even in the presence of noise and distortion introduced by the scanning and raster-to-vector processes.
- In Chapter 6, we present an indexing method to retrieve locations of interest where a query symbol is likely to be found. In order to foster the querying speed, a hashing technique is proposed, which is able to retrieve primitives by similarity very efficiently. Vectorial primitives are coarsely encoded by well-known shape description methods providing a numerical description of the primitives. A relational indexing approach is presented in order to introduce some structural information of the symbols and to provide an accurate hypotheses validation. Experimental results show the performance of the proposed approach.

Part IV

Finally, the fourth part of this book has just one chapter focused on the performance analysis of spotting methods.

- Chapter 7 is centered on the performance evaluation of spotting systems. Since symbol spotting systems and focused retrieval applications shall have the ability to recognize and locate graphical symbols in a single step, the measures to evaluate the performance of a symbol spotting system are defined in terms of recognition abilities, location accuracy and scalability. By testing the spotting method of Chapter 6, we show that the proposed measures allow determining the weaknesses and strengths of the analyzed method.

Finally, in Chapter 8, we give some concluding remarks about this study, and we specify some possible future research lines on symbol spotting techniques.

Throughout this book, different symbolic databases have been used to perform the experiments. All these databases are explained in Appendix A. For each database, we detail the kind of symbols it contains and the distortions which have been introduced in the original elements. Some other characteristics for each database, as the number of elements, the number of primitives in the vectorial representation, their size, etc., are also detailed.

References

1. Bertolami, R., Bunke, H.: Hidden Markov model-based ensemble methods for offline handwritten text line recognition. Pattern Recognition **41**(11), 3452–3460 (2008)
2. Bulacu, M., van Koert, R., Shomaker, L., van der Zant, T.: Layout analysis of handwritten historical documents for searching the archive of the cabinet of the dutch queen. In: Proceedings of the Ninth International Conference on Document Analysis and Recognition, pp. 357–361. IEEE Computer Society, Los Alamitos (2007)
3. Coüasnon, B., Camillerapp, J., Leplumey, I.: Access by content to handwritten archive documents: Generic document recognition method and platform for annotations. International Journal on Document Analysis and Recognition **9**(2–4), 223–242 (2007)
4. Esposito, F., Ferilli, S., Mauro, N.D., Basile, T.: Incremental learning of first order logic theories for the automatic annotations of web documents. In: Proceedings of the Ninth International Conference on Document Analysis and Recognition, pp. 1093–1097. IEEE Computer Society, Los Alamitos (2007)
5. Gish, H., Ng, K.: A segmental speech model with applications to word spotting. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 447–450. IEEE Computer Society, Los Alamitos (1993)
6. Jones, G., Foote, J., Jones, K., Young, S.: Video mail retrieval: The effect of word spotting accuracy on precision. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 309–312. IEEE Computer Society, Los Alamitos (1995)
7. Joty, S., Sadid-Al-Hasan, S.: Advances in focused retrieval: A general review. In: Proceedings of the Tenth International Conference on Computer and Information Technology, pp. 1–5. IEEE Computer Society, Los Alamitos (2007)
8. Liu, X., Doerman, D.: Mobile retriever: Access to digital documents from their physical source. International Journal on Document Analysis and Recognition **11**(1), 19–27 (2008)

9. 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)
10. Rohlicek, J., Jeanrenaud, P., Ng, K., Gish, H., Musicus, B., Siu, M.: Phonetic training and language modeling for word spotting. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 459–462. IEEE Computer Society, Los Alamitos (1993)
11. Saldarriaga, S., Morin, E., Viard-Gaudin, C.: Categorization of on-line handwritten documents. In: Proceedings of the Eighth IAPR International Workshop on Document Analysis Systems, pp. 95–102. IEEE Computer Society, Los Alamitos (2008)
12. Sayre, K.: Machine recognition of handwritten words: A project report. Pattern Recognition **5**(3), 213–228 (1973)
13. Tombre, K., Lamiroy, B.: Graphics recognition—from re-engineering to retrieval. In: Proceedings of the Seventh International Conference on Document Analysis and Recognition, pp. 148–155. IEEE Computer Society, Los Alamitos (2003)
14. Tombre, K., Lamiroy, B.: Pattern recognition methods for querying and browsing technical documentation. In: Progress in Pattern Recognition, Image Analysis and Applications, *Lecture Notes on Computer Science*, vol. 5197, pp. 504–518. Springer, Berlin (2008)
15. Viola, P., Rinker, J., Law, M.: Automatic fax routing. In: Document Analysis Systems VI, *Lecture Notes on Computer Science*, vol. 3163, pp. 484–495. Springer, Berlin (2004)
16. Wiener, E., Pedersen, J., Weigend, A.: A neural network approach to topic spotting. In: Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval, pp. 317–332 (1995)
17. Yoshida, M., Nakagawa, H.: Web document parsing: A new approach to modeling layout–language relations. In: Proceedings of the Ninth International Conference on Document Analysis and Recognition, pp. 203–207. IEEE Computer Society, Los Alamitos (2007)

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

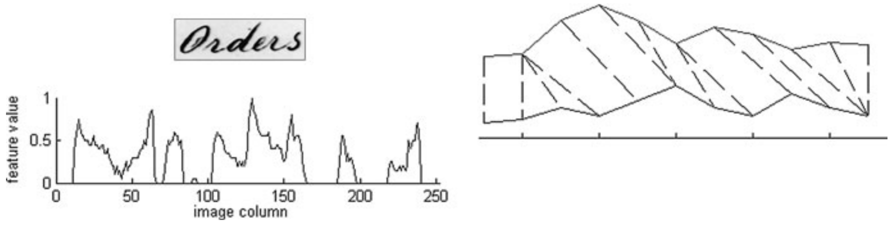


Fig. 2.1 Word image signature for word spotting using the DTW distance (this image is based on Figs. 2 and 4 appearing in [72])

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

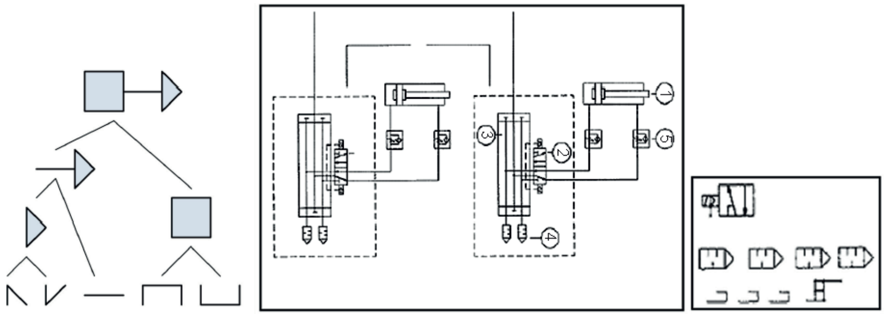


Fig. 2.2 Dendrogram representation for spotting symbols in technical drawings (this image is based on Fig. 3.5 appearing in [103])

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-

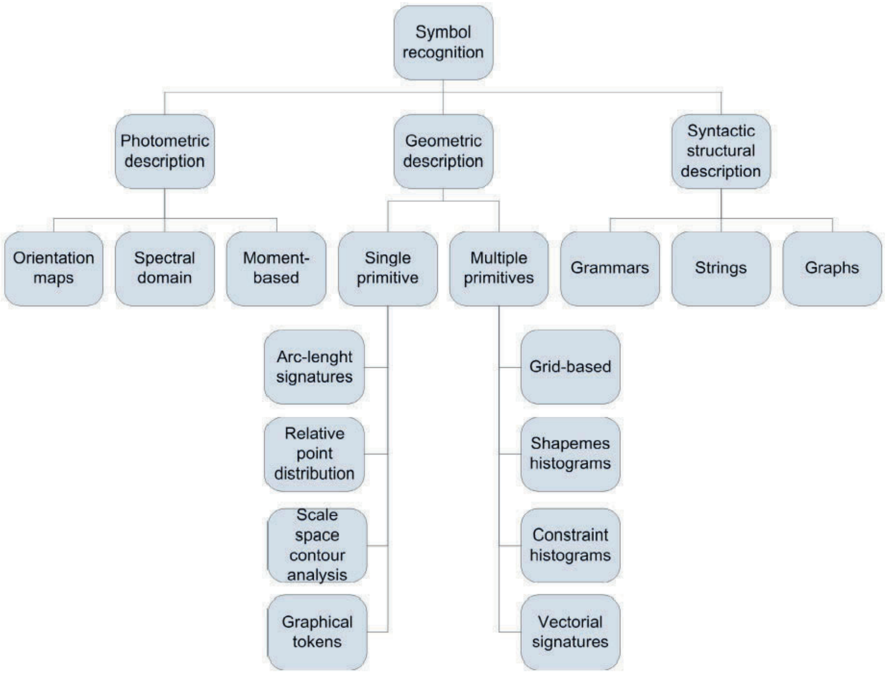


Fig. 2.3 Classification of symbol representation and description techniques

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 recognitionis 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 4×4 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 σ 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 looses 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,

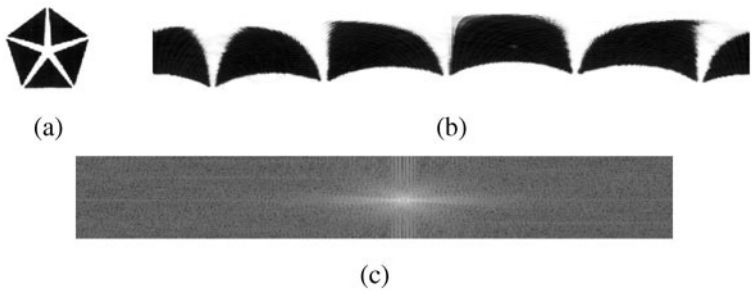


Fig. 2.4 Example of a generic Fourier descriptor. (**a**) An original logo image; (**b**) polar-raster sampled image plotted in Cartesian space; (**c**) Fourier spectra of (**b**); (this image is based on Figs. 4 and 5 appearing in [100])

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 φ-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 σ. As the σ value increases, little inflections are eliminated from the contour and the shape becomes more and more smooth. The inflection points

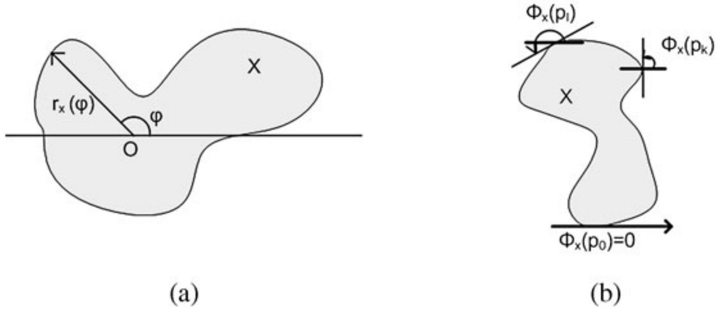


Fig. 2.5 Computation of the arc-length-based signatures. (**a**) The radius-vector function; (**b**) the tangent-angle function; (this image is based on Figs. 2.1.3 and 2.1.9 appearing in [40])

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.

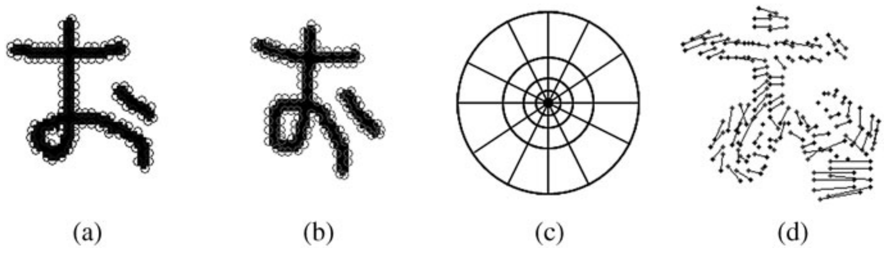


Fig. 2.6 Example of the shape context descriptor for shape matching. (**a**)–(**b**) Original shapes to match with sampled edge points; (**c**) diagram of the log-polar histogram bins used in computing the shape contexts; (**d**) correspondences found using bipartite matching for the two shapes (**a**) 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 τ_i is described through the features (m_i, θ_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 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 built 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

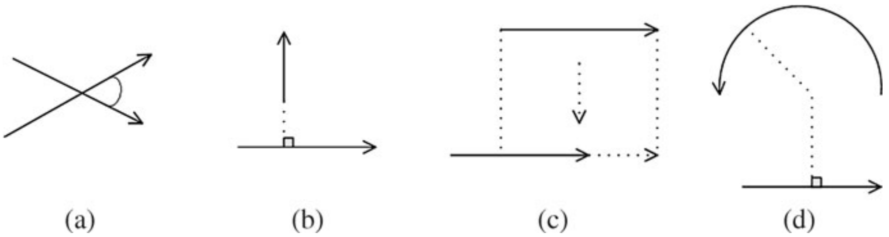


Fig. 2.7 Geometric constraints taken as features to build a vectorial signature. (**a**) Intersection of segments; (**b**) parallelism; (**c**) perpendicularity; (**d**) constraints regarding arcs and circles; (this image is based on Figs. 3, 4, 5, 6, and 7 appearing in [96])

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,

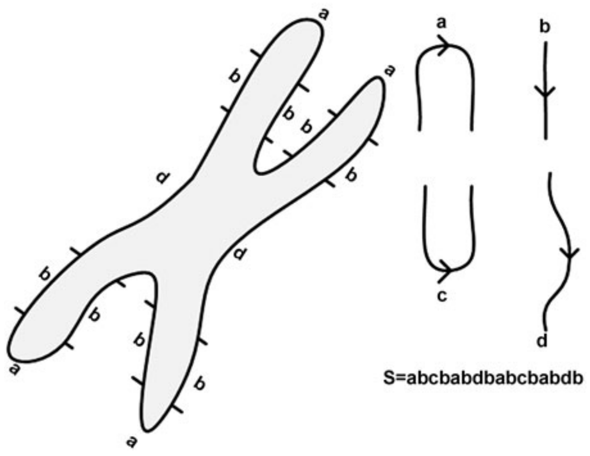


Fig. 2.8 String representation of closed contours (this image is based on Fig. A.1 appearing in [26])

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

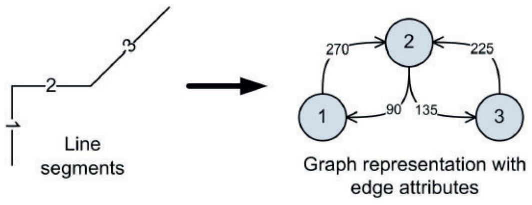


Fig. 2.9 Attribute relational graph for symbol description. Nodes represent line segments and are attributed by their length, while edges represent an adjacency relationship and are attributed by the formed angle between the two segments (this image is based on Fig. 3 appearing in [60])

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

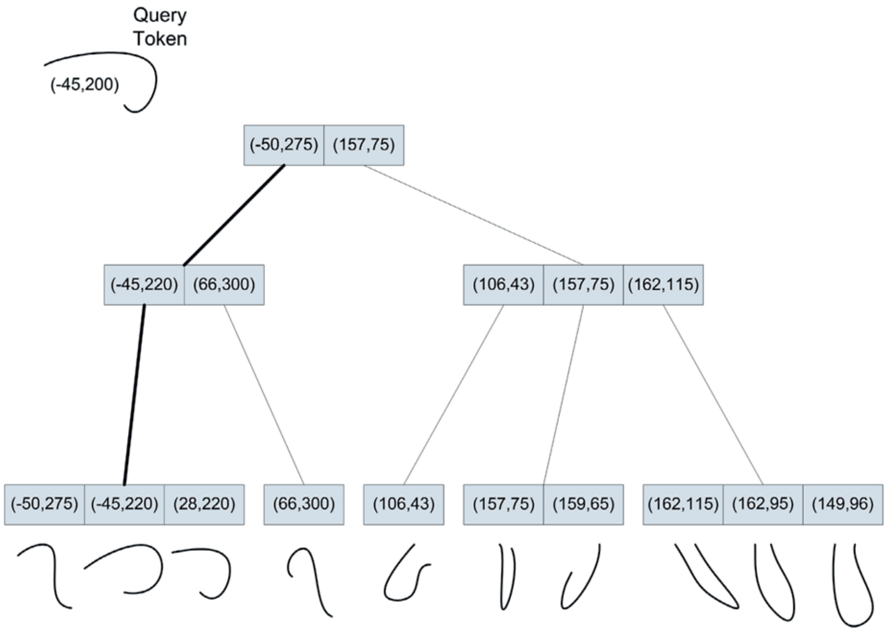


Fig. 2.10 Hierarchical organization of descriptors by using an M-tree structure. Traversing the structure allows a nearest neighbor search in logarithmic time (this image is based on Fig. 11 appearing in [12])

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 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,

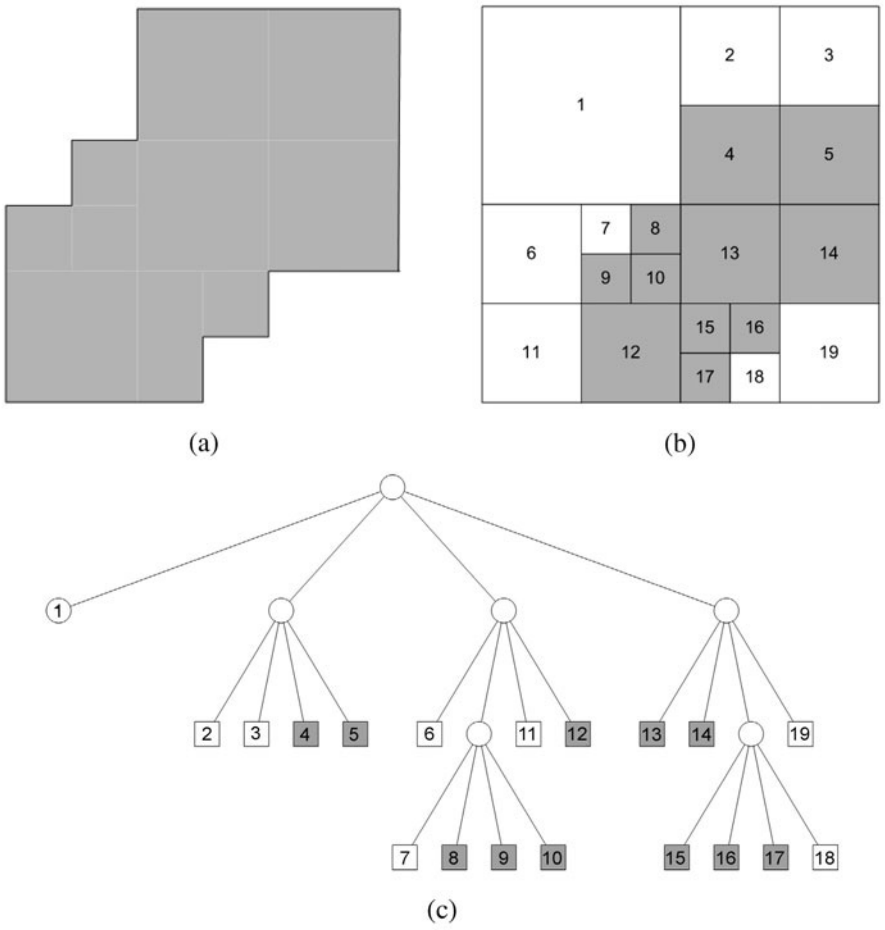


Fig. 2.11 Quadtree representation of a shape. (**a**) Sample region; (**b**) its maximal blocks from the array representation; (**c**) its quadtree representation (this image is based on Fig. 1 appearing in [75])

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^*-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^*-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., Serratosa, 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., Serratosa, 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., Mullot, 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., Hérroux, 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.: Reconaissance de symboles sans connaisance 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)

Part II
On the Use of Photometric Descriptors for Symbol Spotting

Chapter 3
Symbol Spotting for Document Categorization

Abstract In this chapter, we present a method for spotting symbols in document images by using a photometric description of symbols. As a running example we present an application of logo spotting. The presented method uses a bag-of-words model in order to perform a categorization of document images such as invoices or receipts. The hypotheses validation is done in terms of spatial coherence by the use of a Hough-like voting scheme. Experiments which demonstrate the effectiveness of this system on a large set of real data are presented at the end of the chapter.

3.1 Introduction and Related Work

The problem of locating symbols within document images can be seen as a particular case of the object recognition problem from the Computer Vision field. In this first part of this book, we want to test if some well-known techniques from the object recognition field can be applied to the specific case of symbol spotting. Instead of a focused retrieval application, we propose an application of detecting logos in document images such as invoices, receipts, etc. for document categorization.

Companies deal with large amounts of paper documents in daily workflows. Incoming mail is received and has to be forwarded to the correspondent addressee. A study on the invoice processing in several German companies [6] revealed that on average the cost of manually processing (opening, sorting, internal delivering, data typing, archiving) these incoming documents is about 9€ per invoice. These costs represent a substantial amount of money if we consider the number of documents received by a big company at the end of the day.

Several systems intended to automatically process incoming documents have been designed over the years. As an example, Viola et al. [16] presented a system to automatically enroute incoming faxes to the correspondent recipient. However, most of the existing systems only process typewritten information, making the assumption that the recipient information is printed in the document image. In many cases, graphic elements present in the documents convey a lot of important information. For instance, if a company receives a document containing the logo of a bank, usually this document should be forwarded to the accounting department, whereas if the document contains the logo of a computer supplier, it is quite probable that the document should be addressed to the IT department. The categorization of documents may also have other applications besides the automatic rerouting. For instance, it is helpful in organizing documents and providing efficient access to all the documents coming from a certain supplier. The recognition of such graphic elements can help introduce contextual information to overcome the semantic gap between the simple recognition of characters and the derived actions to perform brought by the document understanding. In this chapter, we use the presence of graphical symbols (logos) to categorize the class of the incoming documents.

Many contributions exist in the Graphics Recognition literature that deal with logo recognition and retrieval, e.g., the recent work on trademark recognition from Wei et al. [17]. However, they just focus on isolated or pre-segmented graphic images which are affected by synthetic noise and deformation sources. As noted in [15], one of the big challenges for the next years for the Graphics Recognition community is the localization/recognition of graphic symbols appearing in complete documents without any previous segmentation. To the best of our knowledge, in the literature, only Zhu and Doerman [18, 19] addressed the problem of logo spotting by means of a cascade of classifiers. In this chapter, we propose a method to categorize documents and to detect graphical logos in a single step. The main contribution of this chapter is the use of well-known strategies of the Computer Vision field to this particular kind of images. State-of-the-art photometric descriptors are used to characterize graphical symbols and a bag-of-visual-words approach is presented to categorize the documents. Such approaches are commonly used in object recognition (see, e.g., [14]) and image classification applications. To the best of our knowledge, very few works have been proposed in the literature using such descriptors to the domain of document images. Due to the binary nature of the document images, usually, photometric descriptors are not well suited for document analysis applications. However, the fact that the recently proposed descriptors work at several scales and blur the image makes its use on binary images possible. The bag-of-visual-words is an analogy to the Computer Vision domain of the classic bag-of-words model where a text is represented by an unordered set of words. In that case, an image is represented by a collection of image patches. By the combination of photometric descriptors and a bag-of-visual-words model, we propose a segmentation-free recognition method which does not rely on a learning step but uses a single instance of logo models so as to benefit the scalability of the method.

The presented application, however, differs a little from the main objective of this book. By means of spotting graphical elements, we want to build an indexing mechanism to query a large collection of documents and perform focused retrieval tasks. Whereas object recognition methods rely on an off-line learning of the models to search for, indexing methods should be queried by example without any training step. Object recognition methods have an off-line stage where a classifier is trained with several examples of the features extracted from the models to recognize and, usually, a set of negative examples to be considered as non-objects. Once the classifier is trained, the images are given as input of the system, and the regions of interest of this image, where any of the trained models appear, are retrieved. On the other hand, indexing mechanisms have also an off-line step in which the documents are acquired and some features are extracted and organized, but the input of the system is one single instance of the model to retrieve. The main difference of such applications stems for the training stage and the nature of the input, even if both applications can perform spotting and focused retrieval tasks.

The remainder of this chapter is structured as follows. The next section presents an overview of the proposed method. In Sect. 3.3, we detail the detection procedure from the feature extraction to the bag-of-words model used to categorize the documents. Section 3.4 focuses on the addition of a set of spatial coherence rules which aim to refine the results and, moreover, to perform logo spotting in addition to the categorization. Section 3.5 presents the experimental setup by using a large set of real documents. Finally, the conclusions and a short discussion can be found in Sect. 3.6.

3.2 Outline of the Approach

Our document categorization method is based on the presence of graphical logos in the incoming documents. This application is like a particular case of the problem of object recognition but has certain particularities. First of all, the documents are in binary format and are affected by the noise arising from the different acquisition systems. Since photometric descriptors are used to process gray-level (or even color) images, usually when trying to codify a binary images, we obtain poorly discriminative feature vectors. This may cause the number of false alarms to increase. Secondly, object recognition methods usually rely on a costly learning stage where a classifier is trained with multiple instances of the objects to recognize. In our application, in order to benefit the scalability of the method, no learning stage is involved, and a single instance of the logos to locate is needed. Generally speaking, the presented method has a structure like the one proposed by Sivic et al. in [14], where a bag-of-words model is translated to the visual domain by the use of photometric descriptors over the interest points.

We can see an overview of the presented method in Fig. 3.1. The extracted local features from a document are matched against the codeword dictionary and an accumulator is used in order to decide which category the queried document belongs to. In the next sections, we will further detail the following steps.

3.3 Document Categorization by Logo Detection

The document categorization and the logo detection is performed by using a bag-of-words model of visual words. These visual words are defined in terms of local features extracted from a photometric descriptor. Let us first detail how these features are extracted and computed, and then focus on the bag-of-words model.

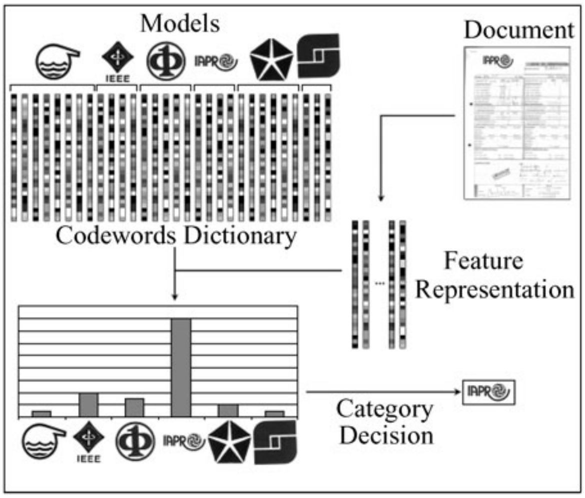


Fig. 3.1 Overview of the proposed document categorization method

3.3.1 Feature Extraction and Description

Our method is inspired by the work of Bagdanov et al. presented in [1], focused on the recognition of trademarks in real images. In that work, the authors use SIFT features to match trademark models against video frames. We use a similar matching approach, whereas our aim is to categorize and to use several different logos as models. Logos are represented by a photometric descriptor applied to a set of previously extracted key points.

3.3.1.1 Interest Point Detection: Harris–Laplace Detector

The interest points are detected by using the Harris–Laplace detector presented by Mikolajczyk and Schmid in [10]. This algorithm extracts points with high curvatures (e.g., corners or junctions) and automatically selects the scale of the region where the photometric descriptor is to be computed. Let us briefly review how this detection algorithm works.

The corner detector proposed by Harris and Stephens in [4] is based on the second moment matrix. This matrix is then adapted to scale changes to make it independent of the image resolution. The scale-adapted second moment matrix is defined by:

$$\mu(\mathrm{x}, \sigma_I, \sigma_D) = \sigma_D^2 \mathrm{g}(\sigma_I) \times \begin{bmatrix} L_x^2(\mathrm{x}, \sigma_D) & L_x L_y(\mathrm{x}, \sigma_D) \\ L_x L_y(\mathrm{x}, \sigma_D) & L_y^2(\mathrm{x}, \sigma_D) \end{bmatrix}, \tag{3.1}$$

where L_a is the derivative computed in the a direction. The local derivatives are computed with Gaussian kernels of size σ_D. The derivatives are then averaged in the neighborhood of the point by smoothing with a Gaussian window of size σ_I.

The Harris measure is then defined in terms of the trace and the determinant of this second moment matrix as:

$$M_c = \det\big(\mu(\mathrm{x}, \sigma_I, \sigma_D)\big) - \kappa \ \mathrm{trace}^2\big(\mu(\mathrm{x}, \sigma_I, \sigma_D)\big), \tag{3.2}$$

where local maxima of M_c determine the location of interest points, with κ being a tunable sensitivity parameter. The Harris–Laplace detector uses the scale-adapted Harris function from (3.2) to localize points in scale-space. The scale-space representation of the Harris function is built for pre-selected scales $\sigma_n = \xi^n \sigma_0$ where ξ is experimentally set to 1.4. The matrix $\mu(\mathrm{x}, \sigma_n)$ is computed with the integration scale $\sigma_I = \sigma_n$ and the local scale $\sigma_D = s\sigma_n$ with an experimentally set parameter $s = 0.7$. For each point, an iterative algorithm that detects the location and the scale of interest points is applied. The extrema over scale of the Laplacian-of-Gaussian, (3.3) are used to select the scale of interest points by rejecting the points for which the LoG response does not attain any extremum or which response is below a certain threshold.

$$\big|\mathrm{LoG}(\mathrm{x}, \sigma_n)\big| = \sigma_n^2 \big|L_{xx}(\mathrm{x}, \sigma_n) + L_{yy}(\mathrm{x}, \sigma_n)\big|. \tag{3.3}$$

3.3.1.2 Interest Point Description: SIFT and Shape Context

After the interest points are detected in an image, a photometric descriptor has to be applied to each region of interest defined by these key points. In our experiments, we use and compare the performance of two different photometric descriptors. On the one hand, we use the SIFT features and, on the other hand, we use the shape context descriptor.[1] As we will see in the experimental results section, each descriptor has its own strengths and weaknesses. Both descriptors are computed with the code provided by Mikolajczyk et al.[2]

SIFT descriptors, presented by Lowe in [7, 8], are computed for normalized image patches arising from the key point detection stage. The descriptor is a histogram of gradient locations and orientations. The locations are quantized into a 4×4 location grid and the gradient angles are quantized into eight predefined orientations. The resulting descriptor has 128 dimensions. Each orientation plane represents the gradient magnitude corresponding to a given orientation. In order to obtain illumination invariance, the descriptor is normalized by the square root of the sum of squared components.

The shape context descriptor implementation, based on the original presented by Belongie et al. in [2], is similar to the SIFT descriptor, but is based on edges. Shape context is a histogram of edge point locations and orientations. Edges are extracted

[1]Note that in Chap. 2 we classified the shape context descriptor as a geometric descriptor since it copes with spatial arrangement of points. However, the enhancement of this descriptor proposed by Mikolajczyk and Schmid in [11], which takes into account not only point locations but also gradient magnitudes and orientations, allows this modified version to be considered as belonging to the photometric class.

[2]See http://www.robots.ox.ac.uk/~vgg/research/affine/index.html.

by the Canny detector. Location is quantized into nine bins of a log-polar coordinate system and orientation quantized into four bins. A 36 dimensional descriptor is therefore obtained. In addition, the point contribution to the histogram is weighted with the gradient magnitude.

In the next section, we will see how we formally describe logos with the above presented key point detection and description methods, and how similar logos can be matched.

3.3.2 Logo Representation and Matching

A given logo S_i is represented by its n_i interest points extracted from the Harris–Laplace detector. Each of these key points is then described by a feature vector arising from a photometric descriptor. A logo instance is thus formally represented as:

$$S_i = \big\{(x_k, y_k, s_k, F_k)\big\} \quad \text{for } k \in \{1, \dots, n_i\}, \tag{3.4}$$

where x_k and y_k are the x- and y-position, and s_k the scale of the kth key point. F_k corresponds to the photometric description of the region represented by the key point. An individual key point k of the logo S_i will be denoted as S_i^k. The same notation applies when the key points and the description vectors are computed over a complete document D_j. The matching between a key point from the complete document and the ones of the logo model is computed by using the first two nearest neighbors:

$$\begin{aligned} N_1\big(S_i, D_j^q\big) &= \min_k (F_q - F_k), \\ N_2\big(S_i, D_j^q\big) &= \min_{k \neq N_1(S_i, D_j^q)} (F_q - F_k). \end{aligned} \tag{3.5}$$

Then the matching score is determined as the ratio between these two neighbors:

$$M\big(S_i, D_j^q\big) = \frac{N_1(S_i, D_j^q)}{N_2(S_i, D_j^q)}. \tag{3.6}$$

If the matching score M is lower than a certain threshold t this means that the key point is representative enough to be considered. By setting a quite conservative threshold ($t = 0.6$ in our experiments), we guarantee that the appearance of false positives is minimized since only really relevant matches are considered. We can appreciate an example of the feature extraction and matching between a model and a document in Fig. 3.2. However, for categorization purposes, we cannot directly apply this matching procedure between the query document and all the model logos we consider. We use instead a bag-of-words model which has reached successful results for topic categorization. Let us describe in the next section how we adapt this model to the visual domain.

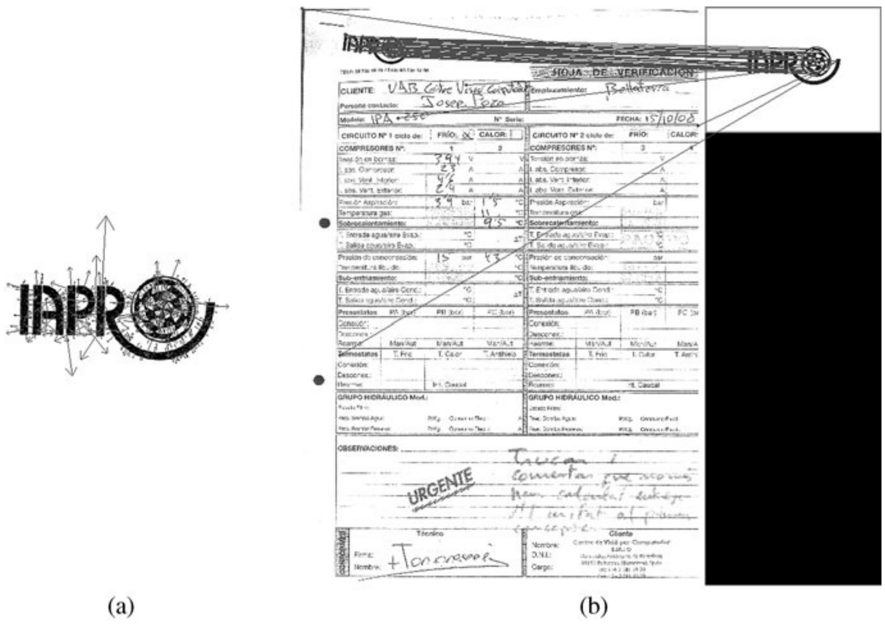


Fig. 3.2 Matching logos in documents with the SIFT features. (**a**) SIFT features computed over an isolated logo model; (**b**) feature matching between the model and the complete document

3.3.3 Bag-of-Visual-Words

The bag-of-visual-words is an analogy to the Computer Vision domain of the classic bag-of-words model, where a text is represented by an unordered set of words. In that case, an image is represented by a collection of image patches. In our particular case, given a set of logo models considered as different categories, we extract all the feature vectors F_k^i from them. Each feature vector is associated with its corresponding logo model S_i. By joining all the feature vectors from all the logos, we obtain the codeword dictionary $W = [F_1^1, F_2^1, \dots, F_k^i]$. This dictionary is computed off-line from all the model logo database. Given a query document D_j, all the feature vectors D_j^q are used as indexes and matched against the codewords of the dictionary W. The matching function M_q returns the index i corresponding to the logo class of the matched feature vector F_k^i as follows:

$$M_q = \{ i \mid M(W, D_j^q) < t \}. \tag{3.7}$$

Finally, the determination of whether a document contains a logo is done by using by accumulating hypotheses of document categories in a histogram H.

$$\begin{aligned} &H[M_q] = 0 \quad \text{at initialization}, \\ &H[M_q] = H[M_q] + 1 \quad \text{for } q \in \{1, \dots, n_j\}. \end{aligned} \tag{3.8}$$

In the original bag-of-words model, a given document is categorized in terms of the frequency of appearance of certain words. For each document category, we have a histogram of frequencies, and in order to determine which category an incoming document belongs to, distances among its histogram and all the model histograms must be computed. We face here a slightly different problem. If in a given document we have several appearances of parts of a given logo, we shall consider that probably the document contains this logo. The document category is thus determined by searching the maximum m of the accumulator H after normalizing each accumulation cell with the total number n_i of features of the corresponding logo k. If the value of m is less than a threshold T, which has been experimentally set, we consider that the document does not contain any logo and is categorized in a rejection class.

3.4 Introducing Spatial Density for Logo Spotting

Whereas bag-of-words models have been very successful in the text domain, the analogy to visual words for image categorization has an important drawback. Bag-of-words models completely ignore the spatial relationship among features. Even if this drawback in the text domain is overcome due to the important impact of few keywords, in the image domain it is an important burden since the spatial layout among features has similar importance as the feature description itself.

It has been shown that the spatial organization of photometric descriptors computed from key points is a powerful tool to recognize objects in scenes and to index images in terms of their contents as, for instance, the work of Mikolajczyk and Schmid presented in [9] shows. In the document analysis field, Nakai et al. [12, 13] introduced a method to retrieve document images acquired with a camera from a large image database using the arrangement of invariants computed over extracted feature points. The results are promising in terms of accuracy, time and scalability.

In order to overcome this drawback, we use a simple, yet effective voting scheme to guarantee that the spatial organization of features maintains certain coherence by introducing a density factor. Before contributing to the accumulator H, we get rid of the all the feature points of the same category i that are isolated in space. A Hough-like approach is used to transform the matched key points from the image domain to a three-dimensional parameter space in order to cluster reliable model hypotheses that agree upon a particular model pose. The three-dimensional parameter space is built from the x- and y-locations of the matched key points and the third dimension i which represents the logo class. This parameter space is quantized, and the problem of finding coherent locations is transformed into the problem of finding maxima in this parameter space. By this we mean that only clusters of key points which belong to the same category and which are close in space are considered. As we can see in Fig. 3.3, all the false alarms when matching key points are eliminated. The gray dots are inconsistent hypotheses, and the black dots maintain a certain spatial coherence and are taken as likely hypotheses. The bounding-boxes of likely hypotheses are returned to the user as the zones of the document image where the logo should

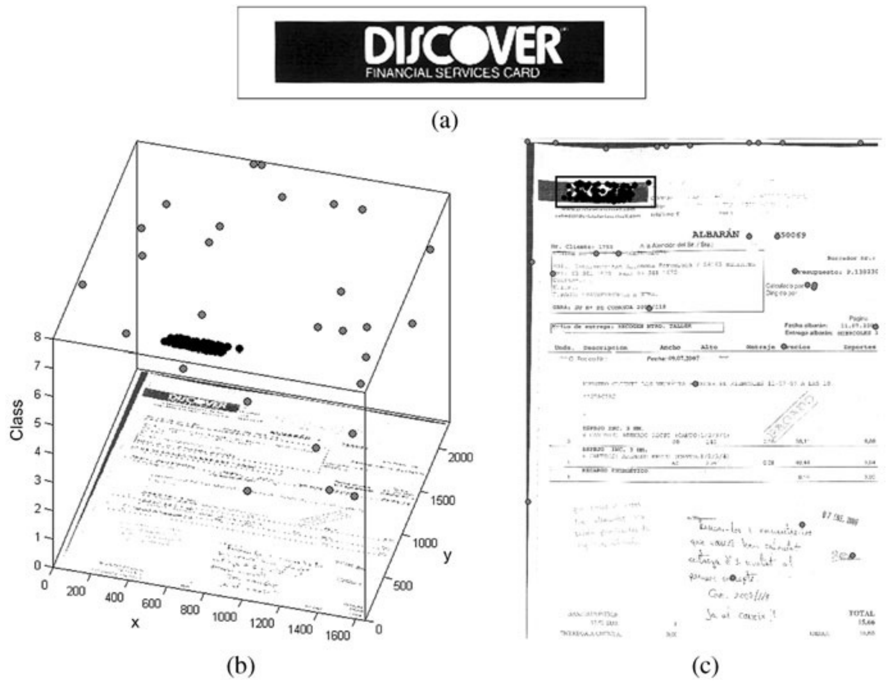


Fig. 3.3 Introducing spatial density information to spot logos. (**a**) Logo model; (**b**) parameter space; (**c**) spotted region of interest

be found. The presented method, given that in a single step a document is able to categorize it in a certain class and return the zones of the document which contain the logo.

3.5 Experiments

To provide a realistic evaluation of the proposed method, we used a large document collection. The collection consists of 1,000 real document images which were sent by fax and then scanned. These images correspond to several kinds of documents such as invoices, letters, receipts, forms, etc. They contain both typewritten and handwritten text. Graphical elements such as logos, stamps, tables, etc. are also present in most of these documents. Typical dimensions of documents are about 2,500 $\times$ 3,500 pixels with varying resolutions. All the images were scanned in binary format by using the built-in thresholding method of the scanner. Ground-truth of the entire collection was manually created identifying 18 different logo classes appearing in nearly 180 images, the rest of document images do not contain any logo and are used to test if the presented method is also able to reject these documents.

3.5.1 Evaluation Methodology

We will base our performance evaluation on how well the categorization of the documents is done. The performance of categorization methods is usually evaluated by confusion matrices to see if the systems under evaluation confuse two classes, mislabeling one as the other. In addition, the true positive rate (*TPR*) and false positive rate (*FPR*) are used as evaluation measures in order to compare the performance among different methods. These ratios are derived from the contingency table and defined in terms of the amount of true positives (*TP*), false positives (*FP*), true negatives (*TN*) and false negatives (*FN*):

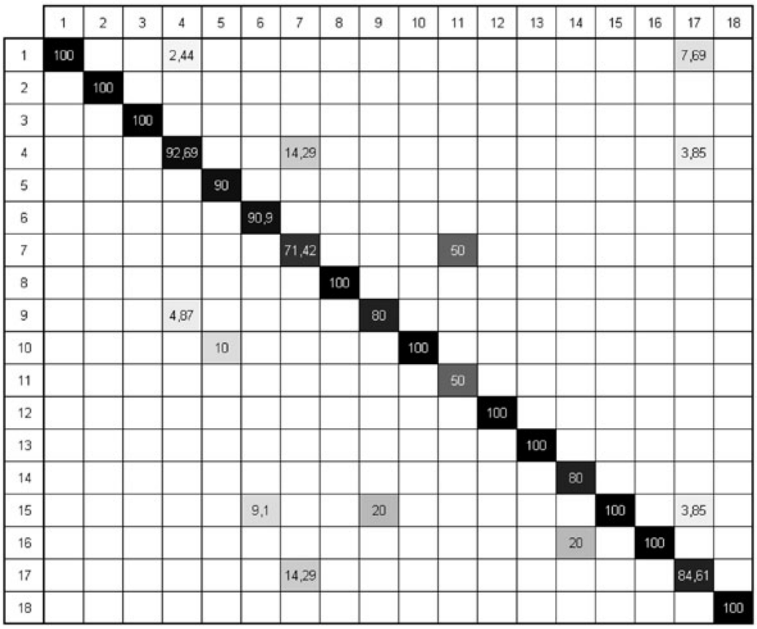
$$\begin{aligned} TPR &= \frac{TP}{(TP+FN)}, \\ FPR &= \frac{FP}{(FP+TN)}. \end{aligned} \tag{3.9}$$

The *TPR* ratio measures the effectiveness of the system in retrieving the relevant items, whereas the *FPR* ratio measures the probability that a non-relevant document is retrieved by the query. In our experiments, we use the *TPR* ratio to summarize the correct categorization of documents containing a given logo. The *FPR* is used to measure the number of documents that do not contain any logo which are incorrectly identified as belonging to a certain class.

3.5.2 Performance Comparison

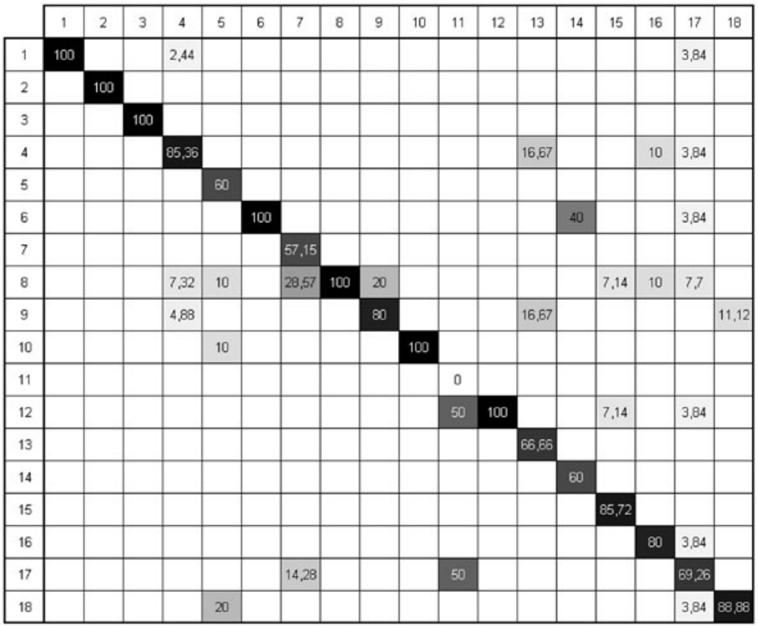
We can appreciate the obtained confusion matrices after running the whole experimental categorization in Fig. 3.4. We can appreciate some differences between the use of SIFT features and the shape context descriptor. For example, when using shape context, a lot of documents are incorrectly classified as class 8 (shown in row 8), or the documents corresponding to class 17 are usually misclassified in other document categories (shown in column 17). These misclassifications lead the overall *TPR* shown in Table 3.1 to be lower when using the shape context descriptors than when using SIFT features. On the other hand, when we test the documents that do not contain any logo and should be categorized in the rejection class, the SIFT features perform worse than the shape context descriptor, as the *FPR* of Table 3.1 shows. In addition, the computational complexity when using SIFT is higher due to the highest number of dimensions of the feature vectors than when using the shape context descriptor, resulting in a higher querying time.

In conclusion, the use of SIFT features should be preferred in applications where it is important to correctly identify the incoming documents, no matter if false alarms (documents which do not contain any logo) are present. On the other hand, if for the intended application it is preferable to minimize the false alarms even if we reject or misclassify some documents, or if we want a faster method, the shape context descriptors should be considered.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	100			2,44													7,69	
2		100																
3			100															
4				92,69			14,29										3,85	
5					90													
6						90,9												
7							71,42				50							
8								100										
9				4,87					80									
10					10					100								
11											50							
12												100						
13													100					
14														80				
15						9,1			20						100		3,85	
16														20		100		
17							14,29										84,61	
18																		100

(a)



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	100			2,44													3,84	
2		100																
3			100															
4				85,36									16,67			10	3,84	
5					60													
6						100								40			3,84	
7							57,15											
8				7,32	10		28,57	100	20						7,14	10	7,7	
9				4,88					80				16,67					11,12
10					10					100								
11											0							
12											50	100			7,14		3,84	
13													66,66					
14														60				
15															85,72			
16																80	3,84	
17							14,28				50						69,26	
18					20												3,84	88,88

(b)

Fig. 3.4 Confusion matrices for the document categorization experiment. (**a**) Using SIFT features; (**b**) using the shape context descriptor

Table 3.1 Evaluation measures for the document categorization experiment

Descriptor	*TPR* (%)	*FPR* (%)	Time (s)
SIFT	92.2	1	3.25
SC	81.6	0.3	1.34

3.6 Conclusions and Discussion

In this chapter, we have presented a method for spotting logos in document images by using a photometric description of symbols. The use of a bag-of-words model reformulated to manage feature vectors arising from photometric descriptors combined with a Hough-like voting approach to guarantee the spatial and density coherence aims at spotting logos inside the document image and, in addition, determining the category of the queried document. The presented experiments demonstrate the effectiveness of the method on a large set of real document images.

The presented application, although it can be understood as a symbol spotting application, has been inspired by the characteristics of the object recognition methods from the Computer Vision field. The spotting of logos by means of a bag-of-words model applied to incoming documents is useful for categorization purposes, but not for indexing or browsing of a large collection of documents. The main application in the rest of this book is the focused retrieval in a collection of line-drawing images rather than the object recognition.

In the following part of this book, we will focus on the use of geometric and structural description techniques for the representation of graphical symbols rather than photometric descriptors. Although photometric descriptors yield good recognition results and can be used as a basis for symbol recognition and matching applications, they have several limitations in the context of spotting graphical symbols from line-drawings.

The first conclusion that can come to our mind is that photometric descriptors encode several visual cues at the same time, but the line-drawing images are usually binary and the symbols appearing in them are usually made just from line segments. Since the only discriminative visual cue to recognize such symbols is the shape, it seems more natural to use a geometric or structural description to really cope with the useful information. However, as we can appreciate in Fig. 3.5, the results of matching line-drawn symbols by a photometric descriptor (the SIFT features in this case) are not bad at all. We can notice nevertheless that for simpler symbol designs (see Fig. 3.5b) very few key points are matched since the description is not discriminative enough. This factor can be problematic when the images in the collection are affected by some noise, and the symbol can be completely lost if these few key points cannot be matched against the model. The main factor provoking the discriminative power loss is that, in the case of line-drawings, the presence of a corner or a junction is not so relevant as in the case of real images (or the logos in the document analysis context). The information conveyed by the gradient magnitudes and orientations is not really discriminant in this particular context.

In addition, there is another limitation to the use of photometric descriptors in the context of spotting symbols for indexing a large collection of line-drawings.

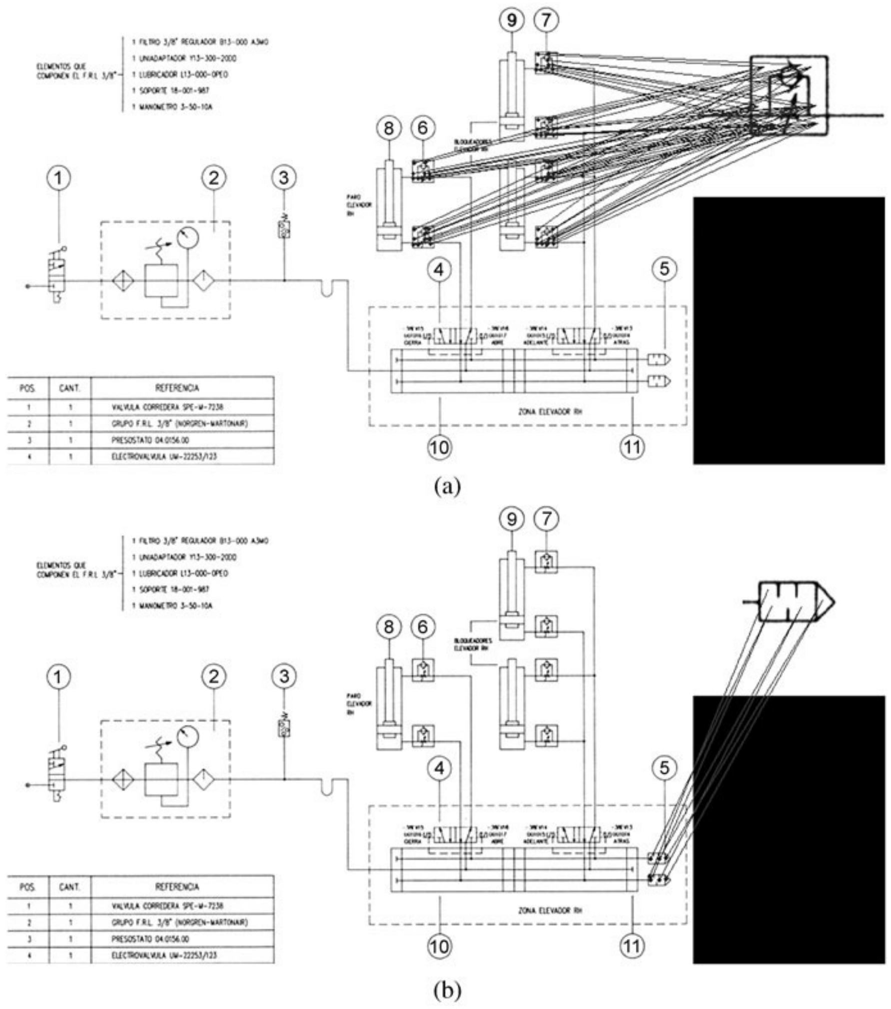

(a)
(b)
1
2
3
4
5
6
7
8
9
10
11
ZONA ELEVADOR RH
POS.
CANT.
REFERENCIA
VALVULA CORREDERA SPE-W-7238
GRUPO F.R.L. 3/8" (NORGREN-MARTONAIR)
PRESOSTATO 04.0156.00
ELECTROVALVULA UW-22253/123
ELEMENTOS QUE COMPONEN EL F.R.L. 3/8"
1 FILTRO 3/8" REGULADOR B13-000 A3MO
1 UNIADAPTADOR Y13-300-2000
1 LUBRICADOR L13-000-OPEO
1 SOPORTE 18-001-987
1 MANOMETRO 3-50-10A

Fig. 3.5 Matching symbols in line-drawings with the SIFT features

Usually, photometric descriptors tend to be high-dimensional. The SIFT descriptor has 128 dimensions, whereas the adaptation of the shape context descriptor has 36 dimensions. This high-dimensionality helps to be discriminant enough to recognize objects in real images, but hinders the possibility of building indices over the high-dimensional description space. Even if Califano and Mohan claim in [3] that multi-dimensional indexing performs better than when using smaller spaces, the curse of dimensionality affects such high-dimensional spaces. In order to reduce the impact of the curse of dimensionally when trying to index such descriptors, a dimension reduction step such as PCA proposed by Ke and Sukthankar in [5] should be studied. Since geometric and structural description techniques are based on a prior primitive extraction, they tend to have lower dimensionalities than photometric descriptors which work at pixel level.

In the next part of this book, we will see three different approaches for symbol spotting in line-drawings which are based on a geometric and structural description of the symbols.

References

1. 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)
2. 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)
3. Califano, A., Mohan, R.: Multidimensional indexing for recognizing visual shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence **16**(4), 373–392 (1994)
4. Harris, C., Stephens, M.: A combined corner and edge detector. In: Proceedings of the Alvey Vision Conference, pp. 147–151 (1988)
5. Ke, Y., Sukthankar, R.: PCA-SIFT: A more distinctive representation for local image descriptor. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 506–513. IEEE Computer Society, Los Alamitos (2004)
6. Klein, B., Agne, S., Dengel, A.: Results of a study on invoice-reading systems in Germany. In: Document Analysis Systems VI, *Lecture Notes on Computer Science*, vol. 3163, pp. 451–462. Springer, Berlin (2004)
7. 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)
8. Lowe, D.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60**(2), 91–110 (2004)
9. Mikolajczyk, K., Schmid, C.: Indexing based on scale invariant interest points. In: Proceedings of the Eighth IEEE International Conference on Computer Vision, pp. 525–531. IEEE Computer Society, Los Alamitos (2001)
10. Mikolajczyk, K., Schmid, C.: Scale & affine invariant interest point detectors. International Journal of Computer Vision **60**(1), 63–86 (2004)
11. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence **27**(10), 1615–1630 (2005)
12. Nakai, T., Kise, K., Iwamura, M.: Camera-based document image retrieval as voting for partial signatures of projective invariants. In: Proceedings of the Eighth International Conference on Document Analysis and Recognition, pp. 379–383. IEEE Computer Society, Los Alamitos (2005)
13. Nakai, T., Kise, K., Iwamura, M.: Use of affine invariants in locally likely arrangement hashing for camera-based document image retrieval. In: Document Analysis Systems VII, *Lecture Notes on Computer Science*, vol. 3872, pp. 541–552. Springer, Berlin (2006)
14. 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)
15. Valveny, E., Dosch, P., Fornés, A., Escalera, S.: Report on the third contest on symbol recognition. In: Graphics Recognition. Recent Advances and New Opportunities, *Lecture Notes on Computer Science*, vol. 5046, pp. 321–328. Springer, Berlin (2008)
16. Viola, P., Rinker, J., Law, M.: Automatic fax routing. In: Document Analysis Systems VI, *Lecture Notes on Computer Science*, vol. 3163, pp. 484–495. Springer, Berlin (2004)

17. Wei, C., Li, Y., Chau, W., Li, C.: Trademark image retrieval using synthetic features for describing global shape and interior structure. Pattern Recognition **42**(3), 386–394 (2009)
18. Zhu, G., Doerman, D.: Automatic document logo detection. In: Proceedings of the Ninth International Conference on Document Analysis and Recognition, pp. 864–868. IEEE Computer Society, Los Alamitos (2007)
19. Zhu, G., Doerman, D.: Logo matching for document image retrieval. In: Proceedings of the Tenth International Conference on Document Analysis and Recognition, pp. 606–610. IEEE Computer Society, Los Alamitos (2009)

Part III
On the Use of Geometric and Structural Constraints for Symbol Spotting

Chapter 4
Vectorial Signatures for Symbol Recognition and Spotting

Abstract In this chapter, we present a method to determine which symbols are probable to be found in technical drawings by the use of vectorial signatures as symbol descriptors. The proposed signature model is formulated in terms of geometric and structural constraints among segments, such as parallelisms, straight angles, etc. After representing vectorized line drawings with attributed graphs, our approach works with a multi-scale representation of these graphs, retrieving the features that are expressive enough to create the signature.

4.1 Introduction and Related Work

Since in the context of recognizing and locating graphical symbols from line-drawing images, the most important visual cue to describe graphical elements is the shape, a geometric and structural description of primitives seems the most natural choice. In order to apply such description techniques, a primitive extraction step is needed. Graphical symbols are broken down into lower level graphical primitives such as contours, loops, connected components, skeletons, etc. In the field of symbol discrimination, the approach which has probably gained most attention is the use of vectorial signatures as the description technique to represent the graphical symbols. Vectorial signatures are geometric symbol descriptors which compactly encode the symbol in terms of particular geometric constraints among line primitives. The total amount of occurrences of each constrain forms the final signature. In order to compute the descriptors, the images must be processed in order to be broken down into segments by the use of a raster-to-vector conversion algorithm. Since spotting techniques are intended to coarsely recognize symbols, these particular descriptors are conceived as being very compact and having enough discriminative power at least to identify most of the zones of interest of a document image where a given symbol is likely to appear. Let us briefly overview the related work on the use of signatures for symbol description and focused retrieval.

One of the first vector-based signatures has been proposed by Ventura and Schettini in [13]. In their work, the authors propose a signature for recognizing symbols from the architectural and electronic fields. First, line-drawing images are pre-processed by a thin/thick line separation algorithm, and then the thin lines are polygonally approximated by straight segments. From the segments composing a symbol, they extract a number of geometric features such as the number of segments intersecting in one point, the angles among these segments, their lengths, etc. Thick structures are described by their area, orientation and second order geometric moments. All these features are combined to create the signature describing the vectorized symbols. In order to make the signature more reliable, two values are added to each feature: a tolerance threshold and a weight. In the recognition step, the signature of the symbol to recognize is compared with all the model signatures. The distance among features is computed dependent on the tolerance threshold and normalized by the corresponding weight. A global threshold finally determines whether the query symbol matches a certain model. Results show efficient recognition, but this approach has the strong limitation that it has been conceived just to recognize isolated or pre-segmented graphical symbols.

Recently, Zhang and Wenyin [15] presented another model of vectorial signature for symbol description. Starting from the assumption that the symbols are in vectorial forms, primitive-pair relationships are recorded and employed to create the signature which is subsequently used as descriptor. Besides the basic relationships among segments, the authors propose a set of measures in order to describe several relationships among primitives having different nature, i.e., relationships among segments and arcs, segments and circles, etc. The proposed descriptor, however, can also only be used to recognize isolated symbols.

Usually, in order to have a powerful representation of the vectorial symbols to easily compute the signatures, an attributed graph is used. We can find several examples in the literature. Coustaty et al. [2] and Qureshi et al. [9] use a graph representation of the symbols. At the nodes of the graph, primitives are stored, and edges encode a certain geometric relationship among pairs of primitives. The use of the adjacency matrix of the attributed graph as descriptor has been widely used. However, despite the representative power of this structure, the proposed signatures are only tested in a symbol recognition framework working with pre-segmented instances of the graphic elements to recognize.

Inspired by the work of Ventura and Schettini and using some of the geometric features conceived to describe line-patterns presented by Etemadi et al. in [4], Dosch and Lladós [3] proposed another signature model. In addition to the description itself, the authors proposed a windowing methodology in order to be able to discriminate the regions of interest within a line-drawing where a given symbol is likely to be found. The signature of a graphic element is defined as a set of elementary features, containing intrinsically a discrimination potential. The method starts by a study on 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 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. The zones of interest are built from a decomposition of the image in several non-overlapping tiles. The graphical primitives are then stored in these buckets, and relationships are only computed between the primitives of a bucket and the primitives of its neighboring buckets. The results show that this simple implementation can discriminate the learned symbols. A lot of false alarms are present, however, especially with symbols not contained in the library. Symbols containing arcs often lead to some non-relevant signatures. But the main drawback is the fixed bucket partition of the image that makes the method not really scale invariant and causes a lack of flexibility.

Besides the fact that most of the approaches of vectorial signatures are focused on the application of symbol recognition and not on symbol spotting, we find that most of these approaches present another important drawback. Since vectorial signatures describe symbols in terms of geometric and structural constraints among sets of primitives, the inclusion of errors in the process of primitive extraction may provoke large variations in the signature, thus entailing a severe loss of discriminative power. We propose a signature model inspired in the work of Huang [5], where the main primitives describing a symbol are not just straight segments but are more complex sub-shapes composing a symbol. Inspired on the work of Dosch and Lladós [3], we also propose a window-based methodology allowing to compute different signatures within the whole graphic document, permitting to locate symbols appearing within a complete document image.

The remainder of this chapter is structured as follows. The next section presents the pre-processing step to transform from the raw images acquired with the scanner to a vectorial format by introducing some state-of-the-art methods we use to achieve a raster-to-vector conversion. In Sect. 4.3, our vectorial signature model is presented. Subsequently, in Sect. 4.4, we define the window-based methodology which allows the computation of signatures within complete graphic documents. Section 4.5 presents the experimental results. Finally, the conclusions and a short discussion can be found in Sect. 4.6.

4.2 Pre-processing Step: Raster-to-Vector Conversion

In Part III of this book, we focus on the particular application of spotting symbols in line-drawings by means of geometric and structural description techniques. Both description families work with primitives such as line-segments, arcs, etc.; thus, they need a prior step of conversion from the raw image to the primitive domain. In this section, we present the pre-processing algorithms we use to convert the raw images to the vectorial format. Basically, we follow three steps. First, gray-scale images are denoised and binarized. In a second step, the skeletons of the foreground components are extracted. Finally, these skeletons are polygonally approximated.

Two different approaches to reach a raster-to-vector conversion can be found in the literature. Some methods are based on the combination of a skeletonization algorithm, followed by some kind of polygonal approximation. On the other hand, there exists another family of approaches which are not based on a recursive thinning but on contour following. These algorithms do not compute the skeleton but extract paths which are equidistant from contour lines, and approximate two parallel contour lines by segments. Although both families have their advantages and drawbacks, we decided to use a skeleton-based vectorization. The interested reader can find a review of the most suitable methods to build a raster-to-vector system in [11].

In this book, we work with the QGAR[1] implementation of the raster-to-vector conversion. In the QGAR library, the raster-to-vector conversion is based on Trier and Taxt's binarization, the $(3,4)$-Distance Transform skeletonization, and Rosin and West's polygonal approximation algorithm. In the following subsections, we review these three methods.

4.2.1 Document Binarization

First, grayscale images should be denoised using simple operations based on morphological operations. When working with scanned documents, the inherent noise and distortions such as warping, paper folds, paper stains, etc. arising from these processes have to be faced. The interested reader is referred to Loce and Dougherty's review [6] of some simple existing techniques for digital acquired document enhancement and restoration.

After the document beautification stage, the graysacle line-drawing images should be transformed into binary format. A lot of well-known binarization methods exist, the interested reader is referred to the recent benchmarking study of binarization methods of Ntirogiannis et al. [8]. We chose to use the approach of Trier and Taxt [12]. This binarization method was conceived to treat document images and yields good results. The interested reader can find a recent comparative study on the performance of different binarization techniques in [8].

Trier and Taxt's method is based on the method by White and Rohrer [14] where a gradient-like operator is used to achieve a three level label image. Pixels with activity below a manually set threshold T_A are labeled '0'. Then if the Laplacian edge operator of the pixel is positive, the pixel is labeled '$+$', otherwise '$-$'. The idea is that in a sequence of labels, edges are identified as '$-+$' or '$+-$' transitions and object pixels are assumed to be '$+$' and '0' labels between a '$-+$' and '$+-$' pairs.

Trier and Taxt improved this method by three modifications. First, by smoothing the input image with a 5×5 mean filter in order to remove some noise. Then a print pixel identification is done in order to delete the false positives corresponding to noise blobs that are still present in the background area. The constraint of the original method, namely that '$+$' marked regions should be surrounded by '$-$' pixels to be labeled as print, is not a sufficient criterion to remove the false print objects. For each '0' marked region, the number of '$-$' and '$+$' labels that are 8-connected is counted, and the pixel is labeled print only if the number of '$+$' pixels is larger. Finally, a postprocessing step is proposed to remove false print objects. The average gradient value at the edge of each printed object is computed. Objects having an average gradient below a threshold T_P are labeled as misclassified, and are removed. In Fig. 4.1, we can see the results of binarizing an old document by several methods.

[1]See http://www.qgar.org.

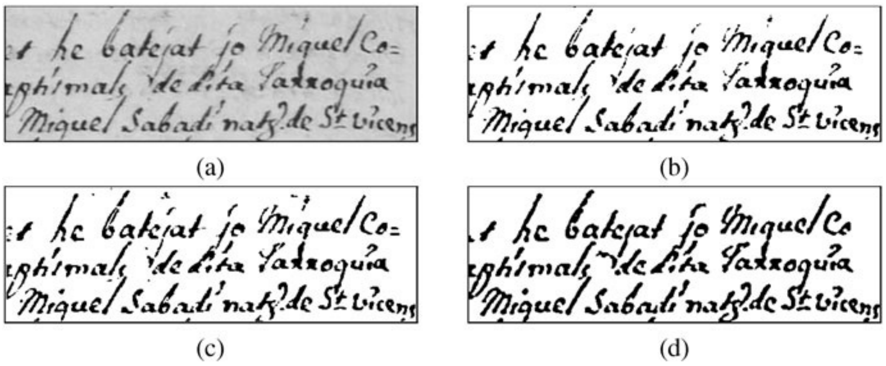


Fig. 4.1 Binarization of an old handwritten document. (**a**) Original image; (**b**) Otsu's binarization; (**c**) Niblack's binarization; (**d**) Trier and Taxt's binarization

Both Otsu's and Niblack's well-known methods to binarize images leave some misclassified regions, whereas Trier and Taxt's method performs a good binarization of document images.

4.2.2 Skeletonization

Sanniti di Baja [1] proposed a skeletonization method which is not based in thinning operations, but rather on the analysis of the $(3,4)$-Distance Transform of the binary image. Each pixel of the shape is labeled with its distance to the contour. Each pixel p can be interpreted as the center of a disc, which includes all the pixels whose distance from p is less than the label of p. A disc D_p not completely included in the disc D_q centered at any neighbor q of p is called a maximal disc. The skeleton of a shape will include all the centers of maximal discs of the $(3,4)$-Distance Transform, except for those whose removal is indispensable to allow the skeleton to be a unit wide set. On the skeleton, the pixels can be classified into *end points*, *normal points* and *branch points*, by taking into account the number of components of neighbors not belonging to the skeleton. This method does not require the iterated application of topology preserving removal operations, and does not need checking a condition specifically tailored to end point detection, since end points are automatically identified when the maximal centers are found. Skeletal pixels found on the distance transform can be classified as "parallelwise detectable" and "sequentially detectable" skeletal pixels. Parallelwise detectable pixels can be directly identified by the distance transform due to the structure of their neighbors. This is the case of the maximal discs. Sequentially detectable skeletal pixels can be found only after some of their neighbors with smaller labels have been identified and marked as skeletal pixels. Sequentially detectable pixels are necessary to link to each other the components of parallelwise detectable skeletal pixels. After detecting the skeletal pixels, a raster scan inspection is done to reduce to unit width and to fill the holes.

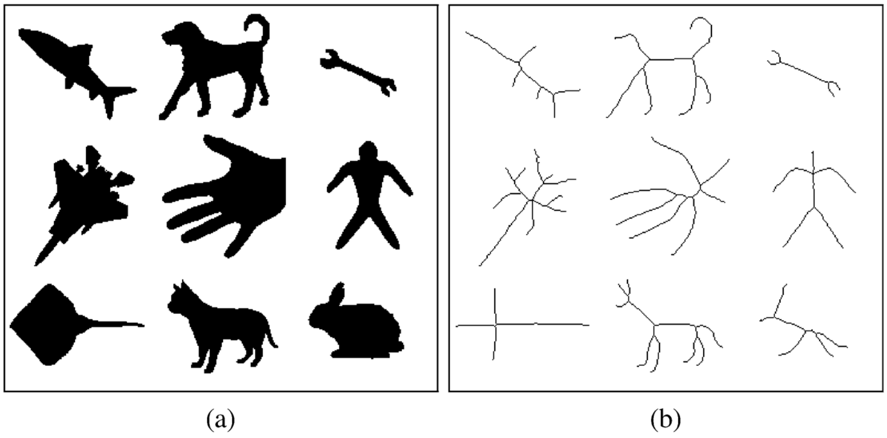


Fig. 4.2 (3, 4)-Distance-based skeletonization. (**a**) Original images; (**b**) obtained skeletons

Then a pruning and beautification step is proposed to erase some non-significant pixels of the skeleton. An example of the obtained skeletons is shown in Fig. 4.2. The computational cost of this method is modest and independent of the thickness of the pattern to be skeletonized.

4.2.3 Polygonal Approximation

The last step of the raster-to-vector conversion is to approximate skeleton images by segments. Rosin and West [10] proposed a method of segmenting curves in images into a combination of circular arcs and lines. The method is an extension of the algorithm proposed by Lowe in [7]. Lowe's algorithm segments each curve by recursively splitting it at the maximum deviation from the approximating straight line. At each level a decision is made depending on whether the single straight line is better than the representation at a lower level consisting of two or more approximating straight lines. The measure of goodness of fit is termed the "significance" and is defined as the ratio of the maximum deviation from the straight line to the length of the straight line.

The attractive property of this algorithm is that no thresholds are used to control the accuracy of the resulting representation. This is controlled by the significance values that can be regarded as the error between the curve and the straight line description weighted by the length of the straight line. Thus long straight lines are regarded as being a better representation even though the error can be greater. This introduces scale invariance such that a contour of different scales will have the same or similar description.

A list L_{ij} of skeleton pixels is hypothesized as being a straight line passing through its end points P_i and P_j, the point P_n at the point of maximum deviation d_{ij} corresponding to the straight line segments of the list L_{ij} divides it into two

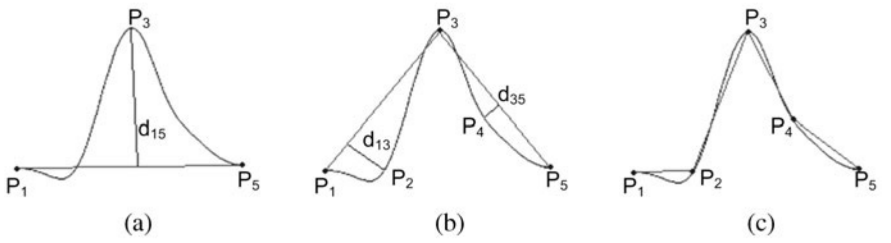


Fig. 4.3 Three levels of the straight line approximation. (**a**) First iteration; (**b**) second iteration; (**c**) third iteration

lists L_{in} and L_{nj}, and the process is repeated recursively on each of the two lists. The recursive process is stopped when a line segment is smaller than four pixels long or the deviation is less than three pixels. The result of the recursive process is a multilevel tree where the description of the list of skeleton pixels at each level is a finer approximation of the level above. The tree is then traversed back up to the root. At each level, if any of the line segments passed up from the lower level are more significant than the line segment at the current level, they are retained and passed up to the next higher level as candidate line segments. If this is not the case, the line segment at the current level is passed up. In Fig. 4.3, we can see an example of the first steps of the algorithm in approximating a curve by a set of straight lines.

The significance measure is the ratio of the maximum deviation divided by the line segment length. Thus, the lower the significance value, the more significant the line. The procedure is weighted in favor of long line segments. The longer the line is, the greater the deviations that will be tolerated. This algorithm produces a high quality, general purpose polygonal approximation. No arbitrary error threshold is required. Instead, the most appropriate values are chosen dynamically throughout the procedure.

4.3 A Vectorial Signature for Symbol Description

Starting from a vectorial representation of the documents, in this section we propose a model of vectorial signature to describe symbols in terms of geometric constraints. In order to have a powerful representation of the vectorial symbols to easily compute the signatures, an attributed graph is used. The nodes of the attributed graph represent the segments of the symbol and graph edges represent spatial relationships between segments. Let us formally define a graph G in the next subsection.

4.3.1 Representing Symbols by Attributed Graphs

Starting from a vectorial representation of the symbols from a line-drawing, we represent these symbols with a graph G defined as follows:

Definition 4.1 An ***attributed graph*** is denoted as $G = (V, E, \mu, \nu)$ where V is the set of nodes representing the segments of the symbols, and E is the set of edges representing the spatial relationships among them. A ***sub-graph*** of G containing the nodes $s_i, \ldots, s_j$ is denoted as $G_{\{s_i,\ldots,s_j\}}$. Σ_V and Σ_E are sets of ***symbolic labels***, and the functions $\mu : V \rightarrow \Sigma_V$ and $\nu : E \rightarrow \Sigma_E$ assign a label to each node and each edge. $\Sigma_V = [\theta_{s_i}, \rho_{s_i}]$ contains the information of each segment s_i according to a polar representation. $\Sigma_E = \{L, T, P, 1, 0\}$ represents the different kind of spatial relationships between a pair of segments. The possible relationships between segments are:

1. L represents a straight angle between a pair of adjacent segments.
2. T represents a straight angle between a pair of non-adjacent segments.
3. P represents two parallel segments.
4. 1 represents two adjacent segments.
5. 0 represents a non-expressive relation between two segments.

We define a signature in terms of a hierarchical decomposition of symbols. Following the idea presented by Huang in [5], a symbol can be described by the number of occurrences of particular sub-shapes. In our proposal, these expressive sub-shapes are extracted from the analysis of the adjacency matrix. Following a combinatorial approach on the number of sub-graph nodes, sub-shapes such as squares, triangles, parallelisms, etc. are taken into account. In Fig. 4.4, we show a graph[2] of a simple symbol. In the next subsection, we will see how the signatures are built from the analysis of the adjacency matrix.

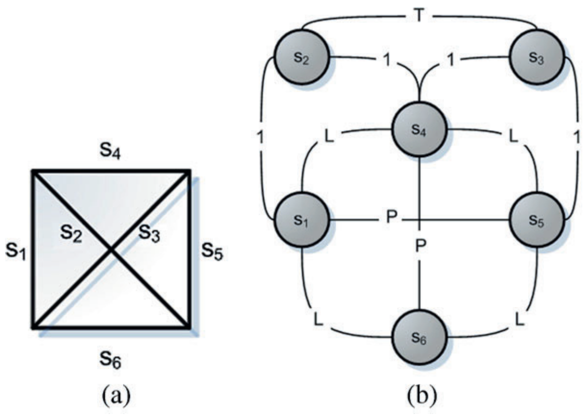


Fig. 4.4 Attributed graph representation of graphical symbols. (**a**) Graphical symbol; (**b**) its graph representation

[2]The edges labeled by 0 are not shown.

4.3.2 Building the Vectorial Signature

Starting from the analysis of the adjacency matrix, we propose a combinatorial approach to extract particular sub-shapes which comprise a symbol. For all the segments, all the sub-graphs formed by at least two nodes and a maximum of four nodes are analyzed to search for some representative shapes. If n is the number of segments, (4.1) gives the number of sub-graphs to analyze.

$$\#G_{\{\dots\}} = \sum_{k=2}^{4} C_n^k = \sum_{k=2}^{4} \frac{n!}{(n-k)! \times k!}. \tag{4.1}$$

For each sub-graph, we work with its adjacency matrix. The matrix M_G is, in fact, only computed once for all the segments, and when we want to focus on a sub-graph, a group of rows and columns of this matrix is selected. Notice that in most cases the relations between segments could be extracted in the vectorization process. In the extraction of these constraints, each comparison has an associated threshold value in order to be more tolerant. For the simple shape of Fig. 4.4, we can see its corresponding adjacency matrix M_G below in (4.2)

$$M_G = \begin{pmatrix} s_1 & 1 & 0 & L & P & L \\ 1 & s_2 & T & 1 & 0 & 0 \\ 0 & T & s_3 & 1 & 1 & 0 \\ L & 1 & 1 & s_4 & L & P \\ P & 0 & 1 & L & s_5 & L \\ L & 0 & 0 & P & L & s_6 \end{pmatrix}. \tag{4.2}$$

From this matrix, we examine all the possible combinations of sub-matrices taking four, three and two of the six possible nodes. Hence three different levels are considered. For all the sub-matrices representing the sub-graphs, normally the analysis of one single row can determine the shape that it encodes.

The vectorial signature of a symbol is then defined as a 40-dimensional vector V_S where in each position we have the number of occurrences of a particular geometric configuration of segments forming a given sub-shape. We have defined a set of 30 geometric configurations among different number of segments which can be efficiently extracted by analyzing the adjacency matrix. We can see some examples the sub-shapes taken into account to build the signature in Table 4.1. In order to

Table 4.1 Examples of sub-shapes composing the vectorial signature

Level	Considered sub-shapes	Level	Considered sub-shapes
4 nodes		3 nodes	
4 nodes		3 nodes	
4 nodes		2 nodes	
3 nodes		2 nodes	
3 nodes		2 nodes	

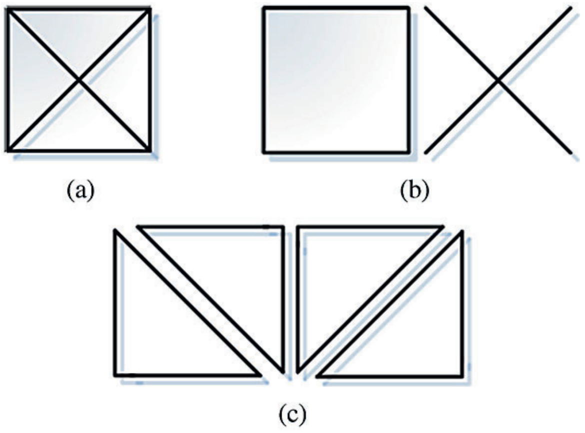


Fig. 4.5 Sub-shapes extracted from a symbol at different levels. (**a**) Original symbol; (**b**) symbol detached at level four and at level two; (**c**) symbol detached at level three

provide a more accurate description of the symbols, some additional information as the length-ratios and the distance-ratios are added at the last 10 positions of the signature. Since these measure features can take values from 0 to 1, this space is split into five bins where the occurrences of these geometric ratios among segments are accumulated.

It may seem redundant to store the information for multiple levels of the subgraph, since if in the level of four nodes we find a square, it is obvious that we will find two parallelisms and four straight angles at the level of two nodes. But this redundancy helps to detach completely all the multiple shapes in the drawing. For instance, a square with a cross inside can be seen as a square and a straight angle, or it can be seen as a set of triangles (see an example in Fig. 4.5). This redundancy helps to be more error tolerant and to store all the geometric configurations of all the multiples sub-shapes of the drawing. The occurrences of each sub-shape are accumulated to build the vectorial signature.

Once the signatures of the model symbols are extracted, in the querying step we can compare the obtained signature with the model signatures and associate a confidence value to each correspondence between the original symbol and all the model symbols depending on a distance function. The used distance function is the χ^2 distance computed as follows:

$$\chi^2(i,j) = \frac{1}{2}\sum_{k=1}^{K} \frac{(V_i[k] - V_j[k])^2}{V_i[k] + V_j[k]}, \tag{4.3}$$

where V_i and V_j are the vectorial signatures of two symbols i and j, and K the length of the vectorial signature (40 in our experimental setup).

However, as we commented in the introduction Sect. 4.1, the approaches based on vectorial signatures have the drawback that they cannot be straightforwardly used to locate a given symbol within a complete document. The spotting approaches based on vectorial signatures need a prior segmentation stage. Inspired by the work on symbol discrimination in complete documents presented by Dosch and Lladós in [3], in the next section we present a window-based strategy to compute the vectorial signatures within documents.

4.4 Sequential Access to Signatures: Defining Regions of Interest

When working with complete drawings, the usual approach is to divide the drawing into windows of fixed size which frame every symbol. In each zone of interest, a signature is computed and compared against the set of model signatures. However, these approaches lack of flexibility and may be quite sensitive to the scale of the documents. We propose to use a more dynamic approach, where the windows are built depending on the original line-drawing.

Regions of interest are computed from the maximum and minimum coordinates of several adjacent segments. So, the size of the regions of interest is variable. Also, a first filter of area and aspect-ratio can be easily implemented in order to delete some non-relevant symbols such as the walls in the architectural field or the wiring connections in electronic diagrams. Formally, for each node n_{s_i} of G (segment in the drawing) we build a list of all the nodes connected to n_{s_i} by an edge. Having a list of all the endpoints of the adjacent segments to a reference segment, we get the maximum and minimum coordinates of the endpoints that will construct a framing window of these segments. We can see an example of how to build the regions of interest in Fig. 4.6. As in most cases of technical drawings, the symbols have a low eccentricity, their bounding-box are square-shaped and such windows frame them. But, as the windows are based on the connections of the segments, the efficiency decreases if the symbols are disconnected or overlapped.

Moreover, in the vectorization step, more problems may happen: small vectors can appear due to noise, straight lines can be split into several collinear vectors, the arcs might be approximated by polylines, some neighboring lines in the drawings may be not adjacent in the vectorial representation because of gaps, dashed lines might appear as a set of small segments instead of one unique instance, etc.

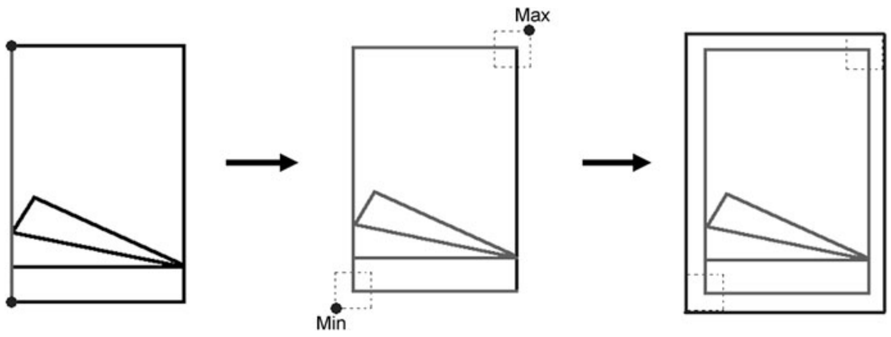


Fig. 4.6 Computing a region of interest from a reference segment. In the second step, all the adjacent segments to the reference segment are considered. A bounding-box is obtained from the minimum and maximum coordinates

To solve such problems, the best results are reached when we work with a lower resolution of the drawing to calculate the windows. This sub-sampling step reduces local distortions in the vectorial representation while preserving the most salient geometrical properties.

First, a contraction of the normalized graph is done, merging the adjacent nodes having a lower distance than a threshold *thr*. This graph contraction by distance allows reducing the jaggness of some collinear straight segments. Then, applying (4.4) to each node coordinate, we get a lower resolution graph. With this representation of decreased resolution, the problems of the gaps or the split segments are solved. Every endpoint is sampled for each step of m, so the minor errors are corrected.

$$\begin{aligned} x &= m \times round\left(\frac{x}{m}\right), \\ y &= m \times round\left(\frac{y}{m}\right). \end{aligned} \tag{4.4}$$

Experimentally, in Fig. 4.7a, the graph has 154 nodes because a horizontal line has been split in the vectorization process. When the graph contraction by distance is done (with a threshold value $thr = 0.06$; see Fig. 4.7b), we get a graph with 52 nodes which lines are crooked due to the node contraction, and with the decreased resolution graph (with $m = 35$; see Fig. 4.7c) we have to deal with only 33 nodes. Finally, we can compare the resulting extracted windows where the vectorial signature is to be computed in Fig. 4.7d.

This change of resolution can cause some errors, for example, some lines which are almost horizontal or vertical can be represented with a very different slope. But these errors do not interfere with the obtained windows, since they continue to frame the symbols. Notice that these lowest resolution images will only be used to calculate the regions of interest, not for the spotting process. Since each segment proposes a region of interest, there is no problem if one of the segments of a symbol gives a mistaken window.

In the next section, we present the experimental results.

4.5 Experimental Results

Our experimental framework consist of two different scenarios. First, we test the performance of the vectorial signatures to recognize and classify isolated symbols. Second, we have used the method for symbol spotting in a small set of real architectural drawings, and we will show some qualitative results.

The first tests were done using the GREC-SEG database[3] which is detailed in Appendix A. This database contains a selection of symbols from the GREC2005 database which does not contain arcs. For each model symbol, we have applied

[3]The GREC-SEG database is available at http://www.cvc.uab.cat/~marcal/GREC-SEG/.

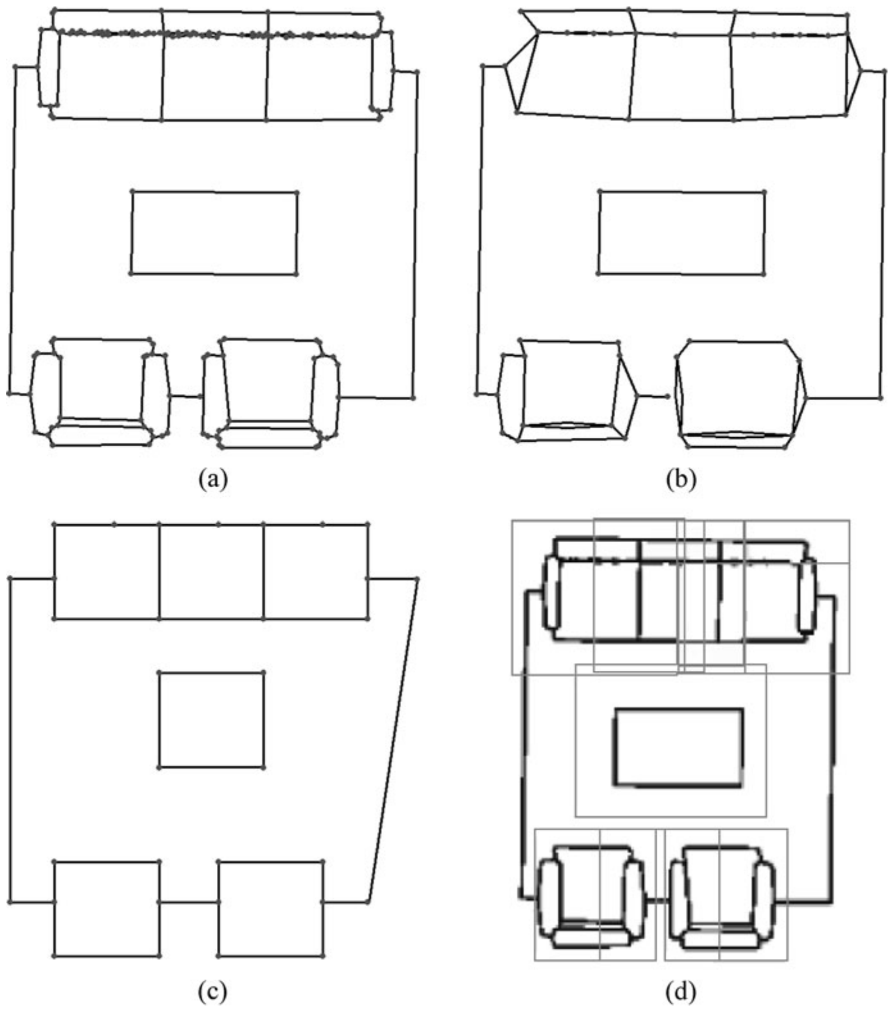


Fig. 4.7 Obtaining regions of interest from a low-resolution representation of line-drawings. (**a**) Original drawing; (**b**) graph contraction by distance; (**c**) low resolution representation; (**d**) obtained windows in the document image

three levels of synthetic distortion, and 20 instances at each level have been generated. The symbols are represented by a graph where the nodes represent the segments endpoints. Each node from the graph is randomly shifted within a predefined radius r. As the symbols are represented by a graph, the connectivity is not lost. We can see an example of these distortions in Fig. 4.8.

We can see the recognition results in Tables 4.2 and 4.3 expressed in terms of the True Positive Rate (*TPR*). We can see that the method yields good results when applying a low degradation of symbols. Most of the symbols of of the GREC-SEG database are "square-shaped", and the computed signatures are discriminative

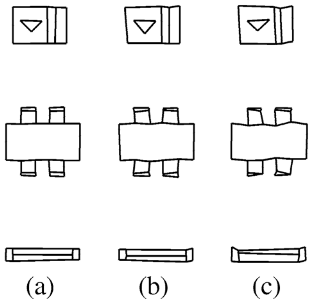


Fig. 4.8 Example of synthetical distortion from the GREC-SEG dataset. (**a**) $r = 5$; (**b**) $r = 10$; (**c**) $r = 15$

Table 4.2 Results of the recognition of GREC-SEG database (1)

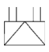
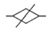
Symbol		*TPR* (%)				Symbol		*TPR* (%)			
		$r = 5$	$r = 10$	$r = 15$	Total			$r = 5$	$r = 10$	$r = 15$	Total
001		100	100	100	100	002		100	100	100	100
003		95	70	45	70	005		100	100	100	100
007		100	100	95	98.3	008		100	100	100	100
011		100	100	100	100	012		100	100	100	100
013		100	100	70	90	014		90	60	25	58.3
015		100	100	25	75	018		100	95	60	85
020		100	100	80	93.3	023		100	100	95	98.3
027		100	100	100	100	028		100	85	70	85
029		100	90	65	85	030		100	100	100	100
031		100	100	85	95	032		100	100	60	86.6
033		100	95	70	88.3	034		100	100	100	100
037		100	100	100	100	041		100	100	100	100
042		100	100	100	100	043		100	100	100	100
044		100	85	40	75	045		100	100	100	100
048		100	100	100	100	051		100	100	100	100
052		100	100	100	100	053		100	100	60	86.6
054		100	85	55	80	055		100	100	100	100
057		100	100	100	100	058		100	100	95	98.3
059		100	100	100	100	060		100	100	100	100
062		100	75	50	75	063		100	100	100	100

Table 4.3 Results of the recognition of GREC-SEG database (2)

Symbol		*TPR* (%)				Symbol		*TPR* (%)			
		$r = 5$	$r = 10$	$r = 15$	Total			$r = 5$	$r = 10$	$r = 15$	Total
065		100	100	70	90	068		100	100	85	95
069		100	100	100	100	072		100	100	100	100
074		100	100	100	100	078		100	90	40	76.6
079		100	100	100	100	084		50	10	10	23.3
085		100	100	100	100	088		100	100	75	91.6
091		100	100	100	100	093		100	100	100	100
094		40	50	80	56.6	098		100	80	60	80
104		100	100	100	100	106		100	100	100	100
107		100	100	100	100	108		100	100	100	100
110		100	100	100	100	111		100	85	60	81.6
113		100	90	40	76.6	114		100	65	30	65
115		100	100	100	100	120		100	100	100	100
121		100	100	100	100	126		100	100	100	100
127		100	100	100	100	128		100	100	100	100
130		100	85	30	71.6	132		100	100	100	100
133		100	100	100	100	134		100	100	95	98.3
136		100	100	90	96.6	137		100	100	95	98.3
138		100	100	100	100	143		100	100	100	100
144		100	100	85	95	145		100	95	95	96.6
147		100	100	100	100	Total		97.79	92.58	79.25	91.18

enough. But, when a symbol is composed of tiny little segments, and not very connected, the results are worse. Let us analyze some problematic symbols. Symbols 014 and 015 are very thin, but composed of an expressive sub-shape which is really discriminative, a square. These two symbols give good results when applying low degradation, 90% and 100% of recognition, respectively, but their recognition rate falls to 25% in both cases when applying a huge geometric deformation. As these symbols are composed of very short segments, a higher deformation distorts the little segments too much and damages the performance. On the other hand, symbols having segments which are not very connected between them also give bad results, for example, Symbols 003, 054, 114, and 130. As the deformation model guarantees the connectivity between segments, deforming the graph representation

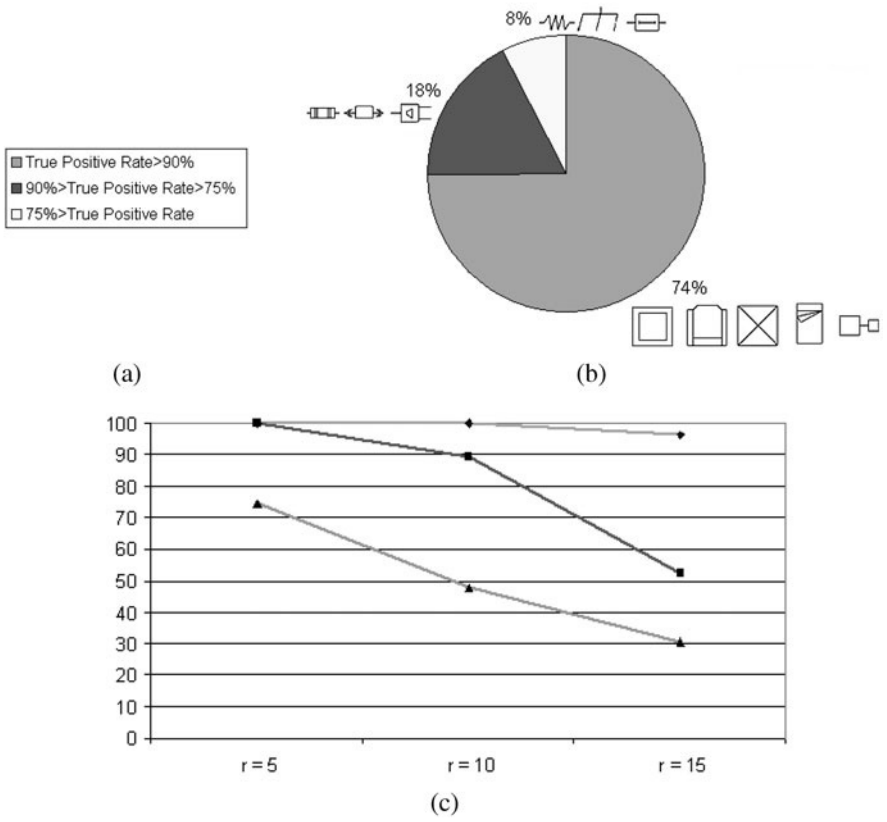


Fig. 4.9 Average true positive rates for three different symbol categories. (**a**) Three different symbol categories depending on their average recognition rate; (**b**) its distribution on the database; (**c**) average recognition rates per different symbol degradation levels

of the symbol and not the symbol itself, when the symbol is composed of non-connected segments, these segments are more affected by the deformation than the connected segments that, in some way, share the deformation between them. Notice that the recognition performance of Symbol 094 evolves inversely to the expected way. Symbol 094 does not have very expressive sub-shapes, only parallelisms and adjacency can be found. As vectorial signatures encode the presence of salient geometric features, when a symbol is composed of few sub-shapes, the recognition performance is very low. When applying a higher geometric noise, most of these sub-shapes are not preserved, but as the model of distortion guarantees the connectivity, this symbol is recognized better at high distortion levels than at the lower ones.

Finally, in order to have an idea of whether the symbol design can affect the performance of the recognition abilities of the signature model and to test if there are some symbol designs which are more sensitive to distortions, we present some indicators on the recognition performance in Fig. 4.9. We classified the symbols in the GREC-SEG database into three different classes dependent on their average true positive rate. We can see that 74% of the symbols of this database attain an average true positive rate greater than 90%. A second symbol family can be identified. 18% of the symbols in the database have an average true positive rate ranging between 75% and 90%. Finally, 8% of the symbols have an average true positive rate below the 75%. As we can observe in the example, the symbols in this last group are formed by less discriminative sub-shapes, and thus, the description ability of the signature is severely damaged. We can compare the different tolerance to the distortions for all the three classes of symbols and how simpler symbol designs are mode affected when we increase the synthetic deformations in Fig. 4.9c.

In the second test, we tried out the vectorial signatures with real architectural drawings by using the windowing approach presented in Sect. 4.4. Using more relaxed threshold values than when we are working with the database of isolated symbols, the symbols appearing within a complete document can be spotted. As we can see in Fig. 4.10, some false positives appear (thick squares), and some symbols are still missed. False positives appear when a window does not correctly frame a symbol. The stairs which consist of a lot of segments give a lot of regions of interest where false positives appear, and the wrong segmentation of the tables means that in the part where the chairs are drawn, a sofa is spotted, because their representation is very close. When we use vectorial signatures in real drawings, there are two factors that may cause the spotting results not to be so good. First of all, the symbols can be adjacent between them or to a wall, or the region of interest could not perfectly frame the symbol, in this case we face up to occlusions and additions of segments which distort the signature too much. On the other hand, in real drawings, the symbol design may be different of the learned model, so the learned features of a symbol could not appear in real drawings; in this case, it is obvious that the symbol cannot be spotted, a semantic organization of different design instances for any symbol is necessary.

4.6 Conclusions and Discussion

In this chapter, we have presented a vectorial signature model which is able to describe graphical symbols in terms of the occurrences of certain spatial configurations of segments. Since signatures are compact and yet effective symbol descriptors, they are very suitable to be used as the basis for a spotting approach. We have also presented a window-based segmentation system which uses the vectorial signatures to spot symbols appearing within complete documents.

We can see that the symbol discrimination using vectorial signatures yields good results when we are working with the database of pre-segmented symbols and with symbols with synthetical distortion, which is a controlled framework. In real scanned architectural drawings, even if the symbols are usually well spotted, a lot of false positives appear. However, since the objective of spotting techniques is not to recognize the symbol but in some way to index the drawing, the false positives problem is not so significant.

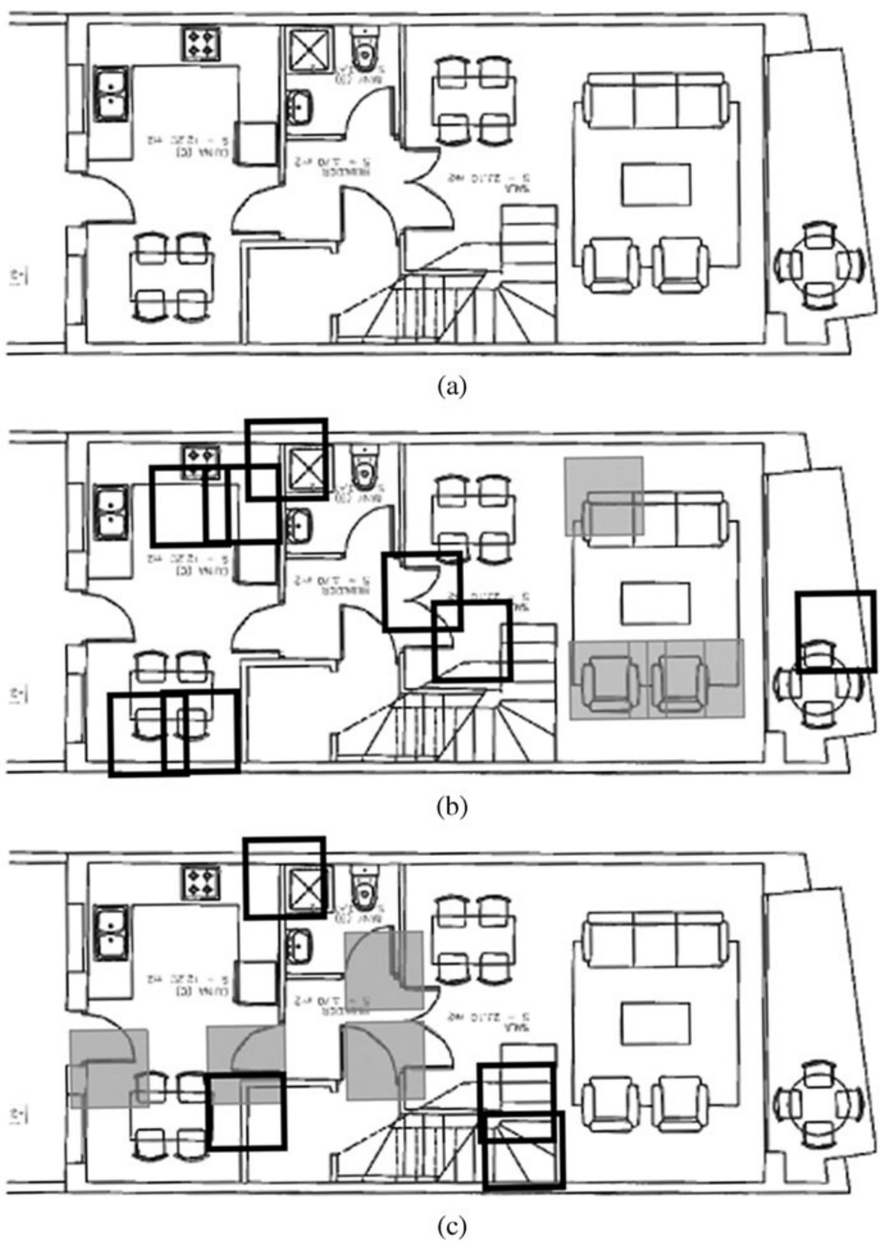


Fig. 4.10 Qualitative results of spotting symbols by using vectorial signatures. (**a**) Original drawing; (**b**) spotting the sofa symbol; (**c**) spotting the door symbol

4.6.1 Limitations of the Vectorial Signatures

Even if the recognition performance of the signatures attains good levels, the proposed method presents some important drawbacks to be used as a spotting method for indexing a document collection and be used by a focused retrieval application. Let us enumerate the most important drawbacks and let us see how we can solve these problems in the next two chapters.

Since the presence of the sub-shapes is determined from the analysis of the adjacency matrix, the method is not tolerant at all to segment fragmentation. If the number of segments comprising a symbol does not correspond to the number of segments of the learned model, the signature can be severely damaged. In addition, the connectivity of adjacent segments also must be guaranteed in order to achieve an acceptable performance. Even if these effects do not occur on synthetic data such as the one of the GREC symbol recognition competitions, these two phenomena occur frequently in real document images treated with a raster-to-vector conversion. A more tolerant set of features to base our signature on must be found to tolerate this kind of errors. We introduce the notion of polylines instead of segments in the next chapter. We will also see how we can represent the expressive sub-shapes defined above at the polyline level.

Moreover, the extracted sub-shapes from the adjacency matrix comprising the vectorial signature were ad-hoc defined to describe a particular family of graphical symbols. We saw in the experimental results section that the recognition performance of the signature model was dependent on the symbol design. However, spotting approaches should be more scalable in terms of the nature of the documents, and not be conceived for a particular collection. In the next chapters, we propose using more flexible description approaches which should perform similarly no matter which nature the line-drawing is of.

In addition, the proposed signature model just captures geometric configurations formed by straight segments. It is not able to deal with circular primitives such as arcs, circles, ellipses, etc. This fact is a strong limitation since only symbols formed by lines can be considered. The methods proposed in the next chapters can deal with any polygonal shape.

Finally, the presented window-based system can be useful in applications with a predefined set of model symbols. Given an input line-drawing, regions of interest are extracted by the windowing approach, and the signatures are sequentially computed and matched against the model database. However, these sequential approaches are hardly useful when facing focused retrieval applications. In these cases, we can have large collections of documents, and the user can query any symbol. We shall provide a more efficient access to the descriptors by using indexing data structures. In the next chapters, we propose using particular data structures having graphical patterns as indices for accessing and navigating large collections of documents and being able to use spotting methods as the basis of a focused retrieval application.

References

1. Baja, G.D.: Well-shaped, stable, and reversible skeletons from the (3, 4)-distance transform. Journal of Visual Communications and Image Representation **5**(1), 107–115 (1994)
2. Coustaty, M., Guillas, S., Visani, M., Bertet, K., Ogier, J.: On the joint use of a structural signature and a Galois lattice classifier for symbol recognition. In: Graphics Recognition. Recent Advances and New Opportunities, *Lecture Notes on Computer Science, pagrs*, vol. 5046, pp. 61–70. Springer, Berlin (2008)
3. 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)
4. 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)
5. Huang, P.: Indexing pictures by key objects for large-scale image databases. Pattern Recognition **30**(7), 1229–1237 (1997)
6. Loce, R., Dougherty, E.: Enhancement and restoration of digital documents: Statistical design of nonlinear algorithms. Society of Photo-Optical Instrumentation Engineers (SPIE), Bellingham, USA (1997)
7. Lowe, D.: Three-dimensional object recognition from single two-dimensional images. Artificial Intelligence **31**(3), 355–395 (1987)
8. Ntirogiannis, K., Gatos, B., Pratikakis, I.: An objective evaluation methodology for document image binarization techniques. In: Proceedings of The Eighth IAPR International Workshop on Document Analysis Systems, pp. 217–224. IEEE Computer Society, Los Alamitos (2008)
9. Qureshi, R., Ramel, J., Cardot, H., Mukherji, P.: Combination of symbolic and statistical features for symbols recognition. In: Proceedings of the International Conference on Signal Processing, Communications and Networking, pp. 477–482. IEEE Computer Society, Los Alamitos (2007)
10. Rosin, P., West, G.: Segmentation of edges into lines and arcs. Image and Vision Computing **7**(2), 109–114 (1989)
11. Tombre, K., Ah-Soon, C., Dosch, P., Masini, G., Tabbone, S.: Stable and robust vectorization: How to make the right choices. In: Graphics Recognition Recent Advances, *Lecture Notes on Computer Science*, vol. 1941, pp. 3–18. Springer, Berlin (2000)
12. Trier, O., Taxt, T.: Improvement of "integrated function algorithm" for binarization of document images. Pattern Recognition Letters **16**(3), 277–283 (1995)
13. 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)
14. White, J., Roher, G.: Image thresholding for optical character recognition and other applications requiring character image extraction. IBM Journal of Research and Development **27**(4), 400–411 (1983)
15. 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)

Chapter 5
Symbol Spotting Through Prototype-based Search

Abstract In this chapter, we present a method to determine which symbols are probable to be found in technical drawings by the use of a prototype-based search. First, symbols are decomposed into primitives representing closed regions. These primitives are then encoded in terms of attributed strings. Second, the strings are organized in a lookup table so that the set median strings act as representative prototypes of the clusters of similar primitives. This indexing data structure aims at efficiently retrieving the locations from the document collection where similar primitives as the queried ones can be found. Finally, a voting scheme formulates hypotheses in the locations of the line drawing image where there is a high presence of regions similar to the queried ones, and therefore, a high probability to find the queried graphical symbol. The proposed approach has been proved to work even in the presence of noise and distortion introduced by the scanning and raster-to-vector processes.

5.1 Introduction and Related Work

The vectorial signature model presented in the last chapter is only able to discriminate symbols if the query symbol has the same number of segments that the symbol present in the collection. Even if vectorial signatures yield good results when recognizing isolated symbols they present important weaknesses when trying to apply them in a focused retrieval application dealing with a collection of real vectorized line-drawings. Basically, one of the main problems to face is the noise and the segment fragmentation introduced by the raster-to-vector conversion process. In addition, the raster-to-vector algorithms used in this book do not detect arcs, but approximate them by a set of adjacent segments. In the literature, the interested reader can find some works concerning the arc detection for vectorization algorithms such as, for instance, in [4, 20]. In the previous chapter, symbols formed by arcs or circles were not considered because the vectorization algorithm was unable to detect them. In order to enhance the robustness of the spotting method, we propose introducing a polyline approximation in the raster-to-vector algorithm as a post-processing step. Let us formally define the term polyline.

Definition 5.1 Given a segment $s_i = (x_1, y_1), (x_2, y_2)$, the ***adjacency*** $A(s_i) = (p, k)$ of s_i is defined as the number of segments incident with (x_1, y_1) for p and incident with (x_2, y_2) for k.

Definition 5.2 Let $s_1 \dots s_i \dots s_n$ be a set of sorted and adjacent segments where

$$\begin{aligned} A(s_1) &= (p, 1) \quad \text{with } p \neq 1, \\ A(s_i) &= (1, 1) \quad \text{with } i \in \{2, \dots, n-1\}, \\ A(s_n) &= (1, k) \quad \text{with } k \neq 1. \end{aligned}$$

The ***polyline*** $P_{s_1 \dots s_i \dots s_n}$ is the geometric shape considering the set of adjacent segments $s_1 \dots s_i \dots s_n$ as a unique instance. A polyline starts and ends at the points were a segment s_n finishes and no other segment is adjacent to s_n, or at the points where more than two segments are adjacent.

When the segment approximation is done in the raster-to-vector step, a grouping of the adjacent segments is implemented. All the segments having $A = (1, 1)$ form part of a polyline. Segments having $A = (p, 1)$ or $A = (1, k)$ with $p, k \neq 1$ are the end-segments of a polyline. We can see an example of the polyline decomposition in Fig. 5.1.

The main idea of treating multiple segments as a unique instance is that we should not have the restriction of the number of segments forming a symbol. Since we obtain more tolerance to the noise arising from the raster-to-vector conversion step, the jaggness problem is not critical. The underlying problem is how to describe these polylines in terms of geometric constraints. Since polylines are ordered sets of adjacent segments, a polyline can be seen as a one-dimensional chain. In the literature, we can find several works describing shapes by chains of adjacent segments. Let us briefly review a few.

Stein and Medioni [15, 16] proposed describing objects by a polygonal approximation of their contours. The segments arising from the raster-to-vector conversion are then grouped into sets of adjacent segments named super-segments. These chains of consecutive segments are then described by a feature vector of geometrical attributes as the lengths or the angles of the segments, and the global orientation and

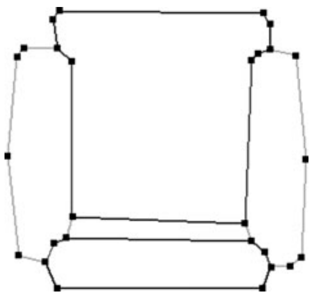


Fig. 5.1 Polyline decomposition of a vectorized symbol. The vectorized symbol has 31 segments which are grouped into 12 polyline instances (displayed in different colors)

eccentricity of the super-segment. In order to be tolerant to changes in the number of segments composing a super-segment, the authors propose working at multiple cardinality scales. However, considering this cardinality scale implies an important grow of the number of feature vectors describing a shape, thus exponentially increasing the time of computing distances among shapes.

In order to provide more tolerance to changes in the number of segments composing a polyline, we can find some works in the literature which, starting from an attributed string representation of the shape, use string edit operations to compute the distance between two strings representing two shapes. As examples, we can, for instance, cite the work of Wolfson presented in [21], the methods of Bunke and Bühler [3], or the more recent work of Kaygin and Bulut [7]. In all these works, polygonally approximated contours are represented by attributed strings. Similar shapes are matched depending on the costs of the operations needed to edit and transform one string into the other. In this chapter, we use this particular shape description and matching technique in order to be tolerant to changes in the number of segments composing a given polyline. Let us see which particular data structure we use in order to provide an efficient access to the stored descriptors.

One of the drawbacks of the method presented in the last chapter is that, in order to retrieve similar symbols, all the matchings between the query descriptors and the ones of the line-drawings have to be computed. In order to face a focused retrieval application with a large document collection, we shall provide efficient access to the descriptors by using indexing data structures. In this chapter, we propose using a lookup table indexing structure to retrieve primitives by similarity which drastically reduces the number of comparisons to be computed. The main idea is to provide what we call prototype-based search. In prototype-based search, we are given a set of distorted samples of the same primitive and want to infer a representative model. In this context, the median concept turns out to be very useful. The use of the string matching algorithm to compute a similarity measure also allows the computation of the set median strings. Given a set of similar strings representing vectorial primitives, a representative of this set having the smallest sum of distances to all the strings in the set can be computed. These set median strings act as indexing keys of a lookup table. When performing a query, the retrieval by similarity of primitives is done efficiently since only the distances between the query primitive and the representative of a cluster of similar polylines has to computed. Avoiding brute-force distance computation allows fast primitive retrieval by similarity. By the use of a lookup table together with a Hough-like voting scheme, in this chapter we propose a framework able to spot graphical symbols within a document image collection.

The remainder of this chapter is structured as follows. The next section reviews the string matching theory and algorithms, and provides details about our particular cost functions. Subsequently, in Sect. 5.3, we detail the proposed prototype-based indexing framework allowing to spot graphical symbols within a document collection. Section 5.4 presents the experimental results. Finally, the conclusions and a short discussion can be found in Sect. 5.5.

5.2 String Matching Theory and Algorithms

String edit distances were first defined by Wagner and Fischer in [19] to find the minimum cost edit sequence to convert the string A into the string B using edit operations. Although the origin of the algorithm is spelling correction, it has been used for different purposes, and particularly as an approach to the problem of recognizing and classifying polygons. The problem is to define dissimilarity measures between polygons, and to find algorithms that compute these measures fast enough. The string matching-based approaches should be independent of the scale, translation and rotation of the polygons under analysis. Let us review the string matching theory and algorithms, and subsequently provide the details about our particular cost functions to match polygons.

5.2.1 Definitions

Let us first introduce some basic notation and definitions of the basic string matching algorithm first proposed by Wagner and Fischer in [19].

Definition 5.3 Let Σ be a set of elements called ***symbols*** and let Σ^* denote the set consisting of all finite ***strings*** over Σ. The length $|A|$ of a string $A \in \Sigma^*$ is the number of symbols in A. And let Λ denote the null string which has length 0.

Definition 5.4 For a string $A = a_1 a_2 \dots a_n \in \Sigma^*$, a ***cyclic shift*** is a mapping $\sigma : \Sigma^* \to \Sigma^*$ defined by $\sigma(a_1 a_2 \dots a_n) = a_2 a_3 \dots a_n a_1$. For all $k \in \mathbb{N}$, let σ^k denote the composition of k cyclic shifts. Two strings A and $\overline{A}$ will be called equivalent if $A = \sigma^k(\overline{A})$.

Definition 5.5 An ***edit operation*** s is an ordered pair $(a, b) \neq (\Lambda, \Lambda)$ of strings, each of a length less than or equal to 1, denoted by $a \to b$. An edit operation $a \to b$ will be called an ***insert*** if $a = \Lambda$, a ***delete*** operation if $b = \Lambda$, and a ***substitution*** operation otherwise.

Definition 5.6 A string B ***results*** from a string A by the edit operation $s = (a \to b)$, denoted by $A \to B$ via s, if there are strings C and D such that $A = CaD$ and $B = CbD$. An ***edit sequence*** $S = s_1 s_2 \dots s_k$ is a sequence of edit operations. We say that S takes A to B if there are strings $A_0, A_1, \dots, A_k$ such that $A_0 = A$, $A_k = B$ and $A_{i-1} \to A_i$ via s_i for all $i \in \{1, 2, \dots, k\}$.

Definition 5.7 Let γ be a ***cost function*** that assigns a non-negative real number $\gamma(s)$ to each edit operation. For an edit sequence S, we define the cost $\gamma(S)$ as

$$\gamma(S) = \sum_{i=1}^{k} \gamma(s_i).$$

The ***edit distance*** $\delta(A, B)$ from string A to string B is then defined as

$$\delta(A, B) = \min\{\gamma(S)\},$$

and the ***edit distance*** $\delta([A], [B])$ of two cyclic strings $[A]$ and $[B]$ is given by

$$\delta\bigl([A], [B]\bigr) = \min\bigl\{\delta\bigl(\sigma^k(A), \sigma^l(B)\bigr) | k, l \in \mathbb{N}\bigr\}.$$

Attributed string matching has been used for polygon matching in several applications. In order to avoid the segmentation inconsistencies due to the noisy images or distorted shapes, Tsay and Tsai [18] introduced two other edit operations.

Definition 5.8 The **split** operation is the result of splitting a symbol a_i into a sequence of k consecutive symbols, denoted as $a_i \to a_{i1}a_{i2}\dots a_{ik}$.

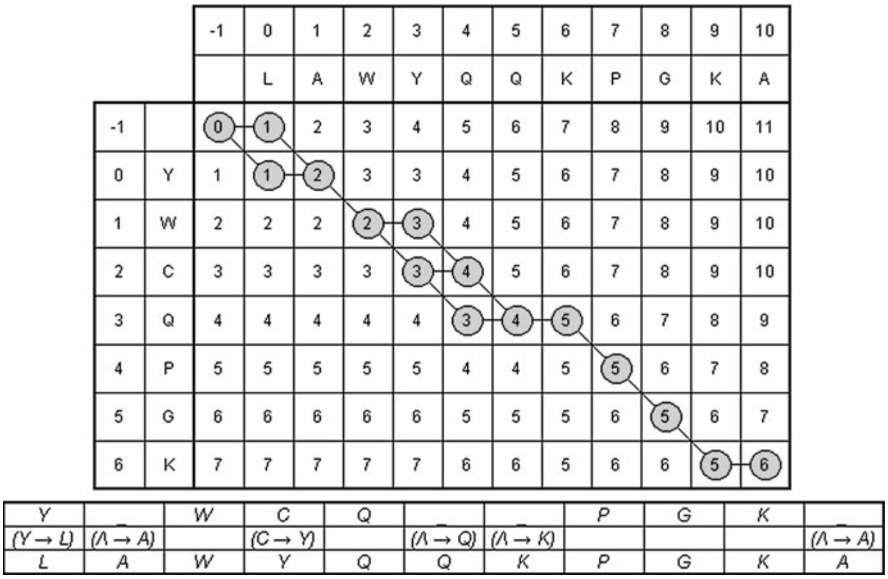
Definition 5.9 The **merge** operation is the result of merging k consecutive symbols into a symbol, denoted as $a_i a_{i+1}\dots a_{i+k-1} \to a'_i$.

5.2.2 Linear String Matching

Let A and B be two strings over Σ^* of length n and m, respectively. The Wagner and Fischer [19] algorithm takes $\mathcal{O}(nm)$ time to a find $\delta(A, B)$ by determining a minimum weighted path in a weighted directed graph. Let $D(i, j)$ denote the cost of a minimum weighted path from the vertex $v(0, 0)$ to the vertex $v(i, j)$, so $D(n, m) = \delta(A, B)$. We can see the details of linear string matching below in Algorithm 5.1.

Algorithm 5.1: Linear string matching algorithm

```
$D(0, 0) := 0;$
**for** $i := 1$ *to* $n$ **do**
    $D(i, 0) := D(i-1, 0) + \gamma(a_i \to \Lambda);$
**end**
**for** $j := 1$ *to* $m$ **do**
    $D(0, j) := D(0, j-1) + \gamma(\Lambda \to b_i);$
**end**
**for** $i := 1$ *to* $n$ **do**
    **for** $j := 1$ *to* $m$ **do**
        $D(i, j) := \begin{cases} D(i-1, j) + \gamma(a_i \to \Lambda) \\ D(i, j-1) + \gamma(\Lambda \to b_i) \\ D(i-j, j-i) + \gamma(a_i \to b_i) \end{cases}$
    **end**
**end**
```



		-1	0	1	2	3	4	5	6	7	8	9	10
			L	A	W	Y	Q	Q	K	P	G	K	A
-1		0	1	2	3	4	5	6	7	8	9	10	11
0	Y	1	1	2	3	3	4	5	6	7	8	9	10
1	W	2	2	2	2	3	4	5	6	7	8	9	10
2	C	3	3	3	3	3	4	5	6	7	8	9	10
3	Q	4	4	4	4	4	3	4	5	6	7	8	9
4	P	5	5	5	5	5	4	4	5	5	6	7	8
5	G	6	6	6	6	6	5	5	5	6	5	6	7
6	K	7	7	7	7	7	6	6	5	6	6	5	6

Y	_	W	C	Q	_	_	P	G	K	_
($Y \rightarrow L$)	($\Lambda \rightarrow A$)		($C \rightarrow Y$)		($\Lambda \rightarrow Q$)	($\Lambda \rightarrow K$)				($\Lambda \rightarrow A$)
L	A	W	Y	Q	Q	K	P	G	K	A

Fig. 5.2 Example of the string matching algorithm. Edit operations and obtained cost to transform the string "YWCQPGK" into the string "LAWYQQKPGKA"

For the split and merge step, a window $q \times q$ is needed. For each $D(i, j)$, the cost of split and merge is considered as the minimum cost of all the possibilities in a region starting at the vertex $v(i, j)$ and ending at $v(i - q, j - q)$. And the costs are computed as a sum of three costs. For example, in the merge case

$$\begin{aligned}&\gamma(a_k a_{k+1} \dots a_{k+p} \rightarrow b_l b_{l+1} \dots b_{l+t})\\&\quad = \gamma\big(a_k a_{k+1} \dots a_{k+p} \rightarrow a'\big) + \gamma\big(b_l b_{l+1} \dots b_{l+t} \rightarrow b'\big) + \gamma\big(a' \rightarrow b'\big).\end{aligned} \tag{5.1}$$

5.2.3 Cyclic String Matching

Linear string matching cannot tackle strings having cyclic shifts since the computed paths always start from a given initial symbol. A cyclic string matching procedure is needed in the case of cyclic strings.

Given two finite strings A and B, the cyclic string matching problem is the problem of determining $\delta([A], [B])$ and an edit sequence realizing this cost.

Let $BB = b_1 b_2 \dots b_m b_1 b_2 \dots b_m$ be the concatenation of B with itself. For all $l \in \{1, 2, \dots, m\}$, we can find a minimum cost edit sequence from A to $\sigma^l(B)$ by determining a minimum weighted path from $v(0, l)$ to $v(n, m + l)$.

Although the computation of only one path takes $\mathcal{O}(nm)$ time, the computation of all these paths can be done in $\mathcal{O}(nm \log m)$ time, since all the paths can be chosen such that two different paths never cross.

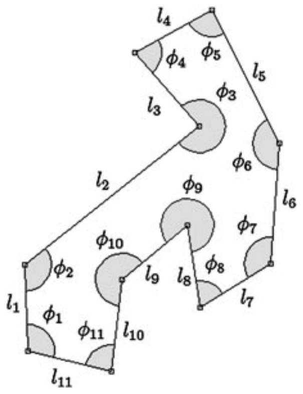


Fig. 5.3 Attributed representation of a chain of adjacent segments

5.2.4 A String Matching Cost Function for Polygon Recognition

Visually, two chains of segments are similar if the length attributes and angles between consecutive segments can be aligned. In the literature on polygonal shape recognition, most approaches base the distance definition between two polygonal shapes on length and angle differences. For example, Arkin et al. [1] used the turning function which gives the angle between the counterclockwise tangent and the x-axis as a function of the arc length. Their results are in accordance with the intuitive notion of shape similarity.

In order to use string matching for polygon recognition, we will use an attributed string matching. Starting from a polygonal approximation of the shape, we will use the segments as primitives, encoding them with a pair of numbers (l_i, ϕ_i), where l_i denotes the length of the segment s_i and ϕ_i denotes the angle between s_i and s_{i-1} in the counterclockwise direction. We can appreciate an example on how these attributes are computed for a sample shape in Fig. 5.3.

Let A and B be two chains of adjacent segments, represented as strings, with total lengths $|A| = n$ and $|B| = m$ and with respectively attributed string representations:

$$A = (l_1^A; \phi_1^A)\dots(l_n^A; \phi_n^A) \quad \text{and} \quad B = (l_1^B; \phi_1^B)\dots(l_m^B; \phi_m^B). \tag{5.2}$$

The costs functions for attributed string matching are as follows:

$$\begin{aligned}
&\gamma\big((l_i^A; \phi_i^A) \to (l_j^B; \phi_j^B)\big) = \frac{|\phi_i^A - \phi_j^B|}{360} + \left|\frac{l_i^A}{|A|} - \frac{l_j^B}{|B|}\right|,\\
&\gamma\big(\lambda \to (l_j^B; \phi_j^B)\big) = \frac{l_j^B}{|B|},\\
&\gamma\big((l_i^A; \phi_i^A) \to \lambda\big) = \frac{l_i^A}{|A|},\\
&\gamma\big((l_{i,j}^A; \phi_{i,j}^A) \to (l_u^A; \phi_u^A)\big) = \frac{(\sum_{k=i}^{j} l_k^A) - l_u^A}{\sum_{k=i}^{j} l_k^A},
\end{aligned} \tag{5.3}$$

which are the proposed cost functions inspired by those introduced by Tsay and Tsai in [18], where the authors use string matching for shape recognition. Maes [10] proposed using a weighting factor in the length costs to compensate undesirable cost bias for angle differences. However, in our experiments we did not observe any improvement in adding such a parameter. Finally, for the sake of simplicity, the previous operations are grouped by a block substitution using the merge operation. The total cost of substituting the whole sequence of symbols by another is computed as follows:

$$\gamma(A_{i,j} \rightarrow B_{k,l}) = \gamma(A_{i,j} \rightarrow u) + \gamma(B_{k,l} \rightarrow v) + \gamma(u \rightarrow v), \tag{5.4}$$

where u and v are the segments starting at the initial point of A_i and B_k and ending at the final point of A_j and B_l, respectively.

As all the length comparisons are weighted by the total perimeter of the chain of segments and the angles are computed relatively to the previous segment, the proposed string matching approach is rotation and translation invariant. In addition, the merge operation attributes low edit costs to primitives undergoing noisy transformations such as the inherent segment fragmentation from the raster-to-vector process and aims at comparing strings with different number of segments, making the system tolerant to segment cardinality and to scale changes.

5.3 Spotting Method

Given a technical document, the main idea of this chapter is to organize clusters of similar string primitives in a lookup table. Each entry of this table is indexed by a representative string allowing the use of a graphical pattern as a query and avoiding the computation of distances over all the stored primitives.

We divide the spotting method in four different parts. First, primitives are extracted from the symbol instances in terms of strings attributed with geometric constraints among their segments. Then, the off-line step builds the indexing data structure to organize the primitive descriptors. Third, the on-line process formulates a graphical query and the search in the data structure for similar primitives. Finally, the Hough-like voting scheme spots the zones of interest where there is a high probability to find the symbol. Let us further describe the above steps.

5.3.1 Symbol Representation in Terms of String Primitives

The first step to consider in any symbol recognition methods using geometric constraints as descriptors is the pre-processing step allowing to decompose into primitives the target documents as well as queries. Let us briefly explain these preprocessing steps of primitive extraction.

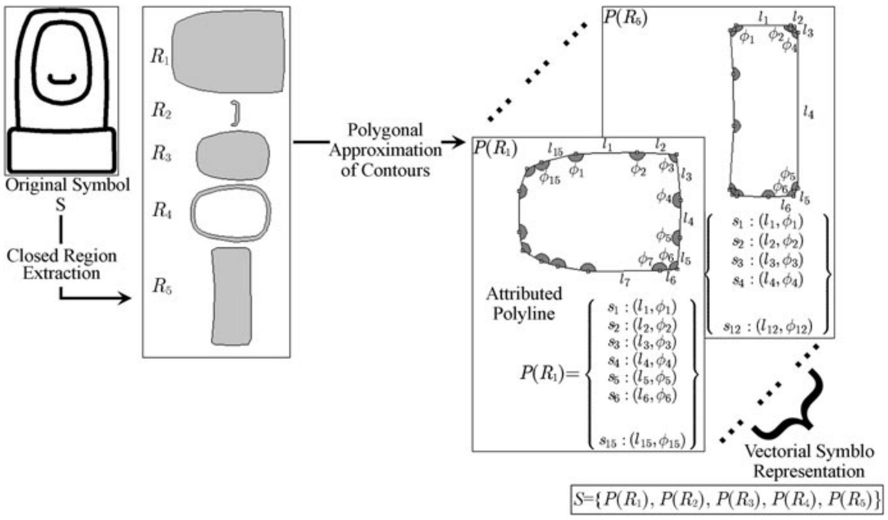

Original Symbol
S
Closed Region Extraction
R_1
R_2
R_3
R_4
R_5
Polygonal Approximation of Contours
$P(R_1)$
$P(R_5)$
Attributed Polyline
$P(R_1)=\{s_1:(l_1,\phi_1), s_2:(l_2,\phi_2), s_3:(l_3,\phi_3), s_4:(l_4,\phi_4), s_5:(l_5,\phi_5), s_6:(l_6,\phi_6), s_{15}:(l_{15},\phi_{15})\}$
$s_1:(l_1,\phi_1)$ $s_2:(l_2,\phi_2)$ $s_3:(l_3,\phi_3)$ $s_4:(l_4,\phi_4)$ $s_{12}:(l_{12},\phi_{12})$
Vectorial Symblo Representation
$S=\{P(R_1), P(R_2), P(R_3), P(R_4), P(R_5)\}$

Fig. 5.4 Symbol representation in terms of polygonal approximation of closed region contours. Each of these region contours is represented by strings attributed by length and angles

Here we use the same raster-to-vector algorithm implementation presented in the last chapter (see Sect. 4.2) with a slight modification. Rather than representing symbols with a polygonal approximation based on a skeletonisation, our choice for this chapter is focused on computing a closed region labeling and extraction based on a connected component analysis. The contours of these closed regions are then polygonally approximated using the Rosin and West's [13] algorithm.

After computing the polygonal approximation of the contours of the closed regions, an association of chains of adjacent segments resulting in a polyline is done. These polylines are encoded as attributed strings, and used as primitives to describe the symbol to be recognized.

Formally, let R be the contour of a closed region which is polygonally approximated and represented by the chain of adjacent segments $P(R) = \{s_1 \dots s_n\}$ consisting of n segments s_i. As we have seen in the previous Sect. 5.2.4, each segment s_i is attributed with the tuple (l_i, ϕ_i), where l_i denotes the length of the segment s_i and ϕ_i denotes the angle between s_i and the previous segment s_{i-1} in the counterclockwise direction. A symbol is then described in terms of p region contours comprising it and denoted as $S = \{P(R_1) \dots P(R_p)\}$. We can see a graphical example in Fig. 5.4.

The distance between two polylines is computed by using the string matching algorithm and the particular cost functions defined in the previous section. The final distance between two symbols is then computed as the sum of distances among the corresponding primitives. We will focus on the primitive description organization and the indexing structure construction in the next section.

5.3.2 Off-line Lookup Table Construction

In this section, we propose using an indexing structure which allows a fast primitive retrieval by similarity. The main idea is to cluster similar polylines into entries of a lookup table. The retrieval of primitives by similarity is done by a prototype-based search. Each entry of the lookup table is identified by a representative of a cluster of similar primitives. When querying this lookup table, a list of locations to find similar primitives is obtained without computing the distance between the query and all the primitive instances, but rather just by computing the distance between the query and the cluster prototypes. In order to correctly identify the lookup table entries, a representative of the clusters of primitives has to be computed.

Each lookup table entry representing a cluster of similar strings appearing in the document collection, consists of two different items: a representative polyline of each cluster which acts as an indexing key and the stored list of locations where we can find the polylines belonging to this cluster. If strings are used for representing the objects under consideration, then we are faced with the task of finding the median of a set of similar strings. Let us see how we can compute the representative string from a set of similar strings.

Generally speaking, the representative polyline of each cluster can be computed in two ways, namely the *mean string* or the *set median string*. As proposed in Sánchez et al. [14], the mean string M over a string cluster $C = \{A_1 \dots A_n\}$ is defined as

$$M = \underset{M \in \Sigma^*}{\arg\min} \left(\sum_{i=1}^{n} \delta(A_i, M) \right). \tag{5.5}$$

The mean string is computed as a new string that represents the average shape among all the strings in the set. The main drawback of this approach is its computational cost which increases with a large number of shapes. On the other hand, the set median string $\widetilde{M}$ is defined as the string in a given set minimizing the sum of distances to all the strings in the set. The set median string is defined as

$$\widetilde{M} = \underset{\widetilde{M} \in C}{\arg\min} \left(\sum_{i=1}^{n} \delta\bigl(A_i, \widetilde{M}\bigr) \right). \tag{5.6}$$

In our case, we have experimentally verified that a set median string is useful enough to be used as an index of a table entry. Besides, it is less expensive since we do not need to compute a new string which is an exact shape average of the set, but to select it between the strings composing the cluster. Let us see how, from the set median string formalism, we can build an indexing structure to retrieve by similarity stored primitive strings.

The lookup table is built as follows. For all the polylines $P(R_i)$ appearing in a document of the collection, we store their locations in the lookup table. In order to be able to query the indexing structure by similarity of graphical patterns, we should select a cluster of similar polylines which they belong to. The selection of this cluster is done by applying the string matching algorithm proposed above. We select the cluster where the cost of editing the string $P(R_i)$ to match the set median string $\widetilde{P_C}$ of the cluster is lower than a threshold *thr*. Once we identified the corresponding cluster, we add $P(R_i)$ to it. The set median string $\widetilde{P_C}$ of the corresponding cluster is recomputed in order to keep offering a good cluster representative. If no cluster has a set median string similar to $P(R_i)$, then we define a new cluster having $P(R_i)$ as representative. The set median strings act as indexing keys of the lookup table where at each entry a list of translation vectors $\overrightarrow{v_i} = (x_i, y_i, d_i)$ are stored. Here (x_i, y_i) are the coordinates of the middle point of the polyline, and d_i identifies the corresponding document in the collection where $P(R_i)$ appears. We can see the details in Algorithm 5.2:

Algorithm 5.2: Algorithm to build a LUT from a list of primitives

```
for $i = 0$ to $length(R)$ do
    if $LUT[P(R_i)]$ is not NULL then
        $LUT[P(R_i)].AddValue(\overrightarrow{v_i})$;
        $LUT[P(R_i)].UpdateKey()$;
    end
    else
        $LUT[P(R_i)].CreateNewPos(\overrightarrow{v_i})$;
    end
end
```

When applying the algorithm described above, the order followed to add polylines into the lookup table is important since the primitive clustering is done in an incremental way. However, since in retrieval applications the user can add more and more documents to the database at any time, we preferred using an incremental primitive clustering to a classical classification method which would need a learning stage, making difficult increasing the data collection. In addition, the coarse primitive clustering offered by the lookup table is compensated by the use of a voting scheme.

In the next subsection, we will detail how we use the lookup table in order to retrieve the location of graphical primitives by similarity.

5.3.3 On-line Querying of Symbols: Activating Table Entries

Given a query symbol $S = \{P(R_1) \ldots P(R_p)\}$ and the lookup table containing q entries, a maximum of p table entries are activated, resulting, on the one hand, in a list of locations to find similar primitives and, on the other hand, in a confidence value depending on the similarity ratio between the query primitives and the prototypes representing these table entries. A table entry having as prototype a certain median string $\widetilde{P_j}$ is activated depending on the following condition:

$$\delta\big(P(R_i), \widetilde{P_j}\big) < thr \quad \text{where } 1 \le i \le p \text{ and } 1 \le j \le q. \tag{5.7}$$

From the traversal of the lookup table, we obtain a list of locations to find similar primitives as the ones that compose the query symbol. The zones of a document in the collection likely to contain the query symbol are the ones where we can find larger accumulation of the symbol's primitives. By using a Hough-like voting scheme, we determine the zones of the collection where we have more primitives by simply looking at maxima in the voting space. The locations where most of the polylines comprising the symbol S are present form clusters of coherent votes. The presence of similar polylines in other locations of the line drawing provokes false positive votes which are scattered in the voting space. The accumulation of evidences is done in terms of the similarity between the query primitive and the prototype, so the values of the votes to distribute in the parameter space are proportional to each $\delta(P(R_i), \widetilde{P_j})$. In the next section, we will detail how we proceed to validate the location hypotheses.

5.3.4 Hough-Like Voting Scheme to Validate Location Hypotheses

The voting space is a four-dimensional space (x, y, s, d) consisting of $2D$ position coordinates, a scale ratio and an index of a given document in the collection. Given a query string P_q, we accumulate votes in the translation coordinates $\overrightarrow{v_i} = (x_i, y_i)$ of the corresponding line-drawing image d. The third dimension of this space represents the scale factor between the query polyline and the polylines stored in the lookup table. This voting scheme formulates hypotheses of spatial location, document instance, and scale of the queried symbol. The zones of the line-drawing where we find similar primitives at a similar scale as the ones that form the query symbol tend to accumulate more votes and thus to form clusters in the voting space. The problem of finding zones where a symbol is likely to be found is then reduced to a local maxima localization problem in the voting space.

For the sake of simplicity, the voting space is split into several bins, $I_{(1,1,1)} \dots I_{(m,n,s)}$, named buckets for each document d. The bin size has to be related to the scale of the symbol so the different votes fall in nearby buckets. In our experiments, the grid size has been empirically set and is determined in terms of the size of the original image. In our case, m and n are determined such as $\max(m, n) = 128$, preserving the aspect ratio of the original image. The scale dimension is sampled to $s = 8$ possible buckets. The parameter d is directly the number of documents stored in the collection. Following a similar idea as the proposed by Lorenz and Monagan in [9], we use a voting method known in signal processing as anti-aliasing to relate $(\overrightarrow{v_i}, s)$ to a set of I_j neighboring buckets, based on the Euclidean distance between the voting location and the discrete bin partition buckets. Each $(\overrightarrow{v_i}, s)$ has the edit cost vote to distribute among its eight neighboring buckets, depending on their proximity.

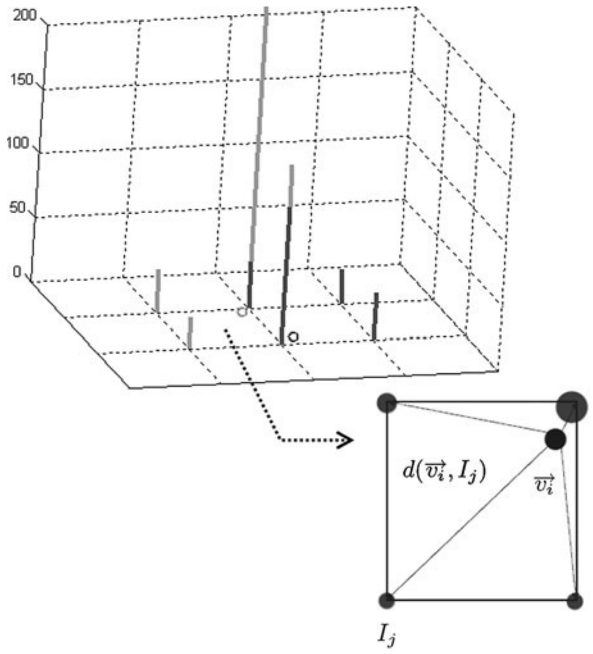


Fig. 5.5 Anti-aliasing method to cast votes. Even if two votes fall into different buckets due to discretization, they still contribute to form coherent peaks in the desired values of the voting space

Given a symbol S, the activation of the lookup table entries results in a list $L = \{(\overrightarrow{v_1}, s_1) \dots (\overrightarrow{v_n}, s_n)\}$ of translation vectors. For $d((\overrightarrow{v_i}, s_i), I_j)$ the Euclidean distance between $(\overrightarrow{v_i}, s_i)$ and one of the eight neighboring buckets I_j, we define the value of the vote $V(I_j)$ received in the bucket I_j as

$$V(I_j) = V(I_j) + \frac{w_1}{d((\overrightarrow{v_i}, s), I_j)} + \frac{w_2}{\delta(P(R_i), \widetilde{P_j})}, \tag{5.8}$$

where w_1 and w_2 weigh the distance factor and the edit cost. We can see an example of the voting distribution scheme in Fig. 5.5.

As we can see, the anti-aliasing method reduces the problem to working with a discrete grid to distribute votes. Voting schemes are only efficient if a high number of votes fall in the right bin, so that the bin can be easily detected among the background noise. If some votes fall in the neighboring bins, the significance of the correct bin decreases. Since the votes are now distributed among nearby buckets, even if the locations of a symbol do not fit a unique bin, the votes of close buckets collaborate between them.

Since the intended application of the spotting methods is a focused retrieval process, given a query symbol, the top k zones of interest in terms of accumulated votes are returned to the user. The more primitives a symbol has, the more votes can be accumulated in a given zone. However, since only one query is done at a time, there is no need to normalize the votes to retrieve the zones of interest.

In the next section, we will see the experimental results of testing the performance of the proposed description technique, and the ability of locating and retrieving symbols within a collection of complete documents.

5.4 Experimental Results

In order to evaluate the proposed spotting methodology, we present three different experiments. The first focuses only on the string matching algorithm as a distance measure between polygonal shapes. It aims at empirically determining a well suited threshold value *thr* which determines whether or not two strings are considered similar. The second experiment is designed to test if the proposed primitives are sufficiently discriminative to represent a graphical symbol. Finally, the third experiment tests the symbol spotting method by querying a document image database of real architectural floor-plans.

5.4.1 Silhouette Shape Matching

The first experiment is designed to test the efficiency of the string matching algorithm as a shape descriptor. The algorithm is used as a distance between two shapes represented by a polygonally approximated contour. This experiment also aims at empirically determining a well suited value of the threshold *thr* which influences whether two polylines are similar or not. We used a subset of isolated silhouette shapes from the MPEG-7 core experiment described by Latecki et al. in [8]. We call this polygonal shapes collection the MPEG-POLY[1] database.

For each of the 15 shape models, the noise model presented by Kanungo et al. in [6] is applied to generate 300 degraded images per class, which are then polygonally approximated. Applying the noise model and then converting the images from raster to vector format introduces a lot of variations in the number of segments approximating a given silhouette. All the details of this dataset can be checked in Appendix A. We run a classification experiment with all this dataset. The distance between each model and the vectorized shapes is computed by using the cyclic string matching algorithm with the proposed cost functions. These results are sorted by increasing distance to extract a Receiver Operating Characteristic (ROC) curve (the interested reader is referred to the paper by Fawcett [5] on ROC analysis), which plots the true positive rates against the false positive rates. These evaluation metrics are the same as previously used by us to evaluate the performance of the document classification method in Chap. 3. We can see how they are computed in (3.9).

We can compare the tradeoff between the correctly classified items and the appearance of false positives in Fig. 5.6. In our framework, as we use a voting scheme toaccumulate evidences, we are more interested in achieving high true positive rates values rather than having low false positive rates. We can find the obtained false positives rates and thresholds for different true positives rates in Table 5.1.

In the presented spotting method, the lookup table offers a coarse clustering that is then refined by the use of a voting scheme. The presence of false positives in a lookup table entry is not a problem, but we want to minimize the missed primitives.

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

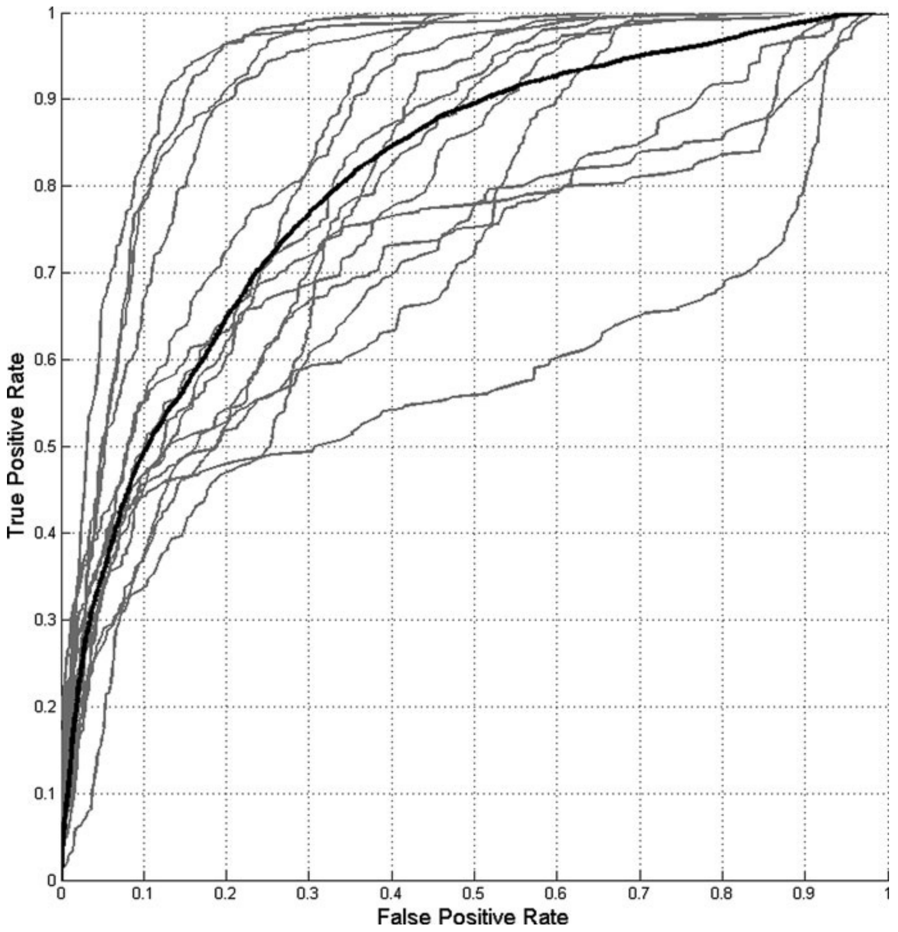


Fig. 5.6 Receiver operating characteristic curve for the silhouette matching experiment. Average ROC curve is shown in *black*

Table 5.1 Obtained false positive rates and decision threshold *thr* for several true positive rates

TPR	*FPR*	*thr*
0.25	0.025	0.008
0.5	0.105	0.014
0.75	0.281	0.024
0.9	0.511	0.035

In our experiments, we used a *thr* value of 0.03 which guarantees about a 75% of correctly clustered shapes in a given lookup table entry. False positives appear, but the voting strategy will hopefully discard them.

5.4.2 Evaluation of the Contours as Primitives

The second test aims at seeing if the region contours are better primitives to represent a graphical symbol than the skeletons which were the extracted features to be polygonally approximated in the last chapter. We compare the performance of the presented method by using both vectorization strategies, one computing the skeletons of the objects and then applying the Rosin and West's algorithm, and the other trying to approximate the contours of the closed regions extracted from the image. To carry this experiment, a real floor-plan has been degraded to build a collection of 500 synthetically distorted plans by using again the noise method of Kanungo et al. These distorted images were then polygonally approximated with both representations: contours and skeleton primitives. We can compare the differences between both primitives in Fig. 5.7.

In Fig. 5.8, we can see the obtained precision and recall graph (the interested reader is referred to van Rijsbergen's book [12] on information retrieval) when

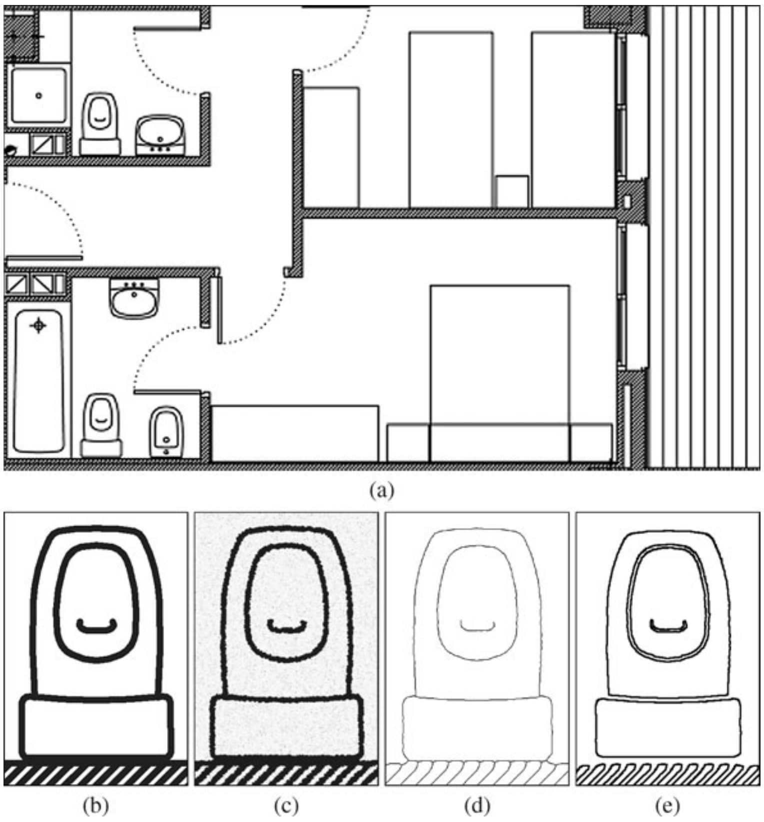


Fig. 5.7 Symbol primitive representations. (**a**) Model floor-plan; (**b**) zoom of the toilet symbol; (**c**) degraded image; (**d**) symbol with skeleton primitives; (**e**) symbol with contour primitives

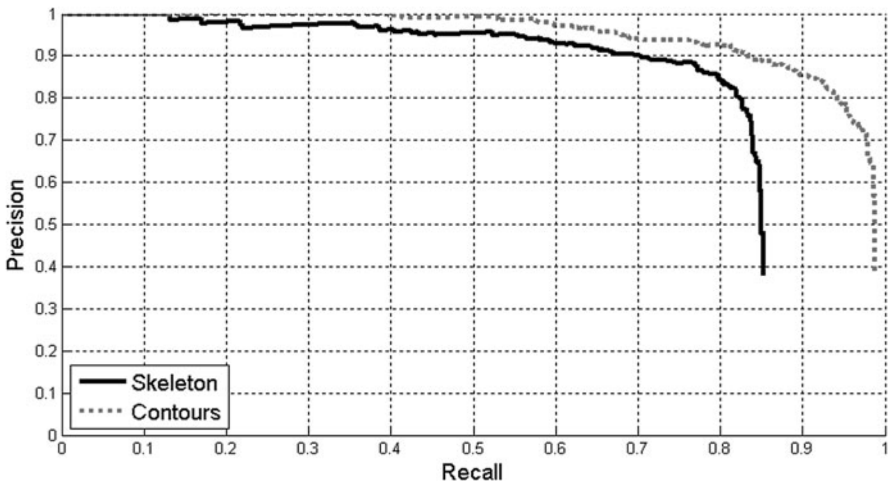


Fig. 5.8 Precision and recall plot when spotting the toilet symbol shown in Fig. 5.7 using two different symbol primitives. The contours outperform the skeleton primitives in both precision and recall

Table 5.2 Number of false positives when requesting a certain number of retrieved zones

Primitives	Retrieved items			
	200	300	400	475
False positives with Contours	6	7	29	159
False positives with Skeletons	73	76	89	273

querying a symbol. The graph shows that the presented primitives are more expressive than skeletons since the spotting method using this representation outperforms the skeleton in all cases. On average, there is a gain of about 17.5% in precision for the same recall values. Details are shown in Table 5.2 where we can see the number of false positives we have when requesting a certain number of the 500 possible solutions. In addition, using contours as primitives, the queried symbol has been missed in only 5 of the 500 images, and we miss the symbol in 73 of the 500 images using the skeleton. This yields a significant gain in the recall value when using contours instead of approximating the symbol skeleton.

Finally, we tested the method on a database of isolated symbols affected by vectorial noise. We can see the obtained ROC curve for the matching experiment with the 150 isolated symbols from the GREC-POLY[2] database (fully detailed in Appendix A) in Fig. 5.9. The experimental setup is the same as in the silhouette matching experiment. The main difference is that we do not try to match individual shapes but rather all the primitives comprising a given graphical symbol. When all the prim-

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

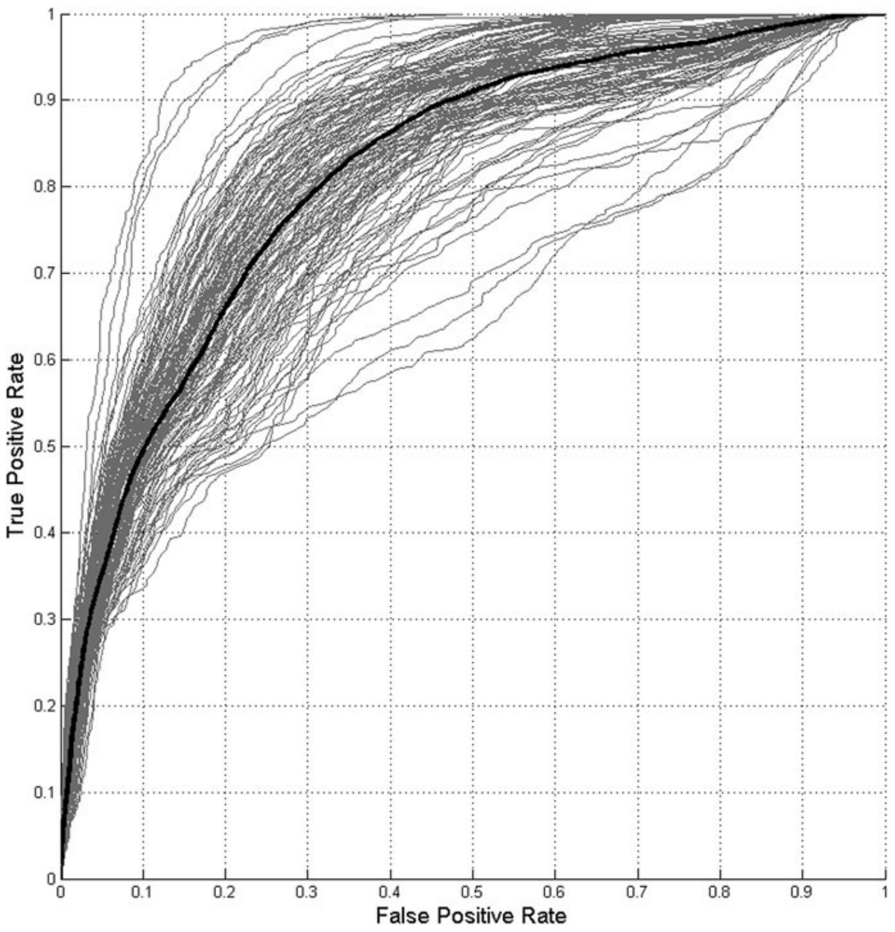


Fig. 5.9 Receiver operating characteristic curve for the symbol matching experiment. Average ROC curve of the 150 symbols is shown in *black*

itives from a symbol are matched against the model primitives, the symbol is then considered as recognized.

5.4.3 Symbol Spotting in a Document Database

Finally, we tested our method on a collection of ten real floor-plans and ten different symbols as queries. This is a subset of the FPLAN-POLY dataset[3] detailed

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

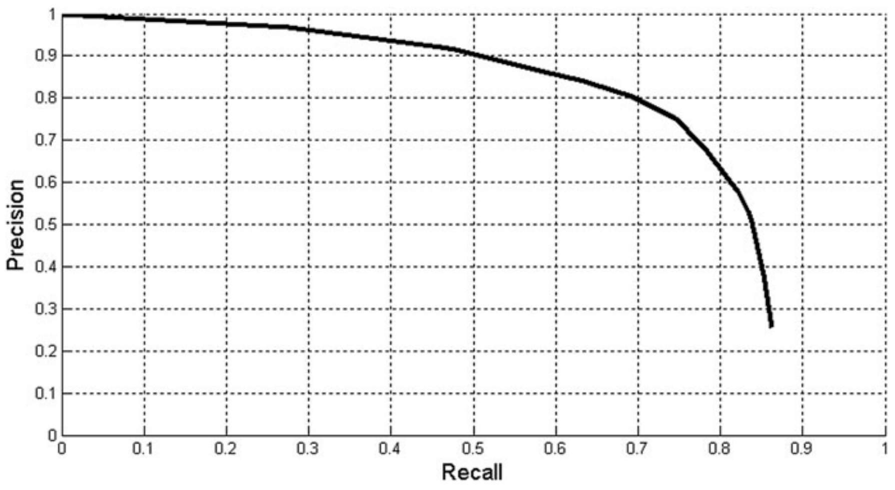


Fig. 5.10 Precision and recall plot for symbol spotting in the document database

in Appendix A. The query symbols appear in the floor-plans several times and are segmented by cropping a zone in the floor-plan image and vectorizing it. Each floor-plan has been polygonally approximated and ground-truthed. The database consists of approximately 14,200 polylines which after the lookup construction result in nearly 320 table entries. The number of distance computations is thus reduced by a factor of 45 with respect to a sequential access to all the primitives appearing in the collection. We can see the precision and recall plot resulting from spotting these symbols in the whole floor-plan database in Fig. 5.10.

In Table 5.3, we present a detailed set of measures to evaluate the performance of retrieval systems to evaluate the spotting architecture. As we can see, the recall ratio is quite good. However, there is a significant number of false positives in the results which harm the precision value. The F-score is a composite measure to rank the results. However, the most interesting point here is to notice the difference between the precision and the average precision $Ave\,P$ values. The average precision is a measure of quality which rewards the earliest return of relevant items. Details on these measures can be found in Chapter 7. As we can see in our experiments, even if the precision values are quite low, the average precisions are significantly higher. This means that the false positives are usually ranked worse than the correct results, as we can also see in the qualitative results shown in Fig. 5.11. Finally, we also show the average time taken by our software prototype to spot a symbol per plan. It is remarkable that the symbols which are composed by common simple primitive shapes (circles, squares, etc.) are usually the ones which are more time consuming since the entries of the lookup are more populated and more hypotheses have to be considered. No significant differences due to the number of polylines comprising a symbol can be observed.

Table 5.3 Detailed retrieval measures for each model symbol for the symbol spotting in a document database experiment

Symbol		Retrieval measures				
Class	p	Precision (%)	Recall (%)	F-score (%)	*Ave* P (%)	Time (s/plan)
Bidet	4	30.8	100	47.1	87.5	0.76
Chair	5	36.8	100	53.8	83.3	0.64
Burners	9	5.1	100	9.6	59.1	1.09
Toilet	5	50	37.5	42.9	27.1	0.98
Toilet sink	5	30	100	46.2	68.7	1.89
Kitchen sink	5	11.8	50	19	33.3	1.16
Single sofa	4	37.5	100	54.6	100	0.43
Double sofa	6	15	75	25	65	0.22
Table	7	16.7	100	28.6	100	0.24
Tv set	4	20	100	33.3	95	0.12
Average	5.4	25.4	86.2	36	71.9	0.75

5.5 Conclusions and Discussion

In this chapter, we have presented a method of symbol spotting and its use in a focused retrieval application from a collection of technical line-drawings. First, a suitable symbol representation as a set of closed region contours and its codification with attributed strings has been presented. The distance definition using a cyclic string matching algorithm allows tolerating the segment fragmentation problem. Then, a clustering of salient zones of interest and a voting method have been presented and tested to spot symbols in real technical line-drawings.

The experiments show that the representation and distance approaches are able to tackle the inherent noise arising from the scanning process and the distortions introduced by the raster-to-vector algorithms. The presence of false positives is not a critical problem since the purpose of spotting methods is to find by a fast technique a coarse identification of zones where a given symbol appears. Finally, we can see that the use of voting strategies is of vital importance for spotting problems. To reach higher precision, one can use better shape descriptors; however, this also entails a complexity increment. The accumulation of evidences allows working with a coarse recognition in the indexing step.

There are still some aspects which should be further studied. The main concern is that the order followed to add polylines in the lookup table is important and in some cases could lead to some misclassifications. However, for spotting applications where the user can add more and more documents at any time, the primitive clustering must be incremental. The use of incremental classifiers such as iPCA [2] or iLDA [11] applied to primitive clustering should be studied. On the other hand, the presented matching approach cannot cope with occlusions which will provoke

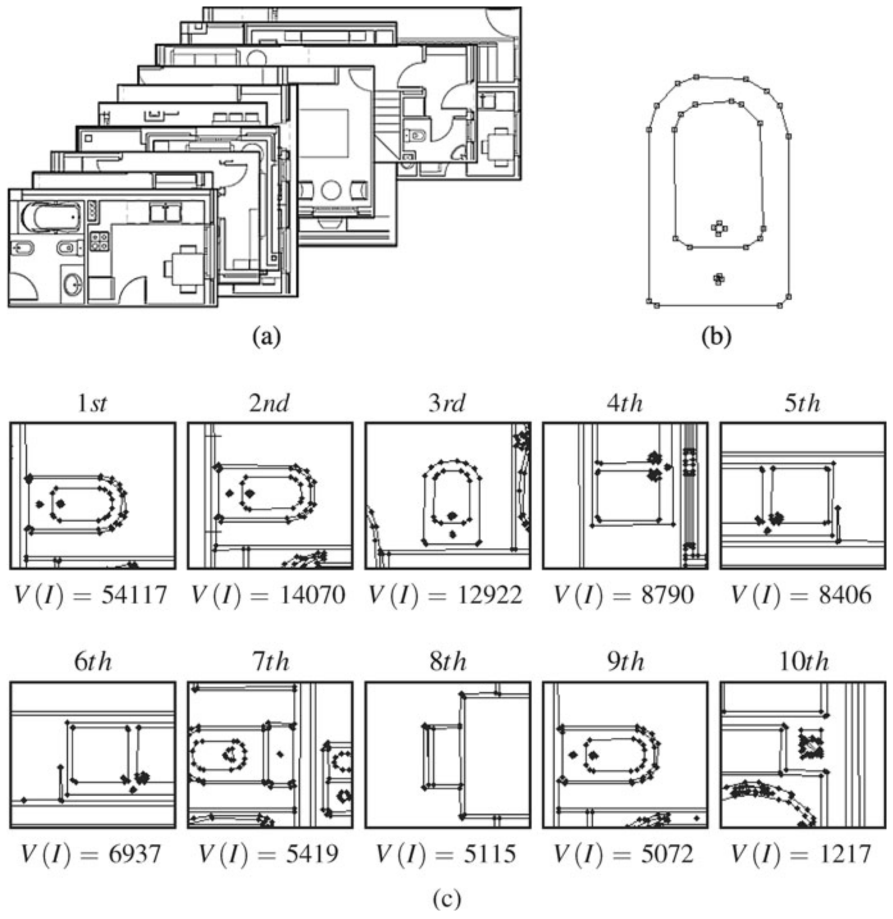


Fig. 5.11 Qualitative results for symbol spotting by cyclic string matching. (**a**) Vectorized floor-plan database; (**b**) query example; (**d**) ranked top-ten results

the polylines to be broken. A partial matching algorithm such as the one presented in [17] by Tănase et al. could be helpful in such situations.

One of the main advantages of the proposed method regarding the vectorial signature approach is the use of a prototype-based search. This indexing technique provides an efficient way to retrieve the locations of graphical patterns by similarity. Even if the implementation of the method is still a prototype and has not been optimized, the times to retrieve the occurrences of a symbol given in Table 5.3 are encouraging. However, depending on the applications, the number of table entries may increase drastically, and even the computation of the distance with the prototypes can be time consuming. The use of hashing structures instead of lookup tables can provide a more efficient access to the data without computing any distance with prototype primitives. We propose using these particular data structures for a faster primitive retrieval by similarity in the next chapter.

There is another drawback in the presented approach. Since graphical symbols are composed of several primitives, querying a symbol consists in separately querying each of its primitives. The locations showing a higher accumulation of primitives are the most plausible places to contain the queried symbol. The structural configuration of these primitives in that location is not taken into account. Most of the false alarms presented in the qualitative results do not contain any similar symbol as in the query. But in those regions, there is usually a high presence of simple primitives, and these locations accumulate several votes. In the next chapter, we present an indexing methodology to add structural information in the primitive queries. An enhanced voting scheme to better validate the spotted locations is also presented in the next chapter.

Finally, our feeling is that representing the primitives as attributed strings is a powerful description technique. The retrieval by similarity with this particular data representation has, however, an important burden compared with descriptors working with a feature vector description. Symbolic descriptions are meaningful but computationally expensive to match. Numerical descriptions are usually less expressive, but easier to match by just the definition of a distance. In the next chapter, we will use several off-the-self numerical description techniques instead of symbolic ones, in order to foster the retrieval by similarity.

References

1. Arkin, E., Chew, L., Huttenlocher, D., Kedem, K., Mitchell, J.: An efficiently computable metric for comparing polygonal shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence **13**(3), 209–216 (1991)
2. Artač, M., Jogan, M., Leonardis, A.: Incremental PCA for on-line visual learning and recognition. In: Proceedings of the International Conference on Pattern Recognition, pp. 781–784. IEEE Computer Society, Los Alamitos (2002)
3. Bunke, H., Bühler, U.: Applications of approximate string matching to 2D shape recognition. Pattern Recognition **26**(12), 1797–1812 (1993)
4. Dori, D.: Vector-based arc segmentation in the machine drawing understanding system environment. IEEE Transactions on Pattern Analysis and Machine Intelligence **17**(11), 1057–1068 (1995)
5. Fawcett, T.: An introduction to ROC analysis. Pattern Recognition Letters **27**(8), 861–874 (2006)
6. Kanungo, T., Haralick, R., Philips, I.: Global and local document degradation models. In: Proceedings of the Second International Conference on Document Analysis and Recognition, pp. 730–734. IEEE Computer Society, Los Alamitos (1993)
7. Kaygin, S., Bulut, M.: Shape recognition using attributed string matching with polygon vertices as the primitives. Pattern Recognition Letters **23**(13), 287–294 (2002)
8. Latecki, L., Lakämper, R., Eckhardt, T.: Shape descriptors for non-rigid shapes with a single closed contour. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 424–429. IEEE Computer Society, Los Alamitos (2000)
9. 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)
10. Maes, M.: Polygonal shape recognition using string-matching techniques. Pattern Recognition **24**(5), 433–440 (1991)

11. Pang, S., Ozawa, S., Kasabov, N.: Inremental linear discriminant analysis for classification of data streams. IEEE Transactions on Systems, Man and Cybernetics **35**(5), 905–914 (2005)
12. van Rijsbergen, C.: Information Retrieval. Butterworth-Heinemann Newton, MA, USA (1979)
13. Rosin, P., West, G.: Segmentation of edges into lines and arcs. Image and Vision Computing **7**(2), 109–114 (1989)
14. Sánchez, G., Lladós, J., Tombre, K.: A mean string algorithm to compute the average among a set of 2D shapes. Pattern Recognition Letters **23**, 203–213 (2002)
15. Stein, F., Medioni, G.: Structural indexing: Efficient 2D object recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **14**(12), 1198–1204 (1992)
16. Stein, F., Medioni, G.: Structural indexing: Efficient 3D object recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **14**(2), 125–145 (1992)
17. Tănase, M., Veltkamp, R., Haverkort, H.: Multiple polyline to polygon matching. In: Algorithms and Computation, *Lecture Notes on Computer Science*, vol. 3872, pp. 60–70. Springer, Berlin (2005)
18. 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)
19. Wagner, R., Fischer, M.: The string-to-string correction problem. Journal of the Association for Computing Machinery **21**(1), 168–173 (1974)
20. Wenyin, L., Dori, D.: Incremental arc segmentation algorithm and its evaluation. IEEE Transactions on Pattern Analysis and Machine Intelligence **20**(4), 424–431 (1998)
21. Wolfson, H.: On curve matching. IEEE Transactions on Pattern Analysis and Machine Intelligence **12**(5), 483–489 (1990)

Chapter 6
A Relational Indexing Method for Symbol Spotting

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

6.1 Introduction and Related Work

The use of a lookup table providing a prototype-based search of similar primitives, as presented in the last chapter, allows avoiding the computation of the similarity measure for all the primitives extracted from the collection. The use of such indexing structures aims at efficiently accessing and retrieving graphic elements by similarity, and becomes a must when dealing with applications which have to face large collections of documents. In the particular usage case presented in the last chapter, we achieved reducing the number of distance computations by almost a factor of 45 without missing a significant number of symbols. However, there is still need to compute several hundreds of distances between descriptors. Even if this is not an important burden when working with numeric descriptors, it may be an important inconvenience when we use symbolic description of primitives such as the attributed strings. In this chapter, we propose enhancing the accessibility to the stored descriptors by two means. First, we will coarsely describe primitives by the use of well-known descriptors with low dimensionality. These descriptors result in a numeric feature vector. The distance among those descriptors is easily computed as the distance between two points in the n-dimensional description space. Second, this description space is efficiently organized and accessed by the use of a hashing technique. The use of hashing techniques in ideal conditions allows retrieving items by similarity with a complexity $\mathcal{O}(1)$. We can find many works which use such efficient indexing structures to organize and retrieve the primitive descriptors in the literature. Califano and Mohan [2] used a hash table indexed by four-dimensional indices describing the geometric configuration of triplets of points extracted from a contour image in order to efficiently locate the location of query objects in an image. Stein and Medioni [19] also used a hash table in order to provide an efficient retrieval of similar portions of a contour described by a set of features extracted from a super-segment. Recently, Lladós and Sánchez [12] proposed a binary codification of the shape context descriptor which is stored in an indexing structure aiming at efficiently retrieving the locations within a document image where a given typewritten word is likely to appear.

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

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

6.2 Description of Graphical Symbols in Terms of Vectorial Primitives

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

6.2.1 Vectorial Primitives

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

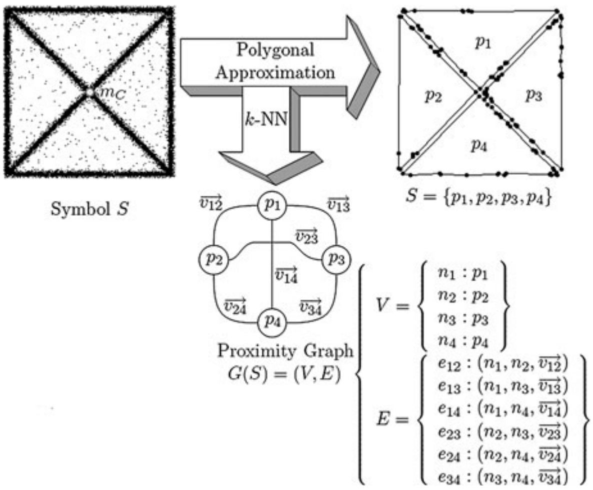
As our work is focused on the management of graphical data in vectorial format, the documents which are in paper format need a digitalization process. In this chapter, we use the same raster-to-vector process as in the previous chapter with just one particularity. Since we want to add relational information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

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

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

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

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



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

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

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

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

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

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

6.3.1 Geometric Moments

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

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

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

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

6.3.1.1 Boundary Moments

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

6.3.1.2 Geometric Moments for Line Segments

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

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

And if the segment s_i is vertical, we use

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

6.3.1.3 Hu's Moment's Invariants

To obtain invariance with respect to translation, the centroid is used as in (6.1). The normalization by the polyline length is used to obtain scaling invariance. Finally, invariance to rotation is achieved by using the set of seven functions proposed in [7] involving moments up to the third order.

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

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

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

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

$$\begin{aligned}
\phi_1' &= \phi_1 = \eta_{20} + \eta_{02}, \\
\phi_2' &= \phi_2/\phi_1^2, \\
\phi_3' &= \phi_3/\phi_1^3, \\
\phi_4' &= \phi_4/\phi_1^3, \\
\phi_5' &= \phi_5/\phi_1^6, \\
\phi_6' &= \phi_6/\phi_1^4, \\
\phi_7' &= \phi_7/\phi_1^6.
\end{aligned} \tag{6.7}$$

6.3.2 Simple Shape Description Ratios

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

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

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

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

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

6.3.3 Fourier Descriptors

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

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

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

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

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

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

6.4 Multidimensional Hashing to Index Primitives

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

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

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

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

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

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

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

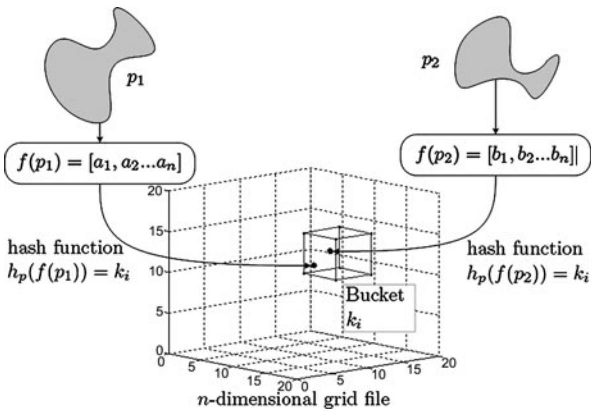


Fig. 6.2 The use of a grid file to index vectorial primitives. The hash function projects the feature vectors into key-indices. Two similar primitives are stored at the same bucket

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

6.5 Relational Indexing and Hypotheses Validation

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

6.5.1 Relational Indexing

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

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

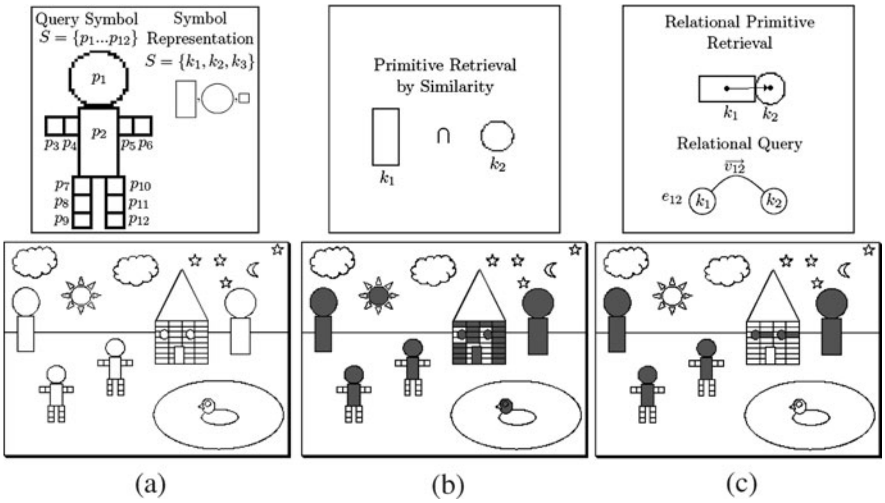


Fig. 6.3 Relational indexing. For the sake of visibility, only two primitives p_1 and p_2 are queried. (**a**) Sample line-drawing and the query symbol; (**b**) results of retrieving a couple of primitives by similarity without taking into account the spatial information, the resulting primitives are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

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

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

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

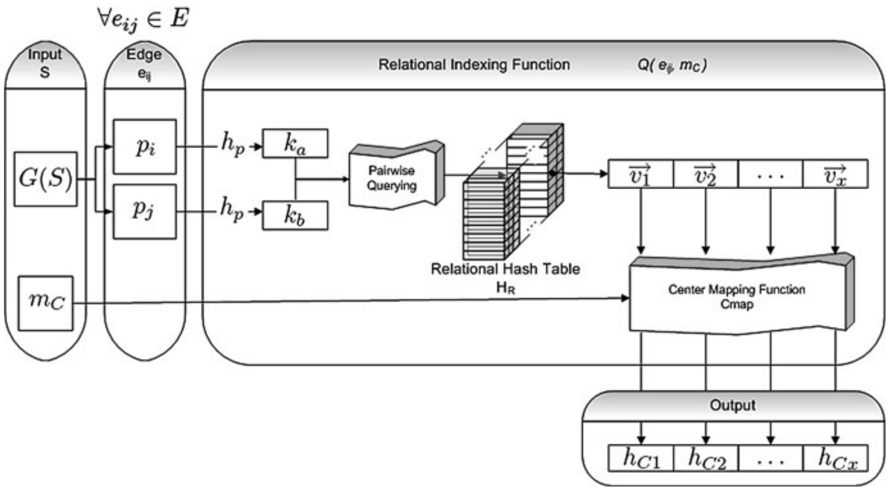

$\forall e_{ij} \in E$
Input S
Edge e_{ij}
Relational Indexing Function
$Q(e_{ij}, m_C)$
$G(S)$
p_i
p_j
h_p
k_a
k_b
Pairwise Querying
Relational Hash Table H_R
$\vec{v_1}$ $\vec{v_2}$ ... $\vec{v_x}$
Center Mapping Function Cmap
m_C
Output
h_{C1} h_{C2} ... h_{Cx}

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

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

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

6.5.2 Voting Scheme

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

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

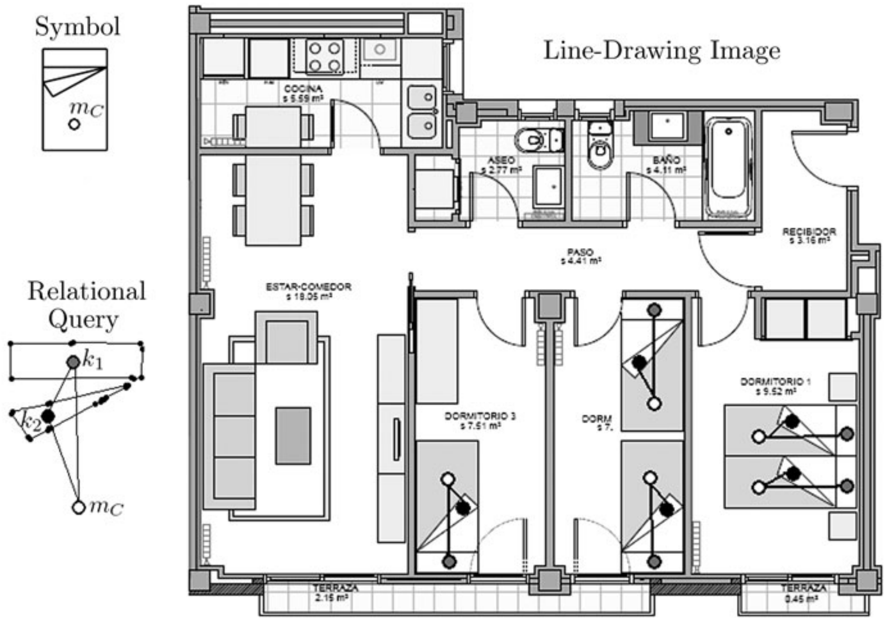


Fig. 6.5 Center mapping function to find the pose of the hypothetic centers given an edge of the relational query and the gravity center of the query symbol

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

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

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

6.6 Experimental Results

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

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

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

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

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

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

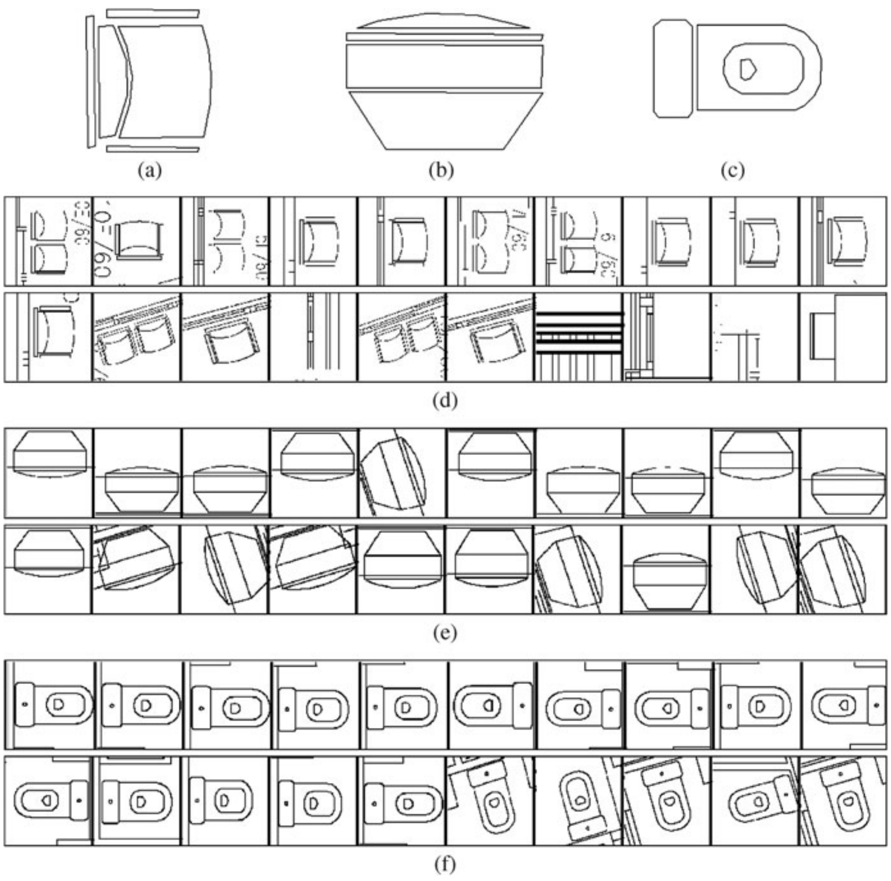


Fig. 6.6 Qualitative results of the relational indexing method (1). (**a**) Query symbol *chair*; (**b**) query symbol *TV set*; (**c**) query symbol *toilet*; (**d**), (**e**) and (**f**) first 20 retrieved regions when querying the symbols of (**a**), (**b**) and (**c**), respectively

6.7 Conclusions and Discussion

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

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

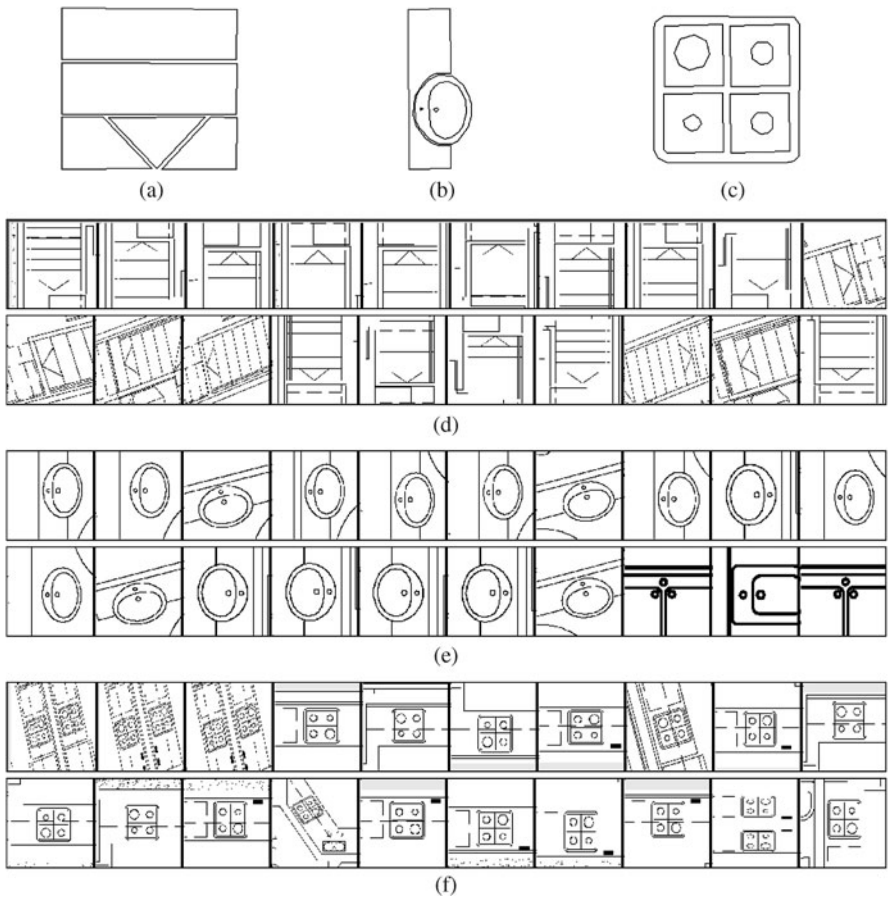


Fig. 6.7 Qualitative results of the relational indexing method (2). (**a**) Query symbol *stairs*; (**b**) query symbol *sink*; (**c**) query symbol *burners*; (**d**), (**e**) and (**f**) first 20 retrieved regions when querying the symbols of (**a**), (**b**) and (**c**), respectively

Table 6.1 Recognition results of the relational indexing method

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

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

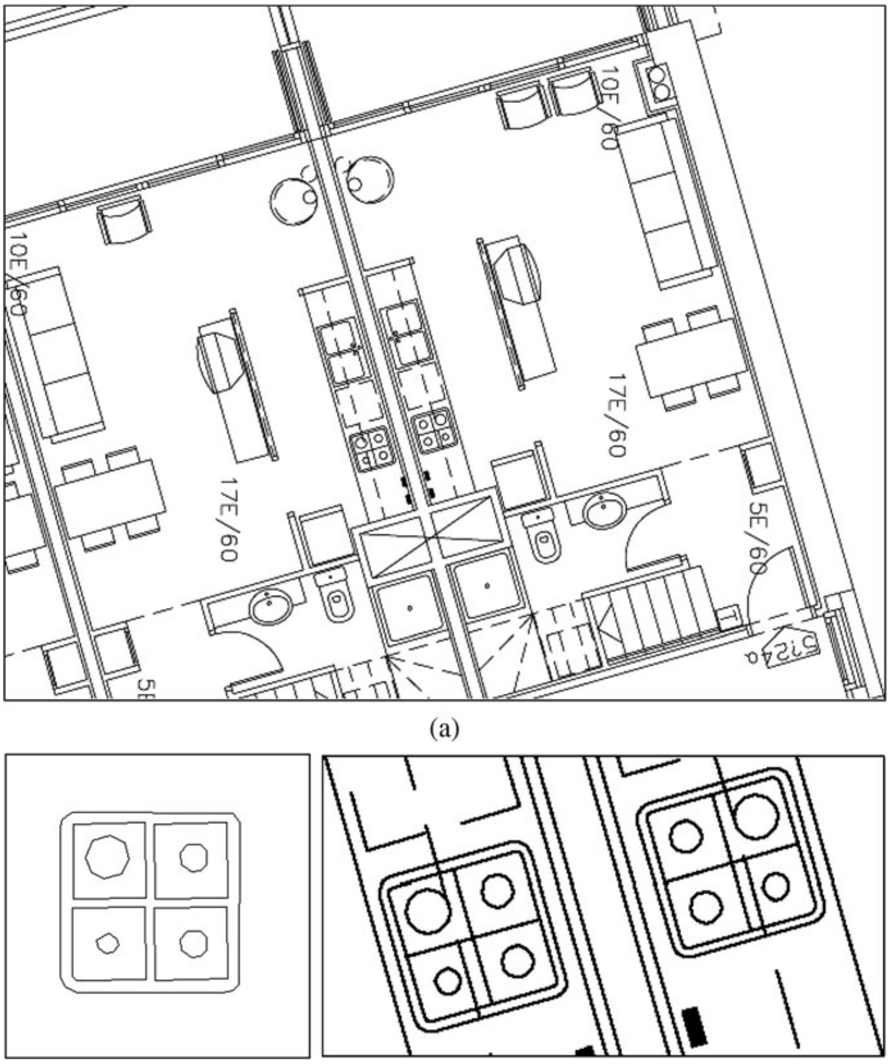


Fig. 6.8 Illustration of a result which is difficult to evaluate. (**a**) Floor-plan image in the collection; (**b**) queried symbol; (**c**) retrieved region

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

In some application domains, as, for instance, in the case of complex electronic diagrams, some symbols share a substantial part of their design and only differ by slight details. Symbols may also be composed of other known and significant symbols. In this context, the proposed focused retrieval methodology might result in an important number of false alarms.

References

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

Part IV
A Performance Evaluation Protocol for Symbol Spotting Systems

Chapter 7
Performance Evaluation of Symbol Spotting Systems

Abstract Symbol spotting systems are intended to retrieve regions of interest from a document image database where the queried symbol is likely to be found. They shall have the ability to recognize and locate graphical symbols in a single step. In this chapter, we present a set of measures to evaluate the performance of a symbol spotting system in terms of recognition abilities, location accuracy and scalability. We show that the proposed measures allow determining the weaknesses and strengths of different methods. In particular, we have evaluated in detail the spotting method presented in Chapter 6.

7.1 Introduction

Performance evaluation methods are essential tools to understand and compare the behavior of algorithms and systems. A performance evaluation protocol should identify the strengths and weaknesses of the methods under test. The analysis of these strong points and drawbacks should determine which method is the most suitable for a certain use case and predict its behavior when using it in real applications with real data.

In the last years, performance evaluation has been a quite prolific research topic in the Document Image Analysis and Recognition field and in particular among the Graphics Recognition community. Several competitions focused on particular topics, namely, symbol recognition, layout analysis, and text detection among others, have been organized in the major conferences and workshops of this field. We can also find a lot of contributions in the recent literature proposing evaluation techniques for different document image analysis applications. Performance evaluation frameworks have been proposed for evaluating low-level applications such as line and arc detection algorithms [37, 38] or raster-to-vector systems [31]. However, in the last years, frameworks to evaluate higher level applications such as symbol recognition [36] or layout analysis [2] have been proposed.

In this chapter, we propose a set of measures and methodologies to evaluate the performance of spotting systems. Although we mainly focus on the specific case of symbol spotting, these measures are also applicable to performance evaluation of other focused retrieval applications such as word spotting, or even object recognition in Computer Vision applications. Symbol spotting techniques should efficiently locate graphical symbols in document images without using full recognition methods. Such systems are intended to index large collections of document images in terms of the graphical symbols which appear in them. Given a graphical symbol as a query, the system has to retrieve a ranked list of locations where the query symbol is likely to be found. Since spotting systems deal with recognition and segmentation at the same time, such abilities must be taken into account by the evaluation process. Segmentation errors must be punished as well as recognition mistakes.

As we illustrated in the review provided in Sect. 7.2, there exist many approaches to measure the performance of different Graphics Recognition algorithms. However, in the particular case of symbol spotting, existing methods in the literature just provide measures based on binary decisions of *found/not found*. Along this book we have evaluated the proposed methods on these binary decisions. Based on a decision of wether to consider a symbol as being correctly located, we have presented all the results by giving information about the true positive rate (TPR) and the false positive rate (FPR) for the recognition and classification applications (Chapters 3, 4, 5 and 6). We gave the results for the focused retrieval application in terms of the precision and recall of the binary decisions (Chapter 5).

In this chapter, we develop the theory keeping in mind that the performance of a symbol spotting system should be defined in terms of two components: the recognition and the location goodness. Starting from these hypotheses, the main contribution of this chapter is to propose a set of performance evaluation measures based on the precision and recall concepts and to evaluate the performance of symbol spotting systems in terms of two criteria, namely recognition and location. In addition, the second contribution is the use of the same formalism to evaluate a third quality criterion, the scalability under an increasing number of symbol prototypes. Most of the work found in the literature dealing with performance evaluation of Graphics Recognition systems is mainly focused on the computation of a score to allow an easy way to rank different methods. We strongly believe that the proposed measures can give a more accurate idea of the real behavior of the system under study than typical recognition rates.

The remainder of this chapter is organized as follows. In Sect. 7.2, we briefly overview the work on performance evaluation for related areas such as retrieval systems, Graphics Recognition and Document Image Analysis applications. In Sect. 7.3, we basically review the well-known measures of precision and recall typically used in retrieval evaluation, and then discuss the measures we can derive from precision and recall. Section 7.4 outlines how these measures can be reformulated and applied to evaluate a spotting system in terms of retrieving regions of interest from a document image database. Section 7.5 shows a use case of such measures, evaluating the performance of the symbol spotting method presented in Chapter 6, which is based on a set of four different off-the-shelf shape descriptors. Finally, the conclusions and a short discussion can be found in Sect. 7.6.

7.2 Related Work

Symbol spotting systems are intended to produce a ranked list of regions of interest cropped from the document images stored in the database where the queried symbol is likely to be found. Symbol spotting can thus be seen as a particular application within the Information Retrieval (*IR*) domain. Usually, retrieval systems are evaluated by *precision* and *recall* ratios which give an idea about the relevance and the completeness of the results (we will briefly review these measures in Sect. 7.3). These basic measures can be enhanced considering many other indicators depending on the application. For instance, Lu et al. [20] evaluate a set of desktop search engines by deriving a set of ratios from precision and recall to indicate the abilities of the systems when incrementally retrieving documents. Müller et al. [25] evaluate content-based image retrieval systems, proposing some strategies to take into account the way the number of items stored in the collection affects the results and how user feedback can improve the response of such systems. Kang et al. [16] evaluate a text retrieval system which uses semantic indexing, focusing on the distribution and amount of key-indices used to index the database. Finally, in [12, 26], we can find the performance analysis of some information retrieval systems having the information distributed in a peer-to-peer network (*P2PIR*), which takes into account the query response time, the network resources requirements and the tradeoff between distributed and centralized systems. As we can see, the coverage of information retrieval topic is so wide that even if researchers use similar indicators to evaluate the performance of their methods, no general evaluation framework can be defined. In our case, we will also base our measures on the notions of precision and recall by adapting them to the recognition and location abilities that the spotting systems should present.

In the Document Image Analysis and more particularly the Graphics Recognition field, some work focused on spotting can be found. However, all this work is evaluated by ad-hoc measures. For instance, Rath and Manmatha [28] presented a system able to spot handwritten words in ancient documents. They evaluate their system with a score based only on the precision value. Marcus [24] presented an algorithm to spot spoken words in an audio signal. The evaluation is based on Receiver Operating Characteristics (*ROC*) graphs [11] which are related to precision and recall measures. Tabbone and Zuwala [33] present a method to spot graphical symbols in a collection of electronic drawings. They base the evaluation of their method in precision and recall graphs. Finally, Valveny et al. [36] present a framework to evaluate symbol recognition methods envisaging a way to evaluate location and recognition of symbols by also using precision and recall measures. However, all these methods are computed on a binary retrieval notion: whether an item is considered retrieved or not. By these measures one can see the ability of the system in retrieving relevant items and discarding negative ones, but these measures do not evaluate how well the system located the queried objects.

To avoid binary relevance labeling, our measurements are inspired by the techniques used to evaluate layout analysis systems. In fact, layout analysis shares some similarities with spotting in the sense that sub-regions from documents have to be labeled according to their content. Layout analysis competitions [3, 4, 6] were held in last editions of the *ICDAR* conference. In these contests, the evaluation of the participants' methods was done according to the overlapping between regions of the results and the ground-truth. Two indicators introduced in [27] are used to formulate an entity detection measure from which an averaged segmentation measure is deducted to score the systems. Following the same idea, in the text detection competitions [21, 22] held in last editions of *ICDAR*, precision and recall measures were computed in terms of overlapping between bounding-boxes of the ground-truth and the results. From the precision and recall numbers, a score was computed to rank the algorithm performance. However, we believe that the use of a single evaluation score allows an easy ranking of the different systems, but hinders the understandability of their behavior and the performance prediction when using other type of datasets.

Finally, in the last symbol recognition competitions [1, 34, 35] held in the *GREC* workshop editions, several symbol descriptors were evaluated. Here the performance was evaluated by the recognition rates the systems yielded. In the last edition, other measures such as the homogeneity and the separability of the symbol classes in the description space have been introduced. We find very interesting the fact that the scalability of the systems was also tested. This test was performed looking at how the performance of the systems evolved as the number of symbol classes to consider increased.

The measures we propose in this chapter are based on precision and recall since this has been demonstrated to be a good way to evaluate recognition (or at least classification) and location at the same time. We formulate the precision and recall notions in terms of overlapping between retrieved areas and ground-truth. The presented measures and plots allow assessing the weaknesses and strengths of the methods in terms of recognition abilities and location accuracy. In addition, we also present a methodology to extract a scalability measure from precision and recall to test if the methods can be used with a larger number of classes. Let us first review the basic measures used to evaluate retrieval effectiveness.

7.3 An Overview on Measures to Evaluate Retrieval Effectiveness

In this section, we review the basic measures provided in the literature used to evaluate the retrieval effectiveness. The measures outlined in this section will be reformulated in Sect. 7.4 for the framework described in this work.

7.3.1 Precision and Recall

In the information retrieval field, most measures to evaluate effectiveness are based on a binary labeling of relevance of the items, namely whether each item is con-

Table 7.1 Retrieval matrix

	Relevant	Non-relevant	Total
Retrieved	$\|ret \cap rel\|$	$\|ret \cap \overline{rel}\|$	$\|ret\|$
Not retrieved	$\|\overline{ret} \cap rel\|$	$\|\overline{ret} \cap \overline{rel}\|$	$\|\overline{ret}\|$
Total	$\|rel\|$	$\|\overline{rel}\|$	$\|tot\|$

sidered as relevant or non-relevant. In addition, these measures are also based on a binary retrieval notion, i.e., whether an item is retrieved or not.

Given a database consisting of a set of elements *tot* and a query item i to retrieve from it, let us label as *rel* the set of relevant objects in the set and $\overline{rel}$ the set of non-relevant items with regard to the query i. When querying this item in the database, we label as *ret* the set of retrieved elements and as $\overline{ret}$ the set of elements from the database which were not retrieved. The retrieval matrix of Table 7.1 shows all the possibilities in terms of intersections between these sets.

The analysis of this table allows defining the well-known ratios of precision and recall (see van Rijsbergen's [29] book on Information Retrieval for more details) to evaluate the behavior of the information retrieval system which are computed as follows:

$$P = \frac{|ret \cap rel|}{|ret|}, \qquad R = \frac{|ret \cap rel|}{|rel|}. \tag{7.1}$$

For a given retrieval result, the *precision* measure P is defined as the ratio between the number of relevant retrieved items and the number of retrieved items. The precision measure measures the quality of the retrieval system in terms of the ability of the system to only include relevant items in the result. A 100% precision means that no false positive has been included in the system response. As the precision value decreases, the more non-relevant items are included in the results.

The *recall* ratio R is defined as a ratio of the number of relevant retrieved items to the total number of relevant items in the collection. It measures the effectiveness of the system in retrieving the relevant items. A 100% recall means that all the items labeled as relevant are retrieved, and none has been missed. As the recall value decreases, the more relevant items are missed by the system which wrongly considers them as non-relevant.

7.3.2 $P@n$ and $P(r)$

The precision and recall measures are computed on the whole set of items returned by the system. That is, they give information about the final performance of the system after processing a query and do not take into account the quality of ranking in the resulting list. Information retrieval systems return results ranked by a confidence value. The first retrieved items are the ones the system believes are more likely to match the query. As the system provides more and more results, the probability to find non-relevant items increases.

Relevance ranking can be evaluated computing the precision at a given cut-off rank, considering only the n topmost results returned by the system. This measure is called precision at n or $P@n$. However, this measure presents the drawback that it does not give information about recall.

Let us define $P(r)$ as the precision at a given recall cut-off, that is, the precision at that point where recall has first reached the value r.

7.3.3 Precision and Recall Plots

The usual way to represent the stability of the system, as the user requires more and more results, is to plot precision and recall against each other. Such plots are computed stepwise retrieving at each step a given item, while varying the decision threshold value over the confidence rate, i.e., computing $P@n$ for the different values of n and plotting these values against their associated recall.

These plots show the tradeoff between precision and recall. Buckland and Gey [8] analyzed the relationship between both ratios and concluded that they are inversely related: trying to increase one usually provokes the other to be reduced. Thus, when comparing several methods, the one yielding the higher values for both precision and recall will be the best. However, it is not always easy to assess which precision and recall plot corresponds to a better system.

7.3.4 Measures of Quality

Sometimes it is difficult to measure the effectiveness by a measure composed by more than a number. In certain cases, the difficulty to assess which method is the best has led to invest in some composite measures which are able to rank the methods under study according to a combination of precision and recall information. However, as claimed by van Rijsbergen in [29], these measures are usually rather ad-hoc and difficult to interpret.

Let us see a couple of composite measures which try to combine both precision and recall information in a single number.

7.3.4.1 Average Precision

We can define the average precision *Ave P* using each precision value after truncating at each relevant item in the ranked list which resulted after a query. Average precision is one of the evaluation measures used by the *TRECVid*[1] community [30].

[1]TREC Video Retrieval Evaluation (http://www-nlpir.nist.gov/projects/trecvid/).

For a given query, let $\mathrm{r}(n)$ be a binary function of the relevance of the nth item in the returned ranked list. We define the average precision as

$$Ave\,P = \frac{\sum_{n=1}^{|ret|}(P@n \times \mathrm{r}(n))}{|rel|}. \tag{7.2}$$

The average precision is a measure of quality which rewards the earliest return of relevant items. Retrieving all relevant items in the collection and ranking them perfectly will lead to an average precision of 1. The average precision can also be seen as the area under the precision and recall plot. However, average precision does not take into account the fact that a system returns non-relevant items after having reached a 100% recall (i.e., having returned all relevant items).

7.3.4.2 *F*-score

Another classical composite measure is the F-score (see [13] for more details) which is the weighted harmonic mean of precision and recall computed as

$$F^\beta = \frac{(1+\beta^2) \times P \times R}{(\beta^2 \times P) + R}, \tag{7.3}$$

which for a value of $\beta = 1$ is equivalent to Dice's coefficient (a well-known similarity measure between two sets X and Y) defined as

$$s = \frac{2|X \cap Y|}{|X| + |Y|}. \tag{7.4}$$

Although there is some work like the one presented by Makhoul et al. in [23] which point out some drawbacks of this measure, the F-score is widely used as a measure of merit in the information retrieval literature.

The F-score can also be computed at several recall cut-offs to evaluate the stability of a system's response. We reformulate the F-score presented in (7.3) for several recall values as

$$F^\beta(r) = \frac{(1+\beta^2) \times P(r) \times r}{(\beta^2 \times P(r)) + r} \quad \text{with } r \in [0, R]. \tag{7.5}$$

We can see some examples of how $F^1(r)$-score evolves for several synthetic precision and recall plots in Fig. 7.1. The better the system responds, the higher its values. As we can notice, the F-score heavily penalizes low values of precision or recall.

7.3.5 *Fall-out and Generality*

Let us finally introduce two more measures, one related to the non-relevant retrieved items and the other related to the dataset, which are computed as

$$Fo = \frac{|ret \cap \overline{rel}|}{|\overline{rel}|}, \qquad G = \frac{|rel|}{|tot|}. \tag{7.6}$$

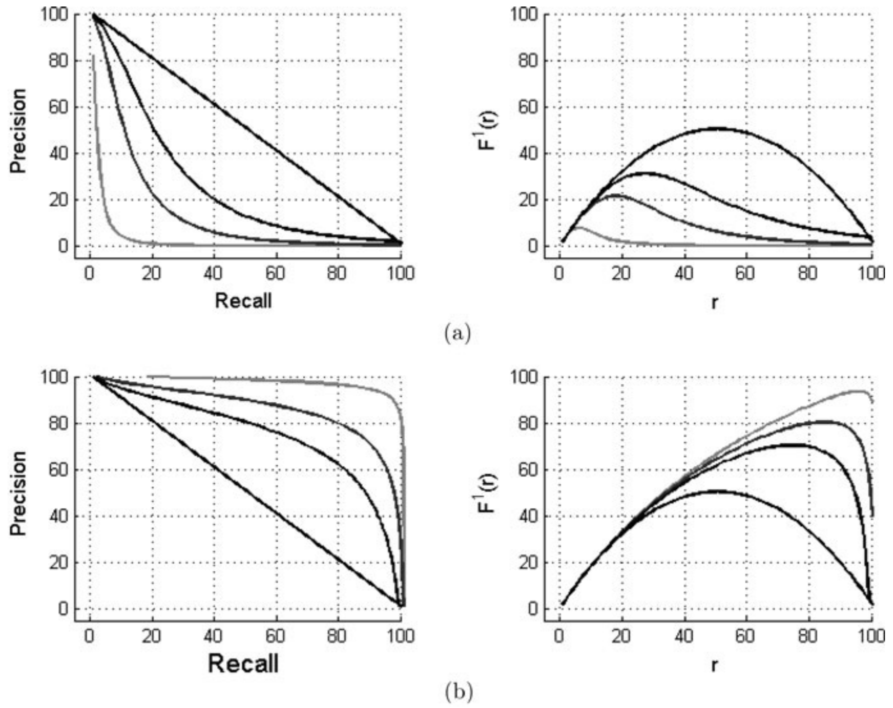


Fig. 7.1 $F^1(r)$-score plots for different synthetic precision and recall plots

The *fall-out* ratio *Fo* gives information about the number of non-relevant retrieved items with respect to the number of non-relevant items present in the collection. This measure is of special interest in unbalanced applications such as symbol spotting, where the number of elements which are not relevant is much larger than the number of relevant elements in the collection. Independent of the precision of a system, to consider the behavior of the system good, this measure should have low values either because very few non-relevant items have been retrieved or because the number of non-relevant retrieved items is negligible in relation to the number of non-relevant items in the dataset. To evaluate the evolution of the system response in terms of false positives, the fall-out is usually plotted against recall. This plot is equivalent to the typical *ROC* graphs [11] which are commonly used to evaluate the performance of classifiers. We can find a study of the relationship between precision-recall and *ROC* curves in [9].

Finally, the *generality* ratio *G* gives information about the collection dataset. It is computed as the number of relevant items in the entire collection for a certain query. It can then be averaged for all the considered queries in the experimental setup and be denoted as the *Ave G* ratio. This ratio does not give any measure about the effectiveness of the retrieval itself, but complements the previous measures. As claimed by Huijsmans and Sebe in [15], when evaluating the performance of a retrieval sys-

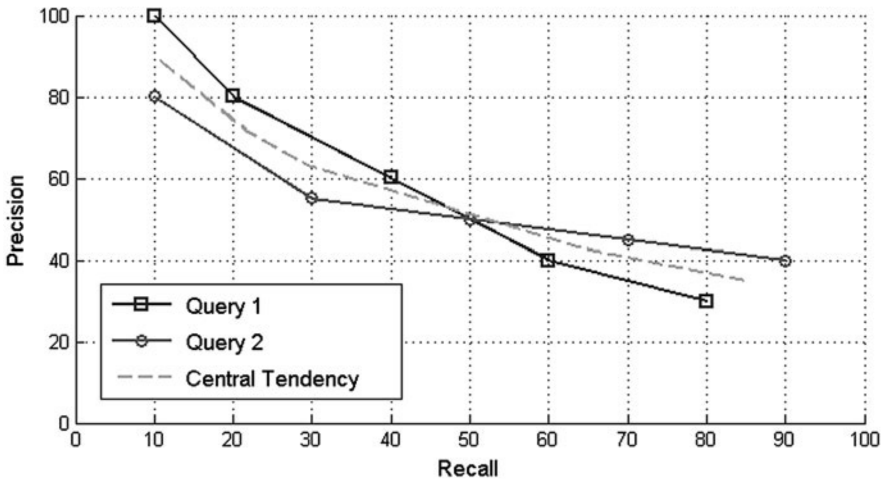


Fig. 7.2 An example of computing the central tendency of precision and recall plots

tem, this measure should be given to really understand the meaning of the values of precision, recall and fall-out.

7.3.6 Central Tendency of Precision and Recall

To evaluate a retrieval system, obviously, many queries have to be performed. Each query under evaluation results in a precision and recall plot. To give an idea of how well the system responds, the retrieval results are averaged over these queries. The central tendency of several precision and recall plots is computed, sampling individual curves at different points and averaging the samples. We can see an intuitive example of the central tendency in Fig. 7.2.

The same averaging technique is applied to fall-out versus recall plots and to $F^{\beta}(r)$-score plots.

7.4 Precision and Recall for Spotting Systems

Spotting systems are intended to perform both recognition and location at the same time, and thus, these abilities have to be evaluated together. Let us first propose a formulation of the precision and recall measures to evaluate both concepts. To help the interpretation of precision and recall plots, we propose using two more measures focused at symbol level, which only consider a binary concept of retrieval. Finally, we propose a scalability test to check the systems ability to achieve similar behavior independent of the number of queried symbols.

7.4.1 Precision and Recall of Regions of Interest

To evaluate the performance of a spotting system, we propose a set of measures inspired by both information retrieval and layout analysis. The idea is to merge both precision and recall measures with area overlapping rates. Precision and recall ratios provide information on the incremental accuracy of the retrieval process in terms of recognized items. On the other hand, the region overlapping between results and ground-truth data is used to evaluate the segmentation accuracy.

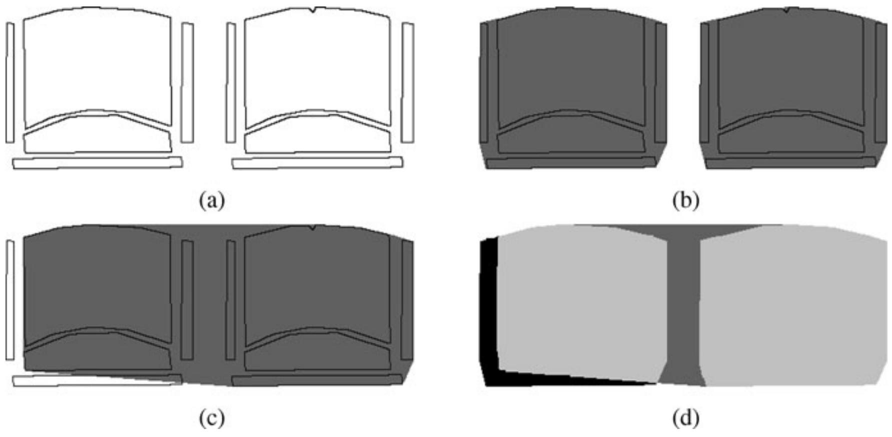
To compute the region overlapping between a result and ground-truth, for both data polygons we define representing regions of interest. The more accurate the definition of the region of interest is, the more reliable the evaluation. To define the region of interest where a symbol is located, we use the convex-hull algorithm presented in [7] of all the points belonging to the symbol. In the particular setup of Chapter 6, the graphical symbols are defined by the contours of the closed regions composing a symbol, so the convex-hull of the contour pixels englobe the whole symbol. Convex-hulls define the zones where a symbol is much more accurately than bounding-boxes or ellipses. This representation can be extended to different formats of the data of the collection (bitmap or vectorial format) and to different symbol representations (internal pixels, skeleton, contours, segments, etc.).

Given a collection of graphical documents, we denote as P*tot* the set of polygons representing the whole document image database. For any graphical symbol S to spot in the collection, we label as P*rel* the ground-truth polygon set which is composed of all the polygons framing the locations where we find an instance of the symbol S. When spotting the symbol S in the document collection, we denote by P*ret* the set of retrieved polygons. To match the results from the system to the ground-truth polygon set, we define the polygon set intersection operation $P_k = P_i \oplus P_j$ that, given two polygon sets P_i and P_j, results in a set of polygons from the spatial overlapping of the polygons belonging to the different sets. To measure the total amount of polygon overlapping, we define the function $A(P_i)$ as the sum of areas of all the polygons in the set P_i.

From the above sets and functions, precision and recall ratios can thus be easily formulated in terms of areas of the overlapping between sets of polygons representing results and ground-truth as follows:

$$\begin{aligned} P_A &= \frac{A(\mathrm{P}ret \oplus \mathrm{P}rel)}{A(\mathrm{P}ret)}, \\ R_A &= \frac{A(\mathrm{P}ret \oplus \mathrm{P}rel)}{A(\mathrm{P}rel)}. \end{aligned} \tag{7.7}$$

We can see an example of ground-truthed symbols and a result from a spotting system in Fig. 7.3. Some background region has been considered as forming part of the symbol. When we compute the overlapping between retrieved regions and relevant ones, this false positive region is identified, resulting in a precision decrease. On the other hand, some part of the symbol has been missed, this results in the recall value smaller than 100%.



$A(P_i)$	Precision (%)	Recall (%)
$A(\mathrm{P}\mathit{rel}) = 60{,}560$ $A(\mathrm{P}\mathit{ret}) = 62{,}307$ $A(\mathrm{P}\mathit{ret} \oplus \mathrm{P}\mathit{rel}) = 55{,}449$ $A(\overline{\mathrm{P}\mathit{ret}} \oplus \mathrm{P}\mathit{rel}) = 5{,}111$ $A(\mathrm{P}\mathit{ret} \oplus \overline{\mathrm{P}\mathit{rel}}) = 6{,}858$	88.99	91.56

(e)

Fig. 7.3 Overlapping between results and ground-truth. (**a**) Original image; (**b**) its ground-truth; (**c**) the result of a spotting system; (**d**) overlapping between results and ground-truth labeled according to $\mathrm{P}\mathit{ret} \oplus \mathrm{P}\mathit{rel}$ (*light gray*), $\mathrm{P}\mathit{ret} \oplus \overline{\mathrm{P}\mathit{rel}}$ (*dark gray*) or $\overline{\mathrm{P}\mathit{ret}} \oplus \mathrm{P}\mathit{rel}$ (*black*); (**e**) detailed areas and obtained precision and recall

7.4.2 Measures of Quality, Fall-out and Generality

Analogously, the measures of quality $Ave\ P$ and F-score, and the ratios of fall-out and generality can be expressed in terms of the area of the overlapping between polygon sets representing the ground-truth and the results from the spotting system.

We reformulate (7.2) by using the area precision at n ($P_A@n$), that is, by computing the area precision value after truncating the result list after each polygon having some overlapping with a polygon in the ground-truth. The average area precision is then computed as:

$$Ave\ P_A = \frac{\sum_{n=1}^{|\mathrm{P}\mathit{ret}|} (P_A@n \times \mathrm{r}(n))}{|\mathrm{P}\mathit{rel}|}. \tag{7.8}$$

By using the area precision and area recall, we reformulate the F-score from (7.3) as

$$F_A^{\beta} = \frac{(1+\beta^2) \times P_A \times R_A}{(\beta^2 \times P_A) + R_A}, \tag{7.9}$$

and, by using the area precision at a certain area recall cut-off ($P_A(r)$), we reformulate (7.5) as

$$F_A^{\beta}(r) = \frac{(1+\beta^2) \times P_A(r) \times r}{(\beta^2 \times P_A(r)) + r} \quad \text{with } r \in [0, R_A]. \tag{7.10}$$

Finally, if $\overline{\mathrm{P}rel}$ is the complementary polygon set for the ground-truth, we can reformulate the fall-out and the generality from (7.6) as

$$Fo_A = \frac{A(\mathrm{P}ret \oplus \overline{\mathrm{P}rel})}{A(\overline{\mathrm{P}rel})}, \qquad G_A = \frac{A(\mathrm{P}rel)}{A(\mathrm{P}tot)}. \tag{7.11}$$

7.4.3 Measures at Symbol Level

As pointed out by Lucas in [21], precision and recall based measures are sometimes difficult to interpret. A precision of 70% could mean that all symbols were found with an accuracy of 70%, or on the other hand, that only 70% of the symbols were correctly identified and the other 30% completely missed. A low precision value can be due to a low accuracy in the recognition or to a bad location due to over-segmenting. The recall value can also be affected by missed symbols or by under-segmentation.

To complement the precision and recall based measures, in our experiments, we also provide two measures focusing on the recognition at the symbol level. There we only consider a binary concept of retrieval, that is, whether a symbol is found or not. Let us consider one symbol S_i and its polygonal representation $\mathrm{P}rel_i$ from the ground-truth; it will be considered as recognized if

$$A(\mathrm{P}rel_i \oplus \mathrm{P}ret) \geq thr * A(\mathrm{P}rel_i), \tag{7.12}$$

that is, if the resulting polygons are able to overlap at least a certain percentage of the ground-truthed representation of a symbol, this symbol is considered as recognized. On the other hand, if the resulting polygons do not cover the ground-truth, the symbol is considered as missed. Of course, as with all decisions implying a certain threshold, its value can be critical, and the system's evaluation can depend on it. Its definition is completely subjective as it depends on what the user considers a symbol as being detected or not. The important thing here is that this value is provided when evaluating a system, so that the readers can easily interpret the meaning of the evaluation results. In our case, we consider a symbol as detected if it overlaps at least 75% of the ground-truth area.

At the symbol level, we derive the *recognition rate* of the spotting system under study. In addition, if one of the polygons $\mathrm{P}ret_j$ in the resulting set does not overlap with any recognized symbol, it is considered a *false positive*. For all the possible queries, the average of false positives $AveFP$ is computed. These two measures help to better interpret the values of precision and recall.

Notice that the recognition rate is expressed as a percentage of the total number of symbols in the ground-truth and can be used as a measure of quality by itself, but the false positives are not normalized and are given in absolute values. This is due to the fact that we cannot define the negative set in terms of symbol items in the dataset. The false positive average can only be expressed in absolute values and used to compare methods between them.

7.4.4 Scalability Test

Finally, one of the main interests for spotting systems is that a system has to be applicable to a large data corpora. To test the scalability of the system, i.e., its ability to achieve similar behavior independently of the number of queried symbols, we propose a measure to evaluate the scalability of the systems under study.

A scalable system has to yield similar responses no matter what the number of model classes taken into account is. We can measure the scalability of a system in terms of its variance in both precision and recall. Let us consider the synthetic example of Fig. 7.4a which is highly damaged by the addition of new classes. Let us define std_R and std_P, the standard deviations in precision and recall, for a certain sampling of the precision and recall plot. In Fig. 7.4b, we can see the central tendency of all precision and recall plots with error bars following the vertical and horizontal axes to check the effect in both precision and recall measures when considering more and more classes. The greater the deviation, the worse the system tolerates changes in the class number; thus the system can be considered as less scalable. To allow an easier interpretation, the standard deviation can be computed for the $F_A^\beta(r)$-score plots (as shown in Fig. 7.4c) having now a single variance measure instead of one for precision and one for recall. To compare the scalability between different methods, both the mean $\overline{std}$ of all the samples of the standard deviation and the maximum $\max(std)$ of all the samples of the standard deviation are given as variance measures.

On the other hand, the performance of a spotting system is not only affected by the increasing number of considered models but also on the size of the document collection. To observe how the system degrades with the expansion of the dataset, we propose to work at the symbol level. Recognition rates and false alarms are given to illustrate the performance variability in relation to the size of the database. These measures help predict how the performance of a system will be affected by the inclusion of more documents in the database. However, increasing the database size has an important drawback. When adding new documents into the database, we can implicitly be adding new graphical symbols contained in these new documents. As a consequence of this, also the number of model symbols has to be increased along with the dataset size.

7.5 Evaluating a Symbol Spotting System

In this section, to show an example of the application of the presented evaluation framework, we tested a symbol spotting architecture. We first explain the ground-

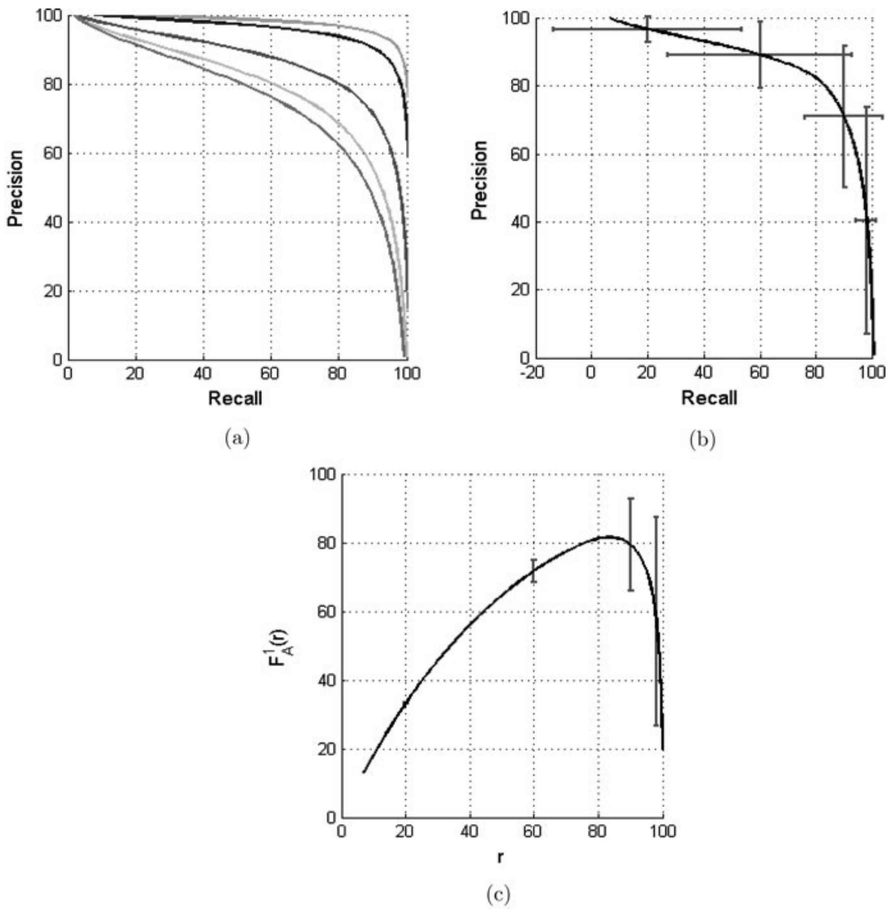
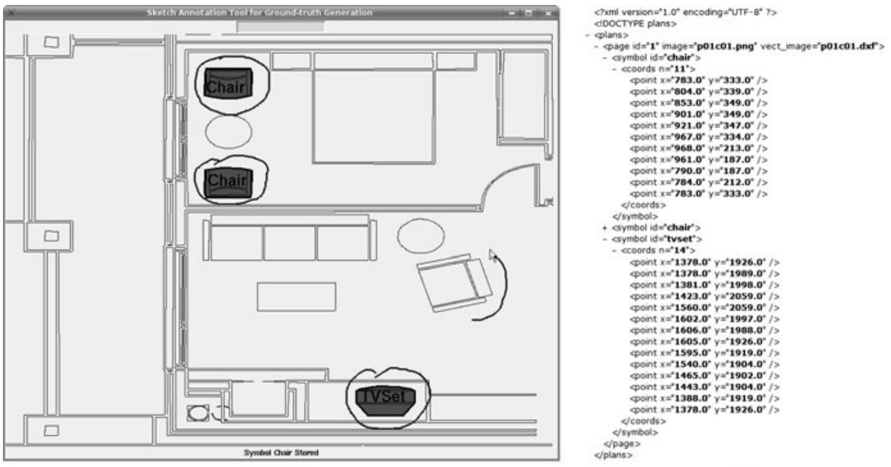


Fig. 7.4 Scalability test example. (**a**) Synthetic precision and recall plots; (**b**) averaged precision and recall plot with standard deviations in recall and precision; (**c**) averaged $F_A^1(r)$-score plot with associated standard deviations

truthing process, then we briefly detail the spotting system and the used dataset, and finally we provide the evaluation results for this architecture.

7.5.1 Ground-Truthing

First, an annotation tool has been developed to build the ground-truth. The user can select graphical entities in the document images roughly segmenting them using a sketching application. All the contour pixels falling inside the delimited zone of interest are taken as being part of the symbol. If a given connected component has more pixels outside the zone of interest than inside, it is considered as being part of the background. This basic annotation tool works fine with architectural drawings



```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE plans>
- <plans>
  - <page id="1" image="p01c01.png" vect_image="p01c01.dxf">
    - <symbol id="chair">
      - <coords n="11">
          <point x="783.0" y="333.0" />
          <point x="804.0" y="339.0" />
          <point x="853.0" y="349.0" />
          <point x="901.0" y="349.0" />
          <point x="921.0" y="347.0" />
          <point x="967.0" y="334.0" />
          <point x="968.0" y="213.0" />
          <point x="961.0" y="187.0" />
          <point x="790.0" y="187.0" />
          <point x="784.0" y="212.0" />
          <point x="783.0" y="333.0" />
        </coords>
      </symbol>
    + <symbol id="chair">
    - <symbol id="tvset">
      - <coords n="14">
          <point x="1378.0" y="1926.0" />
          <point x="1378.0" y="1989.0" />
          <point x="1381.0" y="1998.0" />
          <point x="1423.0" y="2059.0" />
          <point x="1560.0" y="2059.0" />
          <point x="1602.0" y="1997.0" />
          <point x="1606.0" y="1988.0" />
          <point x="1605.0" y="1926.0" />
          <point x="1595.0" y="1919.0" />
          <point x="1540.0" y="1904.0" />
          <point x="1465.0" y="1902.0" />
          <point x="1443.0" y="1904.0" />
          <point x="1388.0" y="1919.0" />
          <point x="1378.0" y="1926.0" />
        </coords>
      </symbol>
    </page>
</plans>
```

(a) (b)

Fig. 7.5 Sketching annotation tool for ground-truth generation. (**a**) Graphical interface; (**b**) the generated ground-truth XML file

where the symbols are usually not extensively connected with background elements. For other kinds of documents, e.g., electronic diagrams or geographical maps, the annotation tool should be enhanced in order to provide a trusted ground-truth. For all the foreground pixels, we compute the convex-hull (as presented in [7]) as the minimum area of interest which contains the symbol. Once the region of interest is shown, the user can modify it using certain control points and label them by their content. We can see a screen-shot of the sketching application in Fig. 7.5a. The use of convex-hulls as the ground-truth primitive may be inadequate for some spotting systems. The inclusion of noisy pixels in the spotting results may provoke considerable deviations of the convex-hull from the one defined in the ground-truth. However, the presented evaluation measures can be easily adapted to other choices of ground-truth primitives. From coarser to more refined primitives, we can select, for instance, to use bounding-boxes, ellipses, isothetic polygons, quad-trees, etc. as ground-truth primitives. In all these cases, the computation of the overlap between ground-truth and automatically extracted primitives is straightforward.

As the user labels the regions containing the graphical symbols, an XML file is constructed to store the information about the whole library. Following the same file structure used for page layout ground-truth presented in [5], the convex-hull coordinates and the symbol category as well as other information about the document are organized in the XML file which we can see in Fig. 7.5b.

As claimed in [19], creating a ground-truth for graphic documents is not always straightforward due to ambiguous cases or subjectivity issues. For example, in the architectural field, each architect tends to use its own symbol designs to represent a furniture element. Whereas human observers have no difficulty in clustering these elements despite the design differences, it is usually impossible for a spotting system to be able to identify different designs as the same object. In the process of ground-truth building, we tried to avoid such problems, but we believe that the use of a collaborative framework as proposed in [36] would enhance a lot the quality and the accuracy of this ground-truth. To avoid subjective decisions on the ground-truthing process, synthetic ground-truth can be generated for graphic rich documents, as recently presented in [10]. Such tools which synthetically generate ground-truthed data present several interesting advantages. Subjective decisions are avoided since no human interaction is needed, thus providing an error-free labeling of graphical items. In addition, we have complete control on the number of items in the collection and the number of symbols which have to appear in each document, making the scalability tests much more easy and reliable. However, nowadays the data generated by these methods still appears quite artificial and the use of real data (when possible) should be preferred.

7.5.2 Spotting Methods Under Test

The symbol spotting architecture we use to test the evaluation measures is based on the relational indexing scheme presented in Chapter 6. Summarizing, symbols are decomposed into basic primitives which are subsequently described by a geometric symbol description technique. The feature vectors arising from the description are indexed with a hashing technique. When querying this hash table, structural information is added by means of a relational indexing technique. That is, only similar primitives sharing the same spatial relationship are retrieved. One of the most important points of the system is the way graphical primitives are described to be indexed. We tested four off-the-shelf geometric symbol descriptors described below.

- **Method a** uses a set of simple ratios described in [32] such as the eccentricity or the non-circularity as shape descriptors. These rough descriptors are formulated from the shape contour of the symbol's primitives. It is expected that the use of such simple shape description can only discriminate very dissimilar shapes; the system should result in a lot of false alarms, but should be tolerant to distortions and thus retrieve almost all the instances of the queried symbol.
- **Method b** uses Hu's geometric invariants [14] to describe contours. These invariants are known as good shape descriptors. The expected performance is to have good spotting rates in all aspects.
- **Method c** is based on a reformulation of the previous one. Geometric moments can be formulated for polygonally approximated contours [18] which are taken as primitives. In this case, the use of simpler primitives should result in smaller tolerance to distortions.
- **Method d** uses the Fourier transform to compactly represent a curvature signature computed over the shape contour. This descriptor is detailed in [17]. This is also a good shape descriptor, and the system performance is expected to be good in all aspects.

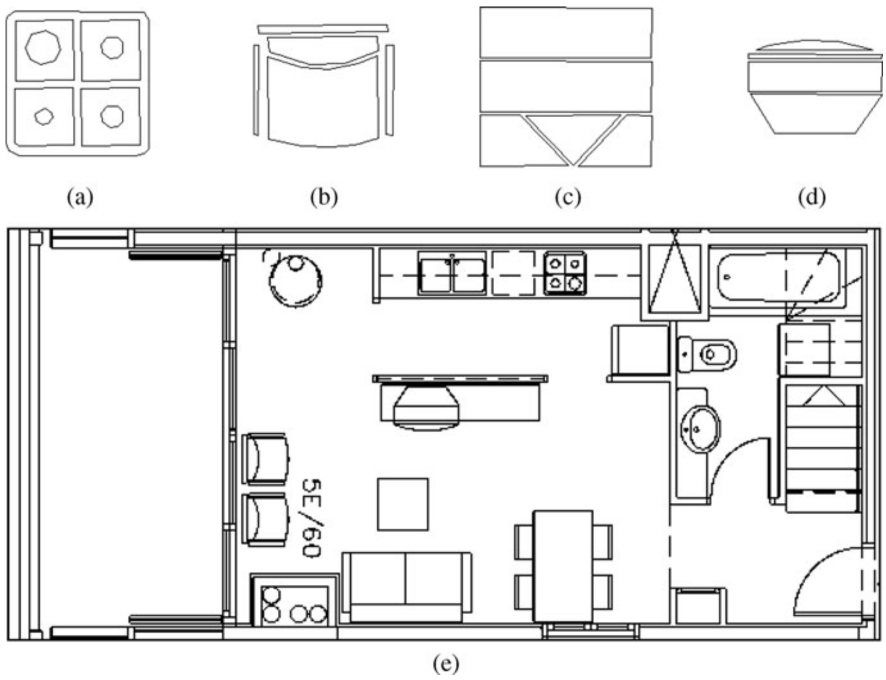
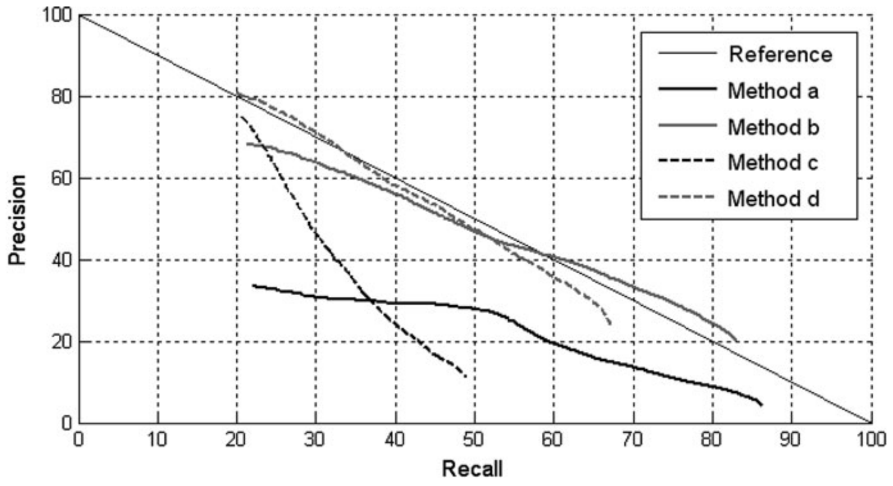


Fig. 7.6 Symbol models and an example of a document in the database. (**a**) Burner symbol; (**b**) chair symbol; (**c**) stairs symbol; (**d**) TV set symbol; (**e**) sample document

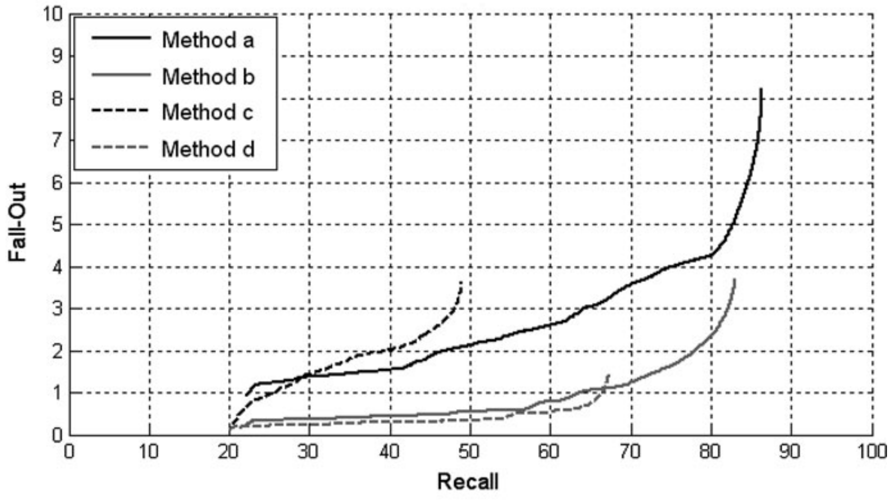
Note that we do not want to perform an exhaustive evaluation of shape descriptors or primitives. These methods have been chosen because of their different nature and to test if the proposed evaluation measures really determine the strong and weak points of each method. As the descriptors are well-known among the Graphics Recognition community, it is easy to assess whether the results correspond to the expected behaviors.

7.5.3 The FPLAN-POLY Dataset

The dataset is a collection of architectural floor-plans consisting of 42 images (of 3,215 × 2,064 pixels in average) arising from four different projects. Any given furniture symbol appears in several images in the database. The symbols taken into account for these experiments are divided into 38 classes, and we have in total 344 instances in the document images. In a single document image, the average of symbols is around 8, and the range is from 0 to 28 symbols. The models to query the document database are cropped from the document images. We can see some examples of model symbols as well as a sample document from the database in Fig. 7.6. More details of this dataset are given in Appendix A.



(a)

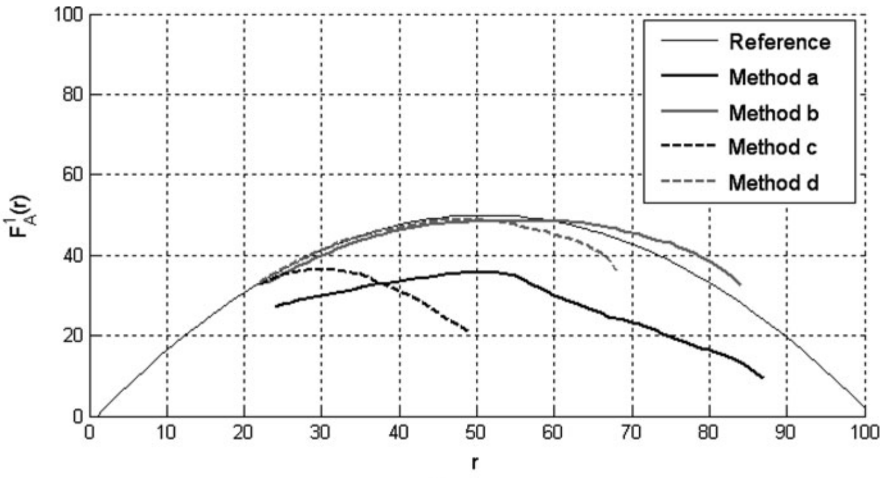


(b)

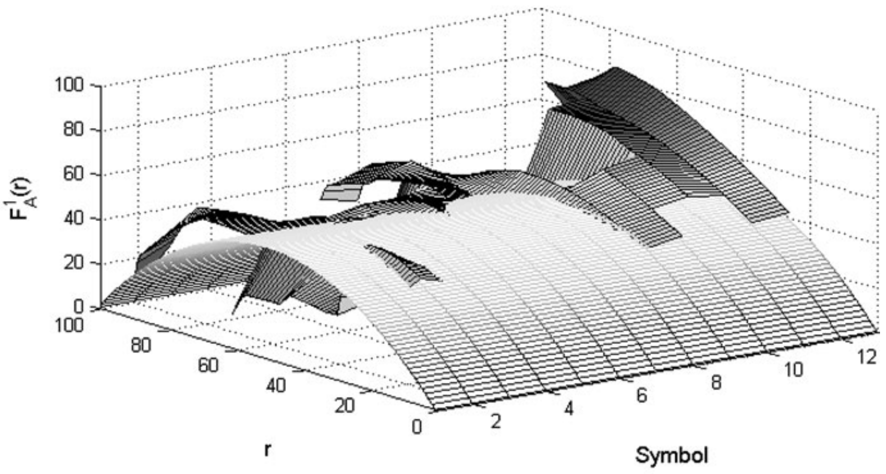
Fig. 7.7 (**a**) Precision versus recall; (**b**) fall-out versus recall

7.5.4 Evaluation

We first present the plots showing precision versus recall and fall-out versus recall in Fig. 7.7 for all the four spotting methods under evaluation using the whole collection of documents. Methods b and d show an acceptable tradeoff between precision and recall as expected. Method d misses many more symbols than method b but gives a significantly smaller number of false positives. Method a yields good recall values, i.e., it succeeds in retrieving most of the symbols in the document database but has



(a)



(b)

Fig. 7.8 (**a**) $F_A^1(r)$-score plot for all methods under test; (**b**) $F_A^1(r)$-score plot depending on the queried symbol for method b

poor precision due to the larger number of false positives. Finally, method d shows good precision values at early recall stages but quickly falls missing more than half of the symbols in the dataset. The proposed measures aim to stress the expected good behavior of methods b and d and to point out the simplicity of method a as well as the lack of tolerance of method c.

We can observe the $F_A^1(r)$-score plots in Fig. 7.8a. In this graph, we can see again the clear dominance of methods b and d over the other two. As the F-score combines both precision and recall, the methods which fail in one of those measures are clearly demoted in the overall evaluation. Method a starts with a low precision

Table 7.2 Measures of quality

Method	*Ave* P_A	F_A^1-score	Rec. rate (%)	*Ave* FP	*Ave* G_A (%)
Ratios	20.08	6.87	**93.62**	153.42	
Boundary	39.77	**23.34**	91.3	76.76	0.16
Line	23.69	12.57	55.62	63.89	
Fourier	**41.99**	21.45	73.33	**58.76**	

Table 7.3 Scalability test details

Method	Recall		Precision		F-score	
	$\max(std_R)$	$\overline{std_R}$	$\max(std_P)$	$\overline{std_P}$	$\max(std_F)$	$\overline{std_F}$
Ratios	11.37	6.06	4.2	2.32	4.09	2.2
Boundary	3.38	2.22	**2.01**	**1.74**	1.84	0.98
Line	**1.6**	**1.02**	2.64	2.18	2.17	1.21
Fourier	2.66	1.96	3.44	2.46	**1.09**	**0.85**

value, while the precision of method c quickly falls stopping at a 50% recall. Those two methods are clearly at a disadvantage as expected. In Fig. 7.8b, we see how we can use the $F_A^1(r)$-score plots to visually check the variance of performance of a given method depending on the symbol the user queries.

In Table 7.2, we can see the measures of quality for all the methods. As the average precision *Ave* P_A measure does not take into account the recall, the method d is ranked as the best. On the other hand, F_A^1-score gives the best mark for method b. The measures working at the symbol level, which are intended to evaluate only the recognition task, are consistent with the results shown in Fig. 7.7b. The number of recognized symbols is related to the recall value, which ranks the methods in the order a, b, d and c, in terms of the amount of correctly retrieved elements. On the other hand, the average of false positives is related to the fall-off ratio, ranking the methods in the order d, c, b and a, in terms of the false alarms present in the results. Finally, the averaged generality gives an idea of the proportion between relevant and total elements in the dataset. These last measures aim at interpreting the precision, recall and fall-out values. For spotting applications, it is typical to have an extremely low generality measure since the documents in the collections will usually have much more background objects than foreground ones. This low generality explains the low precision values in both precision and recall plots and in the average precision *Ave* P_A indicator.

Finally, the scalability test results are shown in Figs. 7.9 and 7.10. Several sets of symbol classes are considered ranging from only 5 to 35 possible symbols to query. We randomly selected n symbols from the dataset and computed the average precision and recall for these queries. This experiment has been repeated 100 times for the sake of stability and the averaged curves are presented in Figs. 7.9a and 7.10a. First, in Figs. 7.9b and 7.10b, we notice that the changes in the number

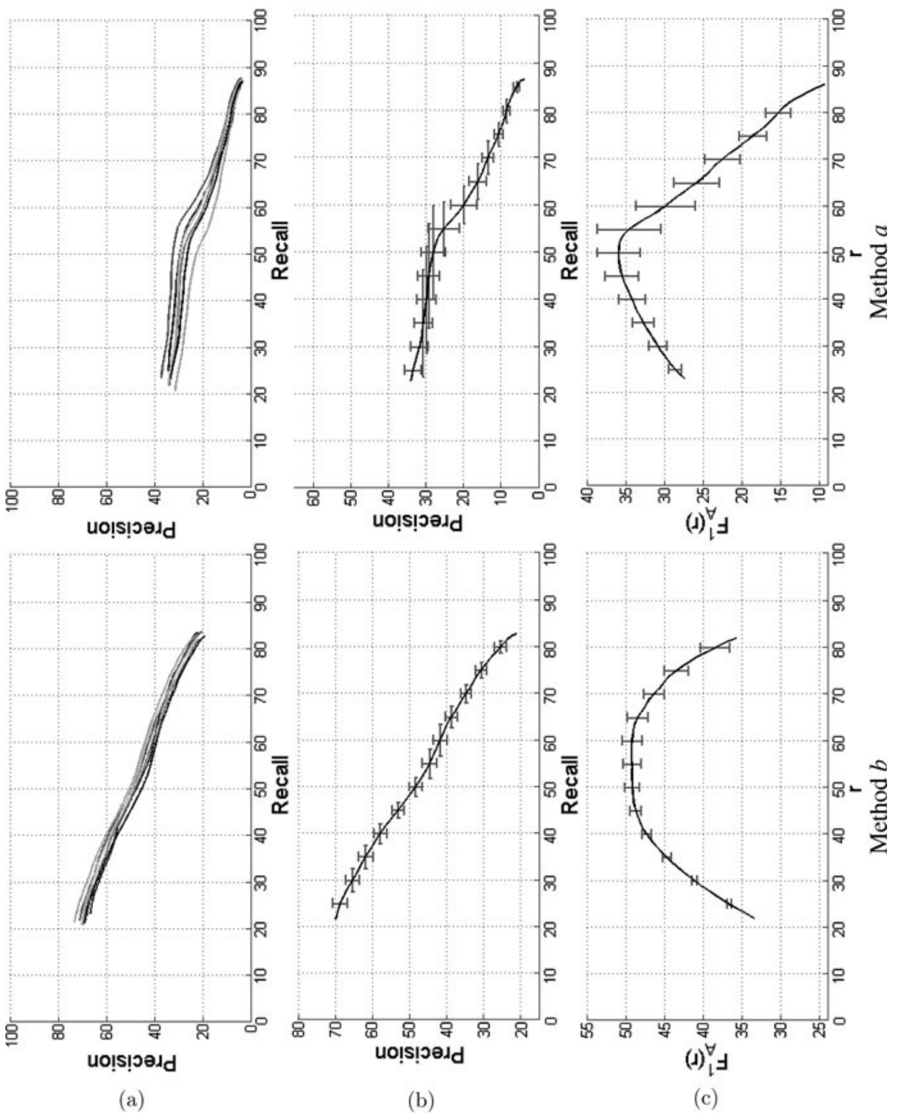


Fig. 7.9 Scalability tests for the two first methods. (**a**) Precision recall plots for several amount of symbol classes; (**b**) averaged precision and recall plot with standard deviations following vertical (precision) and horizontal (recall) axis; (**c**) averaged $F_A^1(r)$-score plot with associated standard deviations

of classes affect different properties depending on the method. The recall of method *a* drastically decreases when introducing more and more symbol classes, whereas the precision of method *c* suffers much more than the recall. On the other hand, methods *b* and *d* seem to be equally affected by changes in scale in both precision and recall. From Figs. 7.9c and 7.10c, we can see how the variations in the

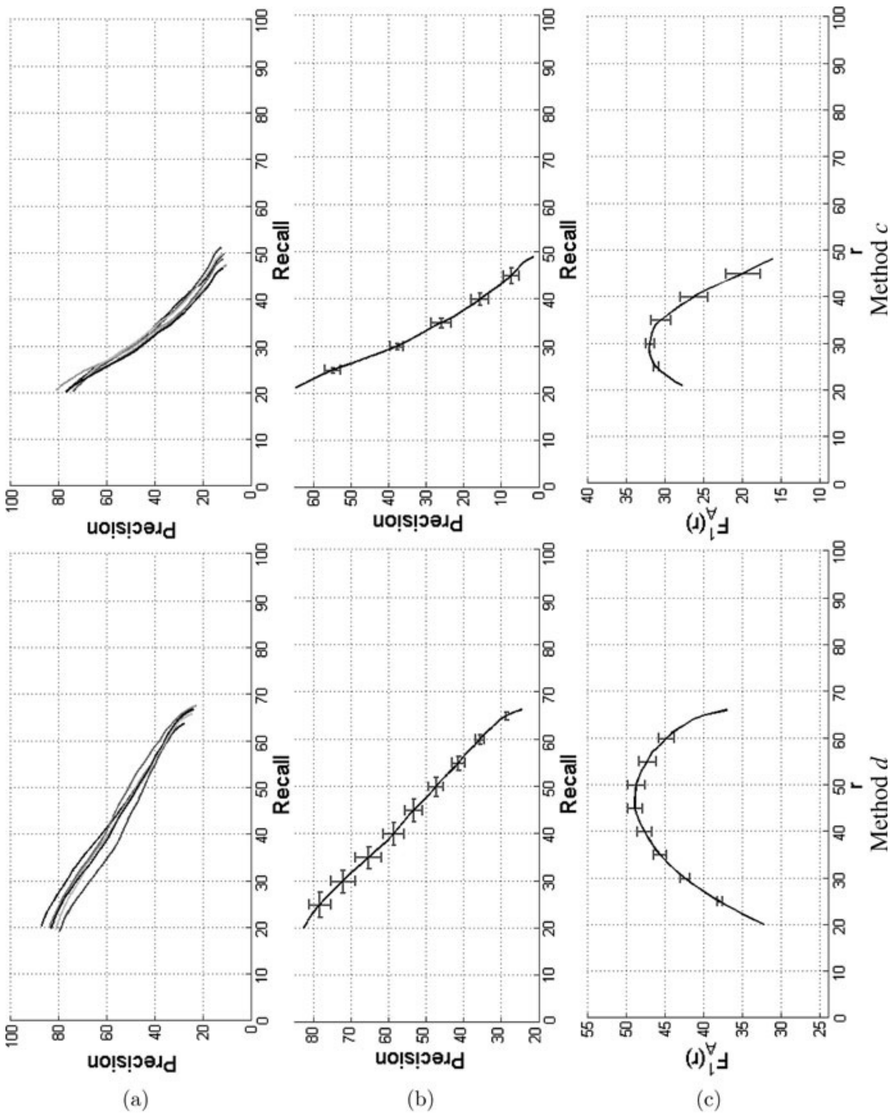
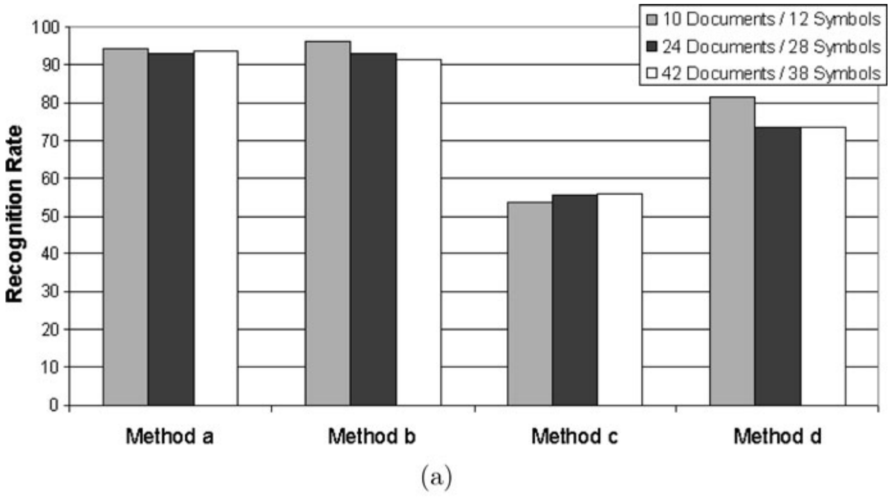


Fig. 7.10 Scalability tests for the two last methods. (**a**) Precision recall plots for several amount of symbol classes; (**b**) averaged precision and recall plot with standard deviations following vertical (precision) and horizontal (recall) axis; (**c**) averaged $F_A^1(r)$-score plot with associated standard deviations

$F_A^1(r)$-score space are good indicators of the scalability of the methods under study. We can see the quality indicators for scalability tests in Table 7.3. We present the mean of the standard deviations and its maxima. Again, methods *b* and *d* show much more scalability than methods *c* and *a* when looking at the composite measure. Finally, Fig. 7.11a and b show the scalability test at the symbol level when increasing



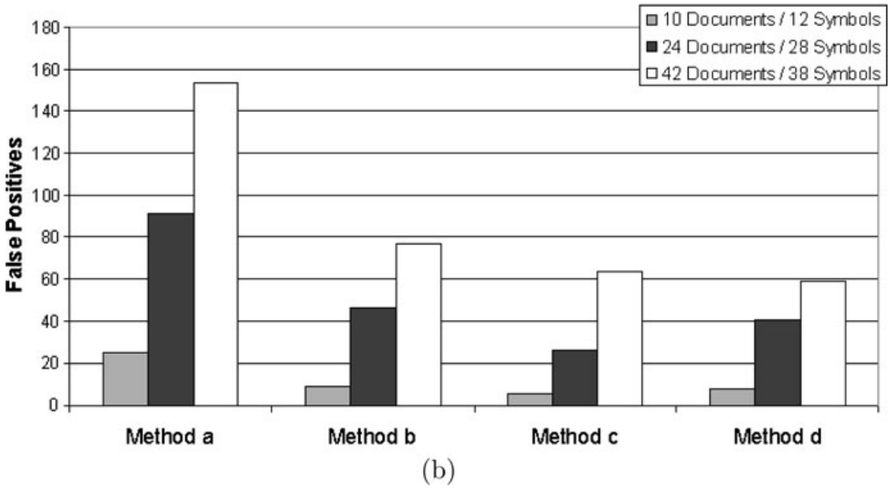


Fig. 7.11 Scalability tests at symbol level. (**a**) Recognition rates; (**b**) false positives

both the number of models and the dataset size. As we can observe, the recognition rates vary slightly, whereas the number of false alarms is exponentially increased in all the cases along with the dataset size.

From these results, we can conclude that methods *b* and *d* seem to be much better than the other two. Method *b* should be chosen when we desire to retrieve as many symbols as possible, and on the other hand, method *d* is suitable if we want to reduce the number of false positives. Method *a* should only be chosen if the presence of false positives is not a problem and the user prioritizes finding all the positive symbols despite the presence of false positives. However, its performance seems to be affected by the number of considered symbols. Finally, method *c* is only suitable if we are interested in retrieving positive symbols at the first positions of the ranked retrieved locations even if we completely miss the rest of the symbols. Methods b and d also tolerate well changes in the number of considered classes and should be considered when facing applications involving a large amount of data. On the other hand, the strong points of methods a and c are compromised when introducing more and more symbol classes. All these conclusions are in accordance with the expected behavior of the studied methods, showing that the proposed evaluation protocol emphasizes the expected strengths and weaknesses of the methods under study.

7.6 Conclusions and Discussion

Times where algorithms were tested with a small set of data are over. Nowadays, it is necessary to use the standard reference ground-truth and performance evaluation protocols. The Graphics Recognition community is one of the most healthy communities within the Pattern Recognition field regarding this aspect. A lot of works and efforts are centered on proposing evaluation methods which aim tracking the progress in a certain specific problem. As far as we know, the works focused on symbol spotting always have been evaluated by an ad-hoc set of measures. We hope that the proposal of the performance evaluation protocol presented in this part of the book can be used to evaluate other spotting methods.

One of the main problems of evaluating spotting methods is that we do not have any public dataset of real documents to test the proposed methods. Nowadays, the only available ground-truthed dataset which can be used to test spotting and focused retrieval of graphics is the one proposed by Delalandre et al. in [10]. The main problem of this dataset is that it is composed only of synthetical generated documents which do not seem realistic, yet. We preferred to evaluate our work on a set of real documents.

One of the main criticisms of using precision and recall to evaluate the performance of classification and location tasks is that it is sometimes difficult to really asses the behavior of the system under study. As claimed in [21], a low precision value can be due to a low accuracy in the recognition or to a bad localization due to over-segmenting. In addition, as pointed out in [39], the amount of overlap between polygons seems not to be a perceptively valid measure of quality. Quality indicators as the F-score have been also questioned, in [23] it is argued that this measure makes the systems look like they are much better than they really are.

We believe that the presented measures are able to evaluate well the behavior of symbol spotting and focused retrieval systems, emphasizing their strong and weak points, and their tolerance to changes in scale. Precision is sometimes hard to interpret or does not provide perceptively good indicators, but the point of a spotting system is to retrieve zones of interest of document images, and the presented measures aim to measure the system's ability to do this task. Quality indicators aim to rank the methods according to certain ability, so even if the numbers by themselves do not have an accurate absolute meaning they are useful to compare methods between them. Finally, precision and recall are enhanced by measures working only at symbol level and the generality factor which helps to interpret the meaning of the plots. As shown in the evaluation section, the results obtained by using the proposed evaluation protocol are consistent with the ratios working at symbol recognition level, and most importantly, emphasize the expected strengths and weaknesses of the methods under study.

References

1. Aksoy, S., Ye, M., Schauf, M., Song, M., Wang, Y., Haralick, R., Parker, J., Pivovarov, J., Royko, D., Changming, S., Farneback, G.: Algorithm performance contest. In: Proceedings of the Fifteenth International Conference on Pattern Recognition, pp. 870–876. IEEE Computer Society, Los Alamitos (2000)
2. Antonacopoulos, A., Bridson, D.: Performance analysis framework for layout analysis methods. In: Proceedings of the Ninth International Conference on Document Analysis and Recognition, pp. 1258–1262. IEEE Computer Society, Los Alamitos (2007)
3. Antonacopoulos, A., Gatos, B., Karatzas, D.: ICDAR 2003 page segmentation competition. In: Proceedings of the Seventh International Conference on Document Analysis and Recognition, pp. 688–692. IEEE Computer Society, Los Alamitos (2003)
4. Antonacopoulos, A., Gatos, B., Bridson, D.: ICDAR 2005 page segmentation competition. In: Proceedings of the Eighth International Conference on Document Analysis and Recognition, pp. 75–79. IEEE Computer Society, Los Alamitos (2005)
5. Antonacopoulos, A., Karatzas, D., Bridson, D.: Ground truth for layout analysis performance evaluation. In: Document Analysis Systems, DAS06, *Lecture Notes on Computer Science*, vol. 3872, pp. 302–311. Springer, Berlin (2006)
6. Antonacopoulos, A., Gatos, B., Bridson, D.: ICDAR 2007 page segmentation competition. In: Proceedings of the Ninth International Conference on Document Analysis and Recognition, pp. 1279–1283. IEEE Computer Society, Los Alamitos (2007)
7. Barber, C., Dobkin, D., Huhdanpaa, H.: The quickhull algorithm for convex hulls. ACM Transactions on Mathematical Software **22**(4), 469–483 (1996)
8. Buckland, M., Gey, F.: The relationship between recall and precision. Journal of the American Society for Information Science **45**(1), 12–19 (1994)
9. Davis, J., Goadrich, M.: The relationship between precision-recall and ROC curves. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 233–240. Omnipress (2006)
10. Delalandre, M., Pridmore, T., Valveny, E., Locteau, H., Trupin, E.: Building synthetic graphical documents for performance evaluation. In: Graphics Recognition. Recent Advances and New Opportunities, *Lecture Notes on Computer Science*, vol. 5046, pp. 288–298. Springer, Berlin (2008)
11. Fawcett, T.: An introduction to ROC analysis. Pattern Recognition Letters **27**(8), 861–874 (2006)
12. Holz, F., Witschel, H., Heinrich, G., Heyer, G., Teresniak, S.: An evaluation framework for semantic search in P2P networks. In: Proceedings of the Seventh International Workshop on Innovative Internet Community Systems (2007)
13. Hripcsak, G., Rothschild, A.: Agreement, the F-measure, and reliability in information retrieval. Journal of the American Medical Informatics Association **12**(3), 296–298 (2005)
14. Hu, M.: Visual pattern recognition by moment invariants. IRE Transactions on Information Theory **8**, 179–187 (1962)
15. Huijsmans, D., Sebe, N.: How to complete performance graphs in content-based image retrieval: Add generality and normalize scope. IEEE Transactions on Pattern Analysis and Machine Intelligence **27**(2), 245–251 (2005)

16. Kang, B., Kim, H., Lee, S.: Performance analysis of semantic indexing in text retrieval. In: Computational Linguistics and Intelligent Text Processing, *Lecture Notes on Computer Science*, vol. 2945, pp. 433–436. Springer, Berlin (2004)
17. Kauppinen, H., Seppänen, T., Pietikäinen, M.: An experimental comparison of autoregressive and Fourier-based descriptors in 2D shape classification. IEEE Transactions on Pattern Analysis and Machine Intelligence **17**, 201–207 (1995)
18. Lambert, G., Gao, H.: Discrimination properties of invariants using the line moments of vectorized contours. In: Proceedings of the 13th International Conference on Pattern Recognition, pp. 735–739. IEEE Computer Society, Los Alamitos (1996)
19. Lopresti, D., Nagy, G.: Issues in ground-truthing graphic documents. In: Graphics Recognition Algorithms and Applications, *Lecture Notes on Computer Science*, vol. 2390, pp. 46–66. Springer, Berlin (2001)
20. Lu, C., Shukla, M., Subramanya, S., Wu, Y.: Performance evaluation of desktop search engines. In: Proceedings of the IEEE International Conference on Information Reuse and Integration, pp. 110–115. IEEE Computer Society, Los Alamitos (2007)
21. Lucas, S.: ICDAR 2005 text locating competition results. In: Proceedings of the Eighth International Conference on Document Analysis and Recognition, pp. 80–84. IEEE Computer Society, Los Alamitos (2005)
22. Lucas, S., Panaretos, A., Sosa, L., Tang, A., Wong, S., Young, R., Ashida, K., Nagai, H., Okamoto, M., Yamamoto, H., Miyao, H., Zhu, J., Ou, W., Wolf, C., Jolion, J., Todoran, L., Worring, M., Lin, X.: ICDAR 2003 robust reading competitions: Entries, results, and future directions. International Journal on Document Analysis and Recognition **7**(2), 105–122 (2005)
23. Makhoul, J., Kubala, F., Schwartz, R., Weischedel, R.: Performance measures for information extraction. In: Proceedings of DARPA Broadcast News Workshop. Morgan Kaufmann, New York (1999)
24. Marcus, J.: A novel algorithm for HMM word spotting performance evaluation and error analysis. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 89–92. IEEE Computer Society, Los Alamitos (1992)
25. Müller, H., Müller, W., Squire, D., Marchand-Maillet, S., Pun, T.: Performance evaluation in content-based image retrieval: Overview and proposals. Pattern Recognition Letters **22**(5), 593–601 (2001)
26. Neumann, T., Bender, M., Michel, S., Weikum, G.: A reproducible benchmark for P2P retrieval. In: Proceedings of the First International Workshop on Performance and Evaluation of Data Management Systems, pp. 1–8 (2006)
27. Phillips, I., Chhabra, A.: Empirical performance evaluation of graphics recognition systems. IEEE Transactions on Pattern Analysis and Machine Intelligence **21**(9), 849–870 (1999)
28. 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)
29. van Rijsbergen, C.: Information Retrieval. Butterworth-Heinemann Newton, MA, USA (1979)
30. Smeaton, A., Over, P., Kraaij, W.: Evaluation campaigns and TRECVid. In: Proceedings of the Eighth ACM International Workshop on Multimedia Information Retrieval, pp. 321–330. ACM, New York (2006)
31. Song, J., Su, F., Tai, C., Cai, S.: An object-oriented progressive-simplification-based vectorization system for engineering drawings: Model, algorithm, and performance. IEEE Transactions on Pattern Analysis and Machine Intelligence **24**(8), 1048–1060 (2002)
32. Stoyan, D., Stoyan, H.: Fractals, Random Shapes and Point Fields (Methods of Geometrical Statistics). Wiley, Chichester (1994)
33. 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)
34. 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)

35. Valveny, E., Dosch, P.: Report on the second symbol recognition contest. In: Graphics Recognition. Ten Years Review and Future Perspectives, *Lecture Notes on Computer Science*, vol. 3926, pp. 381–397. Springer, Berlin (2006)
36. Valveny, E., Dosch, P., Winstanley, A., Zhou, Y., Yang, S., Yan, L., Wenyin, L., Elliman, D., Delalandre, M., Trupin, E., Adam, S., Ogier, J.: A general framework for the evaluation of symbol recognition methods. International Journal on Document Analysis and Recognition **9**(1), 59–74 (2007)
37. Wenyin, L., Dori, D.: A protocol for performance evaluation of line detection algorithms. Machine Vision and Applications **9**(5), 240–250 (1997)
38. Wenyin, L., Dori, D.: Incremental arc segmentation algorithm and its evaluation. IEEE Transactions on Pattern Analysis and Machine Intelligence **20**(4), 424–431 (1998)
39. Wolf, C., Jolion, J.: Object count/area graphs for the evaluation of object detection and segmentation algorithms. International Journal on Document Analysis and Recognition **8**(4), 280–296 (2006)

Chapter 8
Conclusions

Abstract In this chapter, we summarize the contributions of this book to the symbol spotting problem and, in particular, to the application of focused retrieval of graphical symbols from collections of line-drawing images. We also present a discussion and the limitations of the presented approaches. We finally point some possible lines of continuation on the field of symbol spotting and some improvements of the proposed methods which should be further studied.

8.1 Summary of Contributions

In this book, we have introduced a complete framework for symbol spotting and, in particular, for a focused retrieval application. As explained in Chapter 1, our work has been motivated by the specific problem of proposing a spotting methodology able to locate and retrieve graphical content within a database of complete document images. A lot of interest is shown worldwide in mass digitization of document collections and their storage in digital libraries. This results in digital repositories rich in information provided they are semantically accessible. Although such semantic access has been improved a lot for textual queries, iconic access is still in early stages, especially when dealing with documents rich in graphical information like technical documents. This is the starting hypothesis of the work developed in this book. From a methodological point of view, the main challenges stem from the nature of the queries which have to be iconic queries instead of the ASCII strings used in the keyword-based searches. In addition to the nature of the queries, the retrieval of the relevant zones should be done on-the-fly. In our framework, the system is queried by example, that is, the user segments a symbol he wants to retrieve from the document database and this cropped image acts as the input. This particularity reinforces the fact that the proposed spotting methods are not meant to work for a specific set of model symbols nor have a learning stage where the relevant features describing a certain symbol can be trained. The use of data structures having graphical patterns as indices so as to provide an efficient access to the graphic information contained in large data corpora is a must in focused retrieval applications.

We have identified three different levels when conceiving a spotting architecture. The first level aims at representing and compactly describing the primitives that compound the graphical symbols. In the second level, these features describing graphical symbols are organized in a particular data structure. This data structure should be chosen carefully in order to provide efficient access to the symbol descriptors. During the querying process, the data structure is traversed, and the locations within the document images where to find similar primitives as the queried ones are retrieved. The third level consists of a validation stage to determine the valid hypotheses where the queried symbol is likely to be found. Throughout this book, we have made some contributions in each of the three stages. Let us briefly summarize these contributions.

- **Extraction of Vectorial Primitives** Part III of the book has focused on the use of geometric and structural description techniques to describe the graphical symbols. This family of descriptors need a prior step of primitive extraction. Since graphical symbols are usually composed of a union of several simple sub-shapes, the basic primitives we have extracted to represent a graphical symbol are these simple sub-shapes. In Chapter 4, the polygonally approximated skeletons of the shapes have been taken as the basic primitives representing a symbol. A symbol has been then represented by a set of line segments. Since this representation is quite unstable, in Chapters 5 and 6, a higher level entity has been used as a primitive. Adjacent vectors have been merged together into a polyline instance. We have used the contours of the closed loops forming a symbol as the primitives to polygonally approximate and to merge as single polylines. In Chapter 5, we have proven that this primitive representation is more robust and representative than the use of the line segments arising from a vectorization of the skeleton.
- **Geometric Description of Symbols** In order to describe these extracted primitives, three different methods have been presented in the second part of the book. In Chapter 4, we have proposed a signature model which was formulated in terms of geometric and structural constraints among vectorial primitives, such as parallelisms, straight angles, etc. After representing vectorized line drawings with attributed graphs, our approach encodes the features that are expressive enough to create the signature. The proposed description technique is simple, yet effective to discriminate graphical symbols. In Chapter 5, chains of adjacent segments have been described by an attributed string formalism. In this chapter, we have used a symbolic description instead of a numeric one. Distances between two primitives have been computed by following a string matching algorithm with a particular cost functions. Finally, in Chapter 6, we have proposed coarsely describing vectorial primitives by a reformulation of several off-the-shelf shape descriptors in order to apply them in the vectorial domain.
- **Efficient Access to Huge Amounts of Descriptors** Once primitives are extracted and described, we should organize all the data extracted from the document collection in order to provide an efficient access to it. In Chapter 4, we have worked with a window-based algorithm, so the access to the descriptors has been realized in a sequential way. In Chapter 5, we have proposed the use of a lookup table allowing a prototype-based search. By clustering primitives by similarity,

this indexing data structure has aimed at efficiently retrieving the locations from the document collection where similar primitives as the queried ones were to be found. In order to reduce even more the complexity of accessing the data, we have proposed the use of indexing structures based on the idea of multidimensional hashing in Chapter 6. In particular, we have used a grid file structure to organize the descriptor space.

- **Hypotheses Validation** The last of the three levels in the spotting architectures is the validation of hypotheses. Since from the other levels we have obtained spatial locations where similar primitives can be found, those locations should be validated. Throughout this work, we have based our validation steps on the idea of building a Hough-like voting scheme to validate the locations where several hypotheses were present. The main contribution within this part has been mainly introduced in Chapter 6 where spatial relationships among retrieved primitives have been also introduced as a validation criterion, allowing a more robust identification of the zones of interest.
- **Photometric Descriptors for Symbol Spotting** Although we have centered our research on a focused retrieval application dealing with line-drawing images, and in this context, a geometric and structural approach seemed the most convenient, we have also worked on another application in Part II of this book. An application of document categorization via logo spotting has been presented, and well-known photometric descriptors and matching techniques from the computer vision field have been tested. Such description techniques have been rarely used in the Graphics Recognition field, due to the bi-level nature of document images, and yet we have shown their good performance even though the application has to face binary and noised document images.
- **Performance Evaluation Protocol** Finally, in Part IV, we have presented a set of measures to evaluate the performance of symbol spotting systems in terms of recognition abilities, location accuracy and scalability. We have shown that the proposed measures allowed determining the weaknesses and strengths of different methods. In particular, we have evaluated in detail the spotting method presented in Chapter 6. Although within the Graphics Recognition community there is significant interest in the research of the performance evaluation topic, to the best of our knowledge, no framework for evaluating the performance of spotting applications has been proposed in the past.

8.2 Discussion

In this book, we have made some contributions to the symbol spotting methods and to the particular application of such methods to a focused retrieval system for a collection of line-drawings. In the second part of the book, we presented three different symbol spotting methods which base the description of primitives in a set of geometric and structural constraints. The three methods should be seen as an evolution from the most limited method to a more general and applicable in real situations.

Although it reaches good recognition results, our first method, presented in Chapter 4, has several important limitations when being applied for spotting symbols in large collections. In order for the vectorial signatures to reach good recognition results, we have to make the assumption that the number of segments comprising a symbol will remain stable. When facing real data arising from a raster-to-vector conversion step, this assumption is too strong, and the performance of the spotting system drops. In addition, although the presented method is able to spot symbols, it cannot be used as a focused retrieval application. The main cause is the use of window-based systems. Windowing methods provoke a sequential access to the data and are obviously not well suited for large collections.

The use of a prototype-based search as the one presented in Chapter 5 is clearly a better choice than a sequential access to the data. In a retrieval by similarity framework, such indexing structures allow drastically reducing the amount of distance computations. However, the computation of the representatives from a given cluster is not always straightforward. The decision of whether two primitives should be considered as similar can be somehow subjective. In our experiments, we used the MPEG-7 silhouette database in order to experimentally set up this kind of decision thresholds. However, we believe that the performance of such systems might be enhanced by the use of more complex classification and clustering algorithms.

Our feeling is that one of the right directions to follow in spotting-related problems for the next years is the use of coarser descriptors rather than accurate descriptions techniques. We have shown that the combination of coarse description and relational validation, i.e., combining numeric and structural description techniques, yields very good results. In particular, we have proven in Chapter 6 that there is no need for high-dimensional descriptors for spotting purposes, and with really simple shape descriptors we can reach acceptable performances when combining those descriptions with relational information. Obviously, depending on the intended final application, the word "acceptable" may adopt several meanings. As we have seen in Part IV, if the user of the final application is interested in retrieving the most of the relevant portions of images from the collection, no matter the number of false alarms, a simpler description should be used. If the user is more interested in a better precision without caring for the fact that the system misses symbols, then we should start using more and more complex and fine description techniques. However, we strongly believe that for most of the applications, the use of low-dimensional descriptors (in our experiments of Chapter 6, we use a maximum of seven-dimensional feature vectors) is enough. The choice of such low-dimensional feature vectors avoids the so-called curse of dimensionality and provides an efficient access to the data. The good results obtained by such simple description techniques are also favored by the inclusion of relational and structural information of the graphical symbols. The use of a joint local numerical description and structural analysis contributes to obtaining an important discriminative power. However, structural information should be added carefully since the analysis of complex structural relationships (entailing comparisons in the graph domain) cannot be managed in the context of symbol spotting due to its huge complexity.

One of the critical assumptions that we made throughout this book is that the graphical symbols can be well represented by particular primitives, the region contours. Obviously, not in all the cases the symbols are formed by closed loops and such proposed primitives can be used. However, the presented architecture and framework remain valid no matter which kind of primitive representation we choose and no matter which appropriate description we select.

Finally, we would like to mention the importance that the use of a performance evaluation protocol has. Times where algorithms were tested with a small set of data are over. Nowadays, it is necessary to use the standard reference ground-truth and performance evaluation protocols. The Graphics Recognition community is one of the most healthy communities within the Pattern Recognition field regarding this aspect. A lot of works and efforts are centered on proposing evaluation methods to track the progress in a certain specific problem. As far as we know, the works focused on symbol spotting have always been evaluated by an ad-hoc set of measures. We hope that the proposal of the performance evaluation protocol presented in Part IV can be used to evaluate other spotting methods and can help track the progress on this topic as well as identify the strengths and weaknesses of the proposed methods. However, one of the main problems is that we do not have any public dataset of real documents to test the proposed methods. Nowadays, the only available ground-truthed dataset which can be used to test spotting and focused retrieval of graphics is the one proposed by Delalandre et al. in [1]. The main problem of this dataset is that it is composed of only synthetically generated documents which do not seem realistic, yet; however, it is the only one available, and the community related to spotting applications should start using it.

8.3 Open Challenges

Since symbol spotting is quite a novel problem, we are convinced that there is still a lot of room for improvements and some open challenges.

First of all, the scalability of the proposed methods should be better checked by analyzing other technical documents such as electronic diagrams or mechanical schemes. In addition, even if we have used quite large databases, the methods should be tested on massive data collections in order to assess their transference to real systems.

We would also like to further investigate the use of a different architecture than the one proposed in the introduction. We would like to use some of the spatial access methods presented in [2] for spotting purposes. These data structures should be able to describe and organize symbols in a single step and might be very useful for focused retrieval applications. They have been used in related problems such as GIS system querying for long time.

Another possible research line that we would like to investigate further is the use of the proposed techniques to the retrieval of other elements besides graphical symbols. To the best of our knowledge, most of the word spotting techniques existing in the literature work with a prior segmentation of words and with a learning stage where features from the words are extracted and trained. In the existing word spotting methods, indexing strategies are rarely used and the access to the data is often sequential. We would like to apply the indexing mechanisms and the on-the-fly retrieval framework to the topic of retrieving typewritten or even handwritten words in documents.

More generally, the focused retrieval problem dealing with non-textual queries is and will be one of the major topics of interest for the computer vision and data mining communities. The possibility of formulating queries in an abstract level (semantic querying), i.e., graphics appearing in images, sounds being pronounced in speeches, etc. will be one of the major breakthroughs of the next years. Nowadays we are starting to see the first applications that are able to deal with massive data collections. As an example, Google recently released a *beta* version of its *Google Similar Images*[1] search. It allows the user to search for similar images using pictures rather than words. The similarity between images takes into account spatial information, color, shapes, texture, etc., and the results are quite impressive. Although from its behavior it is pretty clear that the software is getting clues from words associated with images, the results are very promising. The similarity computation and the indexation is all done off-line, and the interface still does not perform real-time image analysis. The images the user would like to search for cannot be uploaded or sketched, but need to be previously indexed.

Finally, another interesting topic which should be further studied is the use of sketch queries instead of a query-by-example paradigm for spotting and focused retrieval applications. In this context, users could roughly sketch the symbol to search for in the collection of graphical documents, thus enhancing the usability of the system. Obviously, if the queries are sketched by the users, a distortion model has to be introduced in order to be able to define whether or not a symbol in a document is similar to the sketch, and to tolerate the inherent distortions of the sketches. The use of the Gestalt laws of perceptual organization might be helpful in such applications.

References

1. Delalandre, M., Pridmore, T., Valveny, E., Locteau, H., Trupin, E.: Building synthetic graphical documents for performance evaluation. In: Graphics Recognition. Recent Advances and New Opportunities, *Lecture Notes on Computer Science*, vol. 5046, pp. 288–298. Springer, Berlin (2008)
2. Samet, H.: The Design and Analysis of Spatial Data Structures. Addison-Wesley, Reading (1990)

[1]See http://similar-images.googlelabs.com/.

Appendix A
Databases

Abstract Throughout this work, several databases have been used, namely, the *GREC database* of graphical symbols, the *MPEG database* of silhouette shapes, and the *floorplan database* which is a selection of several real floorplans. In this appendix, we will explain in detail all these databases. For each one, we will detail the kind of graphical data that comprises the database, its vectorial representation, and kinds of deformations and distortions which are applied to the data.

A.1 GREC 2005 Database

The GREC database comprises a set of graphical symbols coming from different technical fields. It was originally created for the symbol recognition contests held in the past GREC workshops. The results of these competitions are summarized in the following communications [4–6].

The main goal of the contest is to provide a framework for the evaluation of different methods for symbol recognition in graphic documents. This framework is intended to be general and flexible enough so that it can be used to evaluate a wide range of symbol recognition methods. The contest is based on a pre-defined set of symbols (Tables A.1 to A.3). Using this set of symbols, different tests were generated, consisting of several images of each symbol with increasing levels of degradation and distortion in both bitmap images and vectorial representations.

Based on the complete collection of 150 model symbols, we have used two variations of the GREC database in our experiments. The first variation only involves the symbols formed by straight lines. A distortion model is used to introduce noise at the location of the endpoints by preserving the connectivity and the number of segments which comprise a symbol. The second variation is applied to all 150 models. A degradation model is applied to the bitmap images which are then polygonally approximated. In this variation, the number of polylines comprising a symbol is guaranteed to be constant, but these polylines can be made with different number of segments.

Table A.1 GREC 2005 database (1)

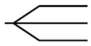
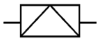
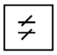
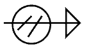
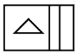
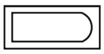
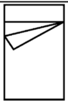
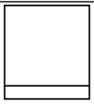
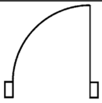
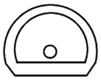
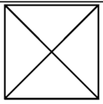
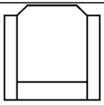
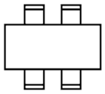
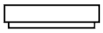
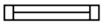
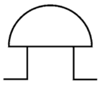
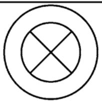
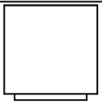
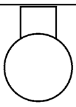
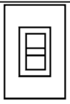
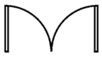
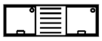
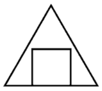
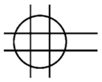
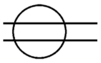
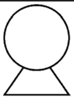
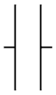
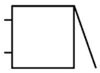
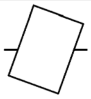
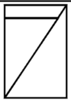
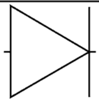
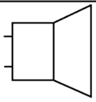
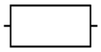
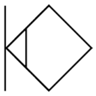
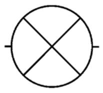
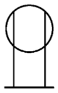
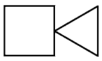
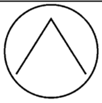
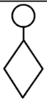
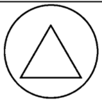
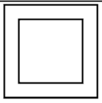
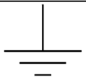
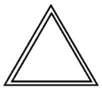
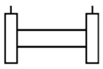
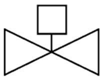
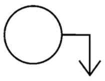
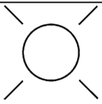
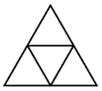
					
symbol001	symbol002	symbol003	symbol004	symbol005	symbol006
					
symbol007	symbol008	symbol009	symbol010	symbol011	symbol012
					
symbol013	symbol014	symbol015	symbol016	symbol017	symbol018
					
symbol019	symbol020	symbol021	symbol022	symbol023	symbol024
					
symbol025	symbol026	symbol027	symbol028	symbol029	symbol030
					
symbol031	symbol032	symbol033	symbol034	symbol035	symbol036
					
symbol037	symbol038	symbol039	symbol040	symbol041	symbol042
					
symbol043	symbol044	symbol045	symbol046	symbol047	symbol048

Table A.2 GREC 2005 database (2)

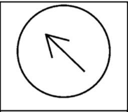
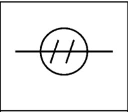
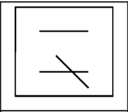
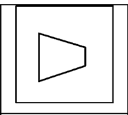
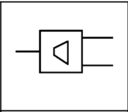
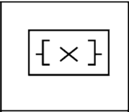
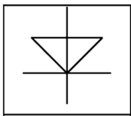
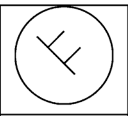
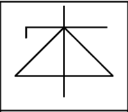
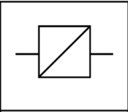
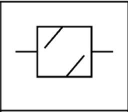
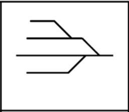
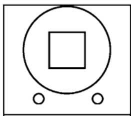
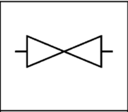
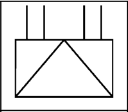
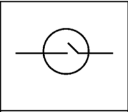
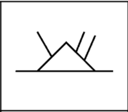
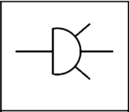
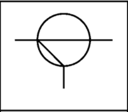
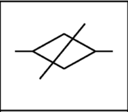
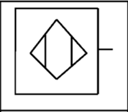
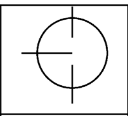
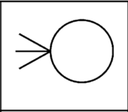
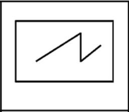
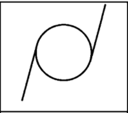
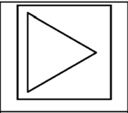
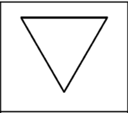
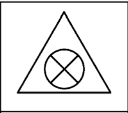
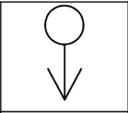
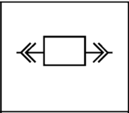
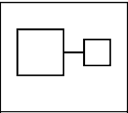
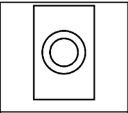
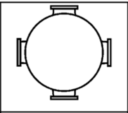
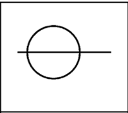
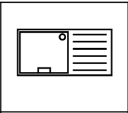
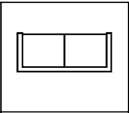
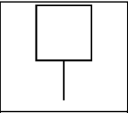
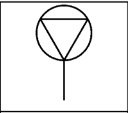
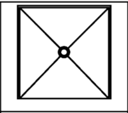
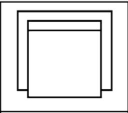
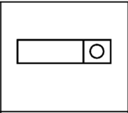
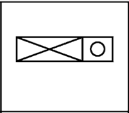
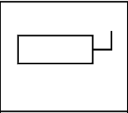
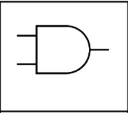
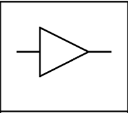
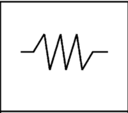
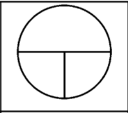
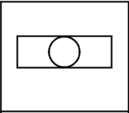
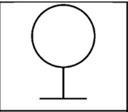
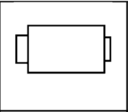
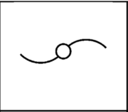
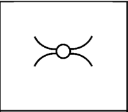
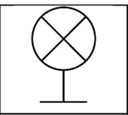
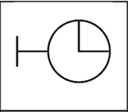
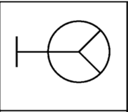
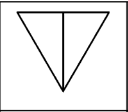
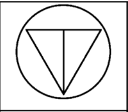
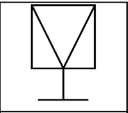
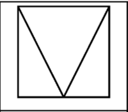
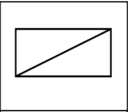
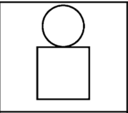
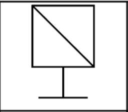
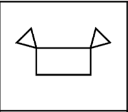
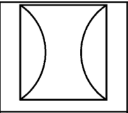
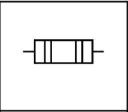
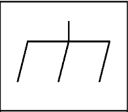
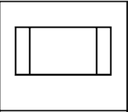
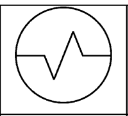
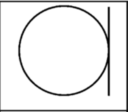
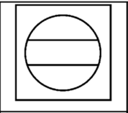
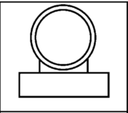
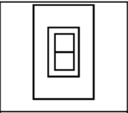
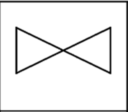
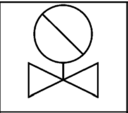
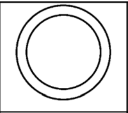
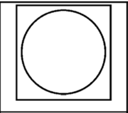
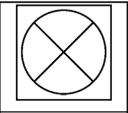
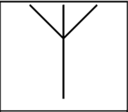
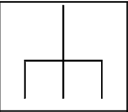
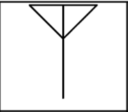
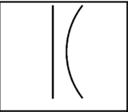
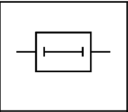
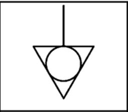
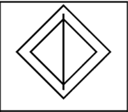
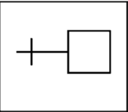
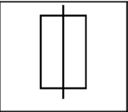
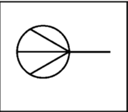
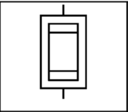
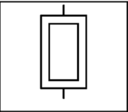
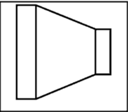
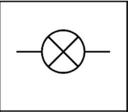
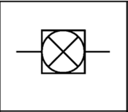
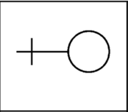
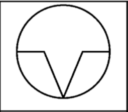
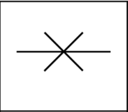
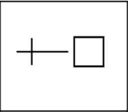
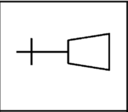
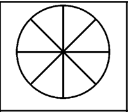
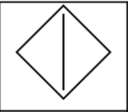
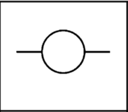
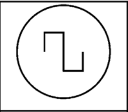
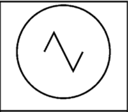
					
symbol049	symbol050	symbol051	symbol052	symbol053	symbol054
					
symbol055	symbol056	symbol057	symbol058	symbol059	symbol060
					
symbol061	symbol062	symbol063	symbol064	symbol065	symbol066
					
symbol067	symbol068	symbol069	symbol070	symbol071	symbol072
					
symbol073	symbol074	symbol075	symbol076	symbol077	symbol078
					
symbol079	symbol080	symbol081	symbol082	symbol083	symbol084
					
symbol085	symbol086	symbol087	symbol088	symbol089	symbol090
					
symbol091	symbol092	symbol093	symbol094	symbol095	symbol096

Table A.3 GREC 2005 database (3)

					
symbol097	symbol098	symbol099	symbol100	symbol101	symbol102
					
symbol103	symbol104	symbol105	symbol106	symbol107	symbol108
					
symbol109	symbol110	symbol111	symbol112	symbol113	symbol114
					
symbol115	symbol116	symbol117	symbol118	symbol119	symbol120
					
symbol121	symbol122	symbol123	symbol124	symbol125	symbol126
					
symbol127	symbol128	symbol129	symbol130	symbol131	symbol132
					
symbol133	symbol134	symbol135	symbol136	symbol137	symbol138
					
symbol139	symbol140	symbol141	symbol142	symbol143	symbol144
					
symbol145	symbol146	symbol147	symbol148	symbol149	symbol150

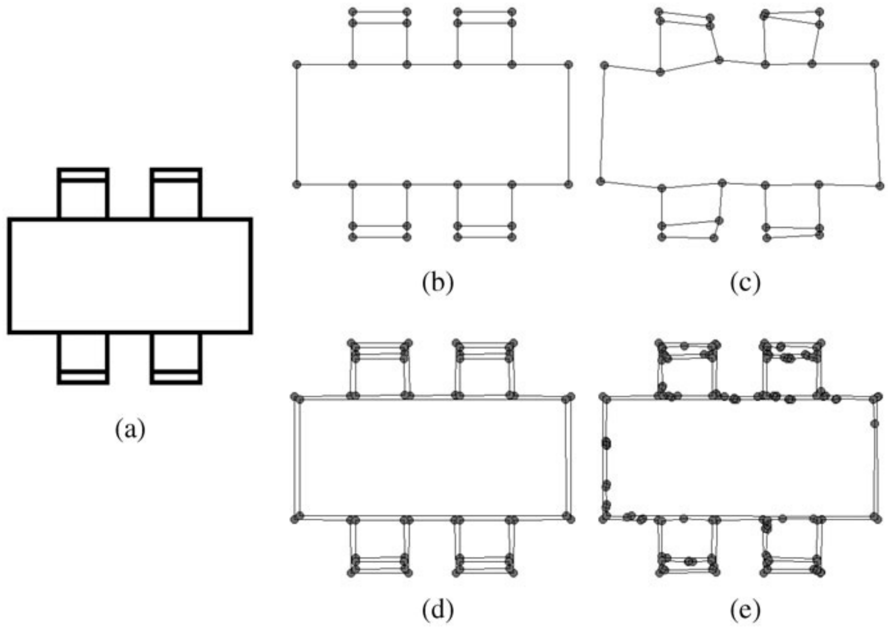


Fig. A.1 Example of variations for the GREC database: (**a**) bitmap model; (**b**) GREC-SEG vectorial model; (**c**) GREC-SEG with endpoint distorition ($r = 15$); (**d**) GREC-POLY vectorial model; (**e**) GREC-POLY with vectorial distortion

A.1.1 Variation GREC-SEG

In this variation,[1] we have used a subset of 80 different symbols of the original GREC database. The used symbols are only those made from straight lines and having no arcs. For each class, 60 distorted instances have been generated by using the following operations.

The vectorial representation of each symbol is represented by an attributed graph where the nodes represent points and the edges are segments between two endpoints. Each node from the graph is randomly shifted within a predefined radius r. Three different levels of distortion are generated with values of r equal to 5, 10 and 15, respectively. At each level, 20 different instances of the symbol are generated. Notice that the graph representations aim to maintain the connectivity and the number of segments which comprise the graphical symbol. In this variation, the symbols are represented by segments stored in VEC format.[2] Figure A.1c shows an example of this vectorial distortion. Some complementary characteristics of this variation can be seen in Table A.4.

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

[2]The vectorial file format used in the symbol recognition contests.

Table A.4 Some characteristics of GREC-SEG dataset

Property	Value
Number of classes	80
Total number of elements	4,800 (60 elements/class)
Max. number of segments in a symbol	36
Min. number of segments in a symbol	3
Mean number of segments in a symbol	10.2

A.1.2 Variation GREC-POLY

In this variation,[3] we have used all the 150 model symbols of the original GREC database. For each class, 300 distorted instances have been generated by using the following operations.

The bitmap images are degraded by using the method presented by Kanungo et al. [1] to simulate the noise introduced by the scanning process. Some simple morphological operations are applied to these degraded images to get rid of the background noise. A connected component analysis is applied to label the closed regions and to extract the internal and external contours comprising a symbol. These distorted contours are then polygonally approximated by using the Rosin and West algorithm introduced in [3]. In this variation, the symbols are composed of several polylines each one made of a set of adjacent segments. The number of polylines which comprise a symbol is constant for a given class, but the number of segments of these polylines is affected by the distortion model and varies from one instance to another. We store these symbols in DXF format. Figure A.1e shows an example of this vectorial distortion. Some complementary characteristics of this variation can be seen in Table A.5.

A.2 MPEG Database

The MPEG database consists of simple pre-segmented shapes defined by their outer closed contours. This database is used in the MPEG-7 Core Experiment CE-Shape-1 (described in [2]) which aims at evaluating the performance of several 2D shape descriptors. We have adapted a subset of this database to build a shape database in vectorial format.

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

Table A.5 Some characteristics of GREC-POLY dataset

Property	Value
Number of classes	150
Total number of elements	45,000 (300 elements/class)
Max. number of polylines in a symbol	16
Min. number of polylines in a symbol	1
Mean number of polylines in a symbol	3.9
Max. number of segments in a symbol	264
Min. number of segments in a symbol	11
Mean number of segments in a symbol	73.7

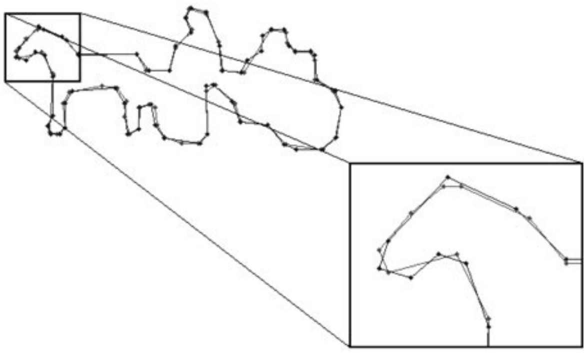


Fig. A.2 Example of the distortions of the MPEG-POLY database

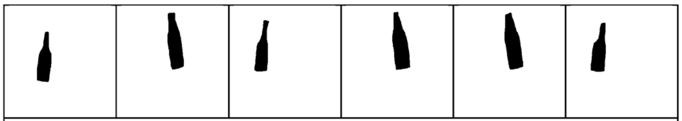
A.2.1 Variation MPEG-POLY

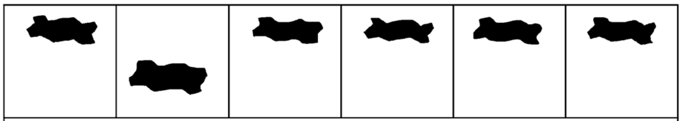
The silhouettes of the MPEG database may be affected by several distortions such as change of view point or non-rigid object motion. As dealing with such strong deformation was not in the scope of our work, we selected a subset of 15 shape classes (20 elements per class) which are only affected by slight changes in shape.[4] We can see some examples of the 15 shape classes in Tables A.6 and A.7.

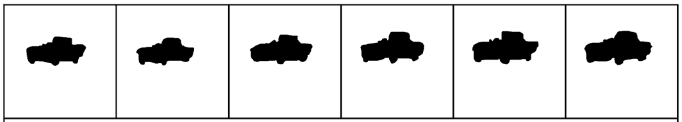
For each contour image, we applied the same distortion model as in the GREC-POLY variation. The noise model proposed by Kanungo et al. is applied to degrade each image. The background noise is cleaned so the distortion only affects the shape contours. These degraded contours are then polygonally approximated. For each image, we generate 300 distorted vectorial shapes. In Fig. A.2, we can see an example

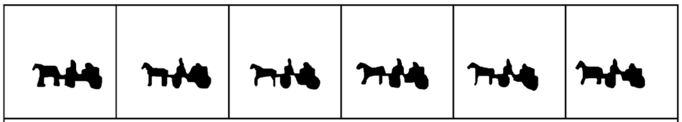
[4]The MPEG-POLY database is available at http://www.cvc.uab.cat/~marcal/MPEG-POLY/.

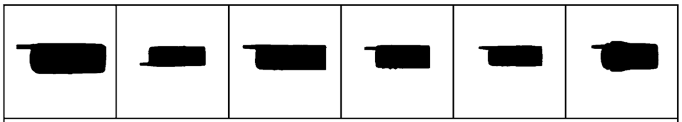
Table A.6 MPEG database (1)

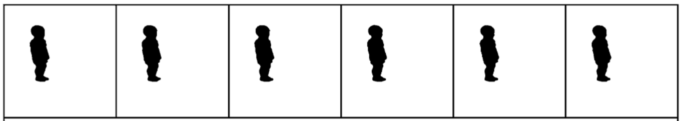

Bottle class.

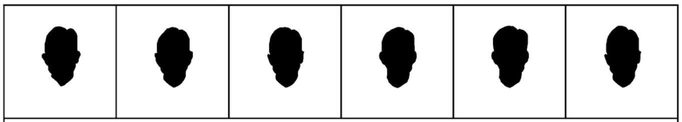

Brick class.

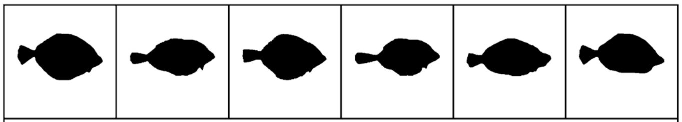

Car class.


Carriage class.


Cellular phone class.

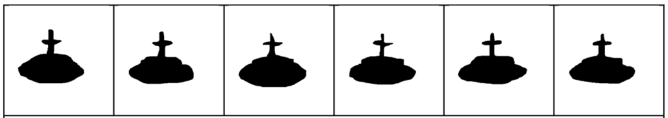
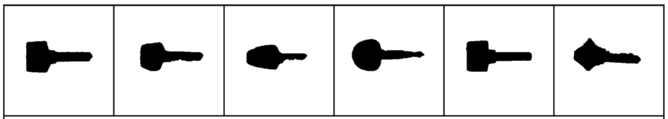
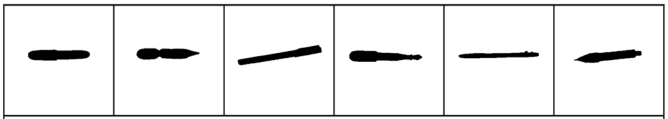
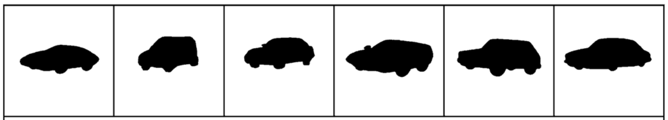
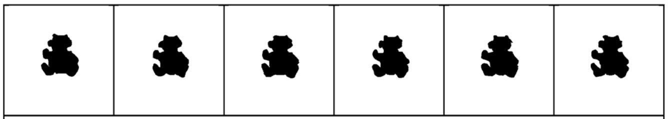
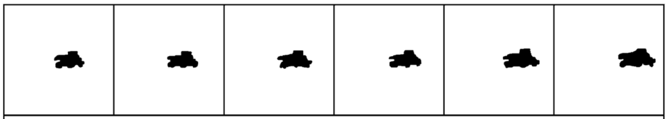
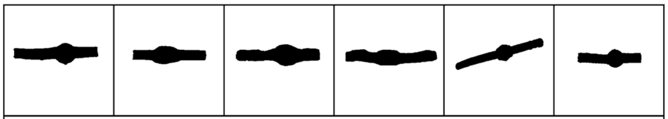

Children class.


Face class.


Flatfish class.

of the resulting polylines after the degradation process is applied to the same instance of the carriage class. Note that as the database is made of closed contours, the resulting vectorial shape comprises only a single closed polyline. These vecto-

Table A.7 MPEG database (2)

	Fountain class.
	Key class.
	Pencil class.
	Personal car class.
	Teddy class.
	Truck class.
	Watch class.

rial shapes are stored in DXF format. Some complementary characteristics of this variation can be seen in Table A.8.

A.3 FPLAN-POLY Database

The FPLAN-POLY[5] database consist of a collection of 42 real floorplans in both DWG and PDF format. The floorplans come from four different projects designed

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

Table A.8 Some characteristics of MPEG-POLY dataset

Property	Value
Number of classes	15
Number of instances	300 (20 instances/class)
Total number of elements	90,000 (300 elements/instance)
Max. number of segments in a shape	91
Min. number of segments in a shape	7
Mean number of segments in a shape	34.1

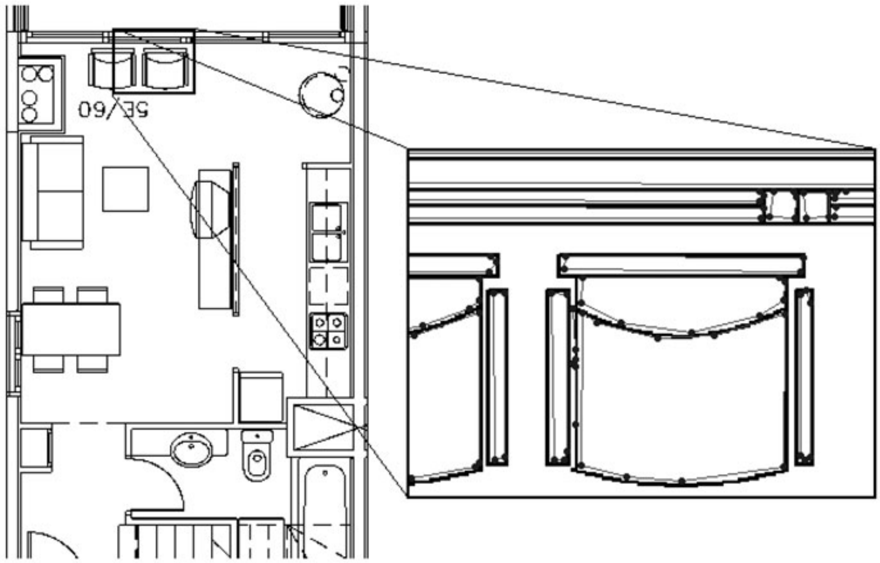


Fig. A.3 Example of the distortions of the FPLAN-POLY database

by the same architect. Nevertheless, the same symbol design can only be found in floorplans from the same project. These originals have been ground-truthed and 388 symbols instances from 38 different symbol classes have been labeled.

To simulate the scanning acquisition process, we applied to each plan the same strategy from the GREC-VECT and MPEG-VECT variations. The floorplan images are degraded via the Kanungo noise, and after a cleaning process they are vectorized to obtain the DXF files. Each floorplan image has been degraded 50 times to have a large database. We can see the results of this distortion process in Fig. A.3. We find the complementary characteristics of this database in Table A.9.

Table A.9 Some characteristics of FPLAN-POLY dataset

Property	Value
Number of real floorplans	42
Number of distorted floorplans	2,100 (50 instances/floorplan)
Number of symbol classes	38
Mean number of symbols per floorplan	8.2
Max. number of polylines in a floorplan	3,710
Min. number of polylines in a floorplan	35
Mean number of polylines in a floorplan	972.3
Mean number of segments conforming a polyline	3.2

References

1. Kanungo, T., Haralick, R., Philips, I.: Global and local document degradation models. In: Proceedings of the Second International Conference on Document Analysis and Recognition, pp. 730–734. IEEE Computer Society, Los Alamitos (1993)
2. Latecki, L., Lakämper, R., Eckhardt, T.: Shape descriptors for non-rigid shapes with a single closed contour. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 424–429. IEEE Computer Society, Los Alamitos (2000)
3. Rosin, P., West, G.: Segmentation of edges into lines and arcs. Image and Vision Computing **7**(2), 109–114 (1989)
4. 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)
5. Valveny, E., Dosch, P.: Report on the second symbol recognition contest. In: Graphics Recognition. Ten Years Review and Future Perspectives, *Lecture Notes on Computer Science*, vol. 3926, pp. 381–397. Springer, Berlin (2006)
6. Valveny, E., Dosch, P., Fornés, A., Escalera, S.: Report on the third contest on symbol recognition. In: Graphics Recognition. Recent Advances and New Opportunities, *Lecture Notes on Computer Science*, vol. 5046, pp. 321–328. Springer, Berlin (2008)

Index

DOI 10.1007/978-1-84996-208-7, © Springer-Verlag London Limited 2010