Automatic recognition of electrical and architectural symbols using trainable COSFIRE filters

Student : Luke Agius

May 2013

Supervisor : George Azzopardi

University of London
BSc Creative Computing (Hons)

Submitted as part of the requirements for the award of the Degree in Creative Computing of the University of London.

Acknowledgements

I would like to give my thanks to many people who, in one way or an other, have helped me through the completion of this degree and ultimately this dissertation.

I would like to first and foremost express my gratitude and appreciation to my supervisor, Dr. George Azzopardi, whose expertise, understanding, and patience, added considerably to my graduate experience. I appreciate his vast knowledge and skill in many areas and his constant assistance and advice in writing this report whilst allowing me the room to work in my own way.

Special thanks go to my parents for their constant support, motivation, encouragement through out my studies. Without their help, patience and perseverance I would not be where I am today.

I am also thankful for and would like to acknowledge many others who helped me along the way: fellow students for going through this academic journey together, colleagues for sharing ideas and my friends for their patience.

Contents

List of Figures

List of Tables

1. Summary

Handheld devices have become ubiquitous due to their flexibility and enhanced productivity. Such devices allow users to write freely on them or to manually sketch symbols. The automatic recognition of handwriting and hand-drawn sketches is important to convert the manual input to a digital representation. Examples of applications include recognition of handwritten letters, digits, signatures, musical notes, electrical and architectural symbols, among others. The objective of this dissertation is to investigate and determine the effectiveness of the COSFIRE filters against isolated symbol recognition with or without degradation applied to the images in question.

The approach used, in order to tackle this problem is based upon the work of G.Azzopardi and N.Petkov where a trainable filter, called COSFIRE, inspired by shape-selective neurons in area v4 of the visual cortex. Previously this filter was used successfully in the detection of vascular bifurcations, recognition of handwritten digits and detection and recognition of traffic signs.

A thorough literature review is presented which focuses on state-of-the-art symbol recognition methods and covers progress in symbol recognition. Several algorithms are presented along with an explanation on how they work. An introduction about COSFIRE filters is then given to put the reader in context. The COSFIRE filter is then described in detail. This includes the COSFIRE filters' configuration and application stages as well. Also a detailed description is given on how COSFIRE filters are used in the classifier built for this project.

Experimental results are then presented which reflect experiments executed against the first three categories of datasets, from the GREC'11 contests, which are publicly available. The following table lists the recognition rates achieved. Results are then interpreted in a discussion which will point out the strengths and weakness of the taken approach. Finally we draw conclusions regarding the effectiveness of COSFIRE filters in isolated symbol recognition.

The proposed approach is effective for the automatic recognition of architectural and electrical symbols. It is robust to scalability, noise and geometrical transformations. In most cases it outperforms the state of the art methods for the GREC datasets. In particular, the results that we achieve for the datasets characterised with different levels of degradation and noise are the best ever reported.

2. Introduction

2.1 Background

Handheld devices have become ubiquitous due to their flexibility and enhanced productivity. Such devices allow users to write freely on them or to manually sketch symbols. Various applications to such devices include the automatic recognition of handwritten letters, digits, signatures, musical notes, electrical and architectural symbols, among others. For instance a musical composer might write down a piece of music on paper or an electrical engineer might also quickly jot down parts of a schematic. Manually converting this handwritten/sketched information into a digital representation may be tedious.

Such devices enable the composer or the engineer to quickly and efficiently convert their ideas into a correct digital representation. After doing so both the composer and the engineer might be able to quickly incorporate their ideas into a larger, already digitally converted, body of work. For instance, electrical or architectural engineers can quickly convert an image of hand drawn/sketched schematics into a set of symbols and then import them into drafting software such as Auto Cad.

Before the recognition process takes place, the document or image in which the manual work is found needs to be segmented. Image segmentation is applied where each symbol is spotted and isolated. This simplifies the recognition process by making the symbol that we need to recognise and classify, more meaningful and easier to analyse.

The automatic recognition of isolated symbols, from larger segmented hand-drawn sketches, is an important step in such applications due to the fact that each symbol in physical works need to be properly recognised and converted into a digital representation. This results in a more convenient and efficient storage, retrieval and manipulation as compared to conventional means which consists of starting the sketch from scratch on a computer.

A method which has been found to be highly effective for the recognition of isolated patterns is the trainable COSFIRE (Combination Of Shifted FIlter REsponses) approach [22]. COSFIRE filters, which are effective for keypoint detection and pattern recognition, is trainable as it can be configured by a given contour-based pattern. The configuration process automatically analyses the dominant orientations around the specified prototype. These dominant orientations are detected by the use of Gabor filters. The response of a COSFIRE filter is computed as the weighted geometric mean of the involved Gabor filter responses. This means that a response is only achieved when all the concerned contour parts are present.The COSFIRE approach can also achieve invariance to rotation, scale, reflection and contrast inversion.

In their paper [22], the authors demonstrated that a shape descriptor can be formed with the collective responses of multiple COSFIRE filters. They have also demonstrated that COSFIRE filters can be effectively applied to the detection of vascular bifurcations, recognition of handwritten digits and detection and recognition of traffic signs in complex scenes. In this work, we investigate the effectiveness of the mentioned COSFIRE filters for the recognition of electrical and architectural symbols.

2.2 Research Questions

The following are the questions that we investigate in this work:-

1. How effective are COSFIRE filters for the recognition of electrical and architectural symbols?

2. How robust are COSFIRE filters to noisy symbols?

3. How does the results achieved by COSFIRE filters compare to published result of state-of-the-art methods?

The effectiveness of COSFIRE filters is evaluated by numerous experiments which are performed on various publicly available data sets [3] [8]. These data sets hold 150 different classes of symbols. Every data set contains a model symbol for each class and many test images that we use for evaluation. Test symbols, are then used to test the configured COSFIRE filters. The test symbols contain both noisy and noise less symbols. Furthermore, some test symbols are scaled or/and rotated.

2.3 Deliverables

In this thesis, the deliverables include a description of the proposed method and a comparison against other state of the art techniques. The three posed research questions will be addressed by various experiments implemented on Matlab followed by in depth analysis of the results. Performance measurements will be computed in the form of true positives and false positive rates which can be used to derive accuracy, precision and recall rates. Documentation of the MATLAB implementation is provided on how it is built, Section A.1. This also includes a step by step guide on how to re-run the experiments, Section A.3. The needed files are included in the CD, Section A.2. A discussion is included concerning some aspects of the proposed approach which highlights the differences that distinguish it from other approaches. Finally, we draw conclusions and provide an outlook of future work.

2.4 Project Plan

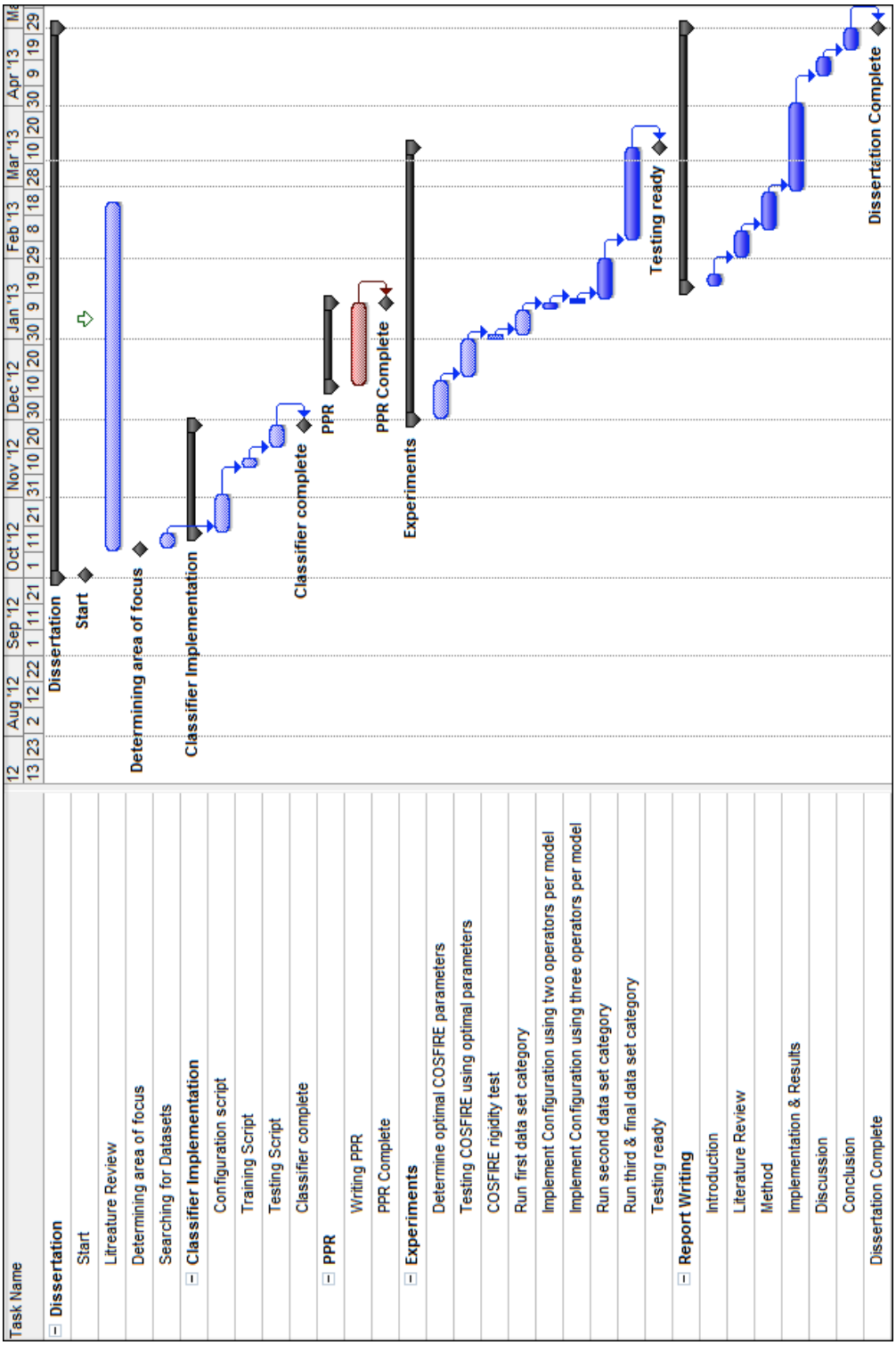


Figure 2.1: Project Plan

Fig. 2.1 shows the project plan for this thesis. The actual project plan deviated a little from the original one in terms of duration for certain tasks. Tasks other than the experiments required less time such as searching for adequate datasets, the building of the classifier and writing the preliminary report. The deviation in terms of task duration took place due to the fact that the some experiments required more time than actually planned because of the size of the data set to be processed and the number of computations done for each data set.

Therefore, due to time restrictions and having datasets with a larger number of classes available, which require more time to process, not all datasets where classified within the allotted time frame. This is explained further in Chapter 6 Discussion.

2.5 Overview of The Report

Below is a brief outline for each chapter of this dissertation.

Chapter 2 - Literature Review

The literature review focuses on state-of-the-art symbol recognition methods and presents an overview of several published algorithms used and results obtained. The research covers the progress in this area of research and discusses potential future work.

Chapter 3 - Methodology

This chapter is divided into two main sections. The first section gives a detailed description of COSFIRE filters. It explains the COSFIRE filters' configuration and application stages in detail.

The second section is the Evaluation section, which gives a detailed description

of how COSFIRE filters are used in the classification of electrical and architectural symbols. This section is further divided into five sections were, the data sets, pre-processing, configuration of COSFIRE filters, forming a shape descriptor and the classification technique used are explained in detail.

Chapter 4 - Experimental Results

This section provides the results obtained for various data sets of different complexity.

Chapter 5 - Discussion

This section provides a discussion about the proposed method to the recognition of isolated symbols in hand drawn images. We also compare our results with those obtained by other state of the art methods.

Chapter 6 - Summary Conclusion

This chapter provides a summary of the entire dissertation. Subsequently conclusions are drawn.

3. Literature Review

3.1 Overview

With the widespread popularity of portable digital input devices like smartphones and tablets, there is increasing interest in the development of systems that can automatically interpret and convert paper based graphical information into an equivalent digital representation. This conversion to digital representations has been a topic of active research throughout the last decade [14] [20].

A fundamental way in which we communicate information, especially technical information, is through the use of diagrams [41]. Much of this effort is focused on the automatic conversion of manually sketched work / diagrams to a digital format which can be understood by CAD systems in order to be included into other larger bodies of work.

A key issue, at the heart of such research efforts, is the recognition of domain dependant symbols using domain-independent techniques. Authors commonly develop ad-hoc techniques which are difficult to reuse in other domains [35]. Various approaches have been proposed to solve this issue, however challenges remain in terms of finding a general and efficient method which can be applied across domains with well establish performance and recognition accuracy.

In this literature review we direct our focus on the recognition of symbols from the electrical engineering and architectural domains. The main goal is to explain the process of symbol recognition in detail and then to highlight the different ap-

proaches adopted to-date to solve the problem of symbol recognition.

3.2 What Is a Symbol?

The oxford dictionary defines the word 'symbol' as a mark or character which is used as a conventional representation of an object, function or process. In a very general way, a symbol can be defined as a contextually meaningful graphical shape of a specific application domain [35]. Depending on the application, we can find various kinds of symbols in different industry fields like architecture, cartography, electronics, engineering etc. Each field use domain-dependant notations to construct their symbol designs.

In technical fields such as in architecture and engineering, symbols are designed in 2-dimensions (2D) using domain dependant notation [35].Simple symbols can be binary, made out of sets of line segments, both straight and in loops, such as electrical or architectural symbols , Fig. 3.2a. Symbols can also be complex, involving different levels of colours and shades, such as company logos [14], Fig. 3.1b.

(a) (b)

Figure 3.1: (a) An example of a contour based symbol and (b) An example of a complex symbol

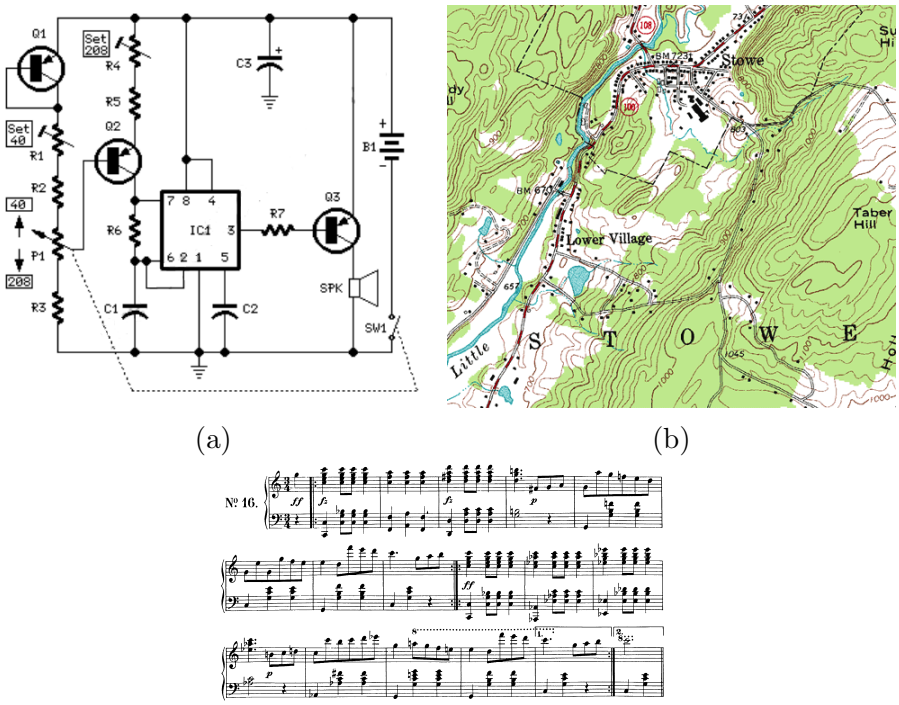
From a symbol recognition stand point, documents of interest which mostly hold the most symbols can be grouped into three main categories [14] :-

1. Technical drawings, such as flow diagrams and electrical schematics, Fig 3.2a

2. Maps, topographical or geographical, Fig 3.2b

3. Others, such as musical glyphs, Fig 3.2c

Symbols in such documents are formed depending on various domain dependent rules[14]. Furthermore, when presented, they can contain added noise, be deformed, incomplete, and/or geometrically transformed. Symbols may also appear in isolation or in complex backgrounds. They may be in close proximity, touching or partially overlap each other.



(a) (b)

(c)

Figure 3.2: Different types of documents in which symbols are found such as (a) Electric schematics[1], (b) topographical map[2], (c) musical scores[3].

[1]Taken From `http://www.redcircuits.com/Page8.htm`
[2]Taken From `http://en.wikipedia.org/wiki/File:Topographic_map_example.png`
[3]Taken From `http://music.unt.edu/mhte/theory/TF`

3.3 Symbol Recognition

Information is shared in many forms. It is often represented by textual or graphical means. At the core of human learning, lies the ability of pattern recognition and inference. This ability helps us to process information in any form it is presented, be it textual, graphical or otherwise.

In Artificial Intelligence (AI) research a constant effort is being put forward in order to model and automate these abilities by mechanical means [19]. The successful modelling of such abilities yield various advantages. As described in Section 2.1, such advantages include the automatic conversion of paper-based information into a equivalently meaningful digital version. This results in more convenient, efficient retrieval and manipulation of the information in question. In various fields, such as architecture, electronics, cartography, etc, information is conveyed in a diagrammatic form [35]. Symbols constitute an important informative source in diagrams [17] and depending on the field and the application being used, different symbols can be found. These symbols can be designed according to domain dependent rules or none at all.

This problem is tackled through symbol recognition. Symbol recognition is a field within pattern recognition area to which a lot of research effort has been devoted [43]. It is the automatic process where an input image of unknown pattern is classified to a particular class in a particular domain. [35] [36]. This process is then used in the conversion of paper based graphical symbols into equivalent digital representations.

The process of symbol recognition can be divided into two main parts namely segmentation and recognition. The first part, segmentation of the document or image, is a prerequisite to the second part, recognition. Segmentation isolates the symbols from the document or image, for an easier and computationally lighter

recognition process.

Combining segmentation and recognition into one system is one of the main problems within the field of symbol recognition. This problem is known as the "segmentation/recognition paradigm" [18]. This problem describes that a document or image should be segmented before recognition can take place. However, at the same time, some kind of recognition needs to be applied for segmentation to take place [48].

An other pre-requisite is that a system needs some priori knowledge about the symbols it needs to recognise in order to successfully recognise symbols [18]. This corresponds to learning a database of model symbols for training purposes. Various shape recognition methods have been applied for this training phase.

Symbol recognition can be classified into two main categories, statistical and structural. Structural symbol recognition is based on the organisation of primitives, such as straight lines and arcs, into higher defined structures such as symbols or signatures [13]. On the other hand statistical symbol recognition is based on representing the symbol in simpler forms by the use of a shape descriptor [35]. The following sub sections explain those two categories in detail.

3.3.1 Statistical Symbol Recognition

In a statistical approach towards symbol recognition, each symbol is represented by a unique signature. This signature is in the form of a feature vector of n dimensions extracted from the same symbol. This approach has two important issues. The first issue is the selection of features for every pattern and the second one is the partitioning of the feature space [35]. In this case, the feature space refers to the set of model symbols available.

The feature vector for every symbol is constructed depending on the properties of the inputted pattern. The chosen set of features for each symbol must be unique. This is critical to attain high discrimination power thus ultimately avoiding confusion between symbols in the classification process [35].

There are various methods with which a set of features is extracted from a pattern or symbols such as: centroids, line intersections, projection profiles, etc, amongst many [35]. These methods adopt an invariant approach towards added noise, geometrical image transformations and distortions [35].Once that a set of features is chosen for each symbol, classification consists in choosing a segmentation method in order to partition the feature space. This is done on order to assign each feature vector to one of the available classes.

One of the simplest feature spaces is the image space it self, where the feature vector is composed of features each corresponding to a pixel with in the image. When an image is used as a feature space it is usually normalised to a certain size [35]. Although this is one of the simplest feature spaces to process it has a number of advantages. It is simpler to process due to its low complexity, moreover it corresponds directly to the input information's visual appearance. However, it also has a number of disadvantages. This feature space does not adhere to an intravariant approach, therefore it does not take into consideration image distortion or geometrical transformations. Furthermore, it is very sensitive to noise.

The simplest way to partition a feature space consists in defining a distance function among the available feature vectors. This can be done by using the k-nearest neighbour algorithm (KNN). Which distance function to use in the Knn algorithm, depends on the application. One of the most popularly used distance functions is the Euclidean distance. The KNN algorithm is based on the concept of similarity [35]. It takes into account the nearest K number of neighbours which

can weight in, in the classification of the presented pattern. Other methods have also been used for the classification of unknown patterns which are also based on the concept of similarity such as neural networks [47] and decision trees [49].

Artificial neural networks (ANNs) have been proven to be adequate for classification purposes often achieving good recognition rates across various domains. One of the advantages in ANNs is their learning capabilities. They adapt quickly to the properties of the provided training set [35].

Decision trees for the purpose of symbol recognition are constructed in such a way that each of node in the tree corresponds to a particular feature. Classification is then carried out by visiting the nodes which satisfy a condition with the inputted image. This approach follows the tree branches which ultimately lead to a leave node which corresponds to a recognised symbol [35].

The main goal is to minimise, as much as possible, the distance between the input pattern and the pattern class it belongs to. At the same time, the distance between the same input pattern and other pattern classes is maximised [35].

3.3.2 Structural Symbol Recognition

In the structural approach towards symbol recognition, symbols are represented by a description of their shape in terms of simpler parts and the relationship between them [35]. These parts are geometric primitives [27] which usually consist of straight lines and arcs. However other geometric primitives have been also used such as loops, contours or simple shapes such as circles, rectangles etc.

Since the symbol is described by the geometric primitives it contains, a previous vectorization step is required. Vectorization is the process by which a raster image is converted into a set of geometrical primitives described above. Vec-

torization of raster images introduces both noise and distortion in the symbol's representation[35].

As cited in [35] there is a large body of work about structural approaches focusing on symbol graph representation [24, 25, 28, 31, 32, 38, 34]. When representing a symbol with a graph, each node and edge correspond to points and lines within the image. This provides a natural and intuitive symbol description. Classification for these methods consist in finding the best matching subgraphs between the input symbol and the models of the symbols. One of the main drawbacks for this method is its computational complexity. However some ways of reducing computation have been explored [35].

An other method of symbol representation in various structural approaches is by defining formal grammars. Graph grammars are the most common because of bi-dimensional structure of symbols [35]. A grammar stores in a very compact manner all valid instances of a symbol or a class of symbols. Correct classification then consists of parsing its representation in order to test whether it can be generated by defined grammar. Grammars are useful in applications where the shape of the symbols can be accurately defined by a set of rules, for example technical drawings.

An other method which has also been used in structural approaches is the use of Hidden Markov Models (HMMs). HMMs is a powerful statistical tool for modelling generative sequences [2]. HMMs are used since the structure of the symbol can be described as a sequence of states which generate the image. Recognition then consists in finding the sequence of states with the higher probability.

3.3.3 Performance Evaluation

During the last decade, attention has been shifted in the development of a generic and standard framework which permits the comparison and performance evaluation

of symbol recognition methods. This effort has resulted in various contests such as the GREC contests taking place which focus on the recognition of isolated symbols and thus creating a framework for evaluation reasons [20].

Datasets

For the evaluation and comparison of the methods used in these contests, publicly available data sets of isolated symbols are generated [3] [8]. These datasets are generated with a set of goals in mind [3]:

1. To provide a set of tests that could evaluate scalability of methods.
2. To be able to test the performance of methods under some realistic increasing depredations.
3. To be able to test the generalisation ability of the methods.

The generated data sets are organised into four categories which have different levels of complexity due to noise, deformation, rotation and scaling. The first category consists of 24 data sets. Each data set contains a number (varying between 25 and 150) of different electrical and architectural symbols each represented by a single image. These data sets contain 1000 test images of deformed symbols of increasing complexity. For instance, the least complex data set contains images that are only slightly deformed, while the most complex data set consists of images of symbols with the highest deformation.

The second category consists of 3 datasets each having 150 different symbols. These datasets contain test symbols with geometrical transformation applied such as scaling, rotation or combined. The third category of datasets consists of 3 datasets each comprising of 150 different models as well with three different kinds of noise applied separately.

The method used to apply noise to the test images was proposed by Kanungo [30]. The remaining data sets contain images of geometrically transformed symbols, such as different orientations and scale. Fig. 3.3 shows a sample of the test images with noise applied.

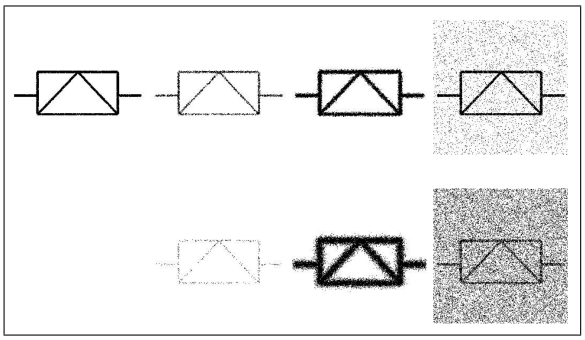


Figure 3.3: A sample set of test images for the model, found at the top left corner of the image, with noise applied according to Kanungo's method [30]. Adapted from [3].

The last, fourth, category contains three data sets. They contain large number of different model symbols and large number of test images that combine different degradation levels and different transformations [20].

Performance Metrics

For the symbol recognition contest the chosen performance metric was the recognition rate [20]. The recognition rate is determined by counting the true positives (TP) and the false positives (FP). The recognition rate is then calculated as follows, where R is the recognition rate in percentage.

$$R = \left(\frac{1}{(TP + FP)} \times TP \right) \times 100 \tag{3.1}$$

Results

The authors report that only one participant submitted a method which approached both symbol spotting and recognition [40]. Amongst the global results reported, the following results are the ones attained by testing the second and third category

of datasets. For each kind of deformation and noise a different result was attained.

Table 3.1: Contest's results for the second category of datasets.

Degradation	Recognition Rate
Rotation	81.07 %
Scaling	89.20 %
RotationScaling	84.27 %

Table 3.2: Contest's results for the third category of datasets.

Degradation	Recognition Rate
Noise A	88.07 %
Noise B	85.73 %
Noise E	85.67 %

The authors think that the datasets used in their work, can serve future similar contests as a stable platform against which evaluation and comparison of symbol recognition methods can be done. Future editions will hold hand drawn symbols [20].

3.4 State of The Art

Shape descriptors have been used in a variety of applications and tasks such as shape analysis, comparison and retrieval [26]. In symbol recognition, as previously described, shape descriptors are an integral part of the process. The goal of the shape descriptor is to represent a pattern by a unique structure which relies on the extraction of local or global features.

As observed in previous studies [10], shape descriptors can be either global or local. Global shape descriptors take into consideration the entire shape of the object while local shape descriptors are used for local shape features [44].Shape descriptors which are either global or local may fail to be robust. Global descriptors

are reported to be robust to local deformations, but fail when trying to capture local shape details. At the same time local descriptors are adequate in representing local shape features, however they are very sensitive to noise [44].

One of the most important challenges in shape descriptors is to come up with a solution by defining a rich shape descriptor, which consists of both Global and local properties. This solution needs to be computationally fast and efficient, compact, and most importantly, general enough to be used in a variety of applications.

The features that a shape descriptor detects represent information about specific structures within the image or pattern itself. This information ranges from simple geometric primitives such as point or edges to much more complex ones such as objects or a set of primitives. The extraction of such features is a very critical problem in computer vision and is used widely in various vision related tasks such as object recognition and tracking [37]. Due to their importance, a lot of research attention has been dedicated to it [26, 37].

Depending on the application an intravariant approach towards feature extraction becomes important. One of the most important jobs of a shape descriptor is to extract unique features which are both repeatable as well as invariant to various conditions such as noise or distortion. Apart from this a good shape descriptor should be informative, discriminative and efficient in matching process [44]. The rest of this section reviews a number of feature descriptors applied to date.

A very popular shape descriptor is the Shape Context approach. It is feature descriptor which was proposed by S. Belongie and J. Malik in 2000 [12]. Given a collection of points within an image which have been extracted from a set of detected edge elements, the shape context descriptor captures the relative distribution of points in the plane relative to each point on the shape [5].

A very popular shape descriptor is the Shape Context approach. It is feature descriptor which was proposed by S. Belongie and J. Malik in 2000 [12]. Given a collection of points within an image which have been extracted from a set of detected edge elements, the shape context descriptor "captures the relative distribution of points in the plane relative to each point on the shape" [5].

The shape context approach operates on the concept of similarity where it is measured between shapes and exploited for pattern recognition purposes. For example, when taking into account two written digits, Fig 3.4, interpretation can differ. When taken as a set of pixel with various brightness values, the two digits are very different. However when observed by a human, they appear to be very similar.

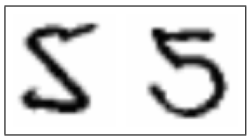


Figure 3.4: Two written digits can be interpreted differently depending who the observer is. Adapted from [12].

The aim of the authors is to put into operation the concept of similarity for object classification. This approach is described as a three stage process [12]:

1. "Solve the correspondence problem between the two shapes"
2. "Use the correspondences to estimate an aligning transform"
3. "Compute the distance between the two shapes as a sum of matching errors between corresponding points, together with a term measuring the magnitude of the aligning transformation"

At the core of the similarity concept that the authors use in their work is the fact that it has been observed that shapes which are related but not identical, can often be deformed to a point where they are aligned using simple coordinate transformations, Fig. 3.5. Therefore, based on this the authors have come up with an

algorithm which finds correspondences between similar shapes.

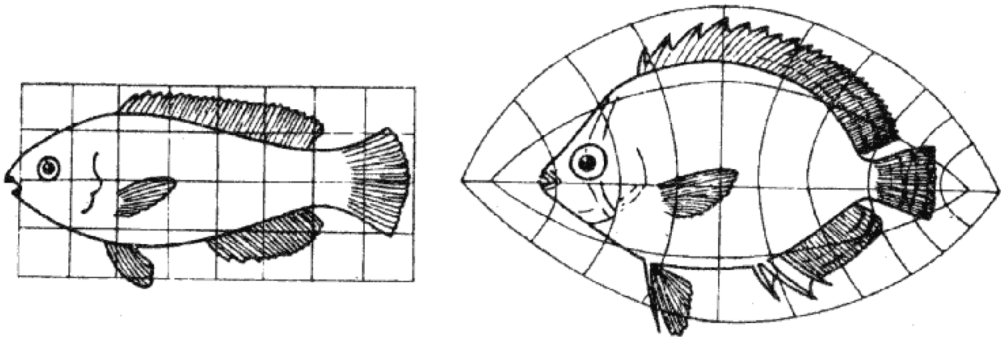
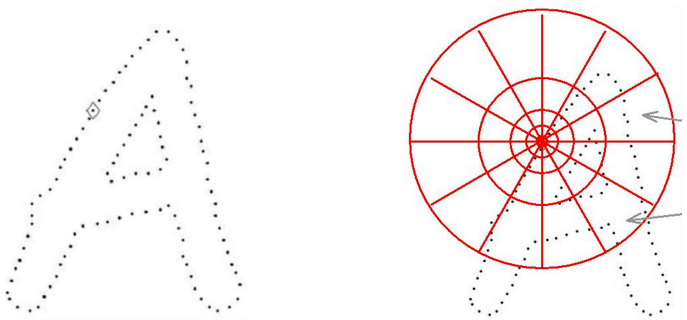


Figure 3.5: An example of coordinate transformation between two similar objects. Adapted from [12].

In this approach, shapes are represented by a collection of points. This collection of n number of points is sampled from the shape's contours by using an edge detector. When these points are taken into consideration on their own, they hold no particular importance.



(a) (b)

Figure 3.6: (a) A point is selected from a collection of points representing the shape. (b) Using a diagram of log-polar histogram bins we compute the point distribution and thus the shape contexts. Adapted from [12].

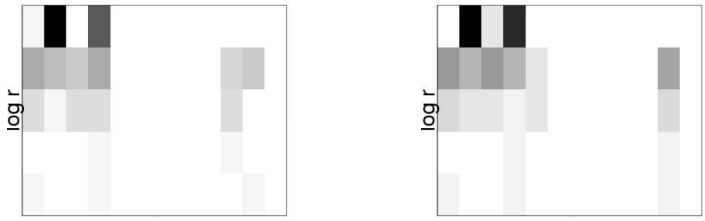
The shape context descriptor then describes the distribution of the rest of the shape relative to a selected point by placing a diagram of log-polar histogram bins on the selected point, Fig. 3.6. The histogram is divided into a number of bins.

From each of these bins the number of points is counted. This will describe the spatial arrangement of point around the selected point. When the selected shape needs to be matched to an other shape, the same procedure is applied to the other shape. Once this is done the arrangement attained from one shape is compared to the other one. Fig. 3.7 shows two different shapes of similar shape. From each a point has been chosen to check for correspondence between the two shapes.

(a) (b)

Figure 3.7: (a) A Point is selected from a shape to be compared to (b) an other point from a similar shape. Adapted from [12].

For each point a shape context is calculated as a log-polar histogram of the coordinates of the rest of the collection of points using the selected point as the origin, Fig. 3.8.



(a) (b)

Figure 3.8: (a) Shows the shape context response for the point in Fig.3.7a. While (b) shows the shape context response for the point in Fig.3.7b. Adapted from [12].

After the responses have been attained, the correspondences are found. This

means that for each sample point on the first shape, a sample point of the most similar context is found on the second shape. The correspondence is then extended by an estimation of transformation alignment that maps one shape onto the other, Fig. 3.9.

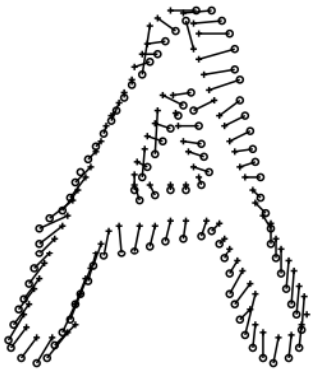


Figure 3.9: The difference between the two shapes. Adapted from [12].

Once this has been attained, and the similarity can be measured, a nearest neighbour technique can be used for recognition. Object recognition using shape context as a descriptor has been used in a variety of applications. The authors present results on the MNIST dataset of handwritten digits, silhouettes, line drawings, and CAPTCHA images [12].

Another recently proposed approach to build a shape descriptor is by the use of height functions, proposed in [44]. In this approach each object is represented by a contour. This contour is made up of a fixed finite number of sample points. For each of those sample points, a height function is defined based on the distances of the other sample points to its tangent lines. The height function is defined as a vector of distances of the other sample points to its tangent line.

Once these height functions are obtained a process called smoothing is performed on them. This makes the descriptor more compact and insensitive to local deformations. Ultimately, the shape of an object is represented as a sequence of

the height functions [44]. The proposed descriptor is reported to be invariant to geometric transformation and also insensitive to deformations due to noise or occlusion.

The process starts by defining the contour of the shape. Once this contour is defined a collection of equidistant points are set on it. This set of points is denoted as $X = \{x_i \ (i = 1, ..., N)\}$ where $N = 100$. Once that those sample points have been set an other important step before the calculation of height functions is to determine the axes.

Previously (Liu et al., 2008), height values were computed in a number of different directions. This however resulted into very high time complexity. Instead, the authors adjusted the angular direction for each of the selected points. In their work, the authors observe that a tangent line is an excellent reference line for height functions. The tangent line, l_i, for each sample point,x_i, is defined according to the contour orientation. Therefore the tangent line is starts from x_{i-1} and goes to x_{i+1}.

Then the distance between an other point, x_j, and the tangent line l_i is defined as height value h_{ij}. According to which side x_j resides a value is assigned. If it is to the left of the tangent line the value is positive, if to the right the value is negative and of the point is just on the axis of the tangent line, the value is zero. The authors point out that if the height value is more precise if its value is positive or negative. That way we can know on which side the point relies. Fig. 3.10 shows an example of this process where the height functions for point x_i are calculated.

In Fig. 3.10, the height functions for x_i are being calculated. The tangent line li is defined by using x_{i-1} and x_{i+1}. In this case, the tangent line is a horizontal one. Then the distance between other points and the tangent line is calculated. For instance, from point x_u to tangent line l_i, the height value is $h_{i,u}$. Upon doing

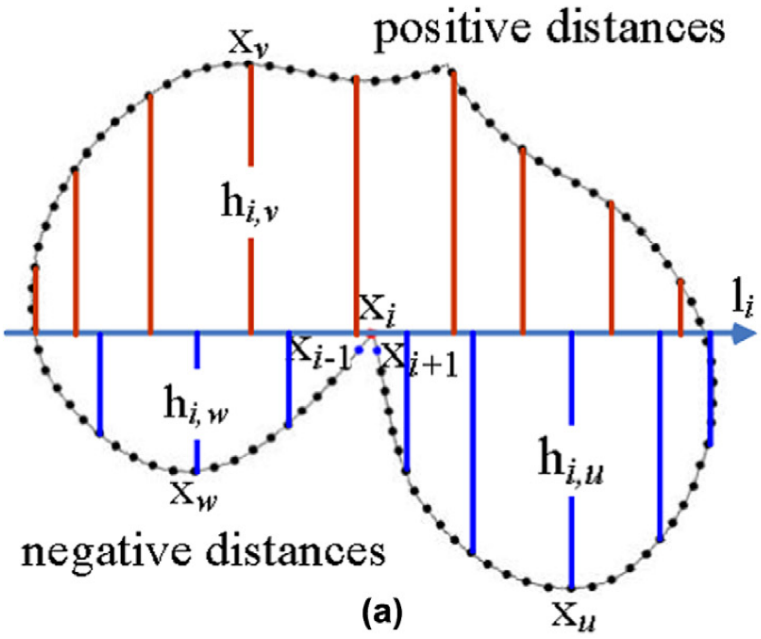


Figure 3.10: (a) Example of height functions being calculated according to sample point x_i. Adapted from [44].

so we can determine on which side of the tangent line the calculated points lie. For instance $h_{i,y}$ has a positive value and therefore its corresponding point,x_y,lies to the left of the tangent line, while $h_{i,w}$ and $h_{i,u}$ have negative values therefore points x_u and x_w lie on the right of the tangent line. This is done for each of the sample points, their height value is calculated according to tangent line l_i. And the shape descriptor for point x_i with respective to the shape X is the ordered sequence of the height values.

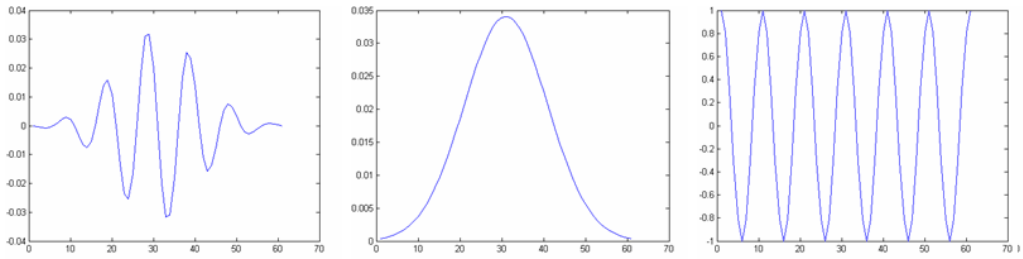
The development of efficient and robust shape descriptors is a topic of on going research. Apart from the ones described in this section are various other techniques have been developed [23, 46, 33, 39, 11].

3.5 COSFIRE Filters

COSFIRE filters are effective for key point detection and pattern recognition. They are trainable because they can be configured with any given prototypes. They are constructed by a configuration process which automatically analyses the dominant orientations and their mutual spatial arrangement of a given prototype pattern of interest [22]. We use Gabor filters to detect the dominant orientations. The response of a COSFIRE filter is computed as the weighted geometric mean of the involved Gabor filter responses. This means that a response is only achieved when all the concerned contour parts are present [22].

Gabor Filters

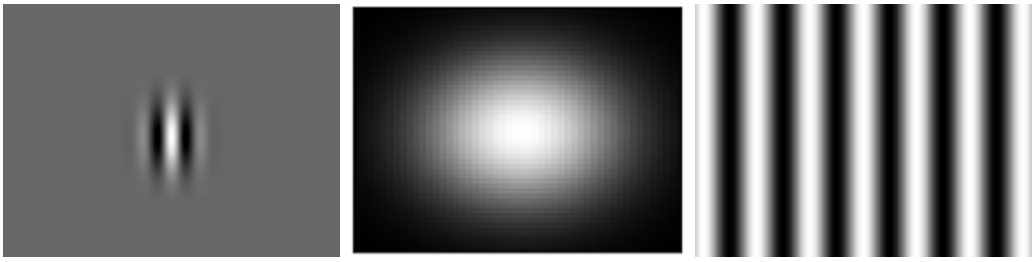
A one dimensional Gabor function is defined as the multiplication of a sinusoid with a Gaussian window as shown Fig. 3.11. Gabor functions have then been extended to two-dimensions, as the product of an elliptical Gaussian and a sinusoid plane wave [15]. Fig. 3.12 illustrates a Gabor function map that is tuned for vertical bars.



(a) Gabor Function (b) Gaussian Function (c) Cosine Function

Figure 3.11: (a) A one dimensional Gabor function is the product of a (b) Gaussian function and a (c) cosine function.

In their research Jones and Palmer [29] demonstrated that two dimensional (2D) Gabor functions, which were proposed by Daugman [15], can be used to model receptive fields of the orientation- selective simple cells of cats. Fig. 3.13 shows the similarity between two dimensional Gabor functions and receptive fields of cats



(a) Gabor Function (b) Gaussian Function (c) Cosine Function

Figure 3.12: (a) A two dimensional Gabor function is the product of an (b) elliptical Gaussian and a (c) complex plane wave.

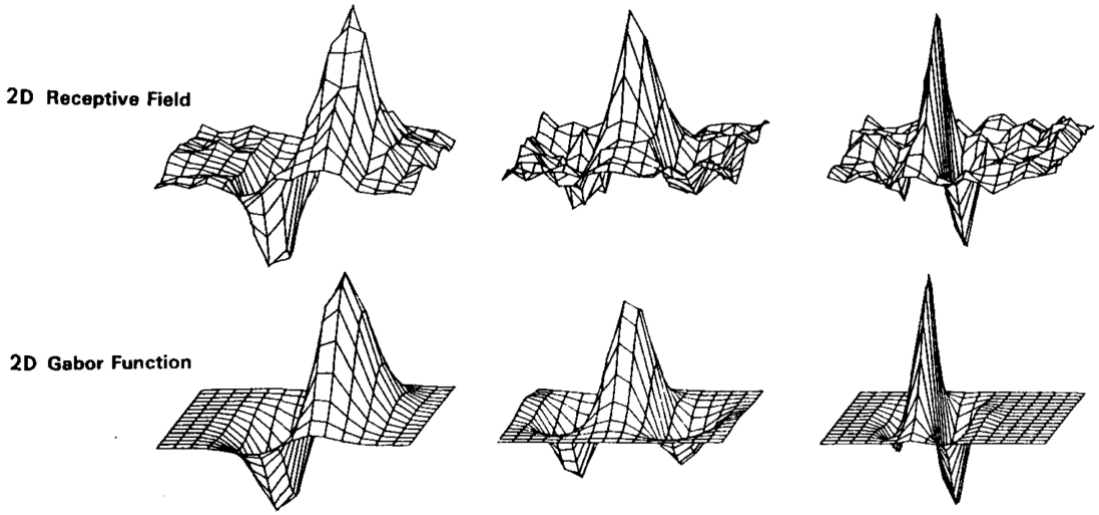


Figure 3.13: (First row) The receptive field profiles of some cells in a cats visual cortex compared to their corresponding best fitting (second row) two dimensional Gabor functions. Adapted from [16].

simple cells. The top row shows 2D receptive field profiles while the second row shows their corresponding best fitting Gabor functions [16].

Daugmans research [15] enabled the use of Gabor functions in computer vision applications as means to analyse images [16]. Recently, a novel CORF (Combination of Receptive Fields) computational model was proposed in [21]. The authors demonstrated that the CORF model exhibits more properties that are typical of simple cells than the Gabor function model. The CORF model also achieves better performance in contour detection tasks.

4. Methods

4.1 Overview

4.1.1 The COSFIRE Approach

COSFIRE filters are effective for keypoint detection and for pattern recognition. They are trainable because they can be configured with any given prototype patterns. They are constructed by a configuration process which automatically analyses the dominant orientations and their mutual spatial arrangement of a given prototype pattern of interest [22]. Gabor filters are used to detect the dominant orientations. The response of a COSFIRE filter is computed as the weighted geometric mean of the involved Gabor filter responses. This means that a response is only achieved when all the concerned contour parts are present [22].

4.1.2 Evaluation

The research method applied in this project is quantitative. This also includes data analysis and hypothesis testing. Through the application of COSFIRE filters and classification techniques on publicly available data sets the performance measurements are computed in the form of true positives and false positive rates which are used to derive accuracy, precision and recall rates. The data sets that are used have different levels of complexity due to noise, deformation, rotation and scaling. The way we evaluate COSFIRE filters for these data set is as follows.

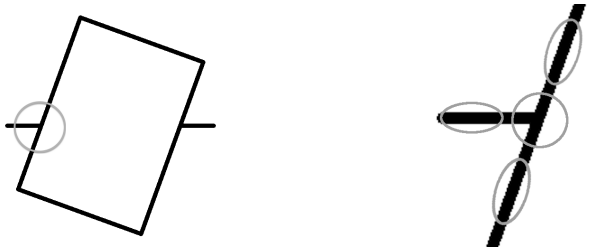
First, in a configuration stage, the COSFIRE filter is applied for each symbol model in the data set to create a collection of COSFIRE operators. Subsequently,

with the attained COSFIRE operators, a shape descriptor is formed which takes an image as input and creates a feature vector representing the input image according to the operators from the configuration phase. Finally we use the resulting feature vectors to classify test images.

4.2 COSFIRE Method

4.2.1 Overview

COSFIRE filters are configured by first specifying a point of interest. In Fig.4.1 a model symbol is used as an input image from which a local contour pattern is chosen as a prototype, encircled in Fig. 4.1a. This prototype pattern is used to automatically configure a COSFIRE filter that responds to similar patterns.



(a) (b)

Figure 4.1: (a) A model symbol (of size 512x512 pixels), of which one bifurcation (encircled) is used as a prototype pattern. (b) The enlarged prototype pattern where the dominant orientations around the point of interest are marked by 3 ellipses.

The three ellipses shown in Fig. 4.1b around the encircled bifurcation illustrate the dominant orientations in the spatial arrangement around the chosen point of interest. These dominant orientations are detected by using symmetric Gabor filters. The selection of Gabor filter responses depends on the analysis of the local prototype pattern. The encircled bifurcation represents the overlapping support of a group of Gabor filters.

During the application of the COSFIRE method, different Gabor filter responses are taken at different locations around a point by shifting the responses of these Gabor filters by different vectors. The type of Gabor filters, the locations at which we take their responses, and the shifting vectors are determined in a automatic configuration stage. Then the shifted responses are used for the pixel-wise evaluation of a multivariate function which ultimately produces the COSFIRE filter's output.The COSFIRE filter achieves a response only when presented with the same or similar spatial arrangement of lines with similar orientations and thicknesses to the prototype.

4.2.2 Using Gabor filters to Detect Orientations

The dominant orientations around a specified point of interest are detected by using symmetric two-dimensional (2D) Gabor filters. A 2D Gabor filter is defined as the product of an elliptical Gaussian and a sinusoid plane wave and is defined as follows:

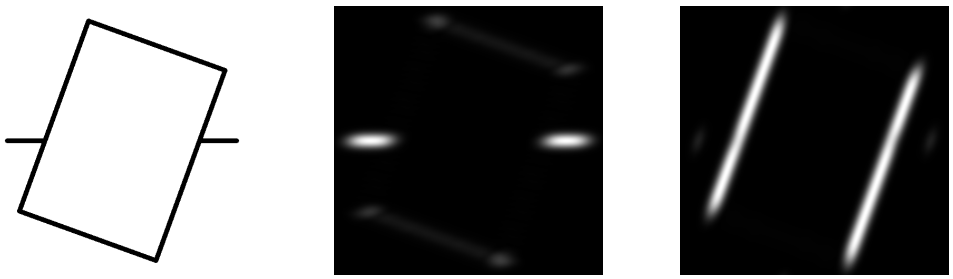
$$g(x, y; \lambda, \theta, \psi, \delta, \gamma) = \exp(-\frac{x'^2 + \gamma^2 y'^2}{2\delta^2}) \exp(2\pi \frac{x'}{\lambda} + \psi) \tag{4.1}$$

$$x' = x\cos(\theta) + y\sin(\theta) \tag{4.2}$$

$$y' = -x\sin(\theta) + y\cos(\theta) \tag{4.3}$$

The λ parameter, whose value is specified in pixels, represents the wavelength of the cosine factor, therefore the preferred wavelength of the filter. The θ parameter, who's value is specified in degrees, represents the orientation. The ψ parameter represents the phase offset of the cosine factor whose values, from 0 to 180, determine whether the Gabor function is symmetric or not. A symmetric Gabor function is produced by setting the ψ,phase offset, parameter value to 0° or 180°.

In this dissertation, the response of a symmetric Gabor filter at a given location (x,y) that is selective for a bar structure of orientation $\theta°$ and thickness 2λ is denoted by $g_{\lambda,\theta}(x, y)$. The response represents the detection of a contour part within the image of wavelength λ and of orientation $\theta°$. Fig.4.2 shows two examples of Gabor filters, of the same wavelength but with different orientations being applied to a symbol model image.



(a) (b) (c)

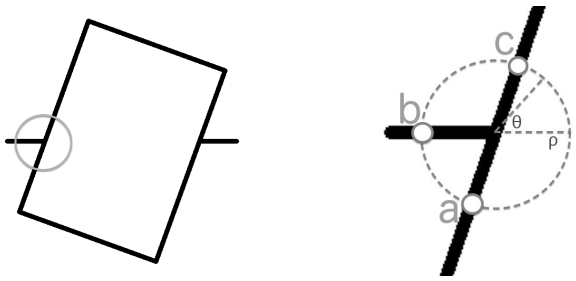
Figure 4.2: (b) A Gabor filter when applied to a (a) model symbol with $\lambda = 15$ and $\theta = 90°$. (c) The response of a Gabor filter when applied to a (a) model symbol with $\lambda = 15$ and $\theta = 160°$.

We then threshold the Gabor responses in order to eliminate undesirable responses to some noise or to undesirable responses around the contour parts. This threshold, $t1 = (0 \leq t1 \leq 1)$, is a fraction of the maximum response of $g_{\lambda,\theta}(x, y)$ across all combinations of values (λ, θ) and all possible positions within the image (x, y). The threshold responses are then denoted by $|g_{\lambda,\theta}(x, y)|_{t1}$. The value of the $t1$ threshold depends on the contrast the image contains. Ultimately, this threshold parameter controls the level at which a Gabor filter's response recognises the presence of a contour part. For our experiments we found that a value of t1 = 0.1 is sufficient.

4.2.3 Configuration

As input, COSFIRE filters use the responses of specific Gabor filters. Each of these Gabor filters responses represent a contour part in an image. Each response is denoted as a set of four parameter values, $(\lambda_i, \theta_i, \rho_i, \phi_i)$. The width of the detected contour part is represented by λ_i. The orientation of the same contour part is represented by θ_i. Finally its location is represented by polar coordinates ρ_i and ϕ_i with respect to the point of interest.

The parameter values of these contour parts are obtained as follows. A circle of radius ρ is placed around the point of interest. Along this circle we consider the responses of a bank of Gabor filters. From all those Gabor filters' responses we only take those that have the maximum response across all filters, that is across all combinations of (λ, θ). These are chosen to be the dominant orientations around the point of interest. We then determine the polar coordinates of the chosen points, in respect to the center of the COSFIRE filter.



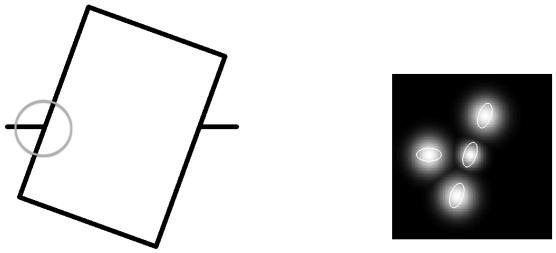
(a) (b)

Figure 4.3: (a) A circle of radius ρ is placed around a point of interest. (b) The points marked by labels a, b, and c represent the locations at which the Gabor filters achieve local maxima responses.

Fig. 4.3 shows an example configuration of a COSFIRE filter by with a circle around the point of interest in a symbol model image. For each location (ρ, ϕ) we then consider all the combinations of (λ, θ) for which the corresponding Gabor filter response, ${}_{g\lambda,\theta}(x, y)$, is greater than a fraction,$t_2 = 0.75$, of the maximum response

across all the combinations of λ and θ. For each value of θ that satisfies the condition above, we consider a single value of λ from the maximum of all responses. For each unique pair of (λ, θ), for location (ρ, ϕ) we then obtain a tuple $(\lambda_i, \theta_i, \rho_i, \phi_i)$.

More than one tuple can also be formed in the same location. Therefore, we express the set of parameter value combinations as $S_f = \Big\{(\lambda_i, \theta_i, \rho_i, \phi_i)|i = 1...n_f\Big\}$. The f subscript stands for the local prototype pattern around the specified point of interest. Fig. 4.4 is an example of the resultant configured operator from Fig. 4.3.



(a) (b)

Figure 4.4: (b) A COSFIRE operator is configured upon selecting (a) a prototype pattern where $\rho = 14$

The following shows the collection of tuples,S_f, returned by the COSFIRE operator shown in Fig.4.4. These tuples are then used in the application of the COSFIRE filter.

$$S_f = \left\{ \begin{array}{llll} (\lambda_1 = 11.3137, & \theta_1 = 2.7925, & \rho_1 = 0, & \phi_1 = 0), \\ (\lambda_2 = 11.3137, & \theta_2 = 2.7925, & \rho_2 = 14, & \phi_2 = 1.2043), \\ (\lambda_3 = 11.3137, & \theta_3 = 1.5708, & \rho_3 = 14, & \phi_3 = 3.1416), \\ (\lambda_4 = 11.3137, & \theta_4 = 1.6581, & \rho_4 = 14, & \phi_4 = 3.1416), \\ (\lambda_5 = 11.3137, & \theta_5 = 2.7925, & \rho_5 = 14, & \phi_5 = 4.4157), \end{array} \right\} \tag{4.4}$$

4.2.4 Application of a COSFIRE Filter

Applying Gabor filters

For each tuple i in the configured set S_f we apply a Gabor filter, to a test image, with the corresponding parameters λ_i and θ_i. A maximum response is achieved for a bar structure that has an orientation θ_i and thickness $2\lambda_i$.

Blurring and Shifting of Gabor responses

First, we blur the Gabor filter responses in order to allow for some tolerance in the position of their respective contour parts.

The blurring operation is defined as the computation of maximum value of the weighted thresholded responses of a Gabor filter. A Gaussian function, $G_\sigma(x, y)$, is used for weighting. The σ parameter represents the standard deviation of the Gaussian function which is expressed as a linear function of distance ρ from the center of the COSFIRE filter as shown in Eq. 1.5. where σ_0 and α are constants and the α parameter value determines the orientation tuning of the COSFIRE filter.

$$\sigma = \sigma_0 + \alpha\rho \tag{4.5}$$

We then take the blurred Gabor filter responses and shift them to the filter's centre so they meet at the same position, the same position we consider the center of support of the concerned COSFIRE filter. Each response is shifted by a distance ρ_i in the direction opposite of ϕ_i. In polar coordinates the shift vector by which the Gabor filter responses are shifted is specified by $(\rho_i, \phi_i + \pi)$ where as in cartesian coordinates it is specified as $(\Delta x_i, \Delta y_i)$ where $\Delta x_i = -\rho_i \cos(\phi_i)$ and $\Delta y_i = -\rho_i \sin(\phi_i)$. The blurred and shifted response of a Gabor filter, specified by the i-th tuple $(\lambda_i, \theta_i, \rho_i, \phi_i)$ in S_f, is denoted by $s_{(\lambda_i,\theta_i,\rho_i,\phi_i)}(x, y)$.

$$s_{(\lambda_i,\theta_i,\rho_i,\phi_i)}(x,y) = \max_{x',y'} \left\{ |g_{\lambda_i,\theta_i}(x - x' - \Delta x_i, y - y' - \Delta y_i)|_{t_1} G_\sigma(x',y') \right\} \quad (4.6)$$

$$-3\sigma \leq x, y \leq 3\sigma \quad (4.7)$$

Response of a COSFIRE filter

The response of a COSFIRE filter, denoted by $r_{S_f}(x,y)$, is defined as the weighted geometric mean of all the thresholded blurred and shifted Gabor filter responses, denoted by $s_{\lambda_i,\theta_i,\rho_i,\phi_i}(x,y)$, that correspond to the contour parts defined in S_f:

$$r_{S_f}(x,y) = \left| \left(\prod_{i=1}^{|S_f|} (s_{\lambda_i,\theta_i,\rho_i,\phi_i}(x,y))^{\omega_i} \right)^{1/\sum_{i=1}^{|S_f|}\omega_i} \right|_{t_3} \quad (4.8)$$

$$\omega_i = exp - \frac{\rho_i^2}{2\sigma'^2}, 0 \leq t_3 \leq 1 \quad (4.9)$$

The $|.|_{t_3}$ stands for the thresholding the response at a fraction t_3 of its maximum across all image coordinates. COSFIRE filters can achieve invariance to rotation, scale, reflection and contrast inversion. We refer to [22] for the technical description of how invariance of COSFIRE filters is achieved

4.3 Evaluation

4.3.1 Pre-Processing

Each dataset's collection of test images varies according to some amount of deformation, noise or geometrical transformation. In order to aid the classification process a number of pre-processing steps were taken in order to clean/enhance the test images. As an initial step all test images are resized by a specific factor. Fur-

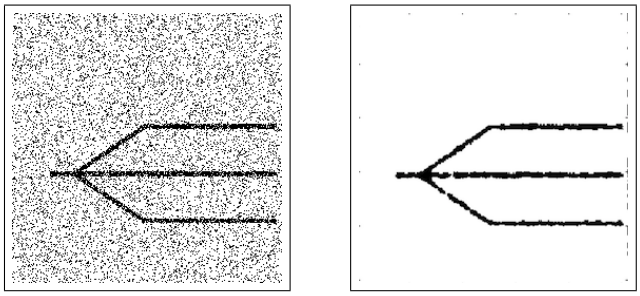
thermore, test images which contain salt and pepper noise, such as noise E's, are de-noised using median filtering and degraded test images, such as Noise A's, are enhanced by the use of dilation. Other test images, such as Noise B's were thickened. Also with this thickening noise was included, therefore both median filtering and thinning were applied. The following are the pre-processing techniques which were applied in this evaluation.

Re-sizing

For efficiency reasons, prior to configuring COSFIRE filter/s we first resize the model by a factor of 0.5, using bicubic interpolation, before being inputted into the configuration and training phases. The original model images are of size (512×512) pixels. By applying a resize factor of 0.5, the images are resized to $(512(0.5) \times 512(0.5)) = (256 \times 256)$ pixels. This reduces the time in which the COSFIRE operator is applied to an image by half due to less information to process. No pre-processing was done on the test images.

De-noising by median filtering

Datasets such as the Noise E contain "salt and pepper" noise. Therefore before running an experiment using those datasets some pre-processing is required in order to reduce noise. De-noising is done through the use of median filtering. A median filter considers each pixel in the image sequentially and analyses the surrounding pixels. The value of the considered pixel is replaced with the median of the values of the surrounding pixels [6]. This reduces the noise from the image as shown in Fig. 4.5 .

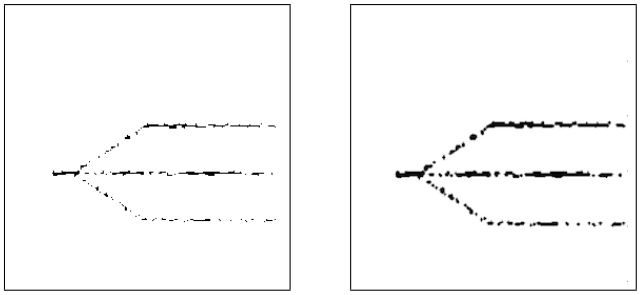


(a) (b)

Figure 4.5: (a) A symbol with salt and pepper noise is (b) processed through the use of median filtering

Dilation

Some datasets, such as Noise A, have degraded test images. Before classifying such as test images an other pre-processing step that is implemented is morphological dilation. The effect of morphological dilation is to gradually enlarge the boundaries of specific regions of foreground pixels. Therefore these regions grow in size while holes or gaps within the symbol become smaller, thus enhancing the image. In order to do this a structuring element is used which determines the dilation effect on the test image [7]. Fig 4.6 shows an example.



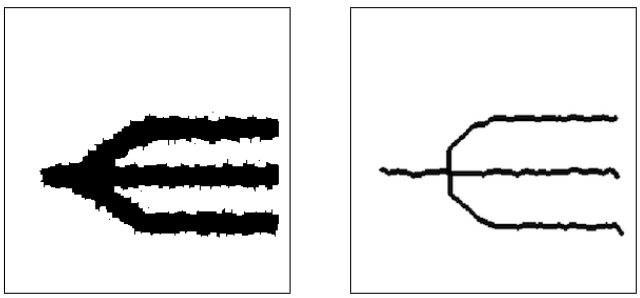
(a) (b)

Figure 4.6: (a) A symbol of degraded quality (b) is processed using morphological dilation

Thinning

Thinning is an other morphological operator which removes pixels so that an object shrinks to a minimal stroke. Like dilation, thinning relies on a structuring element

[9]. Datasets, such as noise B, contain test images which have their lines' thickened. Further more, the process of thickening the symbol's geometric parts adds noise around the edges as well. This is countered by doing the following. Like dilation, thinning relies on a structuring element [9]. In this case, we use a structuring element of line form. Twelve different line structuring elements are created with different orientations. These orientations are applied to the test image and the outputs for each are then superimposed. Subsequently skeletonisation and dilation using a disc structuring elements are applied. A median filter is also applied, in order to remove the noise around the edges, and thinning. Fig 4.7 shows an example.



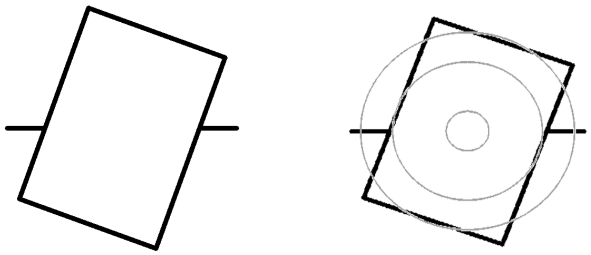
(a) (b)

Figure 4.7: (a) A symbol of thickened quality (b) is processed using median filtering and thinning

4.3.2 Forming a Shape Descriptor

In the configuration process of COSFIRE filters, features of interest are chosen by specifying a point or a collection of points within the model symbol. The system automatically analyses those points and extracts information about the dominant orientations and their mutual spatial arrangement according to the list of concentric circles surrounding the point of interest. The number of concentric circles is not intrinsic to the method but rather to the complexity of the given prototype pattern of interest. These are the prototype patterns that the COSFIRE filter will learn to recognise in the testing images. [22].

For the purpose of electrical and architectural symbol recognition, a COSFIRE filter is configured for every model symbol in a given data set. These configurations were approached in two ways. In the first approach that was used in the first six experiments, was to select a single static point of interest which was placed in the middle of the model symbol. Fig. 4.8 shows a configuration, using the described single point approach.



(a) (b)

Figure 4.8: A COSFIRE filter is configured by first placing (b) the point of interest at the centre of the (a) model symbol

The resulting COSFIRE filters are used to build a shape descriptor as follows. The created COSFIRE operators are applied to each symbol model. Each COSFIRE operator achieves a response for every location in the given image. From this response image, we choose the one that has the maximum value. This means that a symbol model image is described by a vector of values with a size that is equivalent to the number of COSFIRE filters used.

Fig. 4.9 shows an example of the result attained when the model symbols are processed by the shape descriptor. In this example there are 17 models images and 1 operator is applied to each model image. Each model image is therefore represented by a feature vector of 17 maximum values. The diagonal values in this descriptor have the highest responses. These values correspond to the model image being applied to it's corresponding COSFIRE operator.

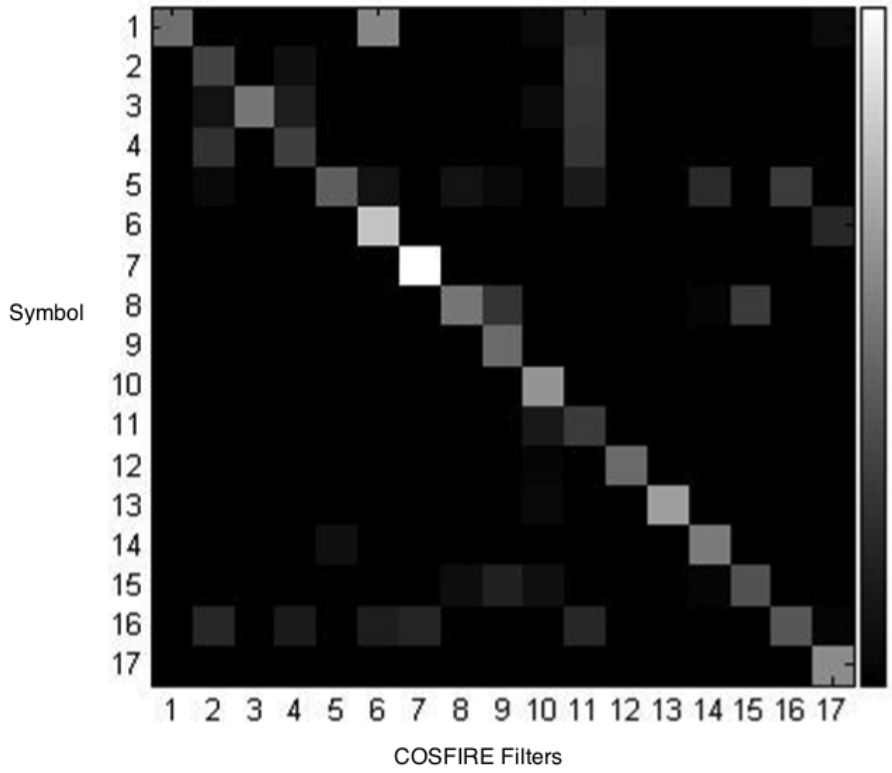
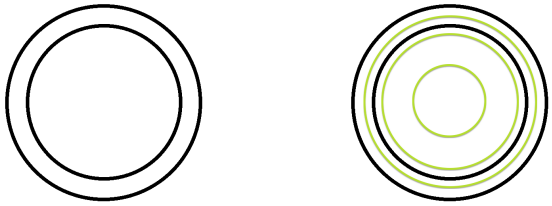


Figure 4.9: A row i represents the feature vector of a symbol model composed of the maximum responses of COSFIRE filters. Element j of a feature vector i is the maximum response of a COSFIRE filter which is configured by model symbol j

Although the single point approach towards configuring filters produces good results in the first few datasets, a problem is introduced concerning the list of concentric circles used. Using the same list of concentric circles for all the datasets means that for some model symbols no dominant orientations could be detected.This resulted in test symbols not being properly recognised. An example is shown in Fig. 4.10.

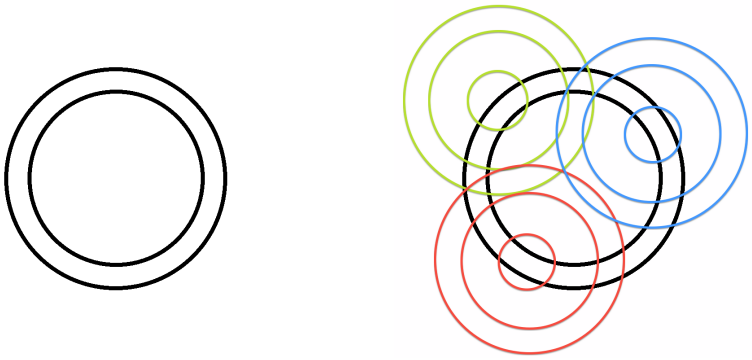
In order to address this issue, an alternative approach is taken in selecting points of interest. The list of concentric circles is set to three and three points of interest are placed randomly in the symbol model image. Each of these points hold the same list of concentric circles. Doing so ensures that each model symbol has



(a) Model Symbol (b) COSFIRE Operator

Figure 4.10: (b) Inappropriate selection of concentric circles around the center of (a) a specific model symbol. Here we use the same three concentric circles used in the previous example.

a unique set of detected contour parts. An example of this approach is shown in Fig. 4.11.



(a) Model Symbol (b) COSFIRE Operator

Figure 4.11: (b) Three COSFIRE filters are being configured for the symbol (a). The concentric circles for each point of interest are shown in different colours.

This approach is driven by three parameters η, β, and ς. The η parameter value determines the number of points of interest placed in the model image. The β parameter value determines the minimum number of tuples that should be attained from the point of interest's surrounding spatial arrangement. The ς parameter value determines the number of unique orientations that should be detected. This means that $\eta = 3$ operators will be applied for each symbol mode. These operators should have a minimum of $\beta = 5$ tuples each with at least $\varsigma = 2$ unique orientations.

As a result of using the three point approach we have available a set of COSFIRE operators with a cardinality of three times the size of the total number of symbol model images available. Therefore for a dataset of k model images, the configuration results in $3k$ distinct COSFIRE filter operators.

4.3.3 Classification

Classification refers to the problem of identifying, the set/s sets of categories/-classes, certain objects or patterns belong to. In this thesis, this problem is tackled by trying to relate a test image to a specific symbol model by recognising contour parts which are similar in both the model image and the test image. This is done by using the feature vectors which represent both images.

Feature Vectors

Inputting each of the Model images into the shape descriptor, described in sub section 4.3.2, it creates a feature vector for each. This means that if there are 150 models in the data set and 3 operators are applied for each model 150 different feature vectors are created with 450 elements each.

Normalisation

Prior to classification of the test images, normalisation is applied on both the training set of feature vectors and the testing set of feature vectors. This is done so that

the test image and it's corresponding model image are rendered more comparable thus aiding the classification process. Normalisation is done using Z-Normalisation also known as zero-mean normalisation. This normalisation technique transforms the distribution of pixel values into a standard normal distribution.

For each 450 elements in the feature vector, the mean, μ_x, and the standard deviation, σ_x, are computed. The configuration feature vectors are then normalised by subtracting the mean and diving the standard deviation. The same is done for the feature vectors which are extracted from the testing images. The following is how a normalised value is obtained using z-score normalisation [4]:

$$NormalisedValue = \frac{v - \mu_x}{\sigma_x} \tag{4.10}$$

K-nearest neighbour algorithm

The K-nearest neighbour algorithm (K-nn), known to be amongst the top 10 data mining algorithms [45], is a versatile technique who's application range from vision to computational geometry [42]. It has been used for the classification of objects or patterns according to the closest training examples available. Before using the Knn algorithm, some prerequisite steps need to be taken. The first step is to ensure that the data is in a feature space. This data can be both single scalars or even multidimensional vectors [42]. The second step is to define a constant K.

The constant K is a user defined value. It is the number of instances / neighbours the algorithm allows to influence the classification [42]. Since the test symbol needs to be assigned to the class of its nearest neighbour, the constant k is set to $k = 1$. An other important step is to determine which distance metric to use. In order to classify the test image's feature vector to the nearest feature vector from the models a distance metric needs to be used. Since both the models and the

test images are represented by feature vectors made of n number of elements, the distance metric used in this classification process is Euclidean distance. Euclidean distance is defined as follows where M is the feature vector for the model image and T is the feature vector for the test image [1].

$$d(M,T) = \sqrt{\sum_{i=1}^{n}(M_i - T)^2} \tag{4.11}$$

Therefore classification is achieved by computing the Euclidean distance between the test image feature vector to each training feature vector in the shape descriptor. A test vector is then classified to the symbol for which the Euclidean distance is the smallest, therefore the most similar. An illustration of the Knn algorithm is shown in Fig. 4.12.

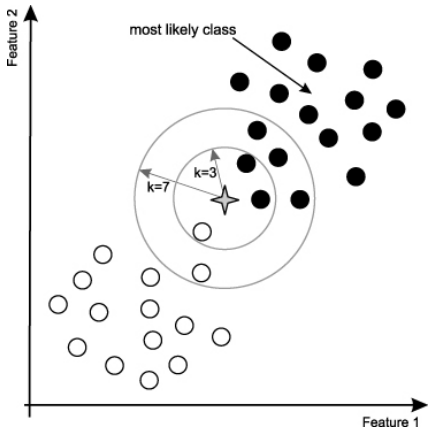


Figure 4.12: An object (star) whose class is not known is being classified according to its nearest neighbours. Two examples are shown, where $k = 3$ and $k = 7$ [1].

.

[1]Taken From http://iopscience.iop.org/1742-5468/2010/11/P11015/fulltext/

4.3.4 Evaluation Summary

Fig. 4.13 shows a work flow of the whole evaluation process for this dissertation.

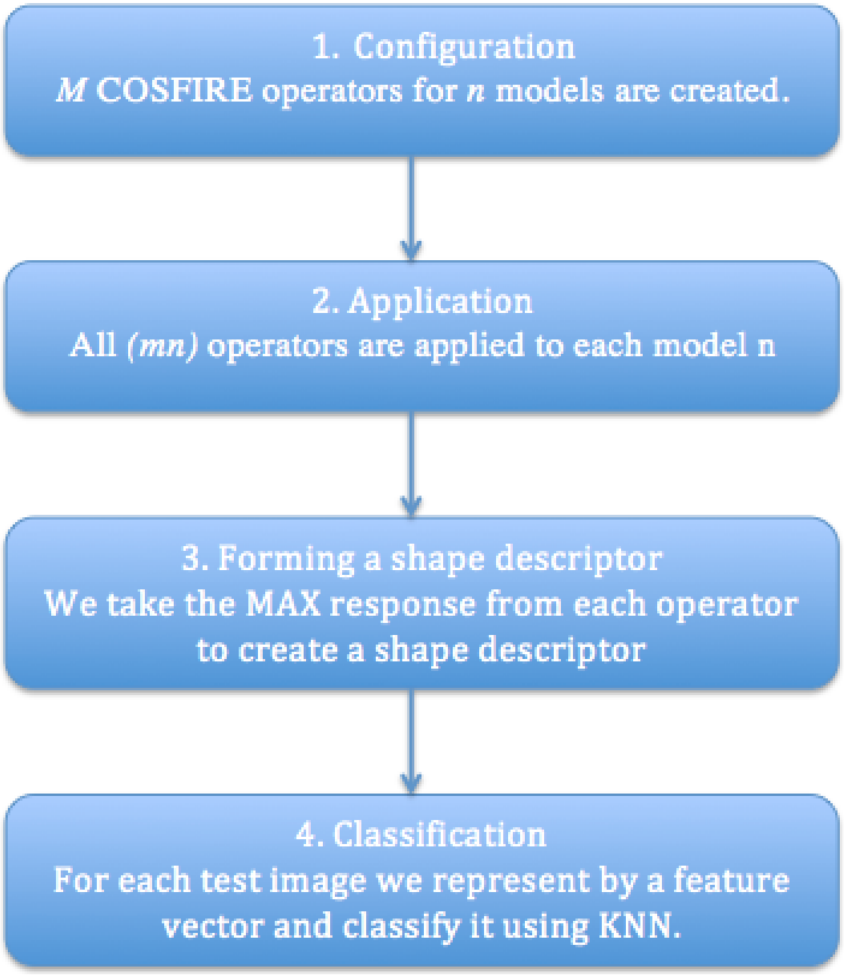


Figure 4.13: (1) M COSFIRE operators for n model images are created. (2) All (mn) operators are applied to each of the model images and (3) from those responses we take the maximum responses and create a feature vector for each model to form a shape descriptor. (4) We finally use the created shape descriptor to classify the images.

5. Experimental Results

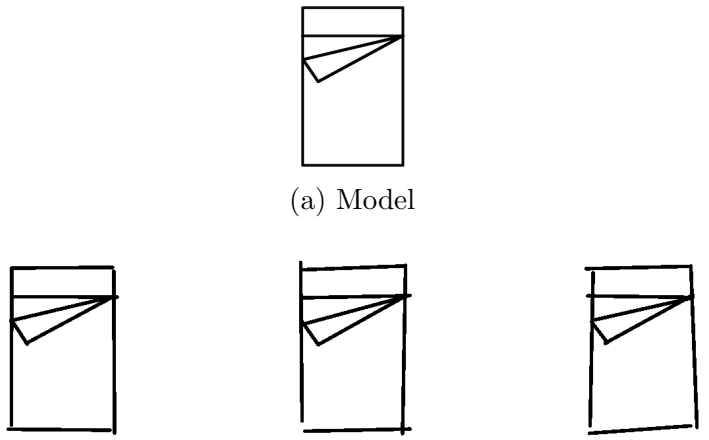
5.1 Overview

We start by evaluating the proposed method on the data set with the least level of complexity, category 1, which contains noiseless test images given in the same orientation and scale of their corresponding models. Then we apply the method to the data set with the second lowest complexity, categories 2 and 3, and finally we evaluate the COSFIRE filters on the data set with the highest level of complexity, category 4. The mentioned order in which the experiments are run facilitates the analysis of the performance results and gives insight on further tuning the method. The following sub-sections are the results attained for each dataset.

5.2 Category 1 Datasets

5.2.1 Sketches25f

This sub category of datasets is split into three different sets namely Sketches25f-level1, Sketches25f-level2, and Sketches25f-level3. Each of these data sets consist of 17 different model symbols. These model symbols consist of straight lines only and do not include any arcs or circles. This data set holds 1000 test images. This test images are given in the same orientation and in the same scale of the models. However, these test images are deformed symbols of increasing complexity according to level. Fig.5.1 illustrates a sample of this dataset.



(a) Model

(b) Level 1 (c) Level 2 (d) Level 3

Figure 5.1: Sample test images from the three different datasets under Sketches25f are shown for (a) a model symbol. (b) is a sample of Sketches25f-level1 , (c) is a sample of Sketches25f-level2 and (d) is a sample of Sketches25f-level3.

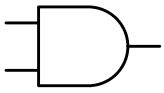
Results

Table 5.1: Recognition results for dataset 'Sketches25f'

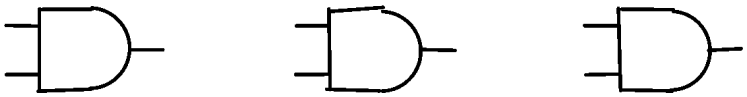
Degradation Level	Recognition Rate
1	100 %
2	100 %
3	100 %

5.2.2 Sketches25

This sub category of datasets is split into three different sets namely Sketches25-level1, Sketches25-level2, and Sketches25-level3. Each of these data sets consist of 25 different model symbols. These model symbols consist of both straight lines and arcs or circles. This data set holds 1000 test images. This test images are given in the same orientation and in the same scale of the models. However, these test images are deformed symbols of increasing complexity according to level. Fig.5.2 illustrates a sample of this dataset.



(a) Model



(b) Level 1 (c) Level 2 (d) Level 3

Figure 5.2: Sample test images from the three different datasets under Sketches25 are shown for (a) a model symbol. (b) is a sample of Sketches25-level1 , (c) is a sample of Sketches25-level2 and (d) is a sample of Sketches25-level3.

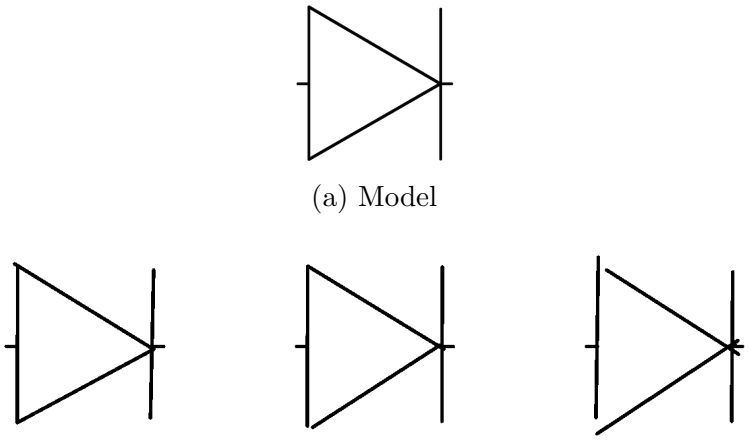
Results

Table 5.2: Recognition results for dataset 'Sketches25'

Degradation Level	Recognition Rate
1	100 %
2	100 %
3	100 %

5.2.3 Sketches50f

This sub category of datasets is split into three different sets namely Sketches50f-level1, Sketches50f-level2, and Sketches50f-level3. Each of these data sets consist of 26 different model symbols. These model symbols consist of straight lines only and do not include any arcs or circles. This data set holds 1000 test images. This test images are given in the same orientation and in the same scale of the models. However, these test images are deformed symbols of increasing complexity according to level. Fig.5.3 illustrates a sample of this dataset.



(a) Model

(b) Level 1 (c) Level 2 (d) Level 3

Figure 5.3: Sample test images from the three different datasets under Sketches50f are shown for (a) a model symbol. (b) is a sample of Sketches50f-level1 , (c) is a sample of Sketches50f-level2 and (d) is a sample of Sketches50f-level3.

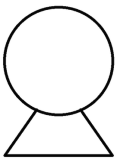
Results

Table 5.3: Recognition results for dataset 'Sketches50f'

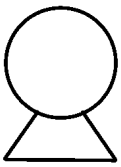
Degradation Level	Recognition Rate
1	100 %
2	99.8 %
3	99.4 %

5.2.4 Sketches50

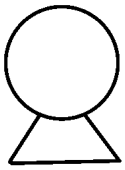
This sub category of datasets is split into three different sets namely Sketches50-level1, Sketches50-level2, and Sketches50-level3. Each of these data sets consist of 50 different model symbols. These model symbols consist of both straight lines and arcs or circles. This data set holds 1000 test images. This test images are given in the same orientation and in the same scale of the models. However, these test images are deformed symbols of increasing complexity according to level. Fig.5.4 illustrates a sample of this dataset.



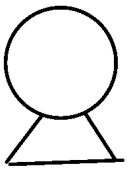
(a) Model



(b) Level 1



(c) Level 2



(d) Level 3

Figure 5.4: Sample test images from the three different datasets under Sketches50 are shown for (a) a model symbol. (b) is a sample of Sketches50-level1 , (c) is a sample of Sketches50-level2 and (d) is a sample of Sketches50-level3.

Results

Table 5.4: Recognition results for dataset 'Sketches50'

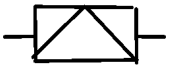
Degradation Level	Recognition Rate
1	100 %
2	100 %
3	100 %

5.2.5 Sketches100f

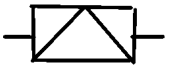
This sub category of datasets is split into three different sets namely Sketches100f-level1, Sketches100f-level2, and Sketches100f-level3. Each of these data sets consist of 51 different model symbols. These model symbols consist of straight lines only and do not include any arcs or circles. This data set holds 1000 test images. This test images are given in the same orientation and in the same scale of the models. However, these test images are deformed symbols of increasing complexity according to level. Fig.5.5 illustrates a sample of this dataset.

(a) Mode

(b) level 1



(c) level 2



(d) level 3

Figure 5.5: Sample test images from the three different datasets under Sketches100f are shown for (a) a model symbol. (b) is a sample of Sketches100f-level1 , (c) is a sample of Sketches100f-level2 and (d) is a sample of Sketches100f-level3.

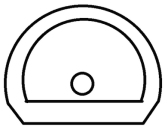
Results

Table 5.5: Recognition results for dataset 'Sketches100f'

Degradation Level	Recognition Rate
1	98.1 %
2	98.7 %
3	98.1 %

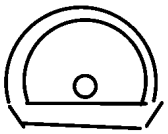
5.2.6 Sketches100

This sub category of datasets is split into three different sets namely Sketches100-level1, Sketches100-level2, and Sketches500-level3. Each of these data sets consist of 100 different model symbols. These model symbols consist of both straight lines and arcs or circles. This data set holds 1000 test images. This test images are given in the same orientation and in the same scale of the models. However, these test images are deformed symbols of increasing complexity according to level. Fig.5.6 illustrates a sample of this dataset.

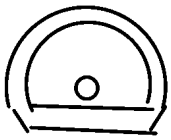


(a) Model

(b) Level 1



(c) Level 2



(d) Level 3

Figure 5.6: Sample test images from the three different datasets under Sketches100 are shown for (a) a model symbol. (b) is a sample of Sketches100-level1 , (c) is a sample of Sketches100-level2 and (d) is a sample of Sketches100-level3.

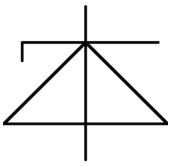
Results

Table 5.6: Recognition results for dataset 'Sketches100'

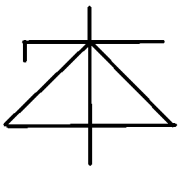
Degradation Level	Recognition Rate
1	99.2 %
2	98.9 %
3	98.2 %

5.2.7 Sketches150f

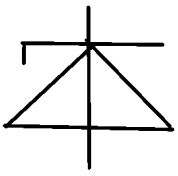
This sub category of datasets is split into three different sets namely Sketches150f-level1, Sketches150f-level2, and Sketches150f-level3. Each of these data sets consist of 80 different model symbols. These model symbols consist of straight lines only and do not include any arcs or circles. This data set holds 1000 test images. This test images are given in the same orientation and in the same scale of the models. However, these test images are deformed symbols of increasing complexity according to level. Fig.5.7 illustrates a sample of this dataset.



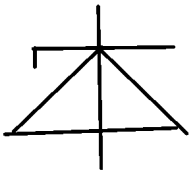
(a) Model



(b) Level 1



(c) Level 2



(d) Level 3

Figure 5.7: Sample test images from the three different datasets under Sketches150f are shown for (a) a model symbol. (b) is a sample of Sketches150f-level1 , (c) is a sample of Sketches150f-level2 and (d) is a sample of Sketches150f-level3.

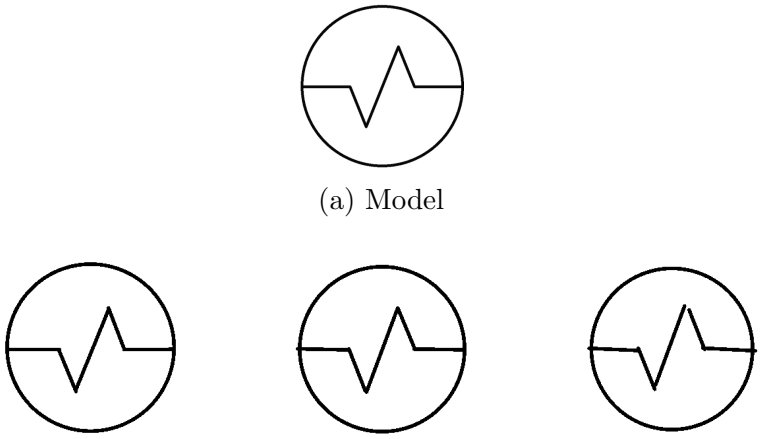
Results

Table 5.7: Recognition results for dataset 'Sketches150f'

Degradation Level	Recognition Rate
1	99.3 %
2	99.2 %
3	98.8 %

5.2.8 Sketches150

This sub category of datasets is split into three different sets namely Sketches150-level1, Sketches150-level2, and Sketches150-level3. Each of these data sets consist of 150 different model symbols. These model symbols consist of both straight lines and arcs or circles. This data set holds 1000 test images. These test images are given in the same orientation and in the same scale of the models. However, these test images are deformed symbols of increasing complexity according to level. Fig.5.8 illustrates a sample of this dataset.



(a) Model

(b) Level 1 (c) Level 2 (d) Level 3

Figure 5.8: Sample test images from the three different datasets under Sketches150 are shown for (a) a model symbol. (b) is a sample of Sketches150-level1 , (c) is a sample of Sketches150-level2 and (d) is a sample of Sketches150-level3.

Results

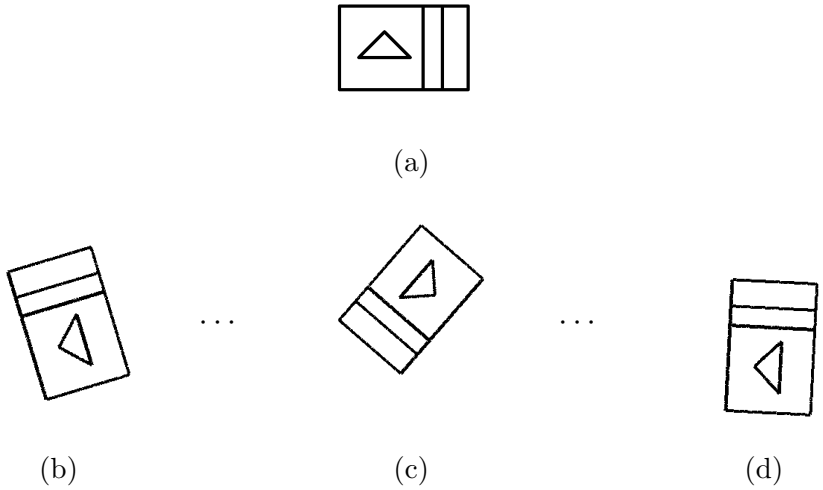
Table 5.8: Recognition results for dataset 'Sketches150'

Degradation Level	Recognition Rate
1	99.3 %
2	98.9 %
3	97.6 %

5.3 Category 2 Datasets

5.3.1 Rotation

This data set consists of 150 different models and 10 test symbols for each of those models.These model symbols consist of both straight lines and arcs or circles. In total this dataset holds 1500 test images. The test images are given in the same scale of the models. Orientation varies through out the 25 test symbols for each model. No noise is introduced to the test images. Fig. 5.9 illustrates a few samples.



(a)

(b) (c) (d)

Figure 5.9: (b) (c) (d) Three test images for (a) a model symbol are illustrated from the Rotation dataset

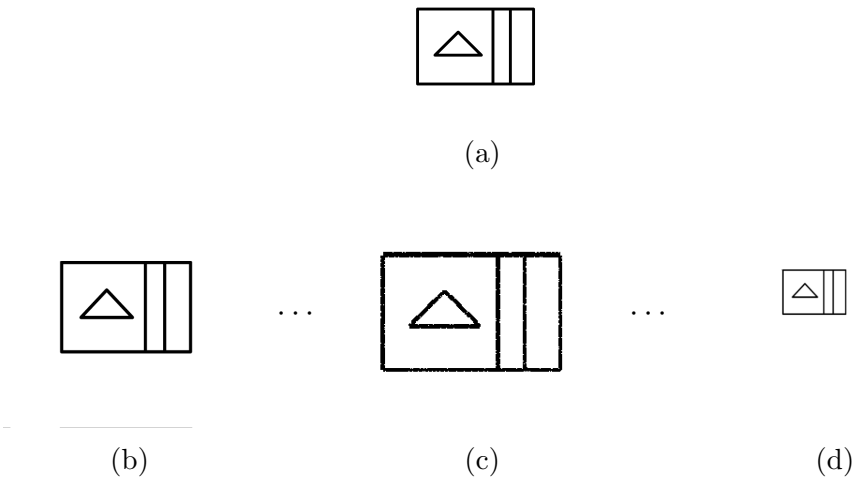
Results

Table 5.9: Recognition results for dataset 'Rotation'

Recognition Rate
72.6 %

5.3.2 Scaling

This data set consists of 150 different models and 10 test symbols for each of those models.These model symbols consist of both straight lines and arcs or circles. In total this dataset holds 1500 test images. The test images are given in the same orientation of the models. Scaling varies through out the 25 test symbols for each model. No noise is introduced to the test images. Fig. 5.10 illustrates a few samples.



(a)

(b) (c) (d)

Figure 5.10: (b) (c) (d) Three test images for (a) a model symbol are illustrated from the Scaling dataset

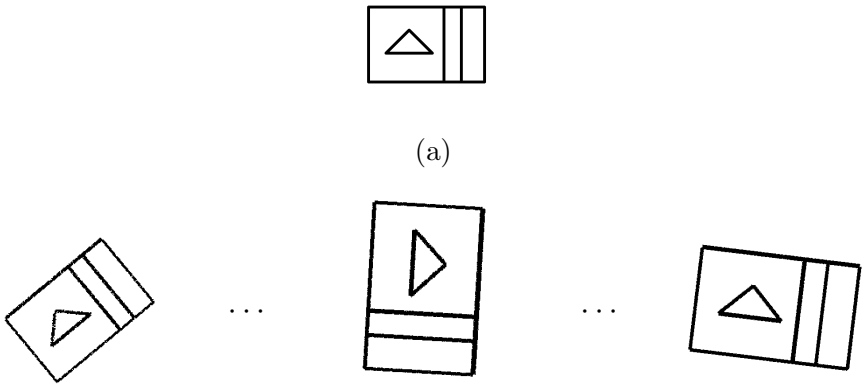
Results

Table 5.10: Recognition results for dataset 'Scaling'

Recognition Rate
89.5 %

5.3.3 RotationScaling

This data set consists of 150 different models and 10 test symbols for each of those models.These model symbols consist of both straight lines and arcs or circles. In total this dataset holds 1500 test images. Scaling and orientation vary from those in the model through out each of the 25 test symbols. No noise is introduced to the test images. Fig. 5.11 illustrates a few samples.



(a)

(b) (c) (d)

Figure 5.11: (b) (c) (d) Three test images for (a) a model symbol are illustrated from the RotationScaling dataset

Results

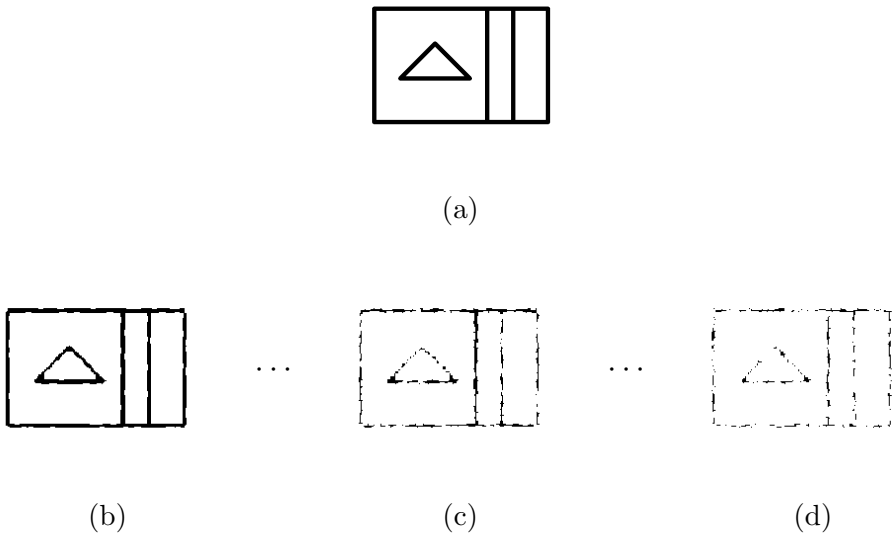
Table 5.11: Recognition results for dataset 'RotationScaling'

Recognition Rate
49.2 %

5.4 Category 3 Datasets

5.4.1 Noise A

This data set consists of 150 different models and 25 test symbols for each of those models.These model symbols consist of both straight lines and arcs or circles. In total this dataset holds 3750 test images. The test images are given in the same orientation and in the same scale of the models. Noise is introduced to the test images in the form of line degradation. Fig. 5.12 illustrates a few samples.



(a)

(b) (c) (d)

Figure 5.12: (b) (c) (d) Three test images for (a) a model symbol are illustrated from the Noise A

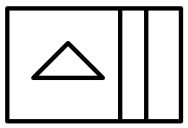
Results

Table 5.12: Recognition results for dataset 'Noise A'

Recognition Rate
96.13 %

5.4.2 Noise B

This data set consists of 150 different models and 25 test symbols for each of those models.These model symbols consist of both straight lines and arcs or circles. In total this dataset holds 3750 test images. The test images are given in the same orientation and in the same scale of the models. Noise is introduced to the test images in the form of line thickening. Fig. 5.13 illustrates a few samples.



(a)

(b) (c) (d)

Figure 5.13: (b) (c) (d) Three test images for (a) a model symbol are illustrated from the Noise B

Results

Table 5.13: Recognition results for dataset 'Noise B'

Recognition Rate
90.8 %

5.4.3 Noise E

This data set consists of 150 different models and 25 test symbols for each of those models. These model symbols consist of both straight lines and arcs or circles. In total this dataset holds 3750 test images. The test images are given in the same orientation and in the same scale of the models. Noise is introduced to the test images in the form of of global noise. Fig. 5.14 illustrates a few samples.

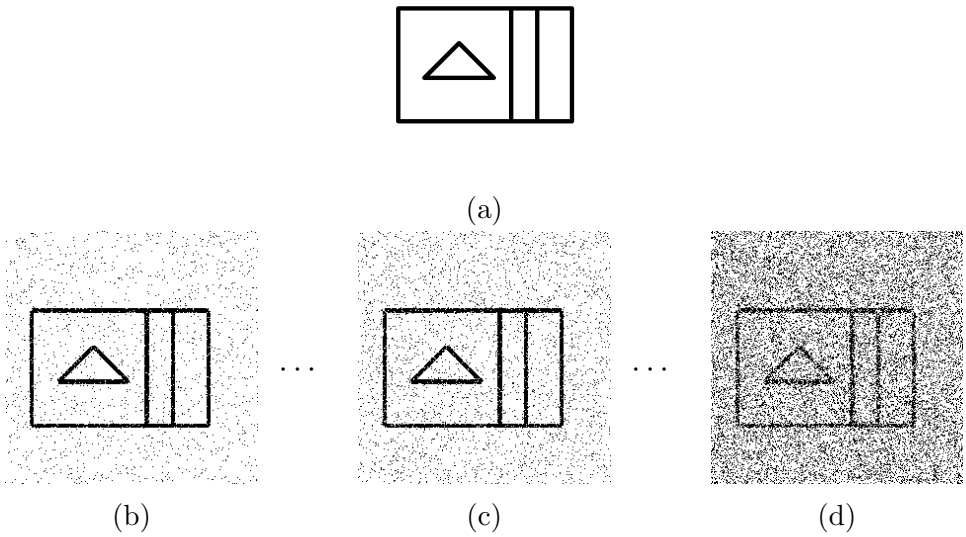


Figure 5.14: (b) (c) (d) Three test images for (a) a model symbol are illustrated from the Noise E dataset

Results

Table 5.14: Recognition results for dataset 'Noise E'

Recognition Rate
94.93 %

6. Discussion

6.1 Overview

The approach presented in this study shows a number of strengths. One of the most important strengths of this approach is that it is a trainable one. This means that no prior knowledge is needed, about the application or how the symbols are designed, in the way it operates. Patterns or prototype features, which are used for their configuration are randomly selected. Subsequently, only identical or similar patterns are detected where all parts of the filter defining the selected prototypes' features are present.

This makes the approach not just effective for electrical or architectural symbols only but any type of symbol which can be found in various documents, as described in Section 3.2. This versatility in application is achieved by the application of the COSFIRE filters in various invariance modes. COSFIRE filters can be applied in rotation, scale and reflection invariant mode. Furthermore, COSFIRE filters are not limited to the use of Gabor filters only. Any filter which provides information about contours can be used.

The presented approach is effective when applied to datasets with symbols, with both straight lines and curvatures, that included different levels of deformations, different types of noises and different types of geometrical transformations. In most of the datasets that have been processed the recognition rate was higher than that reported in the GREC'11 contest paper. The only datasets which did not surpass the recognition rates reported in the contest paper where the Rotation and

RotationScaling datasets.

6.2 Robustness to Scalability

The best recognition rates for the first category of datasets was achieved with the first 9 experiments, achieving a perfect recognition rate. After those experiments the recognition only decreased slightly, with the lowest recognition of 97.6%. This decrease in recognition rate is due to the fact that the number of models available for each dataset increases, therefore classification becomes harder. The first 9 datasets in which we achieved a 100% recognition rate hold 25 or 50 different models while the rest of the datasets hold 100 to 150 different models. Even though the most complex dataset in this category holds the lowest result, over all performance was very good. This shows that our approach is a scalable one, and that if more models are to be added the results remain good.This shows that the approach used in this study is effective for the first category of datasets which contained only deformations. Fig. 6.1, shows a summary of the results obtained for the first category of datasets.

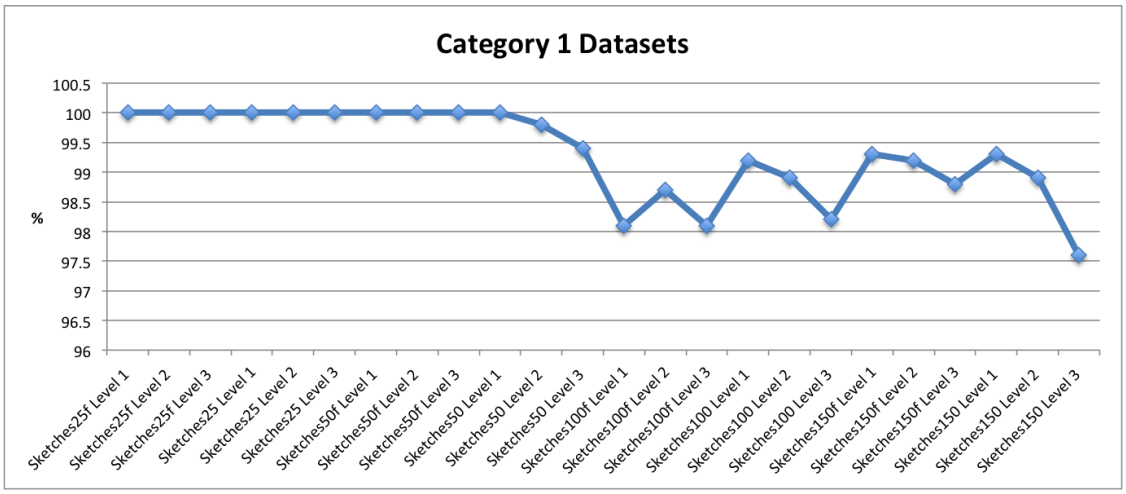


Figure 6.1: A graph showing the recognition rates for the first category of datasets

6.3 Robustness to Geometrical Transformations

The second category of datasets was more difficult due to geometrical transformation such as rotation, scaling or both combined apart from have 150 different models for each dataset. The results show that, Fig.6.2, show that the best performing data set out of this category was the Scaling dataset. The recognition rate achieved with this approached was higher than the recognition rate achieved in the GREC'11 contest. When the rotation transformation was separately introduced in the test images, recognition rates decreased slightly. And the lowest rate was achieved when both scaling and rotation where both introduced in the test image.

The drop in recognition rate is due to the number of different orientations chosen at the configuration stage of our approach. Results show that the number of orientations considered was not enough to be able to pick up the rotations introduced in the test images. However even though results have decreased slightly performance remains good which indicates that our approach can also handle geometrical transformations. The following is a summary of results for the second category of datasets, Fig.6.2.

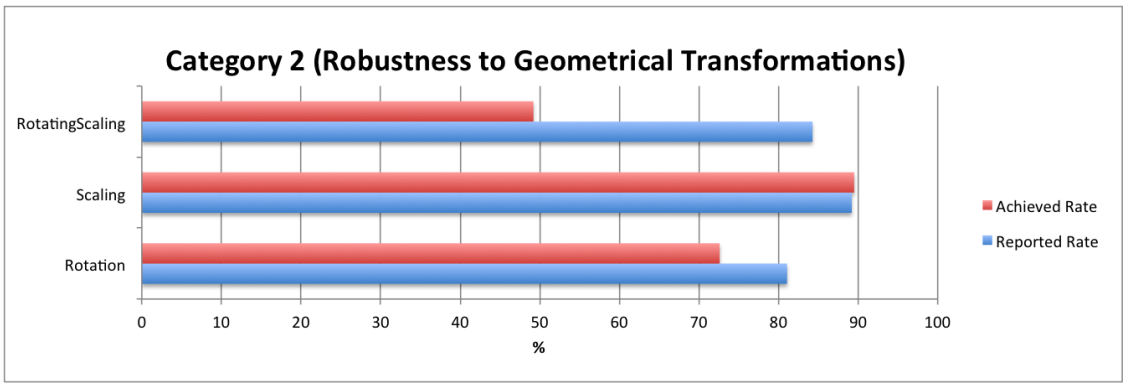


Figure 6.2: A graph showing the recognition rates for the second category of datasets

6.4 Robustness to Noise

The third category of datasets was also an other difficult category due to added noise and degradations. Also these datasets have 150 different models for each datasets. The recognition rates achieved for these datasets with our approach have surpassed those reported in the GREC'11 contest. A number of pre-processing techniques were applied on the test image prior to classification in order to get rid of noise or enhance the image quality. This shows that our approach is also very robust and efficient when handling noise. The following is a summary of results for the second category of datasets, Fig.6.3.

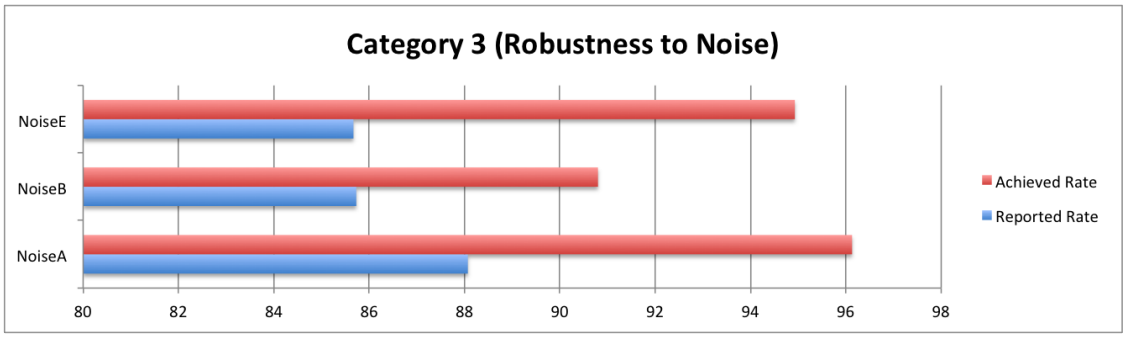


Figure 6.3: A graph showing the recognition rates for the second category of datasets

6.5 Discussion Summary

In comparison to the approach in their study [20, 40], the COSFIRE approach achieves robustness to scale, rotation and reflection. In their approach, in order to classify the test image they needed to rotate and scale the model image and create multiple feature vectors describing the same model image. In the approach presented in this study, only one feature vector is needed in order to describe the model image which makes it more efficient due to refraining from working out various feature vectors for each of the models. With this approach we achieved slightly better performance in the scaled datasets. However performance is less for the

rotated versions of the datasets.

We would like to examine and investigate the datasets which contained rotations further. The drop in recognition rate might be due to the fact that the list of of orientations used during configuration is not sufficient. Therefore, we would like to extend further, the experiments presented in this study by considering more rotations.

In this approach we do not evaluate the effectiveness of the configured COSFIRE filters. Since we do not evaluate the resultant operators there is a probability that two operators for two different model symbols might be very similar. This can be due to the fact that any two symbols can have similar geometric primitives. This can be countered by choosing a small subset of test images for evaluation.

Also, in this approach configuration of filters was done randomly by choosing patterns without any attempt at optimisation. Optimisation can be done by altering the COSFIRE filter's parameters in order to discriminate more between any two symbols. One example is the of concentric circles which are used in the configuration. By first analysing such parameters and by subsequently evaluating the attained COSFIRE operators before actually using them would increase the discriminative power during the classification phase.

7. Conclusion

The presented approach in this study is effective for symbol recognition with symbols with different levels of noise and geometric transformations. Also this approach, is not only effective for symbol recognition only but for any isolated pattern recognition due to the fact that no prior information about the pattern is used. This makes the presented approach a versatile one which can be used in various other applications.

7.1 Outlook

For future work, the next part of this study is to evaluate the effectiveness of the approach used so far on the last set of datasets. The fourth category of datasets consists of three different datasets called Set A, Set B and Set C which have 50, 100 and 150 different models respectively. These model symbols consist of both straight lines and arcs or circles. All datasets contain 50 test images per model symbol. Therefore the number of total test images per dataset varies where Set A contains 2500 testing images, Set B contains 5000 testing images and Set C contains 7500 test images. Scaling and orientation vary from those in the model through out each of the test symbols. Noise is introduced in the form of added noise or line thickening or line degradation. These types of noises are never applied together Fig. 7.1 illustrates a few samples.

The final results during the GREC'11 contest, associated with these datasets, are shown in the following table [20].

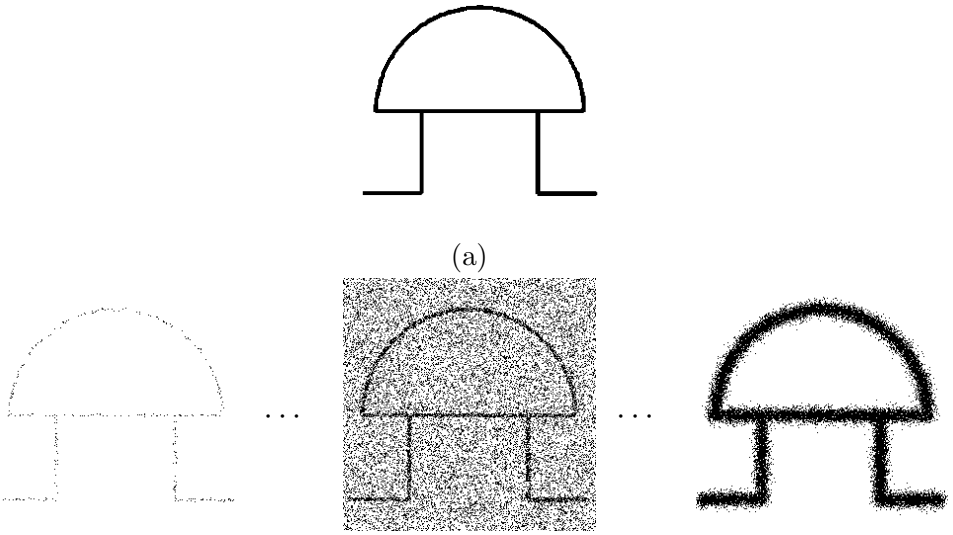


Figure 7.1: (b) (c) (d) Three test images for (a) a model symbol are illustrated from the Set C dataset

Table 7.1: Contest's global results. Adapted from [20]

Test Name	Recognition Rate
set 1 (50 models)	94.76 %
set 2 (100 models)	91.98 %
set 3 (150 models)	85.88 %

In this study, the experiments were implemented using a MATLAB environment. Furthermore, the experiment code was developed using a sequential approach. Future work on this regard may include the re-writing of experiments by adhering to a parallel model using different more efficient languages such as C++. This will help improve time effeciency in which COSFIRE operators are configured and applied, and the time in which test images are classified.

An interesting work would also be to evaluate experimental results using different values for a number of COSFIRE parameters such as alpha and sigma0. These two parameters control the tolerance of the mutual spatial arrangement of the involved contour parts.

A different approach can also be taken towards which classification methods to use. In this study a KNN algorithm was used to classify the test images. The distance metric used in the KNN algorithm was Euclidean. Different distance metrics can be implemented and investigated for classification. Also apart from the KNN algorithm, other classification techniques that can be considered.

Also, this approach's robustness and scalability to noise or transformations can be investigated against different types of symbol datasets such as characters, musical symbols, etc. Also for datasets which are comprised of more than one image per model, different classification methods such as artificial neural networks, graphs, support vector machines, can be used.

A. Appendix

A.1 Program Implementation

The implementation was done using MATLAB 2012b. In order to run the program implementation the image processing toolbox needs to be installed. The source code for the COSFIRE filter, around which this approach was developed, was provided [1]by Dr. George Azzopardi. It is included in the CD along with the other scripts.

The following subsections include a detailed description of each developed script along with it's code.

A.1.1 ExperimentParameters.m

Description

The "ExperimentParameters.m" script contains a list of variables which control the experiment's execution. The following is a description of each of these variables.

1. **LoadConfigurationFromFile**
    - Boolean switch which determines whether to load the configuration file from disk or not
2. **loadTrainingFromFile**
    - Boolean switch which determines whether to load the training file from disk or not

[1]Taken From `http://matlabserver.cs.rug.nl/cosfireweb/web/index.html`

3. **NFiltersPerProtoType**
    - Determines the number of COSFIRE filters to configure for each model symbol.
4. **minNumberOfTuplesPerFilter**
    - Determines the minimum number of tuples that must be achieved for each configured COSFIRE filters.
5. **minNumberOfDistinctOrientationsPerFilter**
    - Determines the minimum number of unique orientations detected for each configured COSFIRE filters.
6. **OS**
    - Determines which OS is being used, so as to switch between different path strings.
7. **ModelsFolder**
    - Specifies the name of the folder in which the model symbols are found
8. **ModelsExtension**
    - Specifies the model symbols' file extension
9. **TestingFolder**
    - Specifies the name of the folder in which the testing symbols are found
10. **TestingExtension**
    - Specifies the testing symbols' file extension
11. **WIN/OSX.COSFIREFolder**
    - Specifies the folder in which the COSFIRE filter code resides

12. **WIN/OSX.GaborFolder**
    - Specifies the folder in which the Gabor filter code resides
13. **WIN/OSX.DataSetsPath**
    - Specifies the path in which the data sets are found.
14. **WIN/OSX.ModelsDirectory**
    - Specifies the path in which the model symbols are found
15. **WIN/OSX.TestingDirectory**
    - Specifies the path in which the test symbols are found

Code

The code for this script is found on the next page.

A.1.2 Utilities.m

Description

The "Utilities.m" script contains a list of functions which are used through out the remaining scripts of the experiment. The following is a list of all the available functions alongside a description of what they do:

1. **SaveOperators()**
    - Saves a visualisation of the configured operator to file
2. **saveFalsePositive()**
    - Saves a visualisation of the failed recognitions to file
3. **printRhoList()**
    - Saves a visualisation of the selected rho list on the model image
4. **rhoAnalyser()**
    - Analyses the selected rho list against all the models to check whether all models can be properly configured
5. **SetupFolderEnviroment()**
    - Sets up directories in which to save files etc.
6. **createResultFolder()**
    - Creates a result folder in which to dump the results file
7. **createIntermediateFolder()**
    - Creates an intermediate folder in which checkpoints are saved to disk.
8. **ReadXMLV1()**

- Reads the XML file for the first category of datasets in order to determine if the recognition was a successful one or not.

9. **ReadXMLV2()**
    - Reads the XML file for the second and third category of datasets in order to determine if the recognition was a successful one or not.
10. **writeTrainingData()**
    - Write the training data(feature vectors) to an excel file
11. **visualizeTrainingData()**
    - Saves a visualisation of all the feature vectors togather
12. **appendElapsedTimeToFile()**
    - Adds the elapsed time to the final result
13. **appendTPFP()**
    - Saves to file the true positives or false positives
14. **appendResultToFile()**
    - Saves to file the final result for the experiment
15. **getCOSFIREFolder()**
    - Gets the folder in which the COSFIRE filter code resides
16. **getGaborFolder()**
    - Gets the folder in which the Gabor filter code resides
17. **getModelsDirectory()**
    - Gets the directory in which the models reside

18. **getTestingDirectory()**

    - Gets the directory in which the testing images reside

19. **Signal()**

    - Gives a signal when the experiment is done.

Code

The code for this script is found on the next page.

A.1.3 ConfigureOperators.m

Description

The "ConfigureOperators.m" script contains the logic by which for every model symbol in a dataset a number of COSFIRE filters are applied. This outputs a collection of COSFIRE operators which are then passed to the "TrainOperators.m" script.

Code

The code for this script is found on the next page.

A.1.4 TrainOperators.m

Description

The "TrainOperators.m" script contains the logic by which for every model symbol in a dataset, every COSFIRE operator yielded from the "ConfigureOperators.m" script, is re-applied. This outputs a collection of feature vectors describing each model, which are then passed to the "TestOperators.m" script.

Code

The code for this script is found on the next page.

A.1.5 TestOperators.m

Description

The "TestOperators.m" script contains the logic by which every test image in the dataset has a number of COSFIRE filters applied to it from which a feature vector is yielded. This feature vector is the cross referenced against the collection of feature vectors yielded from the "TrainOperators.m" script. For every test image, its corresponding XML file is read to determine whether it is a true positive or a false positive.

Code

The code for this script is found on the next page.

A.1.6 Experiment.m

Description

The "Experiment.m" script contains the logic by which the whole experiment is run. It calls the other scripts in order to successfully execute an experiment.

Code

The code for this script is found on the next page.

A.2 CD Contents

Further to this report, a CD was created containing other information that could not be included in this report. The following are the contents of the CD.

A.2.1 Dissertation document

The following two document formats of the dissertation are available on the CD

1. /Report/LukeAgius-080358084-Dissertation.pdf

A.2.2 Video

In the case of MATLAB IDE not being available for the reviewer of this project, a video has been added which shows the setup and execution of an experiment which is found in:

- /Video/SampleRun.mov

This sample experiment run shows the successful classification of 50 test images for the most simple dataset.

A.2.3 Implementation

A full implementation has been included in the CD which consists of:

1. COSFIRE Filter code
    - Implementation/COSFIREFilter/...
2. Matlab Scripts
    - Implementation/Scripts/ExperimentParameters.m
    - Implementation/Scripts/Utilities.m
    - Implementation/Scripts/ConfigureOperators.m

- Implementation/Scripts/TrainOperators.m
- Implementation/Scripts/TestOperators.m
- Implementation/Scripts/Experiment.m

A.2.4 Datasets

The datasets used for this dissertation have been included in the CD as well. The following tables, A.1 and A.2, list all the included datasets.

Table A.1: Available datasets in the CD

Category no.	Dataset name	Path
1	Sketches25f-Level1	/Datasets/Category1/Sketches25f-Level1/
1	Sketches25f-Level2	/Datasets/Category1/Sketches25f-Level2/
1	Sketches25f-Level3	/Datasets/Category1/Sketches25f-Level3/
1	Sketches25-Level1	/Datasets/Category1/Sketches25-Level1/
1	Sketches25-Level2	/Datasets/Category1/Sketches25-Level2/
1	Sketches25-Level3	/Datasets/Category1/Sketches25-Level3/
1	Sketches50f-Level1	/Datasets/Category1/Sketches50f-Level1/
1	Sketches50f-Level2	/Datasets/Category1/Sketches50f-Level2/
1	Sketches50f-Level3	/Datasets/Category1/Sketches50f-Level3/
1	Sketches50-Level1	/Datasets/Category1/Sketches50-Level1/
1	Sketches50-Level2	/Datasets/Category1/Sketches50-Level2/
1	Sketches50-Level3	/Datasets/Category1/Sketches50-Level3/
1	Sketches100f-Level1	/Datasets/Category1/Sketches100f-Level1/
1	Sketches100f-Level2	/Datasets/Category1/Sketches100f-Level2/
1	Sketches100f-Level3	/Datasets/Category1/Sketches100f-Level3/
1	Sketches100-Level1	/Datasets/Category1/Sketches100-Level1/
1	Sketches100-Level2	/Datasets/Category1/Sketches100-Level2/
1	Sketches100-Level3	/Datasets/Category1/Sketches100-Level3/
1	Sketches150f-Level1	/Datasets/Category1/Sketches150f-Level1/

Table A.2: Available datasets in the CD

Category	Name	Path
1	Sketches150f-Level2	/Datasets/Category1/Sketches150f-Level2/
1	Sketches150f-Level3	/Datasets/Category1/Sketches150f-Level3/
1	Sketches150-Level1	/Datasets/Category1/Sketches150-Level1/
1	Sketches150-Level2	/Datasets/Category1/Sketches150-Level2/
1	Sketches150-Level3	/Datasets/Category1/Sketches150-Level3/
2	Rotation	/Datasets/Category2/Rotation/
2	Scaling	/Datasets/Category2/Scaling/
2	RotationScaling	/Datasets/Category2/RotatingScaling/
3	Noise A	/Datasets/Category3/NoiseA/
3	Noise B	/Datasets/Category3/NoiseB/
3	Noise E	/Datasets/Category3/NoiseE/
4	setA	/Datasets/Category4/SetA/
4	setB	/Datasets/Category4/SetB/
4	setC	/Datasets/Category4/SetC/

A.3 How to run an experiment

Before running an experiment, the necessary scripts and files must be copied to disk in a dedicated directory. The following walkthrough is an example of how to setup an experiment for a windows environment.

A.3.1 File And Directory Setup

First, create a new directory called "Experiment" anywhere you like. In this folder copy the, "Datasets" and "Implementation" folders from the CD contents. The datasets folder holds the images that the experiment will go through while the implementation folder contains the experiment's scripts. The directory structure should look as follows:

- C:\Experiment\
- C:\Experiment\Datasets\
- C:\Experiment\Implementation\

Once that the directories has been set up and the correct files have been copied to those directories open MATLAB. Once MATLAB has been opened, input the path "C:\Experiment\Implementation\" in the Address field as shown in the following figure, Fig. A.1:

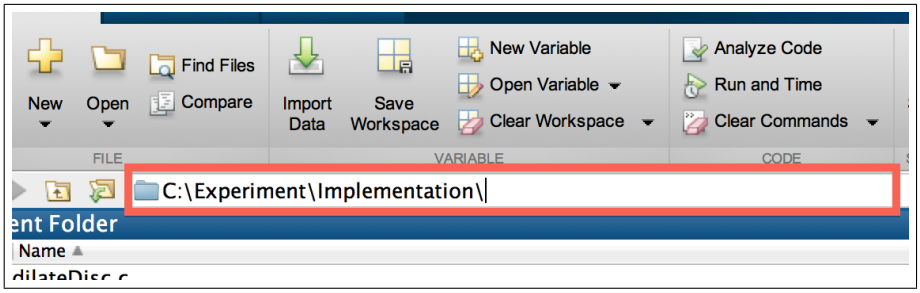


Figure A.1

Once that the correct path has been inputted in MATLAB's address field, the following files should be displayed in MATLAB's current folder section , Fig. A.2.

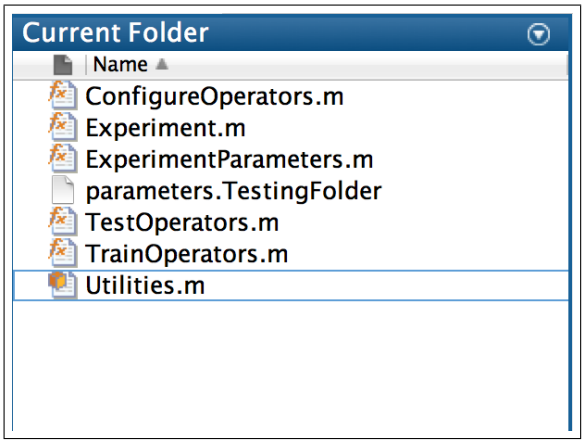


Figure A.2

Upon, having the correct "Current Folder" setup in MATLAB, continue by double clicking the script with the name "ExperimentParameters.m". This will open the parameters script for the experiment. In the next subsection we explain how to set up the parameters for an experiment.

A.3.2 Parameter Setup

Prior to the execution of an experiment, the parameters in the ExperimentParameters.m script must be setup properly. The first parameters that need to be set are the check point parameters. These parameters will determine whether the configuration or the training file should be loaded from disk or generated. The following is an example:

```
params.loadConfigurationFromFile   = 1;
params.loadTrainingFromFile        = 1;
```

The next parameters that should be set up are the NFiltersPerProtoType, minNumberOfTuplesPerFilter, and minNumberOfDistinctOrientationsPerFilter parameters. The following example we are instructing the code to create three filters for each symbol from which five tuples and two distinct orientations must be created at minimum.

```
params.NFiltersPerProtoType                          = 3;
params.minNumberOfTuplesPerFilter                    = 5;
params.minNumberOfDistinctOrientationsPerFilter      = 2;
```

Next, the type of OS needs to be specified. The permitted values for this parameter are "OSX" or "WIN". Each value correspond to either an Apple OSX or a Microsoft Windows environment. Afterwards we need to specify the directory path for the COSFIRE filter folder and that of the Gabor filters as well. Along with these paths, we need to also specify where the datasets are currently situated.

```
params.OS                   = 'WIN';
params.WIN.COSFIREFolder    = 'C:\COSFIREFilter\COSFIRE\';
params.WIN.GaborFolder      = 'C:\COSFIREFilter\Gabor\';
params.WIN.DataSetsPath     = 'C:\Datasets\Category 1\';
```

Finally, the name of the folders in which the model and test symbols reside needs to be specified. Along with this the extension file for each set of symbols needs to be specified as well. In the following example, we are specifying that the models folder is called 'models' which contains files of the type '.tif' and that the testing images folder is called 'NoiseB' which contains files of the type '.tiff'

```
params.ModelsFolder         = 'models';
params.ModelsExtension      = '.tif';
params.TestingFolder        = 'NoiseB';
params.TestingExtension     = '.tiff';
```

Note, that these names must correspond to the same folder names in the the dataset that is to be executed.

A.3.3 Execution

Once that the environment parameters are set up correctly, the following command needs to be entered in the MATLAB command window. This will set in motion all the steps in the experiment.

```
Experiment;
```

A.3.4 Output

The output for an experiment is similar to the following example. This output means that for example, test image file0.tiff has been classified to belong to model symbol012.tiff. Along side each classification, a result is added to indicate whether it was successful or not.

```
file0.tiff is closest to symbol012.tiff - Pass
file1.tiff is closest to symbol091.tiff - Pass
file2.tiff is closest to symbol029.tiff - Fail
file3.tiff is closest to symbol034.tiff - Pass
file4.tiff is closest to symbol014.tiff - Pass
```

Also after each experiment, the final results are shown in the following manner.

```
True Positives : 1000
False Positives  : 0
```

Bibliography

[1] Euclidean and euclidean squared. Available from
http://www.improvedoutcomes.com/docs/WebSiteDocs/Clustering/Clustering_Parameters/Euclidean_and_Euclidean_Squared_Distance_Metrics.htm.
Accessed on: 01/03/2013.

[2] Hidden markov models. Available from
http://digital.cs.usu.edu/~cyan/CS7960/hmm-tutorial.pdf.
Accessed on: 27/02/2013.

[3] International symbol recognition contest 2011. Available from
http://mathieu.delalandre.free.fr/projects/sesyd/symbols/isrc2011.html.
Accessed on: 15/11/2012.

[4] Lecture notes on data transformation. Available from
http://www.informatics.indiana.edu/predrag/classes/2005springi400/lecture_notes_4_1.pdf.
Accessed on: 01/05/2013.

[5] Matching with shape contexts. Available from
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/shape/sc_digits.html.
Accessed on: 21/03/2013.

[6] Median filter. Available from
http://homepages.inf.ed.ac.uk/rbf/HIPR2/median.htm.
Accessed on: 01/05/2013.

[7] Morphological dilation. Available from
http://homepages.inf.ed.ac.uk/rbf/HIPR2/dilate.htm.
Accessed on: 01/05/2013.

[8] Sketched symbols. Available from
http://mathieu.delalandre.free.fr/projects/sesyd/symbols/sketches.html.
Accessed on: 15/11/2012.

[9] Thinning. Available from
http://homepages.inf.ed.ac.uk/rbf/HIPR2/thin.htm.
Accessed on: 01/05/2013.

[10] N. Alajlan, I. E. Rube, M. S. Kamel, and G. Freeman. Shape retrieval using triangle-area representation and dynamic space warping. *Pattern Recognition*, 40(7):1911 – 1920, 2007.

[11] N. Alajlan, I. E. Rube, M. S. Kamel, and G. Freeman. Shape retrieval using triangle-area representation and dynamic space warping. *Pattern Recognition*, 40(7):1911 – 1920, 2007.

[12] S. Belongie and J. Malik. Matching with shape contexts. In *Content-based Access of Image and Video Libraries, 2000. Proceedings. IEEE Workshop on*, pages 20–26, 2000.

[13] K. Bertet, M. Coustaty, and J.-m. Ogier. Robust symbol recognition using a structural approach mathieu delalandre cvc (barcelona, spain).

[14] L. Cordella and M.Vento. Symbol recognition in documents: a collection of techniques. *International Journal on Document Analysis and Recognition*, 3(2):73–88, 2000.

[15] J. Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *Journal of the optical Society of America*, 2(7):1160–1169, 1985.

[16] J. G. Daugman. Complete discrete 2-d gabor transforms by neural networks for image analysis and compression. *IEEE Transactions on Acoustics, Speech, and Signal processing*, 36(7):1169–1179, 1988.

[17] M. Delalandre, P. Hroux, S. Adam, E. Trupin, and J.-M. Ogier. A statistical and structural approach for symbol recognition, using xml modelling. In T. Caelli, A. Amin, R. Duin, D. Ridder, and M. Kamel, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, volume 2396 of *Lecture Notes in Computer Science*, pages 281–290. Springer Berlin Heidelberg, 2002.

[18] M. Delalandre, E. Valveny, and J. Llados. Performance evaluation of symbol recognition and spotting systems: An overview. In *Document Analysis Systems, 2008. DAS '08. The Eighth IAPR International Workshop on*, pages 497–505, 2008.

[19] R. P. Duin and E. Pekalska. The science of pattern recognition. achievements and perspectives. In *Challenges for Computational Intelligence*, pages 221–259. Springer, 2007.

[20] E.Valveny, M.Delalandre, R.Raveaux, and B.Lamiroy. Report on the symbol recognition and spotting contest. *Revised Selected Papers of Workshop on Graphics Recognition (GREC), Lecture Notes in Computer Science (LNCS)*, 7423:1–11, 2012.

[21] G.Azzopardi and N.Petkov. A corf computational model of a simple cell that relies on lgn input outperforms the gabor function model. *Biological Cybernetics*, 106(3):177–189, 2012.

[22] G.Azzopardi and N.Petkov. Trainable cosfire filters for key point detection and pattern recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(2):490–503, 2012.

[23] C. Grigorescu and N. Petkov. Distance sets for shape filters and shape recognition. *Trans. Img. Proc.*, 12(10):1274–1286, Oct. 2003.

[24] F. C. A. Groen, A. C. Sanderson, and J. F. Schlag. Symbol recognition in electrical diagrams using probabilistic graph matching. *Pattern Recogn. Lett.*, 3(5):343–350, Sept. 1985.

[25] A. Hamada. *Structural recognition of disturbed symbols using discrete relaxation.* Rapport: Centre de Recherche en Informatique. Centre de Recherche en Informatique, 1991.

[26] P. Heider, A. Pierre-Pierre, R. Li, and C. Grimm. Local shape descriptors, a survey and evaluation. In *Proceedings of the 4th Eurographics conference on 3D Object Retrieval*, EG 3DOR'11, pages 49–56, Aire-la-Ville, Switzerland, Switzerland, 2011. Eurographics Association.

[27] H. Hse and A. R. Newton. Sketched symbol recognition using zernike moments. In *International Conference on Pattern Recognition*, pages 367–370, 2004.

[28] X. Jiang, A. Mnger, and H. Bunke. Synthesis of representative graphical symbols by computing generalized median graph. In A. Chhabra and D. Dori, editors, *Graphics Recognition Recent Advances*, volume 1941 of *Lecture Notes in Computer Science*, pages 183–192. Springer Berlin Heidelberg, 2000.

[29] J. P. Jones and L. A. Palmer. An evaluation of the two-dimensional gabor filter model of simple receptive-fields in cat striate cortex. *Journal of Neurophysiology*, 58(6):1233–1258, 1987.

[30] T. Kanungo, R. Haralick, H. Baird, W. Stuezle, and D. Madigan. A statistical, nonparametric methodology for document degradation model validation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1209–1223, 2000.

[31] P. KUNER and B. UEBERREITER. Pattern recognition by graph matching-combinatorial versus continuous optimization. *International Journal of Pattern Recognition and Artificial Intelligence*, 02(03):527–542, 1988.

[32] S.-W. Lee. Recognizing hand-drawn electrical circuit symbols with attributed graph matching. In H. Baird, H. Bunke, and K. Yamamoto, editors, *Structured Document Image Analysis*, pages 340–358. Springer Berlin Heidelberg, 1992.

[33] H. Ling and D. W. Jacobs. Shape classification using the inner-distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(2):286–299, Feb. 2007.

[34] J. Lladós. *Combining Graph Matching and Hough Transform for Hand-drawn Graphical Document Analysis: Application to Architectural Drawings.* 1997.

[35] 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*, GREC '01, pages 104–127, London, UK, UK, 2002. Springer-Verlag.

[36] B. T. . R. J.-Y. Luqman, M. M. Graphic symbol recognition using graph based signature and bayesian network classifier. *2009 10th International Conference on Document Analysis and Recognition*, page 13251329, 2009.

[37] S. Maji. A comparison of feature descriptors.

[38] B. Messmer and H. Bunke. Automatic learning and recognition of graphical symbols in engineering drawings. In R. Kasturi and K. Tombre, editors, *Graphics Recognition Methods and Applications*, volume 1072 of *Lecture Notes in Computer Science*, pages 123–134. Springer Berlin Heidelberg, 1996.

[39] F. Mokhtarian, S. Abbasi, and J. Kittler. Efficient and robust retrieval by shape content through curvature scale space. pages 35–42, 1996.

[40] N. Nayef and T. M. Breuel. On the use of geometric matching for both: isolated symbol recognition and symbol spotting. In *Proceedings of the 9th international conference on Graphics Recognition: new trends and challenges*, GREC'11, pages 36–48, Berlin, Heidelberg, 2013. Springer-Verlag.

[41] T. Y. Ouyang and R. Davis. A visual approach to sketched symbol recognition. In *Proceedings of the 21st international jont conference on Artifical intelligence*, IJCAI'09, pages 1463–1468, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[42] S. Thirumuruganathan. A detailed introduction to k-nearest neighbor (knn) algorithm @ONLINE, May 2010.

[43] K. Tombre, S. Tabbone, and P. Dosch. Musings on symbol recognition. In W. Liu and J. Llads, editors, *Graphics Recognition. Ten Years Review and Future Perspectives*, volume 3926 of *Lecture Notes in Computer Science*, pages 23–34. Springer Berlin Heidelberg, 2006.

[44] J. Wang, X. Bai, X. You, W. Liu, and L. J. Latecki. Shape matching and classification using height functions. *Pattern Recogn. Lett.*, 33(2):134–143, Jan. 2012.

[45] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg. Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, 14(1):1–37, Dec. 2007.

[46] J. Xie, P.-A. Heng, and M. Shah. Shape matching and modeling using skeletal context. *Pattern Recogn.*, 41(5):1756–1767, May 2008.

[47] F. Yang and F. Yang. Character recognition using parallel bp neural network. In *Audio, Language and Image Processing, 2008. ICALIP 2008. International Conference on*, pages 1595–1599, 2008.

[48] S. Yoon, G. Kim, Y. Choi, and Y. Lee. New paradigm for segmentation and recognition. In *Workshop on Graphics Recognition (GREC)*, pages 216–225, 2001.

[49] S. Zesheng, Y. Jing, J. Chunhong, and W. Yonggui. Symbol recognition in electronic diagrams using decision tree. In *Industrial Technology, 1994., Proceedings of the IEEE International Conference on*, pages 719–723, 1994.

B. Evaluation

I am very satisfied with the study I have carried out during this dissertation. Through out this project I have gained knowledge in the area of pattern recognition and also managed to implement a classifier around the COSFIRE descriptor whilst learning a new language enviroment, MATLAB. Previously, the COSFIRE filter was used to the detection of vascular bifurcations, recognition of handwritten digits and detection and recognition of traffic signs in complex scenes. In this study I have contributed academically by evaluating the effectiveness of COSFIRE Filters in the recognition of architectural and electrical symbols.

The project plan time line deviated slightly from the original plan resulting in, processing only the first three categories of datasets. However, even though I have delayed in some of the tasks, due to unpredictable circumstances, with the help of my supervisor I have approached this dissertation in a methodical way. As a result, this dissertation helped me to mature both technically and behaviourally.

From a technical point of view, I have broadened my technical knowledge in an area, pattern recognition, which i had no experience in. From a behavioural point of view, I have improved my writing skills by being concise but still getting the point across. All in all, I have found the research process to be stimulating and this project gave me a lot of satisfaction which encourages me to further my studies.