Technical University of Kaiserslautern

Geometric-based Symbol Spotting and Retrieval in Technical Line Drawings

by

Nibal Nayef

Thesis approved by the:
Department of Computer Science
Technical University of Kaiserslautern
for the award of the doctoral degree:
Doctor of Natural Sciences (Dr. rer. nat.)

14 December 2012

D 386

Gedruckt mit Unterstttzung des Deutschen Akademischen Austauschdienstes.

Abstract

The automatic analysis and retrieval of technical line drawings is hindered by many challenges such as: the large amount of contextual clutter around the symbols within the drawings, degradation, transformations on the symbols in drawings, large databases of drawings and large alphabets of symbols. The core tasks required for the analysis of technical line drawings are: symbol recognition, spotting and retrieval. The current systems for performing these tasks have poor performance due to the mentioned challenges. This dissertation presents a number of methods that address these challenges. These methods achieve both accurate and efficient symbol spotting and retrieval in technical line drawings, and perform significantly better than state-of-the-art methods on the same problems. An overview of the key contributions of this dissertation is given in the following.

First, this dissertation presents a geometric matching-based method for symbol recognition and spotting. The method performs recognition in the presence of large amounts of contextual clutter, and provides precise localization of the recognized symbols. On standard databases such as GREC-2005 and GREC-2011, the method achieves up to 10% higher recall and up to 28% higher precision than state-of-the-art methods on the spotting task, and achieves up to 7% higher recognition accuracy on the isolated recognition task. The method is based on a geometric matching approach, which is flexible enough to incorporate improvements on the matching strategy, feature types and information on the features. The method also includes an adaptive preprocessing algorithm that deals with a wide variety of noise types.

In order to improve the performance of the spotting method when dealing with degraded drawings, two novel methods are presented in this dissertation. Both methods are based on combining geometric matching with machine learning techniques. The geometric matching is used to automatically generate training data that contain information on how well the features of the queries are matched in both the true and the false matches found by the spotting method. The first method learns the feature weights of the different query symbols by linear discriminant analysis (LDA). The weighted query features are used in the spotting method and result in 27% higher average precision than the original method, with a speedup factor of 2. The second method uses SVM classification as a post-spotting step to distinguish the true from the false matches in the spotting method. The use of the classification step further improves the average precision of the spotting method by 20.6%.

This dissertation also presents methods for content analysis of line drawings. First, a method for accurate and consistent detection (95.8%) of regions of interest (ROIs) is presented. The method is based on statistical feature grouping. The ROI-finding method is identified as an important part of a symbol retrieval system: the better the detected ROIs,

the higher the performance of a retrieval system. The ROI-finding method is also used to improve the performance of the geometric-based spotting system.
Second, a symbol clustering method for building a compact and accurate representation of a large database of technical drawings is presented. This method uses the output from the ROI-finding method as input, and uses geometric matching as a similarity measure. The method achieves high accuracy (90.1% recall, 94.3% precision) in forming clusters of symbols. The representatives of the clusters (34 symbols) are used as key entries to a symbol index, which is identified as the outcome of an off-line stage of a symbol retrieval system.

Finally, an efficient and high performing large scale symbol retrieval system is presented in this dissertation. The system follows the bag of visual words (BoVW) model, but with using methods that are suitable to line drawings. The system uses the symbol index to represent a database of drawings. During the on-line query retrieval stage, the query is analyzed by the ROI-finding method, matched with the key entries of the symbol index via geometric matching, and finally, a spatial verification step is performed on the retrieved matches. The system achieves a query lookup time that is independent of the size of the database, and is instead dependent on the size of the symbol index. The system achieves up to 10% higher recall and up to 28% higher precision than state-of-the-art spotting systems on similar databases.

Overall, these contributions are major advancements in the research of graphics recognition. The hope is that, such contributions provide the basis for the development of reliable and accurate performing applications for browsing, querying or classification of line drawings for the benefit of end users.

To My Father
To My Mother

Acknowledgments

I would like to express my deepest gratitude to the people who have helped and supported me over the last four years. First and foremost, I would like to thank my thesis advisor, Thomas Breuel, for his support and encouragement. My discussions with him have been interesting, useful and inspiring. I also want to thank my other two committee members, prof. Josep Lladós and prof. Markus Nebel, for taking the time to read my thesis, and for the useful feedback.

I would like to thank my good friends in IUPR for being helpful and nice, specially during the times when I was stressed about my thesis progress. I particularly want to mention: Ludwig Schmidt-hackenberg, Wonmin Byeon, Zeshan Afzal, Wanlei Zhao, Martin Kraemer, Federico Raue, Reza Yousefi, Ilya Mezhirov and Adnan Ul Hasan. Thanks for contributing to my work through many comments on the thesis, and thanks for all the good times we spent together in the different group events. I want also to thank Ingrid Romani for her help with the administrative stuff. Raghad Tarawneh, Lama Osman and Asmaa Radwan have been great friends. Special thanks to all of you for helping me organize a great graduation party after my defense.

DAAD is thanked for providing me with the opportunity to pursue graduate studies. Without DAAD-scholarship, doing a Ph.D. would have been extremely difficult.

Most of all, I would like to thank my brother, Nizar Nayef, for his tremendous help and support with everything since the day I have arrived in Germany.

Contents

List of Figures

List of Tables

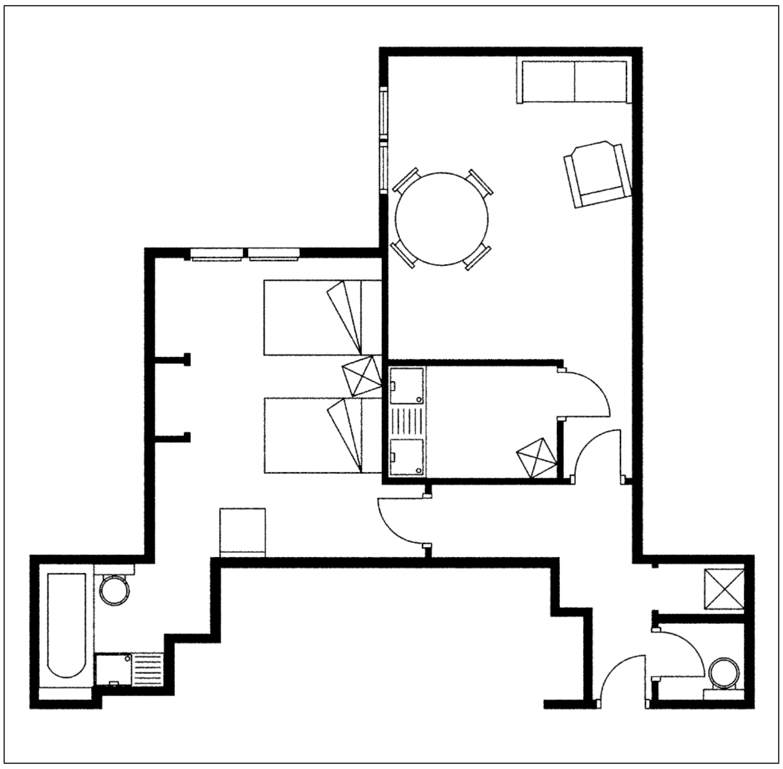
Chapter 1

Introduction

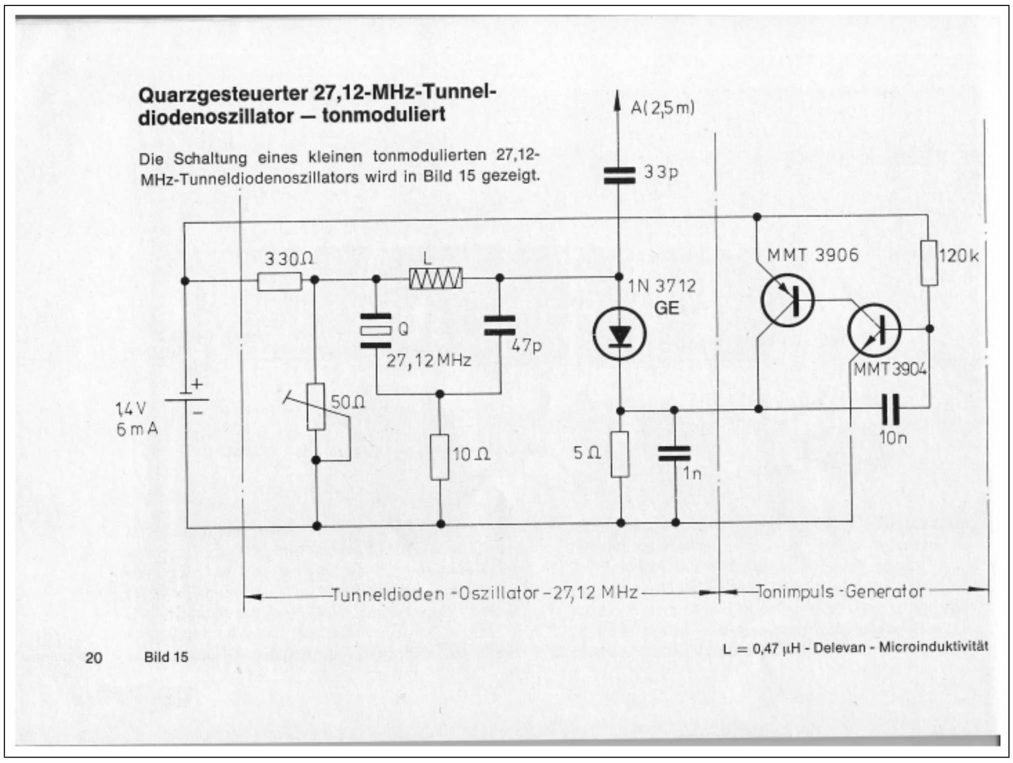
There is a wealth of information contained in graphically-rich documents, in particular, large collections of technical line drawings. In order to exploit this information, techniques for analysis and retrieval of such drawings are essential. Unlike in text documents, the automatic analysis and retrieval of graphically-rich documents remain largely unsolved problems [1, 2]. Graphically-rich documents constitute a large and important class of document images. Such images include line drawings of all types such as engineering drawings, maps, comical drawings, logos etc., the line drawings can be machine-printed or hand drawn. From among the different types of graphically-rich documents, this dissertation is concerned with dealing with printed technical line drawings. Examples of technical drawings are architectural floor plans, electric and electronic circuit diagram and mechanical drawings. Figure 1.1 shows examples of such drawings.

Technical line drawings are composed of symbols, connection lines and possibly text. A symbol is any graphical entity that is usually free-shape and has a specific meaning in a particular application domain. The symbols in technical line drawings come from a set of defined symbols called the ***symbol alphabet***. Such an alphabet of symbols is defined for an application domain and is used to draw the technical line drawings from that domain. For example, architectural symbols such as the symbols of “door”, “window”, “stairs” are used to draw an architectural floor plan, and electronic symbols such as “transistor”, “diode”, “resistor” are used to draw electronic circuit diagrams. Figure 1.2 shows a group of such technical symbols.

The analysis of technical line drawings is useful for many applications. The automatic interpretation of line drawings is among the most important applications. Examples of the automatic interpretation of drawings include converting an image of an architectural floor plan into 2D or 3D CAD formats [3, 4], or an electronic circuit diagram into a CAD format for simulation purposes. The interpretation of line drawings requires identifying the



(a) An architectural floor plan.



Quarzgesteuerter 27,12-MHz-Tunnel-diodenoszillator — tonmoduliert

Die Schaltung eines kleinen tonmodulierten 27,12-MHz-Tunneldiodenoszillators wird in Bild 15 gezeigt.

(b) An electronic circuit diagram.

Figure 1.1: Images of technical line drawings.

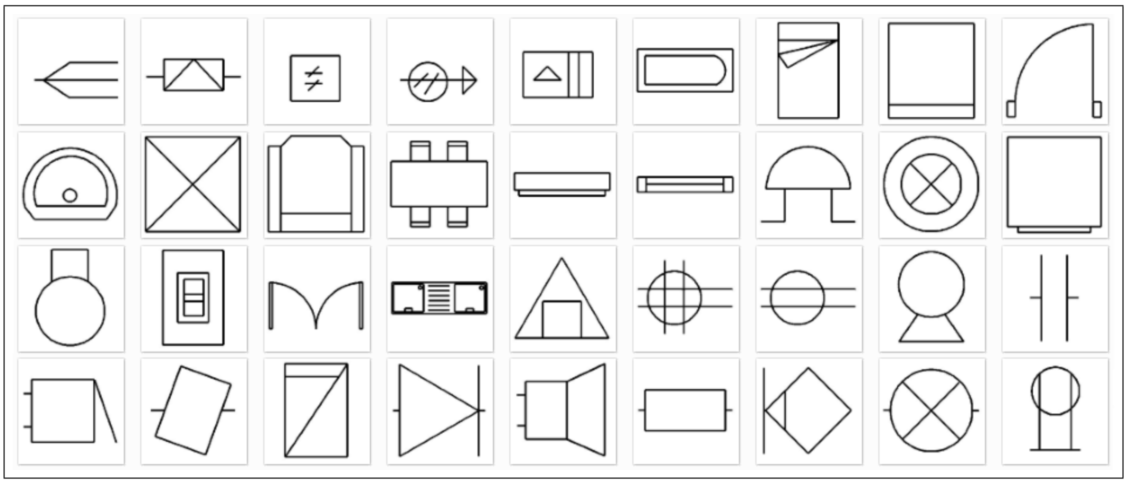


FIGURE 1.2: A subset of a library of technical symbols.

graphical entities (the symbols) that appear in the drawings. Document classification is another useful application, where documents can be classified according to their graphical content of symbols or logos [5]. Recognizing symbols can be used in optical character recognition (OCR) as in [6], and in mathematical symbol recognition as in [7, 8]. Symbol spotting methods can also be used in word spotting as proposed by Rusinol and Llados in [9]. The core task in all these different applications is symbol recognition, whether it refers to recognizing an isolated symbol, or locating a symbol within a line drawing. Another application is browsing and searching line drawings [10, 1], for example, performing content-based search in digital libraries. In this application, the core task is large scale symbol retrieval.

In summary, there are three core tasks required for automatic analysis of technical line drawings. These tasks are: symbol recognition, spotting and retrieval. The next subsection presents the definitions of each of these tasks, and the challenges faced in trying to accomplish them.

1.1 Analysis of Technical Line Drawings: Definitions and Challenges

The processing and analysis of technical line drawings is a branch of graphics recognition research. The community of graphics recognition research uses a certain terminology to refer to the different tasks in analyzing line drawings. This terminology is explained in the following. *Symbol recognition* refers to the problem of recognizing isolated symbols. Given an image of an unknown symbol, and a library of symbols (can be the symbol alphabet), the task is to recognize this symbol as one of the symbols in the symbol library [11, 12, 13].

Symbol spotting refers to the problem of locating instances of a symbol within a line drawing. Given a query symbol and a line drawing, the task is to locate (spot) all the regions within the line drawing that contain instances of the query symbol [1, 13, 14]. *Symbol retrieval* can be viewed as large scale symbol spotting. Given a query symbol and a database of line drawings, the task is to locate the instances of the query symbol in all the drawings. In this retrieval task, the regions of interest are returned to the user rather than the line drawings themselves. This is referred to as focused retrieval [10].

In technical line drawings, the query symbols used in the recognition/spotting tasks usually come from the symbol alphabet. Alternatively, the query symbols can be selected (or cropped) by the user from a line drawing. So, the query symbol may be surrounded by a little clutter (few background lines or lines from other symbols). This introduces another task related to the above mentioned tasks which is *recognition with contextual noise on the query.* The symbols that have contextual noise can also come from methods for finding regions of interest in drawings such as sliding windows. A lot of spotting methods use a step for finding regions of interest before performing the actual matching/spotting steps.

In the graphics recognition community, many research works have been devoted to solving the problems of symbol recognition, spotting and retrieval. The approaches and methods developed to deal with these problems face many challenges. A common challenge for these methods is the textureless nature of line drawings, which means that only shape information is available. Performing recognition in line drawings requires developing methods different from the ones used for textured gray scale images.

For isolated symbol recognition, the main challenge is to recognize symbols under transformation and degradation. The images of symbols may be corrupted with different noise types, some of which result in severe degradation of the symbols. The noise affects the preprocessing and vectorization steps of a recognition method negatively, hence causing a decrease in the recognition accuracy. Another important challenge is the scalability, which refers to performing recognition with a large library of symbols. Having a large number of symbols in the library imposes several difficulties for isolated symbol recognition methods. With an increasing number of symbols, there are more symbols whose shapes are very similar to the shapes of other symbols, which causes false recognition. Furthermore, a large symbol library slows down the recognition process, since the unknown symbol has to be matched to each symbol in the symbol library. Several contests [15, 16, 17] have been organized for isolated symbol recognition. In those contests, the participating methods are tested for their ability to overcome the challenges of transformation, degradation and scalability.

For the problem of symbol spotting, the main challenge is to recognize the symbols within the context of a complete line drawing, i.e. recognition in the presence of large amounts of contextual clutter. The symbols within a line drawing may be connected to other symbols or to connection lines, and they are drawn in different scales and orientations with respect to the query symbol. Moreover, in some application domains, there are some symbols whose shapes are subsets of shapes of other symbols. In addition to these challenges, spotting problems also share the same challenges of isolated recognition problems. The line drawings can be degraded with noise, or can be scans of old documents. The scalability challenge is even more problematic in spotting problems, besides the difficulty of having a large library of symbols, the line patterns in local regions of a line drawing may look similar even if they belong to different symbols.

The symbol retrieval problem naturally has the same challenges as the spotting problem, with the additional big challenge of performing spotting in large databases. The symbol instances that are similar to a query, have to be retrieved (located) in all the drawings in a database within a reasonable response time. The retrieval process should be efficient, as the response time is a very important factor for the usability of a retrieval system.

Due to all these challenges, the methods presented so far in the literature are not yet reliable enough to be used in the previously mentioned applications. Towards achieving the ultimate goal of automatic understanding and interpretation of technical line drawings, this dissertation presents a group of methods to solve the problems of symbol recognition, spotting and retrieval. The recognition of symbols – both isolated recognition and in-context spotting – is achieved by using geometric matching techniques. The efficient retrieval of symbols is achieved by performing content analysis on line drawing database, and then building a compact and indexable representation of the database, which enables fast retrieval of the regions within the drawings.

1.2 State-of-the-art in Symbol Recognition, Spotting and Retrieval

The methods presented in the literature address various graphics recognition problems. For example, recognition in hand-drawn or printed line drawings, isolated and/or non-isolated symbol recognition, indexing and retrieval of line drawings or on-line symbol recognition based on user's interactive drawing.

The survey of Cordella et al. [18], provided an overview about symbol and shape recognition, and reviewed the methods used in different phases of the recognition process. Reviews of symbol recognition methods were presented by Chhabra in [19] and by Llados et al. in [11]. Tombre presented state-of-the-art reviews about analysis of technical drawings in [20] and in [2]. Tombre also discussed the past and future issues in graphics recognition in [12]. Wenyin [21] presented a review about on-line graphics recognition techniques, on-line recognition is based on the interactive incremental input of strokes provided by a user. A more recent review of symbol spotting approaches was presented by Rusinol and Llados in [1].

The following two subsections provide a brief review of methods that perform recognition in printed technical line drawings, in particular, symbol recognition, spotting and retrieval methods. The next section reviews the evaluation protocols used in assessing the performance of these methods.

1.2.1 Isolated Symbol Recognition

Symbol recognition methods rely on using a descriptor for representing the visual cues of symbols as a main step. Matching the descriptors of the symbols can then be done using distance measures, learning methods or otherwise. Based on the type of the descriptor they use, isolated symbol recognition approaches can be classified into structural [22, 23, 24], statistical [25, 26, 27] and hybrid [28, 29] approaches.

Structural approaches extract geometric primitives such as line segments from the images, and arrange the extracted features into a data structure such as a graph, a network or a tree. The resulting arrangement is used as a representation of a symbol and compared to other symbols' representations via different techniques. Zhang et al. [24] created symbol signatures from pairwise relationships of the vectorial primitives of the symbols. The values of the signature are ordered in order to perform the signature comparison efficiently. In their approach, the symbols are assumed to be already in clean vectorized format. Graph-based descriptors have attracted many researchers, Coustaty et al. [22] used an adapted Hough transform to extract the segments of a symbol, and arrange them in a topological graph as a structural signature, and then Galois Lattice classifier is used for recognition. The work in [23] used vectorized symbols and converted them to attributed relational graphs. Feature vectors are computed from the graph representations. A Bayesian network classifier is trained on the feature vectors, and is then used for recognition.

Most of the methods that follow the structural approach rely on the assumption that the symbols are perfectly vectorized, which is usually not the case. Preprocessing and primitives extraction greatly affect the matching results of these methods. Moreover, these methods do not scale well with large symbol alphabets.

Statistical approaches use pixel level information and/or global features to compute simple or complicated feature descriptors of the symbols. In general, these approaches are designed only for presegmented symbols and are sensitive to noise. Min et al. [25] presented a pixel-based descriptor, and a similarity measure called bipartite transformation distance.

In their system, the symbols are aligned by their angular distributions to achieve rotation invariance. The scale invariance is achieved by normalizing the symbols to a common size, depending on that the symbols are isolated. Obviously, such a system cannot be used for symbol spotting. The bipartite transformation distance between two symbols is related to the cost required to transform the first symbol to the second, and the cost required to transform the second symbol to the first. In their preprocessing module, a denoising algorithm is used to deal with different noise types before applying the recognition system. They discussed how the images corrupted with noise slow down the symbol recognition process and cause higher error rates.

Yang [26] developed a descriptor based on a histogram computed from pixel relations to the other pixels. The histograms are statistically integrated to obtain a feature vector, and the sum of absolute differences is then used to measure similarity between feature vectors. Zhang et al. [27] represented symbols as 2D kernel densities, and computed the similarity by Kullback-Leibler divergence. Before describing the symbols, they used their adaptive noise removal method of [30]. In the work of Zhai and Du [31], the symbol descriptor is computed using a multi-scale autoconvolution transform, and the radial basis probabilistic neural network is used for classification.

Hybrid approaches in symbol recognition use a mixture of statistical and structural descriptors in order to try and get the best of the two descriptions. In the recent work of Su et al. [28], symbol segments are used as structural features, and Radon transform as statistical features. Rusinol et al. [29] converted raster symbols to polygonal approximations, then used closed loops of poly-lines as a higher level vectorial representation of the symbols. The loops are described statistically by boundary moment invariants. Finally, a Hasse diagram is derived from the symbol description and is used to recognize symbols.

Many other symbol recognition methods have been proposed. The status of the research in isolated recognition is somewhat mature, and some methods achieve above 90% recognition accuracy even in the presence of noise and transformations. However, most of these methods are time consuming, do not scale well and are designed only to work on isolated symbols, hence, they do not perform well in the presence of contextual clutter [32].

1.2.2 Symbol Spotting and Retrieval

There are two approaches to symbol spotting. The first is a region-based approach, which is based on finding regions of interest in the drawings, describing them using various descriptors, and then indexing them based on the query symbol. The second spotting approach uses some kind of segmentation to get symbols separated from the background, and then recognize those isolated symbols. Most of the spotting methods in the literature belong to the region-based approach. This approach has the following inherent problems. First it is hard to locate zones of interest in a drawing in a scale and rotation-invariant way. Second, special descriptors have to be developed for line drawings, as the usual texture-based descriptors won't perform well on the similar local patterns of line segments. Third, indexing techniques like hashing do not scale well for large databases. Due to these problems, the methods that implement the first approach result in low recall and precision rates. One main reason for following this approach, rather than the segmentation-based approach, is that the current segmentation techniques do not perform well on complex technical drawings.

In the **region-based spotting approach**, state-of-the-art methods use various ways to detect and describe regions of interest within the drawings. Rusinol et al. proposed different spotting methods in [32] and [33], in all of which, closed regions are used to detect regions of interest. In [32], the regions are described by a two-center bipolar coordinate system, and a hash table is used as an indexing scheme. In [33], the regions are described by attributed strings. Similar strings are clustered in a lookup table so that the set of median strings act as indexing keys. String matching is used to compare a query to the regions. Finally, the initially found matches are verified using a Hough-like voting scheme. Ah-Soon et al. [34] used overlapping sliding windows to detect regions. The regions are described as feature vectors that represent the connection relations among the segments of the regions, the descriptions are represented as a network (like a graph). The search for symbols is done by propagating (checking) the similarity of the constraints of segment relations in the network.

Graph representations are popular in spotting methods like they are in isolated symbol recognition methods. Qureshi et al. [35] used attributed relational graphs (ARG) to represent vectorial primitives of line drawings. Regions are detected by examining the relations between the vectorial primitives. A graph description and matching method (introduced by them in [36]) is used to match the query to the regions. Llados and Sanchez [37] presented a combination of graph matching and graph parsing for the representation of line drawings. They presented a graph indexing mechanism for spotting known symbols in the drawings. Locteau et al. [38] used a graph to represent the geometric relations of the geometric primitives of line drawings. The regions are detected based on closed or open curves. Matching a query to the regions is done via edit distance.

The **segmentation-based spotting approach** has been very rarely used. Tabbone et al. [39] presented a spotting method that follows this approach. They used a text-graphics segmentation method with user corrections to segment symbols in simple line drawings. A discretized version of the so called F-signature (a special kind of histogram of forces) is used as a descriptor. The sum of absolute differences is used for matching the query to the segmented symbols. Such method cannot work for more complicated line drawings, or for other types of drawings.

It is clear, for the spotting methods that follow the region-based approach, that if the regions in the line drawings could be more reliably and precisely located, the later steps of describing and matching them would achieve better results. As for the methods of the second approach, they always require good prior segmentation, which is very hard to achieve. Overall, the number of spotting methods is increasing, and some of them have good recall rates, however, all of them suffer from low precision rates.

Approaches to Symbol Retrieval
In order to perform symbol retrieval (i.e. large scale spotting), one cannot search each line drawing of a database sequentially. Symbol retrieval methods should be based on off-line content analysis of a database like text and image content-based retrieval systems. The approach followed in these systems relies on the off-line processing of the database, the result of which, is a compact and indexable representation of the whole database. This representation is used later for fast query retrieval. The off-line stage usually has the following steps: finding regions of interest, describing and clustering them, and finally building an indexable data structure from the clusters. The retrieval stage compares the query description to the entries of the previously built data structure.

Some symbol spotting methods follow this approach. Such methods can be developed into large scale spotting systems. Examples of these methods can be found in [40], [41], [42] and [10]. In general, these methods have weak content analysis steps, this results in inaccurate representations of line drawings, and negatively affect the recall and precision rates of retrieval. These methods are discussed in detail in Sections 5.3, 6.2 and 7.2.

1.3 Performance Evaluation Protocols

As the literature in the fields of symbol recognition and spotting grows, it becomes necessary to develop performance evaluation frameworks for the proposed methods. Various issues should be considered in performance evaluation. The first issue is the availability of standard databases on which the methods can be evaluated. To that end, researchers have collected or generated line drawing databases. A large database of technical line drawings was generated by Delalandre et al. in [43] and [44]. This database has both architectural floor plans and electrical diagrams, they are synthetic images that imitate real technical drawings. Databases of isolated technical symbols have been generated and used in symbol recognition contests in [45], [15], [16] and [17]. The images of the symbols are transformed and degraded in order to test the robustness of the recognition methods to noise and transformations.
The second issue is the availability of common evaluation criteria and metrics for assessing the performance of different methods. A performance evaluation protocol for symbol spotting methods has been presented by Rusinol and Llados in [46]. Evaluation frameworks for

symbol recognition and symbol spotting have been presented by Delalandre et al. in [13], and by Valveny et al. in [47]. In these evaluations protocols, *recognition accuracy* is used as the main performance measure for isolated symbol recognition tasks. Whereas for spotting and retrieval tasks, *recall* and *precision* measures are adapted for evaluating the spotted or retrieved regions in line drawings.

1.4 Contributions of this Dissertation

Based on the discussion presented at the beginning of this chapter, there are three core tasks necessary for analyzing technical line drawings. These tasks are: symbol recognition, spotting and retrieval. The state-of-the-art review presented in the previous section shows that there is a need to develop more accurate, more efficient and more robust solutions for these tasks. This dissertation presents a group of methods that contribute to providing such needed solutions. This section summarizes the contributions of this dissertation, whereas the detailed contributions – along with the experiments and evaluations necessary to prove them – are discussed in the rest of the chapters of this dissertation. The main contributions of this dissertation are summarized in the following.

1. The development of a geometric matching-based method for symbol recognition and spotting, the method achieves significantly lower error rates on standard databases for both the tasks of isolated and in-context recognition. The method has the following contributions.

    - an adaptive preprocessing and noise removal algorithm for dealing with a wide variety of noise types in line drawings, including the clean drawings.
    - the formulation of the symbol recognition problem as a geometric matching problem, with different types of features, and incorporation of higher level information on the features such as weights and neighboring information.
    - the use of the geometric matching algorithm of [48] to perform symbol recognition-in-context.
    - the development of two more efficient variants of the used geometric matching algorithm.
    - the development of a penalization method for non-matched features to reduce false matches.
    - a performance evaluation with comparison to the state-of-the-art recognition and spotting systems on standard databases: for the spotting task: achieving – at the same time – up to 10% higher recall and up to 28% higher precision, for the isolated recognition task: achieving up to 7% higher recognition accuracy.

2. Two novel methods for improving the performance of the geometric matching-based spotting mentioned in the previous contribution. The novelty lies in combining machine learning techniques with geometric matching with the following contributions.

   - learning the feature weights of library symbols, where the weights designate the importance of the features for the spotting task.
   - the automatic generation of training data that contain information on how well the individual features of the query symbols are matched in both the true and the false matches found by the spotting method.
   - the use of linear discriminant analysis (LDA) to learn the feature weights from the training data.
   - the incorporation of the weights in the geometric matching-based spotting to get an enhanced spotting method.
   - the weights-enhanced spotting system achieves a 27% higher average precision rate, and performs spotting twice as fast on a large database of line drawings with different queries.
   - the incorporation of an SVM-classification step as a post-spotting step in the geometric matching-based spotting method. The SVM classifier is trained on the previously mentioned training data to learn to distinguish between the true and the false matches found by the spotting method.
   - the classification-enhanced spotting method achieves a 20.6% higher average precision rate on a large database of line drawings with different queries.

3. A novel method for finding accurate regions of interest (ROIs) in technical line drawings, the method is based on feature grouping with the following contributions.

   - the use of non-accidental properties to group sets of features in line drawings.
   - an algorithm for accurate and consistent detection (95.8%) of ROIs in line drawings based on the previously formulated groups of features.
   - identifying the importance of the ROI-finding method for content analysis of line drawings, hence for the use in symbol retrieval systems.
   - incorporation of the detected ROIs in the geometric matching-based spotting method to improve its speed by a factor of 3 with the same or slightly better recall and precision.

4. Building a compact symbol index from a collection of line drawings by clustering, with the following contributions.

   - a novel symbol clustering method that uses geometric matching as a similarity measure of symbols' shapes.

- getting well formed clusters of symbols (90.1% recall, 94.3% precision) by applying the clustering method on the ROIs detected from a large database of line drawings.
- using the clusters to construct an index of symbols and parts of symbols that appear in all the line drawings of the database. The symbol index is a very compact representation of the database, and it is analogous to the visual vocabulary used in content-based image retrieval systems.
- identifying the importance of the constructed symbol index as a component of a retrieval system.

5. The development of an efficient and high performing large scale symbol retrieval system, with the following contributions.
   - building the symbol retrieval system in a way analogous to content-based image retrieval systems that use the successful bag of visual words (BoVW) model, but with using methods that are suitable to textureless line drawings.
   - using the symbol index as the outcome of analyzing the contents of a database of line drawings, and identifying this analysis as the off-line stage of the retrieval system.
   - the use of geometric matching for matching the ROIs of a query symbol to the entries of the symbol index.
   - the development of a simple and fast spatial verification step to verify the correctness of the query matches found by the geometric matching step, and identifying both steps as the on-line stage of the retrieval system.
   - realizing an efficient symbol retrieval system whose running time is independent of the size of the database. The retrieval time depends on the size of the symbol index.
   - achieving up to 10% higher recall and up to 28% higher precision than state-of-the-art spotting systems on similar databases.

1.5 Organization of this Dissertation

This chapter discussed the importance of the automatic analysis of technical line drawings, and identified the main problems faced in performing such analysis as: symbol recognition, spotting and retrieval. This was followed by a brief state-of-the-art review of these problems. The chapter also summarized the contributions of this dissertation towards solving these problems. The rest of this dissertation is organized as two main parts: (I) symbol recognition and spotting, (II) symbol retrieval, and a conclusion chapter.

The first part consists of three chapters that discuss the contributions related to isolated symbol recognition and symbol spotting as follows. Chapters 2 and 3 present a geometric matching-based method for recognition and spotting, where Chapter 2 presents the theoretical and algorithmic aspects of the method, and Chapter 3 presents the practical aspects with the performance evaluation. Chapter 4 presents two novel methods based on combining machine learning techniques with geometric matching. The chapter shows the improvements of the geometric matching-based spotting when using the two methods.

The second part consists of three chapters that discuss the off-line and on-line stages of a symbol retrieval system. Chapter 5 presents a novel ROI-finding method based on feature grouping. The importance of the ROI-finding step in spotting and retrieval systems is shown. Chapter 6 presents building a symbol index from a database of line drawings, where the main step used in building the index is a geometric matching-based clustering method. The two chapters represent the two main steps in the off-line stage of a retrieval system. Chapter 7 presents a complete symbol retrieval system, where the symbol index is used as the outcome of the off-line stage. The chapter then presents the on-line stage of the system: query matching and spatial verification. The evaluation of each of the steps of the retrieval system is presented in their respective chapters.

Finally, Chapter 8 presents some concluding remarks about the work presented in this dissertation. The chapter also presents possible future directions of the work.

Part I

Symbol Recognition and Spotting

Chapter 2

Theoretical Considerations in Applying Geometric Matching to Symbol Recognition

The chapter presents a review of branch-and-bound geometric matching approaches, adapted to the problem of symbol recognition in line drawings, and proposes extensions necessary for high performance symbol recognition. Within this review, symbol recognition is defined as a model-to-image matching problem, and the solution to the matching problem is defined as a branch-and-bound search algorithm. The following theoretical extensions and modifications[1] to the geometric matching approach of [48] are also presented: (1) The use of mixtures of feature types within the matching solution to provide flexibility in the input features. (2) Incorporation of contextual information on the features such as feature weights or neighboring information. (3) The development of more efficient variants of the search algorithm by reducing the size of the search space and the use of multi-image matching. (4) A feature penalization method for reducing the number of false matches.

2.1 Introduction

Recognition of symbols in line drawings (whether isolated symbol recognition or symbol spotting[2]) is similar to the object recognition problem from computer vision. A main approach for solving object recognition problems is the geometric matching approach [50, 51, 52, 53]. Geometric matching is also used in computer vision problems

[1]Parts of this chapter are based on the author's work in [49].

[2]See Section 1.1 for detailed definitions of these terms.

besides object recognition, such as: image registration [54, 55], document image analysis [56, 57, 58], graphics recognition [59] and spatial verification for content-based image matching and retrieval [60, 61]. Geometric matching is concerned about simultaneously finding the transformation and the correspondences between two sets of feature points using geometric relations. The two sets of features are usually extracted from two images.

This chapter reviews the geometric matching[3] approach described first in Breuel's [48], adapted to the problem of symbol recognition in line drawings. Only the theoretical and algorithmic aspects of the approach are reviewed within this chapter. The next chapter discusses the practical application of the approach in a symbol recognition and spotting method. Within the review of the geometric matching approach of [48], the symbol recognition problem is defined as a model-to-image matching problem. In this definition, the model and the image represent two line drawings, and are assumed to be defined as two sets of features. Next, the solution to the matching problem is defined as the branch-and-bound search algorithm of [48]. The search is performed in the transformation space to find the transformation that achieves the best correspondences between two feature sets. In this chapter, the term "recognition" refers to "recognition in context", this includes both isolated symbol recognition and symbol spotting. An example of using geometric matching in symbol recognition is shown in Fig. 2.1.

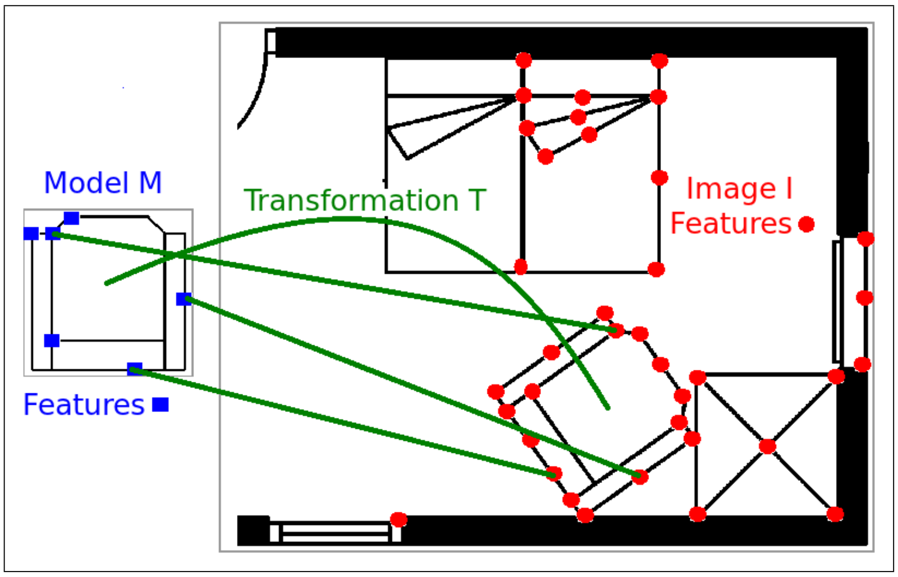


FIGURE 2.1: Geometric matching: The model features (blue square points) are transformed by a transformation T (green curve) to match a subset of image features (red circle points). This matching gives us the correspondences between the two feature sets. For clarity of showing, only few feature points along the edges are shown, and how some of them match.

[3]Within this dissertation, the term "geometric matching" refers to matching geometric objects, like points and lines, under transformations and some form of error bounds.

The review presented in this chapter illustrates the flexibility, robustness and powerfulness properties of the geometric matching approach of [48]. These properties (see below) represent the motivations behind choosing this particular geometric matching approach for solving symbol recognition problems.

These properties are summarized below, along with a number of extensions and modifications to the geometric matching approach based on these properties.

- Ability to recognize symbols in context (with large amounts of clutter).
- Precise localization of the recognized symbols.
- Ability to work with a wide variety of geometrically parametrized features such as: pixels, pixels with orientations, and higher level features such as line segments. The use of area features and mixtures of feature types is presented in this chapter as an extension to the matching approach of [48].
- Robustness to preprocessing and noise removal artifacts such as partial or missing features.
- Ability to incorporate contextual information of the application domain such as: feature weights and neighboring features information. The incorporation of such information into matching is also presented in this chapter as an extension to the matching approach of [48].
- Provision of feature correspondences and transformation information for the output matching symbols. This enables us to easily incorporate heuristic methods such as penalizing non-matched features, which improves the precision. This chapter presents a feature penalization method as a modification on the matching approach of [48]. The penalization can be either incorporated within the matching algorithm or as a post-matching step.
- Simplicity of the search algorithm, which leaves a big room for improvements. This chapter presents two variants of the search algorithm as modifications on the matching approach of [48] for increasing the search efficiency. The first variant depends on reducing the size of the transformation search space. This is done by discretizing the range of one or more dimensions of the transformation space. The second variant depends on matching a model to multiple images simultaneously.
- The base algorithm from [48] that is used for matching, returns a globally optimal solution under some error model [48].

2.2 Branch-and-Bound Approaches to Geometric Matching

This section reviews the geometric matching approach of [48], adapted to recognize symbols in line drawings under similarity transformations and a bounded error model. First, the symbol recognition problem is defined as model-image geometric matching problem. Second, the detailed algorithmic solution for the problem is discussed. Both the problem definition and the algorithmic solution are adapted from [48] for symbol recognition. The use of high level feature types within the matching solution is also adapted from [48] for achieving more efficiency of the matching. The following extensions to the geometric matching approach of [48] are also presented in this section: (1) the use of mixtures of feature types within the matching for demonstrating the flexibility of the matching approach, (2) the incorporation of feature weights and neighboring features information into the matching, for the purpose of improving the performance of the matching.

2.2.1 Symbol Recognition as a Geometric Matching Problem

First, the problem of matching a pair of images is defined. The image pair is: the **model** which is an isolated query symbol, and the **image** (an image of an isolated symbol or an image of a complete line drawing).

Assume that the images have been preprocessed, so that the model M and the image I are represented as two sets of features. The features could be pixels, pixels with orientation, line segments, arcs or areas. The matching solution can be defined for any kind of geometrically parameterized features. The model $M = m_1, m_2, ..., m_r$ has r features, and the image $I = i_1, i_2, ..., i_s$ has s features. In this subsection, the features are assumed to be pixels with orientation, and are called *point* features. Each point is defined by its pixel-coordinates and its gradient orientation within an image as (x, y, a), if the orientation a is not available due to noise, then the point feature is simply defined by its (x, y) coordinates. Higher-order features such as segments and areas will be dealt with in the next subsections.

An instance of the query symbol M can be inside the image I, but in that image, this symbol instance can be transformed with a transformation T which belongs to similarity transformations, so the symbol in the image can be a translated, rotated and/or scaled version of the query symbol M. Furthermore, the symbol instance inside the image I could be surrounded by lots of clutter. In complete line drawings, a symbol occupies just a small region within an image, the clutter in this case is the connecting lines and the other symbols in the image.

The goal is to find the transformation T_{max} that maps a **maximal subset** of the model features to **maximal subset(s)** of image features, with minimal error as defined by some error function. The error model used in this definition is the bounded error model [48] with the Euclidean-distance as a distance function.

A transformation T in the symbol recognition problem belongs to similarity transformations, so, a transformed model feature $T(m)$ is calculated for a model feature $m = (x, y, a)$ as follows:

$$T(m) = (tx, ty, ta) = \Big(\big(s \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} dx \\ dy \end{bmatrix}\big)^{\intercal}, (a + \alpha)\Big) \tag{2.1}$$

where s is the scale parameter, α is the rotation angle, and dx, dy are the translation parameters in the horizontal and vertical directions respectively.

The quality of mapping two sets of point-features using a given transformation T is calculated as:

$$\begin{aligned} &Q(T) = \sum_{m \in M} \max_{i \in I}[\ g((tx_m, ty_m), (x_i, y_i), \epsilon_d).f(ta_m, a_i, \epsilon_a)] \\ &\text{where} \\ &g(arg1, arg2, arg3) = 1 \text{ if Euclidean-distance}(arg1, arg2) < arg3,\ 0 \text{ otherwise} \\ &f(arg1, arg2, arg3) = 1 \text{ if } |arg1 - arg2| < arg3,\ 0 \text{ otherwise} \end{aligned} \tag{2.2}$$

In Eq. (2.2), the ϵ_d parameter is a user-defined error threshold that represents the maximum distance between two points to be considered matching to each other. Similarly, the ϵ_a parameter is the error threshold on the maximum angle difference between two points. $Q(T)$ in Eq. (2.2) will be referred to as **QoM** henceforth. QoM stands for quality of match[4]. Eq. (2.2) means that the QoM of a transformation is the count of the model feature points that match a subset of image features using that transformation. Each model point – when matched to one or more image points – contributes maximally by 1 to the sum over the model points, which means that the maximum QoM that can be achieved is equal to the number of model points.

The solution to the geometric matching problem defined above is to find T_{max} among all transformations T_{all} that maximizes the QoM function for a matching problem instance:

$$T_{max}(M, I, \epsilon) = \arg\max_{T \in T_{all}} Q(T; M, I, \epsilon) \tag{2.3}$$

where ϵ represents all error threshold parameters.

[4]Different $Q(T)$ functions will be defined in the remaining of this chapter. They all will be referred to as QoM.

2.2.2 Branch-and-Bound Search

Finding T_{max} in Eq. (2.3) can be achieved by searching in the 4D transformation space (2D translations, rotation and scale). The transformation space parameters are set as follows: $dx \in [-dm_x, di_x]$, $dy \in [-dm_y, di_y]$, $\alpha \in [0, 2\pi)$ and $s \in [0.5, 3.0]$, where dm_x is the horizontal dimension of the model and dm_y is the vertical dimension of the model, and similarly di_x and di_y are the horizontal and vertical dimensions of the image respectively.

For each of the possible transformations in the search space, the algorithm transforms the model features M using that possible transformation T, and then calculates the quality of mapping those transformed model features to a subset(s) of the image features I using Eq. (2.2). The transformation that achieves the maximum quality of mapping is the sought transformation T_{max} according to Eq. (2.3).

An exhaustive search in the transformation space is impractical, instead, the branch-and-bound search algorithm of [48] is used. This branch-and-bound algorithm implements a best-first search scheme, and returns a globally optimal solution [48]. The search algorithm is described in the following steps.

1. Create an initial search state (features: all model-image feature pairs $M \times I$, transformation region T: full transformation space), evaluate the initial state, and insert it in a priority queue Q.
2. Check stopping criteria, if met, stop and output solution T , else get the top search state S from Q.
3. Split the T region of the S state into T_1 and T_2 regions and form two new search states, evaluate them and insert them into Q.
4. Continue at step 2.

A search state (step 1 and 3) consists of two components: the list of the matching model-image feature pairs (called the *match-list* [48]), and a transformation region. The initial search state has all possible model-image feature pairs $M \times I$ pairs, and has the full transformation region. Evaluating a search state (step 1 and 3) means computing the QoM according to Eq. (2.2), and updating its match-list. Note that the transformation in a search state is not a fixed value, but rather a continuous range of values for each dimension of the transformation space. Such a range of values is called here a *transformation region*. In order to correctly evaluate the QoM using a transformation region (without loosing any solutions inside this region), Eq. (2.2) is used as follows: T is set to be the mid-value of each of the dimensions' ranges, and a value δ is added to the ϵ value in order to account for the uncertainty of using the whole transformation region. The δ value becomes smaller as the transformation region becomes smaller. The δ value is of course related to the upper and lower bound used by the search algorithm. For details on its computation, the reader is referred to [48]. After evaluation of a search state, an updated match-list of model-image feature pairs is available, and is associated with the search state. The updated match-list contains only the model-image pairs that match under the error bounds that are computed based on the transformation region of the search state.

Splitting a search state (step 3) is done as follows. Split the transformation region of the state along the largest transformation dimension, to create two new smaller transformation regions. Create two new states that both have the same list of matching feature pairs as the original state, but each has one of the two new smaller transformation regions.

One stopping criterion is when Q is empty, the algorithm terminates with no solution. This happens when no match with the minimum required QoM exists. The other stopping criterion is when the T region of the extracted top state from the queue is small enough. This means the algorithm has found a solution T and a list of model-image feature correspondences.

The second stopping criterion can be slightly modified to enable the algorithm to recognize/spot multiple instances of the model (the query) in an image. Once the algorithm finds one T result, instead of stopping, it continues splitting states to find the next best T of the state that has the next best QoM.

Insertion in the priority queue is done based on the computed QoM of a state. The QoM value serves as a key to sort the queue. Moreover, the states whose QoM is smaller than a minimum quality value, will be discarded (not inserted in the queue). This value is user-defined. For example, if the model M has m features, the user can define the minimum quality of mapping to be 75% of the model features. This means, for any solution to be accepted by the algorithm, at least 75% of the model M will always be matched to each of the model instances that exist in the image I. In this chapter and the following chapters, this minimum quality value is called the "**accept-match**" parameter, since it is represents a match acceptance criterion.

2.2.3 Matching with Higher Order Features

In complete technical line drawings, the number of extracted point or pixel features is huge, which really slows down the search described in the previous section. One main approach for performing an efficient search, is to use features of a higher order than point features, such as lines, arcs or shapes. The number of higher order features extracted from an image is much smaller than the number of pixel features of the same image. Moreover, higher order features usually contain more information than the simple features, this information puts constraints on what to be considered a correct matching. This leads to speeding up the search process. The geometric matching approach of [48] can be defined for any geometrically parameterized features, this subsection reviews the definition of the matching approach using both line segment features, and partial line segment features. After that, an extension to the geometric matching approach of [48] is presented as follows: given three feature types: points, segments and areas, the symbol recognition problem and solution are defined for matching a mixture of these types. When using different feature types or mixtures of them, only the QoM function needs to be redefined, whereas the rest of the algorithm remains unchanged.

Line Segment Features

Assume that a model and an image have been processed and defined as two sets of line segments. Matching the two feature sets is defined below, based on the definition of geometric matching of line segments in Breuel [48]. A line segment feature i in the image I is defined as $i = ((p, q), a_i)$, where p and q are the end points of the segment, each point is defined by its x, y coordinates, and a_i is the orientation (slope) of the segment. The slope can be computed once a line segment is defined by its two end points. A line segment m in the model M is defined similarly.

A transformed model segment is $T(m) = ((u, v), t_{a_m})$. Note that here, $T(.)$ is applied on line segments unlike $T(.)$ in Eq. (2.1) that is applied on points or pixels. u and v are the two end points of the transformed model segment, and each of them is computed according to Eq. (2.1). t_{a_m} is the slope of the transformed segment $T(m)$. The quality value of matching a single model feature to a single image features is called $q(T(m), i)$, and is computed as follows:

$$\begin{aligned}
&q(T(m), i) = f(t_{a_m}, a_i, \epsilon_a).g(p, l_{u,v}, \epsilon_d).g(q, l_{u,v}, \epsilon_d).\text{overlap}(T(m), i)\\
&\text{where}\\
&\quad l_{u,v} \text{ is the line passing through } u \text{ and } v \text{ ,}\\
&\quad g(arg1, arg2, arg3) = 1 \text{ if distance}(arg1, arg2) < arg3,\ 0 \text{ otherwise ,}\\
&\quad f(.) \text{ is as defined in Eq. (2.2) , and}\\
&\quad \text{overlap}(T(m), i) \text{ is the length of } i \text{ when projected onto the line}\\
&\quad \text{segment } T(m) \text{ along a direction perpendicular to } T(m).
\end{aligned} \tag{2.4}$$

Note that the length value of overlap$(T(m), i)$ is always $\leq$ length$(T(m))$.

The quality of mapping two sets of line segments given a transformation T can then be defined as:

$$Q(T) = \sum_{m \in M} \max_{i \in I} \left[q(T(m), i)\right] \tag{2.5}$$

This is analogous to the QoM function defined in Eq. (2.2). The maximum QoM that can be achieved in Eq. (2.5) is equal to the sum of the lengths of the transformed line segments of the model. The only transformation parameter that affects the lengths of the model features is the scale parameter.

Partial Line Segment Features

The QoM in Eq. (2.5) is defined for the case when the vectorization process is perfect such that each segment in a correct match inside the image corresponds to exactly one segment in the model. The query models (or library symbols) are usually clean images or are provided in vector format, however in the line drawing databases, the preprocessing and vectoriztion processes are rarely perfect due to noise. It is often the case that one line segment from a symbol is represented as two or more smaller line segments, or that a small part of the segment is missing due to preprocessing. This is illustrated in Figure 2.2.

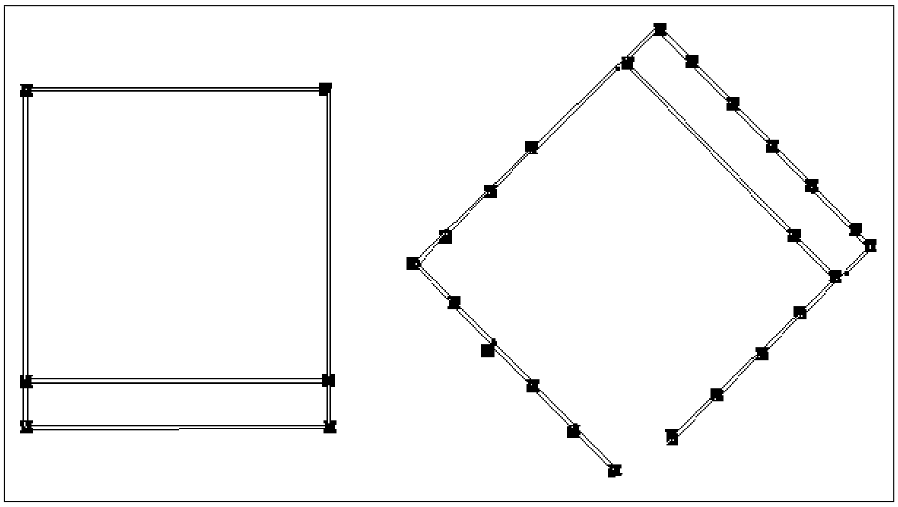


FIGURE 2.2: Preprocessing results in line segments features. The end points of the segments are marked by squared dots. On the left, the segments resulted from preprocessing a clean image, the line segments break only at junctions and corners. On the right, the segments resulted from preprocessing a noisy image, a line segment between two junctions is broken into many smaller line segments. The right symbol also has a missing segment.

For such cases, the matching of two feature sets is defined below, based on the definition of geometric matching of partial line segments in Breuel [48]. The quality of mapping two sets of line segments is defined as:

$$Q(T) = \sum_{m \in M} \min(\text{length}(T(m)) \ , \ \sum_{i \in I} q(T(m), i)) \tag{2.6}$$

In Eq. (2.6), a model segment can be matched to one or more image segments (one-to-many matching), but the maximum QoM per model segment must be $\leq \text{length}(T(m))$, and the total maximum QoM for a model is then the same as in Eq. (2.5).

Mixture of Geometric Primitive Features: Extension to the Matching Approach
Some preprocessing and vectorization processes provide two or more different types of features. This leads to having a mixture of features in the model and the image. The feature extraction module associates a label with each feature to indicate its type. In the following, various QoM functions are defined for mixtures of feature types.

In the different QoM functions defined below, we assume that there is a wide variety of of possible geometric primitive features in the model M and in the image I. For example, a model feature $m \in M$ is associated with a subscript $\in \{p, s, a\}$ that indicates the feature types: point, segment and area respectively, so, a model feature m_s is a line segment. The same applies to the image features $i \in I$. The transformation $T(.)$ of any feature type results in feature of the same type transformed by the transformation T, where transforming different features is carried out differently, and is explained after the equation where it occurs.

When the model and/or the image have more than one feature type, there are two options for carrying out the matching, depending on whether the goal is to match only the features of the same type together, or matching different types of features together. For the case when only features of the same type can be matched together, and the QoM is simply the sum of the QoM values that result from matching the different feature types. For example, the QoM function for matching segment and area features in the model to segment and area features in the image is:

$$\begin{aligned} Q(T) = & \sum_{m_s \in M} \min(\text{length}(T(m_s)) \; , \; \sum_{i_s \in I} q(T(m_s), i_s)) \; + \\ & \sum_{m_a \in M} \min(\text{area}(T(m_a)) \; , \; \sum_{i_a \in I} q(T(m_a), i_a)) \end{aligned} \tag{2.7}$$

where

$$q(T(m_a), i_a) = \frac{T(m_a) \cap i_a}{T(m_a) \cup i_a}$$

A model area feature m_a can be any convex polygon, and is $T(m)$ is calculated by transforming the line segments that represent the sides of the convex polygon, which produces the transformed area. If m_a is a circle, then $T(m_a)$ is calculated by transforming the center point just like point features, and multiplying the radius by the scale parameter of the transformation T. Alternatively, a circle can be represented by its polygon approximation and transformed as mentioned previously. The area overlapping criterion $q(T(m_a), i_a)$ is similar to the PASCAL VOC criterion [62] for evaluating the overlapping of bounding boxes. The transformation of a line segment $T(m_s)$ and length of a line segment are as defined earlier in this section, and area(m_a) is the area of a model feature of type "area" as defined by basic geometry.

Sometimes, matching different types of features together is needed. Consider the images in Figure 2.3. The symbols in the image are severely corrupted by noise. It might not be easy to extract segments or higher order features from that image, only pixel features can be extracted, but at the same time, higher-order features are available in the other images to which these symbols will be matched. As mentioned previously, to perform efficient search, it is desirable to have the higher order features. Hence, it is interesting to see how to match different kinds of features in one pair of images.

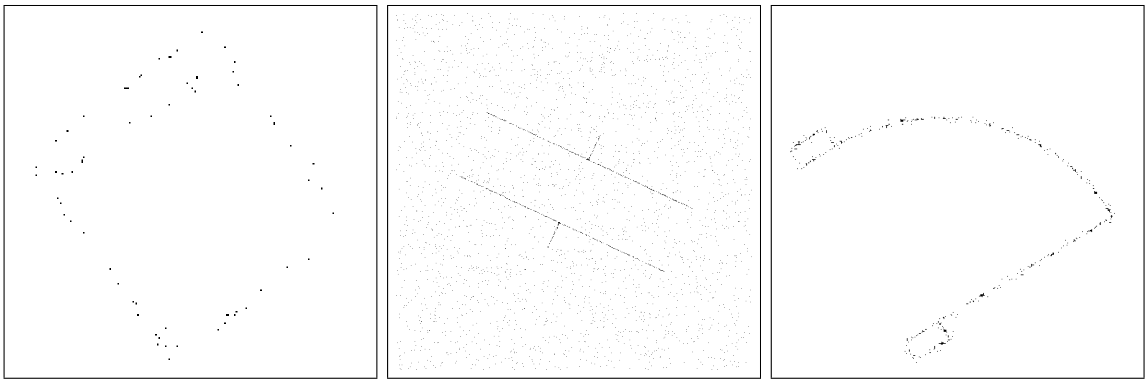


FIGURE 2.3: Examples of images corrupted by noise. It can be difficult to extract high level features from such images with a general simple preprocessing module. In this work, usually only point features (pixels without orientation) are extracted from such images.

Case 1: point features in the model, segment features in the image:

$$\begin{aligned} &Q(T) = \sum_{m_p \in M} \max_{i_s \in I}(f(t_{a_{m_p}}, a_{i_s}, \epsilon_a).g(T(m_p), l_{i_s}, \epsilon_d)) \\ &\textit{where}: \\ &\quad m_p \text{ is a model point, and } i_s \text{ is an image segment} \\ &\quad g(.), f(.) \text{ are as defined in Eq. (2.4)} \end{aligned} \tag{2.8}$$

Case 2: segment features in the model, point features in the image:

$$\begin{aligned} &Q(T) = \sum_{m_s \in M} \min(\text{length}(T(m_s))\ ,\ \sum_{i_p \in I} f(t_{a_{m_s}}, a_{i_p}, \epsilon_a).g(i_p, l_{u,v}, \epsilon_d)) \\ &\textit{where}: \\ &\quad m_s \text{ is a model segment, and } i_p \text{ is an image point ,} \\ &\quad T(m_s) = ((u,v), t_{a_{m_s}}) \text{ as defined earlier in the section , and} \\ &\quad g(.), f(.) \text{ are as defined in Eq. (2.4)} \end{aligned} \tag{2.9}$$

In both equations 2.2.3 and 2.2.3, if the point features are just pixels without orientation, then the terms that contain the orientation are simply removed. As for area features, matching points/segments to areas or vice versa does not make sense in the symbol recognition application. However, it is shown later in this chapter how to use empty areas in the model to penalize the false matches whose corresponding areas contain point or segment features.

2.2.4 Weighting Model Features: Extension to the Matching Approach

Some feature extraction modules associate weights with the extracted features. These weights refer to the discriminance information of the features, and can be used within the matching algorithm to improve its performance. This subsection only shows how the geometric matching approach incorporates feature weights. The effects of using the feature weights on the geometric matching performance will be discussed in detail in Chapter 4.

Again, only the QoM function changes. An abbreviated notation for the QoM functions will be used henceforth. All these functions compute a sum over the model features, each element of that sum represents the QoM for one of the model features, the computation of this element changes according to the feature types of the model and the image. So, the QoM value per model feature – any feature type – will be referred to as q. Eq. (2.2) for matching two sets of point features can be written with the new notation as follows:

$$Q(T) = \sum_{m \in M} q_m \tag{2.10}$$

By incorporating the feature weights, the QoM function becomes:

$$Q(T) = \sum_{m \in M} w_m q_m \tag{2.11}$$

where w_m is the weight value associated with a model feature m. The quality value for each model feature m is multiplied by the weight value w_m of that feature. Eq. (2.11) is general for all feature types, just be substituting the appropriate q_m computation.

2.2.5 Matching Neighboring Features: Extension to the Matching Approach

In images, the pixels (or features) of a local patch are assumed to be homogeneous (or similar). This assumption is used in many different image processing techniques like image restoration and edge detection. Moreover, in the field of object recognition and image matching, images are described using local regions based on the assumption that small local patches belong to the same object.

Analogously, it can be assumed that neighboring pixels or line segments in a line drawing belong to the same object. This assumption can be utilized to perform the matching search efficiently by assigning higher matching scores to neighboring features when they match the model features. When two neighboring features match two features in the model under a transformation T, this provides more evidence that the current T contains the sought solution, than if two non-neighboring features match two model features under the same T. By assigning a higher score for matching neighboring features – assuming that the features actually belong to the same object – then the QoM value for the transformation T will be higher and will go up in the priority queue, hence the search will converge faster to the correct match.

Scoring neighboring features can be directly incorporated within the geometric matching search algorithm. As mentioned previously, the model – the query – is always defined, and the information on which of its features are neighbors, is given. So, the QoM function considering neighboring features can be defined as:

$$\begin{aligned} &Q(T) = \sum_{i \in M} q_i + \sum_{i \in M} \sum_{j \in M} \alpha q_i \beta_{ij} q_j \\ &i \neq j \text{ , } 0.0 \leq \alpha \leq 1.0 \text{ ,} \\ &\beta_{ij} = 0 \text{ if } i \text{ is not beighbor to } j \text{ , 1 otherwise} \end{aligned} \tag{2.12}$$

The QoM function in Eq. (2.12) adds an amount to the score whenever two neighboring features are matched. One can change this amount from $q_i q_j$ to other desirable values.

Weights of model features can also be incorporated in Eq. (2.12) as follows:

$$Q(T) = \sum_{i \in M} w_i q_i + \sum_{i \in M} \sum_{j \in M} \alpha q_i \beta_{ij} q_j \tag{2.13}$$

2.3 Efficient Variants of the Geometric Matching Search

This sections proposes two variants of the geometric matching search algorithm reviewed in Subsection 2.2.2. The two variants are modifications on the search algorithm described in Breuel [48] for the purpose of performing a more efficient search.

Reduction of Transformation Space Size

The size of the transformation space is one of the main factors that slow down the search algorithm described in Subsection 2.2.2. A large transformation region means a large uncertainty ($\epsilon + \delta$) of the transformed model features (See Subsection 2.2.2). This causes more model-image feature pairs to be considered as matching pairs, which in turn, makes the size of the match-lists bigger. The evaluation time of a search state depends on the size of the match-list. The size of the match-lists can only reduce, when the transformation region (hence the uncertainty) becomes small enough so that not all image features are included as matches. Also, a large transformation regions means that more search states will be expanded before the pruning of the search space starts. The number of the states grows exponentially during the initial stages of the algorithm before it starts to reduce rapidly. The initial exponential growth of the the states is prolonged when the size of the initial transformation region is large, since a lot of the states will have a promising QoM value.

In order to make the algorithm run more efficiently, the reduction of the transformation space size is considered. The size of the transformation space is determined by the transformation dimensions and the ranges of these dimensions. One dimension of the transformation space is chosen to be discretized. Usually the dimension that slows down the search the most is chosen, but other dimensions can be discretized as well. Consider the ranges of the search space dimensions mentioned in Subsection 2.2.2. The rotation range is the bottleneck. Although the 2D translation ranges are the largest in our problem (larger than the image dimensions), the translation values are added to the points coordinates not multiplied. The scale range should usually be limited, otherwise the lines' thicknesses largely change. In case of large scale range, many models of the same shape with different sizes can be used. The rotation however, spans the range $[0, 2\pi)$ and greatly affects the speed. Using a discrete rotation range would speedup the search but of course with sacrificing the optimality that comes from covering the whole continuous search range.

The rotation range is discretized to 144 values (with a step of $2.5°$). Recall from Subsection 2.2.2 that the initial search state is created such that it contains the whole transformation region. Instead, 144 initial search states are created such that each state has the same information except for the transformation region. Each state will have the same initial list of all model-image feature pairs, and the transformation region of the state will have one fixed rotation value different from the other states, but the same ranges for translations and scale. The produced initial states are considered as sub-problems of the original matching problem with the full rotation range. The initial states are then evaluated and inserted in the priority queue, which brings the problem back to one matching problem. The matching search proceeds exactly the same way as if those initial states resulted from splitting one initial state. The small transformation regions in the initial states will cause big reductions in the sizes of the match-lists quickly. Moreover, many states will be quickly recognized as non-promising due to small QoMs, hence not explored further.

Combining the sub-problems (the initial states) is very important. If the sub-problems are evaluated as separate problems sequentially, only the memory requirements would be smaller, but the total run time would be slower. When the sub-problems are combined, the states that look non-promising will be at the bottom of the queue, and the promising ones have small transformation regions. In the case of large problem size (large number of image features, large search space), the speedup will dominate the overhead of loading and evaluating multiple initial states. If ϵ_a (the error threshold allowed for matching the orientations of a pair of features) is chosen to be larger than the rotation step size, a solution can be found if it exists, within the precision of ϵ_a.

Practically, this works well for the symbol recognition problem. There are many cases in technical line drawings where the application domain knowledge tells us that a symbol can only be found in specific orientations. For example, the “window” symbols are found in horizontal or vertical orientations only. If the window symbol is queried, then only 4 initial states with fixed rotation values can be created. Such limited rotation range greatly speeds up the search.

In this search variant, the algorithm is not given the full search space (but the other inputs are the same), this means that the states in the queue have different regions and different matching pairs, this makes it hard to compare the time and memory requirements of this search variant with the original search algorithm. Besides, it is difficult to predict the performance of branch-and-bound algorithms on the average case. Practically, this variant is faster by several orders of magnitude. In branch-and-bound search schemes, faster usually indicates smaller memory requirements.

Multi-Image Matching

The geometric matching solution presented previously handles only one pair of images at a time. A more efficient solution is presented here to match a model with multiple image simultaneously. In the case of matching one pair of images, the recognition of an unknown symbol as one of a group of library symbols means that, the unknown symbol will be matched against each of the library symbols sequentially. The unknown symbol will be recognized to be the library symbol that achieves the highest matching quality with the unknown symbol.

A more efficient way of performing the recognition would be to match the unknown symbol to all the library symbols at the same time. This is explained in the following. Assume in the model-image pair to be matched, the unknown symbol is the model and a library symbol is the image. Instead of loading the features of this pair in one initial state of the algorithm, all the image pairs (unknown symbol vs each library symbol) can be loaded in the search queue as different initial states. That means one search process is carried out, with the model as the unknown symbol, and multiple images.

This idea is similar to the multiple initial states, but here the transformation space ranges are roughly the same (the translation range can be different depending on the dimensions of the image of each library symbol). The difference between the initial states is that each initial state has its own list of feature pairs. Again, the non-promising library symbols will be the bottom of the queue, and the search requirements will quickly become equivalent to matching only one pair or few pairs (the unknown symbol vs. the most similar library symbol(s)). Practically, the running time will be a little larger than evaluating one or m pairs, where $m \ll n$ library symbols. The overhead of loading multiple initial states is not significant, but the memory requirements are larger than having one initial state of one image pair, as there will be a number of branching operations before the search starts to converge.

It is worth mentioning that, there are other possible techniques for improving the efficiency of the search algorithm. For example, using application domain knowledge to force constraints on the search space and the features. But in order to keep the algorithm as general as possible, the features are kept as basic geometric primitives, and no domain specific knowledge is used. This way, the algorithm can be applied to different kinds of technical drawings.

2.4 Penalizing Non-matched Features

This section proposes a penalization method as a modification on the geometric matching approach of [48] in order to decrease the number of false matches found by matching approach. The reduction of the number of false matches is done by dealing with the problem of symbols that are very similar to each other.

Assume that the symbol in Fig. 2.4(a) is a query symbol to be recognized. It can be matched against the different available library symbols. Among those symbols, there are some symbols that are very similar to each other like the two symbols in Fig. 2.4(b) and Fig. 2.4(d). Since the geometric matching aligns the points of the model on the points of the image, the test symbol in Fig. 2.4(a), will match both symbols in Fig. 2.4(b) and Fig. 2.4(d) with the same QoM value. The matching results are marked in red in Fig. 2.4(c) and Fig. 2.4(e). It is clear that the same number of the query symbol features were matched to both images, simply because the shape of symbol (b) is a subset of the shape of symbol (d). This might cause the problem of recognizing the test symbol in Fig. 2.4(a) to be the symbol in Fig. 2.4(d) instead of what it truly is: (Fig. 2.4(b)).

This problem is solved by applying penalties on the non-matched features inside the matching area. Assume the features marked with zig-zag blue in Fig. 2.4(e) are represented as 4 line segments, that means there are 4 non-matched features in the found match. The score of the whole match will be penalized by some value for each non-matched feature. This will result in a lower overall score for matched symbols that have non-matched features, which results in better recognition accuracy in isolated recognition, and a smaller number of false positives in symbol spotting. The penalty value is calculated as follows. Each of

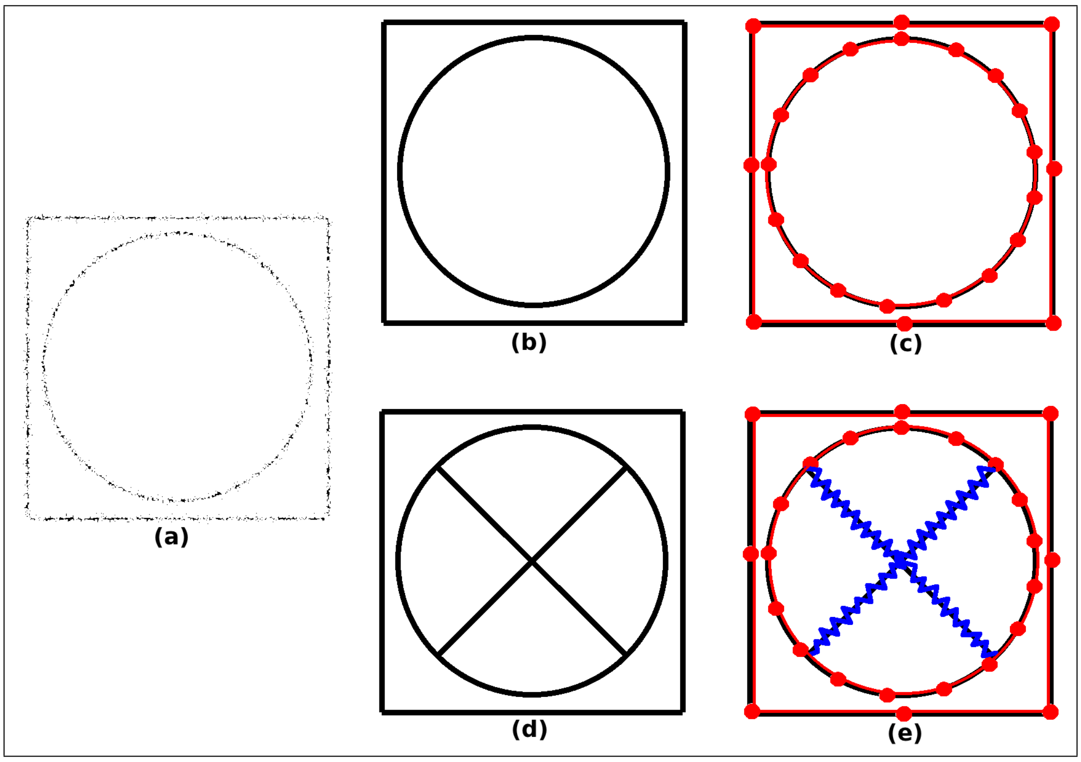


FIGURE 2.4: Matching similar symbols' shapes: (a) query test symbol (b) library symbol-1 (c) result of matching (red) the query symbol in "a" to the library symbol-1 (d) library symbol-2 (e) result of matching (red) the query symbol in "a" to the library symbol-2, some features (blue zig-zag) are not matched, because they do not exist in the query.

the non-matched features has a feature size. For each non-matched feature a value that is proportional to the feature size – for example half the feature size – is subtracted from the overall QoM. In the case of line segments, the size is the length of the line segment, and for point features, the size is 1, as defined in the QoM functions in the previous sections.

Before discussing the penalization procedure, a couple of remarks are in order. First, the problem of similar shapes, does not arise in the following case: if the symbol we want to recognize is the same as the symbol in Fig. 2.4(d), then the quality of matching it to Fig. 2.4(d) will be higher than matching it to Fig. 2.4(b). That is because each extra matching feature increases the overall QoM.

Second, the idea of a applying penalties easily generalizes to the non-isolated cases as follows. Having a number of candidate matching regions in a line drawing, we apply the same penalties by examining the features that are inside the region of a match. This is illustrated in Figure 2.5. Only the features that are inside the matching area are examined. The matching area is the bounding box that includes the matched features (See Figure 2.5(b)). That means the non-matched features that are outside this area are **not** penalized. This is important for this procedure to work on non-isolated symbols, so, the background and the lines connected to a symbol will not play any role in this penalization procedure.

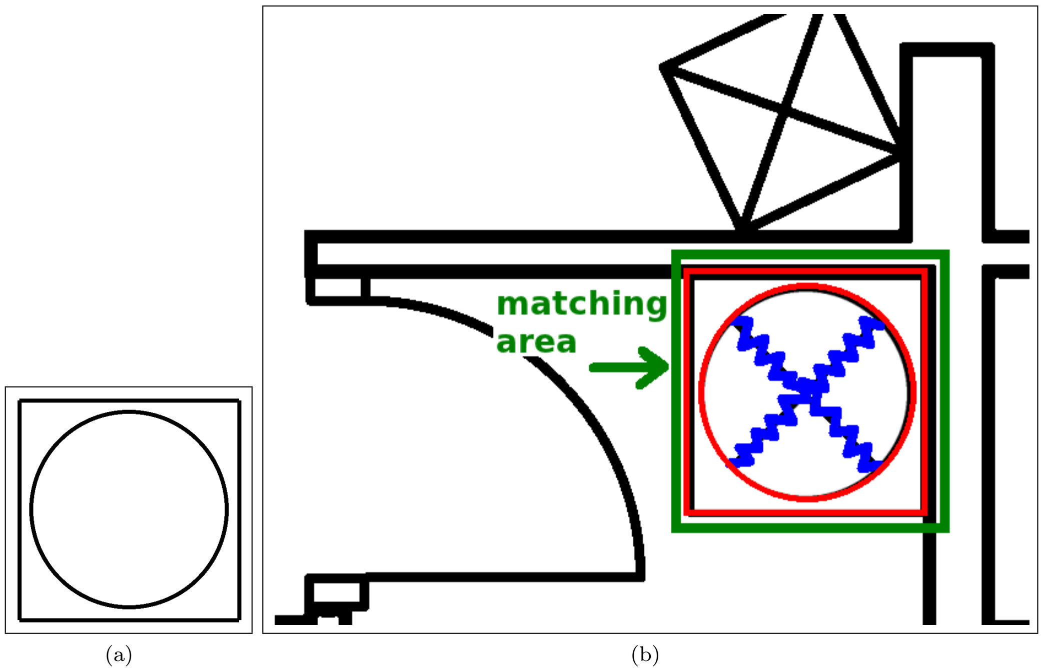


FIGURE 2.5: Penalizing non-matched features for non-isolated cases (spotting): (a) query symbol (b) The algorithm finds the matching region (marked in green) with the feature correspondences, some of the features inside the region match to the query (marked in red), and some features do not match (marked in blue zig-zag lines), this match will be penalized.

Two equivalent methods of penalizing non-matched features are presented. The first method applies the penalties within the matching search algorithm and outputs the final matches directly. The second method applies the penalties after the matching is done by examining the feature correspondences between the query and the resulting matches. Both methods yield similar performance in terms of improved precision, however, the second one is faster. In the first method, the uncertainty inherent in the penalties will not get resolved until late in the search, which leads to slowing down the search. The second method can be considered as a fast post-processing step after finding the matches.

Incorporating Empty-Region Penalties into the Geometric Matching Search

Consider the matches in Figures 2.4 and 2.5 again. In the false matches, there are some non-matched features within the matching area of a false match. Those features are spatially located in an area that corresponds to an empty area in the model. Since the models of a symbol library are known, additional features such as "empty areas" can be defined along with the original model features. If a feature in the image (point or segment) is matched to this empty area of the model, then the QoM function is penalized. Figure 2.6 shows some symbols once with and once without additional "empty area" features. For example, the "arm-chair" symbol on the left most of the top row of Figure 2.6 has 13 segment features (the end points of the segments are shown in the figure as red squared dots). The corresponding "arm-chair" symbol on the left most of the bottom row of Figure 2.6 has 14 features: 13 line segments and 1 additional empty-area feature of type “rectangle”. This can be done for the models (the library symbols), but for the images, it would require longer and more complicated preprocessing. So, the representation of the images is kept as before: point or segment features.

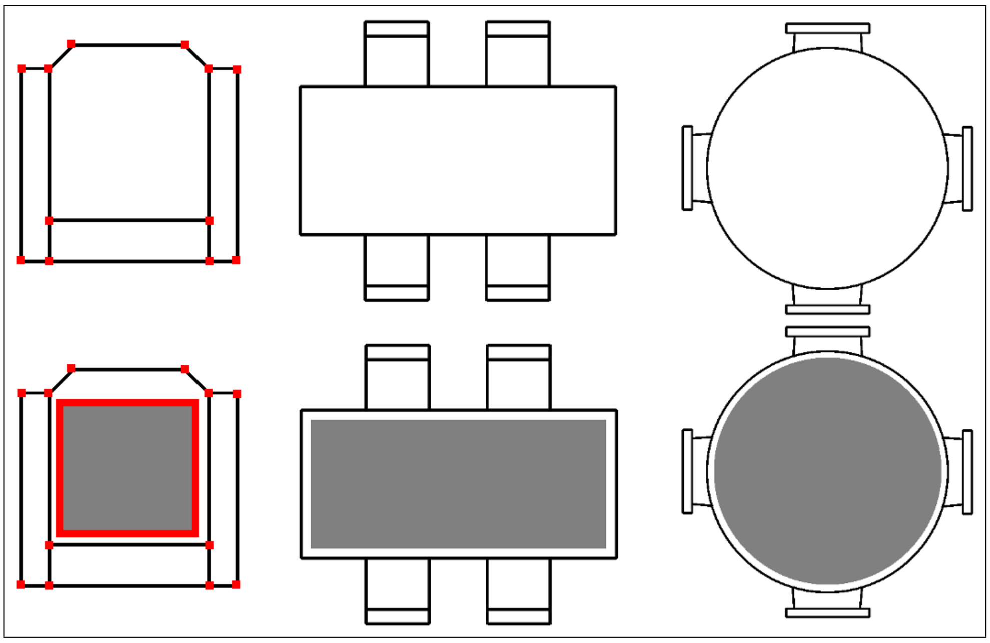


FIGURE 2.6: Adding area features to library symbols. The top row shows the original library symbols. The bottom row shows the corresponding symbols with area features added to them. The shaded areas correspond to the empty areas in the original symbols. The left most symbol of the top row has 13 segment features (the end points of the segments are shown as red squared dots). The corresponding symbol in the bottom row has 14 features: 13 line segments and 1 additional empty-area feature of type “rectangle”.

The QoM function for matching a model represented as segments and areas with an image represented as segments is defined as follows:

$$\begin{aligned} Q(T) = \sum_{m_s \in M} \min(\text{length}(T(m_s)) \;,\; \sum_{i_s \in I} q(T(m_s), i_s)) - \\ \sum_{m_a \in M} \sum_{i_s \in I} \alpha \,\text{length}(\text{frac}(i_s)) \text{ if } \text{frac}(i_s) \text{ matches } T(m_a) \end{aligned} \tag{2.14}$$

where α is an experimentally-set parameter to control the amount of penalty to be deducted, $0.0 < \alpha \leq 1.0$, and frac(i_s) is the fraction of the image segment i_s that lies inside the transformed model area $T(m_a)$.

The test of whether frac(i_s) matches $T(m_a)$ is done as follows. A point p is inside a convex polygon if p lies on the same side of all the area polygon segments. If both p and q are inside $T(m_a)$, then frac(i_s) $= i_s$. If only one of the end points p or q is inside, then the point of intersection $p_{intersect}$ between i_s and a segment of $T(m_a)$ is found, and frac(i_s) is the length of the segment between p or q and $p_{intersect}$. If the m_a is a circle, then a point p

is inside the circle if the distance between p and the center of the circle is smaller than the radius. All these computations are from basic computational geometry.

As explained in Subsection 2.2.2, The QoM is calculated for transformation *regions* rather than transformation values. It was explained that, in the QoM function, a value δ is added to the ϵ value in order to account for the uncertainty of using the whole transformation region. This means that in the early stages of the matching search, a lot of the features will be considered matching features. Later in the search, as T gets smaller, δ also gets smaller, and only the matching features will pass the matching test of being only within a distance of $\approx \epsilon$ of the transformed model point. In Eq. 2.4, the QoM decreases when features in the image match the empty areas in the model. It would not matter in the early stages of the search whether an image represent a correct or a false match. In the correct matches, the deduction of QoM value will only stop later in the search. Such behavior will make the search slower, as correct matches go down in the priority queue. Eventually though, the search will result in correct matches and will reject the false ones.

Re-scoring Matches after the Geometric Matching Search
After the regular matching search is done, the algorithm outputs a set of matches, each match has a list of the correspondences between the model features and the image features. An axis-aligned bounding box of the matched image features is computed. This bounding box is considered the matching area. A quality value is associated with each image feature inside the matching area. This quality value is calculated as the sum of the sizes of the model features matched with this feature. Each image feature inside the matching area is examined, if it does not achieve a minimum quality value, then the overall QoM is penalized. The penalty value is calculated as discussed previously. This results in a lower QoM for the false matches.

This penalization method is equivalent to the previous one. The features that will be penalized are the ones that are located within the areas that correspond to the area features defined in Figure 2.6. Implementing this penalization method is much easier and faster than the previous one. Moreover, all the resulting matches before penalization are available in case one wants to output not only the symbols that are the same as the query, but also the symbols that are largely similar to it.

In both penalization methods, the penalty values are chosen experimentally by the user. In Chapter 4, machine learning approaches will be used with the geometric matching approach to learn how to automatically filter out the false matches. The methods to be presented in Chapter 4 offer principled approaches from machine learning for improving the precision of geometric matching as opposed to the two heuristic penalization methods discussed above.

2.5 Comparison with Other Approaches to Geometric Matching

Many algorithms have been developed for solving the problem of geometric matching. As mentioned at the beginning of this chapter, geometric matching algorithms try to jointly solve for correspondences and transformation between two point sets. These algorithms try to find the transformation that best aligns the model-image features by searching either in the transformation space or in the feature correspondence space. A complete review of these techniques is beyond the scope of this thesis. Some of the main ideas in the geometric matching field are reviewed in the following.

A well known geometric matching algorithm is the generalized Hough transform (GHT) for detecting arbitrary shapes [50]. The GHT tries to find the unknown transformation by dividing the transformation space into bins, and evaluating each bin according to the number of pairs aligned by the transformation of the current bin. The bin that achieves the highest value is considered the sought transformation. GHT requires careful tuning of the sizes of the bins, and it does not accurately account for the error of feature location. Furthermore, for large transformation ranges and large number of features it is very slow and is more likely to output false positives [63, 64]. Many variants and improvements have been developed for the GHT [65, 66, 64].

RANSAC [67] is a widely used algorithm that works by selecting a random set of model-image feature pairs, and computes the transformation that puts them into correspondence. This step is repeatedly carried out till finding the transformation that brings the largest number of feature pairs into correspondence. Although RANSAC outputs a transformation that is inaccurate (because of error on the locations of the features), it is very popular in image registration and as verification step after matching. Many variants and improvements have been developed for RANSAC [68, 69], where it can be seen that RANSAC (and its variants) are mainly used for estimating homographies not as a main matching algorithm for object recognition.

The RAST (Recognition by Adaptive Subdivision of Transformation Space) algorithm [51] uses a branch-and-bound search in the transformation space. This algorithm combines practicality with the ability to return a globally optimal transformation that maximizes the overall number of feature correspondences. Many variants and improvements for RAST were presented; some of them can be found in [48, 70, 71]. Variants of the RAST algorithm have been used in line finding [72, 59], object recognition [53], motion estimation [73] and document image analysis [56, 57, 58]. The variant of the RAST algorithm presented in [48] is used as the base algorithm in the geometric matching approach presented in this chapter. The superiority of the algorithm compared to other geometric matching algorithms has been discussed in [74, 48] from certain theoretical and practical aspects.

2.6 Discussion

This chapter has reviewed the theoretical and algorithmic aspects of the geometric matching approach of Breuel's [48] adapted to the problem of symbol recognition in line drawings. Within the review of the matching approach, the symbol recognition problem is defined as a model-to-image matching problem. Then, the algorithm introduced by Breuel [48] is used as a base algorithm for matching. This algorithm has been proven to find the globally optimal solution for the defined matching problem under a certain error model [48]. The algorithm posses intrinsic characteristics that makes it attractive for performing symbol recognition. The main characteristic is the ability to perform recognition in context with large amounts of clutter. In technical line drawings for example, a symbol occupies just a small region within a line drawing, with the rest of the features of the line drawing represent clutter around the desired symbol. The algorithm is also robust, it performs matching even with partial or missing features.

A couple of theoretical extensions to the geometric matching approach of [48] have been presented in this chapter. First, mixtures of feature types are used within the matching approach. Second, contextual information of the features are incorporated within the matching approach. It is argued that these extensions help improve the performance of the geometric matching approach, and provide flexibility in the types of the input features.

Different modifications of the geometric matching solution have also been presented in this chapter, for the goal of having improved performance. To achieve efficiency, two variants of the search algorithm have been presented. The first uses a discretized search space, and the second searches in multiple images simultaneously. To improve the precision, a feature penalization method has been presented. This method penalizes the non-matched features of the matches found by the geometric matching search algorithm.

From theoretical and algorithmic points of view, the matching approach reviewed in this chapter seems to be quite suitable for dealing with symbol recognition in the context of line drawings. It is also quite flexible for incorporating different kinds of improvements. The use of geometric matching seems to be the right solution for dealing with textureless shapes such as line drawings, as it depends on shape features like geometric primitives, and geometric relations for matching the features. The literature shows successful and effective application of the chosen matching algorithm in different fields, as well as its superiority to other geometric matching algorithms such as Hough transform.

The next chapter shows the practical application of geometric matching approach reviewed in this chapter in symbol recognition and spotting problems. The next chapter proves experimentally that the use of geometric matching is very promising for performing recognition in line drawings, and it outperforms state-of-the-art methods for the same problem.

Chapter 3

Symbol Recognition and Spotting Based on Geometric Matching

Symbol recognition and symbol spotting are core problems in the automatic analysis of line drawings. This chapter presents a geometric matching-based method for both isolated symbol recognition and symbol spotting[1]. The method uses the geometric matching approach presented in Chapter 2 as the main recognition module, hence, the method has the ability to recognize symbols in the presence of large amounts of contextual clutter. The presented method achieves significantly lower error rates than the state-of-the-art recognition and spotting methods. This is demonstrated on different databases including the databases of symbol recognition and spotting contests: GREC-2005 and GREC-2011.

3.1 Introduction

Recognizing a symbol in the context of a line drawing is faced by the following challenging problems: the large amount of background clutter, the transformations on the symbols within a drawing, degraded (or noisy) drawings and large number of symbols. Current approaches to symbol recognition and spotting deal poorly with these challenges, specially with the contextual clutter. Current symbol recognition approaches are based on pattern recognition methods that work well only on isolated objects (see Section 1.2). As for the symbol spotting approaches, they always require finding regions of interest within line drawings as a first step, the techniques that these approaches use for finding regions, do not deal well with clutter and noise (Chapter 5 discusses this issue in great detail). This chapter presents a recognition and spotting method that deals with these challenges significantly

[1]Parts of this chapter are based on the author's work in [75, 49].

better than state-of-the-art recognition and spotting methods. Being based on the geometric matching presented in Chapter 2, the method presented in this chapter posses the same desirable characteristics of the geometric matching approach discussed in Section 2.1 such as: providing precise localization of the recognized symbols within the line drawings, and providing the features' correspondences and the transformation between the query and the recognized symbols as additional outputs. Most state-of-the-art methods do not posses such characteristics.

As discussed in Chapter 1, symbol recognition (whether isolated or in-context) is a core task in the automatic analysis of line drawings. Isolated symbol recognition is concerned with recognizing an unknown query symbol as one of a set of known library symbols. Symbol spotting is defined as locating a query symbol within a line drawing. Chapter 1 discusses the symbol recognition and spotting tasks and their important applications in more detail. This chapter contributes to solving the symbol recognition-in-context task by presenting a symbol recognition and spotting approach based on the ideas discussed in Chapter 2.

The symbol recognition and spotting method presented in this chapter consists of two main modules: First, the preprocessing and feature extraction module, which includes adaptive denoising, edge detection and the representation of the line drawings as geometric primitive features. The adaptive denoising is developed to deal with extremely different noise types including clean documents, it automatically infers the noise type and invokes the appropriate preprocessing steps. Edge detection is carried out using simple morphological operations. The features are extracted by sampling points or line segments along the edges. The second module is the recognition module. Within this module, the different algorithms and methods discussed in Chapter 2 are applied on both recognition and spotting tasks.

The method presented in this chapter is evaluated on a number of databases of technical line drawings including both real scanned images and synthetic images. The databases include the standard graphics recognition databases of the GREC-2005 and GREC-2011 symbol recognition contests. Based on the experimental results presented later in this chapter, the geometric matching-based method is highly accurate in performing recognition and spotting tasks in different types of technical line drawings. The geometric matching-based recognition method successfully recognizes symbols in the presence of preprocessing artifacts such as partial or missing features. This shows experimentally the robustness of this matching approach (discussed in Subsection 2.2.3). For spotting tasks, the recognition method is able to recognize symbols in the presence of large amounts of clutter (the clutter in the line drawings is the other symbols and the background connection lines). Furthermore, the efficient search variants discussed in Section 2.3 achieve the same accuracy but with a significant speedup over the original version of the search algorithm discussed in Subsection 2.2.2. The penalization method discussed in Section 2.4 is used as a post matching step for eliminating the found false matches.

This chapter also presents a comparison between our recognition method and the state-of-the-art recognition and spotting methods that use the same or similar line drawing databases. The comparison shows that the recognition approach presented here significantly outperforms state-of-the-art recognition and spotting methods. Overall, the results suggest that geometric matching is a promising approach for analysis of technical line drawings.

3.2 Geometric Techniques for Symbol Recognition and Spotting: State-of-the-art

A review on symbol recognition and symbol spotting methods has been presented in Section 1.2. This section reviews only the symbol recognition and spotting methods that use geometric techniques, whether for feature extraction, recognition or spatial verification.

Coustaty et al. [22] used a geometric matching technique for feature extraction in an isolated symbol recognition system. They used an adapted Hough transform to extract the segments of the symbol. First, a mean-filter is applied on a symbol to remove noise, then only skeleton points are used as input for Hough transform to reduce the complexity resulting from a large input. The lines are detected by Hough transform, then they are projected back to the image to detect their end points, hence extracting line segments as features. The extracted segments are arranged in a topological graph as a structural signature of the symbols, and then, Galois Lattice classifier is used for recognition.

Wong et al. [76, 77] developed a Hough-based representation for the lines of isolated symbols. The Hough representation is modified to be invariant to translation, rotation, scale and reflection. The translation invariance is achieved by making a line representation relative to the symbol centroid. The reflection invariance is achieved by considering the absolute value of the angle in the Hough line representation. The rotation invariance is done by projecting the angle dimension onto the position dimension. This results in a 1-D vector, which is normalized to get scale invariance, additionally, certain values are truncated from the vector to remove noise components. In [76], they used the sum of absolute differences to compare the descriptors of symbols, while in [77], they trained a knowledge-based hierarchical neural network to perform recognition. The methods of Wong et al. [76, 77] are based on ideas from the methods in [78, 79] that use similar Hough-based representation.

Some recognition and spotting methods used modified shape context. Matching by shape context [80] can be considered a geometric-related technique. Su et al. [81] presented an isolated recognition system based on multi-resolution shape context. In their work, the shape context descriptor is computed for each symbol skeleton point by considering nested regions of different sizes around the point. Weights are associated with the descriptors of the different regions by giving higher weights to the larger regions. They also added other weights related to the complexity of the region contents around a point. They claimed that computation of a pyramid of shape contexts at each shape point makes the method robust to noise. As known, shape context is invariant to scale, and the rotation invariance is achieved by considering the tangent direction at the point as the x-axis. The matching between the descriptors of a symbol against library symbols is done by finding point correspondences via the Hungarian algorithm. This is called weighted bipartite matching, and it is the same as presented originally for matching by shape contexts [80], except they also consider matching across different region sizes. The robustness to noise in their system has been tested on a part of the GREC-2005 database, however, the system has not been tested on transformed symbols.

The spotting method by Nguyen et al. [40] also used shape context to describe parts of symbols within the regions of interest. After detecting regions of interest within a line drawing, the shape context descriptor is computed locally on those regions. The descriptors from all the drawings are then clustered using k-means and represented as visual words. The visual words of a query are then compared to the visual words of the line drawings. The cosine distance is used as a similarity measure instead of the shape matching proposed originally for shape contexts. They used the same method for recognition of isolated symbols in [82].

As discussed in Section 1.2, symbol spotting methods usually follow the scheme of finding regions of interest, describe them with some descriptor and index them based on a query using an appropriate indexing technique. The scheme followed in these spotting methods is unrelated to our recognition and spotting method. The method presented in this chapter uses geometric matching as a core recognition/spotting component. Within the recognition/spotting process, the method searches for matching symbols in an image whether it contains an isolated symbol or a complete line drawing, without identifying or locating regions of interest first. The method also does not use descriptors of any kind.

However, some of state-of-the-art spotting methods use geometric techniques as a final verification step. Rusinol et al. [32, 10] used a Hough transform-based voting scheme to verify the correctness of the initial candidate matches found by the spotting method. The correct matches cast more votes in locations than the false matches. In [32], the Hough-voting space is a 3D space: 2D position coordinates and scale, while in [10], the space is the 2D space of the spatial positions.

In summary, geometric matching has not been used so far as a main recognition module in symbol recognition or spotting methods. Geometric matching techniques such as Hough transform have been used for feature extraction, invariant symbol representation and for spatial verification after matching. This chapter shows that the use of geometric matching for recognition is able to solve the symbol recognition in context problem with lower error rates than state-of-the-art methods.

3.3 The Symbol Recognition and Spotting Method

This section describes the two main modules of the symbol recognition and spotting method. The preprocessing and feature extraction module is described in Subsection 3.3.1, and the recognition module for isolated recognition and spotting is described in Subsection 3.3.2.

3.3.1 Adaptive Preprocessing and Feature Extraction

Although independent of the actual matching, preprocessing has an important effect on matching results [25], specially when the images are degraded by extremely different noise types. As a first step in the preprocessing module, an adaptive noise reduction algorithm is used. This algorithm is inspired by the adaptive noise reduction algorithm for line drawings developed by Zhang et al. [30]. Zhang's algorithm is based on the assessment of noise distributions and line widths, and then, according to this automatic assessment, different median and morphological filters are applied on an image. Inspired by the idea of the assessment of the image noise to automatically infer the noise type, an adaptive denoising algorithm is developed for the recognition approach presented in this section. The algorithm is called "adaptive" since it automatically infers the noise type and adapts to deal with it accordingly.

The noise assessment of the adaptive denoising algorithm presented here is based on: the number of small connected components, the sizes of these components and the stroke width (line width) in an image. The algorithm removes noise components (pixels or groups of pixels) while keeping as much information of the foreground symbols as possible. In many cases, the noise components are large, and its density distribution is higher than foreground symbols. Hence, instead of using actual median filters, the small connected components that are smaller than a certain size are simply removed from an image. This size is adaptively and automatically determined based on noise distribution. This operation has the same effect as a median filter, but it removes noise components of different sizes with the same filter template.

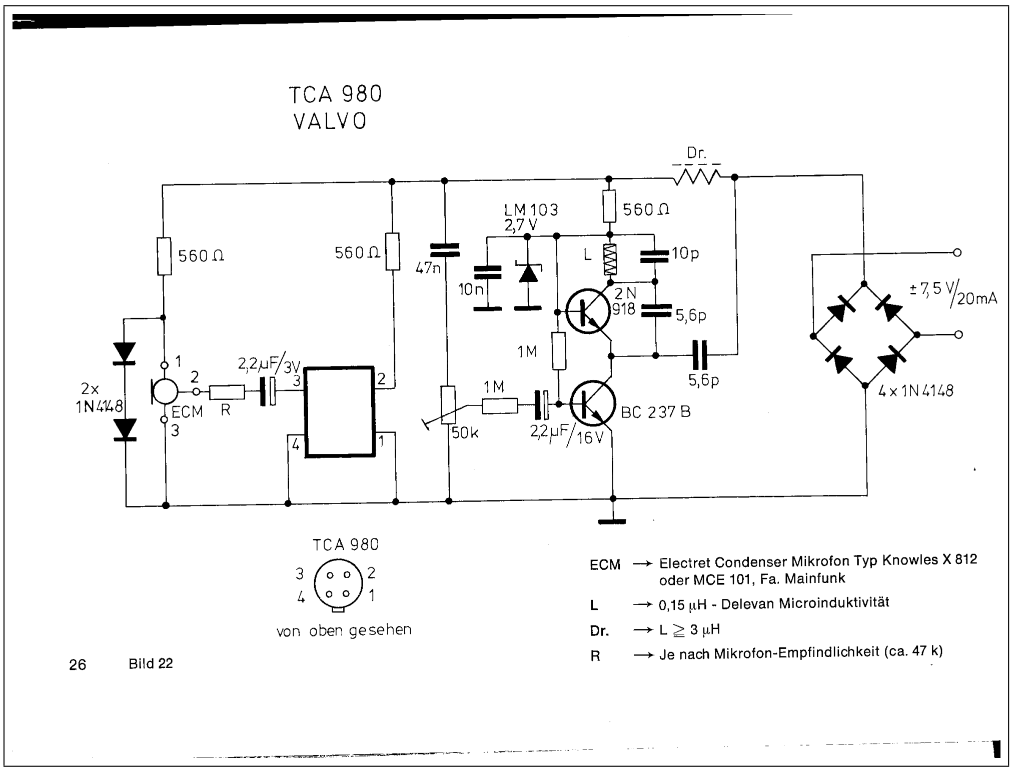
The second step of the preprocessing module depends on the inferred noise type from the previous step. There are two cases, the first is if the images are relatively clean, like scanned documents or images that are slightly degraded. In this case, the preprocessing proceeds by morphological edge detection: morphological closing and dilation by 1 are applied separately on an image. Then the morphologically-closed image is subtracted from the original image, and separately, the morphologically-dilated image is subtracted from the morphologically-closed image. Finally, the two images resulted from the subtraction operations are added together. The result of that is a thinned edge image, in which the edges are the inner and outer contours of all the lines within a line drawing. Fig. 3.1 shows the results of applying the preprocessing steps discussed so far in case of scanned images of real technical line drawings. Before applying the second preprocessing step, the gradient of the denoised image is computed. The gradient will be used to get orientation information of the features in the next steps.

The second case is if the images are degraded with heavy noise. In this case the preprocessing proceeds by a smoothing step. The smoothing is needed to get better results for the subsequent feature extraction step. A Gaussian filter is applied on an image followed by global binarization and then thinning. The size of the Gaussian filter and the binarization threshold are also adaptive, they are chosen based on the previously mentioned criteria. The thinning operation is not carried out when the inferred noise type indicates that the lines of the foreground pixels are so thin that they are composed of disconnected dots or dashes. Fig. 3.2 shows the results of applying the preprocessing steps discussed so far in case of noisy images. The figure shows images of symbols with different noise types from the GREC-2005 database.

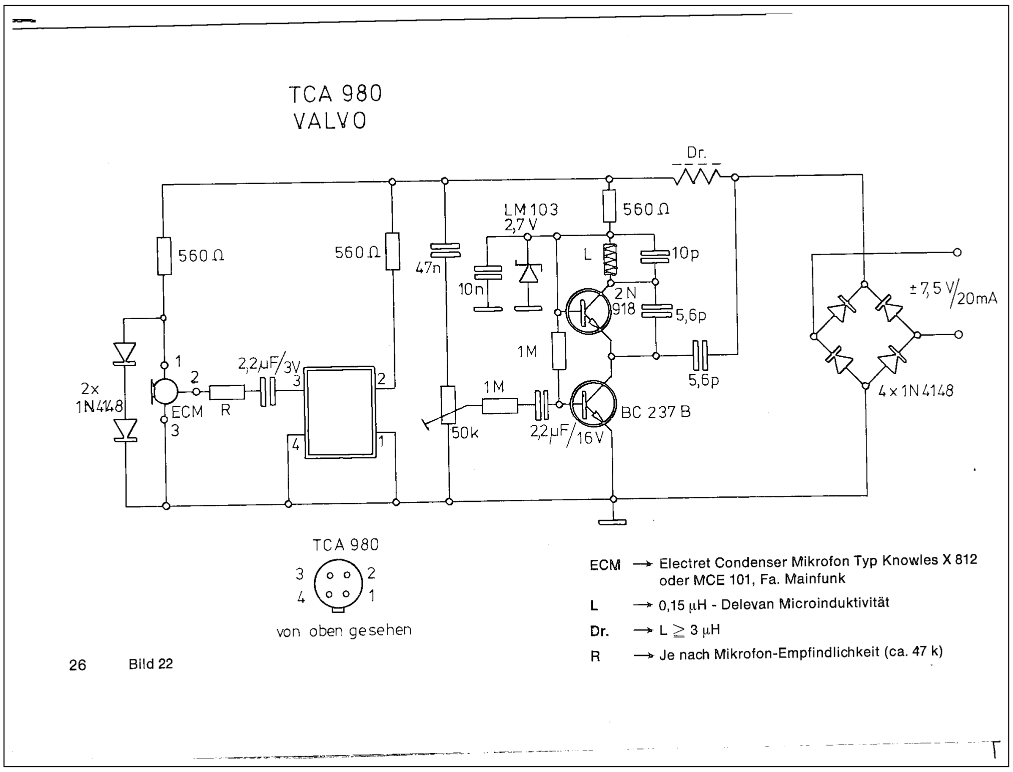
The third and last preprocessing step is the feature extraction step. Either points (pixels) or line segments are sampled along the contours of the edge image. For point features, equi-distant pixels are extracted as features, each extracted pixel is associated with an orientation. For segment features, line segments are extracted with a specified minimum length, segments that are shorter than this length are discarded. The extracted features are the geometric primitives that are used as input to the recognition module.

For relatively clean scanned images, using simple morphological preprocessing operations produces a good enough input for the recognition module presented here. It is also the case for images that are slightly degraded. Such images can actually be processed as clean or noisy images. As discussed in Subsection 2.2.3, the recognition algorithm can deal with partial or missing features. However, in the severely degraded images, the lines' thickness largely increase or decrease because of the noise. If only edge detection were applied, the recognition module would not be able to match such image to the library symbols since the corresponding lines do not have the same thickness (with considering the scale factor). Also thinning cannot be directly applied as it is very sensitive to noise and very thick lines. That makes it necessary to remove the noise components and smooth the lines before edge detection or thinning steps.

The images of the databases of the graphics recognition contests GREC-2003 up to and including GREC-2011 [45, 15, 16, 17] are degraded by different binary synthesis noise types. The noise types imitate real world noise that come from scanning or otherwise. The noise types are generated using Kanungo's method [83]. Some of the types result in heavily degraded images [16], and some of the images are extremely corrupted [15, 17] and cannot be recognized even by a human. Some of these extreme cases are shown in Figure 3.3.



(a) The input image: a clean image of a scanned line drawing.



(b) Preprocessed Image: the output image after applying noise removal and morphological edge detection. The noise removal has no effect on relatively clean image such as the input image in (a).

FIGURE 3.1: Preprocessing: noise removal and morphological edge detection (a) input image (b) edge image of thin contours.

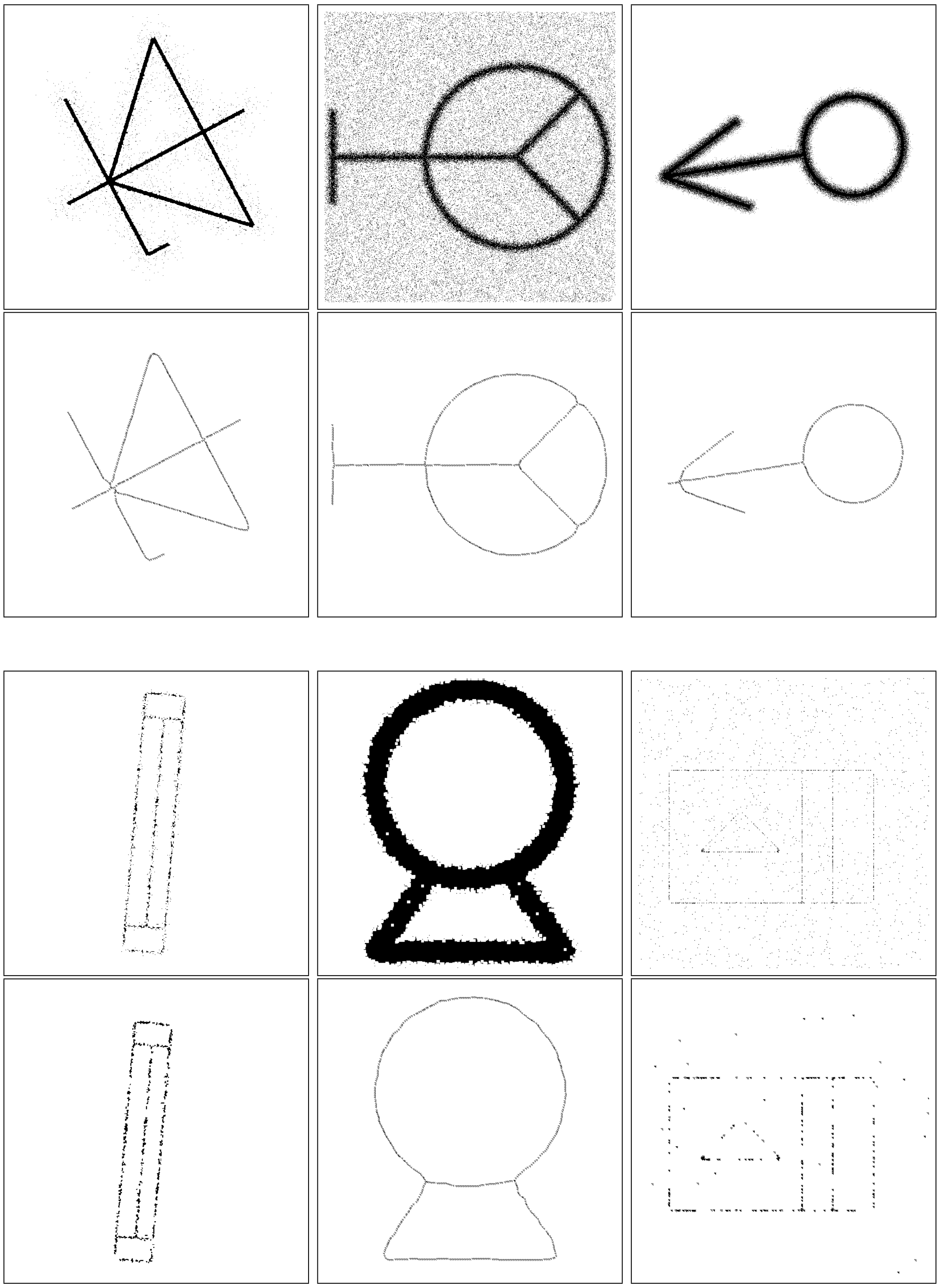


FIGURE 3.2: Preprocessing of degraded images: The first and third rows are the input images. The corresponding images in the second and forth rows are the output images after applying the preprocessing steps for degraded images.

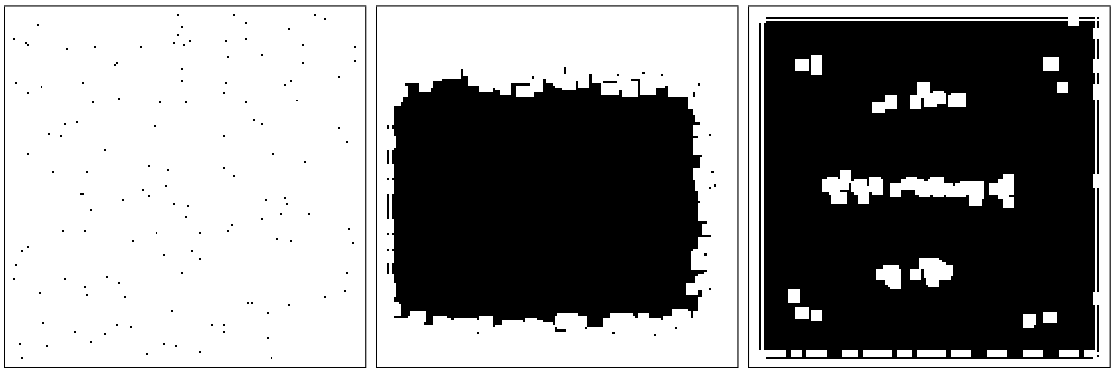


Figure 3.3: Examples of extremely corrupted images from the GREC-2005 database. Such symbols cannot be recognized by a human. The recognition method presented in this chapter also fails to recognize these symbols.

In general, all the noise types used in the different contests can be roughly divided into two main types. The first type makes the strokes of the shape irregularly thicker, with or without surrounding noise by adding corruption pixels to the pixels constituting the shape. The second noise type makes the strokes of the shapes irregularly thinner (or even disappear), with or without surrounding noise. Usually in case of an image degraded by the second noise type, the thinning step (within the preprocessing module described in this section) is not applied after binarization, the feature extraction step takes place directly.

3.3.2 Recognition based on Geometric Matching

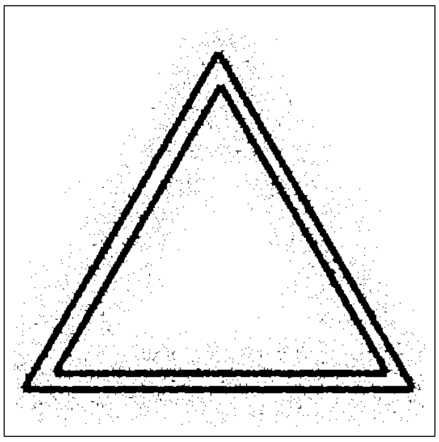
The inputs to the recognition module are the feature representations of two symbols or of a symbol and a line drawing. The feature representations are the result of the preprocessing and feature extraction module discussed in the previous section.

The geometric matching approach discussed in Section 2.2 is used as a recognition module for performing both isolated symbol recognition and symbol spotting. The geometric matching approach matches two line drawings: a model and an image. In isolated recognition, the models are clean library symbols (from the symbol alphabets provided in the databases). The models are represented as line segment features. The images are the noisy transformed test symbols, and they are represented by pixel (without orientation) features. For a lot of the test symbols, it is usually hard to extract features of a higher level than the pixels because of noise.

The task is to recognize an unknown query symbol (or test symbol). The query symbol is matched against the predefined set of library symbols one by one. The detailed procedure to carrying out the geometric matching is discussed in Subsections 2.2.1 and 2.2.2. Dealing with mixture of point and segment features is discussed in Subsection 2.2.3. The multi-image matching scheme discussed in Section 2.3 of Chapter 2 can be also used as follows. The pairs of the test symbol with each of the library symbols are loaded into the geometric matching algorithm as initial states, and the matching proceeds as if it were one big matching problem. The details of this procedure can be found in Section 2.3. The matching results is the same in both matching schemes, however the time and memory requirements differ.

After the matching, the penalization method discussed in Section 2.4 is carried out as a post matching step. As discussed in Section 2.2, the matching outputs the feature correspondences and the transformation between the query and a matching symbol. This information is used to find out the non-matched features and penalize them as described in Section 2.4.

The query symbol is recognized to be the one that achieves the highest matching score among the library symbols. Figure 3.4 shows a query symbol and its three best matches.


(a) A test symbol.

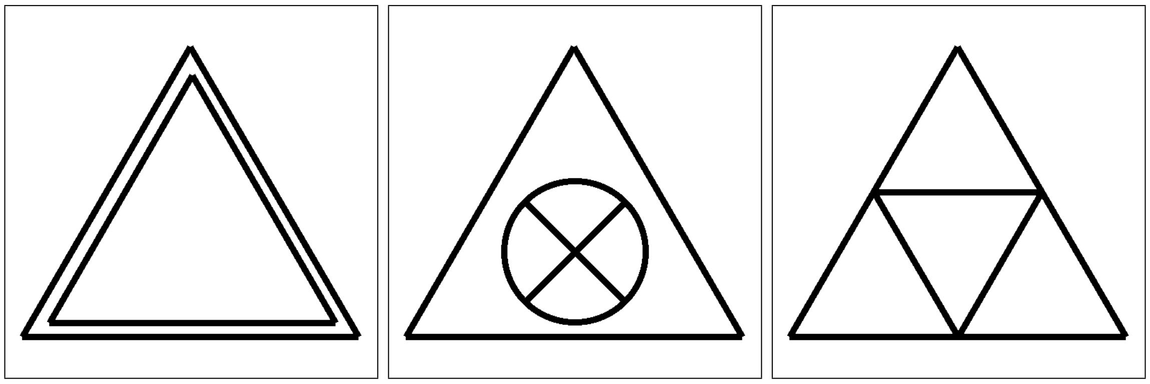

(b) The three best matches of the test symbol in (a).

FIGURE 3.4: Isolated recognition results. The test symbol in (a) is correctly recognized. The left most symbol in (b) is the best match (the symbol that achieved the highest matching quality with the test symbols). The second and third best matches are also similar symbols.

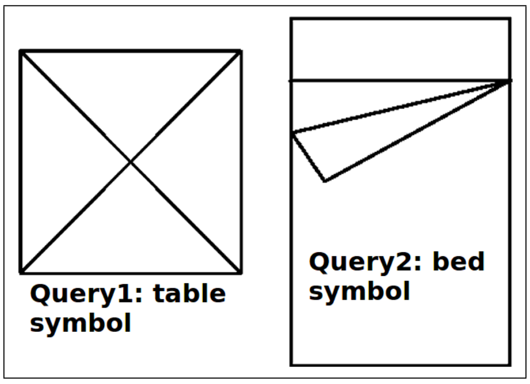
For symbol spotting, the models are symbols from a symbol alphabet from some application domain (called query symbols), and the images are complete line drawings from the same application domain. The images contain either line drawings only, or both text paragraphs and line drawings. The symbols within a line drawing could exist at different orientations and scales, and can be anywhere in the image. Both the models (query symbols) and the images are represented by line segment features. The spotting (or recognition in context) is carried out as described in Subsections 2.2.1 and 2.2.2. Dealing with segment features is discussed in Subsection 2.2.3. Since the image are large line drawings with thousands of line segment features, the spotting is carried out using the efficient matching variant presented in Section 2.3, namely the variant that uses the discretized rotation range.

A query symbol is matched against a line drawing, and all the regions that contain an instance of the query symbol in the line drawing are located. There could be one or more instances of the query symbol in an image, or no instances at all. The recognition module should be able to correctly find those instances or announce that there is no instance of the queried symbol in an image. Same as is done for isolated recognition: after the matching, the penalization method discussed in Section 2.4 is carried out as a post matching step. The penalization is applied on the features that are inside the matching region (see Section 2.4 for details). After the penalization, the recognition module rejects the regions that have an overall matching quality that is smaller than the value defined for accepting the matches.

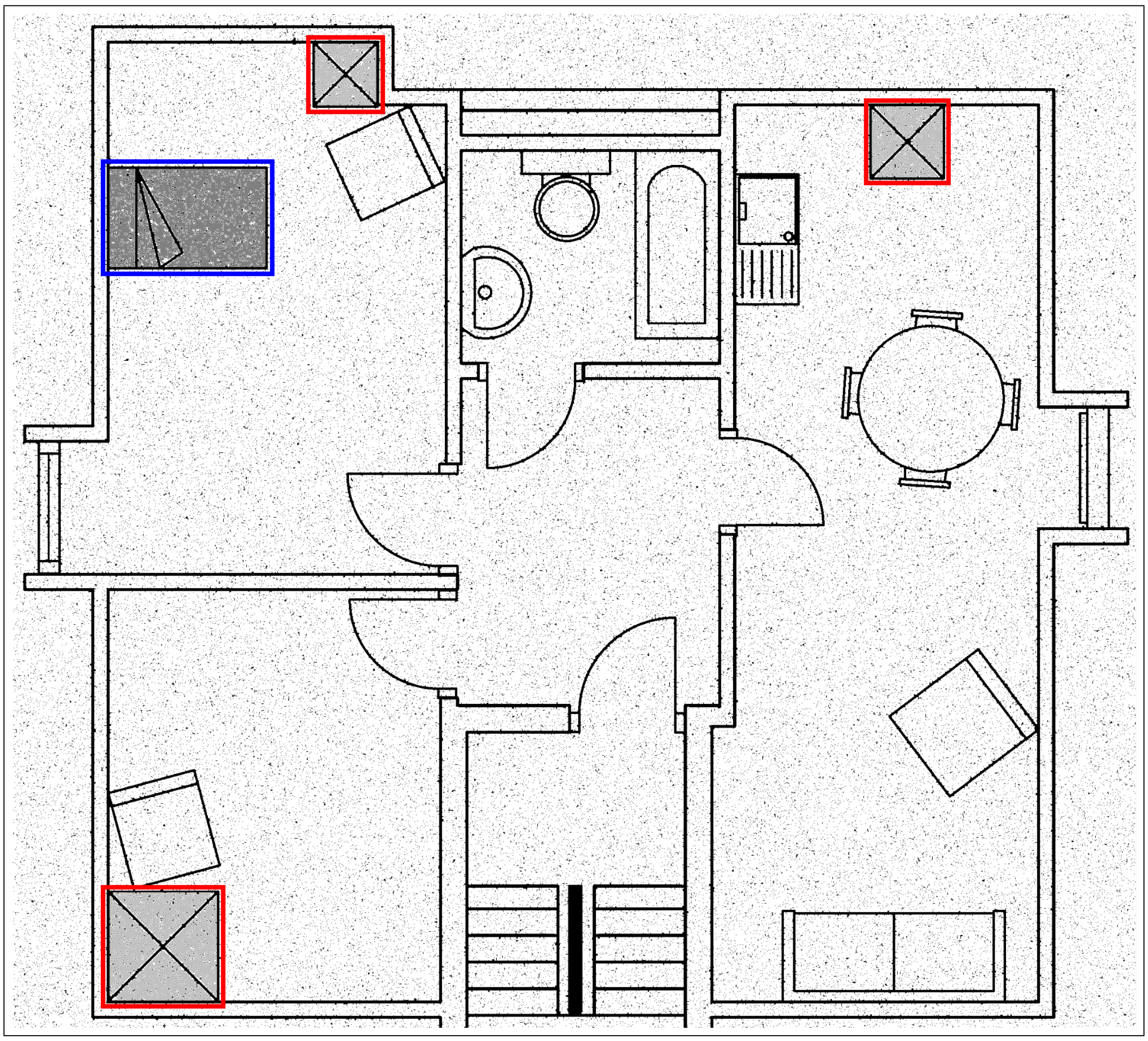
Figure 4.4 shows the spotted instances of two query symbols in a line drawing of an architectural floor plan from the training set of the GREC-2011 database. The queries are taken from the isolated query symbols provided with the database. Figure 3.6 shows another spotting example in an electronic circuit diagram of a real drawing scanned from a book. The query is a symbol cropped from another drawing. In both figures, the query instance found by the system are surrounded by shaded bounding boxes. As can be seen in the figures, the recognition module provides precise localization of the region of a spotted query instance within a line drawing.

3.4 Experiments and Performance Evaluation

This section describes the databases used for evaluating the performance of the method presented in this chapter. The experimental set up of the method parameters is also described. Finally the results of both the isolated recognition and the spotting are presented.

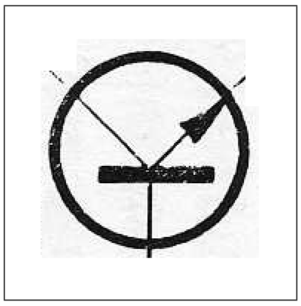


(a) Two different query symbols (table, bed).

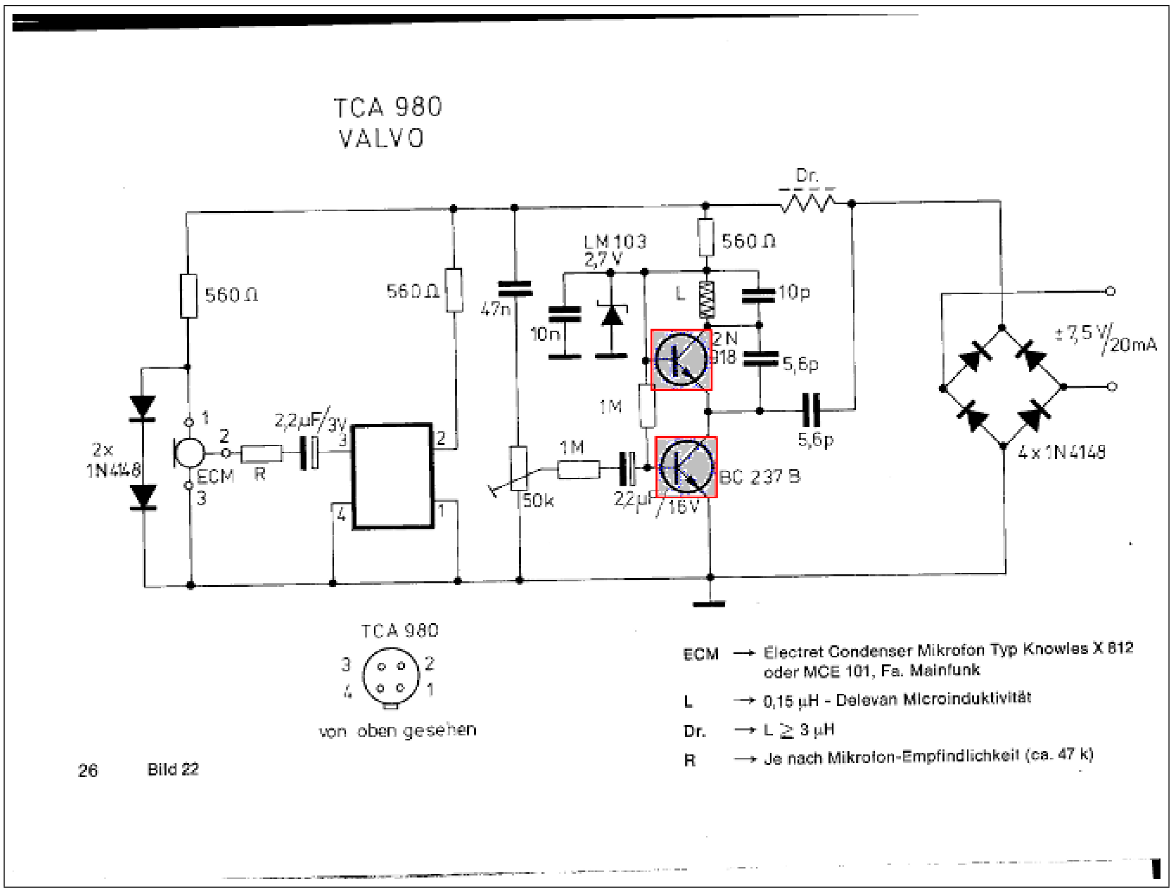


(b) Spotted instances of the query symbols. The results of both queries are merged into one figure.

Figure 3.5: Spotting results on an architectural floor plan from the GREC-2011 database. The spotted instances are shown surrounded by bounding boxes with shaded interiors (a red box for the table symbol, and a blue box for the bed symbol).



(a) Query symbol (transistor).



(b) Spotted instances of the query symbol.

FIGURE 3.6: Spotting Results on a scanned image of a real electronic line drawing. The spotted instances are shown surrounded by red bounding boxes with shaded interiors.

3.4.1 The Databases

A number of different databases are used for evaluating the performance of the recognition method presented in this chapter. These databases include tests for both isolated recognition and spotting. For isolated recognition, the test set of the GREC-2005[2] database

[2]http://iapr-tc10.univ-lr.fr/index.php/symbolrecognitionhome

and a subset of the training set of the GREC-2011[3] database are used. Both databases are publicly available, and they were used for symbol recognition contests. both databases have a symbol library of 150 symbols segmented from architectural and electrical drawings. The symbols are composed of lines and circular arcs, they are available in both raster and vector formats.

The symbols in the test images of the GREC-2005 database are corrupted by 6 different synthetic noise types, and transformed by different similarity transformations (rotation and/or scaling). All the 6000 test images of the GREC-2005 database are used in the evaluation. Similarly, the symbols in the images of the GREC-2011 database are corrupted by noise and transformed. There are 3 noise types with up to 25 levels of degradation within each type. A subset (3150 images) of the training set of the GREC-2011 are used in this evaluation. For more details about the databases and their characteristics, the reader is referred to [15, 17].

For symbol spotting, three databases are used. The first databases consists of 22 scanned book pages that contain electronic diagrams. The pages are scanned at 300dpi resolution, using a regular scanner and without any additional set up. The second database is the symbol spotting training set of the GREC-2011 database. It has 20 architectural drawings and 20 electrical drawings, also corrupted by 3 different noise types. There are 16 architectural and 21 electrical queries. The queries are segmented symbols from architectural and electrical drawings. The third database is the "Sysed" graphics recognition database generated in [43] and [44]. This database will be used for evaluation in the next chapters for the purpose of comparing the performance of the recognition method with improved versions of it. In this section, the first two mentioned spotting databases are used for evaluation.

3.4.2 Experimental Setup

The geometric matching approach presented in Chapter 2 is the main recognition component used in the method presented in this chapter. The parameters of this approach are the user-controlled parameters of the symbol recognition method presented in this chapter. First, the error threshold parameters ϵ_d and ϵ_a from Eq. 2.2 in Chapter 2 are set as follows: $\epsilon_d = 6.0$ and $\epsilon_a = 0.15(radians)$. The accept-match parameter discussed in Subsection 2.2.2 is set to 0.7. The transformation ranges are set the same as mentioned in Subsection 2.2.2. The penalty values for non-matched features as discussed in Section 2.4 is set to 0.5 of the size of the non-matched feature.

The same parameter setting are used for both isolated recognition and spotting. However, sometimes some parameters are automatically changed by the method when the adaptive

[3]http://iapr-tc10.univ-lr.fr/index.php/symbolrecognitionhome

noise removal module detects that the image is severely corrupted by noise. Such images have a lot of corrupted features that might not be matched, so the accept-match parameter is lowered to 0.3, and the error threshold ϵ_d increased to 10.0.

The method presented in this chapter has the advantage of having only few parameters to be controlled by the user. Those parameters and their effects are well defined in the geometric matching approach discussed in Chapter 2.

3.4.3 Experimental Results

First the isolated recognition performance is evaluated. The recognition system described in the previous section is evaluated on *all* of the tests of the GREC-2005 database, and also a random subset of the GREC-2011 tests. The recognition accuracy metric is used for evaluation. The recognition accuracy is the number of the correctly recognized symbols over the number of all the test symbols. This evaluation metric is used to evaluate all of state-of-the-art isolated symbol recognition methods, and in the previously mentioned symbol recognition contests. Tables 3.1 and 3.2 show the results of testing with GREC-2005 and GREC-2011 databases respectively.

Table 3.1: Isolated recognition results on all the tests of GREC-2005 database (6000 images). The results are divided according to the degradation and transformation types used in the test images.

	noise1	noise2	noise3	noise4	noise5	noise6
no transformation	100	100	100	98.2	91.0	96.2
rotation	98.0	99.0	97.2	98.7	94.0	73.7
scaling	98.5	96.5	96.0	94.0	87.5	58.5
rotation+scaling	97.0	91.5	92.0	92.0	72.5	41.5

Table 3.2: Isolated recognition results on a subset of the training images of the GREC-2011 database. N is the number of test images in each different case, total=3150 images.

Noise type (no transformation)	noise A (N=750)	noise B (N=750)	noise C (N=750)
Recognition accuracy %	98.6	94.0	97.0

Transformation & random degradation	rotation (N=300)	scaling (N=300)	rotation & scaling (N=300)
Recognition accuracy %	97.0	96.4	97.0

In general, the recognition accuracies decrease with scaling more than with rotation, because the scaled down versions of the symbols are severely corrupted even with simple noise types, and they do not have much information about the symbol at the first place. Examples of such images were shown earlier in the previous section in Figure 3.3. Those images are hard to recognize even by humans. Other less extreme examples are shown in Figure 3.7, however, they still cannot be recognized by the method.

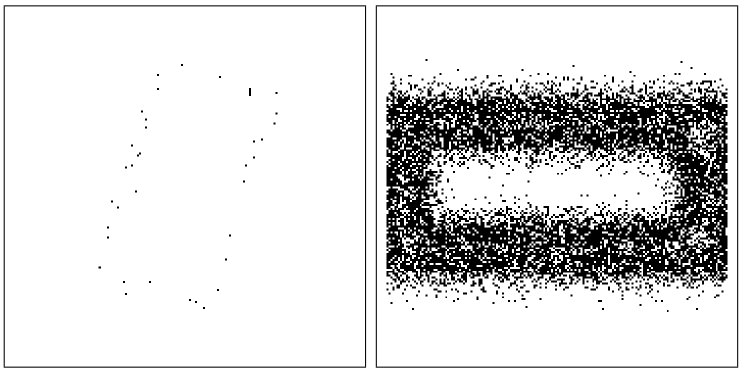


FIGURE 3.7: Examples of scaled down corrupted symbols. The recognition method fails to recognize those symbols.

The scalability of the method is tested on the GREC-2005 database. Figure 3.8 shows that the system scales well as the number of library symbols increases.

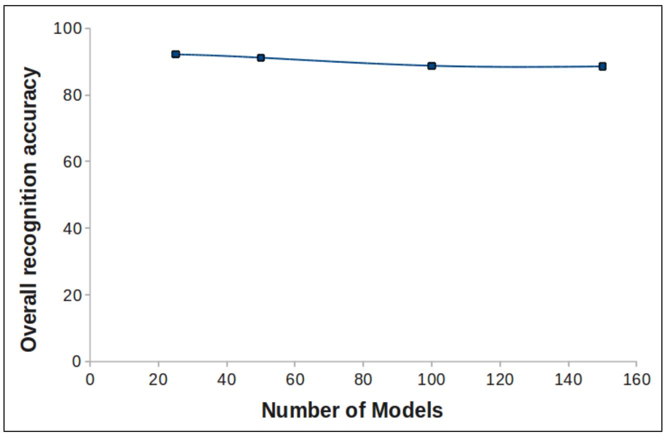


FIGURE 3.8: Scalability of the recognition method. The isolated recognition accuracy is shown as a function of an increasing number of library symbols.

The ability of the recognition method to perform spotting tasks is evaluated next. As discussed earlier in this chapter, a spotting method outputs regions that contain instances of a query symbol. The standard recall and precision measures are adapted to evaluate the spotted regions. For a query, "Recall" is defined as the number of correctly spotted symbols over the total number of ground-truth symbols. "Precision" is the number of correctly spotted symbols over the total number of spotted symbols.

An instance of a query within a line drawing is counted as “correctly spotted” according to the PASCAL VOC criterion [62] as follows. Assume the area of the bounding box of a spotted symbol instance is B_s, and the area of the bounding box of the ground-truth symbol instance is B_{gt}. The overlap ratio between the two areas is defined as:

$$a_o = \frac{B_s \cap B_{gt}}{Bs \cup B_{gt}} \tag{3.1}$$

A symbol is counted as correctly spotted if a_o in Eq. 3.1 is $\geq 90\%$. Alternatively, a symbol is counted as missing if a_o is $< 90\%$. The area-overlapping threshold that is commonly used in the evaluation of the spotting systems [46, 17] is: $a_o \geq 75\%$ for correctly spotted symbols. We use a more strict evaluation criterion since our method locates query instances very precisely.

Similar evaluation measures are used in the performance evaluation protocol presented by Rusinol and Llados [46] for spotting systems. But in their evaluation protocol, all the spotted areas and the ground-truth areas for a query are considered simultaneously. The area of a spotted region is the area of the polygon that contains the convex hull of the region. The same metrics in [46] are also used for evaluation in the symbol spotting contest GREC-2011. However, in the contest, the area of a spotted or a ground-truth region is simply considered to be the bounding box that contains the region.

Table 3.3 summarizes the results on the electronic drawings database. In the table, the recall and precision values are shown for each query separately. Table 3.4 shows the spotting performance of the method on the GREC-2011 database. In the table, each of the recall and precision values is the average of spotting a number of different queries in all the drawings.

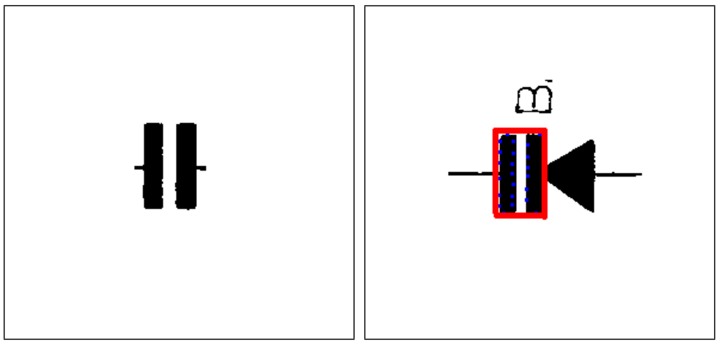
TABLE 3.3: Symbol spotting results on 22 scanned images of real electronic line drawings. The queries are symbols cropped from one of the drawings.

Query symbol	# of instances of the queried symbol	Recall (%)	Precision (%)
Transistor	43	100.0	98.0
Capacitor	80	94.0	94.0
Average	61.5	97	96

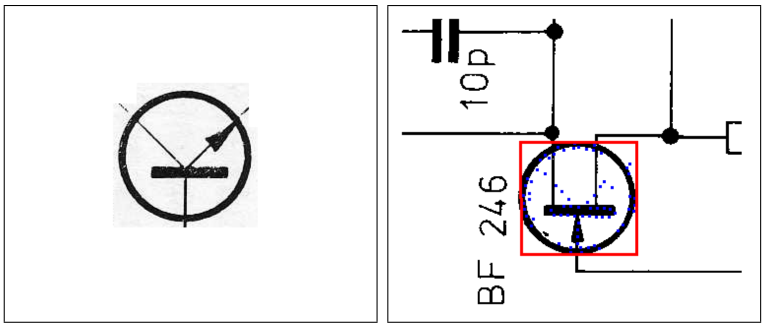
By examining the cases where the method found false matches, the falsely matched symbols were actually very similar to the queried element, examples of false matches are shown in Figure 3.9.

Table 3.4: Symbol spotting results on the GREC-2011 training images of architectural and electric line drawings. The recall and precision values are the averages for 14 queries in 20 electrical drawings, and for 12 queries in 20 architectural drawings.

database	Noise type	# of instances of the queried symbols	Average Recall (%)	Average Precision (%)
Architectural	Random	366	98.1	98.9
Electrical	Random	223	98.7	94.1



(a) Left image: query symbol (capacitor). Right image: spotted instance of the capacitor query symbol.



(b) Left image: query symbol (transistor). Right image: spotted instance of the transistor query symbol.

Figure 3.9: Examples of false matches. In the images of the right column, the matches are surrounded by red bounding boxes. The blue dots inside the boxes show the query features overlaid on the falsely spotted matches.

The running time for matching a pair of images – whether isolated symbols or line drawings –, ranges from a fraction of a second to few seconds. For isolated recognition, in order to recognize a symbol against a large symbol library, the recognition process can take up to few minutes. The running time depends on many factors such as: number of matching instances in an image (true or false), number of model and image features, the size of the transformation space and the accept-match parameter. The geometric matching-based method can be considered practical, but it is slow to be interactive. As discussed in Section 2.3, there are various ways of speeding up the matching process. One such way relies on using feature weights, and it will be discussed in the next chapter.

3.5 Comparison with Previous Methods for Symbol Recognition

Table 3.5 shows a comparison with previous methods for both recognition and spotting. The comparison is shown for the methods that use the same or very similar databases as the method presented in this chapter. Note that the methods that have higher overall accuracy than our method, have actually used a much smaller set of models and test images, also with simpler noise types and only a subset of the transformations.

Table 3.5: Comparison with previous methods. Note that some methods have used different numbers of library symbols and test images, with different noise types and different transformations.

Isolated Symbol Recognition - only for GREC-2005 database					
Method	Accuracy	Models	database - no. of tests	Noise	Transform
Our method	90.1	25 - 150	GREC-2005 - 6000	Yes (all types)	Yes
Min et al. [25]	83.3	25 - 150	GREC-2005 - 6000	Yes (all types)	No
Zhang et al. [27]	82.8	25 - 150	GREC-2005 - 6000	Yes (all types)	Only rotation
Luqman et al. [23]	94.5	20 - 100	GREC-2005 - 40	Yes (their own noise)	No
Wong et al. [76]	94.0	25	GREC-2005 - 100	yes (random)	Yes (random)
Symbol Spotting - for GREC-2011 and similar databases					
Method	Recall	Precision	database (architectural)		Noise
Our method	98.1	98.9	12 queries, 20 drawings		Yes
Nguyen et al. [40]	88.0	70.0	6 queries, 15 drawings		No

3.6 Discussion

This chapter has described a geometric matching-based method as a practical solution to the graphical symbol recognition problem. The method is highly accurate in recognizing symbols in context and with interfering strokes (the spotting task), and it also performs competitively on isolated recognition tasks. The method has been evaluated on different databases, the databases include tests for isolated symbol recognition as well as for symbol spotting. The experimental results show a performance that is significantly better than state-of-the-art methods on the same or similar databases.

The geometric matching approach presented in the previous chapter is used to perform recognition. The use of geometric matching as a symbol recognition method has been presented only in our work. As discussed in Chapter 2, geometric matching has many desirable characteristics. Using this matching made the recognition method posses the same characteristics. The experiments have practically shown the following characteristics of the recognition method: ability to perform recognition in context with large amounts of clutter, robustness to preprocessing artifacts, precise localization of the recognized/spotted symbols, working with different feature types, ability to incorporate penalties for non-matched features and finally being practical via the use of efficient variants of the matching algorithm. Other characteristics such as using feature weights to improve recognition/spotting rates, will be presented in the next chapter.

The chapter has also shown that processing that is adaptive to noise types can be an important part of symbol recognition methods. The good qualitative results of the preprocessed images indicate that it might be better to deal with noise – even if partially – prior to the recognition process itself. Such adaptive preprocessing approach is useful when the test images contain extreme variability in image noise such as the images of the GREC-2005, GREC-2011 databases. It is important to point out that the appropriate preprocessing procedure was **not** selected manually for each noise condition, but that choice is made automatically by the system. We expect that this kind of approach to dealing with a wide variety of noise will become increasingly important.

The symbol recognition method presented in this chapter, can work with different feature types such as pixels, line segment features or a mixture of them. This gives the flexibility to use either statistical or structural representations of symbols. Hence, preprocessing modules from other systems can be used within the recognition method, as different preprocessing modules output different features.

The recognition/spotting method will be used as a step for symbol matching within the methods that will be presented in Chapters 6 and 7. The overall recognition/spotting method can also be a basis for future work on applications like automatic analysis of line drawings or symbol retrieval. For example, using a set of known symbols, the method can be used off-line to spot the symbols within line drawings of a database, and then for on-line retrieval, the previously spotted symbols can be quickly indexed and retrieved.

The recognition method presented in this chapter can be improved in various ways. For example, using other types of features that are not necessarily geometric in addition to the geometric primitives. Another possibility is using the relations between line segment features to improve the precision of the method. Another direction of improvement is to group the similar symbols of the symbol library into few groups in a tree-like structure. This enables the method to simultaneously reject many non-similar symbols, and explore only the symbols within one tree branch. Finally, more efficient variants of the matching algorithm can be developed to achieve interactive speed.

Chapter 4

Improving Geometric Matching-based Symbol Spotting using Machine Learning

The geometric matching-based method presented in Chapter 3 finds many false matches when performing recognition/spotting in degraded line drawings, specially if high recall rates are desired. Furthermore, when solving large scale problems, the method has a running time in the order of few minutes, which is very slow to be interactive. This chapter presents **two** novel methods for improving the performance of the geometric matching-based spotting method on line drawings.[1]. Both methods are based on combining machine learning techniques with geometric matching.

The first method combines geometric matching with linear discriminant analysis (LDA) to learn the importance of the features of symbols, and assign weights to these features accordingly. The learned weights are assigned to query features in the geometric matching-based spotting method. When tested on a database of line drawings, the use of weights improves the precision of the spotting from an average of 71% to an average of 98%, while maintaining the recall rates, with a speedup factor of 2.1.
The second method combines geometric matching with an SVM classifier to train the classifier to distinguish between the true and the false matches found by the geometric matching-based spotting method. The trained SVM classifier is used as a post-spotting classification step. When tested on a database of line drawings, the classification step improves the precision of the spotting from an average of 76.6% to an average of 97.2% while maintaining the recall rates.

---

[1]This chapter is based on the author's work in [84, 85].

4.1 Introduction

The geometric matching-based methods discussed in the previous chapter have been based on manually chosen parameters such as error thresholds and quality measures of geometric similarity. Error rates of those methods could potentially be lowered by manually “tuning” those parameters, but that is a slow and laborious process. In this chapter, we will look at automatic ways, based on machine learning, for improving quality-of-match measures for the purpose of lowering recognition error rates.

For tasks such as the automatic interpretation of line drawings, it is very important that the interpretation is as accurate as possible. Spotting methods are usually used to perform such drawing interpretation tasks. Although many spotting methods have been proposed in the literature, they are not yet reliable enough to be used in the mentioned task due to relatively low recall and/or precision values. Such poor performance is due to many difficulties faced in the symbol spotting problem. These include the following (also discussed in Section 1.1): large amounts of clutter, similar local line patterns even if they belong to different symbols, transformations on symbols within line drawings and degraded drawings. These difficulties make spotting methods result in a lot of false positives (low precision), specially when the methods are adjusted to get a high recall value.

Although the geometric matching-based spotting method presented in Chapter 3 overcomes such difficulties to a large extent and outperforms state-of-art spotting methods, it might still have relatively low precision values due to the following two reasons. First, when dealing with degraded line drawings, a lower threshold for accepting matches must be used in order to keep a high recall rate. This comes at the cost of finding extra false matches, which causes the precision rate to decrease. Second, some query symbols have very simple shapes with just few lines, such symbols are very similar to parts of other larger or more complicated symbols. When trying to spot such symbols within a line drawing, many false matches will be found. Additionally, the geometric matching techniques tend to be slow (i.e. non-real time performance) when solving large scale problems, for example, spotting in large noisy drawings, recognition with large symbol alphabets, or spotting a query of a very simple shape in large drawings.

For the purpose of improving the performance of symbol spotting and increasing its reliability, this chapter presents two novel methods based on combining machine learning techniques with the geometric matching techniques presented in Chapters 2 and 3.

The first method is based on learning the importance of the features of the symbols within a symbol library, and assigning weights to the features accordingly. The learning of feature weights is accomplished by combining geometric matching techniques with linear discriminant analysis (LDA) [86, 87] as follows. The symbols in a symbol library are represented as features (such as line segments) and are used as query symbols for spotting tasks. The geometric matching-based spotting method is used to get information on the matching between the line segments of a query symbol and the line segments of the spotted symbols found by the spotting method (both the true and the false matches). The matching information is used to compute feature vectors for the query symbol. The vectors represent how well the segments of a query are matched to the segments of the true and false matches. Then, LDA is trained on these vectors to get the weights of the line segments of different query symbols. When using a symbol with weighted features as a query in a spotting task, the performance of the spotting method increases in terms of both speed and precision while keeping the same recall rate.

The second method is based on training a classifier to distinguish between the true and the false matches found by a spotting method. An SVM classifier [88, 89] is used for that purpose as a post-spotting classification step. The training data is obtained in the same way as in the above mentioned method for learning weights. When using the classification step within a spotting method, the precision of the method improves significantly while keeping the same recall rate.

Both methods use geometric matching to automatically generate the training data for the machine learning techniques of LDA and SVM. The methods then use the machine learning techniques to get information that help improve the performance of spotting methods. For each method, the new “learned” information is incorporated within the geometric matching-based spotting method presented in Chapter 3 to get a new spotting method. The new method is tested on a large database of line drawings. The results show big improvements in terms of the precision while keeping the same high recall value achieved by the original spotting method. Additionally, the first method makes the spotting run twice as fast compared to the original spotting method. The causes of the improved performance will be discussed within the sections of this chapter.

4.2 Prior Work on Using Machine Learning for Symbol Spotting

The ideas of combining geometric matching techniques with machine learning techniques as presented in this chapter are novel, and to the best of my knowledge, they have not been presented in the literature. Combining the two techniques is done for the purpose of improving the performance of the geometric matching in the context of symbol spotting applications. This section reviews the following: the use of machine learning techniques in symbol recognition/spotting problems, and the use of LDA-based feature weighting or SVM-based classification in spotting applications.

In general, machine learning techniques are not directly applicable on spotting problems, because the input is a complete line drawing not isolated symbols. Unlike complete line drawings; isolated symbols can be directly described in vectors for learning. Machine learning techniques have been more prominently used in isolated symbol recognition. The scheme followed there is developing an invariant descriptor of the symbols, and training a classifier to distinguish different symbols. The descriptor can be graph-based such as in [23, 90], geometric-based as in [77] or a combination of different symbol descriptors as in [91]. Different learning tools were used in these works, such as artificial neural networks (ANN) or Bayesian classifiers. A combination of non-Bayesian classifiers is used in [92].

Machine learning techniques were rarely used as an approach or part of an approach to solve spotting problems. Luqman et al. [41] presented a spotting system that involves a Bayesian classifier. In their work, the line drawings are represented as attributed relational graphs, and regions of interest are extracted from the graphs. The regions are converted to fuzzy structural signatures, and the similar signatures are clustered together. A Bayesian classifier is then trained on the signatures from the line drawings, and is used to spot a query symbol. In their work, many processes are carried out before the training (extracting regions, describing them with the fuzzy signatures and clustering them). This indicates much information loss before the training process. The results indicate better performance if the symbols within the drawings are well separated, which is not usually the case.

As for feature weighting, it has been used in the literature to improve the performance of classification and information retrieval algorithms [93]. Reviews of feature weighting methods can be found in [94] and [93]. A lot of research work has been conducted in the area of LDA-based feature weighting. For example, Zafeiriou et al. [95] have a number of works on using discriminant weights in elastic graph matching to improve face verification. Kitahara et al. [96] used LDA for dimensionality reduction of feature space as a step for musical instrument identification. Song et al. [97] proposed a method for feature selection based on LDA analysis. However, the published research includes neither combining LDA-based feature weighting with geometric matching to improve geometric matching techniques, nor the use of such improved geometric matching for symbol spotting applications.

As for SVM-based classification, SVM [88, 89] is a very popular learning/classification tool, and it is used in a wide variety of applications [98]. However, it has not been used in the context of spotting problems.

4.3 Learning Feature Weights

Feature weighting is a process to assign weights to features according to their importance to a particular task [93]. This section describes feature weighting for the task of symbol spotting, and the effect of using the weighted features on the performance of spotting.

A symbol spotting method takes a query and finds matching symbols within a line drawing. The query symbol is represented as a set of features, so are the matches found by the spotting method. In the case of correct matches, all the features of the query symbol will be matched to the features of the found matching symbol, and for false matches, some or all of the features of the query will not be matched. The importance of query features is signified by their presence in the correct matches and their absence in the false matches. Based on that, more weight can be assigned to the discriminant – i.e. important – features and less weight to the less-discriminant features.

Symbol spotting methods that compute the score of a candidate region based on matching the region features to the query features, can use the weights of the query features to get less false matches as follows. If a candidate region is a false match, then its features correspond (match) to the less-discriminant features of the query. Hence, the overall score of the region will be lower than the score of a true match. Furthermore, as some of regions will have low scores, a spotting method will be faster as it examines a smaller number of promising candidate regions in the subsequent steps.

The problem is: which features are more (or less) discriminant, and what are the discriminance values – i.e. weights – that should be assigned to these features. The solution presented in this section for this problem is to ***learn*** these weights. In order to learn such weights, a machine learning technique needs training data that contain information on how well the features of a symbol are matched in both the correct matches and the false matches of that symbol. Getting such training data requires a spotting method that provides such information as output with the matches. The method for learning the feature weights consists of two main steps. First, using geometric matching to automatically generate the training data (Subsection 4.3.1). Second, using LDA for learning the feature weights from the training data (Subsection 4.3.2).

The learned weights are tested in a symbol spotting application. As discussed in Chapter 1, technical line drawings have a library of symbols that are used as queries. The weights of the features of different query symbols are learned. The weighted query features are used within the geometric matching-based spotting method presented in Chapter 3. Subsections 4.3.3 and 4.3.4 discuss the details and the performance of the spotting method improved by weighted query features.

4.3.1 Generating the Training Data via Geometric Matching

In order to learn the feature weights of a symbol for the task of symbol spotting, a machine learning technique needs training data the contain: (1) the spotted matches of the symbol, labeled as "true" or "false" matches, (2) for each match, information on how well the features of the symbol are matched in both the correct matches and the false matches. Getting such training data requires a spotting method that provides such information as output. The geometric matching-based spotting method presented in Chapter 3 is used for that purpose. A set of query symbols and a database of line drawings are given as inputs to the spotting method. The query symbols and the line drawings are preprocessed and defined as sets of **line segment** features.

Recall that in technical drawings from different domains such as architectural floor plans or electronic circuit diagrams, there is a defined set of symbols (*the symbol alphabet*) that is used to draw these line drawings, and naturally the query symbols come from this defined set of symbols. Each of these symbols has a different number of features, and when a symbol is matched to line drawings, it has a different set of true and false matches than the other symbols. The goal is to learn the weights of the features of the query symbol based on its true and false matches. This means that each symbol has to have its own training data. In the following, the process of generating the training data for a query symbol is described.

When a query symbol and a line drawing are given to the spotting method as input, the spotting method finds regions within the line drawing that match the query. For each found matching region, the method outputs[2]:

- **Feature correspondences**: which line segments of the query symbol correspond (i.e. are matched) to which line segments inside the candidate matching region within a line drawing.
- **Transformation information**: the translation, rotation and scale that make the features of the query symbol correspond to the features inside the matching region within a line drawing.

The true positive found regions contain symbols that are **the same** as the query symbol, whereas the false positive found regions may contain either symbols similar to the query but not the same symbol or just random line patterns that happen to be partially similar to the query. As a first step in generating the training data, the spotting method is applied on a query and the training set of line drawings. The output of this step is the matching regions of the query (both true and false matches). Labels are assigned to these regions using ground truth information of the line drawings database.

For generating the training data, the "accept-match" parameter in the spotting method is tuned to get a high recall value of $\sim$100% (all correct matches), hence, the precision

[2]See Chapters 2 and 3 for details on how the geometric matching-based spotting method works.

value will be relatively low (many false matches). The same setting will be also used for testing. As discussed in Chapter 2, the “accept-match” parameter refers to the minimum fraction of query features that must be matched in the candidate match. The “accept-match” parameter is set to 0.5, this means that at least half of the query features have to be matched in order for a candidate match to be accepted by the method. It is desirable to set this parameter to a low value when dealing with noisy documents, to ensure that no correct matches are missed. Figure 4.1 shows the output of spotting a query in a drawing. In the figure, many false matches are found because the “accept-match” parameter is set to 0.5. Some of the found false matches are just few random segments.

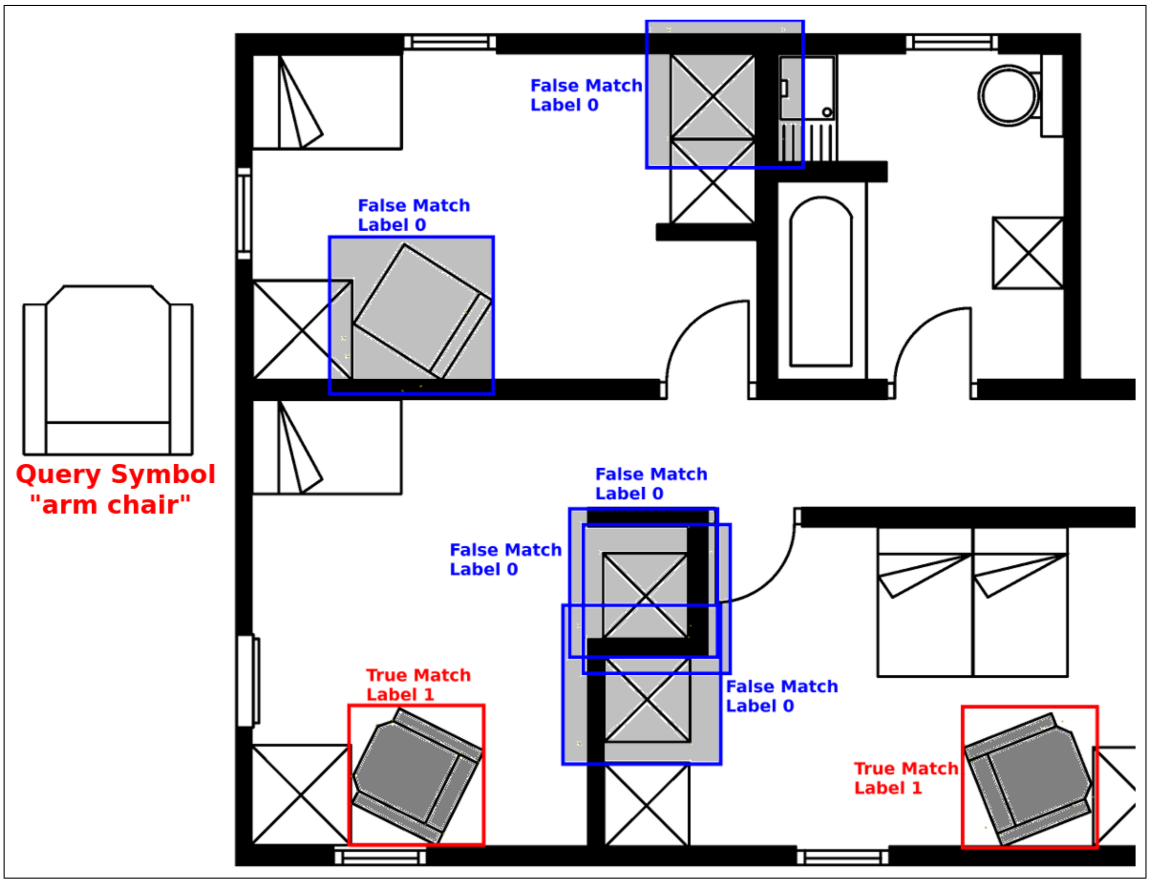


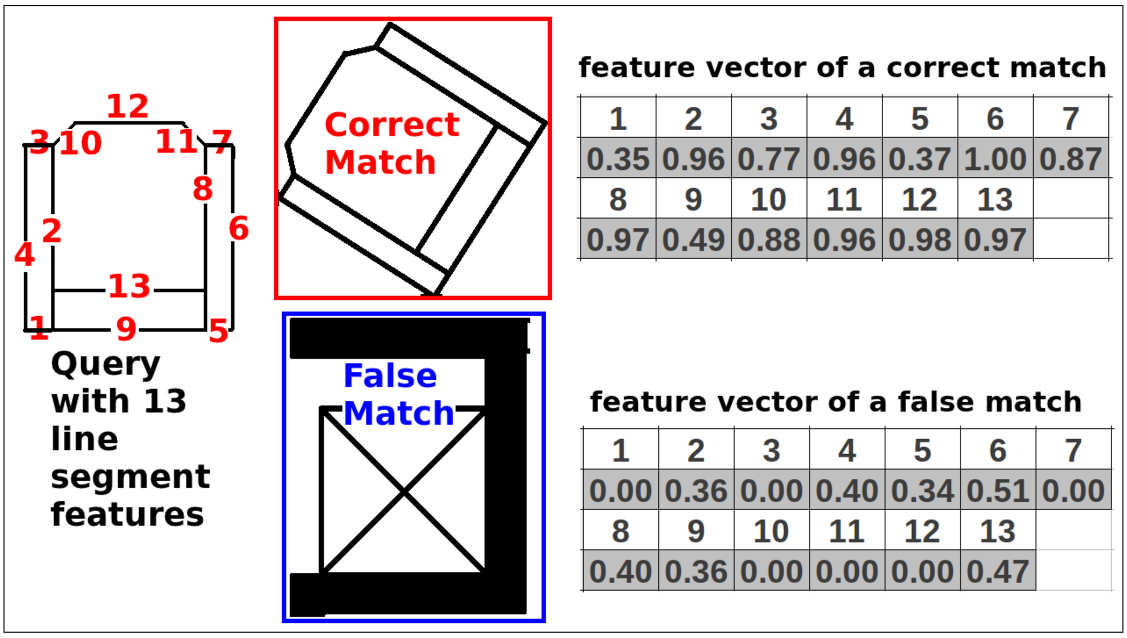
FIGURE 4.1: Training data generation (first step). Spotting a query symbol in a line drawing with a low “accept-match” parameter. The correct matches are labeled “1” (red bounding box), the false matches are labeled “0” (blue bounding box).

The second step in generating the training data is to compute feature vectors from the matches that were found in the first step. The feature vectors represent the regions found by a spotting algorithm, and they are the training samples to be given to the LDA. The following describes the computation of a feature vector from a match.

A vector of the same size as the number of the query features is created. Each entry of that vector is filled with a number (between 0.0 and 1.0) that represents the score of a query feature. So, the training feature vector is a vector of scores, where each score is computed as follows. Assume the M line segments of the query are matched to a group of line segments L in the drawing, with some transformation (that has a scale *scale*). Then the score S_m of each query feature is:

$$\forall m \in M,\ S_m = \frac{\text{length}(l)}{(\text{length}(m) \times scale)} \qquad \text{where } l \in L,\ l \text{ matches } m \tag{4.1}$$

This is illustrated in Figure 4.2. In the figure, two examples of matching regions are shown, one is a correct match (positive sample) and one is a false match (negative sample). For the positive sample, most features have high score values (close to 1.0), but the values vary due to preprocessing and noise. As for the negative sample, only some of the features match, and they match only partially.



feature vector of a correct match

1	2	3	4	5	6	7
0.35	0.96	0.77	0.96	0.37	1.00	0.87
8	9	10	11	12	13	
0.97	0.49	0.88	0.96	0.98	0.97	

feature vector of a false match

1	2	3	4	5	6	7
0.00	0.36	0.00	0.40	0.34	0.51	0.00
8	9	10	11	12	13	
0.40	0.36	0.00	0.00	0.00	0.47	

FIGURE 4.2: A query and two matches with corresponding feature vectors. The segments in the red bounding box are matched (partially or fully), while some of the segments in the blue bounding box are not matched. Each vector has 13 entries corresponding to the 13 query features. The values indicate the scores of the query features in each match.

The training feature vectors are computed from all the matches of a query symbol. Each vector contains values of how well each of the query features is matched, and each vector is labeled as a positive or a negative training sample. Obviously, the positive training samples are correct matches, hence they are the symbols that are the same as the query symbol. A negative training sample can be any symbol other than the query symbol, or just a set of random segments. Such training data have the potential to distinguish between the symbols that are the same as the query, and all the other symbols or line patterns that are not the same as the query. The process of generating training data for a query symbol is repeated for all the symbols in the available set of queries from the symbol library.

4.3.2 Weighting the Features via LDA

Once the training data is ready, linear discriminant analysis (LDA) can be used to learn weights for symbols features. LDA fits Gaussian distributions to the samples in each class, and finds the feature weights that maximize the between-class covariance while minimizing the within-class covariance.

The goal is to discriminate a symbol versus all other symbols using the training data from the matches of that symbol, and then repeat this process **separately** for each symbol. This is a two-class problem.

The training data consist of n samples $\{(x_i, y_i)_{i=1}^{n}\}$, where $x_i \in \mathbb{R}^d$ is a d-dimensional feature vector, and $y_i \in \{0, 1\}$ is the corresponding class label. LDA seeks a linear transformation $W \in \mathbb{R}^{d\text{x}1}$, to map the original x_i in the d-dimensional space to a 1-dimensional space such that: $W^t x_i \in \mathbb{R}$ where $d > 1$. The optimal W can be found by maximizing the Fisher criterion: $J(W) = |W^t S_b W|/|W^t S_w W|$, where S_b and S_w are the between-class and within-class scatter matrices of the training samples respectively, their definition can be found in [87]. Finding the W matrix that maximizes $J(.)$ has a well known solution [87]: The columns of an optimal W are the generalized eigenvectors that correspond to the largest eigenvalues in $S_b W = \lambda S_w W$.

The learning process is carried out the same as regular LDA classifier learning. LDA is used to compute $W = [w_1, w_2, ..., w_d] \in \mathbb{R}^{d\text{x}1}$ where d is the number of features of a query symbol. Note that the LDA-singularity or undersampling problem will not occur, since the feature vectors are not high dimensional. The symbols represented as line segments have between 5 to 35 features, and the feature vectors given to LDA have the same sizes. After the learning is done, the resulting LDA classifier is not the desired output, but the matrix W that is computed as a result of the learning process. Usually, an LDA classifier multiplies the test vector by W and outputs the result as the class label. But the goal here is to get the weights matrix W itself. The values of W are normalized to the range $[0.5, 1.5]$, and the normalized values are used to weight the symbol's features. Even though some features are less important than others, the less important features should not, however, be totally discarded. Query symbols in general have few line segments as features, and removing some of them will affect the spotting negatively.

The learned feature weights can be interpreted as follows. The features whose score values are very different in different classes will be given high weights by LDA. The values of the feature scores are different because they are matched in the positive class and not matched in the negative class. Whereas the features whose score values are similar in the different classes, will be given low weights.

A simplified explanation of what the LDA classifier has actually learned is described in the following. Assume that the features are point features along the contours of a query symbol (instead of line segments). Then the distribution of the feature scores is a binomial distribution that represents the presence or absence of the feature in a candidate match. The score of a point feature in a candidate match would be 0 if it is not-matched, or 1 if it is matched. An LDA classifier will learn the weights W that correspond to the ability of the features to distinguish a match from a non-match.
For computational convenience, and to get a smaller number of features, the queries are represented as line segments. A line segment approximates a number of point features. The score of a line segment feature is the sum of the scores of the matching point features on the segment, which corresponds to the length of the line segment. By incorporating the lengths of the query features within the scores as in Eq. 4.1, the learned weights will also incorporate the length of the line segment features with their ability to distinguish a match from a non-match. Assuming that the point features along the segment are independent, this approximation works well for our case.

4.3.3 Application to Symbol Spotting

This subsection describes how the feature weights can be incorporated in a symbol spotting method. The geometric-based spotting method presented in Chapters 2 and 3 will be used.

Recall from Section 2.2 that a single query feature m matches an image feature i if the transformed query feature maps to the image feature within a distance error threshold ϵ. Also recall that Subsection 2.2.4 describes the incorporation of weights that are associated with features within the geometric matching algorithm. In this subsection, the learned feature weights will be incorporated the same way.

Without weights, the quality of mapping two sets of features using a transformation T is calculated as:

$$Q(T) = \sum_{m \in M} \sum_{i \in I} q(T(m), i) \tag{4.2}$$

where $q(T(m), i)$ is a quality value of matching a single transformed query feature to a single image feature. These individual features are defined as line segments. The reader is referred to Chapter 2 for details.

In order to incorporate the weights, Eq. (4.2) is modified as follows:

$$Q(T) = \sum_{m \in M} w_m \sum_{i \in I} q(T(m), i) \tag{4.3}$$

where w_m is the weight associated with the symbol feature m, and is a part of the weights vector $W = [w_1, w_2, ..., w_d]$. W has the same size as the number of the symbol features M.

The spotting method carries out a branch-and-bound search for the transformation T_{max} that maps a maximal subset of the query features to maximal subset(s) of image features. During the search, the quality of matching for different transformations is computed according to Eq. (4.3). Since the weights are higher for the features that exist in the correct matches but do not exist in the false matches, the correct matches will have higher overall qualities. This will make the correct matches go up in the priority queue maintained by the branch-and-bound search algorithm. Hence, the correct matches will be found faster. Furthermore, the overall quality for the false matches becomes lower, because their matching features have lower weights. This causes the rejection of the false matches, which improves the precision. Figure 4.4 shows spotting a query in the same line drawing once using the weights and once without using them, note that the same parameter setting in the spotting algorithm is used in both cases.

4.3.4 Performance Evaluation

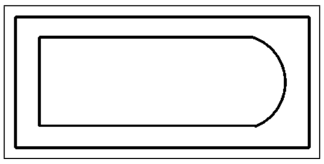
This subsection presents the evaluation of the spotting method with and without weights. The database used for evaluation has 400 images of architectural floor plans. The images are a subset of the graphics recognition database called "Sysed". The database is publicly available[1], and it contains different types of technical line drawings.

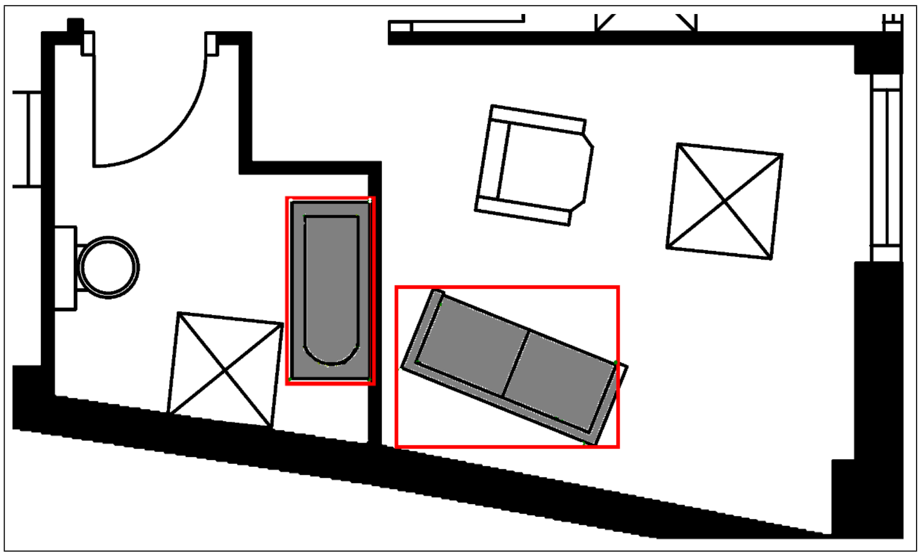
The generation of the database is described in [43] and [44]. The images of the database are synthesis line drawings that imitate real complete floor plans with sizes between 2M to 7M pixels. Different subsets of this database have been used for evaluation of symbol spotting systems in [40] and [41]. The GREC-2011 symbol spotting contest[2] [17] used a set of images very similar to the images of the Sysed database, as the images of the contest were generated using the same system of [43, 44].

Different subsets of the Sysed database are used for evaluating the methods presented within this chapter and the next chapters of this thesis.

First, the training data is generated as described in Subsection 4.3.1. The training data is generated from 200 images of the database for different query symbols. Then, an LDA

[1]http://mathieu.delalandre.free.fr/projects/sesyd/index.html
[2]http://iapr-tc10.univ-lr.fr/index.php/symbolrecognitionhome


(a) A query symbol (tub).


(b) Spotting output without using the weights of the query features. Two matches are found, one (on the left) is a true match, and the other (on the right) is a false match.

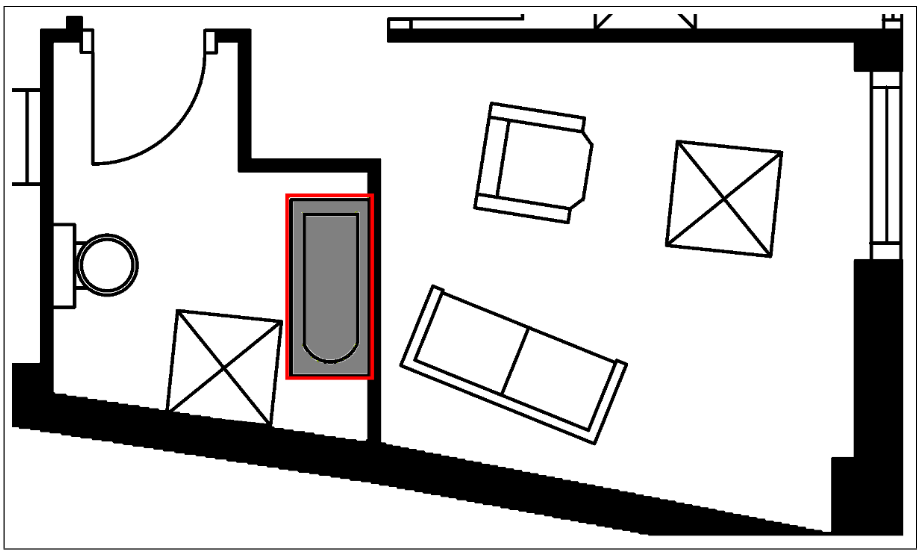

(c) Spotting output using the weights of the query features. One true match is found.

FIGURE 4.3: Spotting results on a line drawing, once with and once without using the weighted query features. Note that the true matches found in both cases are the only true matches that exist in that line drawing.

classifier is trained on the data of each query symbol to get the weights vector as described in Subsection 4.3.2. The weights computed for a query symbol are associated with its features. Finally, the weighted query features are used to spot instances of that query symbol in the test set of the drawings. The test set has 200 images (the database has 400 images, 200 are used for training and 200 are used for testing).

Table 4.1: Results of the spotting method (with/without weights of query features) in 200 test images of architectural drawings. All the results are at a recall value of 100%.

Query	Symbol instances	Precision	
		weights	no weights
arm chair	253	100%	77.7%
door	700	92.8%	82.1%
sofa	228	100%	49.6%
table	167	100%	100%
tub	200	100%	50%

In order to evaluate the effect of weights on the performance; the spotting method is tested once with the weighted query features, and once without the weighted features. The spotting method without using the weights is just the same spotting method presented in Chapter 3. The spotting method that uses the weights differs only in that the quality of match function is computed according to Eq. (4.3) instead of Eq. (4.2). The same parameter settings is used in both methods. The results are shown in Table 5.1.

The results show a significant improvement in the precision values while not affecting the recall values. All the results are at a recall value of 100%, which is normal performance for the spotting method with clean images and with the "accept-match" parameter set to 0.5.

As for the running time on a 2.80GHz CPU, the spotting method takes 17.57 seconds per image on average without using weights, and takes 8.23 seconds with weights. This means a speedup factor of 2.1 using the weighted features of the queries.

4.4 Classification of Matches as a Post-spotting Step

As discussed in the previous section, a symbol spotting method takes a query symbol and a line drawing as input, and finds regions within the drawing that contain symbols that match the query symbol. The found matching regions, however, may be true or false matches. One way to eliminate false matches is by adding a step after the spotting step to distinguish the true matches from the false matches. In order to be efficient, such step has to work directly on the representations of the matching regions that are found by the spotting step.

The method presented in this section uses a machine learning technique for developing such step. To distinguish a true match from a false one, a classification step is used after the spotting. The classifier is trained on true and false matches of a query symbol. The following subsections describe the development and testing of the classification step.

4.4.1 Generating the Training Data via Geometric Matching

In order to develop the classification step, the automatic generation of training data is needed. The training data should be vectors that represent the matching regions, and the vectors should encode information about how well the query is matched to a matching region. Such training data are generated in Subsection 4.3.1.

So, for the classification step presented here, the training data is generated exactly the same way as described in Subsection 4.3.1. Namely, training data for each query symbol from the symbol library is generated.

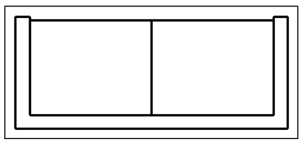
4.4.2 Classifying the Matches via SVM

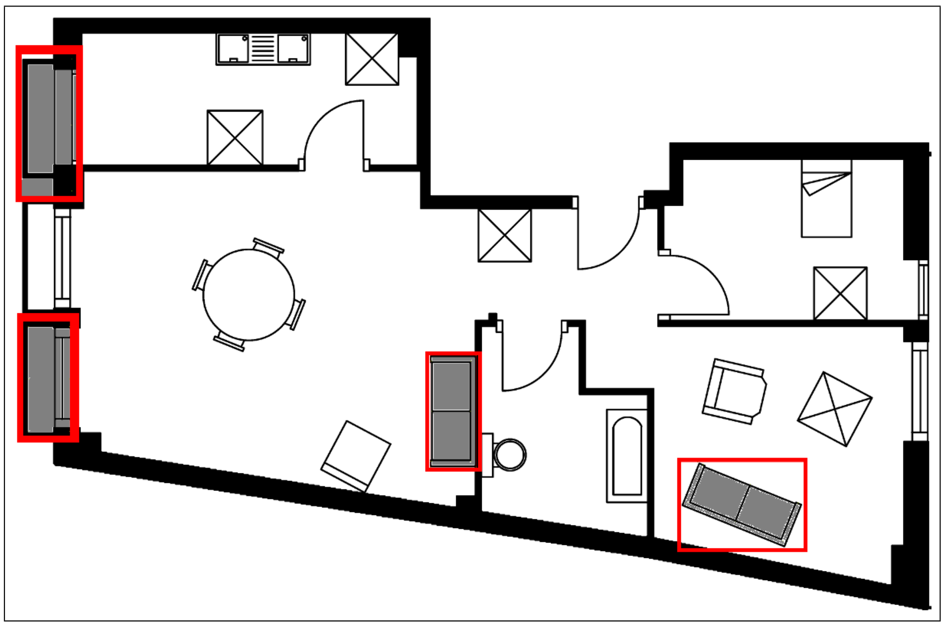
Once the training data are available, a classifier is trained on them, and then is added as a step after the spotting step. The SVM classifier is chosen as it is a fundamental machine learning tool with a strong theoretical foundation [88, 89], and it has shown good empirical performance in a wide variety of pattern recognition and data mining applications.

The classification problem here is a two-class problem. Similar to what has been discussed in Subsection 4.3.2, the goal here is to discriminate a symbol versus all other symbols using the training data from the matches of that symbol, and then repeat this process **separately** for each symbol.

A number of two-class SVM classifiers with a linear kernel are trained, and each trained model is associated with a symbol. When this symbol is used as a query for spotting, the corresponding SVM-trained model is called for classifying the matches after the spotting step is carried out.

This post-spotting step does not increase the time complexity of the spotting method. The training is done off-line, and for the spotting process, the complexity is dominated by the search for the matches during the spotting step. The classification step adds the following operations. For each of the found matches (typically below ten in one line drawing), the feature vector is computed, and classified. Feature vectors are not high dimensional, as mentioned in Subsection 4.3.2, the symbols represented as line segments have between 5 to 35 features, and the feature vectors given to SVM have the same sizes. The feature vectors are computed from data that are already available as output by the spotting method. Figure 4.4 shows the output of the spotting algorithm, once with SVM classification step and once without it.


(a) A query symbol (sofa).


(b) Spotting output without the SVM classification step. Four matches are found, two of them are true matches, and two are false. The false ones are the ones on the left side of the drawing.

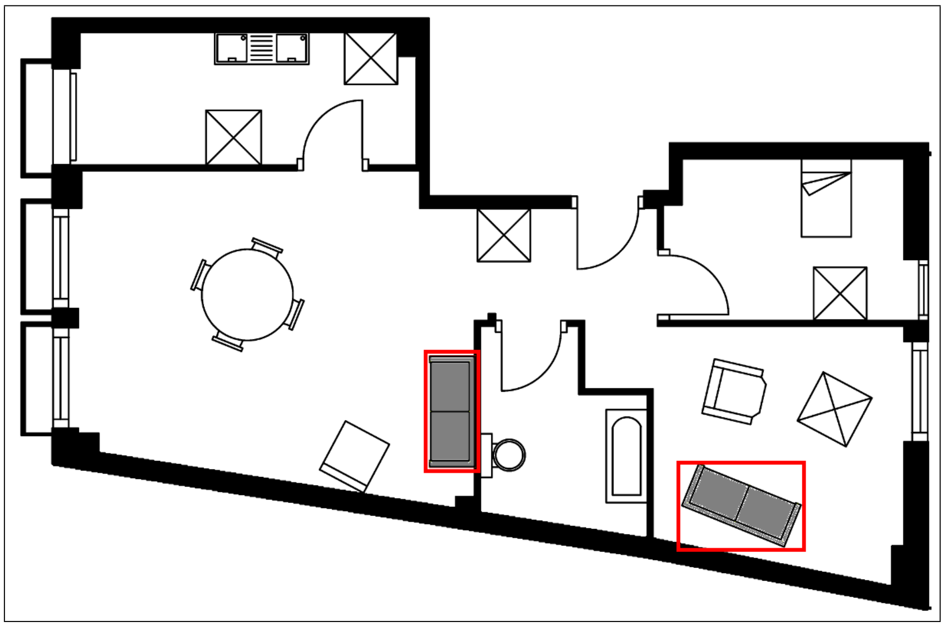

(c) Spotting output using the SVM classification step. Two true matches are found.

FIGURE 4.4: Spotting results on a line drawing, once with and once without using the SVM classification step as a post-spotting step. Note that the true matches found in both cases are the only true matches that exist in that line drawing.

4.4.3 Application to Symbol Spotting with Performance Evaluation

This section discusses the addition of the SVM-classification step to the geometric-based spotting method presented in Chapter 3. The section also presents the evaluation of the spotting method with and without the SVM classification step. The database used for evaluation has 500 images of architectural line drawings. The images are a subset of the Sysed database described in Subsection 4.3.4[3].

First, the training data are generated as described in Section 4.4.1 (alternatively in Section 4.3.1). The training data are generated from 250 images of the database. Then, the SVM classifier is trained on the data as described in Sec. 4.4.2. The used SVM is from the python MLPY package that implements libSvm [99], with a "linear" kernel type. Finally, the spotting method enhanced by the classification step is tested by spotting different queries in the drawings of the test set of the database. The test set is the other 250 image that were not used for training.

TABLE 4.2: Results of the spotting method (with/without SVM classification) on 250 test images of architectural drawings. The classification step did not affect the recall values negatively. All the results are at a recall value of 100%.

Query	Symbol instances	Precision	
		with SVM	without SVM
door	850	94%	86%
sofa	278	100%	50%
table	193	100%	100%
tub	250	100%	55%
arm chair	253	100%	77.7%

For evaluating the effect of adding the classification step on the performance of the spotting method, the method is tested with and without SVM classification. The same parameter settings are used in both cases. The results are shown in Table 5.1. The results show a significant improvement in the precision values while not affecting the recall values. All the results are at a recall value of 100%.

[3]Just for clarification: the 500 images used in this evaluation include the 400 ones that were used for evaluation in the previous section.

4.5 Discussion

This chapter has presented two novel methods for improving the performance of the geometric matching-based spotting method presented in Chapter 3. The two methods are based on combining machine learning techniques with geometric matching in order to improve the performance of the geometric matching in the context of symbol spotting applications.

Both methods exploit the useful information provided by the geometric matching for the automatic generation of training data. When used for spotting instances of a query, the geometric matching outputs both the correspondence and the transformation information between the query features and the features of the spotted regions. This information is used to generate vectors that represent how well the query features are matched in the spotted regions. The spotted regions may be true or false matches of a query, hence, the vectors generated for these regions can be used to discriminate between true and false matches.

The first method uses the training data to learn the weights of the features of different symbols. The weights represent the importance of the features of a query symbol for the task of spotting that query in line drawings. LDA is used for learning these weights. The weights are incorporated within the geometric matching-based spotting method to produce a new and improved spotting method.

The second method uses the training data to train an SVM classifier to distinguish false matches from true matches. The SVM classification step is added to the geometric matching-based spotting method as a post-spotting step to produce a new and improved spotting method.

The two improved spotting methods are tested in spotting queries in a large database of line drawings. For both methods, the results show significant improvements on the precision values over the original spotting method. More importantly, the high recall values of the original spotting method remained the same. In other words, the added machine learning techniques did not affect the recall negatively. Additionally, the spotting method enhanced by the feature weights is twice as fast with respect to the original spotting method. Obviously, both methods are applicable to isolated symbol recognition tasks.

The SVM classification-based method is related to the feature penalization method presented in Section 2.4. The penalization method uses experimentally chosen thresholds to judge whether a feature in a spotted region is matched or not matched to the query features, and then uses experimentally chosen penalties to reduce the overall matching score for the spotted region that contains non-matched features. It can be assumed that the classification method learns such thresholds and penalties, and judges whether a spotted region is a match or a non-match.

The improvements of the geometric matching-based spotting method by machine learning techniques show the flexibility of the geometric matching approach discussed in Chapter 2, and provide further evidence that geometric matching is a very promising approach for solving symbol recognition/spotting problems.

Part II

Symbol Retrieval

Chapter 5

Finding Regions of Interest via Feature Grouping

Finding better regions of interest (ROIs) can greatly improve the performance of symbol spotting and symbol retrieval systems. Methods that find ROIs should satisfy a set of criteria such as repeatability, accuracy, distinctivity and capturing all objects of interest. The implicit ROI-finding steps in existing symbol spotting and retrieval systems do not satisfy the required criteria of an ROI-detector, hence, the found ROIs are neither consistent nor accurate. This chapter presents an ROI-finding method – called the grouping method – that satisfies all these criteria[1]. The method is based on statistical feature grouping. The grouping method is evaluated on a database of line drawings and is found to be highly accurate in capturing the contents of the line drawings. This makes the grouping method effective as a content analysis step in a symbol retrieval system. Additionally, the method is used to improve the spotting method presented in Chapter 3. When evaluated on a database of line drawings, the improved spotting method is found to be faster and more accurate than the original spotting method.

5.1 Introduction

Regions of interest (ROIs) in gray scale images represent parts of objects up to complete objects. Similarly, ROIs in line drawings represent parts of symbols up to complete symbols, possibly with little additional noise (lines from the background or neighboring symbols). Identifying ROIs is an important step of line drawings analysis. Additionally, finding better ROIs in line drawings, can greatly improve the performance of symbol spotting and retrieval

[1]Parts of this chapter are based on the author's work in [100].

systems. This chapter presents an ROI-finding method based on feature grouping. The method satisfies the criteria required from an ROI-finding method, such as repeatability, accuracy, distinctivity and capturing all objects of interest. Satisfying these criteria means that the method find "good" ROIs (the next section explains the meaning of "good" ROIs in detail). The chapter also shows how the ROI-finding method is used to improve the performance of symbol spotting and retrieval systems.

In the following, the motivations behind developing methods for finding ROIs in line drawings are discussed. Using ROIs to represent line drawings is inspired by the popular and successful use of ROIs to represent gray scale natural images. Gray scale images – as well as line drawings – usually contain more than one object – possibly touching or overlapping –, along with background contextual noise. Hence, finding a global description of an image would be very challenging. On the other hand, finding objects of interest by image segmentation is a great challenge in computer vision [101]. That's why the trend in computer vision over the last decade has been to describe the statistics and content of images in terms of local image regions. Local regions-based representations have the advantages that they are robust to transformations, occlusion, clutter, object and image variation, while retaining information about image content [102]. This suggests that local regions-based representations will remain popular in computer vision. They have been – and still are – used in many applications, such as object detection and recognition [103], [104], scene classification [105], stereo matching [106] and content-based multimedia retrieval [107], [108], [109].

For line drawings, finding "good" ROIs is important for symbol spotting and retrieval systems. As discussed in Subsection 1.2.2, the most popular symbol spotting approach is the ROI-based approach. In this approach, it is important to find ROIs in order to match them with the query symbol, the query symbol cannot be matched to the complete line drawing image. However, existing spotting systems do not find ROIs explicitly, and their implicit ROI-finding steps do not satisfy the required criteria of an ROI-detector, hence, the found ROIs are neither consistent nor accurate. The ROI finding step is arguably the most difficult step in a spotting method. Moreover, it greatly affects the spotting accuracy. If the ROIs were correctly and consistently found, the later steps of describing and matching them would be easier and would give better results. In the next section we will discuss what it means to "correctly and consistently" find ROIs.

Finding ROIs is also important for symbol retrieval. Assume a very fast and accurate spotting method is available, one still cannot afford to use it for retrieving a specific query symbol from a large database of drawings. Because this would mean that the spotting will be performed on each line drawing one by one sequentially. So, alternative approaches must be found for retrieval. The retrieval methods in the literature – whether text, image or video retrieval – follow the same high level approach. This approach has two stages. First, the off-line stage where a database is processed and analyzed to be converted to a compact indexable representation. The off-line stage consists of two main steps, finding ROIs and clustering. Second, the on-line stage, where a query is processed the same way an image of the database is processed, and then a matching or indexing step is carried out to retrieve the database images that are similar to the query. The matching/ indexing step is performed on the compact representation of the database not on the original database. Hence, without the ROIs-finding step, the further steps of obtaining the compact and indexable representation of the database are not possible.

One might argue that the spotting method introduced in Chapters 2 and 3 can be used as an off-line step to find the symbols in a database of line drawing as an alternative to finding ROIs. In principle, this is applicable, however it can only work for a set of known models from a symbol library. A query can be an unknown symbol, and the line drawings may contain unknown symbols in addition to the previously defined symbols of a symbol library. This emphasizes that retrieval methods should not be designed for a specific set of model symbols. Using an ROI-finding method as a first step in a retrieval system makes the system able to process unknown symbols.

Based on the above discussion, one can see the importance of developing methods for finding better ROIs in line drawings. To contribute to finding better ROIs, a method for finding ROIs is presented in this chapter. The method – called the grouping method – extracts parts of symbols from line drawings, thus converting a line drawing to a set of isolated parts of symbols. The region that contains an extracted part represents an ROI, and the features of which a symbol part is composed, are the features of the ROI.

The grouping method applies statistical grouping on the features of a line drawing, followed by making combinations of these initial groups to create the final parts of symbols. Statistical grouping means that grouping the features is based on statistically non-accidental properties of combinations of these features. The features are line segments or as they are called "vectorial primitives". The ROIs that result from this grouping method are seen to correspond to meaningful parts of the symbols up to complete symbols.

The grouping method is evaluated as an ROI-detector, and is found to satisfy the required properties of a region detector very well. The method is also highly accurate in capturing the contents of line drawings. As an ROI-detector, the presented method is used as a first step of the off-line stage of a retrieval system. Additionally, it is shown how the grouping method can be used to improve the performance of spotting methods.

5.2 The Properties of a Good ROI-Detector

Whether finding the ROIs is used for spotting or for retrieval, an evaluation is needed to judge how well a particular method of finding local ROIs works. Such evaluation also helps in determining how much percent of the error rates of a retrieval system is due to an ROI-finding method. In order to carry out such an evaluation, one needs to know what properties an ROI-finding method must/should have, and how the method can be evaluated according to these properties. In later sections of this chapter, the grouping method will be evaluated as an ROI-detector to assess the quality of the ROIs found by the method.

In the following, the properties of a good ROI-detector are discussed. Tuytelaars et al. [102] have argued that, local ROIs should correspond to semantically meaningful object parts, and that is in practice, however, not feasible, as it requires high-level interpretation of the scene content. Tuytelaars et al. [102] provided a good survey for the required and/or desired characteristics of an ROI-detector, and Dickscheid et al. [110] have also discussed similar desired properties of ROI detectors and descriptors. Mikolajczyk et al. [111] have characterized and compared the performance of different region detectors for gray scale images based on the required properties of an ROI-detector.

The properties discussed in [111] and [102] are widely accepted and used in the community of content-based image retrieval. These properties include: repeatability, accuracy, distinctivity, locality and efficiency among others. Tuytelaars et al. [102] argued that the repeatability is the most important property as it relates to invariance and robustness of a region detector. Mikolajczyk et al. [111] have used the repeatability property as the main evaluation criterion.

In this work, the following properties are used characterize the performance of the ROI-finding method presented in this chapter. The first three properties are from [111, 102] – adapted to technical line drawings –, and the last one is an additional proposed property that – I argue – is more desirable for line drawings.

- Repeatability
- Accuracy
- Distinctivity
- Covering all symbols “meaningfully”

Repeatability
Repeatability is defined as always detecting the regions in different images that correspond to the same pre-image region. This property relates to transformation invariance. Suppose there is a region that corresponds to a specific symbol in a line drawing, and this symbol appears in other different line drawings but in different locations, rotations and scales, then repeatability means always detecting the region that corresponds to this symbol in the different drawings.

Accuracy
Accuracy is the degree of overlap between the found region and the pre-image region. Assume that, according to some definition, a region that corresponds to some symbol, is a reference region, and this symbol appears in different line drawings. Upon the detection of a region that corresponds to this symbol, the accuracy is measured by the degree of overlap between the reference region and the detected region after considering the transformation.

Distinctivity
Distinctivity means that the regions that correspond to different symbols or different parts of symbols should be different than each other. This property is actually important for the later steps of region description and matching. That means if a region detector has the first two properties, the detected regions still can be similar even if the regions correspond to different symbols. This happens for example when different symbols have parts that are similar but with a different spatial layout. This problem can be solved at the later stages unless it is due to a problem in the region detector. Usually the distinctivity is measured by matching accuracy i.e. at the stage after region detection.

Covering all symbols "meaningfully"
Covering all symbols is more essential to technical line drawings than to gray scale images. All the symbols that appear in a line drawing are important for the applications of spotting, interpretation of line drawings or retrieval. Not detecting the region(s) of a symbol or a large part of it, means missing that symbol in the spotting/retrieval step. While in natural images, an image could contain many objects in the background that are not of interest to the user. It would be nice however if a natural image could be fully analyzed. The second part of the property, "meaningfully" covering the symbols, is also more relevant when dealing with textureless shapes such as the ones found in line drawings. "Meaningfully" means that the detected regions should not correspond to a random set of line segments that come from two different symbols or from the background of the connection lines. The best case is when a detected region correspond to a complete symbol with no additional lines from other symbols or the background.

Without satisfying these properties, the later steps of describing, clustering, matching or indexing will be harder to develop, or will yield bad results, as their outcome highly depends on their input of ROIs.

5.3 Finding ROIs in Technical Line Drawings: State-of-the-art

Many different region detectors have been developed for gray scale natural images [112], [113], [114] ,[115], [116]. However, little work has been done on region detection in technical line drawings. This is probably due to the textureless nature of symbols and line drawings.

In textured gray scale images, the region detectors find patches around key points as local regions, or find regions from a grid or sliding window. As discussed in [117] and [118], these methods do not work well for textureless objects. In the case of line drawings, the local line patterns in a drawing are very similar, which makes the patches that contain these local patterns not discriminative enough. Moreoever, the detected scale of the local regions in these methods is usually not correct due to the lack of texture and due to the unified stroke width used to draw symbols of different sizes.

All the ROI-finding methods to be discussed in the rest of this section, were actually introduced within ROI-based state-of-the-art symbol spotting systems. Only the ROI-finding steps of these spotting systems are discussed here. The other steps of the spotting systems such as describing, matching or indexing the detected ROIs are not relevant for this chapter, and are discussed in Sections 1.2.2 and 7.2.

The spotting methods proposed in [32], [33] and [10] used closed regions in architectural line drawings to find ROIs. Closed regions are rotation and scale invariant, but they have the problem of not being distinctive, most of the detected regions will be just circles and rectangles, which are the shapes of the two most common parts of which the technical symbols are composed. Rusinol et al. [33] have noticed that this problem causes slower matching and higher false positive rate.

In the spotting system of Nguyen et al. [40], difference of Gaussians (DOG) [103] is used to detect interest points, where the sizes of the ROIs around those points are determined by the scale detected by DoG for the points. This is the same as the methods used to detect ROIs in gray scale images. As discussed above, such detectors do not work well for textureless line drawings, and yield high false positive rate in the matching step.

Sliding windows are also used to detect ROIs in line drawings. Kong et al. [42] used overlapping sliding windows to find ROIs, while Dosch et al. [14] used non-overlapping sliding windows (similar to placing a fixed grid over the line drawing). Sliding windows have the following problems. A lot of the detected regions contain a random set of lines that come from different parts of symbols, the separation of symbols or symbol parts need to be tuned [42]. Furthermore, when using small windows, it is not clear what to do with over-segmented detected ROIs. Sliding windows are not scale and rotation invariant. Using sliding windows usually implies that the later spotting/retrieval steps are complicated and yield low recall and precision rates, which is the case in [42] and [14].

Locteau et al. [38] presented the idea of using graph representations of line drawings, where the graph-cliques correspond to ROIs. They claimed that those cliques also correspond to a perceptual grouping of vectorial primitives. Both opened and perceptually closed curves are identified from aggregation of cliques. Qureshi et al. [35] also used a graph-based representation in order to find ROIs (the same method was later used by Luqman et al. [41]). First a line drawing is vectorized into quadrilateral primitives and represented as an attributed relational graph (ARG), where the nodes of the graph are the quadrilateral primitives and the edges represent the relationships between these primitives. Parts of the ARG are detected as ROIs by checking the attributes of the nodes and edges such as lengths of segments, angles between segments and node degrees.

As for the literature on the use of feature grouping in line drawings, the work by Dosch et al. [14] discussed some grouping mechanisms for line drawings. It is based on studying the relations between pairs of line segments, those relations include collinearity, parallelism and intersections. The relations from local image regions are clustered in buckets. For matching, the signatures of these buckets are compared to the signatures of the symbols models. The feature grouping in their work is used to describe ROIs not to detect them. As mentioned earlier in this section, they used non-overlapping sliding windows as potential ROIs.

The ROI-finding method introduced in this chapter overcomes all the problems faced by state-of-the-art methods. It is based on feature grouping. The features are vectorial primitives (line segments) of the line drawings. Sets of line segments are grouped together to form a symbol part or a symbol. The area that contain a group of segments is detected as a region of interest. As will be discussed in the next few sections, the ROI-finding method has the advantage of satisfying the properties of a good regions detector very well. The ROIs are consistently and accurately found by the grouping method. Moreover, the regions are precisely located as they are detected based on the line segment features not on some overall score for a patch or a region.

5.4 The Grouping Method

The grouping method presented in this chapter relies on *non-accidental properties* for grouping the features (line segments) of line drawings. Non-accidental properties of line segments include collinearity, parallelism, co-termination of segments end points, proximity and/or convexity. These properties are the so called "Gestalt laws of perceptual organization" [119], and they originate from psychology research long before their use in machine vision. They show how components or parts can be perceived as overall patterns. These laws play an important role in determining object grouping and region boundaries [120]. Lowe [121] provided a good summary of these concepts.

The research works in [119], [122] and [121] have shown with statistical analysis that, it is very unlikely that a random group of segments are placed such that they obey the perceptual organization rules. This means when a group of line segments obey these rules, then the probability is high that the segments belong to the same object. That is why these properties are called “non-accidental” or “statistically non-accidental” properties. Feature grouping based on non-accidental properties is a computer vision technique that has been used to improve the performance of object recognition [122], [123].

From among the different non-accidental properties, “convexity” is used in the grouping method to group sets of line segment features in a line drawing. A set of line segments is grouped together if the line segments form a convex group. Convexity is an effective grouping property because, even though most objects are not convex, they can be decomposed into convex parts. Moreover, convex groups are rotation and scale invariant.

The grouping method finds convex groups of segments as the main feature grouping step. For that purpose, Jacobs [124] algorithm is used. Jacobs [124] developed an algorithm for robustly finding salient convex groups from line segments. In his work in [122] and [124], he has shown that, based on the statistical analysis, it is unlikely that a random group of line segments will form a convex group.

The input to the grouping method is complete line drawings. It starts by applying simple preprocessing steps on the input images. First, a morphological edge detection step, that produces an image of thin contour lines. Second, a vectorization step is applied simply by sampling line segments along the contours. Those segments are the geometric primitives used as input to the grouping method. After the preprocessing, the method uses Jacobs’ algorithm [124] to find convex groups. In the following, Jacobs’ grouping algorithm is described. After that, the complete grouping method is presented.

Jacobs’ algorithm finds sequences S of line segments that are acceptable as convex groups. The following is a recursive definition of a sequence of segments that is acceptable as a convex group. The definition and the terminology used in it, are adapted from [124].

- Any sequence of a single line segment is an acceptable sequence S_i.
- S_{i+1} is acceptable only if $l_{i+1} \notin S_i$. (no line segment may appear more than once in a group).
- S_{i+1} is acceptable only if the oriented line segments in it are mutually convex. (If the sum of the absolute values of the angles turned is 2π when one travels from the first endpoint of the first line to each additional endpoint in turn, returning finally to the first endpoint).
- S_{i+1} is acceptable only if: $L_{1,i}/(L_{1,i} + G_{1,i}) > k$ where $L_{1,i}$ is the sum of segments lengths in the sequence, $G_{1,i}$ is the sum of gaps between the line segments of the sequence, and k is some fixed threshold.

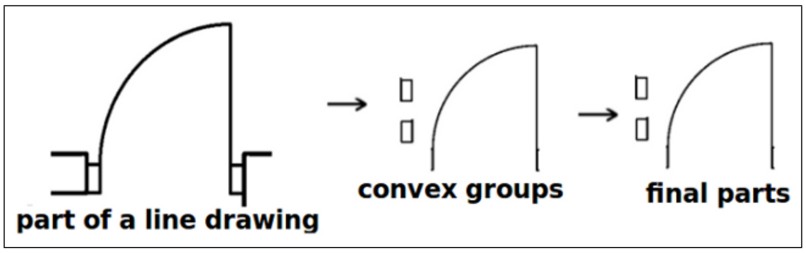
The overall description of the grouping method is described in the following (note that the steps of the method are numbered, and will be explained after the description of the method):

1. Apply preprocessing to convert a line drawing into line segments.
2. Apply Jacobs' grouping algorithm for finding salient convex groups, modified to find groups only in counter clock-wise direction.
3. Clean up the groups found in the previous step by removing:
    - Groups that have less than 3 segments.
    - Groups that are subsets or cyclic permutations of other groups.
    - Groups with any of the dimensions $>$ x pixels (x=the largest dimension of the largest symbol).
4. Keep the convex cycles only (the closed groups), and the open groups that have a series of neither horizontal nor vertical short segments (like arcs).
5. Keep the outer groups, i.e. remove the groups that are completely contained in other groups.

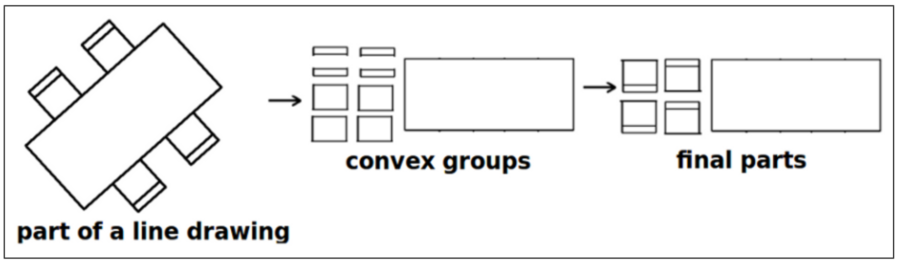
Figure 5.1 shows how the method performs grouping on three different inputs. Each of those inputs is a small part cropped from a line drawing, such that the part contains one symbol with the background lines connected to it if any. The figure also shows the intermediate output of the method (after step 2) before any cleaning up steps. It can be seen that the final output groups (after step 4) are fewer and more distinctive than the groups that were initially found (after step 2).

In the description of the grouping method described above, steps 1 and 2 were explained earlier in this section. Step 4 is optional and application domain dependent: the database is architectural drawings, the architectural symbols consist of closed parts, with very few exceptions like "doors" symbols. In other technical drawings where more symbols do not consist of closed parts, step 4 can be changed.

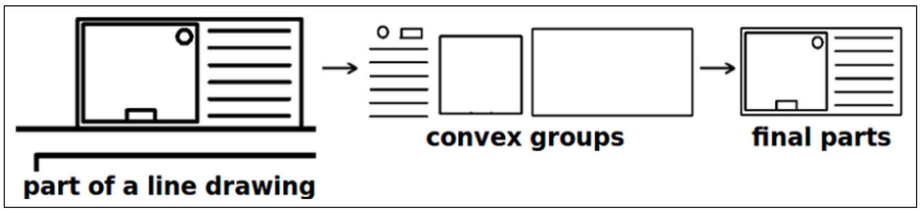
Step 5 produces the final output groups of line segments. The final groups are parts of symbols. A *symbol part* is the group of all segments that are located inside a group including the segments that constitute the group itself. Clearly a symbol part does not have to be convex. Before step 5, a lot of parts from different symbols are similar, like –for example- the rectangle shape. This step greatly reduces the number of the found groups, and more importantly, it outputs parts that are distinct (different than each other). Figure 5.2 shows the final output of the grouping method on a complete input image.



(a) The convex groups are the same as the final parts.



(b) Some convex groups are combined to form a smaller number of final parts.



(c) All the convex groups are contained in 1 outer group and form 1 final part.

Figure 5.1: The grouping method applied on line drawings: the final detected parts correspond to meaningful parts of symbols up to complete symbols.

As illustrated in Figure 5.2, the grouping method converts a line drawing to a set of parts of symbols, which can be viewed as effective segmentation for the symbols from their interfering background. The grouping method is considered an ROI-finding method as follows. The ROIs of a drawing, are the regions that enclose the parts of symbols found by the grouping method. A symbol part is constituted of the ROI features, so a symbol part (could be a complete symbol) is the set of line segments contained in an ROI (in Figure 5.2, this corresponds to all the line segments that lie inside a red bounding box). For each symbol part, the location information is kept, i.e to which image a part belongs and its location in that image. It can be seen that the found ROIs in Figure 5.2 correspond to meaningful parts of symbols up to complete symbols. If a human were asked to decompose the line drawing into parts, then the parts in Figure 5.2 could be one way to do it.

The complexity of the grouping method is determined as follows. The main step in the method is applying Jacobs algorithm. According to the complexity analysis by Jacobs in [124], his algorithm takes an expected running time of $O(n^2 log(n) + nm)$, where n is the number of segments in an image, m is number of found convex groups, and s is the average number of segments in a convex group. The cleaning up steps and "keeping the

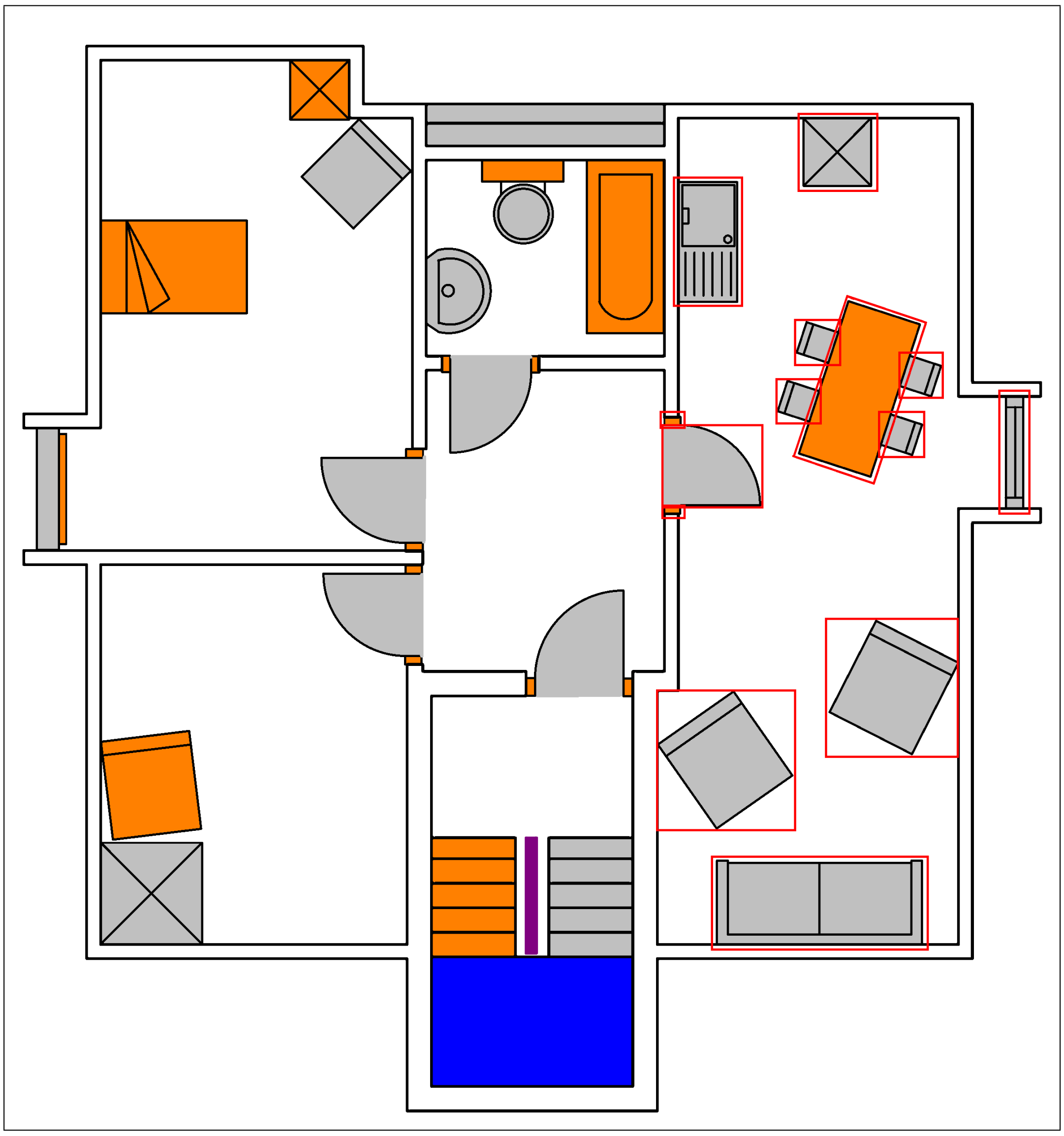


FIGURE 5.2: Output of the grouping method on a complete line drawing. The found **ROIs (symbols' parts)** of a line drawing are shown shaded (adjacent different ROIs have different colors). Red bounding boxes are drawn around only some of the found ROIs for clarity of illustration. The features of an ROI or a symbol part are the line segments that are located within the bounding box.

outer groups" step require two nested loops each of size m in the worst case, where each loop also iterates on the segments s of the group that is being considered in the current iteration, which gives a complexity of $O(m^2s+ms)$. This makes the total expected running time of the grouping method: $O(n^2log(n)+nm+m^2s+ms)$ where n, m and s are as defined above. As can be seen from Figure 5.1, the number of the final output parts is usually much smaller than m. This means that the complexity of the grouping method is dominated by the step of applying Jacobs algorithm. The running time is practice will be discussed in the next section.

5.5 Evaluation of the Grouping Method as an ROI-detector

5.5.1 The Database

The database used for evaluation is a set of 300 images taken from the "Sysed" publicly available[1] graphics recognition database described in Subsection 4.3.4. The images of the database are synthesis documents that imitate real complete floor plans with sizes between 2M to 7M pixels [43, 44].

The same set of 300 images will be also used for evaluation in the next two chapters. The grouping method will be used as the first step in a complete symbol retrieval system. The development of the retrieval system will be discussed in the next two chapters, where each step in the system will be evaluated on the same database.

5.5.2 Grouping Method Evaluation

The performance of the presented grouping method as an ROI-detector is evaluated with respect to the properties discussed in Section 5.2. In this section, three properties will be evaluated, namely: "repeatability", "accuracy" and "covering all symbols meaningfully". The "distinctivity" property can be judged based on the performance of clustering the ROIs which is done in the next chapter. As mentioned in Section 5.2, distinctivity of ROIs is usually evaluated based on the performance of the step that uses the detected ROIs like matching or clustering.

In a database of line drawings, there are many images, but the symbols inside the images come from the same symbol alphabet. Note that not all the symbols are present in each image. The ROIs that are detected by an ROI-detector should correspond to parts of the symbols (or complete symbols) of the symbol alphabet. In other words, the ROIs should correspond to the parts of the symbols that appear in the drawings. For evaluating an ROI-detector on line drawings, **"reference ROIs"** need to be defined. Those reference ROIs are defined as the parts of symbols (or the complete symbols) that would be detected by the grouping method of Section 5.4 given clean images of isolated symbols of the symbol alphabet.

The symbol alphabet of the database used in this evaluation consists of 17 symbols. Those symbols make 34 ROIs[3] (34 parts) when the grouping method is applied on them. Figure 5.3 shows the detected ROIs of some symbols of the symbol alphabet. In the figure, some symbols are represented by one part (one ROI), others are represented by more parts. In

[1] http://mathieu.delalandre.free.fr/projects/sesyd/index.html

[3] The symbol models provided by the database are 16, and they make 31 parts, but the "stairs" symbol is added as it appears in a lot of the drawings. The stairs symbol makes 3 parts.

the symbol alphabet of the used database, symbols are represented as a maximum of five parts.

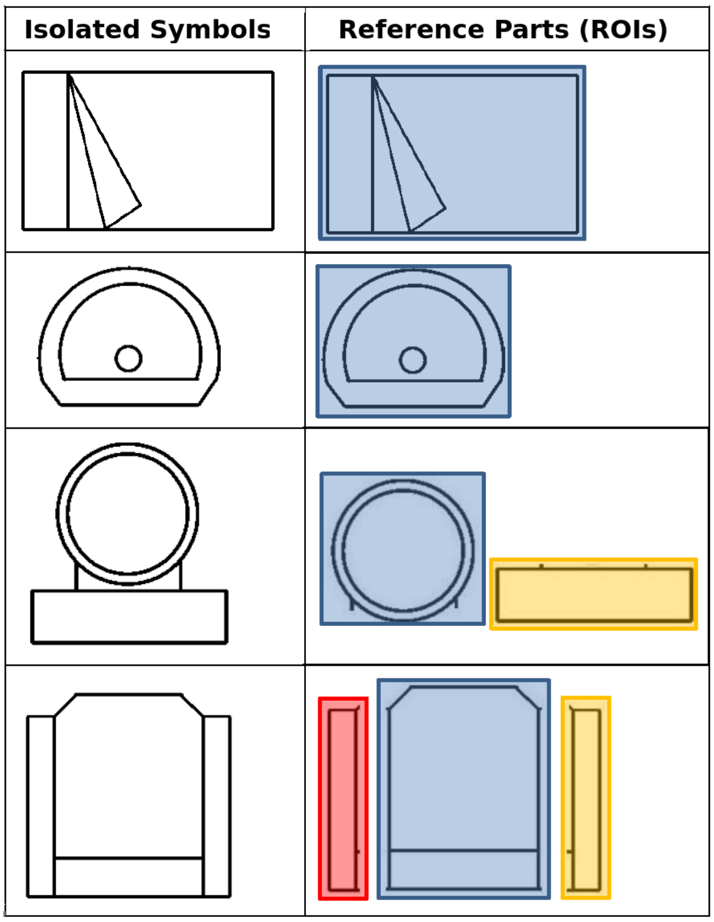


FIGURE 5.3: The grouping method applied on clean images of isolated symbols. The result is the “reference ROIs” (shown surrounded by colored bounding boxes) that will be used for evaluating the detected ROIs in complete line drawings.

Evaluation of the repeatability property of the grouping method

According to [111], the repeatability score of an ROI detector for a given pair of images is computed as: the ratio between the number of region-to-region correspondences and the smaller of the number of regions in the pair of images, considering only the regions located in the part of the scene present in both images. To adapt this to line drawings, the repeatability score is measured based on the correspondences between the ROIs found by the grouping method and the reference ROIs that appear in the drawings.

The *repeatability score* (“rscore”) for a reference ROI is defined as follows:

$$\text{rscore} = \frac{\#\text{ times a reference part is detected in all images}}{\#\text{ times the same reference part appears in all images}} \tag{5.1}$$

How is a reference part counted as detected?. Let a be the area of the ground truth bounding box of a reference part, and let b be the area of the bounding box that includes the detected ROI for the same reference part. The overlap error between the two areas is defined according to the PASCAL VOC criterion [62] for evaluating the overlapping of bounding boxes. The overlap error is calculated as follows:

$$\text{overlap error} = 1 - \frac{a \cap b}{a \cup b} \tag{5.2}$$

A reference part is counted as detected if the overlap error as defined in Eq. 5.2 is below some threshold. Note that there is no need to account for the transformation when comparing the areas a and b since the area a is set each time to the area of a reference part in a drawing, the bounding box includes the part as it appears in the drawing. The ground truth has the information about the bounding boxes of the symbols that appear in the drawings. For this evaluation, the overlap error threshold is set to 25%. This threshold value is used in evaluating spotting systems [46], where a symbol is counted missing if $> 25\%$ of its area is not covered in a spotted region.

The overall repeatability score is the average of the scores from all the 34 reference ROIs. In the database of 300 line drawings: **overall repeatability score = 95.8%**. Such a score indicates that the grouping method is a highly consistent region detector. The ROIs found by the method are consistent (i.e. similar regions correspond to similar symbols) despite the transformations on the symbols within the images. The score as an absolute percentage is very high compared to the scores achieved by regions detectors in gray scale images [111].

Evaluation of the accuracy property of the grouping method

The accuracy of the detected ROIs is related to the overlap error defined in Eq. 5.2. Choosing a lower overlap threshold means that the detected ROIs are more accurate, and it also means that a smaller number of the found ROIs count as *detected* parts. The opposite is clearly true, if the overlap error is increased, then more found ROIs count as detected parts (increasing the repeatability score), but they are less accurate. According to [111], the accuracy of a region detector is judged based on measuring the repeatability score as a function of the overlap error. The same is done here for the evaluating the accuracy of the presented region detector. Figure 5.4 shows that starting from an overlap error of 10% up to 60%. Overlap errors $\geq 60\%$ are not considered in [111]. The same is done here assuming that the overlap of $\geq 60\%$ between the detected region and the reference part is random. The figure shows that: (1) the repeatability score is constant from a strict overlap error of 10% up to 45%, the score increases a bit after that, (2) the repeatability score is very high with the strict overlap error (10%). This indicates that the presented region detector is highly accurate.

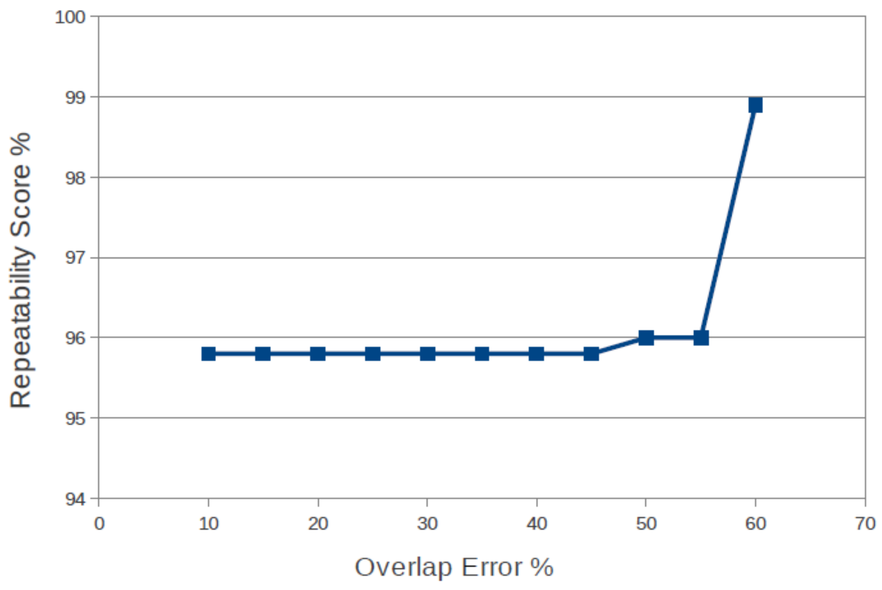


Figure 5.4: The repeatability score as a function of the overlap error. The behavior of this function is an indicator of of the accuracy property of the grouping method as an ROI-detector. Both the high value of the repeatability score at a small overlap error and the very small change in repeatability score as the overlap error increases, indicate that the ROIs found by the grouping method are very accurate.

Evaluation of the covering-all-symbols-meaningfully property of the grouping method

The last property of an ROI-detector is divided into two parts: "covering all symbols" and "***meaningfully*** covering the symbols". Finding or covering all the symbols can be measured based on the number of missing symbols. A symbol is counted as missing if more than 25% of its area is not covered by any of the found parts (the found ROIs).

$$\text{score of finding all symbols} = \frac{\#\text{ of non-missing symbols}}{\text{total }\#\text{ of symbols in all images}} \tag{5.3}$$

This score is similar to the repeatability score for complete symbols, but it is more relaxed as it does not require a specific overlap ratio with the reference ROIs. This means that, if a group of detected ROIs cover $\geq 75\%$ of a symbol, then the symbol is counted as non-missing. Either of the two scores can be used to indicate the covering of the symbols. In the database of 300 line drawings, there are **8264 symbols**, and **finding all symbols score = 99.7%**. This score value shows that the grouping method has successfully found almost all the symbols of interest in the line drawings.

All these scores (repeatability, accuracy, finding symbols) measure the quality of the detected regions that correspond to symbols, they do not evaluate, however, the other ROIs that might have been detected. This can be measured by evaluating the "meaningfulness" of the detected ROIs. A detected ROI or a detected part is counted as *not meaningful* or *irrelevant* if it does not represent a “reference ROI” or a reference part as defined earlier in this subsection (see Figure 5.3). For example, if the found part consists of random segments from one or more symbols or of segments that do not belong to any symbol.

$$\text{mscore} = \frac{\#\text{ relevant found parts}}{\text{total }\#\text{ of found parts}} \tag{5.4}$$

In the database of 300 line drawings, the number of the relevant ROIs found by the grouping method is **14215**, and the number of the total found ROIs is **14956**. This means: **mscore = 95.0%**. The high score value means that the grouping method finds almost always semantically meaningful ROIs.

The number of all the reference ROIs that appear in the drawings is **13224**. This number represents the ideal output of the grouping method, when it finds only the reference ROIs. However, the number of the ROIs that are actually found is 14956, which is larger than the number of ROIs in the ideal output. But note that, a lot of the extra found ROIs are actually relevant, for example, due to noisy line segments, the same symbol part can be found twice in two slightly different locations, and does not get removed in the cleaning step. Other extra parts are irrelevant, for example, the rectangles found below the “stairs” symbol in Figure 5.2, they are convex cycles, so the grouping procedure finds them as ROIs.

Running time of the grouping method
Finally, regarding the running time of the grouping method, it takes **16.6 seconds per image on average** on a 2.80GHz CPU. The ROI-finding is considered an analysis step, and is carried out off-line. The images of the line drawings are actually of big sizes, and a running time of < 20 seconds per image is considered practically fast for the off-line analysis step of ROI-finding. As discussed in Section 5.4, the running time depends mainly on the number of line segments in an image and the number of convex groups that can be formed.

5.6 The Grouping Method for Spotting and Retrieval

The grouping method introduced in Section 5.4 can be used to find ROIs for spotting methods. Alternatively, it can be considered as a content analysis step in the off-line stage of a retrieval system. As discussed in Section 5.1, Finding good ROIs can greatly improve the performance of spotting and retrieval methods. According to the evaluation presented in the previous section, the grouping method performs very well as an ROI-finding method. The ROIs found by the grouping method are consistent and accurate, and they meaningfully cover all symbols of interest. The following two subsections discuss how such good ROIs can be used for symbol spotting and retrieval methods.

5.6.1 Improving Symbol Spotting via the Grouping Method

As discussed in Section 5.1, a lot of spotting methods are based on finding regions of interest as a first step. The grouping method presented in Section 5.4, can be used within a spotting method for finding ROIs. Then, one can proceed in the different ways proposed in these spotting methods. One possibility is to compute descriptors of the found ROIs, those descriptions would be easier and more accurate to compute on isolated parts than on complete drawings with other graphical context. Another possibility is simply using isolated recognition methods on the found symbols' parts. As discussed in Section 5.4, the grouping method converts the hard spotting task into an isolated recognition task. One can also use graph representations of symbols' parts. In principle, the ROI-finding step in the different spotting methods can be replaced by the presented grouping method, and then the later steps of matching or indexing according to the query can be carried out.

This subsection shows how the grouping method can be used within the spotting method introduced in Chapter 3. The spotting method in Chapter 3 performs global search based on geometric matching. It will be shown later in this subsection that using the grouping method with spotting, improves the spotting in terms of both the speed and accuracy.

Recall that the input of the spotting method from Chapter 3, is a query symbol and a line drawing. The same preprocessing steps as the grouping method –morphological edge detection followed by segments sampling along the edges– are applied to both the query and the line drawing image.

The grouping method is incorporated in the spotting method as follows: After preprocessing, the grouping method is carried out as the next step. This results in regions that contain symbols or parts of symbols. Assume that each region is a separate small line drawing image. All those small images are then given as input to the spotting method. In this case, the "multi-image matching" variant of the spotting algorithm (Section 2.3) will be used. As explained in Section 2.3, the algorithm treats those inputs as different initial candidate matches that could have resulted from performing the matching on the complete image. After the matching is done, then the output can be located back in the complete line drawing, since the location of each small image within the line drawing is known from the grouping step.

Even though this way involves applying the matching on many initial images, it would be most of the times faster than matching the query against the complete image, and that is due to three reasons:

1. The number of segments in the image greatly affects the running time, so for big images with a large number of segments, it is less costly to do the matching on small

parts of the image many times –recall that an image part is a group of segments–, than to do it on the whole image once.

2. The running time is also affected by the size of the transformation space, and in small images the vertical and horizontal translation ranges are smaller than them in the complete image.

3. Given that the small images are isolated symbols or parts of symbols, properties like the scale range and feature weights can be computed, this helps speed up the matching, and eliminate the false positives.

Figure 5.5 shows the complete spotting operation sequence.

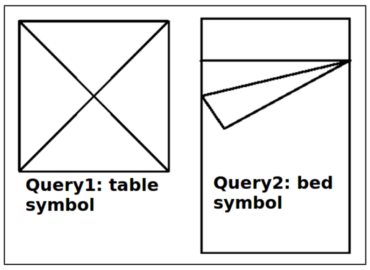
For evaluating the effect of using the grouping method with the spotting method, a database of 300 line drawings is used (a subset[4] of the graphics recognition database described in Subsection 5.5.1). As done in Chapter 2, standard recall and precision metrics are used to evaluate the spotting performance.

Note that the spotting method takes its input symbols from the output of the grouping method. Hence, if the grouping method missed some symbols, they will not be found by the spotting method affecting the "recall" value of the spotting negatively. In order to test the spotting performance independently, the query symbols that correspond to symbols that were not 100% found by the grouping have been excluded. On the other hand, the recall of the spotting is not affected by the extra irrelevant found parts. The irrelevant parts only increase the running time of the spotting method, as the spotting method has to perform extra matching operations. The precision of the spotting method is positively affected. The use of isolated parts (whether relevant or irrelevant) as input to spotting results in fewer false matches, as the effect of background lines is diminished.

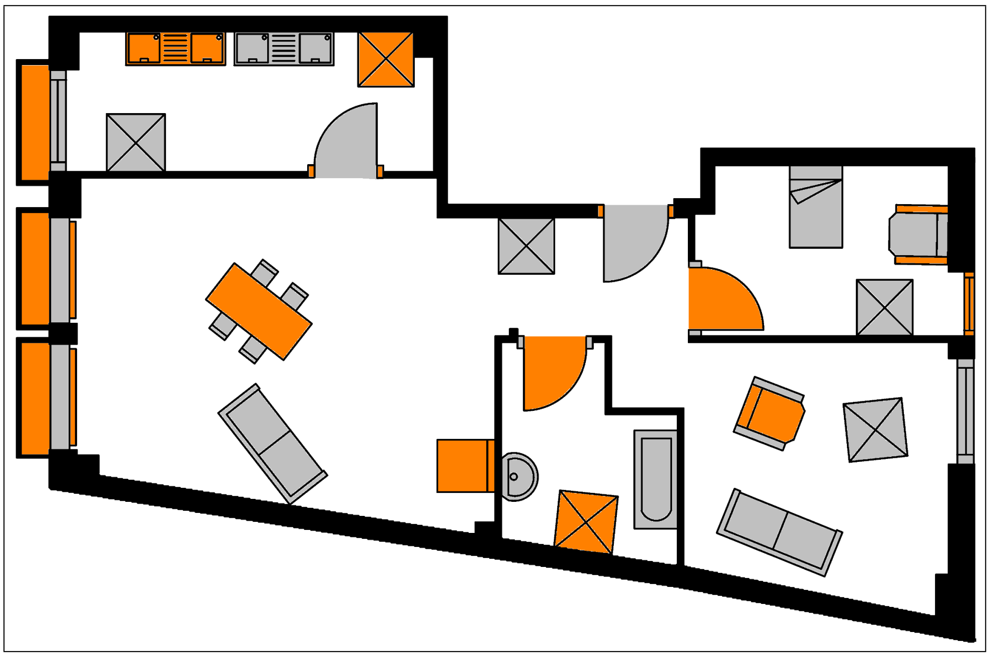
The results are shown in Table 5.1, where the performance of the spotting method is tested with and without grouping information. In both cases, the same parameters setting is used for accepting or rejecting the matches. (The spotting method "without" grouping information is the same spotting method discussed in Chapter 3.

As can be noticed in Table 5.1, the spotting results for the "sofa-1" symbol are lower than the average, that is mainly because it is a simple shape that has few segments, and it is almost a subset of some other symbols. Figure 5.6 shown the sofa-1 symbol. The simplicity of the shape of this symbol also causes slower matching even when it is being matched against isolated parts of symbols.

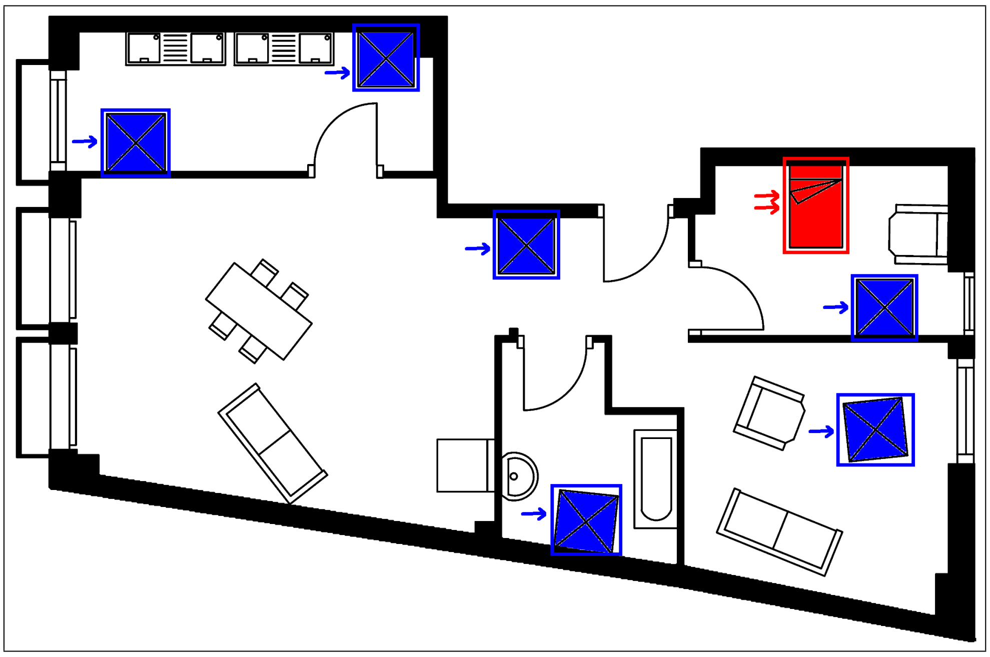
[4]In Subsection 5.5.2, a set of 300 drawings is used for evaluating the grouping method, and as mentioned, it will be used for evaluation in the next two chapters. However, this part of the work (Improving Symbol Spotting via Grouping) uses a **different** set of images that happens to also have 300 images. This part of the work was done independently and in a different time than the work described in the next two chapters.



(a) Query symbols.



(b) The output of the grouping method.



(c) The output of the spotting method improved by the grouping method.

FIGURE 5.5: The complete operation sequence of the spotting method using the grouping method as an intermediate step.

TABLE 5.1: Results of applying the spotting method (both the original and the one improved by grouping) to 300 images of architectural drawings. The 2nd column entries show the ground truth total no. of instances of each query symbol in all images, they sum to L=3900. And P1=3860, P2=3972 are the total no. of spotted symbols with and without grouping respectively.

Query	Symbol instances	Recall (**L=3900**)		Precision (**P1=3860, P2=3972**)		Avg. time per image (sec.)	
		with grouping	no grouping	with grouping	no grouping	with grouping	no grouping
bed	300	100%	100%	100%	100%	5.3	15.4
table	1172	100%	100%	100%	100%	9.2	19.1
sink-1	201	100%	100%	100%	100%	5.1	15.2
sink-2	112	100%	100%	100%	100%	4.7	88.4
tub	300	100%	100%	100%	100%	27.9	47.8
sofa-1	850	95%	85%	99.6%	80%	23.0	13.0
sofa-2	374	100%	100%	100%	100%	23.0	55.8

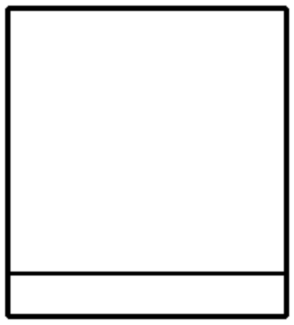


FIGURE 5.6: The "sofa" symbol. This symbol has a simple shape and is composed of few line segments. When such simple symbols are used as queries in the spotting method, they cause lower precision and slower running time than other symbols.

As for the running time, only the spotting running time is measured, assuming that the grouping step is done off-line. The use of the grouping method within the spotting system improves the spotting precision values and speeds up the spotting process.

5.6.2 Improving Symbol Retrieval via the Grouping Method

The importance of finding ROIs in a retrieval system has been discussed in the introduction of this chapter. The grouping method can be used as an ROI-detector in the off-line stage of a symbol retrieval system. Recall that the output of the grouping method is a set of local interest regions, or equivalently, parts of symbols. These symbols' parts can be considered as the "visual words" of the images. This is analogous to the visual words extracted by regions detectors in gray scale natural images. Once the visual words are extracted from each line drawing in the database, they are ready for the next off-line step in a retrieval system.

This chapter and the next two chapters describe a complete symbol retrieval system by dividing it into its three main steps: ROI-finding to identify visual words, building a visual vocabulary via clustering, and matching a query to the visual vocabulary. The performance of each step in the retrieval system to be presented is evaluated separately. The evaluation started by assessing the performance of the grouping method as an ROI-detector (Subsection 5.5.2), and will be continued for the steps presented in the next chapters till the final query retrieval step.

5.7 Discussion

This chapter has presented an ROI-finding method –called the grouping method – that is based on statistical feature grouping. The grouping method works on the vectorized representation of images as line segment features. The vectorized representation is obtained by simple preprocessing steps. The method is based mainly on grouping sets of features together if they satisfy the non-accidental property of convexity. The use of such grouping criterion enables the grouping method to find – with high probability – regions that correspond to semantically meaningful parts of objects up to complete objects.

The effectiveness of the grouping method as an ROI-detector is evaluated on technical line drawings. It has been shown that the method satisfies the required and desired properties of a region detector such as repeatability, accuracy and finding all objects of interest. It will be shown in the next chapter that the regions found by the method are also distinctive (i.e. different from each other) in the vast majority of the cases. Considering these properties for the task of finding ROIs in line drawings, the presented method has obvious advantages over state-of-the-art methods such as sliding windows, closed regions and methods used in gray-scale images like DoG-SIFT.

This chapter has also shown how the grouping method can be used to improve spotting methods. It has been shown that the grouping improves the spotting precision values and helps with the overall speedup. However, the grouping has its own major benefits that are independent of spotting: First, the presented method can be used as an effective segmentation method for segmenting symbols from the background. As shown in this chapter, the method converts a complete connected line drawing into a set of isolated and mostly meaningful symbols' parts. Second, the method can be used within the content analysis stage of a symbol retrieval system. This, actually, will be shown in the next two chapters.

Region detection is very important for describing the contents of images. Good ROI-detectors pave the way for the automatic analysis, interpretation, searching and retrieval of image databases. A good region detector for architectural line drawings has been presented in this chapter. By examining the performance (cases of both success and failure), it can be concluded that feature grouping is a very promising direction for the analysis of line drawings. The presented detector can be further developed to deal with different types of line drawings. Combining multiple non-accidental properties is one possible direction for improvement.

Chapter 6

Fast Retrieval via Automatically Created Symbol Indexes

Current retrieval systems use the concept of extracting a set of index terms from a set of images. Those terms can be indexed in the retrieval step to provide rapid access to elements in the image collection. A common model for extracting the index terms is to identify the visual entities in all images, then cluster the similar ones together, and use the centers of the clusters as index terms.
A similar model is presented in this chapter for the purpose of providing fast retrieval of symbols in line drawings databases[1]. The ROI-finding method presented in the previous chapter is used to identify the visual entities of line drawings. The visual entities are symbols or parts of symbols. Then, a novel symbol clustering method is presented for clustering the previously identified symbols' parts. The symbol clustering method uses geometric matching as a similarity measure between symbols' parts. The clustering results in a visual vocabulary of symbols' parts. The vocabulary is then used to build an index of symbols that appear in the line drawings. This symbol index is a compact representation of line drawings that allows building an efficient retrieval system. The presented symbol clustering method is highly accurate in measuring the similarity of symbols under transformations.

6.1 Introduction

A popular and successful model in the current content-based image retrieval systems is the bag of visual words (BoVW) model [126]. In this model, the contents of a database of images are represented as a "code-book" or a "visual vocabulary" of index terms. The index terms

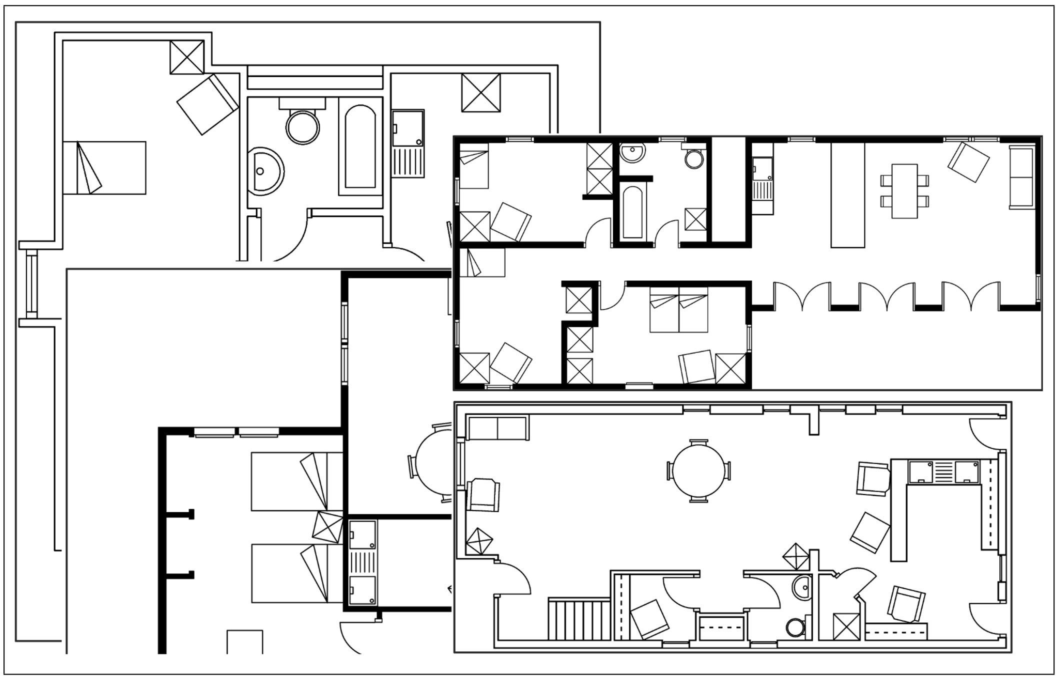
[1]Parts of this chapter are based on the author's work in [125].

represent clusters of the visual entities of the images. Building such a representation of the database is done during off-line stage of a retrieval system. The code-book representation is a compact and indexable representation of a database, that allows fast query retrieval in the on-line stage.

An off-line stage in a retrieval system that follows the BoVW model has two steps. The first step is identifying regions of interest (ROIs) in the images. This step might also include describing the ROIs with transformation-invariant descriptors. Applying an ROI-finding method on a database of images results in a very large number of image regions (corresponding to parts of objects within each image). Those image regions are the visual words (or visual entities) of the images. Clearly, searching to retrieve the desired visual words based on a query in such a space of visual words is computationally expensive. Instead, a more compact representation is sought in order to provide acceptable search response time. This makes it necessary to develop another step in the off-line stage of retrieval systems. This second step is usually done by applying clustering algorithms such as k-means clustering [127], self-organizing maps [128] or vector quantization [129, 130] on the visual words. The clustering reduces the very large number of visual entities into a relatively small set of clusters. The centers of the clusters are representatives of sets of similar visual words. Those representatives constitute the visual vocabulary. The visual vocabulary (or the index terms) is used to build an indexable data structure such as the famous inverted file index. Such an index is the required representation of the database.

In order to develop a complete symbol retrieval system that follows the BoVW model described above, the steps of the off-line stage need to be developed first. For ROI-finding, the grouping method introduced in the previous chapter is used. This ROI-finding method is the first step in the off-line stage of the retrieval system. For clustering, a novel symbol clustering method is presented in this chapter. The clustering method is used to build an index of symbols from a large collection of line drawings. The symbol index is a compact and indexable representation of the line drawings database. This clustering method is the second step in the off-line stage of the retrieval system. The outcome of the off-line stage is the symbol index, which can be used for the development of an efficient on-line stage of a symbol retrieval system.

As discussed in the previous chapter, applying the ROI-finding method on all the line drawings results in thousands of parts of symbols. Those parts are the visual words of the line drawings. The symbol clustering method is used to group the similar parts of symbols together. The input to the clustering method is the list of parts of symbols. Recall that a symbol part is represented as a set of line segments. This representation is kept, no invariant feature descriptors are used to describe the symbols' parts. So, the parts that correspond to the same part of some symbol could be rotated or scaled versions of each other. The clustering method uses geometric matching as a similarity measure. This is the same geometric matching-based method presented in Chapter 3. The symbol clustering method results in a set of clusters of symbols' parts. The representatives of the clusters represent the visual vocabulary. This vocabulary is the key component in constructing a "symbol index" that enables efficient access to the contents of the drawings. The symbol clustering method is evaluated on a database of line drawings and is found to be highly accurate in grouping the similar parts of symbols together.


(a) A collection of architectural line drawings.

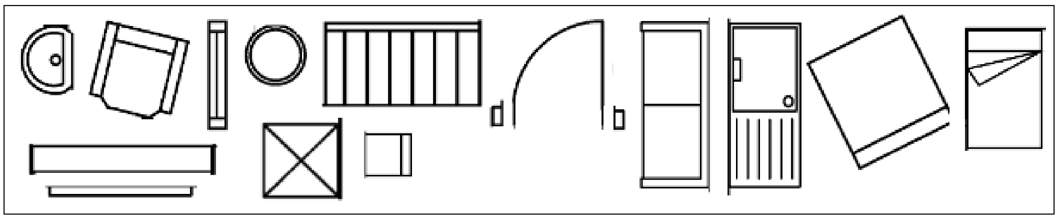

(b) A vocabulary of symbols and parts of symbols.

FIGURE 6.1: The symbol clustering method: building a visual vocabulary of symbols as in (b) from a collection of architectural drawings as in (a). For clarity of showing, only some drawings are shown, and only a part of the visual vocabulary.

Figure 6.1 illustrates the input and output of the symbol clustering method, from a collection of drawings as in Figure 6.1(a), we get a set of clusters' representatives as shown in Figure 6.1(b), each representative is a symbol part up to a complete symbol, and each cluster contains all the parts of symbols that are similar to the representative of the cluster. Using the representatives of the clusters to build a symbol index is also presented in this chapter.

6.2 Clustering and the BoVW Model in Line Drawings: State-of-the-art

Clustering can be defined as the unsupervised partitioning of patterns (observations, data items or feature vectors) into a number of clusters (groups, subsets, or categories) such that the data points in a cluster are more similar to each other than points in different clusters [131]. Clustering has many similar definitions in the literature [132, 133]. Clustering is a useful tool for exploring and analyzing the structure of data, and it has many applications such as: image segmentation, object and character recognition, data mining and information retrieval [131]. In text or image retrieval systems, clustering is used to improve the efficiency of retrieval algorithms. This is done by exploring the similarities between images (or parts of images) in a database in order to reduce the number of matching operations during the on-line retrieval stage.

The notion of clustering visually similar parts in gray scale images has been introduced in the BoVW model [126], and has been used frequently since then in object recognition and multimedia retrieval [109, 134]. However, such a model has not been used much in the recognition and retrieval of line drawings. This is probably due to the difficulty of identifying ROIs in line drawings, which is a necessary step that precedes clustering. Moreover, clustering requires a similarity measure that is capable of comparing the shapes of symbols.

Very few spotting systems in the literature used the concept of clustering. In the following, the off-line stage (ROI-detection and clustering) within these spotting systems is described. The performance of the clustering step largely depends on the ROI detection and description step in addition to the characteristics of the clustering method itself. The ROI-detection steps of these methods were discussed in Chapter 5; in this section, the emphasis is on the steps of ROI-description, ROI clustering and the indexing structure of these methods.

Nguyen et al. [40] proposed a symbol spotting method that is very similar to the BoVW model. As mentioned in Section 5.3, DOG-SIFT [103] is used to detect local ROIs. The ROIs are described using shape contexts, and then clustered by k-means to build a visual vocabulary. It is not clear how this model would scale using a larger number of symbols and line drawings. An inverted file index is built from the clusters (similar to the inverted index of text retrieval systems). The location information of the ROIs are also kept in the index, since the spotting system should locate regions in the drawing not the complete drawings.

Kong et al. [42] presented a spotting method where the ROIs are found using overlapping sliding windows. The ROIs are then described by a shape descriptor called the deformed blurred shape model. The descriptions are computed at different scales and orientations.

Then the vectors describing the ROIs are clustered using k-means. In their method (as well as in the method of Nguyen et al. [40]), k-means is used as a clustering method. The use of such a simple clustering method means relying more on the goodness of the output of the preceding ROI detection and description steps. There are inherent problems in their ROI-detection and description step such as the use of overlapping sliding windows to detect regions. The unsuitability of this method as an ROI detection method in line drawings was discussed in the previous chapter. Moreover, the used ROI descriptor is not invariant to rotation and scale, so, the ROIs have to be stored many times at different scales and orientations. The increased number of ROIs makes it harder for a clustering algorithm to output the correct partitioning of the data. For indexing, hash tables are built from the clusters for each image, where the labels of the clusters are the keys to a hash table.

Luqman et al. [41]) extracted ROIs from graph-based representations as discussed in Section 5.3. The ROIs are converted to fuzzy structural signatures. This descriptor is also introduced by them where a structural graph representation is converted to a numerical vector representation. The similar signatures are then clustered together using agglomerative (or hierarchical) clustering method from the Statistics Toolbox of Matlab. The city distance is used as a similarity measure of clustering. Labels are assigned to clusters. An inverted file index is used for indexing the contents of the drawings. The keys of this index are learned by a Bayesian network, and the index naturally contains the ROIs locations and the labels of the clusters to which they belong.

The clustering method presented in this chapter is to be applied on the output of the ROI-finding method presented in the previous chapter. This approach is conceptually similar to the BoVW model, however, it uses techniques that are more suitable for dealing with shapes and line drawings. The presented clustering method uses geometric matching (see Chapter 3) as a similarity measure, which has been shown to be quite effective in comparing shapes whether isolated or in context. The use of such matching measure for clustering symbol shapes has been only proposed in our previous work [125].

6.3 The Symbol Clustering Method

6.3.1 Constructing a Visual Vocabulary of Symbols via Clustering

The input to the clustering method is the large number of visual words that were detected by the ROI-finding method described in the previous chapter. Having a list of symbols' parts from all the images of the database, clustering is required to find the similar parts and put them together. Recall that a visual word (an ROI or a symbol part) is represented as a set of line segments. This representation is kept, no invariant feature descriptors are used to describe the symbols' parts. So, parts that correspond to the same part of a symbol could be rotated or scaled versions of each other. Matching the symbols' parts together is done using geometric matching.

This clustering algorithm is fairly straightforward. A symbol part is picked from the list, and is matched, via geometric matching, to the rest of the parts. All the matching parts are placed in a cluster. Another symbol part is picked from the remaining parts and is matched to the remaining parts and so on. Since highly similar parts should be clustered together, a match between two parts is only accepted if one part matches $\geq$ 70% of the other part's segments. This threshold is set experimentally, and it works well with relatively clean images (this threshold is set using the "accept-match" parameter discussed in Subsection 2.2.2). After that, the very small clusters are matched again to the rest of the clusters with a smaller accept-match threshold, then either combined with other clusters or discarded. The following is the clustering algorithm.

- *clusters*: an initially empty list of lists.
- *parts*: list of all parts, marked "false" meaning they do not belong to any cluster yet.
- for each part p of the parts list:
  - if p does not belong to any cluster:
    - add p to *clusters*.
    - use the geometric matching from Chapter 3 to match p with each of all other marked "false" parts (using an accept-match parameter of 0.7).
    - add the accepted matches of p to the list of *clusters[p]* with their information.
    - mark the accepted matches of p as "true".
  - else: (p already belongs to some cluster):
    - skip p.
- for each cluster that contains $< n$ parts:
  - use the geometric matching from Chapter 3 to match the cluster representative with each of all other representatives of the clusters (using an accept-match parameter of 0.5).
  - if the cluster representative is matched to another cluster representative: merge the two clusters.
  - else: discard the cluster.

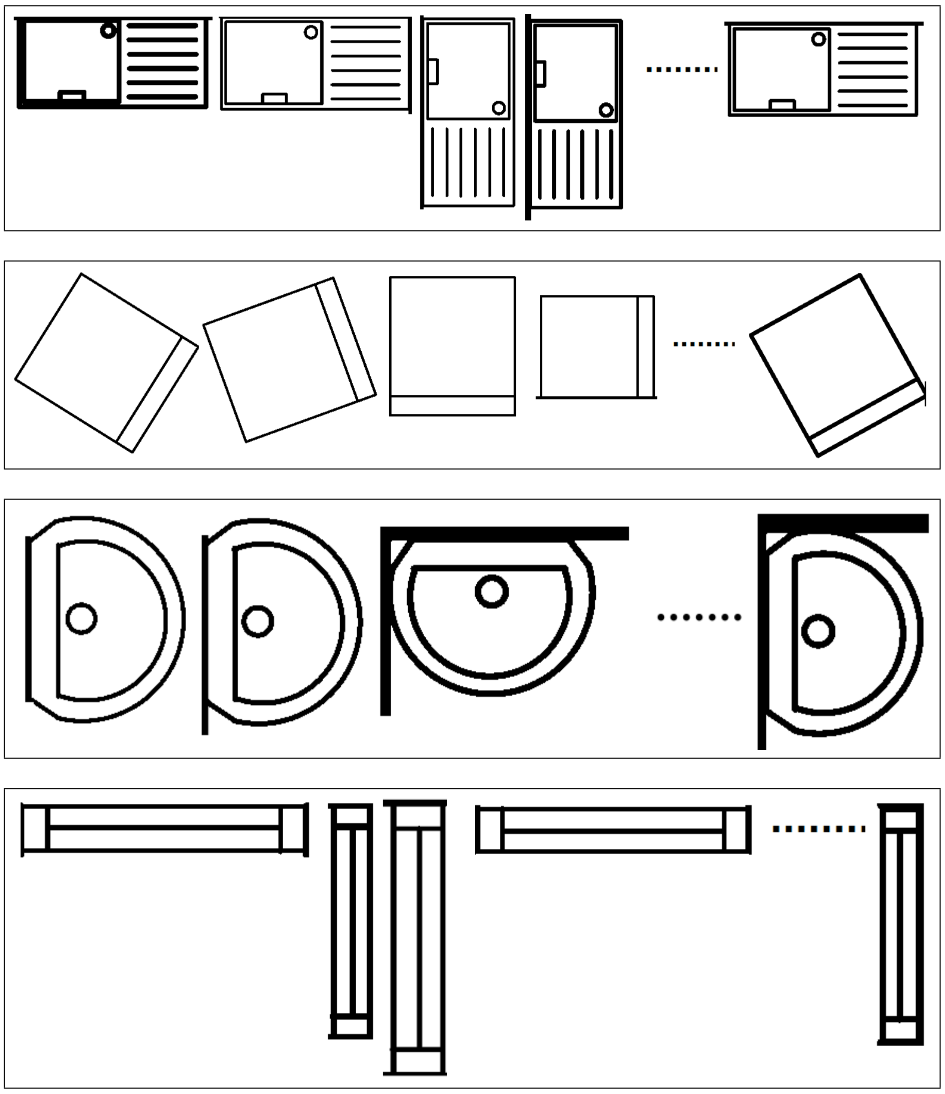


Figure 6.2: Resulting clusters: example clusters of parts of symbols.

Figure 6.2 shows some of the resulting clusters. In this basic version of the algorithm, we do not use soft clustering. That means, the parts that belong to a cluster are **not** considered in creating the next clusters. The part that is currently being matched is considered as the cluster center, or -strictly speaking- the cluster representative, and it is matched to the other parts using a strict acceptance criterion, so, it can be assumed that the matched parts do not belong to any other cluster. In case of false matching, extra clusters will be formed. It would be useful to investigate the effect of soft clustering or other clustering variants on the performance of the presented basic clustering method.

In the presented clustering method, there is no need to specify the number of clusters beforehand, the geometric matching step will control the output number of clusters, based on the mentioned acceptance criterion (a pattern should match at least $\geq$ 70% of the other pattern). This is advantageous to k-means where the number of clusters needs to be specified in addition to the similarity measure and its threshold.

Note that using clean images or using a good vectorization algorithm, the order in which the parts are examined for clustering, does not affect the clustering results. Consider the case of two parts composed of line segments, where the segments have one-to-one correspondence (each segment completely matches another segment under some transformation). In this case, the geometric matching as defined in Chapter 2, will give the same matching score if the first part is matched to the other part or vice versa. However, due to noise or using simple vectorization algorithms, a line segment can be represented as multiple smaller line segments (like a dashed line possibly with different dash lengths). In this case, a part that contains such segments cannot be used as the model (as a first part). The partial segment matching defined in Section 2.2.3 assumes that the model-image matching is one to many, where one or more segments of the image (the second part) can be matched to one segment of the model (the first part) under some transformation. This problem can be handled easily as follows. The input parts are sorted in an ascending order according to the number of segments. An alternative solution is to use a split-and-merge algorithm such as the one presented in [135] on the segments of the parts before clustering.

The clustering method results in a set of different clusters of parts of symbols. The number of clusters is much smaller than the total number of parts extracted from all images in the ROI-finding step, hence, this set is a compact representation of the database. After clustering, the visual vocabulary of the line drawings is built from the clusters as follows. A cluster representative, i.e. a symbol part, is selected from each cluster. The set of the clusters' representatives compose the visual vocabulary.

6.3.2 Building a Symbol Index

Having identified clusters of the visual words and built the visual vocabulary, one last step remains: building the symbol index. The symbol index that will be constructed is similar to the inverted file index [136, 137] used popularly in text retrieval and content-based image retrieval. The inverted index is a data structure that is an important component of a typical search/retrieval engine. The goal of a retrieval system is to make the querying as fast as possible. After analyzing the contents of the images off-line, a list of visual words per image is available, this is called the forward index. Querying the forward index would require sequential iteration through each image and each visual word. So, instead, an inverted index is built as a list of images per visual word of the vocabulary. The inverted index allows us to access the visual words based on the query words, instead of accessing image by image.

In its basic structure, the inverted index stores a mapping from content to location: for each visual word, a list of images that contain this word. Such a data structure, allows for efficient searches even for huge amounts of images. It allows for searching without accessing the images, all information about the visual words in images can be precomputed and stored during the off-line stage. In the area of information retrieval, much work has been done on the development and improvement of the inverted file index from different aspects like storage and construction [136, 137]. For the symbol retrieval system presented in this work, a basic symbol index is constructed for the purpose of providing a complete working off-line stage of a symbol retrieval system.

The previous section shows how to obtain clusters' representatives that represent the contents of the database. Those representatives are the key entries of what we call the *"symbol index"*. The symbol index is constructed based on the list of all clusters' representatives. Each representative is a symbol part, and it contains a list of nodes that correspond to its cluster members, each node contains the "id" of the image to which a cluster member belongs, and the location of the cluster member within that image. The location refers to the coordinates of the bounding box that contains the symbol part. In other words, this bounding box is the ROI that was detected for the symbol part. Recall that in symbol spotting/retrieval applications, the regions that contain a query symbol within an image have to be returned to the user, not only the images themselves. One more item can optionally be added to each node of the symbol index, namely, the set of line segments that constitute the symbol part.

As will be shown in the next chapter, such symbol index is enough for the retrieval of regions in line drawings based on a query. However, such an index (also the inverted file index) maintains no information about the relative spatial layout of the visual words per image. Hence, a spatial verification step is typically performed on the images retrieved for a given query to improve the precision of the system. In order to implement such a spatial verification step, additional information needs to be stored in the symbol index as follows. Each cluster is assigned a "cluster-id", and every symbol part in every cluster is also assigned an id (ROI-id). Then, for each symbol part, the ids of the 4 closest parts of symbols – within the same image – are stored. In the next chapter, we explain how to use this information for the spatial verification step during on-line retrieval. Figure 6.3 shows the structure and the contents of the symbol index.

Using such an index will allow query retrieval in time that depends on the size of the symbol index i.e. the size of the visual vocabulary, and is independent of the size of the database. It is worth mentioning that an inverted index is even more efficient for sparsely populated spaces [138]. This does not hold for our specific application domain (architectural floor plans) or technical line drawings in general, where most of the visual words are present in most of the drawings. The vocabulary space can be sparse when the database is a collection

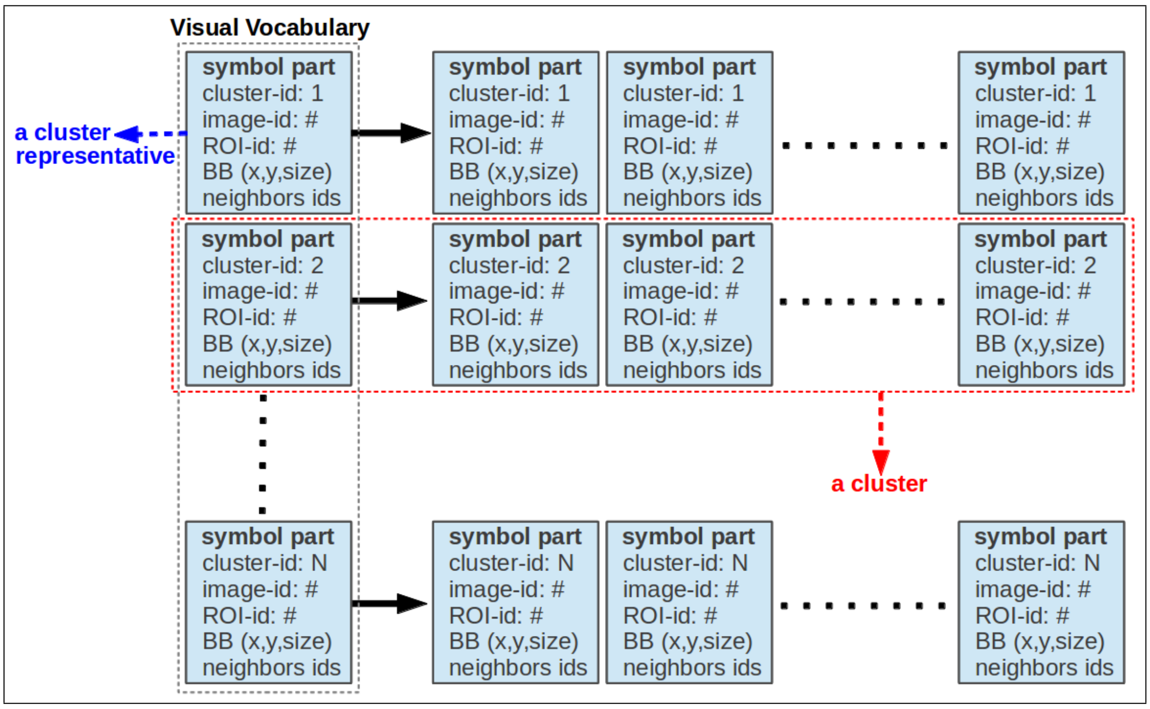


FIGURE 6.3: Symbol index structure. Each node is a symbol part or a visual word that contains location and other information. The symbol parts in the left most column are the clusters' representatives (visual vocabulary) and each of them points to its list of cluster members.

of images from different domains. However, the symbol index makes all the desired regions in the desired images available efficiently, this suits the user purposes of browsing and retrieval of line drawings.

Building the symbol index concludes the off-line stage of a retrieval system. Adding new images to the database of the system does not require any re-clustering. When a new image is added, the same steps discussed in the previous subsections will be performed off-line on the new image. First, the grouping method is applied on that image, and a list of ROIs (symbol's parts) is obtained as a result. Then these parts are matched –via geometric matching – to the clusters' representatives. Each part is assigned to the best matching cluster. If a symbol does not match any of the clusters, then a new cluster is created. Ids are assigned to the newly added parts of symbols. Adding a new image requires only the processing needed for that new image.

6.4 Performance Evaluation

The input to the clustering method is the list of all symbols' parts (visual words) that resulted from applying the ROI-finding method on all the images of the database. Hence, evaluating the performance of the symbol clustering method is done on the same database used to evaluate the ROI-finding method in the previous chapter. The same 300 images of architectural floor plans is used (see the database description in Subsection 5.5.1). The symbol clustering method should partition these symbols' parts into a number of clusters, such that the parts within a cluster are similar, and the parts in different clusters are different.

Since the symbol clustering method essentially performs the clustering operation, it will be evaluated based on criteria that are similar to the ones used in the literature for the evaluation of clustering. A lot of research has been done on finding the appropriate criteria and techniques to evaluate the correctness of clustering results. Such evaluation techniques are referred to as clustering validation techniques [139]. The different measures used for clustering evaluation are all basically evaluating the "internal homogeneity and the external separation" [140], i.e. patterns within the same cluster should be similar to each other, while patterns in different clusters should be dissimilar.

The main approach for evaluating a clustering algorithm on two-dimensional data is to *visually* verify the validity of the results. Visualization of the data is an effective strategy to the verification of clustering, and is crucial to assess clustering validity [139]. However, with dimensions larger than three, such visualization is very difficult, which limits the use of this evaluation strategy [139, 140]. Assessing the performance of the symbol clustering method presented in this chapter is done based on visual verification. Recall that the input to the clustering method is meaningful parts of symbols. Those parts can be easily visualized, and after clustering, the parts (visual words) within each cluster can be visualized and examined.

The clustering evaluation is based on the following **adapted** recall and precision metrics:

- **cluster recall**: # of symbols' parts in the cluster that are similar to the cluster representative divided by # of all occurrences of that symbol part in the database.
- **cluster precision**: # of symbols' parts in the cluster that are similar to the cluster representative divided by the total # of parts in the cluster.

The clustering recall shows its ability to find a compact set of clusters i.e. similar symbols are not divided into two many clusters. While the clustering precision indicates the within-cluster similarity. Table 6.1 summarizes the results.

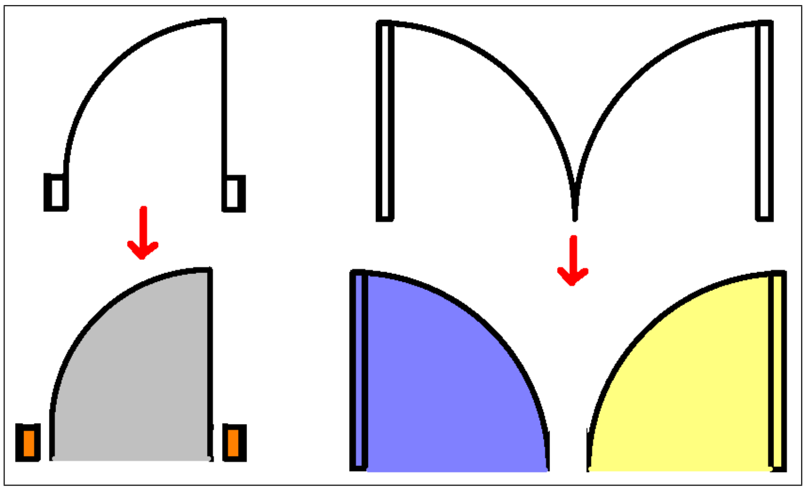
In Table 6.1, the "parts to be clustered", are the parts found by the ROI-finding method (see Subsection 5.5.2). Clearly, the performance of the clustering is affected by the performance of the ROI-finding method, as the output parts of the latter method are the input to the clustering method. The symbols that are missed by the ROI-finding method will naturally

TABLE 6.1: Results of applying the symbol clustering method to 300 images of architectural drawings. The provided recall and precision values are the average values for all the formed clusters.

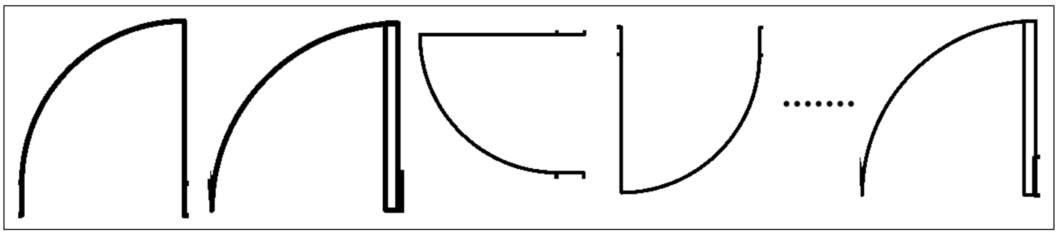
Number of parts to be clustered		**14956**
Number of the formed clusters		**34**
All Clusters	Avg. Recall	Avg. Precision
	90.1%	**94.3%**

be missed by the clustering and later by the retrieval. This means the recall value of the ROI-finding method affects the recall value of the clustering method.

Note that a symbol could have a part that is similar to a part of another different symbol. Those similar parts (belonging to different symbols) should end up in the same cluster. In this case, the recall for this cluster is calculated as if it were two different clusters, and then the two recall values are averaged. Figure 6.4 shows an example of a cluster that contains parts of two different symbols. The parts themselves are similar.



(a) Review of the result of the ROI-finding method when applied on parts of line drawing that contain the two different "door" symbols. Note that each of the two symbols has one part that is similar to another part of the other symbol.



(b) A cluster of parts coming from the symbols shown in (a).

FIGURE 6.4: Example of a cluster that contains parts of two different symbols.

In some cases, the ROI-finding method outputs some irrelevant parts. Those irrelevant parts will either be placed in the other relevant clusters (hence affecting clustering precision), or cause extra irrelevant clusters to be formed. In either case, this slows down the clustering process (as more parts need to be matched). Figure 6.5 shows examples of extra irrelevant clusters.

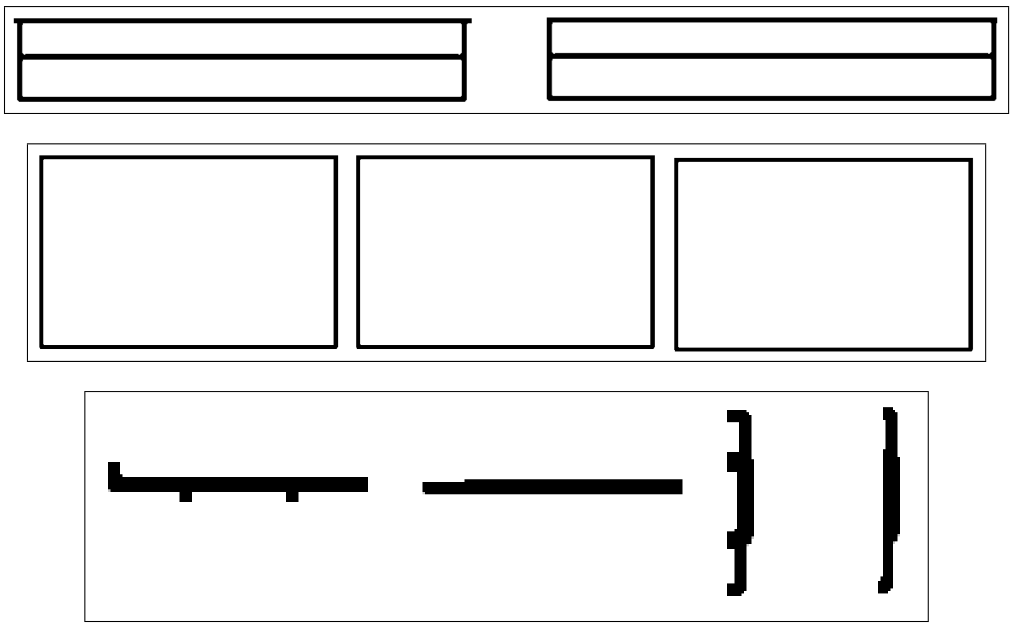


FIGURE 6.5: Example of clusters that contain irrelevant patterns. Irrelevant patterns are patterns that do not correspond to symbols or parts of symbols (meaningful symbols' parts are defined in Subsection 5.5.2).

As Table 6.1 shows, the clustering module has placed the similar parts of symbols in clusters with high accuracy. The drop in both the recall and precision values is caused by unsuccessful matching between parts of symbols. Figure 6.6 shows an example of a false matching. If a part was misplaced in a different cluster, then the recall value of the cluster missing that part will decrease, and the precision value of the cluster in which that part is misplaced will decrease.

The running time of the symbol clustering method is 45 min. on average per forming one cluster on a 2.80GHz CPU. This can be significantly improved by speeding up the matching step or by using different clustering schemes. However, the running time is still reasonable given the large number of parts to be clustered (14956) and the achieved high accuracy. The clustering step is to be carried out off-line.

In the following, the impact of the symbol clustering method is discussed. In the database used for the evaluation in this section, only 17 different symbols are used to draw the line drawings. Those symbols appear many times in many of the drawings. The total number of symbols that appear in all the 300 drawings is 8264, and they appear within the drawings

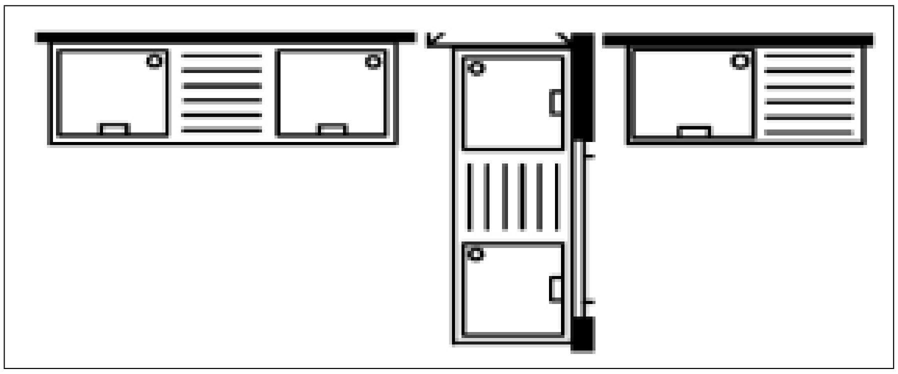


FIGURE 6.6: An example of a cluster that contains a false match. The third symbol (the false match) is very similar to the other symbols that were correctly matched.

connected to other symbols and lines, which means they have to be searched for and located. The 17 symbols, when decomposed into parts by the grouping method, make 34 parts, but many of those parts are actually similar, so, there are only 22 distinct parts. With perfect clustering – also assuming perfect ROI-detection –, 22 clusters should be formed.

Using the symbol clustering method presented in this chapter, a symbol index composed of 34 clusters is built. The symbol index represent the whole database with all of the 17 symbols and all their repeating 8264 instances, along with their location information in the database. This symbol index makes the potential applications of fast symbol retrieval from the database easy. If the user wants to retrieve a specific query symbol from this database, the query is matched only to the 34 clusters' representatives, and the best matching cluster will be retrieved. Without the symbol index, retrieving a query symbol would require searching in all the images sequentially using a symbol spotting method.

6.5 Discussion

Clustering the similar parts of symbols, is crucial for representing line drawings compactly. The symbol clustering method presented in this chapter is able to create a very compact and highly accurate representation of a database of line drawings.

The symbol clustering method takes parts of symbols in their vectorized line segment representation as input. The ability to work directly on the symbols' parts as line segments eliminate the necessity to develop (or use) feature descriptors (invariant or not), adding a feature description step means loosing more information along the way to the final representation of the database.

The clustering method is based on geometric matching as a similarity measure. The symbols' parts are matched under similarity transformations. The geometric matching (presented in Chapters 2, 3) is able to match symbols both isolated and in context. Such property is useful for clustering when the parts that result from the ROI-finding method contain additional background connection lines or some lines from the neighboring symbols.

The method presented in the previous chapter (the ROI-finding method) is combined with the clustering method presented in this chapter to develop an off-line stage of a symbol retrieval system. During this stage, the contents of a database of line drawings are analyzed off-line to create another representation of the database (the symbol index representation). This representation is much more compact and is used to build and indexable structure that can later be used for efficient on-line retrieval.

The approach for such off-line content analysis is conceptually similar to the BoVW approach. This approach has been and still is successful and popular in the field of content-based image retrieval. The researchers continue to develop new methods and improve existing methods for retrieval systems that are based on the BoVW approach. Many image and multimedia retrieval systems have been presented in the literature for gray scale images, however the retrieval of shapes and textureless line drawings remains a largely unexplored field. Dealing with line drawings requires developing new techniques that suit their textureless nature. In Chapters 5 and 6, such techniques are developed and combined to create a successful off-line stage of a retrieval system.

Chapter 7

An Efficient Large scale Symbol Retrieval System

Current symbol spotting and retrieval systems cannot achieve both high accuracy and efficiency on large scale databases of line drawings. This chapter presents a complete symbol retrieval system that achieves both efficient and high accuracy retrieval[1]. This is achieved by using a representation of line drawings that captures the contents of the drawings compactly and accurately.

The system follows the bag of visual words (BoVW) model by having an off-line content analysis stage, and an on-line query retrieval stage. The system uses the symbol index presented in the previous chapter as the outcome of the off-line stage. The development of the on-line query retrieval stage is presented in detail in this chapter. During the on-line stage, the ROIs of a query symbol are matched –via geometric matching – to the key entries of the symbol index. The matching symbols' parts from all the drawings are retrieved from the index, and spatial verification is performed on the matching parts.

Using the compact symbol index representation, the system achieves a query look-up time that is independent of the database size, and dependent on the size of the symbol index. This makes large scale symbol retrieval efficient. The retrieval system also achieves higher recall and precision than state-of-the-art symbol retrieval systems.

---

[1]This chapter is based on the author's work in [141].

7.1 Introduction

Current symbol spotting and retrieval systems cannot achieve both high accuracy and efficiency on large scale databases of line drawings. The systems that achieve relatively high recall and precision, are not efficient, they have to process each line drawing in a database sequentially. The systems that perform efficient query retrieval by indexing a compact representation of line drawings, do not achieve high recall and precision. This is mainly because a lot of information is lost while creating the compact representation. This chapter presents a complete symbol retrieval system that performs efficient query retrieval from a large database, and also achieves recall and precision rates that are significantly higher than state-of-the-art spotting and retrieval systems. The system uses a compact representation of a line drawings database, this representation has been shown to be highly accurate in capturing the contents of the database.

Symbol retrieval can be defined as large scale symbol spotting, which is concerned with locating a queried symbol in the images of a database of line drawings (See the detailed definitions of symbol spotting and symbol retrieval in Section 1.1). Symbol retrieval is similar to content-based image retrieval with the peculiarity that, instead of returning complete relevant images to the user, the retrieval system returns relevant information within the retrieved documents as a set of regions [10]. Such retrieval systems are referred to as focused retrieval systems.

Since symbol retrieval is similar to content-based image retrieval systems, one should benefit from the research ideas that have been introduced in this field. As discussed in the previous chapter, current image retrieval systems generally have two stages. The first is an off-line stage for analyzing the contents of a database. The outcome of this stage is creating a compact representation of a database that can be indexed efficiently during the querying process. The second stage of a retrieval system is an on-line stage where the actual retrieval of some query happens. During this stage, a query is analyzed and matched to the representative items in the compact representation created before. This stage might also include additional steps such as spatial verification of the retrieved items to eliminate falsely retrieved ones.

The symbol retrieval system presented in this chapter follows the same model of content-based image retrieval systems. The off-line stage of the system uses the methods presented in the previous two chapters 5 and 6. The output of the off-line stage (as detailed in Chapter 6) is a symbol index representation of a database of line drawings. The symbol index has a set of symbols' parts as key entries, each part is a cluster representative and has a cluster of similar symbols' parts. Each part in the clusters is represented by a node that contains information about the image to which the part belongs, and its location in that image. In the on-line stage of the system, the parts (ROIs) of a query symbol are identified using the ROI-finding method presented in Chapter 5. Each part is matched to the key entries of the symbol index. Matching the query parts in done using the geometric matching-based method presented in Chapter 3, which is also the same matching used for the clustering method presented in Chapter 6. The clusters whose representatives match the query parts are retrieved, and a final spatial verification step is performed on the retrieved parts. Figure 7.1 illustrates the idea of the symbol retrieval system.

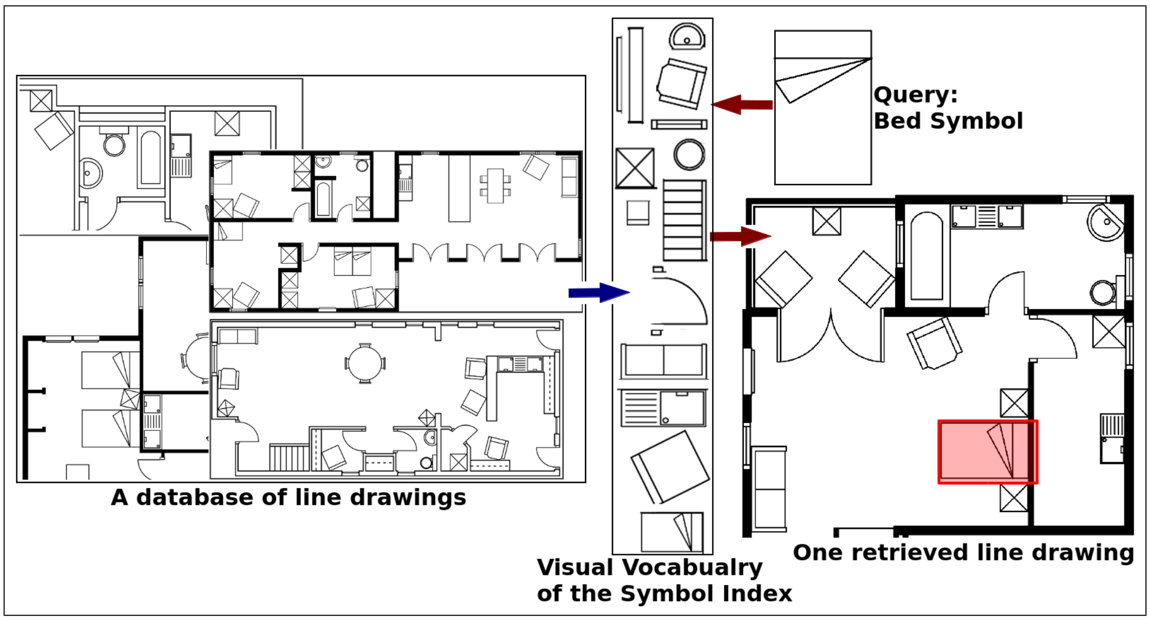


FIGURE 7.1: The proposed symbol retrieval system. A database of technical line drawings is processed off-line (blue arrow) and represented as symbol index whose key entries are the words of the visual vocabulary (due to space limits, only the visual vocabulary part of the symbol index is shown). A symbol is queried on-line, the symbol index is accessed (red arrows), and the retrieved ROIs are marked (red box) within the retrieved images.

By using the symbol index, the retrieval system is able to achieve a retrieval time complexity that is independent of the database size. The retrieval time depends on the size of the symbol index multiplied by the cost of the matching operation between a query and one symbol in the index. The retrieval system also achieves higher recall and precision than state-of-the-art symbol spotting and retrieval systems.

7.2 Symbol Retrieval in Line drawings: State-of-the-art

Symbol retrieval systems based on content analysis, i.e. systems that use concepts and ideas from text retrieval and content-based image retrieval, have been proposed only very few times for line drawings. Few symbol spotting approaches have been proposed bearing the retrieval concepts in mind. Such approaches have the potential of getting further developed into symbol retrieval systems. Section 6.2 presented a review of the off-line stages of some of these systems (the ones that follow the BoVW model). This section presents the overview of the ***same*** systems with emphasis on the on-line querying stage, and the indexing mechanisms, if any.

Nguyen et al. [40] used the BoVW approach [126] for the off-line stage of their spotting system. DOG-SIFT is used to detect ROIs, and shape context is used to describe them. The descriptors are clustered using k-means to build the visual vocabulary. An inverted file structure is used for indexing. For the on-line querying, the ROIs are matched to the centers of the clusters. A voting scheme based on the visual word frequencies is used to retrieve the best candidates from the matching ROIs. Some promising preliminary results were introduced, however, the scalability seems to be an issue in this system. The suitability of the used techniques to line drawings cannot be verified on the small database used in the performance evaluation of their work.

In the spotting system of Kong et al. [42], the ROIs are found using overlapping sliding windows, and then described by a shape descriptor called the deformed blurred shape model (DBSM) at different scales and orientations. Then the vectors describing the ROIs are clustered using k-means. For indexing, a hash table is built for every image where the labels of the clusters are the table keys. In the on-line stage, the ROIs of a query are identified and described, then matched to the centers of the clusters using Euclidean distance. The similar ROIs of a drawing are retrieved by indexing the hash table of that drawing. A voting scheme is used to retrieve the blocks of a drawing that contain all the ROIs that match the ROIs of the query. The results do not look very promising compared to the spotting systems in the literature that use the same database [10, 33]. In their system, a hash table is used for every image, which requires processing each image of the database sequentially at the spotting/retrieval time. Moreover, scalability is an issue when using a hash table. It might be worth investigating a better indexing mechanism where all the clusters of all the images can be stored.

Luqman et al. [41] presented a spotting system, where a mix of learning approaches with content-based retrieval approaches is used. An index is built for representing a database of technical drawings. The index contains clusters of ROIs (see Sections 5.3 and 6.2 for details). Class labels are assigned to the members of the clusters (the ROIs), and a Bayesian network classifier is trained on the descriptors of the members of the clusters. During the on-line querying, the ROIs of a query are extracted and described, and then classified as a member of some cluster. The corresponding ROIs in the drawings are retrieved by looking up the index. The system was evaluated on a fairly large database of electric and architectural drawings. The performance of the system is better on electric diagrams, this is because extracting the ROIs is easier on the electric drawings of the database as the symbols are well separated from each other. Using a learning approach requires re-training when adding new documents or symbols to the database. The system can be developed into a retrieval system by developing a similarity measure for the ROI-descriptors, this similarity measure can be used to compare the query descriptors to the centers of the clusters.

In the symbol spotting work of Rusinol et al. [10], the contents of line drawings are described as follows. Closed loops are used as ROIs to represent the sub-shapes of symbols, those sub-shapes are represented as poly-lines (group of line segments) and described by different off-the-shelf descriptors. Multi-dimensional hashing is applied as an indexing strategy. The hashing is improved through incorporating spatial information, where a proximity graph is used to represent the relation between the polylines. To retrieve a query symbol, the query is processed and its hash key is computed and used to index and retrieve the similar polylines of the documents. A Hough-like voting scheme is applied to retrieve the locations where the maximum number of the query polylines appear. The system showed promising results on a database of 42 architectural drawings, but as mentioned, scalability is an issue with hash tables. The drawbacks of the methods used for the off-line stage of this system were discussed in Section 5.3.

In the retrieval system presented in this chapter, a symbol index is created from a database of line drawings. Creating such an index of symbols and parts of symbols, is done as follows. Feature grouping is used an ROI-finding method, and geometric matching is used for clustering the ROIs. The symbol index is built based on the centers of the clusters. Using an approach that is conceptually similar to the BoVW approach, with techniques that are suitable for textureless line drawings, we are able to perform efficient and highly accurate symbol retrieval in large collections of line drawings.

7.3 The Symbol Retrieval System: the On-line Stage

This section describes the on-line stage of the symbol retrieval system. First, the off-line stage of the system is reviewed in the following. In the off-line stage, a database of technical line drawings is analyzed off-line and is represented as a symbol index. Each of the key entries in the symbol index is a symbol part (up to a complete symbol) and it represents a cluster of symbols that are similar to this key entry. The information about the location of each symbol part is stored in the symbol index during the off-line stage (see chapters 5 and 6 for details). For the on-line stage, a simple retrieval scheme is presented for the purpose of implementing a complete working retrieval system. The on-line querying stage has three steps. First, processing the query: vectorization and ROI finding. Second, matching the ROIs of the query to the key entries of the symbol index and retrieving the matching ROIs from all images. Third, spatial verification on the matching ROIs. The on-line stage is described in the following two subsections.

7.3.1 Efficient Query Matching

Assume a user interface for a digital library search is available for the users. A method for selecting the query is needed. Two options are provided to the user: either clicking on one of the representatives of the clusters that are displayed to the user, or selecting / cropping a region in a drawing. Note that some of the symbols' parts that are displayed are in fact complete symbols. If the user clicks one of queries (clusters' representatives) displayed to him, then this is a *Known* query. For such query, all the information of the query matches is already available, the parts to be retrieved are simply the members of the cluster whose representative is selected as a query. For each member of the cluster, the image number (image-id) is used to display the image to the user, and the relevant regions inside the image are marked by bounding boxes. Recall that each item (or cluster member) in the symbol index contains the image-id and the location (expressed as the coordinates of the bounding box) of that item.

If the user selects a region from a drawing, then the query is *unknown* to the system, and the symbols similar to the query need to be identified and retrieved. The query in this case is a segmented symbol (a symbol cropped from a drawing) possibly with parts of connection lines around it. Retrieving such a query is done as follows. First, the query image is vectorized and represented as line segment features. Then, the ROIs (symbols' parts) of the query are identified using the ROI-finding method presented in Chapter 4. This step converts the query symbol into one or more symbol's parts. Performing this step is fast as the query is a small image and consists of a small number of segments. Figure 7.2 shows examples of three processed queries.

Now, the query part(s) are ready to be matched. The query parts are to be matched only with the key entries of the symbol index (i.e. only the clusters' representatives). There are two cases, first: processing the query results in only one part. Recall that this part (ROI) is a actually a set of line segments, so as the clusters' representatives, hence, they can be matched using geometric matching (the same geometric matching that was used as a similarity measure for the clustering method in Chapter 6). This matching method is used to compare the query symbol to each of representatives of the clusters in the key entries of the symbol index. The system retrieves the members of the cluster whose representative achieves the highest matching score with the query. Figure 7.3 shows a sample input / output of the retrieval system for the one part-query case.

The second case is when the query has more than one part when processed by the ROI-finding method. Then, in addition to the matching step, a spatial verification step is needed. The next subsection explains this case.

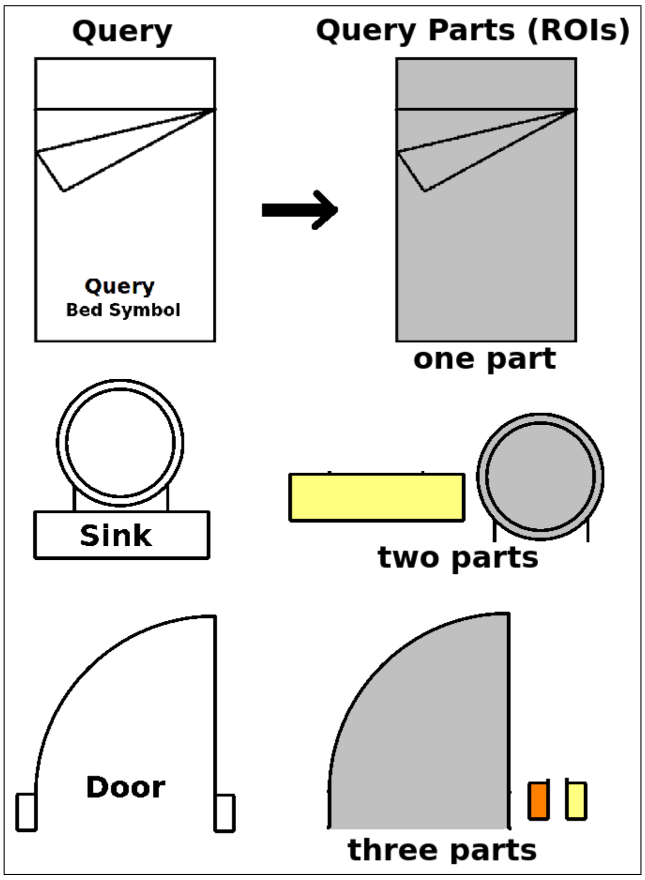


FIGURE 7.2: The first step of the on-line stage of the retrieval system: processing different queries using the ROI-finding method.

7.3.2 Spatial Verification of Query Parts

A part of a query symbol could be similar to parts of other different symbols. The similar parts (originating from different symbols) are naturally in the same cluster, and they all get retrieved when their cluster representative matches the query part. Obviously the parts that belong to symbols that are different from the query are false matches, and should be removed from the retrieved list. Even though some parts of different symbols are similar, not all the parts of the query are similar to the other parts of the different symbols. Using this observation, false matches can be detected by verifying that the retrieved parts are in the same region, and that they are arranged spatially in the same way as the parts of the query symbol. Such a procedure is called spatial verification, and it is a popular post-retrieval step used in most retrieval systems [138, 142, 143]. The main purpose of spatial verification is the elimination of false matches.

Spatial verification is a computationally expensive step (needs to be verified in each image). In content-based image retrieval, a lot of work has been done to make this step efficient [143, 144]. Zhang et al. [144] exploits the information of the visual words neighbors. In [144],

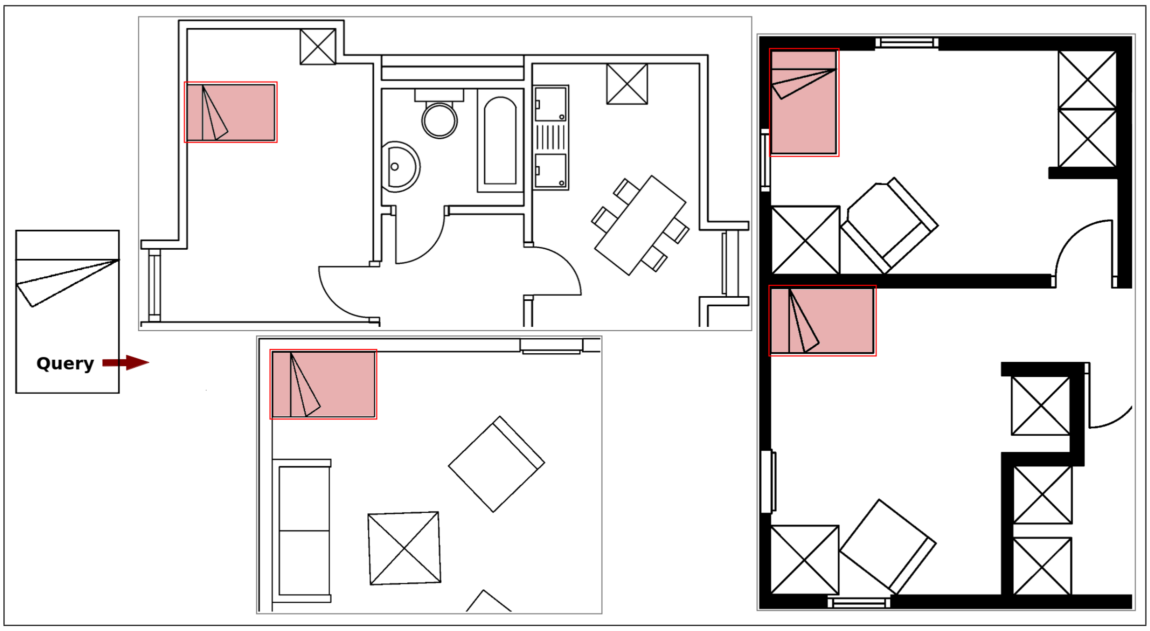
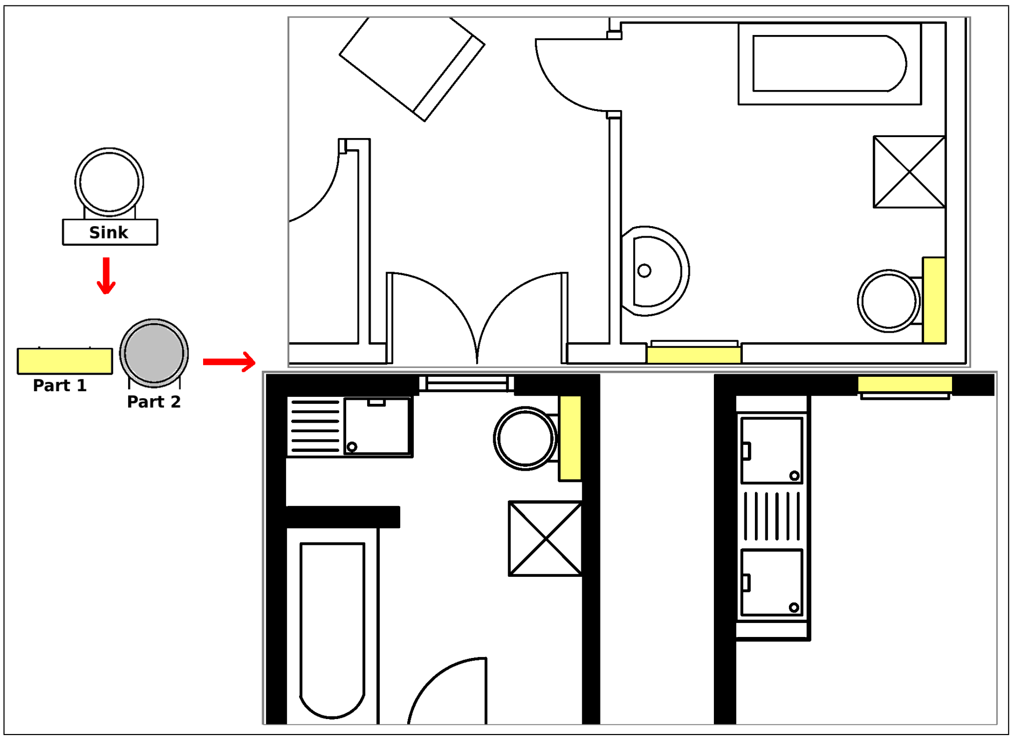


Figure 7.3: Results of the symbol retrieval system. A symbol is queried. Some of the retrieved results are shown (only parts of 3 different line drawings are shown due to limited space). Within each drawing, the retrieved regions that are found similar to the query are marked with red bounding boxes.

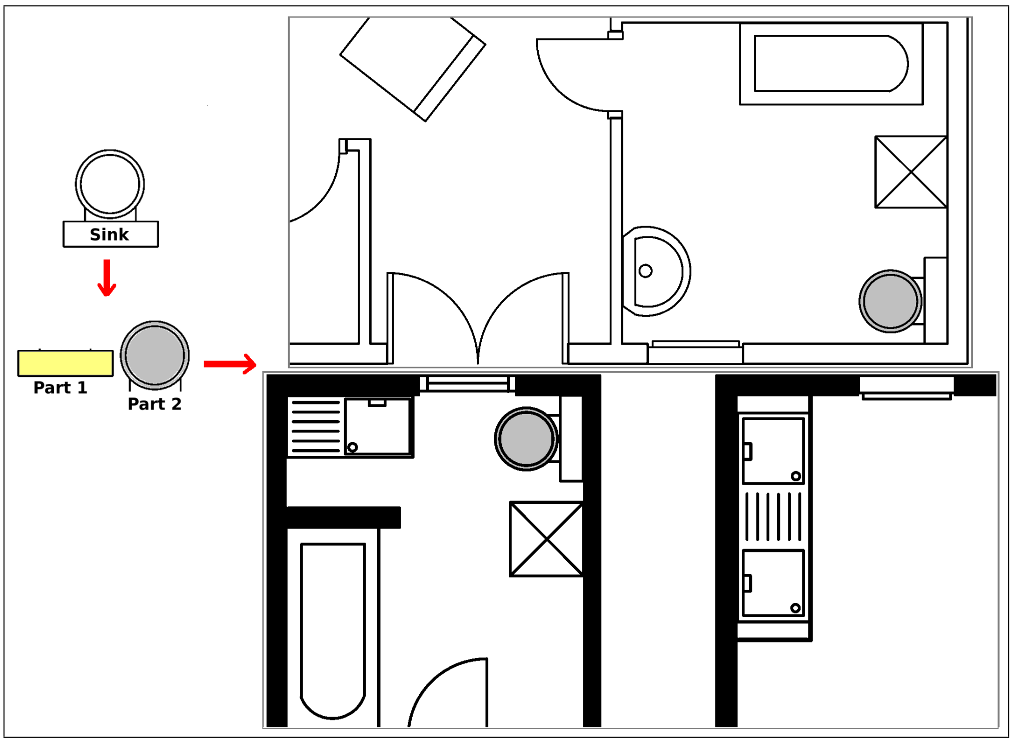
the inverted file is built such that, in the entry of each visual word, not only all the images containing this word but also the 15 spatial nearest visual words in each image. This information can be computed during the off-line stage, hence, the spatial check can be done simultaneously while retrieving the images. In their method, a visual word is retrieved if it not only matches a particular visual word of the query, but also has at least two neighbors that match other two visual words of the query.

A simplified version of the verification method of [144] is adapted for retrieval system presented in this chapter. During the off-line stage, the symbol index is built such that, the information of the neighboring parts are added to each symbol part. Recall from Section 6.3.2, the symbol index is built such that, for each symbol part, the ids of the closest four parts are stored (only the parts that belong to the same image are checked). The number of neighboring parts is chosen to be four because the symbols in the symbol alphabet of the drawings are decomposed to at most five parts by the ROI-finding method.

The retrieval of a multi-part query is done as follows. Each part of the query is matched – via geometric matching – to each cluster representative in the symbol index. The clusters that correspond to the parts that match the query parts are retrieved. Then, the query parts are associated with the cluster-ids to which they are matched. A retrieved image could have a number of retrieved parts. The part that matches the largest part of the query is picked, and its neighbors are checked to verify that, the other query parts have



(a) Retrieving part 1: The retrieved parts (shaded with yellow) are similar to the query part but they do not all originate from the same symbol. The parts that belong to other symbols are false positives.



(b) Retrieving part 2: The retrieved parts (shaded with gray) are all true positives.

FIGURE 7.4: Retrieval of a two-part query before spatial verification. The final retrieval result is shown in the next figure.

matches with the same cluster-id as the neighbors. Finally, to mark the bounding box of the final output region, the coordinates of the bounding box of the part that was initially picked is combined with the bounding boxes of its neighbors that match the rest of the query parts. Figures 7.4, and 7.5 show the steps of a sample input/output of the retrieval system for a two-part query.

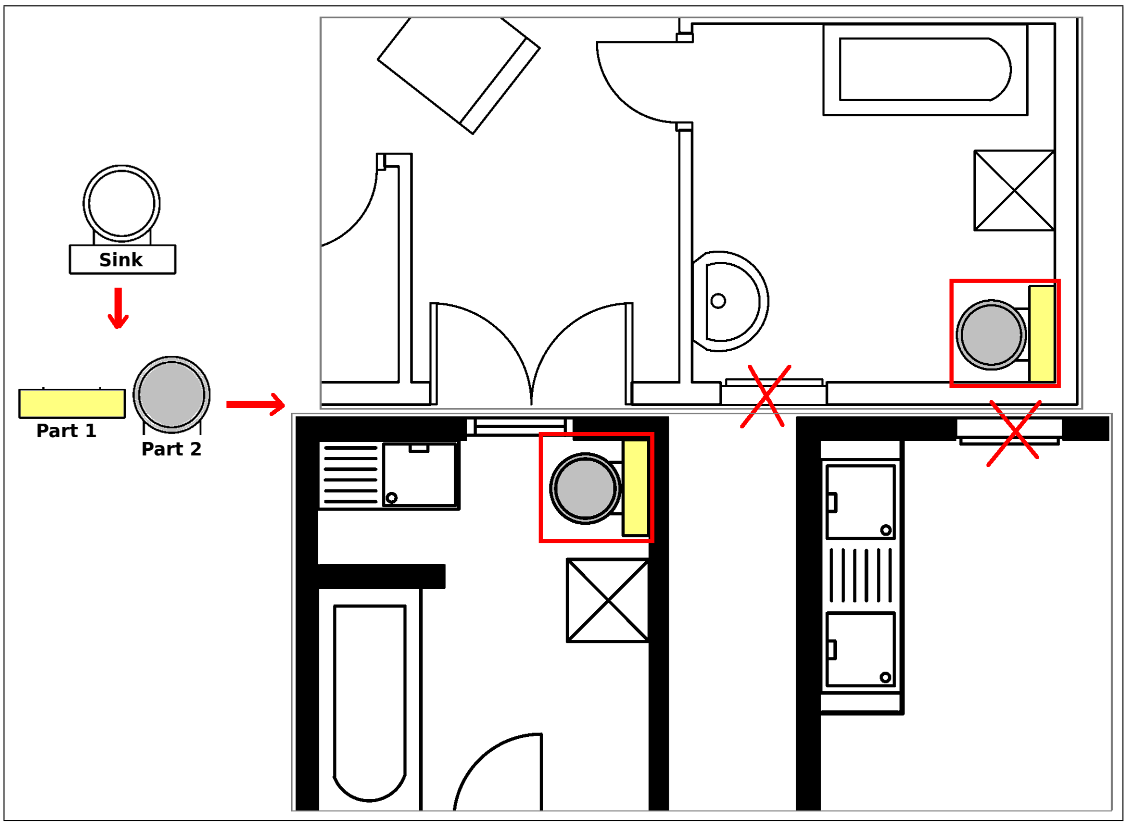


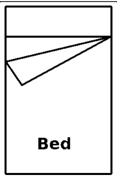
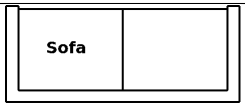
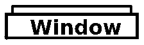
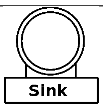
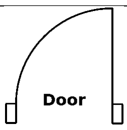
FIGURE 7.5: Retrieval of a two-part query: The final retrieval result after the spatial verification step. The retrieved symbols (surrounded by read bounding boxes) are all true positives.

7.4 Retrieval System Evaluation

In chapters 5 and 6, a database of 300 architectural floor plans was analyzed and represented as a symbol index. Evaluating the performance of the retrieval system is naturally done on the same database. The input to the retrieval system is a query and the symbol index. The system displays the retrieved regions within images of the database as output.

Standard recall and precision measure are used for query retrieval evaluation. Different queries from the symbol models of the database are tested. The on-line retrieval is carried out as described in Section 7.3. Table 7.1 shows the query retrieval results. The queries that have 100% recall and precision ("bed" and "sofa") come from clusters that are correctly formed. Other queries ("door" and "sink") come from clusters that have 100% recall, but they contain extra different symbols, however, because those queries are composed of more than one part, the retrieved false positives were eliminated during the spatial verification step.

TABLE 7.1: Results of the retrieval system on 300 images of architectural drawings using different queries. "Query parts" is the # of resulting parts after processing the query by the grouping method. "Symbol instances" is the ground truth # of occurrences of a symbol in all the images.

Query	Query parts	Symbol instances	Retrieval	
			recall	precision
	1	600	100%	100%
	1	290	100%	100%
	2	486	90.7%	91.6%
	2	300	100%	100%
	3	1000	90%	100%

The retrieval performance is naturally affected by the clustering performance in terms of both the recall and precision. If a part was misplaced in a different cluster, or one cluster got split into two or more clusters, then all the parts that were misplaced will not be retrieved (recall). At the same time, the cluster in which non-similar symbols are placed, will cause the retrieval of false positives (precision). Usually the retrieval recall rate can be improved by using soft clustering, where a symbol part can be assigned to more than one cluster.

Compared to the spotting approaches [40] and [41] that use subsets of the same database, our retrieval results are significantly higher and on a much larger subset of the database. This shows the effectiveness of the used techniques for dealing with technical line drawings.

7.5 Discussion

This chapter has presented a symbol retrieval system that is both efficient and highly accurate. The efficiency and the accuracy both come from using a compact and accurate representation of a database of line drawings. This representation is the “symbol index” discussed in the previous chapter. This chapter focused on the on-line stage of the retrieval system.

The on-line stage of the system follows a standard scheme for query retrieval as other content-based image retrieval systems. The query parts are identified and then matched to the words of the visual vocabulary (the clusters’ representatives). The matching clusters are retrieved and a final spatial verification step is performed on the matching parts.

Geometric matching is used for matching the query parts with the representatives of the clusters. As shown in Chapters 3 and 6, this matching works well for comparing shapes under similarity transformations. As in the clustering method, the geometric matching works on the line segments representation of the query, no descriptors are computed on the query or the clusters’ representatives. As for the spatial verification, a simple and efficient algorithm is chosen to verify that the parts that match the query parts lie in the same region. The spatial layout (geometrical consistency of the corresponding parts) is not checked. The adopted simple verification algorithm is mainly intended for the purpose of having a complete working retrieval system.

Chapters 5, 6 and 7 have presented a complete symbol retrieval system from technical line drawings. The performance of each step in this retrieval system is evaluated separately. The importance of such intermediate step evaluation is to know the sources of problems in a retrieval system. How much information is lost after each step. This helps identify the problems and solve them. The state-of-the-art retrieval systems usually evaluate only the final query retrieval performance, which does not help determining the causes of performance rising or dropping. Another reason for doing such evaluation, is that the different steps of a retrieval system may be useful on their own for other purposes. For example, using an ROI-finding method as a segmentation method. By examining the stages of retrieval systems (the state-of-the-art systems and the one presented in this chapter), it is argued that the ROI-finding step is the most important step. If the ROIs are found consistently and accurately such that they correspond to symbols, then the later steps of clustering and retrieval become easier to develop and yield better performance.

The steps of the on-line stage of the retrieval system can be improved in many ways. For example, a more efficient scheme can be developed for matching the query parts with the clusters’ representatives. The representatives can be further clustered and divided into groups of similar symbols, such that the number of the resulting groups is smaller than the number of the representatives. Then the query can be matched to the new fewer representatives and then matched only to the members inside the matching representative. This resembles a tree or a hierarchical representation of the clusters' representatives, and the purpose of such representations is to perform fewer matching operations. To improve the speed of query matching, low dimensional feature descriptors can be used to describe the query and the clusters' representatives, and fast nearest neighbor approaches can be used to find the matching clusters. Another possible improvement is to make the spatial verification step verify the spatial layout of the query part. There exist many methods in the literature for implementing this step efficiently.

The methods presented in Chapters 5, 6 and this chapter, are important steps along the way to provide means of indexing and retrieval of heterogeneous collections of line drawings without knowledge about the domain of the drawings. Focused symbol retrieval with such a precise localization of the queried symbols can also serve as means for the automatic interpretation of technical drawings.

Chapter 8

Conclusions

This dissertation has made a number of contributions towards the goal of fully automatic interpretation and mining of technical line drawings. Key contributions are: the use of branch-and-bound geometric matching algorithms, the use of statistical grouping algorithms, and the automatic creation of symbol indexes. Let us discuss these points in more detail.

The symbol recognition method presented in this dissertation performs recognition in context. Hence, the same method can be used for isolated symbol recognition as well as symbol spotting. The method uses the branch-and-bound geometric matching algorithm developed by Breuel in [48] as a recognition module. The use of geometric matching in a symbol recognition/spotting method has many advantages over other spotting methods. First, it deals well with large amounts of clutter. Second, it allows the use of different feature types such as pixels or line segments, and makes the recognition robust to partial or missing features. Third, it results in precise localization of the recognized symbols. Forth, it allows incorporation of contextual information such as feature weights and neighboring features directly into the matching. Finally, the geometric matching can be applied to all kinds of line drawings.

Based on the performance of the symbol recognition/spotting method presented in this dissertation, it can be concluded that using geometric matching with shape features is effective for recognition in line drawings in the presence of large amounts of contextual clutter. Additionally, the experiments show that dealing with noise at the preprocessing phase is probably better than dealing with it at the matching phase.

Such a geometric matching approach seems very suitable in dealing with textureless line drawings (whether simple shapes like isolated symbols or complicated shapes such as complete technical drawings). Being mainly composed of lines, the Line drawings can be easily vectorized and represented as geometric primitive features. The use of geometric matching approaches seems more effective for matching line patterns than the use of photometric feature descriptors with nearest neighbor approaches.

This dissertation also explored the use of machine learning techniques to improve the performance of symbol spotting. The idea of using geometric matching to automatically generate training data made the machine learning techniques applicable to spotting problems.

This dissertation has also presented methods for content analysis of technical line drawings. Content analysis is useful for content-based search in the line drawings. A key task in content analysis is to find regions of interest (ROIs) that represent symbols or parts of symbols. An ROI-finding method has been developed to accomplish this task. The method is based on statistically justified feature grouping, which shows high ability to capture the contents of line drawings consistently and accurately. A key advantage of this method is that, its found ROIs are meaningful parts of symbols up to complete symbols. The method can be used as a segmentation method for segmenting symbols from the background. The ROI-finding method is also useful for symbol spotting. Although the geometric matching-based spotting system discussed earlier does not require prior segmentation, it can benefit from segmentation information for getting improved accuracy and speed.

Another key task in content analysis is to build a compact representation of a database of line drawings. Such representation can be used for efficient large scale symbol retrieval. A symbol clustering method has been developed to accomplish this task. The use of geometric matching as a similarity measure in the clustering method results in the formation of highly accurate clusters. The clustering method is used as a basis for building a symbol index whose key entries are the clusters' representatives. The symbol index is a compact representation of the database.

Finally, an efficient symbol retrieval system has been presented in this dissertation. The system follows the successful bag of visual words (BoVW) model that is widely used in content-based image retrieval systems. In the symbol retrieval system, the symbol index is used as the outcome of the off-line analysis stage. In the on-line stage, the ROIs of a query symbol are found, then matched to the key entries of the symbol index via geometric matching. As a final step, spatial verification is performed on the retrieved parts to eliminate false matches. Following the BoVW model for retrieval of line drawings is made practical through the use of the content analysis methods described above. Those methods (the ROI-finding, geometric matching-based clustering) are suitable for images of textureless nature such as technical line drawings. The performance of the symbol retrieval system shows the effectiveness of these methods for the domain of architectural line drawings.

Overall, these contributions are major advancements in the research of graphics recognition. The hope is that, such contributions provide the basis for the development of reliable and accurate performing applications for browsing, querying or classification of line drawings for the benefit of end users.

Future Work

There are many directions to proceed in the work presented in this dissertation. In terms of algorithmic improvements, each method presented in this dissertation can be improved. One possible improvement is developing more efficient variants of the geometric matching algorithm while keeping the high performance in recognition and spotting. Preprocessing and feature extraction algorithms can be enhanced to be more robust to noise and degradation. As for the ROI-finding method, combinations of non-accidental properties can be used as basis for feature grouping. This helps in dealing with other types of line drawings that have symbols that consist mostly of non-convex parts. The symbol clustering method can be improved to form better clusters through the use of soft clustering.

More databases of different types of line drawings can be collected for more reliable performance evaluations. Larger symbol libraries can be used to help improve the scalability of the symbol recognition, spotting and retrieval methods.

In terms of applications, user friendly systems can be developed for different purposes: rebuilding the high-level design from an engineering drawing, recognizing cities, roads, rivers and regions on a map, converting a scanned electrical diagram into an electrical CAD system, etc.

The interpretation of hand-drawn or sketched line drawings can be of interest to the users. The research in the area of sketched drawings is still new, and there have not been many advances in satisfying the users needs in this area. Developing methods to deal with hand-drawn drawings may be worthwhile in the future.

Bibliography

[1] M. Rusiñol and J. Lladós, *Symbol Spotting in Digital Libraries: Focused Retrieval over Graphic-rich Document Collections*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[2] K. Tombre and B. Lamiroy, "Pattern recognition methods for querying and browsing technical documentation," in *Proceedings of the 13th Iberoamerican congress on Pattern Recognition: Progress in Pattern Recognition, Image Analysis and Applications*, ser. CIARP '08, 2008, pp. 504–518.

[3] P. Dosch, K. Tombre, C. Ah-Soon, and G. Masini, "A complete system for the analysis of architectural drawings," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 3, no. 2, pp. 102–116, 2000.

[4] L. Wenyin, W. Zhang, and L. Yan, "An interactive example-driven approach to graphics recognition in engineering drawings," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 9, no. 1, pp. 13–29, 2007.

[5] M. Rusiñol and J. Lladós, "Logo spotting by a bag-of-words approach for document categorization," in *10th Int. Conf. on Document Analysis and Recognition*, 2009, pp. 111–115.

[6] E. Hassan, S. Chaudhury, and M. Gopal, "Shape descriptor based document image indexing and symbol recognition," in *10th Int. Conf. on Document Analysis and Recognition*, 2009, pp. 206–210.

[7] S. Marinai, B. Miotti, and G. Soda, "Mathematical symbol indexing," in *AI*IA 2009: Emergent Perspectives in Artificial Intelligence*, 2009, pp. 102–111.

[8] ——, "Using earth mover's distance in the bag-of-visual-words model for mathematical symbol retrieval," in *Proceedings of the 2011 International Conference on Document Analysis and Recognition (ICDAR)*, 2011, pp. 1309–1313.

[9] M. Rusiñol and J. Lladós, "Word and symbol spotting using spatial organization of local descriptors," in *The 8th IAPR Int. Workshop on Document Analysis Systems*, 2008, pp. 489–496.

[10] M. Rusiñol, A. Borràs, and J. Lladós, "Relational indexing of vectorial primitives for symbol spotting in line-drawing images," *Pattern Recognition Letters*, vol. 31, no. 3, pp. 188–201, 2010.

[11] J. Lladós, E. Valveny, G. Sánchez, and E. Martí, "Symbol recognition: Current advances and perspectives," in *Selected Papers from the Fourth International Workshop on Graphics Recognition Algorithms and Applications*, ser. GREC'01, 2002, pp. 104–127.

[12] K. Tombre, "Graphics recognition: The last ten years and the next ten years," in *Graphics Recognition. Ten Years Review and Future Perspectives*, 2006, pp. 422–426.

[13] M. Delalandre, E. Valveny, and J. Lladós, "Performance evaluation of symbol recognition and spotting systems: An overview," in *Proceedings of the 2008 The Eighth IAPR International Workshop on Document Analysis Systems*, ser. DAS '08, 2008, pp. 497–505.

[14] P. Dosch and J. Lladós, "Vectorial signatures for symbol discrimination." in *Graphics Recognition. Recent Advances and Perspectives*, 2003, pp. 154–165.

[15] P. Dosch and E. Valveny, "Report on the second symbol recognition contest," in *Graphics Recognition Workshop (GREC)*, vol. 3926, 2005, pp. 381–397.

[16] E. Valveny, P. Dosch, A. Fornés, and S. Escalera, "Report on the third contest on symbol recognition," in *Graphics Recognition Workshop. Recent Advances and New Opportunities*, 2008, pp. 321–328.

[17] E. Valveny, M. Delalandre, R. Raveaux, and B. Lamiroy, "Report on the symbol recognition and spotting contest," in *Graphics Recognition Workshop (GREC'11)*, 2011, p. To appear in the selected papers in LNCS proceedings.

[18] L. P. Cordella and M. Vento, "Symbol and shape recognition," in *Graphics Recognition Recent Advances*, vol. 1941, 1999, pp. 167–182.

[19] A. K. Chhabra, "Graphic symbol recognition: An overview," in *Graphics Recognition Algorithms and Systems*, vol. 1389/1998, 1998, pp. 68–79.

[20] K. Tombre, "Analysis of engineering drawings: State of the art and challenges," in *Graphics Recognition Algorithms and Systems*, vol. 1389/1998, 1998, pp. 257–264.

[21] L. Wenyin, "On-line graphics recognition: State-of-the-art," in *Graphics Recognition Workshop*, vol. 3088/2004, 2004, pp. 291–304.

[22] M. Coustaty, S. Guillas, M. Visani, K. Bertet, and J. Ogier, "On the joint use of a structural signature and a galois lattice classifier for symbol recognition," in *Graphics Recognition Workshop (GREC)*, 2008, pp. 61–70.

[23] M. M. Luqman, T. Brouard, and J. Ramel, "Graphic symbol recognition using graph based signature and bayesian network classifier," in *Proceedings of the 2011 International Conference on Document Analysis and Recognition (ICDAR)*, 2009, pp. 1325–1329.

[24] W. Zhang and L. Wenyin, "A new vectorial signature for quick symbol indexing, filtering and recognition," in *Proceedings of the 2011 International Conference on Document Analysis and Recognition (ICDAR)*, 2007, pp. 536–540.

[25] F. Min, W. Zhang, and L. Wenyin, "Symbol recognition using bipartite transformation distance and angular distribution alignment," in *Graphics Recognition Workshop (GREC)*, vol. 3926, 2005, pp. 398–407.

[26] S. Yang, "Symbol recognition via statistical integration of pixel-level constraint histograms: A new descriptor," *Pattern Analysis and Machine Intelligence (PAMI)*, vol. 27, no. 2, pp. 278–281, 2005.

[27] W. Zhang, L. Wenyin, and K. Zhang, "Symbol recognition with kernel density matching," *Pattern Analysis and Machine Intelligence (PAMI)*, vol. 28, no. 12, pp. 2020–2024, 2006.

[28] F. Su, T. Lu, and R. Yang, "Symbol recognition combining vectorial and pixel-level features for line drawings," in *20th International Conference on Pattern Recognition (ICPR)*, 2010, pp. 1892–1895.

[29] M. Rusiñol, K. Bertet, J.-M. Ogier, and J. Lladós, "Symbol recognition using a concept lattice of graphical patterns," in *Proceedings of the 8th international conference on graphics recognition: achievements, challenges, and evolution*, ser. GREC'09, 2010, pp. 187–198.

[30] J. Zhang, W. Zhang, and L. Wenyin, "Adaptive noise reduction for engineering drawings based on primitives and noise assesment," in *Graphics Recognition Workshop (GREC)*, 2005, pp. 140–150.

[31] C.-M. Zhai and J.-X. Du, "Graphic symbol recognition of engineering drawings based on multi-scale autoconvolution transform," in *Proceedings of the 4th international symposium on Neural Networks: Part II–Advances in Neural Networks*, 2007, pp. 793–800.

[32] M. Rusiñol and J. Lladós, "A region-based hashing approach for symbol spotting in technical documents," in *Graphics Recognition. Recent Advances and New Opportunities*, 2008, vol. 5046, pp. 321–328.

[33] M. Rusiñol, J. Lladós, and G. Sánchez, "Symbol spotting in vectorized technical drawings through a lookup table of region strings," *Pattern Analysis and Applications*, vol. 13, no. 3, pp. 321–331, 2010.

[34] C. Ah-Soon and K. Tombre, "Architectural symbol recognition using a network of constraints," *Pattern Recognition Letters*, vol. 22, no. 2, pp. 231–248, 2001.

[35] R. J. Qureshi, J. Ramel, D. Barret, and H. Cardot, "Spotting symbols in line drawing images using graph representations," in *Graphics Recognition Workshop (GREC)*, 2007, pp. 91–103.

[36] R. J. Qureshi, J. Ramel, and H. Cardot, "Graphic symbol recognition using flexible matching of attributed relational graphs," in *6th IASTED International Conference on Visualization, Imaging, and Image Processing (VIIP), Palma de Mallorca-Spain*, 2006, pp. 383–388.

[37] J. Lladós and G. Sanchez, "Graph matching versus graph parsing in graphics recognition - a combined approach," *Int. Journal of Pattern Recognition and Artificial Intelligence*, vol. 18, no. 3, pp. 455–473, 2004.

[38] H. Locteau, S. Adam, E. Trupin, J. Labiche, and P. Heroux, "Symbol spotting using full visibility graph representation," in *Graphics Recognition Workshop (GREC)*, 2007, pp. 1–7.

[39] S. Tabbone, L. Wendling, and K. Tombre, "Matching of graphical symbols in line-drawing images using angular signature information," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 6, pp. 115–125, 2003.

[40] T. Nguyen, S. Tabbone, and A. Boucher, "A symbol spotting approach based on the vector model and a visual vocabulary," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2009, pp. 708–712.

[41] M. M. Luqman, T. Brouard, J. Ramel, and J. Llodos, "A content spotting system for line drawing graphic document images," in *International conference on pattern recognition (ICPR)*, 2010, pp. 3420–3423.

[42] X. Kong, E. Valveny, G. Snchez, and L. Wenyin, "Symbol spotting using deformed blurred shape modeling with component indexing and voting scheme," in *Graphics Recognition Workshop (GREC)*, 2011, p. online proceedings.

[43] M. Delalandre, T. Pridmore, E. Valveny, H. Locteau, and E. Trupin, "Building synthetic graphical documents for performance evaluation," in *Graphics Recognition. Recent Advances and New Opportunities*, 2008, pp. 288–298.

[44] M. Delalandre, E. Valveny, T. Pridmore, and D. Karatzas, "Generation of synthetic documents for performance evaluation of symbol recognition and spotting systems," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 13, no. 3, pp. 187–207, 2010.

[45] E. Valveny and P. Dosch, "Symbol recognition contest: A synthesis," in *Graphics Recognition. Recent Advances and Perspectives*, J. Llads and Y.-B. Kwon, Eds., 2004, pp. 368–385.

[46] M. Rusiñol and J. Lladós, "A performance evaluation protocol for symbol spotting systems in terms of recognition and location indices," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 12, no. 2, pp. 83–96, 2009.

[47] E. Valveny, P. Dosch, A. Winstanley, Y. Zhou, S. Yang, L. Yan, L. Wenyin, D. Elliman, M. Delalandre, E. Trupin, S. Adam, and J.-M. Ogier, "A general framework for the evaluation of symbol recognition methods," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 9, no. 1, pp. 59–74, 2007.

[48] T. M. Breuel, "Implementation techniques for geometric branch-and-bound matching methods," *Journal on computer vision and image understanding (CVIU)*, vol. 90, no. 3, pp. 258–294, 2003.

[49] N. Nayef and T. M. Breuel, "On the use of geometric matching for both: Isolated symbol recognition and symbol spotting," in *Graphics Recognition Workshop (GREC)*, 2011, p. To appear in the selected papers in LNCS proceedings.

[50] D. H. Ballard, "Generalizing the hough transform to detect arbitrary shapes," *Pattern Recognition Letters*, vol. 13, no. 2, pp. 111–122, 1981.

[51] T. M. Breuel, "Fast recognition using adaptive subdivisions of transformationspace," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1992, pp. 445–451.

[52] T. A. Cass, "Polynomial-time geometric matching for object recognition," *International Journal Computer Vision*, vol. 21, no. 1-2, pp. 37–61, 1997.

[53] D. Keysers, T. Deselaers, and T. M. Breuel, "Optimal geometric matching for patch-based object detection," *Electronic Letters on Computer Vision and Image Analysis*, vol. 6, no. 1, pp. 44–54, 2007.

[54] C.-S. Chen, Y.-P. Hung, and J.-B. Cheng, "Ransac-based darces: A new approach to fast automatic registration of partially overlapping range images," *Pattern Analysis and Machine Intelligence (PAMI)*, vol. 21, no. 11, pp. 1229–1234, 1999.

[55] K. Wang, Y. Yan, T. Shi, S. Liu, and Q. Xia, "Image registration based on geometric pattern matching," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 8335, 2012, pp. 83 351K–83 351K–7.

[56] T. M. Breuel, "Two geometric algorithms for layout analysis," in *Document Analysis Systems (DAS)*, 2002, pp. 188–199.

[57] J. Van Beusekom, F. Shafait, and T. M. Breuel, "Image-matching for revision detection in printed historical documents," in *DAGM conference on Pattern recognition*, 2007, pp. 507–516.

[58] F. Shafait, J. Van Beusekom, D. Keysers, and T. M. Breuel, "Page frame detection for marginal noise removal from scanned documents," in *Proceedings of the 15th Scandinavian conference on Image analysis (SCIA'07)*, 2007, pp. 651–660.

[59] D. Keysers and T. M. Breuel, "Optimal line and arc detection on run-length representations," in *Graphics Recognition Workshop (GREC)*, 2005, pp. 369–380.

[60] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[61] Y. Ke, R. Sukthankar, and L. Huston, "Efficient near-duplicate detection and subimage retrieval," in *ACM International Conference on Multimedia*, 2004, pp. 869–876.

[62] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[63] C. F. Olson, "A general method for geometric feature matching and model extraction," *International Journal Computer Vision*, vol. 45, no. 1, pp. 39–54, 2001.

[64] J. Illingworth and J. Kittler, "A survey of the hough transform," *Computer Vision, Graphics, and Image Processing*, vol. 44, no. 1, pp. 87–116, 1988.

[65] M. Atiquzzaman, "Multiresolution hough transform-an efficient method of detecting patterns in images," *Pattern Analysis and Machine Intelligence (PAMI)*, vol. 14, no. 11, pp. 1090–1095, 1992.

[66] R. Dahyot, "Statistical hough transform," *Pattern Analysis and Machine Intelligence (PAMI)*, vol. 31, no. 8, pp. 1502–1509, 2009.

[67] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[68] R. Raguram, J.-M. Frahm, and M. Pollefeys, “A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus,” in *Proceedings of the 10th European Conference on Computer Vision: Part II*, 2008, pp. 500–513.

[69] S. Choi, T. Kim, and W. Yu, “Performance evaluation of ransac family,” in *British Machine Vision Conference (BMVC)*, 2009, pp. 81.1–81.12.

[70] T. M. Breuel, “On the use of interval arithmetic in geometric branch and bound algorithms,” *Pattern Recognition Letters*, vol. 24, no. 9-10, pp. 1375–1384, 2003.

[71] ——, “A comparison of search strategies for geometric branch and bound algorithms,” in *7th European Conference on Computer Vision (ECCV)*, 2002, pp. 837–850.

[72] ——, “Finding lines under bounded error,” *Pattern Recognition Letters*, vol. 29, no. 1, pp. 167–178, 1996.

[73] A. Ulges, C. H. Lampert, D. Keysers, and T. M. Breuel, “Optimal dominant motion estimation using adaptive search of transformation space,” in *DAGM conference on Pattern recognition*, 2007, pp. 204–213.

[74] T. M. Breuel, “A practical, globally optimal algorithm for geometric matching under uncertainty,” *Electronic Notes on Theoretical Computer Science*, vol. 46, pp. 188–202, 2001.

[75] N. Nayef and T. M. Breuel, “Graphical symbol retrieval using a branch and bound algorithm,” in *International conference on image processing (ICIP)*, 2010, pp. 2153–2156.

[76] A. Wong and W. Bishop, “Robust invariant descriptor for symbol-based image recognition and retrieval,” in *IEEE Int. Conf. on Semantic Computing*, 2007, pp. 637–644.

[77] ——, “Robust hough-based symbol recognition using knowledge-based hierarchical neural networks,” in *Proceedings of the 2008 International Conference on Image Processing, Computer Vision, & Pattern Recognition, IPCV 2008*, 2008, pp. 3–9.

[78] P. Fränti, A. Mednonogov, V. Kyrki, and H. Kälviäinen, “Content-based matching of line-drawing images using the hough transform,” *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 3, no. 2, pp. 117–124, 2000.

[79] M. Vlachos, Z. Vagena, P. S. Yu, and V. Athitsos, “Rotation invariant indexing of shapes and line drawings,” in *Proceedings of the 14th ACM international conference on Information and knowledge management*, ser. CIKM ’05, 2005, pp. 131–138.

[80] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *Pattern Analysis and Machine Intelligence (PAMI)*, vol. 24, no. 4, pp. 509–522, 2002.

[81] F. Su, T. Lu, and R. Yang, "Symbol recognition by multiresolution shape context matching," in *Proceedings of the 2011 International Conference on Document Analysis and Recognition*, ser. ICDAR '11, 2011, pp. 1319–1323.

[82] T.-O. Nguyen, S. Tabbone, and O. R. Terrades, "Symbol descriptor based on shape context and vector model of information retrieval," in *Proceedings of the 2008 The Eighth IAPR International Workshop on Document Analysis Systems*, ser. DAS '08, 2008, pp. 191–197.

[83] T. Kanungo, R. M. Haralick, H. S. Baird, W. Stuetzle, and D. Madigan, "Document degradation models: Parameter estimation and model validation." in *Machine vision applications (MVA)*, 1994, pp. 552–557.

[84] N. Nayef, M. Z. Afzal, and T. M. Breuel, "Learning feature weights of symbols, with application to symbol spotting," in *International Conference on Pattern Recognition (ICPR)*, 2012, p. Accepted to appear in the proceedings.

[85] N. Nayef and T. M. Breuel, "Combining geometric matching with svm to improve symbol spotting," in *Document Recognition and Retrieval XX (DRR)*, 2013, p. accepted to appear in the proceedings.

[86] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 7, pp. 179–188, 1936.

[87] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*, 2nd ed. Wiley-Interscience, 2001.

[88] V. N. Vapnik, *The nature of statistical learning theory.* Springer-Verlag New York, Inc., 1995.

[89] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *The fifth annual workshop on Computational learning theory*, ser. COLT'92, 1992, pp. 144–152.

[90] M. M. Luqman, M. Delalandre, T. Brouard, J. Ramel, and J. Lladós, "Fuzzy intervals for designing structural signature: an application to graphic symbol recognition," in *Graphics recognition workshop (GREC'09)*, 2010, pp. 12–24.

[91] S. Barrat and S. Tabbone, "A bayesian network for combining descriptors: application to symbol recognition," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 13, no. 1, pp. 65–75, 2010.

[92] O. R. Terrades, E. Valveny, and S. Tabbone, “Optimal classifier fusion in a non-bayesian probabilistic framework,” *Pattern Analysis and Machine Intelligence (PAMI)*, vol. 31, no. 9, pp. 1630–1644, 2009.

[93] H. Yu, J. Oh, and W. Han, “Efficient feature weighting methods for ranking,” in *ACM - Information and knowledge management*, 2009, pp. 1157–1166.

[94] D. Wettschereck, D. W. Aha, and T. Mohri, “A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms,” *Artificial Intelligence Review*, vol. 11, pp. 273–314, 1997.

[95] S. Zafeiriou, A. Tefas, and I. Pitas, “The discriminant elastic graph matching algorithm applied to frontal face verification,” *Pattern Recognition Letters*, vol. 40, pp. 2798–2810, 2007.

[96] T. Kitahara, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno, “Instrument identification in polyphonic music: feature weighting with mixed sounds, pitch-dependent timbre modeling, and use of musical context,” in *Music Information Retrieval*, 2005, pp. 558–563.

[97] F. Song, D. Mei, and H. Li, “Feature selection based on linear discriminant analysis,” in *Intelligent System Design and Engineering Application (ISDEA)*, 2010, pp. 746–749.

[98] H. Byun and S.-W. Lee, “Applications of support vector machines for pattern recognition: A survey,” in *Proceedings of the First International Workshop on Pattern Recognition with Support Vector Machines*, ser. SVM’02, 2002, pp. 213–236.

[99] C.-C. Chang and C.-J. Lin, “Libsvm: A library for support vector machines,” *ACM Transactions Intelligent System Technologies*, vol. 2, no. 3, pp. 27:1–27:27, 2011.

[100] N. Nayef and T. M. Breuel, “Statistical grouping for segmenting symbols parts from line drawings, with application to symbol spotting,” in *Proceedings of the 2011 International Conference on Document Analysis and Recognition (ICDAR)*, 2011, pp. 364–368.

[101] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient graph-based image segmentation,” *International Journal Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[102] T. Tuytelaars and K. Mikolajczyk, *Local Invariant Feature Detectors: A Survey.* Hanover, MA, USA: Now Publishers Inc., 2008.

[103] D. G. Lowe, “Object recognition from local scale-invariant features,” in *Proceedings of the International Conference on Computer Vision*, 1999, pp. 1150–1157.

[104] S. Lazebnik, C. Schmid, and J. Ponce, "Semi-local affine parts for object recognition," in *British Machine Vision Conference*, 2004, pp. 959–968.

[105] D. Gokalp and S. Aksoy, "Scene classification using bag-of-regions representations," in *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2007, pp. 1–8.

[106] T. Tuytelaars and L. V. Gool, "Wide baseline stereo matching based on local, affinely invariant regions," in *British Machine Vision Conference*, 2000, pp. 412–425.

[107] J. Sivic, F. Schaffalitzky, and A. Zisserman, "Efficient object retrieval from videos," in *Proceedings of the 12th European Signal Processing Conference (EUSIPCO '04), Vienna, Austria*, 2004, pp. 1737–1740.

[108] T. Tuytelaars and L. J. V. Gool, "Content-based image retrieval based on local affinely invariant regions," in *Proceedings of the Third International Conference on Visual Information and Information Systems*, 1999, pp. 493–500.

[109] J. Sivic and A. Zisserman, "Video google: Efficient visual search of videos," in *Toward Category-Level Object Recognition*, ser. LNCS, J. Ponce, M. Hebert, C. Schmid, and A. Zisserman, Eds., vol. 4170. Springer, 2006, pp. 127–144.

[110] T. Dickscheid, F. Schindler, and W. Förstner, "Coding images with local features," *International Journal Computer Vision*, vol. 94, no. 2, pp. 154–174, 2011.

[111] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool, "A comparison of affine region detectors," *International Journal Computer Vision*, vol. 65, no. 1-2, pp. 43–72, 2005.

[112] T. Tuytelaars and L. Van Gool, "Wide baseline stereo matching based on local, affinely invariant regions," in *British Machine Vision Conference*, 2000, pp. 412–425.

[113] J. Matas, O. Chum, U. Martin, and T. Pajdla, "Robust wide baseline stereo from maximally stable extremal regions," in *Proceedings of British Machine Vision Conference*, 2002, pp. 384–393.

[114] K. Mikolajczyk and C. Schmid, "An affine invariant interest point detector," in *Proceedings of the 7th European Conference on Computer Vision-Part I*, 2002, pp. 128–142.

[115] ——, "Scale & affine invariant interest point detectors," *International Journal Computer Vision*, vol. 60, no. 1, pp. 63–86, 2004.

[116] T. Kadir, A. Zisserman, and M. Brady, "An affine invariant salient region detector," in *In Proceedings of the 8th European Conference on Computer Vision*, 2004, pp. 345–457.

[117] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *Computer Vision, IEEE International Conference on*, 2011, pp. 858–865.

[118] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, "Gradient response maps for real-time detection of textureless objects," *Pattern Analysis and Machine Intelligence (PAMI)*, vol. 34, no. 5, pp. 876–888, 2012.

[119] K. Koffka, *Principles of Gestalt Psychology.* Harcourt, Brace and company, New York: Springer Series in Statistics, 1935.

[120] B. Dubuc and S. W. Zucker, "Complexity, confusion, and perceptual grouping. part i: The curve-like representation," *Journal of Mathematical Imaging and Vision*, vol. 15, no. 1-2, pp. 55–82, 2001.

[121] D. G. Lowe, *Perceptual Organization and Visual Recognition.* Norwell, MA, USA: Kluwer Academic Publishers, 1985.

[122] D. W. Jacobs, "Grouping for recognition," Cambridge, MA, USA, Tech. Rep., 1989.

[123] D. J. Lowe, "Three-dimensional object recognition from single two-dimensional images," *Artificial Intelligence*, vol. 31, no. 3, pp. 355–395, 1987.

[124] D. W. Jacobs, "Robust and efficient detection of salient convex groups," *Pattern Analysis and Machine Intelligence (PAMI)*, vol. 18, no. 1, pp. 23–37, 1996.

[125] N. Nayef and T. M. Breuel, "Building a symbol library from technical drawings by identifying repeating patterns," in *Graphics Recognition Workshop (GREC)*, 2011, p. To appear in the selected papers in LNCS proceedings.

[126] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on Statistical Learning in Computer Vision (ECCV)*, 2004, pp. 1–22.

[127] S. P. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, pp. 129–137, 1982.

[128] T. Honkela, T. Kohonen, "Kohonen network," *Scholarpedia*, vol. 2, no. 1, p. 1568., 2007.

[129] A. Gersho and R. M. Gray, *Vector quantization and signal compression.* Norwell, MA, USA: Kluwer Academic Publishers, 1991.

[130] Y. Linde, A. Buzo, and R. Gray, "An Algorithm for Vector Quantizer Design," *Communications, IEEE Transactions on*, vol. 28, no. 1, pp. 84–95, 1980.

[131] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: a review,” *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[132] A. K. Jain and R. C. Dubes, *Algorithms for clustering data.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988.

[133] V. S. Cherkassky and F. Mulier, *Learning from Data: Concepts, Theory, and Methods*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1998.

[134] F. Jurie and B. Triggs, “Creating efficient codebooks for visual recognition,” in *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV’05)*, 2005, pp. 604–610.

[135] T. Pavlidis and S. L. Horowitz, “Segmentation of plane curves,” *IEEE Transactions on Computers*, vol. 23, no. 8, pp. 860–870, 1974.

[136] J. Zobel and A. Moffat, “Inverted files for text search engines,” *ACM Computing Survey*, vol. 38, no. 2, 2006.

[137] R. A. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[138] J. Philbinand, O. Chum, M. Isard, J. Sivic, and A. Zisserman, “Object retrieval with large vocabularies and fast spatial matching,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007, pp. 1–8.

[139] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, “On clustering validation techniques,” *Journal of Intelligent Information Systems*, vol. 17, no. 2-3, pp. 107–145, 2001.

[140] R. Xu and D. C. W. II, “Survey of clustering algorithms,” *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645–678, 2005.

[141] N. Nayef and T. M. Breuel, “Efficient symbol retrieval by building a symbol index from a collection of line drawings,” in *Document Recognition and Retrieval XX (DRR)*, 2013, p. accepted to appear in the proceedings.

[142] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman, “Total recall: Automatic query expansion with a generative feature model for object retrieval,” in *Proceedings of the 11th International Conference on Computer Vision*, 2007, pp. 1–8.

[143] Y. Zhang, Z. Jia, and T. Chen, “Image retrieval with geometry-preserving visual phrases,” in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 809–816.

[144] Y. Zhang, L. Wang, R. Hartley, and H. Li, “Handling significant scale difference for object retrieval in a supermarket,” in *Proceedings of the 2009 Digital Image Computing: Techniques and Applications (DICTA ’09)*, 2009, pp. 468–475.

Short Curriculum Vitae

Name: Nibal Nayef
Website: http://nayef.iupr.com/

Education and Work Experience

Oct. 2008 - Dec. 2012
Ph.D. in computer science
Image Understanding and Pattern Recognition Group (IUPR)
Technical University of Kaiserslautern (TU KL) - Germany

Sept. 2007 - July 2008
Instructor - Information Technology Department
University College of Applied Science (UCAS) - Gaza

May 2007 - Aug. 2007
Internship (software development)
Company of Altariq Systems and Projects - Gaza

Oct. 2004 - Jan. 2007
M.Sc. in computer engineering
Jordanian University of Science and Technology (JUST) - Jordan

Sept. 2003 - Sept. 2004
Teaching and Research Assistant - Computer Engineering Department
Islamic University of Gaza (IUG) - Gaza

Sept. 1998 - June 2003
B.Sc. in computer engineering
Islamic University of Gaza (IUG) - Palestine

Hobbies and Interests

Hiking, Traveling, Reading, Movies, Photography, Learning foreign languages.