An Expert System for the Recognition of General Symbols

by

Maher M. Ahmed

M. Sc., Queen's University, 1994
M. Sc., Cairo University, 1988
B. Sc., Cairo University, 1984

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY in

THE FACULTY OF GRADUATE STUDIES 
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

We accept this thesis as conforming to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

© Maher M. Ahmed, 1999
In presenting this thesis in partial fulfilment of the requirements for an advanced
degree at the University of British Columbia, I agree that the Library shall make it
freely available for reference and study. I further agree that permission for extensive
copying of this thesis for scholarly purposes may be granted by the head of my
department or by his or her representatives. It is understood that copying or
publication of this thesis for financial gain shall not be allowed without my written
permission.

Department of Electrical and Computer Engineering
The University of British Columbia
Vancouver, Canada

Date Oct 4, 1999
Abstract

This thesis addresses the problem of automatic recognition of any hand written symbol. The number of different styles of handwritten symbols demonstrates the difficulties that an automatic recognizer must cope with. For example, Some handwritten styles of capital A letters are:

The main technical approaches for solving the problems in recognizing patterns are statistical pattern recognition and structural pattern recognition.

Statistical pattern recognition systems use pixel-features for recognition. Some of these features are moments, histograms, Fourier transforms, and the percentage of ink pixels in different zones. Although statistical pattern recognition techniques, including Artificial Neural Networks (ANNs), carry a high recognition rate, there are some disadvantages that are in these systems. Disadvantages include the requirements of a very large number of training data, and the inability to justify its answer. In addition, the output is only a classification and not a description of the actual pattern.

As opposed to statistical pattern recognition techniques, structural pattern recognition techniques extract commonly used descriptions of the patterns (structural-features). These features include loops, end points, and arcs. After extracting these similarities, the system then finds the common relationships among these structural-features (descriptions).

In this research, the structural pattern recognition approach was used for developing an expert system that extracts structural-features (descriptions) from the symbol at each stage of recognition. The developed system enabled us to automatically recognize handwritten symbols, assuming that the symbols are in their isolated forms. This system is unique in that it is not limited for a specific application, but it can be used to recognize any general symbol of any language.

To obtain a representation of a symbol the system performs four basic steps. First, the system adjusts the symbol by rotating it around its central point until its principal axis aligns with the vertical axis or having a multiple of 20° to the vertical axis. Second, the system scales the symbol to a predefined size.

The third step is to thin the symbol. A novel rule-based system for thinning is developed in this research. The resultant thinned image is composed of the central lines of the image. Finally, the last step involves extracting and describing the thinned symbol in terms of strokes. These strokes will be approximated by a set of line segments.

The resulting representation of the symbol is compared with different stored models of the different symbols in the system knowledge base. For each symbol many models are
stored. The results of our system depend on a certain threshold. Using a low threshold will decrease the space for this symbol, increase the rejection rate and increase the recognition rate.

The system was tested with 5726 handwritten English characters. When the system learned an average of 97 models per symbol and used a low threshold, the recognition rate was 95% and the rejection rate was 16.1%. The tested data were all test data (binary data) taken from the Center of Excellence for Document Analysis and Recognition (CEDAR) database. When the threshold is 100, the recognition rate was 87.6% and the rejection rate was 0%.

The recognition rate of our system can be increased by storing more models for each symbol or by increasing the rejection rate. The system is capable of learning new symbols by simply adding models for these symbols to the system knowledge base. The system is implemented using C++ running on a 120 MHz Pentium PC.
# Table of Contents

Abstract .................................................................................................................. ii  
Table of Contents ................................................................................................... iv  
List of Tables ......................................................................................................... vi  
List of Figures .......................................................................................................... vii  
Chapter 1 .................................................................................................................. 1  
Introduction ............................................................................................................. 1  
1.1 Motivation ........................................................................................................... 1  
1.2 Problem Definition ............................................................................................ 1  
1.3 Applications ....................................................................................................... 4  
1.4 Thesis Objective ................................................................................................ 6  
1.5 Recognition Methods ......................................................................................... 6  
1.6 Contributions ..................................................................................................... 8  
1.7 Thesis Overview ................................................................................................ 8  
Chapter 2 .................................................................................................................. 10  
Literature Overview ................................................................................................ 10  
2.1 Introduction ........................................................................................................ 10  
2.2 Different Pattern Recognition Tools ................................................................... 11  
2.2.1 Expert Systems .............................................................................................. 11  
2.2.2 Artificial Neural Networks .......................................................................... 12  
2.3 The Main Pattern Recognition Techniques ...................................................... 18  
2.3.1 Statistical Pattern Recognition ..................................................................... 19  
2.3.2 Structural Pattern Recognition .................................................................. 32  
2.4 Preprocessing ..................................................................................................... 37  
2.4.1 Thresholding ................................................................................................. 37  
2.4.2 Noise ............................................................................................................... 37  
2.4.3 Overlapping ................................................................................................. 38  
2.4.4 Rotation and Slant Correction .................................................................... 40  
2.4.5 Segmentation ............................................................................................... 41  
2.5 Postal code and Address Recognition .................................................................. 42  
2.6 Summary ........................................................................................................... 43  
Chapter 3 .................................................................................................................. 45  
Thinning ................................................................................................................... 45  
3.1 Introduction ........................................................................................................ 45  
3.1 Thinning Algorithms Survey ............................................................................. 45  
3.2 Good Thinning Characteristics .......................................................................... 49  
3.3 The Central Line Thinning Rule-Based System ................................................ 49  
3.4 Edge Detection One-Pass Sequential Thinning ............................................... 59  
3.4.1 System Performance .................................................................................... 60  
3.4.2 Explanation of the Rules .............................................................................. 62  
3.5 Results ............................................................................................................... 64  
3.6 A Combined Thinning Method ......................................................................... 65  
3.7 The Speed of the Developed Thinning Algorithm .............................................. 68  
3.8 Summary ........................................................................................................... 69  
Chapter 4 .................................................................................................................. 70  
The Developed System ............................................................................................. 70  
4.1 Introduction ........................................................................................................ 70  
4.2 The Developed System ..................................................................................... 71  
4.2.1 The Rotation Stage ...................................................................................... 74  
a) The Central Point ............................................................................................... 74  
b) The Principal Axis ............................................................................................. 74  
4.2.2 The Scaling Algorithm ............................................................................... 74  
4.2.3 The Thinning Algorithm ............................................................................. 77  
4.2.4 Symbols Representation .............................................................................. 78
List of Tables

Table 1. The number of misrecognized symbols (handwritten English letters) for each method............... 32
Table 2 Thinning speed versus the maximum thickness of a symbol in a document of size 480 x 576 pixels. .................................................................................................................. 68
Table 3 The accuracy obtained for each symbol and the number of test symbols.................................. 87
Table 4 The accuracy obtained for each symbol and the number of test symbols...................................... 88
Table 5 The error rate versus the rejection rate ......................................................................................... 88
Table 6 Recognition rate in percentage versus the number of models for the letter ‘A’ when symbol rejection is not allowed................................................................. 97
Table 7 Recognition rate in percentage versus the number of models for the letter ‘A’ when symbol rejection is allowed........................................................................................................... 97
Table 8 The percentage of the recognition rate versus the number of models for the digits when symbol rejection is not allowed........................................................................................................... 97
Table 9 The percentage of the recognition rate versus the number of models for the digits when symbol rejection is allowed........................................................................................................... 97
List of Figures

Figure 1 Linearly separated classes. .......................................................... 13
Figure 2 Non-linearly separable classes. .................................................. 14
Figure 3 An example of a digit written in 7x5 pixels. ............................... 15
Figure 4 An example of a 3-layer ANNs Perceptron for digit recognition. .... 15
Figure 5 The symbol is divided into 16 regions. ....................................... 26
Figure 6 Vertical and horizontal histogram distributions of the symbol are divided into 16 slots. ......................................................... 28
Figure 7 A symbol before thinning (a) and after thinning (b). .................. 29
Figure 8 An example of a transformational grammar .................................. 33
Figure 9 An example of context sensitive grammar ................................... 33
Figure 10 two overlapped objects and their distance transform. ................. 39
Figure 11 The thinning rules with θ = 0°. ............................................... 50
Figure 12 The special case conditions relevant to 2-pixels thickness lines. .... 51
Figure 13 The original symbols ............................................................... 51
Figure 14 Thinning by applying the 16 rules with θ = 0°. ......................... 52
Figure 15 Thinning by applying the 16 rules with θ = 90°. ...................... 52
Figure 16 Thinning by applying the 16 rules with θ = 180°. .................... 53
Figure 17 Thinning by applying the 16 rules with θ = 270°. .................... 53
Figure 18 Thinning by applying 4 passes (16 rules in each pass) with θ = 90°, 180°, and 270° in second, third, and fourth pass, the 4 passes are repeated in each iteration. .............................................................. 54
Figure 19 Thinning by applying 16X4 rules in each iteration without testing for the special cases of 2-pixels thickness lines. ......................... 55
Figure 20 Thinning by using our new developed system which uses a parallel one-pass 16X4 rules and testing for the special cases of 2-pixels thickness lines. .............................................................. 56
Figure 21 Step by step thinning. ............................................................... 58
Figure 22 Thinning by using the ZS method. .......................................... 59
Figure 23 Thinning: (a) an image, (b) by applying the ZS method, and (c) by applying our developed parallel one-pass 16X4 rules. ................. 59
Figure 24 Different symbols .................................................................... 60
Figure 25 Symbols processed from left to right going from top to bottom and concerning: (a) a large number of branches, (b) a medium number of branches and (c) a small number of branches. ......................... 61
Figure 26 Symbols processed from right to left going from top to bottom and concerning: (a) a large number of branches, (b) a medium number of branches, and (c) a small number of branches. ......................... 61
Figure 27 Symbols processed from left to right going from bottom to top and concerning: (a) a large number of branches, (b) a medium number of branches, and (c) a small number of branches. ......................... 61
Figure 28 Symbols processed from right to left going from bottom to top: (a) a large number of branches, (b) a medium number of branches, and (c) a small number of branches. ......................... 61
Figure 29 The sequential thinning rules. .................................................. 62
Figure 30 A document that has Arabic, English, and Chinese characters. ........ 65
Figure 31 (a) The document after thinning using the developed one-pass 16x4 rules parallel method, (b) part of the image is enlarged. .............................................................. 66
Figure 32 (a) The document shown in Figure 31 after removing the extraneous pixels by applying the one-pass sequential thinning algorithm, (b) part of the image is enlarged. .............................................................. 67
Figure 33 Thinning of symbols that have different thickness. ...................... 68
Figure 34 The 16 possible images in a zone of 10 x10 pixels. ..................... 72
Figure 35 The structural pattern recognition steps. ................................... 73
Figure 36 A symbol after scaling to different dimensions and using a scaling threshold = 0.2: (a) the original symbol, (b) 32X32 size, (c) 48X32 size, (d) 24X16 size, (e) 8X8 size, and (f) 8X4 size. ........ 77
Figure 37 Scaling a symbol with different scaling threshold values and using 32X32 size: (a) the original symbol, (b) threshold = 0, (c) threshold = 0.2, (d) threshold = 0.4, (e) threshold = 0.6, (f) threshold = 0.8, and (g) threshold = 1.0. ......................... 77
Figure 38 The different direction possibilities. ........................................... 78
Figure 39 Processing of a symbol: (a) A symbol, (b) after rotation, (c) after scaling, (d) after thinning, (e) after marking the special pixels, and (f) after isolating the symbol strokes and coding each pixel. ... 79
Figure 40 Different mappings of a symbol by using different segment lengths of: (a) 1-pixel, (b) 2-pixels, (c) 3-pixels, (d) 4-pixels, (e) 5 pixels, (f) 6-pixels, (g) 7-pixels, (h) 8-pixels, (i) 9-pixels, and (j) 10 pixels. .......................................................... 80
Figure 41 Some models used by the system and the number of models used for each of these symbols. ... 84
Figure 42 Some models used by the system for the letter A. .......................................................... 86
Figure 43 A document processing. .................................................................................................. 91
Figure 44 Recognition rate in percentage versus the number of models for the letter ‘A’ when symbol rejection is not allowed. .................................................................................. 97
Figure 45 Recognition rate in percentage versus the number of models for the letter ‘A’ when symbol rejection is allowed. .................................................................................. 97
Figure 46 The number of allowed symbols and the number of possible symbols versus the number of zones. .............................................................................................................. 99
Figure 47 Example of different symbols. ......................................................................................... 99
Figure 48 The symbols after thinning. .............................................................................................. 100
Figure 49 The symbols after thinning and removing the extraneous pixels. .................................. 100
Figure 50 The processing time for extracting and coding the strokes of the symbols versus the number of strokes. ........................................................................................................ 100
Figure 51 The time required to convert the strokes of the eight symbols shown in Fig 47 versus the line segments. ........................................................................................................ 101
Acknowledgements

First and foremost, I would like to thank God for His everlasting mercy & compassion.

I would like to take this opportunity to thank the many people who have helped me in big and small ways in the preparation of this thesis.

I would like to acknowledge Dr. Rabab Ward for her invaluable comments and support during the course of this thesis. She has provided me with priceless suggestions and ideas that contributed a great deal in developing this thesis. She has given me her constant support and encouragement throughout the course of this thesis.

Finally, I owe my parents a debt of appreciation that I will never be able to repay.
Chapter 1

Introduction

1.1 Motivation

It is necessary to develop representational and computational techniques for the automatic processing of a variety of symbols. Such symbols are widely used in documents to communicate information including letters, digits and diagrams. Automated symbol recognition and understanding have great economical relevance, as evidenced by the high industry participation in many conferences on document analysis and recognition.\(^1\) An important pattern recognition application is the postal code recognition.\(^2\)

Understanding a general symbol involves two activities: symbol recognition and symbol arrangement analysis. The current research focuses on the development and implementation of a system that can recognize any handwritten symbol. Examples of these symbols are digits, characters, electrical circuit symbols, and mathematical symbols.

In addition to the obvious advantage of such research, it will provide us with a machine to assist visually disabled individuals. The research is also valuable for converting a handwritten document into a clearer typed one.

1.2 Problem Definition

Character Recognition or Optical Character Recognition (OCR) is the process of converting scanned images of machine printed or handwritten text (digits, letters, and symbols) into a computer processed format (such as ASCII).

There are two main pattern recognition problems: on-line and off-line pattern recognition depending on whether the characters are written before or during the recognition process. OCR falls under the off-line pattern recognition category.
The objective of on-line handwriting recognition is to recognize the symbols while they are being written. Such systems should be fast enough to keep up with the writing speed. A survey of a state of the art of on-line handwriting recognition is described by Tappert et al. (3) Generally, each symbol consists of one or more strokes. A stroke is defined as the writing from pen down to pen up.

The features in on-line pattern recognition might be based on the static as well as the dynamic properties of a character. A decision tree can be used to classify characters. For example, characters f, g, j, p, q, y, and z all have a descender. However, j is the only character that is associated with a dot. Other features include time sequence of zones, directions, extremes and curves.

Curve fitting (as a function of time) is used for matching two symbols. Stroke codes are used to classify sub-parts of a character. The whole character is classified based on a unique sequence of classified sub-parts.

Markov models are suitable for dynamic information. A first order Markov model uses eight states corresponding to eight pen-tip directions. An example of automatic recognition of unconstrained on-line handwritten text by using hidden Markov models is described by Nathen et al. (4)

Off-line handwriting recognition is performed after writing is completed, typically on scanner images. Off-line data captures is the process that occurs when the data is captured some time after the writing is done. A survey of the state of the art of off-line character recognition (for Arabic) is provided by Amine. (5)

On-line recognition has the following characteristics as opposed to off-line recognition:

- It captures dynamic information of the writing process such as pen velocity, pressure, and pen direction (up or down).
- It uses a one-dimensional string of data and not a two-dimensional image as in the case for off-line recognition.
In This research, we are interested in studying off-line recognition of isolated symbols.

Digital images consist of a set of pixels arranged in a two dimensional array. Each pixel has a value that represents the brightness or the color at this point of the image. Digital images are approximations of the scene being represented. The number of pixels reflects its resolution and the number of levels is called quantization.

Bi-level images contain pixels with only two grey values: 1 (black) and 0 (white). Images in which the important information consists of lines are called bi-level line images. The lines’ directions, lengths and the relationships among them contain the vital information in these images. Examples of line images are text, maps, graphs, charts, signatures and fingerprints. In these images, the geometry of the objects in the image conveys the information and not the relationship between the pixel grey levels (these grey levels are important in pictures). In this research, we are interested in analyzing bi-level line images.

Typical input to a recognizer system is a binary image (in a bit map “.bmp” format). Generally, these images are produced by compressing the range of grey levels to two levels. A binary image can be obtained by optically scanning a typical document page or by use of a camera. A typical scanning resolution that is suitable for text recognition is 300 dots per inch (DPI).

Recognition of unconstrained handwritten text can be difficult because of the large number of different styles of symbols. Some handwritten styles of capital A letters are

![Characters](image)

Characters may not be reliably isolated especially when
the text is handwritten cursive as for example

Problems also arise from noise, slant and rotations. Other problems arise from symbols that are overlapped as well as discontinuous ones. Another problem is the presence of graphics within the document. Thus, image separation from text may be useful in speeding up the process of recognition.

Some characters and digits may look alike. Examples of these characters are \{V,U\}, \{C,L\}, \{Z,2\}, \{n,h\}, \{1,1\}, \{S,5\}, and \{0,O\}. The context sometimes provides the necessary information to distinguish between them.

There are some similarities between symbols and sentences. A sentence or a symbol consists of primitives. Examples of these primitives in a sentence include a verb, noun, subject and object. In a symbol, these primitives include an upper stroke, a left curve and a circle. Any sentence or symbol has rules that determine the right place of each primitive. My research concentrates on the recognition of general symbols by using models for these symbols.

1.3 Applications

Hundreds of OCR systems have been developed since the 1950s and some are commercially available today. Most of the commercial OCR systems are task-specific readers. A task-specific reader handles only specific document types. High performance reading machines are described by S.N. Srihari. Some of the most common task-specific readers are:
• Mail sorter: postal codes and addresses in mail.

• Data entered in forms, e.g., tax forms, job applications.

• Bills.

• Bank cheques.

• Detecting forged signatures.

• Airline passenger tickets.

• Passports.

These readers usually utilize custom-made image lift hardware that captures only a few predefined document regions. For example, a bank cheque reader may just scan the currency amount field and a postal OCR system may just scan the address block on a mail piece. Such systems emphasize high throughput rates and low error rates. The character recognizer in many task-specific readers is able to recognize machine-printed text and hand-printed text.

Many character recognizers are based on mathematical formalisms that minimize a measure of misclassification. Artificial neural networks employ mathematical minimization techniques and are used in commercial OCR systems. Recognition rates for machine-printed characters can reach over 99%, but handwritten character recognition rates are typically lower because every person writes differently, as reported by S. Lam in the Center of Excellence for Document Analysis and Recognition, State University of New York at Buffalo. ANNs have a type of error that is based on training with an inappropriate data, which is difficult to overcome.

Other more general systems capture an image of a document page and separate the page into text regions and non-text regions. Non-text regions such as graphics and line drawings are often saved separately from the text and associated recognition results. Text regions are segmented into lines, words, and characters and the characters are passed to
the recognizer. Recognition results are in a format that can be processed by the application software.

1.4 Thesis Objective

There are many implemented systems for pattern recognition. Each system addresses one specific language or symbol set (e.g., English, Arabic or Chinese characters or mathematical, musical or electrical symbols).

The purpose of this research is to develop a system that can recognize any general symbol by accessing the appropriate database models and using only one algorithm. Examples of these symbols are characters in any language, mathematical symbols or electrical circuit symbols.

The author uses structural pattern recognition techniques. Some symbols are described in the system knowledge base. Structural descriptions of symbols can be formulated by a set of rules. Hence, the suggested tool to implement this technique is an expert system. Expert systems are used efficiently in many applications. The advantages of using expert systems (also known as rule-based systems) are their modularity and inference capability. The algorithm is independent from the database. The database contains rules. There could be more than one database. Each database has rules that are suitable for certain applications. The rules can be easily modified, deleted or added.

Expert systems can be easily implemented by using the PROLOG language. Examples of expert systems previously implemented by the author are: an expert system for teaching students electrical circuits and expert systems for understanding Arabic sentences.

1.5 Recognition Methods

Many algorithms and techniques are useful in analyzing document images. These include noise filtering, thinning, line detection, segmentation, feature recognition and classification, pattern matching and semantic interpretations.
OCR is the process of converting scanned images of machine printed or handwritten text into a computer processed format (such as ASCII). The techniques for solving the OCR problems fall under two main approaches: statistical pattern recognition and structural pattern recognition.

Statistical pattern recognition systems use some pixel-features for recognition. Examples of these features include: moments, histograms, Fourier transforms, and the percentage of ink pixels in different zones. These systems have an upper limit of recognition rate determined by Bayes decision theory. Bayes theory minimizes the probability of error and determines the best boundary regions for each class in the feature space.

Statistical pattern recognition techniques, including Artificial Neural Networks (ANNs), have the following disadvantages: the requirement of a large number of training data and the inability of justifying the answer. In addition, the output is only a classification and not a description of the pattern.

ANNs have been used successfully for extracting and clustering symbol features. These networks are composed of several layers of interconnected elements. Each element (neuron) computes a weighted sum of its inputs and transforms it into an output using a nonlinear function. In the learning phase, the weight of each connection is modified until the desired outputs are obtained. ANNs have the disadvantage of requiring retraining (or redesigning the system) every time a new symbol is introduced to the system. As for example, if a neural network is constructed for recognition of the digits 0 to 9, then it is very difficult to modify it to make it capable of recognizing letters. There is no guarantee of a certain recognition rate for any statistical pattern recognition method.

Structural pattern recognition techniques extract commonly used structural features (descriptions) of the patterns such as loops, end points, and arcs, then find the relationships among these descriptions. Structural pattern recognition techniques can justify the answer similar to human reasoning by listing the reasons of why the symbol is recognized as such. Although they have a better recognition rate estimate, they are usually slower than statistical pattern recognition systems.
1.6 Contributions

In this research, the author used the structural pattern recognition approach for developing an expert system for symbol recognition.\(^{15}\) The system extracts descriptions from the symbol at each stage of the recognition. The system maps similar handwritten symbols to the same representation. The proposed system is not designed for a specific application but can be used for general symbol recognition. The system uses the structural pattern recognition technique for modeling symbols by a set of short straight lines (line segments). The system performs four basic steps before modeling the symbol. First, the system adjusts the symbol by rotating it around its central point \((C_r, C_c)\) (where \(C_r = \frac{1}{n} \sum x, \ C_c = \frac{1}{n} \sum y\) for an \(n \times n\) image) until its principal axis aligns at a certain angle with the vertical axis (0° or having a multiple of 20°). Second, the system scales the symbol to a predefined size (selected by the user or calculated by the program). Third, thinning is performed. Thinning is accomplished by using a novel thinning algorithm developed by the author and R. Ward.\(^{16, 17}\) After that, the system extracts and describes the thinned symbol in terms of strokes. Finally, each stroke is approximated by a set of line segments.

Analysis and results for the recognition of a document that has 5762 different handwritten symbols are described. The tested data are all binary test data taken from the Center of Excellence for Document Analysis and Recognition (CEDAR) database. When the system has stored some models for each symbol (an average of 97 models/symbol), the recognition rate is 95% and the rejection rate is 16.1%. Storing more models will increase the recognition rate. The recognition rate can also be increased by increasing the rejection rate. The system is capable of learning new symbols by simply adding their models to the system knowledge base. The system is implemented using C++ running on a 120 MHz Pentium PC.

1.7 Thesis Overview

The thesis consists of five chapters. The first chapter is an introduction. Chapter two presents an overview of different algorithms and techniques for symbol analysis and
recognition. Chapter three provides a study of various existing thinning algorithms as well as two novel thinning systems that were developed as a result of this research. Chapter four describes the details of the developed symbol recognition system and results. Finally, chapter five offers the conclusions and suggestions for future work.
Chapter 2

Literature Overview

2.1 Introduction

The study of machine simulation of human understanding of images has been attracting researchers as early as 1929\(^1\). The aim is to develop methods for computers to interpret images and characters in particular. Character Recognition or Optical Character Recognition (OCR) is the process of converting scanned images of machine printed or handwritten text (digits, letters, and symbols) into a computer processed format (such as ASCII).

Computer images are bi-level images, multi grey-level or coloured images. They can be stored in different formats. These formats include JPEG (Joint Photographic Experts Group), GIF (Graphics Interchange Format), TIFF (Tagged Image File Format), PBM (Portable Bit Maps), PPM (Portable Pix Maps), and PGM (Portable Gray Maps).

Different tools are used to implement pattern recognition systems. Recent tools for solving the pattern recognition problems include expert systems and Artificial Neural Networks (ANNs).

Different methods are used to recognize characters. Pattern recognition techniques fall under two main categories: statistical pattern recognition and structural pattern recognition.

Pattern recognition methods typically have problems with eliminating noise from images, as well as segmenting connected and overlapped symbols. Noise reduction may be achieved by simply applying suitable filters to the original image. Segmentation on the other hand is an open problem and an active area of current research. Segmentation, technically speaking, is the breaking up of an image into meaningful sub-images or primitives. Segmentation can be used, as a preprocessing step, for example, for breaking up a connected word into its characters, or a human face into its parts (mouth, nose, etc).
In this research, an expert system (or a rule-based system) is used to solve the off-line symbol recognition problems. Structural pattern recognition techniques are utilized. Also, the image is assumed to be a noise-free bi-level image and to contain only isolated handwritten symbols.

2.2 Different Pattern Recognition Tools

2.2.1 Expert Systems

Expert systems are suitable for solving pattern recognition problems. They have the advantages of flexibility and modularity. Knowledge is distributed over rules which are independent of each other. An expert system is a computer program that performs a specific task normally carried out by a human expert. The system should be able to draw conclusions without examining all possible solutions. The system should learn and use new knowledge efficiently. The system should be able to interact with the user and show how the conclusion was reached.

Expert systems have the ability to manipulate problem statements and integrate relevant pieces of knowledge from the knowledge base using heuristic techniques. Expert systems are problem-solving programs that use their knowledge in flexible ways. Expert systems can reason with inexact knowledge which might be declarative or procedural. Expert systems could learn by being told, by induction from examples, by parameter adjustment, or by observation and discovery. Expert systems are only appropriate for situations when the human expertise can be written explicitly in a formal language. Applications not suitable for expert systems are those that are deterministic such as solving a quadratic equation. Expert systems should perform difficult tasks comparable to human experts. Expert systems can handle problems through symbolic manipulation. They select the best answer through a set of values called certainty factors. A central problem lies in establishing an inference mechanism utilizing the different rules and applying it to the knowledge base. The user can ask how the conclusion was reached and what rules were applied. The expert system should be able to answer and explain how the answer was derived.
A program containing knowledge about the problem is called a knowledge-based system. The difference between knowledge-based systems and other systems is that knowledge-based systems do not imbed the knowledge within, for example, a computer program, rather they represent the knowledge explicitly in higher-level forms. These forms include rules and facts. Knowledge-based systems are close to the way people perceive problem solving to be. They have an inference engine for reasoning with these rules and facts.

The semantic network is one of the oldest scheme used in artificial intelligence for representing knowledge. A semantic network is described by a graph that is used for representing objects and the relations between them. The objects are called nodes. The relations between the objects are expressed as directed arcs.

Another method for representing knowledge is a data structure known as the frame. Frames perform the same function as semantic networks in addition to representing contextual information about objects, concepts, or events. A frame consists of a series of slots where each slot contains a fact.

2.2.2 Artificial Neural Networks

The development of Artificial Neural Networks (ANNs) has progressed rapidly as a result of their ability to mimic natural intelligence. ANNs have been used successfully for recognizing symbols. Their drawbacks are that they cannot justify the answers, as well, if we want to teach the system new patterns, the whole training process would have to be repeated. Another disadvantage of using ANNs is that it is not easy to add existing pre-acquired knowledge to the system.

A neural network is a system composed of many simple processing elements operating collectively. The function of each element is determined by the network structure and connection strengths. Some basic ANNs are described by Lippmann. Digital artificial neural networks are described by Kung. ANNs are classified into two main groups: supervised and unsupervised networks. While supervised ANNs use the output as well as the input values in the training process to adjust the weights of the links, unsupervised ANNs use only the input values.
there are networks that accept binary input while others accept continuous input. Examples of artificial neural networks include the Perceptron, the Hopfield network, the Hamming network, the probabilistic neural network, the radial basis-function network, the Carpenter/Grossberg classifier, the Kohonen network and the Neocognitron. \(^{(18, 19)}\)

The Perceptron is a popular example of ANNs. A simple Perceptron consists of an input layer and an output layer. It can classify an input vector into classes. Assume we have three classes of handwritten letters a, b and c, and each is contained in a matrix. Assume we have defined two features, feature 1 is the density of black pixels in the upper half, and feature 2 is the density of black pixels in the lower half. The simple Perceptron will be able to correctly classify these classes if they are linearly separated as shown in Fig. 1. Classification of unclassified symbols can be determined by measuring the minimum distance between the unclassified symbol and all classes. Then, the unclassified symbol will be considered to belong to the closest class.

On the other hand, non-linearly separable classes, shown in Fig. 2, cannot be solved by using a simple Perceptron because they are not linearly separated. However, such a problem can be solved by using a 3-layer or a multi-layer Perceptron.
The Perceptron structure can be used to implement a Gaussian maximum likelihood classifier. However, neither the Perceptron convergence procedure nor the Gaussian classifier is appropriate when the classes cannot be separated by hyper planes. No more than three layers are required to implement any nonlinear function because a three-layer network can approximate any non-linear function. The backpropagation training algorithm is an iterative gradient algorithm designed and used to minimize the mean square error between the actual output of a multi-layer feed-forward Perceptron and the desired output.

An example of a supervised 3-layer Perceptron is shown in Fig. 4. The backpropagation algorithm can be used to adjust the weights of the connections between the neurons. Assuming that digits are drawn in a 7x5 matrix of binary pixels (as shown in Fig. 3), there will be 35 input neurons (for the 35 pixels in the matrix) and 10 output neurons for digits 0 to 9. Assuming a single hidden layer, ANNs will look like the one shown in Fig. 4.
The backpropagation algorithm involves the following steps:

1. Apply the input vector $X$ to the first layer.

2. Calculate each input of the second layer

$$\text{Inp}_{12} = X_1 \times W_{11} + X_2 \times W_{21} + \ldots + X_n \times W_{n1},$$
\[ \text{Inp}_{22} = X_1 W_{12} + X_2 W_{22} + \ldots + X_n W_{n2}, \]
\[ \text{Inp}_{m2} = X_1 W_{1m} + X_2 W_{2m} + \ldots + X_n W_{nm}. \]

Where \( W_{ij} \) is the value of the connection strength from neuron \( i \) in the first layer (the input layer) to neuron \( j \) in the second layer (the hidden layer).

3. Calculate the outputs from the second layer

\[ \text{Out}_{12} = F(\text{Inp}_{12}), \]
\[ \text{Out}_{22} = F(\text{Inp}_{22}), \]
\[ \ldots, \]
\[ \text{Out}_{m2} = F(\text{Inp}_{m2}). \]

Where \( F(y) \) is equal to \( 1/(1-e^{-y}) \), the derivative is \( g(y) \)

4. Calculate the inputs to the third layer

\[ \text{Inp}_{13} = \text{Out}_1 M_{11} + \text{Out}_2 M_{21} + \ldots + \text{Out}_m M_{m1}, \]
\[ \text{Inp}_{23} = \text{Out}_1 M_{12} + \text{Out}_2 M_{22} + \ldots + \text{Out}_m M_{m2}, \]
\[ \ldots, \]
\[ \text{Inp}_{l3} = \text{Out}_1 M_{1l} + \text{Out}_2 M_{2l} + \ldots + \text{Out}_m M_{ml}. \]

Where \( M_{ij} \) is the value of the connection strength from neuron \( i \) in the hidden layer to the neuron \( j \) in the output layer.

5. Calculate the outputs from the third layer

\[ \text{Out}_{13} = F(\text{Inp}_{13}), \]
\[ \text{Out}_{23} = F(\text{Inp}_{23}), \]
\[ \ldots, \]
Out$_{13} = F$(Inp$_{13}$).

6. Calculate the errors in the third layer

\[ \delta_{13} = (Y_1 - Out_{13}) \cdot g(Inp_{13}), \]

\[ \delta_{23} = (Y_2 - Out_{23}) \cdot g(Inp_{23}), \]

\[ \ldots \]

\[ \delta_{33} = (Y_1 - Out_{13}) \cdot g(Inp_{33}). \]

7. Propagate the errors to the second layer

\[ \delta_{12} = g(Inp_{12}) \cdot (\delta_{13} \cdot M_{11} + \delta_{23} \cdot M_{12} + \ldots + \delta_{13} \cdot M_{1i}), \]

\[ \delta_{22} = g(Inp_{22}) \cdot (\delta_{13} \cdot M_{21} + \delta_{23} \cdot M_{22} + \ldots + \delta_{13} \cdot M_{2i}), \]

\[ \ldots \]

\[ \delta_{m2} = g(Inp_{m2}) \cdot (\delta_{13} \cdot M_{m1} + \delta_{23} \cdot M_{m2} + \ldots + \delta_{13} \cdot M_{2i}). \]

8. Update the weights $M$

\[ M_{11}^{\text{new}} = M_{11}^{\text{old}} + \eta \cdot \delta_{13} \cdot Out_{12}, \]

\[ M_{12}^{\text{new}} = M_{12}^{\text{old}} + \eta \cdot \delta_{23} \cdot Out_{12}, \]

\[ \ldots \]

\[ M_{21}^{\text{new}} = M_{21}^{\text{old}} + \eta \cdot \delta_{13} \cdot Out_{22}, \]

\[ \ldots \]

\[ M_{m2}^{\text{new}} = M_{m2}^{\text{old}} + \eta \cdot \delta_{13} \cdot Out_{m2}. \]

Where $\eta$ is a learning factor $\in [0,1]$. 

17
9. Update the weights $W$

$W_{11}^{new} = w_{11}^{old} + \eta \delta_{12} X_1$

$W_{12}^{new} = w_{12}^{old} + \eta \delta_{22} X_1,$

..., 

$W_{21}^{new} = w_{21}^{old} + \eta \delta_{12} X_2,$

..., 

$W_{nm}^{new} = w_{nm}^{old} + \eta \delta_{m2} X_n.$

10. Calculate the error $E = ((\delta_{13})^2 + (\delta_{13})^2 + (\delta_{23})^2 + ... + (\delta_{33})^2)$

Stop when $E$ is small enough (e.g. 0.01) else go to step 1.

After training an ANN with enough samples, the weights $M$ and $W$ will be stable. For an unclassified input $X$, (e.g. 7x5 pixels), the neuron with the maximum output value will be selected and hence, the corresponding digit.

2.3 The Main Pattern Recognition Techniques

Pattern recognition techniques, as mentioned earlier, fall under the following main categories: statistical pattern recognition and structural pattern recognition. In most statistical pattern recognition systems, symbol features are first defined, then the decision boundaries in the feature space are determined as shown by Duda and Hart.\(^{(13)}\) Many methods based on statistical pattern recognition techniques have been developed and proved to be very effective.

Statistical pattern recognition systems (including artificial neural networks) need a large number of training samples to select (or extract as in some ANNs) the features. If the features are not selected (or extracted) properly the feature space regions will overlap. As a result, some symbols that occur in the overlapped regions will be incorrectly classified.
Generally, statistical pattern recognition systems are developed for a specific application. For example, if an artificial neural networks is used for recognizing English letters and then it is required to recognize one or more new digits, this ANN would have to be retrained with the entire training data set. This tedious retraining must include both the new and the old symbols. In addition, the system may not function properly, especially, if one or more of the new symbol features are close to or overlap with other existing symbol features. In this case, the structure of the artificial neural network may need to be redesigned (e.g. by changing the number of neurons or the number of layers).

The second class of pattern recognition techniques is structural pattern recognition. In these techniques, complex patterns are divided into simpler ones, and the relations between the sub-patterns are described. Structural pattern recognition methods generally use rules and grammars to describe symbols as shown by Fu. These methods require few or no training samples. In addition, they can also justify their answers.

Pattern recognition problems that are also common arise when the characters overlap or when the writing is cursive. English characters can be written in isolated or connected forms. However, in the Arabic language, characters can only be written in cursive form. Hence, it is often necessary to segment a word into characters before recognition can be applied.

### 2.3.1 Statistical Pattern Recognition

Statistical pattern recognition techniques are based on extracting features from symbols, then clustering these features, hence, recognizing the symbols. Various features as well as methods are described by Trier et al. The methods include template matching, image transforms, zoning, and moments. Template matching is the simplest technique for symbol recognition. For some methods, the feature vector could be the symbol pixels themselves, or a mapping of them while other methods use short feature vectors. Such vectors result from transformations such as the Fourier transform and the Cosine transform. These methods use pixel by pixel manipulation which is not very suitable for the recognition of noisy or handwritten symbols, but is suitable for the recognition of typed symbols.
Methods which use the density of black pixels in different parts of the image (zones), directional histograms, and different types of moment-based invariant techniques have been proven to be useful in many statistical pattern recognition systems.\(^{21,22,23,24}\)

Statistical pattern recognition is useful in solving classification problems. N features lead to an N-dimensional space. There is no general method for defining the feature space. If the features are defined properly, then all feature vectors belonging to the same class will fall in the same cluster, therefore, enabling the cluster not to contain any feature vectors from any other class.

Classification is achieved by partitioning the feature space into regions, one (or more) region for each class. The decision boundary is chosen so as to minimize the probability of misclassification. As a result, selection of the appropriate feature vector is probably the most important factor in achieving a high rate of recognition.

The distribution of the training samples and the probability density function can be estimated, either by determining the parameters of a suitable distribution function or by estimating the density function itself. These parameters can be adjusted either by using supervised or unsupervised methods.

After studying many features, we found that selection of the appropriate feature vector is probably the most important factor in achieving a high rate of recognition.

Bayes decision theory is the ultimate statistical pattern recognition technique for minimizing the probability of error. This upper limit determines the best boundary regions for each class in the feature space. In order to increase the recognition rate above this limit, the features themselves have to be re-selected which requires major changes in the system.

The clustering procedure is the most important step in recognition. Clustering relies on a similarity measure typically based on a distance measure defined over the feature space. The dissimilarity measure is the mean-square distance between the input vector and each of the stored classified vectors. Clustering may be performed by minimizing a criterion
function, such as the sum of the squared error criterion. The distance between clusters can be measured by determining:

- The distance between the two closest points.
- The distance between the two furthest points.
- The average distance between points.
- The distance between the means of the two clusters.

With each technique there are advantages and disadvantages. The main disadvantages of using the statistical pattern recognition techniques are:

- They need a large number of training data.
- There is a probability of mis-recognizing data, this probability can be determined by using Bayes decision theory.
- They can not justify the answer.
- The output is only a classification and not a description of the pattern, and the relations between the features are ignored.

The main advantage, however, of using statistical pattern recognition techniques is that the decisions performed are based on a well-founded mathematical theory. Some statistical pattern recognition techniques have proved to be very useful for handwritten Arabic numeral recognition.\(^{(21)}\)

Artificial Neural Networks are the most successful statistical pattern recognition techniques with respect to the classification of symbols. Symbol features are extracted automatically from the training samples. In addition, classes can be created automatically by using supervised or unsupervised artificial neural networks.
However, ANNs have the following disadvantages: they can not justify the answer, they have to be trained (or even reconstructed) every time a new symbol is added, and there is no guarantee of a certain recognition rate.

A survey of some statistical pattern recognition techniques is conducted in the following section. These techniques used different features for the recognition of symbols. These features include:

- The percentage of black pixels in different zones (the zoning method).
- Various histogram based features such as the ratio between the vertical and the horizontal histogram distributions and the directional histogram.
- Moments.
- Features extracted using ANNs.

2.3.1.1 The Zoning Method

Multiple expert systems have been shown to be a promising strategy for handwritten pattern recognition by Cao et al.\(^\text{21}\) The image is divided into 4x4 rectangular zones. A histogram is calculated for each zone. The contour direction takes one of four possible values. An incremental clustering neural network algorithm with histogram feature extraction is used for recognition. Whenever a new training pattern is presented, a group of similarity distance measures between the current input feature vector and the existing weight vectors are calculated and examined. If the minimum distance is larger than a pre-assigned threshold, a new cluster is formed and its corresponding weight vector is set to the current input feature vector. Otherwise, the current input pattern is joined to the existing cluster closest to it. Following that, the current input vector and the weight vector are updated. A total of 13,200 handwritten digits were used. The system was trained using 6800 samples. The system was tested using 6400 samples. The lowest error rate achieved was 0.17%, and the rejection rate was 14.5%.
Twenty triangular regions are defined by Burel et al.\textsuperscript{(22)} Each of these twenty regions constitute a feature. A five-layer ANN Perceptron is used for clustering these and 76 other features. A recognition rate of 90\% was achieved.

### 2.3.1.2 The Histograms

Other important pixel-features are the histogram distributions. The histogram or the description of the histogram (by the mean, variance, or other moments) can be used as features for symbols. Statistical features based on the description of the profile by a histogram is suggested by Burel et al.\textsuperscript{(22)}

### 2.3.1.3 The Moments

Al-Yosef and Udpa\textsuperscript{(23)} implemented a histogram description method for Arabic character recognition where a vector of nine features is used. A recognition rate of 85.5\% was achieved for isolated printed character using a linear discriminate function.

First, the system identifies the connected regions in the image. Then, the vertical and horizontal histograms are calculated. The normalized moments of these histograms are used to describe the histogram distribution. Nine features are used to describe the symbol.

Features 1 and 2 measure the kurtosis, the flatness of distribution in the horizontal and vertical directions. Features 3 and 4 measure the skewness, the asymmetry of the distribution in the horizontal and vertical directions. Features 5 and 6 measure the normalized skewness and kurtosis, and relate the symmetry to flatness of the distribution in the horizontal and vertical directions. Features 7, 8 and 9 measure the relation between the moments of the vertical and horizontal projections of the same symbol.

Given that \( \mu_{r} = \sum (x_{i} - \mu)^{r} g(x) \) where \( g(x) = \sum x \) in the vertical or horizontal directions and \( x_{\mu} \) is the mean value of \( x \).

Features 1, 2 are equal to \( \mu_{r}/(\mu_{2})^{2} \) in the vertical and horizontal directions, respectively.

Features 3, 4 are equal to \( \mu_{3}/(\mu_{2})^{1.5} \) in the vertical and horizontal directions, respectively.
Features 5, 6 are equal to \( \mu_3/(\mu_4)^{25} \) in the vertical and horizontal directions, respectively.

Feature 7 is equal to \( \mu_1 \) vertical/\( \mu_1 \) horizontal.

Feature 8 is equal to \( \mu_2 \) vertical/\( \mu_2 \) horizontal.

Feature 9 is equal to \( \mu_4 \) vertical/\( \mu_4 \) horizontal.

During the training phase, the symbol features and the corresponding symbol are determined and stored in a database.

Other methods of calculating geometric moment invariant have proved to be useful in many pattern recognition systems. Cheng et al.\(^{(24)}\) introduced a hierarchical neural network for electrical symbol recognition. Six invariant moment features are calculated, and a multi-layer Perceptron classifier with a back propagation learning model is used for recognition. The recognition rate for 340 training input symbols was 98.5\%, and the rate for 170 test symbols was 89\%.

It is clear that the number of training and testing symbols is low. It is believed that the recognition rate will be much lower than 89\% if a larger database is used.

### 2.3.1.4 Features Extracted By Using ANNs

The Neocognitron is a neural network developed to recognize handwritten Latin characters by Fukushima et al.\(^{(25)}\) It is an attempt to imitate the human visual system. A selective attention model has been extended by Fukushima\(^{(26)}\) to perform both recognition and segmentation of connected Latin characters.

The Neocognitron consists of a sequence of simple layers followed by a complex layer. The system is designed to recognize the digits 0 through 9, regardless of where they are placed in the retina’s field of view. On the simple layers (S), the cells on each plane are sensitive to simple features. Each layer of the simple cells acts as a feature extraction system that uses the layer preceding it as its input layer. Each S-cell on a single plane is sensitive to the same feature. The Complex cells C integrate the responses of groups of S-cells. The C-cells’ response is less sensitive to the exact location of the feature on the
input layer. This gives the Neocognitron the ability to identify characters regardless of their exact position or size in the field of view of the retina. The weights of each template window which relates complex layers to simple layers are calculated by using supervised techniques and updated by performing certain training samples for each cell. These training samples teach the network how to extract the features.

The process of pattern recognition in the multi-layer network can be summarized as follows. In the first module, several features are extracted. In the next module, these features are combined by observation over a larger range, and higher order features are extracted.

The recognition of connected characters can not be successfully performed by using the simple pattern matching technique. This is because the same character can be written in different ways. The model used for recognizing connected character has top down as well as bottom-up connections between the layers. The signals through the forward paths manage the function of pattern recognition while the signals through the backward paths manage the function of selective attention, segmentation and associative recall. The routes of the backward signals are controlled by gate signals. Only one pattern is recognized at a time by this feedback path. Once a pattern has been recognized and segmented, the process of recognizing the second pattern begins.

2.3.1.5 Features Studied by the Author

The author conducted the following experiments to study the effectiveness of some statistical features for handwritten character recognition. The initial results show that statistical pattern recognition is not useful for handwritten character recognition. A small set of data (40 samples/class), and 25 classes are used. It is found that the classes overlap in the feature space and each sample is far from the center of its class. Thus, it was concluded that statistical pattern recognition techniques, as have been shown in the previous sections, are likely successful for typed symbol recognition but are not as useful for handwritten symbols. Statistical pattern recognition is only applicable if the features are well clustered in the feature space. In addition, there will always be some mis-
recognized symbols. The features that we used in our studies should be classified as follows:

1. The percentage of black pixels in different zones (the zoning method).

2. Various histograms based features such as the ratio between the vertical and the horizontal histogram distributions and the directional histogram.

3. Moments.

4. Other miscellaneous features, such as the aspect ratio.

We now describe the details of the features we considered and the results we obtained.

a) The zoning method

The percentage of the black pixels in each zone is used as a feature for this zone. During the training phase, symbol features are extracted, the corresponding symbol is determined and then both symbol and features are stored in the database. The symbol is divided into 16 (4x4) regions, as shown in Fig. 5. The corresponding feature vector is:

\[
\begin{bmatrix}
0.79, 0.46, 0.51, 0.42, \\
0.77, 0.31, 0.40, 0.47, \\
0.72, 0.14, 0.20, 0.67, \\
0.80, 0.46, 0.47, 0.53
\end{bmatrix}
\]

Figure 5 The symbol is divided into 16 regions.
The normalized pixel density is determined in each zone. The feature vectors of 40 symbols are summed together, slot by slot (each vector has 16 slots). The mean vector for the forty symbols is determined for each of the 25 classes. The same data is used to determine how many symbols lie in the overlap regions. It was found that the percentage of the symbols lying in the overlap regions is 13.7\% i.e. the substitution rate is 13.7\%.

b) Ratios of histograms

The experiment uses the histogram distributions as features. The symbol is divided into 256 (16x16) zones as shown in Fig. 6. Horizontal and vertical histogram distributions are calculated. Finally, the ratio between the two distributions is determined as follows:

The value in slot \([i]\) is equal to the ratio of the difference between the horizontal and vertical projections \([i]\) to their summation.

The resulting ratio constitutes a feature vector of the original symbol. This vector comprises of 16 slots. The mean vector for the forty symbols is determined for each of the 25 classes. The same data is used to determine how many symbols lie in the overlap regions. The percentage of the symbols in the overlap regions is 29.4\% i.e. the substitution rate is 29.4\%.

For example, the features vector for the symbol shown in Fig. 6 is as follows:

\([-0.15, -0.1, -0.4, -0.01, -0.01, -0.01, 0.16, 0.39, 0.35, 0.02, -0.05, -0.15, -0.31, -0.32, 0.19, 0.55]\)
Another feature extraction method studied in this research is the directional histogram. This method can be used for pattern recognition, especially in signature recognition. The method is described by the following algorithm:

1. Thin the image as shown in Fig. 7.

2. Calculate the image area (the total number of black pixels in the thinned symbol).

3. Calculate an approximation of the direction of each pixel by fitting the best line passing through the pixel (using a 5x5 window and the least square method for minimizing the error). Define the slope of this line as the direction of the pixel.
4. Calculate the number of pixels with different slopes. The number of slopes that are used in this experiment is 16 (i.e. a slope step = 180/16 degree).

5. Normalize each value by dividing it by the area (the total number of black pixels).

![Figure 7 A symbol before thinning (a) and after thinning (b).](image)

A vector of length 16 is used to represent the symbol. Each one of the vector’s 16 slots contains the percentage of pixels that have a certain slope. In this experiment, the slopes are [0.00, 11.25, 22.50, 33.75, 45.00, 56.25, 67.50, 78.75, 90.00, 101.25, 112.50, 123.75, 135.00, 146.25, 157.5, 168.75].

For example, the symbol shown in Fig 7(a) is described by the following vector:

\[0.37, 0.020, 0.010, 0.020, 0.060, 0.020, 0.020, 0.000, 0.320, 0.020, 0.030, 0.050, 0.0200, 0.020, 0.000\]

The mean vector for the forty symbols is determined for each of the 25 classes. The same data is used to determine how many symbols lie in the overlap regions. 62.5% of symbols lie in the overlap regions, i.e. the substitution rate is 62.5%.

d) Normalized moments

The first six features introduced by Al-Yosef and Udpa\(^{(23)}\) are used in the following experiment. An example of the feature vector for the symbol shown in Fig. 5 is as follows: \[0.35, 0.35, 0.034, 0.14, 0.07, 0.31\]
The mean vector for the forty symbols is determined for each of the 25 classes. The same data is used to determine how many symbols lie in the overlap regions. 42.9% of the symbols lie in overlap regions, i.e. the substitution rate is 42.9%.

e) Moments invariant

1000 symbols are used to test the first four moments, which are used by Cheng et al.\textsuperscript{(24)} The mean vector for the 40 symbols is calculated for each of the 25 classes. The same test symbols are used to determine how many symbols lie in overlap regions. It was found that 75.2% of the symbols lie in the overlap regions, i.e. the substitution rate is 75.2%.

f) Miscellaneous features

A combination of different features is used for pattern recognition. The following six features are defined:

1. Density: the ratio of the number of black pixels to the total number of pixels in the minimum rectangle that contains the symbol.

2. Perimeter: the circumference of the minimum rectangle that contains the symbol.

3. Centre of mass: if the origin of the entire image is considered to be the pixel at location (0,0), then the center of mass of an object C is \((Cr, Cc)\)

   \[
   Cr = \frac{1}{n} \sum x, \quad Cc = \frac{1}{n} \sum y \text{ for } n \times n \text{ image.}
   \]

4. Form Factor: the form factor is the ratio between the square of the perimeter to the symbol area.

5. Directed Density: it is similar to the density calculated in the zoning method. However, in this case the rectangle is oriented toward the principal axis.

The principal axis of a bi-level symbol is a line passing through the central point having a minimum total distance from all pixels belonging to the symbol. This is achieved as follows:
1. Locate the central point.

2. Locate the boundary pixels. A point in the boundary and the C point forms a line, one of these line is the principal axis.

3. Calculate the sum of the distance between all points and the assumed principal axis.

4. Select the point that has a minimum sum. Then, find the oriented rectangle:

5. Locate the principal axis and construct an oriented rectangle that encloses the object.

6. Identify the two points that have the greatest distance from the principal axis (the minor axis).

7. Identify the two points that have the greatest distance from the minor axis.

Although each feature by itself is suitable for describing the symbols, the combination of these features could cause a serious overlap among the regions of the classes. An example of a vector of these 6 different features is defined for the symbol shown in Fig. 5 is \[0.52, 0.04, 0.66, 0.34, 0.20, 0.5\].

The mean vector for the forty symbols is determined for each of the 25 classes. The same data is used to determine how many symbols lie in the overlap regions. The percentage of the symbols in the overlap regions is 74.1%, i.e. the substitution rate is 74.1%.

2.3.1.5 Features Summary

The following table summarizes the results of using different statistical features for handwritten character recognition. Some characters will be recognized by applying a certain method. Others will be recognized by applying a different method. It was concluded that it is important to select the appropriate features to achieve a high recognition rate using statistical pattern recognition techniques.
The goal of this thesis work is the construction of a system for recognizing general symbols. The system should be able to justify the answer and have a recognition rate that can always be increased to any value (as close as possible to 100% with no misrecognized symbols, but with possible some rejected symbols). In the following section we will survey different methods of structural pattern recognition.

### 2.3.2 Structural Pattern Recognition

Structural pattern recognition is quite different from statistical pattern recognition. The basic idea behind structural pattern recognition is that objects are constructed from smaller components using a set of rules.

Syntactic pattern recognition techniques are a sub-set of structural pattern recognition. In syntactic pattern recognition, complex patterns are divided into simpler ones, and the
relations between the sub-patterns are described. The techniques are based on mathematical and formal language theories. However, these techniques are not powerful enough to handle real-world applications.

An example of a syntactic pattern recognition method is the two-dimensional mathematical notation recognition developed by Anderson. The recognition algorithm starts with the ultimate syntactic goal which is partitioned into a set of sub-goals, these in turn are broken up into other sub-goals, and so on until either every sub-goal is reached or all possibilities have failed. A set of rules is used to describe the mathematical expressions. Different grammars can be used to describe a large set of complex patterns and to recognize them. Examples of grammars are shown in Figs 8 and 9.

Figure 8 An example of a transformational grammar.

Figure 9 An example of context sensitive grammar.
Four expert systems were developed for recognizing unconstrained handwritten digits by Suen et al.\textsuperscript{(28)} Different primitives are used in each system. The first expert system obtains the skeleton by thinning, then determines the end points and the junction points. To simplify the skeletons, branches less than 10\% of the total pixels are removed. Freeman codes are used to code the skeletons. Then, the skeleton is simplified by an average direction of one code for few codes. The bounding box for a symbol is divided into nine regions: center, north-west, north, north-east, east, south-east, south, south-west, and west. The classification is performed by using eleven modules. The first module is dedicated to the recognition of one-branch symbol parts, the other ten modules are used for the recognition of multi-branch symbol parts. The recognition rate is 86.05\%, and the substitution rate is 2.25\%.

The second expert system performs the following steps:

- Thinning.
- Tracing the skeleton.
- Approximating the curves by lines.
- The arrangement of the line segments into primitives.

The recognition rate is 93.1\% and the substitution rate is 2.95\%.

The third expert system uses the following features:

- the locations of the end points,
- the junction points in each region of the digit,
- a bend between two lines,
- horizontal curves at top or bottom,
- distance between two points,
• location of special points in the digit,

• direction of a stroke from a special point,

• inflection between two points,

• a curve at the left or right profile,

• comparative lengths between two strokes,

• straight stroke between two points,

• width of the digit at different rows,

• width of a stroke,

• symmetry of character,

• horizontal intersection at a row.

The system recognition rate is 92.95%, and the substitution rate is 2.15%.

The fourth expert system relies on extracted features from the contours. The recognition rate is 93.9% and the substitution rate is 1.6%.

Another structural pattern recognition method for digit recognition is described by H. Jianming and Y. Hong. In this method, four points for bends (B), points for curves (D), terminal points (T), and intersections (I) are defined. A primitive is defined as the skeleton segment which starts and ends at feature points. A feature code of 11 elements is used to describe the local information of the character and a five-element vector is used to describe the global information.

The eleven elements are:

1. the curvature of the primitive.

2. the orientation of the primitive,
3. to 9. the types and locations of the start and end points

10. the moving direction of the primitive,

11. the length of the primitive.

The five elements are:

1. the number of loops,

2. the number of paths from T to T points,

3. the number of paths from T to I points,

4. the number of paths from I to I points,

5. the number of T points.

The recognition rate for the digits using prototype matching is 97.86%. While the recognition rate for neural networks is 97.29%, the substitution rate is 2.71%.

A new technique integrating a neural network and a knowledge-based system for image recognition was introduced by Kim et al.\(^{(30)}\) The system was designed to recognize handwritten digits. The model is capable of inductive learning from example data and logical inference from the rule base. The system has the ability to justify the answer, e.g., why a pattern is recognized as 5, and not 6 or 9. This system is well suited for applications where only partial knowledge is available. For a typical sample of 200 handwritten digits, the recognition rate ranges from 69.5% to 81.5% and depends on the number of rules.

Another system that integrates the use of neural networks with expert systems is reported by Amine et al.\(^{(31)}\) The original image is transferred into a binary image using a parallel thinning algorithm. The resulting skeleton is traced. Some primitives such as straight lines, curves and loops are extracted. Finally, a five-layer artificial neural network is used for classification. The neural network is trained with 2000 isolated Arabic characters and tested using 1000 new characters. The recognition rate is 92%.
A system by Burel et al. uses a combination of statistical and morphological features and is successful in recognizing handwritten digits. The system divides a digit into twenty regions. Each region has a feature associated with it. This feature is the ratio of the number of black pixels in this region to the number of black pixels in some other region. The morphological features include cavities (west, east, north, south and center) and holes. A five-layer Perceptron neural network is used for classification. The artificial neural network is trained with 1414 digits and the evaluation is performed on 1175 digits. The recognition rate is 93.6%.

A structural feature extraction technique for English character recognition is described by Starzyk et al. An effective method for thinning using a 3x3 template windows is used. Although this method uses a 3x3 window, it also uses a non-boundary pixel that looks at pixels just outside this window. These windows are applied sequentially and are implemented by a hardware circuit. After thinning the image, critical points are marked, then the segments are determined. These segments are scaled, matched, and classified.

2.4 Preprocessing

Preprocessing of off-line handwriting patterns is done prior to recognition. This usually involves cleaning and smoothing strokes.

2.4.1 Thresholding

Thresholding is used in the process of converting a grey-level image into a bi-level one. It involves looking at each pixel and deciding whether it should be white or black. In simple cases, the mean or median value of the gray levels over the entire image may be used as the threshold. This technique does not always produce good results. Using multiple thresholds and dividing the image into regions gives better results.

2.4.2 Noise

Noise manifests itself as random variations in the pixel levels caused by digitization or other processes applied to the image. There are different types of noise. One type of noise causes a black pixel to become white or a white pixel to become black, at random. For
this type smoothing filters are effective. A better tool is the median filter, which acts to minimize the edge-blurring effect.

2.4.3 Overlapping

The steps in separating overlapping objects using the Watershed method are as follows: First, compute the distance transform, which is the minimum distance from each pixel to the background. The distances 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16, are marked by the characters 0,1,2,3,4,5,6,7,8,9,:,;,<,= >,?,@ in Fig. 10.

Locate the pixels with peak values ‘>’ where K=14 and ‘@’ where K=16. These pixels are assigned a value of K. Pixels adjacent to the peaks are given a value of K-1.

The regions will be identified by starting at the peaks and moving to the background pixels. Adjacent regions are not allowed to overlap or touch. This method can be effective for separating objects such as blood cells.
If the characters are touching such as \( \text{BB} \), then using the histogram distribution in different directions may also help to separate them. The separating points will be the points that have low histogram values. Other methods are also suggested by Y. Lu and M. Shridhar.\(^{34}\) The recognition problem becomes harder if the characters are overlapped such as \( \text{HE} \).
2.4.4 Rotation and Slant Correction

Slant is the angle in degrees measured in a clockwise direction from the vertical at which the characters were drawn. Many methods have been developed for the correction of slanted symbols \( B \). Other methods have been developed for the correction of tilted (or rotated, skewed) symbols, \( \mathbf{w} \) in a document. A fast algorithm for tilt correction in printed document images is presented by C. Sun and D. Si.\(^{(35)}\) The tilt angle is obtained by searching for the histogram with a sharp peak in a set of histograms projected in different directions. The document is corrected by a geometric rotation operation. The algorithm is described below:

- Find the gradient at each dark pixel by using a Sobel filter.\(^{(36)}\)
- Calculate the oriented histograms.
- Search for the maximum in this histogram and hence calculate the tilted angle.

Slant correction is the process that tries to adjust the slant of the characters in a line or paragraph. After the slant angle is detected, a shear operation is applied to correct the slant characters. Slant can be determined to each line in the text by first finding the bounding box which is not a rectangle. Slant can be corrected by applying a shear operation to make the bounding box a rectangle. Slant can also be corrected by applying a shear operation using the chain code method suggested by Shridhar and Kimura.\(^{(2)}\)

Some methods for slant correction include:

- Using projection profile, where the projection that gives the maximum variation corresponds to the projection with the best alignment.
- Using Hough transform where each point \((x,y)\) is mapped to \((\rho,\theta)\), the dominant lines are found from the peaks in the Hough space.
• Using the Fourier transform.
• Using nearest neighbor clustering.
• Using correlation.

2.4.5 Segmentation

There are two general methods for the recognition of words. These two methods are recognition with segmentation and recognition without segmentation. Segmentation is the process of breaking down the word into sequences of smaller size units. An example of this segmentation process would be breaking a sentence into words, a word into characters or characters into strokes.

Segmentation can be done by determining the points that have a low value of histogram vertical projections. It can also be done by tracing the outer contour and calculating the distance between the extreme points of intersection of the contour with a vertical line.

Segmentation is the process in which observed patterns are converted into units of patterns that seem to form characters. Segmentation is an active research area. A survey of different segmentation methods is conducted in (37, 38). For characters written in predefined boxes, the writer does most of the segmentation.

A word recognition technique without segmentation is introduced by B. Al-Badr and R. Haralick. (39) The system is used for detecting typed Arabic words. The system detects the word primitives, then matches them with the symbol models. The system uses thirteen primitives. The recognition rate is 99.4% for noise free text, and 73% for scanned text.

A lexicon-free handwritten word recognition is described by G. Houle and M. Shridhar. (40) The system creates a hypothesis about the segmentation of the word into characters, then the system selects the best hypothesis. Results on 12,968 numeric fields extracted from the Bureau of Census 95 database yielded a correct field rate of 92.7% with a 1.4% rejection rate. However, using 6785 character fields, the correct field performance was 51.7% with a 20.1% rejection rate.
2.5 Postal code and Address Recognition

A very important application of optical character recognition is found in postal code and address recognition. Significant research has been carried out. \(^{(2, 41, 42)}\) Shridhar and Kimura\(^{(2)}\) introduced an address interpretation system developed for determining the street number, the US ZIP code, street name and PO Box number in mail pieces. The system carries out the following stages of processing:

- Slant correction.
- Line segmentation.
- Recognition of ZIP code field, street number field, and PO Box field.

During recognition, this system uses a database to order the addresses, states, streets, and ZIP codes.

There are two approaches to correct the slant problem. The first one is the estimation and correction of the slant prior to line segmentation. The second one is applied after line segmentation for each line. The basic strategy of slant estimation is to find the direction in which the projection of the document has maximum separability regarding the peaks and valleys. Once the slant of the image is estimated, slant correction is implemented as a shear transformation. Slant can be also corrected by using the Hough transform method.

Many word recognition algorithms attempt to segment words into characters and sub-characters. There are two fundamental approaches to word recognition. The first approach uses a lexicon, which is used as a post processor to select the best match. The second approach matches pre-segmented word image against all the words of the lexicon to obtain the best match.

If the lines that make up words are separated and not tilted, the horizontal projection of the word will have clearly separated peaks and valleys. These often determine the location of boundaries between lines.
3540 images having ZIP codes and street numbers were used in the test. The top correct ZIP code recognition rate was 82.20%. The top correct recognition rate for the street number was 77.29%.

2.6 Summary

The objective of this thesis is to develop a method that can recognize any handwritten or typed symbol. To determine a suitable approach we reviewed the representation of different pattern recognition techniques discussed in the literature. These techniques mainly fall under two major categories, the statistical pattern recognition techniques (including ANN's) and the structural pattern recognition techniques.

Several statistical pattern recognition techniques were discussed. These techniques rely on different features extracted from the patterns to be recognized and which we call pixel features. These features are based on pixel by pixel manipulation. It is shown that usually there will be symbols in the overlapped regions in the feature space. There is no way to avoid this problem in statistical pattern recognition techniques. Artificial neural networks are the best statistical pattern recognition techniques because they achieve a high recognition rate. These high rates are achieved when ANNs finds the features implicitly. ANNs have been used for handwritten character recognition. Since the ANNs can not explicitly describe the features or the mechanism they used to arrive to their answers they can not justify the answer. They can not easily be modified to recognize newly added symbols. In addition, they require a large number of training samples.

Bayes decision theory is a statistical technique which minimizes the probability of error in classification. It determines the best boundary regions for each class in the feature space. In order to increase the recognition rate beyond that offered by this optimization process, the features themselves have to be re-selected.

Using some features and a small set of data (i.e. 40 samples and 25 classes), we have found that the classes overlap in the feature space. Statistical pattern recognition is only applicable if the features are well clustered in the feature space, but there will always be misclassified symbols.
The main disadvantages of using the statistical pattern recognition techniques are: they need a large number of training data and there will always be misclassified data limited by Bayes decision theory. In addition, these techniques can not justify the answer, and the output is only a classification and not a description of the pattern. Also, the relations between the features are ignored.

The main advantage of using the statistical techniques is that its decision is based on a well-founded mathematical theory and the recognition rate is very high.

The author believes that the structural pattern recognition techniques are the most suitable methods for handwritten symbol recognition since they are somehow similar in their operation to humans when recognizing patterns. The structural pattern recognition system which we developed is described in detail in chapter 4. Since this system relies on thinning the symbols, we discuss different thinning algorithms in chapter 3.
Chapter 3

Thinning

In this chapter, a survey of different thinning methods will be discussed. Then, a novel parallel thinning rule-based system will be described. This system thins symbols to their central lines. Finally, we will describe a novel fast one-pass sequential thinning rule-based system for detecting certain edges of the symbol. These edges are sometimes sufficient to represent the thinned symbol. This one-pass system can also be used to delete any extraneous pixels resulting from the first thinning system.

3.1 Introduction

Thinning is the process of reducing the thickness of each line of patterns to just a single pixel. Thinning algorithms can be classified into two general types, parallel and sequential. Parallel algorithms peel off the boundaries until the objects have been reduced to thin lines. The process is repeated until no further changes occur in the pattern. The number of iterations is approximately equal to half of the maximum line width of the objects. Pixels are examined for deletion based on the results of the previous iteration. However, in sequential thinning algorithms, the deletion of a current pixel depends on the results of the previously processed pixels.

Thinning is usually used as the first step in applications such as optical character recognition (OCR). A thinning algorithm is considered to be useful if it preserves the shape of the original image, preserves the connectivity, and does not leave extraneous pixels (branches).

3.1 Thinning Algorithms Survey

A comprehensive survey of thinning algorithms is described by Lam et al.\textsuperscript{(43)} Templates of 3X3 windows are usually used to perform the thinning and trimming of the extraneous pixels. The parallel processing of (3X3) windows simultaneously may cause the loss of connectivity. Kim et al.\textsuperscript{(44)} introduced 3X4 and 4X3 windows and added extra conditions
to perform parallel thinning which preserves connectivity. The above methods, however, do not always result in thinned patterns represented by their central lines.

A fast parallel algorithm for thinning digital patterns by deciding as to whether or not a pixel can be eroded is described by Zhang and Suen. This ZS algorithm is effective, easy to use, and has been so far the most popular method. Assuming \( P_i \) is the bi-level value of the pixel written in a 3x3 window as:

\[
\begin{array}{ccc}
| P_9 | P_2 | P_3 \\
| P_8 | P_1 | P_4 \\
| P_7 | P_6 | P_5 \\
\end{array}
\]

the ZS algorithm uses 2 passes (sub-iterations) to obtain a central skeleton. In the first pass, the ZS algorithm deletes the pixel \( P_i \) if it satisfies the following conditions:

1. \( 7 > B(P_i) \geq 2 \),
2. \( A(P_i) = 1 \),
3. \( P_2 * P_4 * P_6 = 0 \),
4. \( P_4 * P_6 * P_8 = 0 \).

Where \( A(P_i) \) is the number of 01 patterns in the ordered \( P_2, P_3, \ldots, P_8, P_9 \) sequence and \( B(P_i) = P_2 + P_3 + \ldots + P_9 \).

In the second sub-iteration, the first two conditions remain the same as above, the third and fourth conditions are changed.

1. \( 7 > B(P_i) \geq 2 \),
2. \( A(P_i) = 1 \),
3. \( P_2 * P_4 * P_6 = 0 \),
4. \( P_2 * P_6 * P_8 = 0 \).
The iterations are repeated until no change occurs. The ZS algorithm results in central lines of the image for the vast majority of cases. For the case of 45° and 135° diagonal lines that are 2 pixels wide only, the ZS algorithm reduces these lines irrespective of their lengths to a dot formed of 1 or 2 pixels. A solution for this case is suggested by Raju and Xu.\(^{(46)}\)

A 2-pass parallel thinning algorithm with template matching is described by Zhang and Wang.\(^{(47)}\) This algorithm preserves image connectivity, produces thinner results (less number of pixels) than the above ZS algorithm, maintains fast speed, and also generates one-pixel wide skeletons.

A fast parallel thinning algorithm HSCP is described by Holt et al.\(^{(48)}\) The first two conditions of this method determine the pixels that can be deleted and are the same two conditions as (a) and (b) above for the ZS algorithm. All deletable pixels will be deleted except for those that satisfy any of the following conditions (a) \(P_2=P_6=1\) and \(P_4\) is deletable, (b) \(P_4=P_8=1\) and \(P_6\) is deletable or (c) \(P_4, P_5\) and \(P_6\) are all deletable.

The HSCP algorithm is compared to the ZS algorithm by R. Hall.\(^{(49)}\) Hall suggests changing condition (a) to \(7 > B(P_i) > 2\) to solve the diagonal line problems in the ZS and HSCP algorithms.

Another parallel thinning algorithm which is based on weight values was described recently by Han et al.\(^{(50)}\) The weight values are the sum of the black pixels around the tested pixel in a 3x3 window. The pixel is deleted if it satisfies certain conditions.

An iterated parallel thinning algorithm that uses the 3x3 templates are implemented by C. Arcelli et al.\(^{(51)}\) The pattern is thinned down in an isotropic way using four passes per iteration. The algorithm thins the pattern to its central lines. The thinned templates are as follows:
where blank positions are “do not care” pixels, x and y are Boolean variables such that: 

\((\neg(x) \lor y) = 1\). Refer to L. Lam and C. Y. Suen\(^{52}\) for the study on the performance and evaluation of 10 different thinning methods.

An artificial neural network based on adaptive resonance theory (ART) has been used successfully for thinning Arabic characters by Altuwaijri and Bayoumi.\(^{53}\) This thinning algorithm clusters the data image, then the skeleton is generated by plotting the cluster centers. Finally, adjacent clusters will be connected by straight lines.

Our goal here is to implement a thinning system that is fast, rotation invariant, and can represent patterns by their central lines. The thinned pattern should be one pixel wide and should not have any discontinuity. Ideally, the method should be invariant to rotation of the original pattern i.e. if the input is rotated by an angle \(\theta\) then the output is also rotated by the same angle. We relax the last condition to rotations of the original patterns by 90°, 180°, and 270°.

We have developed two knowledge-based thinning systems. The objective of the first system is to achieve a thinning system that can approximate patterns by their central lines, see Ahmed and Ward.\(^{16}\) The thinned pattern should be one pixel wide and should not have any discontinuity. Several iterations may be used. Ideally, the method should be invariant to rotation of the original pattern.

The second system uses the sequential thinning method. It is a fast knowledge-based system that preserves the symbol shape to some extent and also guarantees the connectivity, see Ahmed and Ward.\(^{17}\) This system has the advantage in that its rules are applied in one sequential pass only, i.e. only one iteration is needed to obtain the thinned
image. This system can also be used to eliminate any unnecessary pixels in an image after thinning.

3.2 Good Thinning Characteristics

Thinning reduces the binary image into one in which most dark pixels have one, two or three dark neighbors. A good thinning system should have the following characteristics:

- It preserves and ensures the continuity of the symbol.
- It preserves the shape of the symbol.
- It preserves the end points.
- It has the least possible number of pixels, however achieves the previous three characteristics.

3.3 The Central Line Thinning Rule-Based System

As mentioned in section 2.2.1 a knowledge-based system is a computer program that performs a specific task normally done by a human expert. Knowledge based systems have the ability to manipulate problem statements and integrate relevant pieces of knowledge from the knowledge base using heuristic techniques. These systems are problem-solving computer programs that use their knowledge in flexible ways, and can reason with inexact knowledge which may be declarative or procedural. They have an inference engine for reasoning with rules and facts.

Prolog built-in features for backtracking, strong logic handling and rule manipulation have proved to be very helpful in implementing these systems. Prolog has the advantage of being a descriptive language; it solves the problem by describing it rather than showing how to solve it. The Prolog language is the most appropriate language for implementing knowledge-based systems. The C++ language has the advantages of being fast and capable of handling large documents.
The system should be able to draw conclusions about how the thinned output pattern was reached, and which rules were applied. The system should learn new rules if desired. Knowledge is distributed over the individual rules, which are independent of each other. An important goal is to establish an inference mechanism among these rules.

A rule-based system for thinning that uses one parallel pass and 16X4 rules is introduced. The 16 rules are applied in the four directions 0°, 90°, 180°, and 270°. This results in the system being rotation invariant in these directions. The resultant thinned image is composed of the central lines of the image. The proposed method is an improvement of the ZS algorithm. The ZS algorithm requires twice the clock speed of our one-pass method, and the resultant thinned pattern may vary when the original pattern is rotated indicating that this algorithm does not always result in the central lines of the original pattern. Our system has 16 rules listed in Fig. 11.

![Figure 11 The thinning rules with θ = 0°.](image)

These rules aim at deleting the pixels which are north-west corner pixels or pixels that lie in the east boundaries or south boundaries. The conditions for 2-pixels thickness lines are given special attention. The conditions are listed in Fig. 12. The conditions are first
checked by the system before any rule is applied. If one of the eight conditions is satisfied, then the original 16 rules will be applied without their rotations i.e. Only the 16 rules are applied and not the 16x4 rules. The pixel that will be deleted is marked by a darker square. Where 0 refers to a white pixel, 1 refers to a black pixel, and x is either a black or white pixel (i.e. don’t care). This case will be referred to later in this section as shown in Figure 20.

![Figure 12 The special case conditions relevant to 2-pixels thickness lines.](image)

A document that has English, Arabic, and Chinese characters as well as some geometric shapes is shown in Fig. 13. The thinned patterns resulting from applying the 16 rules are shown in Fig. 14. The rules are repeatedly applied to the patterns at every iteration until no further changes occur.

![Figure 13 The original symbols.](image)
The 16 rules listed in Fig. 11 are rotated by 90°, 180°, and 270° to produce 3 more sets of rules. The thinned patterns that resulted from applying each set of these three 16 rules are shown in Figs. 15, 16, and 17 respectively.
Figure 16 Thinning by applying the 16 rules with $\theta = 180^\circ$.

Figure 17 Thinning by applying the 16 rules with $\theta = 270^\circ$.

Although the results are good for the majority of the patterns, these figures show that as the 16 rules are rotated (which is equivalent to rotating the image and applying the original 16 rules), the resulting thinned patterns are not always the rotated version of the original patterns. Also, the thinned patterns that resulted from applying any of these sets of 16 rules set are not always the central lines of the original patterns.

Another method may use four passes in each iteration. In each pass the same 16 rules are applied but in the second, third, and fourth passes the rules are rotated by $90^\circ$, $180^\circ$, and
270° respectively. The thinned patterns that have resulted from this procedure, i.e. from applying four passes sequentially in the above order, in each iteration, are shown in Fig. 18.

![Figure 18 Thinning by applying 4 passes (16 rules in each pass) with \( \theta = 90°, 180°, \) and 270° in second, third, and fourth pass, the 4 passes are repeated in each iteration.]

However, for this method, the order in which the rotated sets are applied affects the resulting patterns. Also, the resulting thinned patterns are not always the central lines. To remedy the above problems we propose the following algorithm: We use only one pass where the 64 rules (composed of the 16 rules and their rotated versions) are applied on every pixel in this one-pass. It is easy to show that if this method is used and the original image is rotated by 90°, 180°, or 270° the resultant thinned output will also be rotated by the same angle, i.e. this method is rotation invariant for these directions. Furthermore, the thinned patterns are always represented by their central lines but as Fig. 19 shows, when a pattern in any iteration is 2-pixels wide, there may be a discontinuity. The rules in Fig. 12 are used to solve the discontinuity problem.
The central line of 2-pixels thickness line lies between these two pixels. Therefore, the central line can approximately pass through any of these 2 pixels. When a pattern is formed which is 2-pixels wide in any direction, we modify our procedure and we only apply the original 16 rules and not their rotated versions. Thus, the modified system first checks if the pattern at the pixel of concern is two pixels wide. If this is true the 16 rules are applied otherwise the 64 rules are used. The thinned patterns that have resulted from this modification are shown in Fig. 20. The conditions a pixel should satisfy to be considered to belong to a 2-pixels thick line are listed in Fig. 12. Fig. 20 shows that our method results in the central lines, which are connected (no discontinuity) and are rotation invariant.
The following example shows the different iterations to thin a symbol. It took four iterations by using our central thinning algorithm to thin the symbol ‘A’.

The system peels off the boundaries until the objects are reduced to thin lines of one pixel thickness.

Another example is shown in Fig. 21. The original pattern is shown in Fig. 21 (a). The pattern after applying the 16 rules without rotations is shown in Fig. 21(b) where the deleted pixels are shown in black. Notice that the east and south boundaries are deleted, the north west corners are also deleted.

When the 4x16 rules are applied, all the boundary pixels are deleted as shown in Fig. 21(c). The next iteration is shown in Fig. 21(d). After few iterations the pattern is thinned to the central lines as shown in Fig. 21(e).
We can also show that the ZS algorithm outlined previously can be derived using the same rules shown in Fig. 11 that we have developed for our thinning algorithm. Rule 3 should be changed to rule 3 and rule 14 should be changed to rule 14.

The main difference between the ZS algorithm and our algorithm is that the ZS algorithm requires two passes in each iteration and our algorithm requires only one pass in each iteration. The first pass uses the 16 rules (with rules 3 and 44 instead of rules 3 and 14) with $\theta = 0^\circ$ and the second pass uses the same 16 rules with $\theta = 180^\circ$.

After changing rules 3 and 14 as described above an exact equivalence of the ZS algorithm is obtained. The ZS algorithm has problems of reducing 2-pixels wide $45^\circ$ and $135^\circ$ diagonal lines to just 1 or 2 pixels see Fig. 23(b). To address these problems we can
show that if our rules 3 and 14 are used instead of rules 3-and 44, then these problems are solved. The thinned patterns that resulted from using the ZS algorithm are shown in Fig. 22. The patterns which are not thinned properly using the ZS method are shown in Fig. 23(b). These patterns are better thinned using our new rule based system as shown in Fig. 23(c).

![Figure 22 Thinning by using the ZS method.](image)

![Figure 23 Thinning: (a) an image, (b) by applying the ZS method, and (c) by applying our developed parallel one-pass 16X4 rules.](image)

### 3.4 Edge Detection One-Pass Sequential Thinning

Most of the knowledge in any digitized pattern is contained in a small fraction of pixels. These pixels are mainly formed of the central lines of the patterns. However, many iterations are required to determine the central lines as explained in the description of the
previous system. As an alternative to thinning the symbol to their central lines, we suggest to represent the symbol strokes by their edges.

Our goal is to determine pixels in a symbol that can be used for the purpose of identifying this symbol. We introduce an algorithm that produces 12 different outputs, the edges of the symbol. Generally, one output will be sufficient to identify the symbol, however, more than one output can be requested. This method also has the advantage of speed. Our algorithm requires only one iteration to thin i.e. represents the symbol by its edges. However, there may be more than one edge in a stroke (e.g., for a thick straight line there will be right, left, top, and bottom edges). Thus, we shall represent the stroke by one (or more) of these edges.

A knowledge-based system that detects certain edges of the symbol is developed. There are different outputs depending on the processing direction and the rules that are selected by the user. The user will select the edges of concern (e.g. left top, right top, left bottom, right bottom) which depend on the processing direction. The user also selects the degree of having branches (3 degrees), low number of pixels, medium number of pixels and large number of pixels. The system rules guarantee the connectivity, hence can be used for eliminating any extraneous pixels resulting from using other thinning algorithms.

3.4.1 System Performance

Five different symbols are shown in Fig. 24. Processing each line of the pattern (pixel by pixel) in the direction going from left to right and top to bottom will preserve the right and bottom edges as shown in Fig. 25. Alternatively, processing the pattern from right to left and top to bottom will preserve the left and bottom edges as shown in Fig. 26. Similarly, processing the pattern from left to right and bottom to top will preserve the right and upper edges as shown in Fig. 27. Processing the pattern from right to left and bottom to top will preserve the left and upper edge as shown in Fig. 28.

![Figure 24 Different symbols](image-url)
It is usually desired to have a small number of branches in the output pattern (Figures 25(c), 26(c), 27(c), 28(c)), however, some strokes may vanish as shown above for the letter A.

It is possible to remedy this problem by superimposing the results from two different directions. The first direction is processing the pattern from top to bottom and from left to right. The second direction is processing a copy of the original pattern from top to bottom and from right to left. Then, a union of the two outputs will have the following image:
Generaly, the selected thinning method depends on the application.

![Figure 29 The sequential thinning rules.](image)

### 3.4.2 Explanation of the Rules

The system has 12 rules listed in Fig. 29. The rules are applied to the pattern in a specific order (rule 1, 2, 3 and so on). Rule 1 prevents the system from creating unnecessary holes. The rules that preserve the connectivity are rules 4, and 7 through 11. The rules that preserve the shape and the end points are rules 2, 3, 5 and 6. Finally, rule 12 deletes the pixel if none of the previous rules are applied.
Rule 4 describes the pattern \[
\begin{array}{ccc}
  x & x & x \\
  0 & 1 & 0 \\
  x & x & x \\
\end{array}
\]
where the middle pixel should not be deleted in order to maintain the connectivity. For this pattern, only if we have \[
\begin{array}{ccc}
  0 & 0 & 0 \\
  0 & 1 & 0 \\
  x & x & x \\
\end{array}
\]
or \[
\begin{array}{ccc}
  x & x & x \\
  0 & 1 & 0 \\
  0 & 0 & 0 \\
\end{array}
\]
we could delete the middle pixel without losing the connectivity. The last two conditions (patterns) must be checked before applying rule 4, they are described by rules 2 and 3. Similarly, rule 7 describes the pattern \[
\begin{array}{ccc}
  x & 0 & x \\
  x & 1 & x \\
  x & 0 & x \\
\end{array}
\]
where the middle pixel should not be deleted in order to maintain the connectivity. For this pattern, only if we have these conditions

\[
\begin{array}{ccc}
  0 & 0 & x \\
  0 & 1 & x \\
  0 & 0 & x \\
\end{array}
\]
\[
\begin{array}{ccc}
  x & 0 & 0 \\
  x & 1 & 0 \\
  x & 0 & 0 \\
\end{array}
\]

or \[
\begin{array}{ccc}
  0 & 0 & x \\
  0 & 1 & x \\
  0 & 0 & x \\
\end{array}
\]
we should delete the middle pixel without losing the connectivity. We have to check the last two conditions (patterns) before applying rule 7; they are described by rules 5 and 6. Consequently, rules 2, 3, 5 and 6 (shown in Fig. 29) are responsible for the number of branches (extraneous pixels) in the output pattern. Each of these rules has two shaded squares. If we replace one of these shaded squares by \(x\) (don’t care), we will get thinned symbols that have a lower number of branches (strokes). If we replace both shaded squares by \(x\), we will get thinned symbols that have an even lower number of branches (strokes). Hence, the system produces three outputs. Rule 8 addresses the pattern \[
\begin{array}{ccc}
  1 & 0 & x \\
  0 & 1 & x \\
  x & x & x \\
\end{array}
\]
where to maintain the connectivity, we can not delete the
middle pixel. Rules 9, 10, and 11 are similar to rule 8. Finally, rule 12 deletes the middle pixel if none of the previous 11 rules were applied.

Some symbols before thinning are shown in Fig. 24. The symbols after thinning and without changing rules 2, 3, 5 and 6 (i.e. having ones in the shaded squares) are shown in Figs. 25(a), 26(a), 27(a) and 28(a). The symbols after thinning and changing rules (3, 4, 6 and 7) by having one x in either one of the two shaded squares are shown in Figs. 25(b), 26(b), 27(b) and 28(b). The symbols after thinning and changing rules (3, 4, 6 and 7) by having two x in the shaded squares are shown in Figs. 25(c), 26(c), 27(c) and 28(c).

We conclude that the processing direction and the selected rule method will affect the results to some extent. One of the advantages of our knowledge-based system is that we can determine which output is correct, and what rules are suitable for each application. The default scanning method processes the pattern from left to right, going from top to bottom. The default thinning rules (2, 3, 5 and 6) are the ones that have one don’t care in the shaded squares.

3.5 Results

A document that has Arabic, English, and Chinese characters is shown in Fig. 30. The document after thinning by using the developed central line rule-based system presented in section 3.2 is shown in Fig. 31. There are extraneous pixels in the resultant image. An extraneous pixel is a pixel whose deletion will not affect the continuity of the thinned pixel and it has more than one neighbor black pixel (i.e. it is not an end pixel). These extraneous pixels are located in special positions. Examples of these pixels are as follows: . The extraneous pixels can be deleted from the thinned image by applying our sequential thinning algorithm (processing from left to right going from top to bottom and having a large number of branches) discussed in the previous section. The resulting thinned patterns after deleting the extraneous pixels will be . Fig. 32 shows the document (shown in Fig. 31) after deleting the extraneous pixels.
3.6 A Combined Thinning Method

In section 3.3 we have discussed our novel parallel thinning algorithm. In section 3.4 we have discussed our new sequential thinning algorithm. In order to take advantage of both methods, we apply the parallel algorithm first, then we apply the one-pass sequential thinning algorithm to remove the extraneous pixels. The developed combined method has proved to be useful in thinning different symbols including Arabic characters in a database Kharma et al. (54)
Figure 31 (a) The document after thinning using the developed one-pass 16x4 rules parallel method, (b) part of the image is enlarged.
Figure 32 (a) The document shown in Figure 31 after removing the extraneous pixels by applying the one-pass sequential thinning algorithm, (b) part of the image is enlarged.
3.7 The Speed of the Developed Thinning Algorithm

Using C++ running on a 120 MHz Pentium PC, the fast one-pass sequential thinning algorithm (discussed in section 3.4) thins the symbols in the document shown in Fig. 30 in 2 seconds. Our central line algorithm (introduced in section 3.3) thins the same symbols in 3.18 seconds. On the other hand, using the ZS algorithm, the system thins the same symbols in 6.21 seconds. The document size is 480 x 576 pixels (12.66 x 15.24 centimeters). All the symbols including the thick upper part of the frame in the document (Fig. 30) were thinned and the width of that part of the frame is 20 pixels.

For all parallel thinning algorithms, the speed of thinning depends on the maximum thickness of any part of any symbol in the document. For example, if the thick part of the frame shown in the top of Fig. 30 were not included in the document, then our central line rule-based system thins the document in 2 seconds (instead of 3.18 seconds).

We have performed experiments to find the relationship between the speed of thinning using the central line method and the thickness of the widest part of any symbol in a document. The results show that this relationship is almost linear. An example of this relationship is shown in Fig. 33 and table 2.

![Figure 33 Thinning of symbols that have different thickness.](image)

<table>
<thead>
<tr>
<th>Thickness in pixels</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
</tr>
</thead>
<tbody>
<tr>
<td>time in seconds</td>
<td>3.13</td>
<td>6.26</td>
<td>9.72</td>
<td>13.9</td>
</tr>
</tbody>
</table>

**Table 2 Thinning speed versus the maximum thickness of a symbol in a document of size 480 x 576 pixels.**

The speed of thinning symbols in a page, for example page 69 in this thesis is 2.2 seconds. The speed of thinning the first line in the same page is 0.38 second.
3.8 Summary

A novel rule-based system for parallel thinning is implemented in this research. The system has 16 rules. These rules and their rotated versions are applied in parallel on every pixel in each iteration. The number of iterations depends on the maximum thickness part of the pattern. The resultant thinned pattern is formed of the central lines of the pattern. Moreover, as the original image is rotated in four directions, the resulted thinned image is also rotated in the same directions. Experimental results show that the developed method is very fast, effective and solves some of the problems inherent in the ZS algorithm.

A novel fast sequential thinning algorithm is also presented. The later method can be used to detect certain edges of the symbol. These edges depend on the processing direction and the selected rules (three options that determines the number of branches). This system is very fast and needs only one pass and one iteration. It has 12 independent thinning rules that preserve the connectivity and the shape of the symbol to some extent. This system also has the advantage that it eliminates extraneous pixels in any symbol that has been thinned by another thinning algorithm. Extraneous pixels are pixels whose deletion will not affect the connectivity of the thinned symbol and they pixels are not end pixels.

A combined thinning system that uses the central lines parallel thinning method followed by the fast sequential thinning algorithm are shown to be effective in producing thinned symbols which was represented by their central lines and have no extraneous pixels. The combined system is used in the thinning stage in our developed OCR algorithm, see next chapter.
Chapter 4

The Developed System

In this chapter, a novel expert system for symbol recognition will be described. The system is used for a general two-dimensional bi-level symbol analysis and recognition. The system uses the structural pattern recognition technique for modeling symbols by a set of straight lines referred to as line segments. The system rotates, scales and thins the symbol, then extracts the symbol strokes. The pixels in these strokes are represented by a chain of codes in eight directions. Finally each stroke is transferred into segments. The system is shown to be able to map similar symbol styles to the same representation.

A system that has learnt an average of 97 models per symbol, using a low threshold (radius = 15), was tested with 5726 handwritten English characters, was resulting in a recognition rate of 95% and rejection rate of 16.1%. The tested data were all the test data (binary one) taken from the Center of Excellence for Document Analysis and Recognition (CEDAR) database. When the threshold is 100 (radius = 100), the recognition rate was 87.6% and the rejection rate was 0%.

The recognition rate of our system can be increased by simply storing more models for each symbol or by increasing the rejection rate. The system is capable of learning new symbols by simply adding models for these symbols to the system knowledge base. The system is implemented by using the C++ language and is running on the 120 MHz Pentium PC.

4.1 Introduction

A new general structural pattern recognition system was developed. While successful existing pattern recognition methods are designed for a specific application e.g. Chinese symbols, or Arabic numerals recognition, our system is general in that it is designed to recognize any symbol whether it is an English character, an Arabic numeral, or language-independent, e.g. an electrical symbol.
Our system has the following characteristics: it describes the symbol itself by models. It does not use syntactic grammars but rules to rotate, scale, thin, and model symbols. One advantage of our system is that it has the capability of recognition even when few model samples are used. It measures the similarities as well as the differences between the representation of the symbol to be recognized and those of the stored models. Our system can justify its answer. Another advantage is that if a new symbol is added, our system can easily be updated by simply adding the models of the new symbol to the system knowledge base.

Our system uses different stages for analyzing and recognizing symbols. Each stage reduces the symbol details until the symbol is described by one or more representation which contain the most necessary knowledge needed to enable the symbol recognition. Not many models of a symbol are required to achieve high recognition rate. These models are stored in the system knowledge base.

4.2 The Developed System

Pattern recognition problems and chess problems are quite similar in that there is no unique solution. Methods of artificial intelligence can be applied to solve these kind of problems. Expert systems are the most suitable tools for implementing structural pattern recognition techniques because there is no exact solution for the pattern recognition problems. People use their expertise to recognize patterns. Complex patterns can be described in terms of simpler ones and simple patterns are described by sub-patterns. Expert systems help solve difficult pattern recognition problems. More rules and human experience can be added easily using rule-based systems. Hence, the system performance could be improved without rebuilding the system.

As mentioned above an expert system for two-dimensional symbol analysis and recognition is developed. Our system is a general method for recognition of bi-level symbols. Our system is not application-specific and can be used for the recognition of any symbol. These include the recognition of mathematical symbols, electrical circuit symbols, and characters such as English, Chinese, Arabic, etc.
The basic idea behind our system is that a symbol can be constructed from smaller components. Here, four basic components are used. These components are the horizontal line ' — ', the vertical line ' | ', the 45° diagonal line ' / ', and the 135° diagonal line ' \ '. These components are from now on referred to as line segments. The system transforms the symbol into a set of line segments. All these segments have the same length (in pixels). The symbol area is partitioned into square zones all of the same size. In every zone one of 16 possible configurations of segments is allowed. Fig. 34 shows the 16 different allowed configurations (including the empty zone).

Fig. 34 The 16 possible images in a zone of 10 x 10 pixels.

A symbol can be handwritten in a large number of different ways in the same space. Our system maps this enormous number of different representations of a handwritten symbol to a smaller number of possible representations. We partition the symbol into a number of square zones. There are \(2^{(L \times W)}\) possible symbols for an image that has \(W\) pixels in width and \(L\) pixels in length. We map this large number \(2^{(L \times W)}\) of symbols to just \(16^N = 2^{4N}\) symbols, where \(N\) is the number (integer) of zones used to model the symbol. This is achieved by allowing a limited number of sub-symbols in a zone as shown in Fig. 33. As an example, consider a symbol in an 10x10 pixels area which is partitioned into one zone (\(N=1\)), \(L=10\) pixels and \(W=10\) pixels. For this zone there will be 16 \(^1\) allowed symbols as shown in Fig. 34 instead of 2\(^{100}\) possible one. As another example, assume that a symbol is written within a 30 x 20 pixels where each zone is 10x10 pixels. This area will have 6 zones. Since each zone has 16 possible sub-symbols, there are 16\(^6\) allowable models instead of 2\(^{600}\) possible ones. As an example, consider the letter 'B'. One possible model
for the letter 'B' is . The top two zones are , the middle zones are , and the bottom zones are , i.e. the model has 2 zones in each row and 3 zones in each column. Our system stores the models as vectors in the system knowledge base.

When a symbol is presented to the system for recognition, the different steps that the system performs are shown in Fig. 35.

After rotating, scaling, and thinning, the system decomposes the symbol into strokes. After that, each stroke is decomposed into short straight lines (segments). A vector is used to store these segments and to represent the symbol. The distance between two
vectors enables the system to measure the differences and similarities between the two symbol representations.

4.2.1 The Rotation Stage

The objective of the first stage is to adjust tilted symbols. Symbols may be written or drawn tilted in different directions. This problem is solved by rotating the symbols. The central point (defined below) is considered as the origin of the symbol. The symbol direction is considered to be zero when its principal axis (defined below) is vertical (or having a multiple of $20^\circ$ to the vertical direction). Each symbol will be rotated (by an angle $\leq 20^\circ$) around its central point until the symbol direction, i.e. its principal axis, is at a certain direction which is vertical or having a multiple of $20^\circ$ to the vertical axis.

a) The Central Point

The physical concept of the center of mass refers to that point in an object that has the same amount of matter around it in any direction. If the origin for the entire image is considered to be the pixel at location (0,0), then the center of mass of an object C is $(Cr, Cc)$ Where $Cr = 1/n \sum x, Cc = 1/n \sum y$ for an $n x n$ image.

b) The Principal Axis

The principal axis of a bi-level object is a line passing through the object’s center of mass having a minimum total distance from all pixels belonging to the object.

4.2.2 The Scaling Algorithm

Scaling by reducing the symbol size is useful for mapping some of the different handwritten styles of the same symbol to the same representation. A straight line and lines with some deformation will have the same representation after scaling them to a certain length. Consequently, this can be useful if the above three lines represent the same symbol.
There are two types of documents. The first type includes documents that have symbols of similar sizes (such as documents of English text typed or printed by a machine). The second type includes documents that have symbols of different sizes (such as documents containing graphs, mathematical symbols, electrical circuit elements and handwritten text).

If the document has symbols of similar sizes, our system will scale the symbols so that all the symbols have the same dimensions. The new dimensions of the symbols may be selected by the user or defaulted to 32 pixels width and 48 pixels height symbol. The new dimensions are usually smaller than the dimensions of the symbols in the original document. For the case of a document that has symbols of different sizes, the new dimensions of each scaled symbol will also be the same for all symbols but these dimensions will determined by the program. Hence, the system will automatically choose certain values for the symbol height and width. Examples of these values are 32, 48, 64, 72, or 96 pixels for each of the width and the height.

The new symbol dimensions are determined by the following algorithm:

Let \( st = 16 \);

\[
\text{if (old\_height < st) then new\_height = st;}
\]

\[
\text{if (old\_height} \geq \text{st+(i\_d)} \&\& \text{old\_height<st+((i+1)*d)) then new\_height = (i+2)*st;}
\]

A symbol height (or width) less than 16 will be mapped to 16. A symbol height (or width) between [16, 96] will be mapped to 32, and between [96, 176] will be mapped to 48, ... The value of \( d \) determines the mapping size; in our program it is set to 80. Using a large value of \( d \) will map more heights to the same height.

Scaling i.e. reducing the symbol size, usually deletes some of its details but keeps the global shape, this tends to increase the opportunity of matching the considered symbol with the stored models. However, when the information in the small details is of relevance then reducing the symbol dimensions will decrease the opportunity of correct matching. Hence, the choice of the new symbol dimensions is important.
Assume that the document contains symbols with different symbol sizes. After rotation, the system scales the symbols to some predefined size (as mentioned above). The scaling algorithm is described by the following steps:

1. For each symbol, create a new empty (generally smaller) image with the predefined dimension values.

2. Find \((Sr, Sc)\), the row and column scale values as the ratios of the original and the new symbol dimensions.

3. Use a sliding window of size \((Sr, Sc)\) on the original symbol to determine the gray level of each pixel in the new symbol image. The window used to transform Fig. 36(a) to Fig. 36(b) has \((Sr, Sc) = (2.28, 1.31)\), the original dimensions are \((73, 42)\), the new ones are \((32, 32)\).

4. The new grey level value of a pixel in the scaled version is equal to the area of all ink (black) inside the window (normalized by the area of the window) in the original version, thus it is a value between 0 and 1 inclusive.

5. Since the obtained new value of the pixel according to the previous algorithm falls between 0 and 1, we use a threshold value ‘Th’ to determine whether the final value of the pixel will be black or white. If the calculated value \(\geq Th\), the pixel will be black otherwise it will be white. Possible values of the threshold are in the range \([0,1]\). Experiments show that a value for \(Th = 0.2\) is a good choice. Results when using threshold values of 0, 0.2, 0.6, 0.8 and 1.0 are shown in Fig. 37. If the original symbol is thin, using a high threshold value may cause discontinuity. However, if the original symbol is not thin, all values between \([0,1]\) can be used. Increasing the threshold value tends to thin the pattern but preserve its shape.
4.2.3 The Thinning Algorithm

Thinning, as discussed in chapter 3, is the process of reducing the thickness of each line of patterns to just a single pixel. The thinning algorithm that has been developed and described in the previous chapter has been used. Alternatively, any other effective thinning algorithm can also be used. Using our thinning method will have the advantages of better speed and rate of recognition. In addition, less models will be needed when using our thinning method.
4.2.4 Symbols Representation

The objective of this stage is to represent a symbol by a vector. The length of the vector is proportional to the details required to describe the symbol. Similar symbols will be closer to each other in the N-dimensional space.

After thinning, each symbol is described by its strokes. A stroke is a sequence of pixels that starts and ends at special pixels. A special pixel is a pixel that has three or more neighbours or exactly one neighbour. Each special pixel is marked by the letter 'A' as shown in Fig. 39(e). After marking all the special pixels, the symbol is decomposed into strokes. Then, each of the unmarked pixels in every stroke is marked by a code (an integer number that takes a value between 1 and 8 inclusive); the code in each directions is shown in Fig. 38. The code indicates the direction between the present pixel and the following one. The marked pixels are shown in Fig. 39(f).

![Figure 38 The different direction possibilities.](image)

After isolating the symbol strokes and marking each pixel by a code Fig. 39(f), a set of rules is applied to each symbol stroke to transfer it into segments (straight lines), each of fixed length. All segments have the same length in pixels. Depending on the chosen length of the segments, a stroke may be converted to one or more segments or may vanish. These segments are the symbol primitives. A vector of the primitives will be used to represent the symbol. Using shorter segments to represent a stroke preserves all details of the stroke, including unnecessary ones, while using long segments may delete important details. However, if very short segments are used then many more stored models will be needed.
Figure 39 Processing of a symbol: (a) A symbol, (b) after rotation, (c) after scaling, (d) after thinning, (e) after marking the special pixels, and (f) after isolating the symbol strokes and coding each pixel.

The use of different segment lengths (2 through 10 pixels) is shown in Fig. 40. The rules used to represent a stroke by segments are described in the following section.
4.2.5 The System Mapping Rules

In this section, we will see how different styles for a symbol, for example the letter 'A'

\[
\begin{array}{c|c}
\text{AA} & \text{AAAA} \\
\text{AAAA} & \text{A AAA A} \\
\end{array}
\]

are all mapped to

\[
\begin{array}{c|c}
\text{A} & \text{AAAAA} \\
\text{AAAA} & \text{A AAA A} \\
\end{array}
\]

i.e. to the same representation.

We will show how we transfer each stroke into one or more segments. All of these segments will have the same length $T$. In forming a segment, consecutive pixels are processed one at a time. To form a segment of length $T$ pixels, $T$ or more pixels are processed and converted to a segment (except at the end of the stroke, where $T/2$ pixels are here sufficient).

A set of rules that map a symbol stroke into segments of fixed length $T$ is developed. To find the direction of the line segment, the consecutive pixels of each stroke are analyzed pixel by pixel. For each pixel, a certain probability is assigned for each of the eight possible directions. Then, the sum of the probabilities over the previous pixels and the present one is calculated for each of the eight possible directions. If one of these sums exceeds the threshold value ‘$T$’ (the segment length), then, a complete segment is formed and the segment direction is assumed to be the direction of that of the largest probability.
sum. Next, the same analysis for the remaining pixels in the stroke continues so as to form new line segments and so on. For the remaining pixels between the end of the last formed segment and the end of the stroke, if any of the eight values exceeds half the value of the threshold it will be considered as a segment otherwise, no segment is formed. The following algorithm describes the mapping rules in more detail:

1. Select the segment length e.g. 10 pixels, which is the same as the threshold ‘T’.

2. Find the code for every pixel of the stroke as shown in Fig. 38.

3. Consider the first pixel after the special pixel of the stroke.

4. For i = 1 to 8, initialize the eight directions that the segment can take s[i] to 0.0.

5. Assign a probability 1 for the direction of the considered pixel, a probability 0.7 for its 2 adjacent directions, a probability 0.49 for the next two adjacent directions, and a probability 0 for the remaining directions. For example if the code of the pixel is 5 then p[5] = 1, p[4] = 0.7, p[3] = 0.49, p[6] = 0.7, p[7] = 0.49, and p[2] = p[1] = p[8] = 0. These probabilities were determined experimentally. For i = 1 to 8, update s[i] by adding the new p[i] to the old s[i].

6. Calculate the maximum of s[1], ..., s[8], if it is greater than or equal to the threshold ‘T’ then we form a segment of length T pixels in that direction. The length of the segment formed is always equal to T pixels. Then, consider the next pixel of the stroke, and go to step 4 to process the remaining pixels of the stroke.

7. If the end of the stroke is not reached, consider the next pixel of the stroke and go to step 5 (note that each of the s[i] is still less than ‘T’).

8. At this point, the end of the stroke has been reached. Determine the maximum of all of s[1] through s[8], if it is greater than or equal to half of the threshold ‘T’, then form a segment in that direction.

9. End of a stroke analysis. The result is a new stroke consisting of segments (straight lines), each has one of 8 directions.
10. After processing all the symbol strokes and finding all the segments, the segments are grouped so that each group corresponds to one of the 16 zones shown in Fig. 34.

11. The number of zones in each row and column will be determined and each zone shape will be determined as one of the 16 zones shown in Fig. 34.

4.2.6 The System Models and Recognition

Models for each symbol are stored in the system. Each model is represented by a vector that defines the contents of the different zones of the symbol. The number of zones is determined by the prior choice of the segment length T (straight line). The model vector contains the number of zones in each row and each column, and the total number of zones followed by a series of integers. Each of these integers (0 through 15) represents one of the 16 possible segment images (shown in Fig. 34).

Our system can use one of two methods for constructing the models. The first method uses training to extract the models. In the second method, the human designer constructs the different shapes representing the symbol, i.e. for each fixed number of zones in a row and column, the different possible representations of the symbol should be found. Then, these models are stored as vectors in the system knowledge base. In our present implementation, we used the second method but we did not include all possible representations for each symbol. The number of zones in each row ranges from 1 to 8, similarly, the number of zones in each column ranges from 1 to 8. The size of any zone in the models is irrelevant as we only need the shape of its content for later comparison with the vector representing the symbol to be recognized. The symbol to be recognized is compared with the stored models with the same number of zones in a row and column.

Models that use a large number of zones are suitable for representing symbols that have small strokes as in the case of most Arabic characters and electrical symbols. For symbols with short strokes, it is more appropriate to use short segments (i.e. a large number of zones). This, however, will increase the number of models necessary to represent the symbol. In the ultimate case, when using a 1-pixel length segment, the number of
possible representations of a symbol will be enormous and equal to that of all the possible combinations of writing the thinned symbol, which is impractical.

In this work, as mentioned above, we used the second method for constructing the system models with the guide of the first method. In our present implementation, an average of 97 models for each symbol was used. Some models are shown in Fig. 41. The first model for the letter 'a' is of size 2x2 i.e. two zones in each row and two zones in each column, a total of four zones.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>60</td>
</tr>
<tr>
<td>b</td>
<td>90</td>
</tr>
<tr>
<td>c</td>
<td>120</td>
</tr>
<tr>
<td>d</td>
<td>30</td>
</tr>
<tr>
<td>e</td>
<td>70</td>
</tr>
<tr>
<td>f</td>
<td>40</td>
</tr>
<tr>
<td>g</td>
<td>20</td>
</tr>
<tr>
<td>h</td>
<td>70</td>
</tr>
<tr>
<td>i</td>
<td>55</td>
</tr>
<tr>
<td>j</td>
<td>60</td>
</tr>
<tr>
<td>k</td>
<td>125</td>
</tr>
<tr>
<td>l</td>
<td>185</td>
</tr>
<tr>
<td>m</td>
<td>185</td>
</tr>
<tr>
<td>n</td>
<td>65</td>
</tr>
<tr>
<td>o</td>
<td>115</td>
</tr>
<tr>
<td>p</td>
<td>180</td>
</tr>
<tr>
<td>q</td>
<td>140</td>
</tr>
<tr>
<td>r</td>
<td>71</td>
</tr>
<tr>
<td>s</td>
<td>60</td>
</tr>
<tr>
<td>t</td>
<td>70</td>
</tr>
<tr>
<td>u</td>
<td>70</td>
</tr>
<tr>
<td>v</td>
<td>100</td>
</tr>
<tr>
<td>w</td>
<td>100</td>
</tr>
<tr>
<td>x</td>
<td>100</td>
</tr>
<tr>
<td>y</td>
<td>100</td>
</tr>
<tr>
<td>z</td>
<td>100</td>
</tr>
<tr>
<td>A</td>
<td>170</td>
</tr>
<tr>
<td>B</td>
<td>170</td>
</tr>
<tr>
<td>C</td>
<td>170</td>
</tr>
<tr>
<td>D</td>
<td>170</td>
</tr>
<tr>
<td>E</td>
<td>170</td>
</tr>
<tr>
<td>F</td>
<td>170</td>
</tr>
<tr>
<td>G</td>
<td>170</td>
</tr>
<tr>
<td>H</td>
<td>170</td>
</tr>
<tr>
<td>I</td>
<td>170</td>
</tr>
<tr>
<td>J</td>
<td>170</td>
</tr>
<tr>
<td>K</td>
<td>170</td>
</tr>
<tr>
<td>L</td>
<td>170</td>
</tr>
<tr>
<td>M</td>
<td>170</td>
</tr>
<tr>
<td>N</td>
<td>170</td>
</tr>
<tr>
<td>O</td>
<td>170</td>
</tr>
<tr>
<td>P</td>
<td>170</td>
</tr>
<tr>
<td>Q</td>
<td>170</td>
</tr>
<tr>
<td>R</td>
<td>170</td>
</tr>
<tr>
<td>S</td>
<td>170</td>
</tr>
<tr>
<td>T</td>
<td>170</td>
</tr>
<tr>
<td>U</td>
<td>170</td>
</tr>
<tr>
<td>V</td>
<td>170</td>
</tr>
<tr>
<td>W</td>
<td>170</td>
</tr>
<tr>
<td>X</td>
<td>170</td>
</tr>
<tr>
<td>Y</td>
<td>170</td>
</tr>
<tr>
<td>Z</td>
<td>170</td>
</tr>
<tr>
<td>0</td>
<td>170</td>
</tr>
<tr>
<td>1</td>
<td>170</td>
</tr>
<tr>
<td>2</td>
<td>170</td>
</tr>
<tr>
<td>3</td>
<td>170</td>
</tr>
<tr>
<td>4</td>
<td>170</td>
</tr>
<tr>
<td>5</td>
<td>170</td>
</tr>
<tr>
<td>6</td>
<td>170</td>
</tr>
<tr>
<td>7</td>
<td>170</td>
</tr>
</tbody>
</table>

Figure 41 Some models used by the system and the number of models used for each of these symbols.
Ideally, the stored system models should include all possible shapes for each symbol. The length of the segment controls the detail descriptions of the symbol. Using long segment length, hence few zones (such as 2x2 or 2x3) result in a fewer number of models for a symbol, but may not preserve some important symbol details. On the other hand, using short segments (hence, many zones such as 4x5 or 5x5) to model symbol will preserve the symbol details but results in a large number of models.

The system models for different symbols are shown in Fig. 41. These models include Latin letters, digits 0 to 9, the electric symbol for a diode , the Arabic digit , the Greek letter and a general symbol .

In our present implementation, for each input symbol to be recognized we use three representations (vectors). Each vector corresponds to using segment length \( T \) equal to 6, 10, or 14 pixels. Depending on the segment length \( T \) used, each tested vector will have a certain number of zones in each row and column. Each vector will only be compared to the stored model vectors which have the same number zones in each row and column. The distance between the input vector and the stored vectors is calculated by comparing each zone in the input symbol vector with the corresponding zone in the stored symbol vectors. If there is a missing or extra segment, then there is a distance equal to the length of a segment. The total distance between two models is the sum of the total length of mismatching segments.

Finally, the stored vector which has the minimum distance with the input vector will be selected as long as the minimum distance is less than a certain “rejection” threshold. If the minimum distance is larger than the threshold then the system rejects the symbol as unrecognizable (the symbol does not correspond to any of its stored models). In case of a tie, the model with the larger number of zones (i.e. with more details) will be chosen.

Some models for the character ‘A’ are shown in Fig. 42. The complete models that are used by the system are included in Appendix I.
4.3. Results

As mentioned above an average of 97 models were stored per symbol. The system was tested with 5726 handwritten English characters. When a rejection threshold value of 15 was used the recognition rate was 95% and the rejection rate was 16.1%. When a higher rejection threshold value of 100 was used, the recognition rate was 87.6% and the
rejection rate was 0%. The tested (input) data were constituted of all the different handwritten bi-level English characters data available in the Center of Excellence for Document Analysis and Recognition (CEDAR) database as well as another 120 symbols representing Arabic letters, Chinese characters, and mathematical and electrical symbols.

The accuracy obtained for each symbol and the number of test symbols are shown in tables 3 and 4. In table 3 states the results when the program is allowed to reject symbols i.e. to reject the symbols that are not recognized. In table 4 the rejection rate is zero.

<table>
<thead>
<tr>
<th>The symbol</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>The number of symbols</td>
<td>147</td>
<td>163</td>
<td>138</td>
<td>106</td>
<td>90</td>
<td>70</td>
<td>103</td>
<td>121</td>
<td>123</td>
<td>85</td>
<td>428</td>
</tr>
<tr>
<td>The number of rejected symbols</td>
<td>12</td>
<td>12</td>
<td>41</td>
<td>19</td>
<td>9</td>
<td>8</td>
<td>16</td>
<td>54</td>
<td>32</td>
<td>5</td>
<td>79</td>
</tr>
<tr>
<td>The number of misrecognized symbols</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>9</td>
<td>7</td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>The recognition rate in percentage</td>
<td>100</td>
<td>100</td>
<td>96</td>
<td>95</td>
<td>100</td>
<td>97</td>
<td>98</td>
<td>87</td>
<td>92</td>
<td>95</td>
<td>97</td>
</tr>
<tr>
<td>The rejection rate in percentage</td>
<td>8.2</td>
<td>7.4</td>
<td>30</td>
<td>18</td>
<td>11</td>
<td>16</td>
<td>45</td>
<td>26</td>
<td>5.9</td>
<td>18</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 3 The accuracy obtained for each symbol and the number of test symbols when the rejection is allowed.
The symbol

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>a</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>147</td>
<td>163</td>
<td>138</td>
<td>106</td>
<td>90</td>
<td>70</td>
<td>103</td>
<td>121</td>
<td>123</td>
<td>85</td>
<td>428</td>
<td>116</td>
</tr>
</tbody>
</table>

The number of symbols

The number of misrecognized symbols

The recognition rate in percentage

Table 4 The accuracy obtained for each symbol and the number of test symbols when the rejection is not allowed.

| b | B | c | d | e | E | f | g | h | H | l | l | J | k | l | L | m | n | O |
| 99 | 107 | 127 | 109 | 156 | 96 | 136 | 103 | 103 | 71 | 186 | 44 | 95 | 135 | 241 | 113 | 98 | 150 | 192 |
| 9 | 18 | 0 | 7 | 20 | 15 | 9 | 15 | 13 | 20 | 15 | 2 | 7 | 29 | 31 | 29 | 22 | 11 | 22 |
| 91 | 83 | 100 | 94 | 87 | 84 | 93 | 85 | 87 | 72 | 92 | 95 | 93 | 79 | 87 | 74 | 78 | 93 | 89 |

Table 5 The error rate versus the rejection rate.

The error versus the rejection rate is shown in the following table:

<table>
<thead>
<tr>
<th>Error in percentage</th>
<th>12.4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>The rejection rate in percentage</td>
<td>0</td>
<td>16.1</td>
</tr>
</tbody>
</table>

A subset (101 symbols) of this database is shown in Fig. 43(a). After rotating the symbols, some of its similar symbols become closer in shape as shown in Fig. 43(b). Next, the system scales the symbols to a predefined size as shown in Fig. 43(c). As mentioned earlier, the user has the option of selecting the size, however, here, we select the option where the system selects the new size automatically (according to the original symbol size). Then the symbols are thinned as shown in Fig. 43(d). Next, the special pixels are marked and the strokes of the symbols are isolated. At this stage, the system transfers the strokes into segments where all segments have one fixed length T. The symbol representations using segment lengths equal to 6, 10, and 14 pixels are shown in Fig. 43(e), 43(f), and 43(g) respectively. Most symbols are recognized by the system as shown in Fig. 43(h) when a threshold “rejection” distance of 100 was used. There are 5 substitutions i.e. the recognition rate is 95.04% and the rejection rate is 0%. Using a small
“rejection” distance value, as shown in Fig. 43(i), increases the recognition rate to 100%. However, it also increases the number of symbols rejected by the system to 13 symbols, so the rejection rate is 12.87%.

Generally, if a symbol is mis-recognized but a high recognition rate is required, then more models with a larger number of zones for that symbol should be added.

After studying the mis-recognized symbols we found that a symbol may be recognized incorrectly due to one of the following reasons:

1. The symbol is written in such a way that it closely resembles another symbol. In this case, the human may also be confused about the meaning of the symbol and has to study the syntax or the semantic to understand the symbol and select the closest meaningful symbol.

2. The symbol model is not in the system. This model then should be added to the system to improve the recognition rate.

3. The final models for different symbols may be similar if we use long segments (as for some handwritten u’s and v’s). In this case, we should use short segments to keep as much details as required to differentiate between them so as to increase the recognition rate.
(a) A document.  (b) The symbols after rotation.  (c) The symbols after scaling.

(d) After thinning.  (e) After modeling using 6-pixels segment length.  (f) 10-pixels segment length.
4.4 The System's Execution Time

The system is presently implemented using C++ running on a 120 MHz Pentium PC which is a relatively less expensive PC. The software code is at present not optimized. The following describes the results of the speed performance of the system. The system converts the symbols into structural feature vectors. Using the 64 symbols (the English letters shown in the middle of the document in Fig. 43) the system, on average, converts each symbol into its structural feature vector representation (final representation of the symbols by line segments) in 1.9 seconds. This includes rotating the symbol, scaling, thinning, decomposing it into strokes, coding each stroke, mapping each stroke into line segments of three different lengths and finding the representation of the line segments by zones. This results in three representations of the symbol. Each representation corresponds to using a segment length.

After finding the structural feature vectors for a tested document symbols, the system finds the stored model which has the shortest distance to each feature vector. The searching algorithm, at present, is implemented in a sequential search fashion and is not...
optimized. It takes an average of 1.51 seconds per symbol to output the typed symbol answer. However, this stage is expected to take only milliseconds if the searching algorithm is optimized and the data representing the stored models are sorted.

4.5 The Program Description

The software program size is 9800 lines written in C++ without code optimization. The program includes many utilities. After compiling the program, its size becomes 257 K bytes. The system variables are stored in another text file. Its size is 1K bytes. The system knowledge base consists of different files. Each file stores models for a certain symbol. The file size that we have used ranges from 2 K to 10 K bytes. An example of a knowledge base file that is used to store the models for the letter ‘a’ is described below.

The knowledge base file is an ASCII file and its size is 3 K bytes. The first line of the file has four entries. The first two entries for this model are ‘43’ and ‘50’, which are the width and height of the bit map pattern for the letter ‘a’. The third entry is ‘40’ which is the length of each feature vector (model). The fourth entry is ‘38’ which is the total number of models in the file. The following lines contain the bit map pattern for the letter ‘a’.

Below the bit map pattern are the models for the letter ‘a’. Each line has a different model (feature vector). The length of the feature vector is defaulted to 40 and can be changed by the user in the system variable file (the 1 K text file). Each vector represents a model representation of the letter ‘a’. The first entry in the vector represents the length of the line segment. This length can be changed by the user (in the 1K text file). The second and third entries are the number of zones in each row ‘i’ and the number of zones in each column ‘j’. The fourth entry is the total number of zones = i x j. The following entries are the codes of the possible 16 shape combinations of the zones.

The first model for the letter ‘a’ in the knowledge base file is the vector [14 2 2 4 2 1 6 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0], where the line segment length is 14, the number of zones in each row is 2 and the number of zones in each column is 2, and the total number of zones is 4. These four zones have the following
codes 2, 1, 6, and 1. The corresponding zone shapes are \[\text{shape 1}, \text{shape 2}, \text{shape 3}, \text{and shape 4}\]. Combining the four zones will have the following representation for the letter 'a' \[\text{representation of letter a}\].

Adding, editing, and deleting a model can be simply performed in the knowledge base file directly or through the program menu under the option “adding a model”. In the later case, the user enters the name of the file that has certain symbols (the file should have the same symbols but may contain different styles), then select the knowledge base file name corresponding to that symbol or select the option ‘new’ for a new symbol. Then, the program extracts the models for these symbols and adds them to the knowledge base. The user controls the model size by changing the line-segment length in the system variable file (the 1 K text file).
4.6 The System’s Limitations and Future Work

Our system has the main advantage of being capable of recognizing any symbol in any language. It can also justify its answer. On the other hand, in our system, the following characters have the same models (c, C), (u, U), (v, V), (k, K), (p, P), (s, S), (t, T), (w, W), (x, X), (y, Y), (f, F), (m, M), and (z, Z), i.e. for these characters it is at present not possible to determine whether or not it is written in lower or upper case. This problem can be solved by comparing the size and location of the symbols to determine whether the symbol is in the lower or upper case. Also, since our system performs the recognition at different stages, thus for some English characters, the scaling stage can be useful in determining whether a character is in its upper case or lower case.

In our system, we also use the same models for the following symbols (1 and l), (2, z), (5, s), (0, o, O), (q, 9), (8, B), and (g, 9). Note that, the models used for q are also used for the digit ‘9’, and the models used for the letter ‘g’ are also used for the digit ‘9’. However, the models used for the letter ‘q’ and ‘g’ are different.

Having some prior knowledge about the symbols will help to solve the previous problem. For example, in the case of the Canadian postal code, the symbol sequence is always letter, digit, letter, the hyphen symbol ‘-’, digit, letter, digit.

The current system is so general that some models for the handwritten letters ‘A’ and ‘R’ are similar. The same applies for the handwritten letters O, Q and D. This problem arises because our system may omit small strokes and approximate small changes in the strokes by straight segments. To remedy this problem, further checks may be used to determine the final decision. For example, for the case of handwritten letters ‘A’ and ‘R’, after the system recognition process, if the symbol is assumed to be A or R, then we can examine the direction and straightness of certain strokes. If the left stroke is vertical with respect to the right stroke, then it is the letter ‘R’. Otherwise it is the letter ‘A’.

Similarly, after the recognition of the handwritten letters ‘O’, ‘D’, and ‘Q’, if there is a small stroke in the bottom-right part, then it is the letter ‘Q’. Otherwise, it is the letter ‘O’ or the letter ‘D’. If there is a straight stroke in the left part then it is the letter ‘D’ and not
the letter ‘O’. The final decision will be determined after these tests have been carried out.

In future work, the employment of basic components which are different from the currently used four basic straight lines segments (/, \, l, –) will be considered to recognize more complicated symbols. These components will include symbols. Then, the symbols will be described by sub-symbols. Examples of such complicated symbols are maps, roads, and pictures.

For large symbols, e.g. symbols with many strokes such as a Chinese character, the algorithm which measures the distance can be improved by including the possibilities of deleting a row or adding a column. (The idea here is similar to that used in spell checking algorithms). In the later if a word, e.g. book was presented as bok, boak, or boook to the spell checker, then the checker would suggest adding, changing or deleting a letter. In our case, the algorithm would also suggest adding, changing or deleting a zone or more.

4.7 Recognition Rate and The Number of Models

The results of the following experiments show that it is possible to improve the recognition rate by increasing of the number of stored models. It is always possible to transfer a handwritten symbol written without any constraints into a model that consists of only line segments. The line segments lie in a limited number of zones. The zones have only 16 possible shapes. The relation between the number of models and the recognition rate for the handwritten letter ‘A’ is shown in Fig. 44 and 45 and table 6 and 7. This experiment was also repeated for the digits 0 to 9. The results are shown in table 8 and 9. The input handwritten data that we have used were extracted from the CEDAR database. The number of inputs for all digits was 1146 and for the handwritten letter ‘A’ was 139.
The previous experiments show that it is possible to increase the recognition rate by simply adding more models to the system knowledge base. It is also possible to teach the
system new models by using the symbols which were unrecognized by the system. The user in this case will simply select the symbol from the knowledge base that corresponds to the unrecognized symbol. Then, the system will add the symbol model to the appropriate knowledge base. The system will then automatically learn the new symbol and the models for this new symbol and add them to its knowledge base. For example, the average recognition rate for the characters, section 4.3, was 95.0%, however for the letter ‘A’ the recognition rate was 92.99%. We then added the models corresponding to some of the unrecognized symbols to the system’s stored knowledge base. We then reran our experiment after increasing the number of models for the letter ‘A’ from 108 to 132, and the recognition rate increased from 92.99% to 96.73% as expected.

4.8 System Characteristics

As mentioned in section 4.2, the number of all possible handwritten symbols in a zone of NxN pixels is $2^{NxN}$. This includes discontinuous symbols, however, for connected symbols, the number of possibilities is less than $2^{NxN}$ but it is still a very large number. Our system maps this large number of possibilities into 16, since only 16 shapes are allowed in a zone. However, to represent a symbol meaningfully, we need more than one zone. Fig. 46 shows the reduction in the number of possible handwritten symbols and the number of allowed symbols versus the number of (10x10) pixels zones. We have found that using 3x3 and 4x4 zones are sufficient to achieve a high recognition rate for the English characters and digits. Thus, we can use a feature vector of length 4+(3x3) or 4+(4x4) where the first four entries are:

- the segment length,
- the number of zones in each row of the model,
- the number of zones in each row of the model,
- the total number of zones in the model.
The entries that follow the first four entries are for the code descriptions of the zones. In our implementation, the default value for the length of the feature vector is 40. This allows for larger number of zones up to 7x5, 5x7 and 6x6.

![Number of zones](image)

**Figure 46** The number of allowed symbols and the number of possible symbols versus the number of zones.

An experiment was conducted to determine the time required to extract and code the symbol strokes as a function of the number of strokes. First, the document shown in Fig. 47 is thinned to that shown in Fig. 48 by applying our novel central line thinning algorithm in 2.09 seconds. Then, extraneous pixels are deleted by applying our one-pass sequential thinning algorithm (Fig. 49) in 1.27 seconds. The image size is 768x384 pixels (20.26x10.16 centimeters). The experiment indicates that the time required to extract and code the symbols shown in Fig. 49 is approximately linearly dependent on the number of strokes as shown in Fig. 50. It is not easy to find this relationship mathematically. However, it is expected that symbols that have more strokes would require more processing time to be recognized than symbols with fewer strokes.

![Symbols](image)

**Figure 47** Example of different symbols.
Another experiment was conducted to show how that the time required to map the thinned symbols vary with the length of the line segment which is specified by the user. The thinned symbols in Fig. 49 are used. It is shown that mapping the symbol strokes into short line segments is faster than mapping them into longer length segments. The time is approximately linearly dependent on the segment length as shown in Fig. 51. In our system we have used the default values of 6, 10 and 14 pixels for the segment lengths.
4.9 Summary

A rule-based system for the recognition of any 2-D bi-level line symbol has been introduced. Examples of these symbols are typed or handwritten mathematical and electrical symbols and characters such as Greek, Arabic, English or Chinese.

The system performs the recognition in different steps. The first step adjusts the tilted symbols by rotation, the second step scales the symbols to predefined sizes. This is followed by the thinning step. After thinning, each symbol is described by its strokes. Then, apply rules to transform each symbol stroke into a combination of straight lines (segments). A vector representing these segments is used to model the symbol. The distance between this vector and the vectors of stored models is used to identify the input symbol.

The system was tested with different symbols. The tested data symbols consisted of bi-level 5726 English characters taken from the Center of Excellence for Document Analysis and Recognition (CEDAR) database. In addition, there were 120 other arbitrary symbols. The recognition rate was 95.0% with a rejection rate 16.1%. In order to increase the recognition rate and decrease the rejection rate, more models should be used for each symbol.

In future, different basic components than the four basic line segments will be considered. The components will include symbols, i.e. the symbols will be described by
sub-symbols. For large symbols, the algorithm which measures the distance can be improved by including the possibilities of deleting or adding a row or a column.
Chapter 5

Conclusion

We have developed a system for general symbol recognition. Examples of these symbols are characters in any language, mathematical symbols or electrical circuit symbols. The system can be extended to achieve a 100% recognition rate by having enough models stored in its knowledge base. We have developed a novel method for thinning symbols to their central lines. The system preserves the shape of symbols.

5.1 Statistical versus Structural Pattern Recognition

Statistical pattern recognition techniques are based on pixel by pixel manipulations. Examples of these techniques are template matching, measurements of density of points, moments, characteristic loci, and mathematical transforms (Fourier, Walsh, Hadamard). Alternatively, structural pattern recognition techniques are aimed at capturing the essential shape features of characters, generally from their skeletons or contours. Such features include loops, endpoints, junctions, arcs, concavities, convexities, and strokes.

5.2 System Characteristics

We have developed a structural method for recognition of symbols formed of mainly curved lines. These symbols include characters in any language, and mathematical, electrical or other symbols.

The system is rule-based and has different rules for scaling, rotating, and thinning symbols. It has also rules for simplifying the skeleton of the symbol and representing them by a set of line segments.

Symbols are described by a set of models in the database. There is a database for each application but only one algorithm for all applications. Symbols are described by short line segments. The system measures the distance between the unclassified symbol and all stored models, then, the system selects the symbol that has the closet model to the
5.3 System Limitations

The system is capable of modeling symbols that have a limited number of strokes as in the case of English characters. If this number is large, as for Chinese characters, the number of models needed to be stored for the symbol will be also large and hence the recognition speed will be low. In our system, we have 16 possible shapes in each zone. We have used a maximum of 5x7 zones to model English characters. These models were good enough to achieve a high (95%) recognition rate. If a symbol has few strokes but some of them are too short and others are too long and both types are important for describing the symbol, then a large number of models will be needed to describe the symbol.

5.4 Contributions

We have developed a novel parallel thinning algorithm. This algorithm is implemented by using a rule-based system. The system has 16 rules. The system thins symbols to their central lines. The system produces better results than the well-known ZS algorithm.

If after thinning, some extraneous pixels remain, these pixels may be deleted by applying our new one-path sequential thinning algorithm.

We have developed a novel rule-based algorithm for handwritten symbol recognition. The algorithm rotates, scales, and thins symbols to their central lines. Then, it marks the special pixels which determine the start and end points of each stroke in the symbol. Finally, the algorithm simplifies the strokes by a set of line segments and the line segments are grouped in zones. A group of zones is used to model the symbol.

The system was tested with different symbols. The test data are the binary test data (5726 English characters) taken from the Center of Excellence for Document Analysis and
Recognition (CEDAR) database. In addition, we have added other arbitrary symbols (representing an Arabic letter, an Arabic digit, an electric symbol, a mathematical symbol, and a general symbol) as examples. The recognition rate is 95.0% with a rejection rate 16.1%. In order to increase the recognition rate and decrease the rejection rate, more models should be used for each symbol.

5.5 Future Research

In this research, we have developed a rule-based system for general symbol recognition. However, we assumed that symbols are in their isolated form. For future work, the problem of cursive handwritten character recognition should be addressed. Currently, the system can be used for applications such as postal code recognition which are written in boxes. In future, we aim at improving the system so it can read the cursive addresses as well.

Different application will be considered in future. Examples of these applications are data entered in forms, e.g., tax forms, job applications, bills, bank cheques, airline passenger tickets, and passport verifications.
References


Appendix I

The System Models

We have used the following models for the recognition of English characters. The model size ranges from 1x1 to 7x7 zones. The models for (1,l), (2,z), (5,s), 0,0), and (8,B) are the same. The models for 9 are the same as those for g and q, but the models for the letter ‘q’ are different from the models for the letter ‘g’. The following pages include all the stored models used in our system.
Appendix II
Symbols Before and After Recognition

The symbols in the left column are some binary data taken from the CEDAR database. The symbols in the middle are the recognized symbols using a threshold of 15. The symbols in the right column are the recognized symbols using a threshold of 100.