**TRAINING WITH POSITIVE AND NEGATIVE DATA SAMPLES:
EFFECTS ON A CLASSIFIER FOR HAND-DRAWN GEOMETRIC SHAPES**

By

HANAA BARAKAT

A thesis submitted to the
Department of Computing and Information Science
in conformity with the requirements for
the degree of Master of Science

Queen's University
Kingston, Ontario, Canada

May, 2001

Copyright © Hanaa Barakat, 2001

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file / Votre référence

Our file / Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-59360-6

Canada

Abstract

This thesis examines the use of positive and negative training data, on a nearest-neighbour classifier for hand-drawn geometric shapes, to improve reliability. A reliable classifier must feature the ability to reject miss-segmented and unknown shapes (ie. negative symbols). A recognition system's performance hinges on the performance and reliability of its classifier. In diagram recognition, where the crowding of line-segments and text often causes segmentation errors, a method for rejecting negative symbols is necessary to improve the performance of the system. This allows feedback from the classifier and contextual information to be used to re-evaluate segmentation hypotheses. We explore the issues involved in the development of a reliable classifier by training it on both positive and negative data, and we discuss the trade-off between reliability and correctness.

Acknowledgements

This thesis would not have been completed without a lot of help from,

God,

my Father who is in heaven,

your blessings overwhelm me, my cup runneth over.

Dr. D. Blostein, my M.Sc. supervisor, thank you very much for your patient and expert advice on all aspects of this thesis and throughout the research process. I am honoured to have worked with you these past couple of years.

Maged Barakat, my Dad, I love you. Without your never-ending love and support, I could not have begun, continued or finished this thesis.

Nagwa and Youssef, my mother and brother, thank you for being there for me from the start.

Prof. S. Quigley, my teacher at Ryerson, your kind words of encouragement and sincere interest (as well as good reference) brought me to Queen's.

Dr. D. Voit, my teacher at Ryerson and past Queen's alumni, thanks for believing in me. Your reference was invaluable!

Prof. D. Mason, my undergraduate supervisor at Ryerson, my true mentor and guide. I will always be grateful for your time, patience and excellent example.

Richard Zanibbi, my colleague in the Diagram Recognition Lab, thank you for taking the time to discuss the thesis making process.

NSERC, Canada's National Sciences and Engineering Research Council, for providing the funding for this work.

All my friends at Queen's, you know who you are, you made it fun to be here. I'll miss you! :-)

Table of Contents

Abstract	i
Acknowledgements	ii
Table of Contents	iv
List of Tables	vii
List of Figures and Illustrations	x
Chapter 1	1
<i>1.1 Introduction</i>	<i>1</i>
<i>1.2 Motivation for Domain Choice</i>	<i>3</i>
Chapter 2	5
<i>2.1 Background</i>	<i>5</i>
<i>2.2 Specifying Visual Languages: Syntax and Semantics</i>	<i>6</i>
<i>2.3 Recognizing Text & Graphics: The Lexical Phase</i>	<i>11</i>
<i>2.4 Towards Complete Systems: The State of the Art</i>	<i>26</i>
<i>2.5 Open Issues in Symbol Recognition and Document Recognition Systems in General</i>	<i>35</i>
Chapter 3	38
<i>3.1 Definitions</i>	<i>38</i>

3.2	<i>Considerations</i>	39
3.3	<i>On Reliability and Correctness</i>	45
3.4	<i>Classifier Design</i>	46
	Chapter 4	55
4.1	<i>Experimental Setup</i>	55
	Chapter 5	63
5.1	<i>Analysis of Results</i>	63
5.2	<i>About Sample Quality</i>	64
5.3	<i>Experimental Results</i>	65
	Chapter 6	69
6.1	<i>Summary, Conclusions</i>	69
6.2	<i>Future Work</i>	71
	Bibliography	73
	Appendix A: Detailed Results of Tests	77
	Appendix B: Mouse-Drawn Positive Training Samples	91
	<i>Mouse-Drawn Circles</i>	91
	<i>Mouse-Drawn Rectangles</i>	95
	<i>Mouse-Drawn Diamonds</i>	101

<i>Mouse-Drawn Triangles</i>	<i>108</i>
Appendix C: Pen-and-Paper Drawn Positive Training Samples	115
<i>Pen-and-Paper Drawn Circles.....</i>	<i>115</i>
<i>Pen-and-Paper Drawn Rectangles.....</i>	<i>120</i>
<i>Pen-and-Paper Drawn Diamonds.....</i>	<i>125</i>
<i>Pen-and-Paper Drawn Triangles</i>	<i>129</i>
Appendix D: Mouse-Drawn Negative Training Samples	132
<i>Mouse-Drawn OVER-SEGMENTED Samples.....</i>	<i>132</i>
<i>Mouse-Drawn UNKNOWN Samples.....</i>	<i>134</i>
<i>Mouse-Drawn UNDER-SEGMENTED Samples</i>	<i>136</i>
Appendix E: Pen-and-Paper Drawn Negative Training Samples.....	141
<i>Pen-and-Paper Drawn OVER-SEGMENTED Samples.....</i>	<i>141</i>
<i>Pen-and-Paper Drawn UNKNOWN Samples</i>	<i>143</i>
<i>Pen-and-Paper Drawn UNDER-SEGMENTED Samples</i>	<i>145</i>
Appendix F: Illustrations of Visual Languages and Diagrams.....	147
Glossary of Terms.....	151
Vita	161

List of Tables

Table 3-1: Comparison between our work and Anquetil et al.'s work.	44
Table 6-1: Confusion Matrix for Reject_Classifier, 0 negative training samples, mouse-drawn.....	77
Table 6-2: Confusion Matrix for Reject_Classifier, 50 over-segmented negative training samples, mouse-drawn.	78
Table 6-3: Confusion Matrix for Reject_Classifier, 100 over-segmented negative training samples, mouse-drawn.	78
Table 6-4: Confusion Matrix for Reject_Classifier, 150 over-segmented negative training samples, mouse-drawn.	79
Table 6-5: Confusion Matrix for Reject_Classifier, 200 over-segmented negative training samples, mouse-drawn.	79
Table 6-6: Confusion Matrix for Reject_Classifier, 50 unknown negative training samples, mouse-drawn.	80
Table 6-7: Confusion Matrix for Reject_Classifier, 100 unknown negative training samples, mouse-drawn.	80
Table 6-8: Confusion Matrix for Reject_Classifier, 150 unknown negative training samples, mouse-drawn.	81
Table 6-9: Confusion Matrix for Reject_Classifier, 200 unknown negative training samples, mouse-drawn.	81
Table 6-10: Confusion Matrix for Reject_Classifier, 50 under-segmented negative training samples, mouse-drawn.	82

Table 6-11: Confusion Matrix for Reject_Classifier, 100 under-segmented negative training samples, mouse-drawn.	82
Table 6-12: Confusion Matrix for Reject_Classifier, 150 under-segmented negative training samples, mouse-drawn.	83
Table 6-13: Confusion Matrix for Reject_Classifier, 200 under-segmented negative training samples, mouse-drawn.	83
Table 6-14: Confusion Matrix for Reject_Classifier, 0 negative training samples, pen-and-paper drawn.	84
Table 6-15: Confusion Matrix for Reject_Classifier, 50 over-segmented negative training samples, pen-and-paper drawn.	84
Table 6-16: Confusion Matrix for Reject_Classifier, 100 over-segmented negative training samples, pen-and-paper drawn.	85
Table 6-17: Confusion Matrix for Reject_Classifier, 150 over-segmented negative training samples, pen-and-paper drawn.	85
Table 6-18: Confusion Matrix for Reject_Classifier, 200 over-segmented negative training samples, pen-and-paper drawn.	86
Table 6-19: Confusion Matrix for Reject_Classifier, 50 unknown negative training samples, pen-and-paper drawn.	86
Table 6-20: Confusion Matrix for Reject_Classifier, 100 unknown negative training samples, pen-and-paper drawn.	87
Table 6-21: Confusion Matrix for Reject_Classifier, 150 unknown negative training samples, pen-and-paper drawn.	87

Table 6-22: Confusion Matrix for Reject_Classifier, 200 unknown negative training samples, pen-and-paper drawn.	88
Table 6-23: Confusion Matrix for Reject_Classifier, 50 under-segmented negative training samples, pen-and-paper drawn.	88
Table 6-24: Confusion Matrix for Reject_Classifier, 100 under-segmented negative training samples, pen-and-paper drawn.	89
Table 6-25: Confusion Matrix for Reject_Classifier, 150 under-segmented negative training samples, pen-and-paper drawn.	89
Table 6-26: Confusion Matrix for Reject_Classifier, 200 under-segmented negative training samples, pen-and-paper drawn.	90

List of Figures and Illustrations

Figure A: The boundary direction code.....	19
Figure B: Hierarchical Symbol Network from [30]	24
Figure C: Example of Negative and Positive Symbols	39
Figure D: A POSITIVE example of an arch and its semantic net description.....	40
Figure E: A NEAR-MISS example of an arch and its semantic network description.....	40
Figure F: A specialized arch description to exclude the near miss example.	41
Figure G: Illustration of thinning application on a hand-drawn circle.	50
Figure H: Simple smoothing operation.....	50
Figure I: Result of contour finding (and ignoring spurs) on a UML Class box.....	52
Figure J: Result of contour finding method on an under-segmented negative sample. Note that the triangle looks like it has been segmented from the rest of the region, but we treat it as if it were still part of the connected-region.	52
Figure K: Example of possible rotation before page is scanned.....	56
Figure L: Results of training on OVER-SEGMENTED mouse-drawn data.....	66
Figure M: Results of training on UNKNOWN mouse-drawn data.	66
Figure N: Results of training on UNDER-SEGMENTED mouse-drawn data.	67
Figure O: Results of training on OVER-SEGMENTED pen-and-paper drawn data.	67
Figure P: Results of training on UNKNOWN pen-and-paper drawn data.	68
Figure Q: Results of training on UNDER-SEGMENTED pen-and-paper drawn data.	68
Figure R: A utility map.	147
Figure S: An electronic circuit.	148

Figure T: Part of a scaled topographic map.	148
Figure U: Music Notation	149
Figure V: Various Types of Logos.....	149
Figure W: UML Diagram (taken from [6]).....	150
Figure X: Pixel Neighbour Identification	151
Figure Y: 3-by-3 filter.....	154

Chapter 1

1.1 Introduction

A desirable feature of a recognition system is the ability to feedback information from a later phase to an earlier phase to improve recognition results [11]. According to Kanungo et al., most diagram recognition systems have three phases [23]. The three phases are the lexical phase, the syntactic phase and the semantic phase. The lexical phase deals with the image pre-processing steps (such as binarization and noise-reduction) and primitive recognition of basic diagram constituents. The syntactic phase is where the syntax of the diagram is checked, and the semantic phase is where 2-D and 3-D understanding takes place [23]. “The three phases need not be implemented in a feed-forward fashion. Feedback from other phases could serve for resolving ambiguities or for speeding up compilation” [23]. For example, contextual information about the language syntax and semantics can be fed back into the lexical phase to provide some domain-related knowledge and improve performance. Feedback between phases of a recognition system is one way of improving recognition results.

In [3], Anquetil et al. describe a recognition system where the feedback flow from the syntactic/semantic phase is combined with feedback from the symbol classifier and

passed to the segmentation* module. Anquetil et al. show that a symbol classifier trained with positive and negative data can be used to check segmentation hypotheses, by reliably rejecting miss-segmented and unknown symbols (referred to as *negative symbols*). When a segmentation error has occurred, the symbol classifier reacts by rejecting the unknown or miss-segmented symbols. The rejection feedback from the classifier combined with syntactic/semantic information is then used to redo the segmentation.

Anquetil et al. report increased reliability when a music recognition classifier and a digit recognition classifier, which were trained on positive and negative data, are tested on both positive and negative type-set music and digit symbols. In our work, we adapt these ideas to another domain. Our goal is to increase the reliability of a hand-drawn geometric shape classifier by training it on both positive and negative data. The following questions arise.

- How should negative samples be defined?
- What fraction of the training samples should be negative samples?
- How should classifier reliability be measured?
- Do negative training samples improve classifier reliability in a cost-effective way?

* See the glossary of terms on page 151 for a definition of this term.

Even though the domain of hand-drawn geometric shape recognition offers different challenges than that of the type-set symbol recognition domain used by Anquetil et al., our results are encouraging: we do achieve an increase in classifier reliability. Since our main interest for this thesis is to investigate the effects of using negative and positive training data on the reliability of a classifier, we made our classifier as simple as possible. The details of our classifier design are in section 3.4.

1.2 Motivation for Domain Choice

Early researchers in computing anticipated a “paperless office”, where paper and pen would be replaced by electronic documents and computing programs. For informal activities at least, computer interfaces can be inflexible [2]. In the early stages of design, for example, people may prefer to brainstorm using a whiteboard rather than huddle around a computer terminal, and prefer freehand sketching on paper to the rigidity of a computer-drawing program. For rough sketches, users often prefer to explore their ideas using pieces of scrap paper over powerful software packages [2]. In the absence of imparting greater discipline to the human problem solving process, there is a need for creating tools to support the transition between formal and informal work. The research in this thesis stems from a larger project of creating a tool to recognize UML (The Unified Modeling Language [6]) notation online^{*} (ie. diagrams drawn using an electronic interface). The classifier described in this paper and used in

^{*} See the glossary of terms on page 151 for a definition of this term.

our tests is an offline* geometric shape classifier, which distinguishes between circles, triangles, diamonds, and rectangles. These shapes were taken from the basic geometric constituents of UML symbols. For example, a **class** symbol is made up of a rectangle with 0 or more horizontal lines segmenting it. Some of the arrowheads used in UML are diamonds and triangles (sometimes filled or hollow, dashed or solid). An example of a UML diagram can be seen in Figure W on page 150.

Other reasons we chose the domain of hand-drawn geometric shape recognition were:

- Industry attention to hand-drawn geometric shapes has increased due to the widespread use of Personal Digital Assistants (PDAs).
- The need for geometric shape recognition arises in many diagram recognition domains. A classifier for geometric shapes has the potential of being reused in a variety of recognition systems.

Chapter 2

2.1 Background

Visual languages have existed and have been utilized in industry for a long time. Various forms of visual languages or diagram notation are being used to communicate music, mathematics, geography, architectural plans, and more recently, computer programs [10][12][31] and system designs. Illustrations of various diagrams can be seen in Appendix F: Illustrations of Visual Languages and Diagrams on page 147.

In 1990, 55% of all mechanical engineering drawings in North America were done on a drawing board using traditional paper-and-pencil drafting techniques [23]. A similar percentage of diagrams drawn on paper probably exists in other fields that rely on visual languages for communication. Most of these diagrams, certainly in the case of technical diagrams, are constantly being updated and re-used. For this reason, it is useful and cost-effective for these diagram drawings to exist in a format that is searchable by computer, and that makes them easily accessible and updateable. In order to transfer diagrams from paper to computer, a system must read the scanned image of the drawing, then convert it from raster format to a high-level format that can be understood by CAD/CAM software. The challenge of designing and implementing such a conversion tool is to first, describe the syntax and semantics of the visual language (diagram notation), and second, design and implement a recognition system that can recognize graphics and text and whose user interface

allows human intervention to correct recognition mistakes [5]. The following sections discuss past work done with respect to these two challenges.

2.2 Specifying Visual Languages: Syntax and Semantics

A *visual language* consists of a set of symbols that can be arranged in two or three dimensions to form valid sentences of that language [29]. A *diagram* is such a sentence in the visual language and its meaning can be derived from the spatial configuration of its parts as well as from the individual meanings associated to each part/symbol [29]. In order to specify a visual language, we need to formally define its syntax (i.e. the valid ways of arranging the symbols) and its semantics (i.e. the meaning of various symbol arrangements).

Formal models of textual languages, such as regular grammars, context-free grammars, context-sensitive grammars, etc., have been around for a long time [29]. Since the late 1960s, researchers have continued to work on formal models for pictorial languages [29]. Unlike string grammars that are suited to the one-dimensional format of textual languages, the methods developed to describe visual languages have had to be able to deal with the multi-dimensionality of visual languages. (Most visual languages are two-dimensional, but there are some such as the American Sign Language that are three-dimensional.) String grammars for visual

languages proved difficult and inconvenient to use due to their linearity [5]. Alternative techniques such as graph grammars based on richer data structures were developed [5]. In addition, since visual symbols usually carry multiple meanings, the combination of visual symbols can also lead to multiple interpretations [47]. This multiplicity of meanings makes the specification of visual language semantics quite difficult [47].

The survey by Marriot, Meyer and Wittenburg [29] provides a good overview of the methods currently being used to specify visual languages. The authors categorize most of the specification approaches being used in the field into three main categories: grammatical, logical and algebraic. The rest of this section will mostly summarize the topics presented in Marriot et al.'s paper.

In the first category, Marriot et al. state that graph grammars and attributed multiset grammars are the two most interesting and generic approaches to the specification of visual languages.

Graph grammars were developed especially for image processing, and contributed more in terms of descriptive ability than did string grammars such as PDL, 2-Dimensional Context-Free Grammars and Positional Grammars (all of which are explained in [29]). Some of the graph grammars surveyed in [29] are: Web Grammars, PLEX grammars, subclasses of edNLC graph grammars (edge-labeled directed node-label controlled) and Attributed Programmed Graph Grammars.

Marriot et al. also describe some of the parsing algorithms for the grammars presented.

Attributed Multiset Grammars are general approaches to the specification of visual languages in which production rules are used to rewrite sets or multisets of symbols which have geometric (and sometimes semantic) attributes associated with them [29]. In [5] and [29], Anderson's work is presented as the earliest example of attributed multiset grammars in use. Anderson defined a co-ordinate grammar to specify two-dimensional mathematical notation [29]. The production rules assign meaning to the math notation, as well as specifying the syntax. Other kinds of attributed multiset grammars surveyed in [29] include Picture Layout grammars (which are described in full in Golin & Reiss's paper [16]), Relational Grammars, Indexed Set Grammars and Conditional Set Rewrite Systems.

Other grammar-related approaches according to [29] are Shape Grammars (which have been used to specify a variety of visual languages) and Visual Grammars (grammars which are themselves specified using a visual notation). In addition, [29] also mentions the existence of three production system-based approaches that use visual rules for the specification of pictures. These are: BitPict, Vampire, and ChemTrains. Unfortunately, grammar-based approaches to visual language specification have some disadvantages. According to the authors of [29], the two main problems are:

- Even though they are formally tractable and have been researched extensively in the field of formal language theory, grammatical approaches had to be extended to handle the specification of visual languages. This meant that most of the results in formal language theory could no longer be used.
- The semantics of spatial relations used in grammar productions of multidimensional grammar frameworks are typically not well defined, or are defined using very different formal frameworks. Since no new geometry has been defined (which is based solely on grammars) this problem has not been solved.

Marriot, Meyer and Wittenburg believe that using a **logic-based** approach to the specification of visual languages is a better starting point. The logic-based approach allows the same formalism to be used to specify both the syntax and the semantics of diagrams. It also uses first-order mathematical logic or other forms of mathematical logic and comes from the field of Artificial Intelligence [29].

There is a close connection between logic programs and grammars [29]. In fact, "Definite Clause Grammars (DCGs) ... are now an integral part of the de-facto standard of Prolog ..." [29] (Prolog is a declarative programming language commonly used in AI to write logic programs). Marriot et al. show how string grammars can be converted into logic programs. This is done by replacing each grammar rule by a Prolog rule that has the left-hand side non-terminal as its rule head and all the right-hand side symbols as predicate instances on the right-hand side. (But in Prolog rule ordering matters, and in grammars, it does not).

Constraint Logic Programming is introduced in [29], as well as the topological calculus of individuals based on connection, which was the basis of Region-Connection-Calculus (RCC). RCC was used to specify Pictorial Janus which is a visual programming language [29]. In addition, other approaches to logic-based visual language specification are surveyed. They include: MAPSEE (a system which used predicate logic to analyze and interpret sketched maps), CLASSIC (a description logic system) and Picture Logic Programs (which integrate visual expressions into the logic formalism).

The third main visual language specification approach according to [29] is algebraic-based formalisms. Marriot et al. present several of these algebraic-based methods but it is not clear what these approaches offer over and above the grammar-based and logic-based approaches. In addition, [29] examines alternative approaches to visual language specification that do not fit into the three main categories. These include: Graphic Functional Grammars, space tiling, and specialized formalisms developed for particular languages.

The descriptive power of grammars is limited, and as knowledge bases of rules become larger in response to attempts to describe more complicated structures, they become harder to maintain (and keep consistent) [43]. A variety of syntactic methods have been used to represent many kinds of diagrams, with varying success. To our knowledge, no specification method has emerged as the winner or best method. This

is probably due to the large variety of graphic symbols and applications that must be recognized and described.

2.3 Recognizing Text & Graphics: The Lexical Phase

The first phase of processing in a recognition system is the lexical phase. Within it any number of image pre-processing steps can be performed (such as noise-reduction and binarization), and text and basic graphical symbols or parts of symbols are recognized. This section will focus on some of the common techniques used in the lexical phase. Papers that use these methods and other popular techniques are listed in [11].

(a) Noise Reduction

Recognition systems deal mainly with image files as input, whether operating on data gathered online or offline. Therefore they must deal with noise^{*}. With offline methods where a scanner is often used to convert documents into computer-readable image files, imaging noise is to be expected. Dust, stains, paper creasing, and the age of documents all contribute to the presence of noise in offline data. In addition, the noise caused by too much or too little ink application affects both online and offline data.

^{*} See the glossary of terms on page 151 for a definition of this term.

Too much ink can cause the blurring of multiple symbols into one connected set of lines that are then interpreted as a single symbol. Too little ink causes gaps within symbols, which may affect the reliability of their classification. Because noise has a negative effect on classifier reliability, it is necessary to deal with noise by performing image enhancement procedures.

Many noise-reducing methods have been developed over the years. The methods developed for noise reduction include mathematical morphology and smoothing. Mathematical morphology is a method that uses sequences of dilate and erode operations to fill up small holes and erase small black regions in binarized images. Sometimes this can result in a loss of data or sharpness, such as when a period mark is erased, but there are techniques that get around this problem [32]. According to Blostein et al. [5], because of the inevitable presence of noise in all scanned documents, the segmentation phase may not produce perfectly correct results. Two methods of handling noise are reviewed in [5]:

- Sequential processing with list of alternatives (this involves producing a list of alternatives for each recognized symbol)
- Concurrent execution with contextual feedback (control mechanisms such as blackboard architecture are used amongst interleaved or concurrently executing processes)

Other noise reduction techniques such as smoothing are discussed in [7][17][32][37][38]. One of the line enhancing and smoothing methods is polygonal

approximation. Polygonal approximation (also referred to as polygonalization) is the problem of fitting a polygon to a curve while having as small a number of vertices as possible (in the fitted polygon). Polygonal approximation and curve-fitting are discussed in [17][32][36].

(b) Thinning or Skeletonization

Thinning (or skeletonization) is an image processing operation in which binary valued image regions are reduced to lines that approximate the center skeletons of the regions [32]. A common thinning approach involves inspecting each image pixel and labeling it as part of the skeleton if its deletion would break the connectivity of the skeleton. This process is done iteratively over the whole image using 3-by-3 or 3-by-4 filters to examine the candidate pixels. A detailed description of this method using 3-by-3 filters* is given in [18][36]. There are a variety of ways to find the skeleton of an image, some of which are described in [44]. For a more detailed look at these methods and others, the reader is referred to [32], which discusses basic techniques used in document processing as well as having many bibliography references which can be examined for more information. A recent comprehensive survey of thinning methodologies including parallel and sequential methods is [24].

* See the glossary of terms on page 151 for a definition of this term.

(c) Segmentation

Segmentation is the process of finding the line segments that make up the graphics and text in a document. Some of the techniques used in segmentation include (i) thresholding, (ii) connected-component labeling, and (iii) text/graphics separation.

(i) Thresholding

Thresholding is an image processing method that is commonly used to separate the foreground pixels from the background pixels of an image. Thresholding on its own is not generally a useful form of segmentation. It is usually teamed with connected-component labeling for adequate segmentation results. The simplest implementation of thresholding is bi-level thresholding or binarization. The image pixels are examined and compared to a global threshold (or adaptive threshold that changes from region to region). If the value of the pixel (a 32-bit integer indicating its color) is less than the threshold, the pixel is “erased” (colored white or the color of the background), otherwise the pixel is colored black or the color of the foreground. The following references discuss thresholding: [7][17][26][32][36].

(ii) Connected-component Labeling

Thresholding segments the image into two or more regions; the foreground and the background are separated. But it does not segment the image into specific regions that may be candidates for recognition. *Connected-component labeling* is a common way

of segmenting the image into separate objects by tracing connected image pixels. Algorithms for connected-component labeling are described in detail in [18][36].

(iii) Text/graphics Separation

Text/graphics separation is a type of segmentation used in many types of documents. A method for erasing text from engineering drawings is presented in [28]. Lu claims in [28] that his method for separating text from graphics is robust enough to handle text-graphics touching, different text fonts and writing orientations. Lu's method also handles Chinese as well as Western characters, and is capable of handling significant amounts of noise in the original image.

The method described in [28] is applied before thinning or vectorization and produces two images, one containing text, and the other containing graphics. The differences between text and graphics are realized through a set of rules set by the author and that do not always hold. The rules compare measured features such as the size of characters and length of lines/strokes, as well as the orientation of words and local stroke density of text. The algorithm detects more of the text and graphics by analyzing connected components, as well as newly connected components after the application of a brushing technique. Results of various tests of the method are presented in [28]. The performance of the algorithm depends on the value of a set of threshold variables used to differentiate between text and graphics. These thresholds in turn depend on the accuracy of the value of H_{av} , which is the average height of characters (in unit pixels). H_{av} can either be set manually or calculated automatically,

and will provide good performance if it falls within a reasonable distance from the "best" value. This method is better than most methods because it relies on a single parameter that should be set for good performance, and it attempts to automatically determine the value of this parameter. Tombre in [42] refers to older, less versatile methods for separating text from graphics.

(d) Vectorization

Vectorization^{*} (i.e. raster-to-vector conversion) is a mature field with many algorithms and systems developed and implemented; none of these methods/systems work perfectly [42]. In [42], Tombre summarizes some of the results in the field, and points to some of the post-processing methods proposed to enhance the vectorization step. In a later paper, Tombre et al. discuss several relevant factors in evaluating and choosing a vectorization method [44]. In this paper, the authors emphasize their belief that no methods for vectorization exist that are sufficiently stable and robust to work as standalone "black boxes". They concur that the robustness of a vectorization method or system may be judged by an examination of the number of parameters and thresholds needed. They also believe that in order to do performance evaluation, the methods/systems being evaluated should be tested on synthetic images rendered from selected CAD drawings, for which ground truth is available as a set of vectors. Nevertheless, Tombre et al. agree that it would be hard to decide on a clear winner when comparing different methods/systems.

^{*} See the glossary of terms on page 151 for a definition of this term.

According to [44] the main steps involved in the vectorization phase are:

- finding the lines in the original raster image
- approximating the lines found into a set of vectors
- performing post-processing to improve vectorization
- finding circular arcs (if necessary)

A variety of approaches exist to find the lines in the original raster image [44]. They include:

1. computing the medial axis*, i.e. the skeleton of the raster image
2. matching the opposite sides of the line
3. sub-sampling to get a broader view of the line (a sparse-pixel approach)

For approach 1, Tombre et al. describe in detail two methods for finding the skeleton of an image. The authors of [44] prefer to use a distance transform (especially chamfer distance) to find the set of pixels that are part of the medial axes of the lines of the drawing. These points must be linked to extract the chains making up the lines. the authors presented an algorithm to do this.

Tombre et al. also look at methods for matching opposite contours and sparse-pixel approaches similar to the method described in [27]. In the later case, the difficulty lies in choosing a sub-sampling rate (size of mesh) [44]. Although contour-matching

* See the glossary of terms on page 151 for a definition of this term.

methods have a better detection rate than skeleton-based methods, they rely on heuristics and complex matching schemes, which do not guarantee a single unique response in all cases [44].

As in Tombre's earlier paper [42], the post-processing step is touted as a method to enhance vectorization results by Tombre et al. in [44]. Several methods of doing this are presented. In the authors' opinion, the most promising of these post-processing methods is "... that of introducing models of the 'ideal' junctions (T junctions, L junctions, X junctions, Y junctions ...) and connecting each junction by fitting one of these models to it" [44].

According to Cordella and others, thinning followed by polygonal approximation is the most common vectorization method [11]. An algorithm for performing polygonal approximation on the boundary chain code is described in [36]. Castleman describes the process of boundary-tracking and chain-code production. The boundary chain-code starts with the x,y co-ordinates of a randomly chosen pixel on the boundary of the object (we assume that a boundary pixel is one that has at least one 4-neighbour pixel that is outside the region). The chain-code specifies the direction in which a step must be taken to go from the present point to the next boundary point. The eight possible locations of the next boundary point are numbered from 0 to 7. Figure A shows a common assignment of the eight direction codes.

3	2	1
4	P	0
5	6	7

Figure A: The boundary direction code

The boundary chain-code is also a space-saving method of storing image data. The silhouette of the shape can be reconstructed from the boundary chain-code alone. Pavlidis provides an algorithm for boundary tracking in [36] but it ignores spurs, so it is not appropriate for application to noisy hand-drawn data. He also provides an algorithm for polygonal approximation which can be applied to the boundary chain code and which will produce a list of vectors that approximates the best-fitting polygon according to a number of programmer-set thresholds [36]. An algorithm for boundary tracking that does not ignore spurs (ie. it trims spurs) is outlined in [18].

(e) Symbol Recognition

Symbol recognition is usually considered as the heart of a document recognition system [3]. The challenges involved in performing good symbol recognition are mainly:

- Extraction of features able to distinguish between various symbols with the added complications of noise.
- Segmentation of the image and symbols in a crowded context of lines, text and graphics.
- Adaptability of recognition method to new symbols or variants of existing symbols.

These issues are tackled throughout this chapter in various sub-sections.

(f) Feature Extraction

Features* are identifiable and measurable attributes of images. For the purpose of describing simple geometric shapes and their combinations, topological or geometric features are generally used for description [11]. An example of a topological feature is the number of holes* in the region. An example of a geometric feature is the area* of a region.

Feature extraction is the identification and measurement of features that can be useful in the recognition task. One type of feature extraction is based on mathematical image transformations such as the Fourier transform. Sometimes it can be easier to measure features in an image transform than in the image itself [26]. Some of the most popular techniques used in feature extraction are surveyed in [11]. Image transforms are described in detail in [17][26].

Since the largest family of symbols used in maps and technical drawings is composed of simple geometric shapes and their combinations [11], some of the most important feature extraction techniques are based on shape analysis [26]. One of the simplest ways to describe shapes is by shape metrics (or parametric shape features) [32]. For example, the area measurements of connected components of a document can be used

*See the glossary of terms on page 151 for a definition of this term.

to separate small regions from large ones [32]. For shape recognition purposes, features should be invariant with respect to affine transformation (translation, rotation, scale changes) [11]. Invariant geometric features include: aspect ratio*, compactness or circularity*, and triangularity*. Some of these and others are described in [7][22][26][32].

(g) Character Recognition

The field of optical character recognition is a mature field. A large number of systems and algorithms have been developed to recognize a variety of alphabets. In addition, the recognition of different fonts and hand-written text has been addressed extensively. Surveys of the area include [20][41].

(h) Graphic Symbol Recognition

Graphic symbol recognition is concerned with recognizing symbols that are not text. In most diagrams, a document recognition system must recognize three things: symbols, interconnections and annotations [9]. Most published methods do so by trying to exploit the loop structure of all layout symbols and the mostly rectilinear interconnections among the symbols.

Graphic symbol recognition has been applied in many fields. Some of the work done for these fields is surveyed in Chhabra's paper [9]. The systems developed for the

* See the glossary of terms on page 151 for a definition of this term.

following fields are summarized in [9]: Circuit Diagram Recognition, Geographic/Topographic Maps, Utility Maps / Facility Drawings, Mechanical Engineering Drawings, Architectural Drawings, Musical Scores and others.

(i) Machine Learning

The ability to reliably recognize shapes that are familiar to a classifier is a necessity for any recognition system. But in order to progress from recognition of well-known shapes to learning of new shapes and future recognition of these new shapes, aspects of machine learning must be considered and incorporated in the design of the recognition system.

In the Pattern Recognition literature, most recognition systems recognize symbols they have been trained to recognize, where training and test datasets only consist of positive symbols. Since in the “real” world, uncertainty abounds, it is not realistic to train and test recognition systems simply on positive symbols and to expect them to encounter only positive symbols in future runs. It is likely that over time, new symbols will be added to the vocabulary of visual languages or old symbols will be modified. These new symbols will have to be recognized without going to the expense of re-training the recognition systems (or worse, adding new rules). Also, it is likely that due to the segmentation errors that are commonly made in the processing of diagrams to be recognized, miss-segmented symbols will be sent to classification systems and should be rejected (instead of misclassified). The remaining part of this

section summarizes two papers that attempt to incorporate well-known machine learning concepts into recognition systems.

Messmer and Bunke present a system for the recognition and automatic learning of handdrawn graphic symbols in engineering drawings [30]. The authors propose a system that combines pattern recognition techniques with machine learning concepts in order to solve the problem of recognizing and learning graphical symbols in engineering drawings. The authors note that it is not sufficient to simply recognize symbols that are already known, but it is also desirable to automatically learn unknown symbols. "It is impossible to design a computer vision or pattern recognition system that incorporates all models and knowledge that might be needed during runtime." [30].

The system described in [30] accepts scanned grayscale images as input (with the restriction of all lines being drawn straight). Binarization, thinning and straight line segment extraction is performed. Each line segment is represented as a vertex in an attribute relational graph, which is the final output of the system. The attributed relational graph is translation, scale and rotation invariant. This handles symbols that appear in the input image at any rotation or size.

Once the graph is created, noise-handling is performed via error-correcting graph edits. Finally, recognition of symbols is performed based on sub-graph isomorphism. Learning of new symbols is based on error-tolerant sub-graph isomorphism.

The system uses a step whose aim is to minimize the number of new symbols learnt. It involves figuring out if these new symbols can be derived from existing symbols in the hierarchical symbol network (initially created offline at compile time, then augmented with new symbols during runtime). This is done using a clustering method. An example of a small hierarchical network given in the paper is included below.

Figure B: Hierarchical Symbol Network from [30]

Anquetil et al.'s paper presents a neural network based symbol classification method that can reliably reject negative symbols [3]. The authors differentiate between two types of graphic symbols, symbolics and constructed. The classification method described is for symbolics recognition.

- Symbolics are symbols that have a stable shape ratio (eg. Arrowheads, characters, digits, accidentals, noteheads)
- Constructed symbols are those that are made up of resizable primitives (such as line segments in the case of beamed notes, or square root)

Anquetil et al. describe a symbolics classifier that has good generalization properties on learned cases, in addition to rejecting all shapes that were not previously learned. This classifier is used to validate re-segmentation hypothesis made when contextual information is applied during segmentation. Contextual information is necessary due to the crowded context that symbolics are most often drawn in (such as musical notes on staff lines). The use of syntactic or semantic level information is useful in resolving ambiguities and correcting possible segmentation errors.

The classifier framework described makes use of:

- RBF (Radial Basis Function) neural networks to generate a transparent classifier.
- Fuzzy clustering for a robust learning phase.
- Genetic algorithms for feature selection.

The authors designed two systems. A typeset music notation classifier and a typeset digit classifier. They trained and tested the classifiers on two databases of symbols (split 50/50 for training and testing); one containing 2,100 music symbols and the other 12,000 digit symbols. In addition, two other databases containing reject shapes were used (1,800 badly segmented musical symbols, and 2,200 miss-segmented

printed letters and digits). Anquetil et al. reported good results for tests of these classifiers [3].

2.4 Towards Complete Systems: The State of the Art

In this section, we will survey some of the representative papers that present complete diagram recognition systems.

(a) Diagram Recognition Models

Blostein et al. [5] survey a number of diagram recognition models, which include blackboard systems and recognition kernels.

In [5], Blostein et al. state that blackboard architectures are a general and flexible framework for combining diverse knowledge sources. For example, in [46] Vaxiviere and Tombre used four levels of blackboard-based abstraction in their tool CELESSTIN, which converts mechanical engineering drawings into high-level CAD format. In CELESSTIN, a number of specialist programs written to handle feature extraction or simple recognition tasks used the blackboard model for problem solving and knowledge distribution. Other examples of blackboard based recognition systems are described in [5].

“A recognition *kernel* makes common recognition algorithms available for general use” [5]. An example of such a kernel is ADIK [35], which is summarized in [5], and [43]. Pasternak and Neumann present ADIK – Adaptable Drawing Interpretation Kernel and claim that their work is better than other systems in the field because their system can be adapted or interactively modified to handle different kinds of engineering diagrams for different application fields [35]. Although this may be a bit overstated, Tombre supports ADIK as the most thorough study of a possible generic methodology for drawing interpretation [43].

(b) Circuit Recognition

In the field of circuit diagram recognition, Okazaki et al. [33] and Lin et al. [27] propose two different methods for symbol recognition.

Okazaki et al.’s paper [33] lists two challenges that an automatic circuit diagram reader would need to overcome in order to deliver timely conversion of paper format circuit diagrams into higher level CAD/CAM format. The first is the problem of symbol recognition, and the second is the problem of system “readjustment” (or the ease with which the system can be modified for greater accuracy or additional symbol recognition) [33]. The problem of symbol recognition is addressed by classifying symbols as either loop symbols or loop-free symbols. First, loops are found by component labeling of the background, then their geometric features are calculated. A combination of template matching and feature extraction is used with heuristic rule mediation in order to recognize the symbols. This is a multi-step process performed

under a decision-tree control strategy, which Okazaki et al. claim makes readjustment of the system very easy. In addition, line tracing was used to find and identify rectangles, connection lines and loop-free symbols.

Lin et al. [27] propose a system to recognize connection diagrams using heuristic graph analysis of feature-point graphs (1st and 2nd order FP graphs) in order to extract three kinds of essential components mentioned earlier (symbols, interconnections and annotations). This step is accomplished by grouping together line segments and feature points described in two graph structures called 1st order and 2nd order FP graphs. Graphical symbols are classified into two types in [27]: loop symbol such as logic symbols and non-loop symbols. The non-loop symbols in turn contain two types of symbols: black painting symbol (constituted by black painting areas, such as diode symbols), and unclosed line symbol (constituted by unclosed line segments, such as resistor symbols).

(c) Online Recognition Systems

The need for online recognition systems in recent years has increased as more and more diagrams are drawn using electronic interfaces (such as electronic whiteboard^{*}). The following paragraphs summarize papers describing online recognition systems in the fields of geometric shape recognition, UML diagram recognition and mathematic equation recognition.

^{*} See the glossary of terms on page 151 for a definition of this term.

Jorge and Fonseca describe an online recognition system that recognizes six basic geometric shapes and two gestures [22]. The method they use measures temporal adjacency of strokes and a number of global geometric features for drawn figures. The system recognizes triangles, ellipses, circles, rectangles, diamonds and straight lines (whether drawn solid or dashed, scaled or rotated). In addition they recognize the SELECT and DELETE gestures which are indicated by drawing a closed shape around a region to be selected (SELECT gesture) or drawing zigzagged lines across a region to be deleted (DELETE gesture). The classifier described in [22] uses a decision tree to distinguish between shapes on various criteria. Fuzzy logic is also used to associate degrees of uncertainty to recognized shapes, thereby handling hand-drawing ambiguities.

Jorge and Fonseca collect input describing strokes made by a user on a digitizing tablet starting from the first pen-down event until a set timeout value after the last pen-up. Unfortunately the timeout threshold can never be adequately estimated, and it is a cause for classification error. The authors point out that their system was designed to handle only positive samples of shapes for simplicity of design. A number of invariant global geometric features are measured for each shape. The features measured include the aspect ratio, circularity, and triangularity. The program tries to recognize closed shapes (such as rectangles and circles) versus open shapes (such as polygonal lines and arcs). In order to choose a good fuzzy set description to act as a shape filter, the authors used training data drawn by three different people.

The data consisted of each of the six geometric shapes drawn using solid lines ten times, and using dashed lines five times. The chosen feature ratios produced for this data provided the needed information to describe the fuzzy sets.

The authors present encouraging results, with room for improvement. Future work included improving the recognition rate by employing better algorithms from computational geometry [22].

Lank et al. present an online system for recognition of hand-drawn UML diagrams [25]. Starting with temporal information about strokes gathered from an online medium (such as an electronic whiteboard or digitizing tablet), the system described goes through a series of steps in order to perform complete recognition. Unlike Jorge et al., Lank et al. do not start processing the collected input data until required to do so by the user. The front-end of their system (which is a retargetable part of their system and can be used in any context) acts as a draw pad and an interface for the user to invoke the recognition process as well as to correct segmentation and recognition results. The retargetable front-end kernel displays results of segmentation and recognition simultaneously by re-drawing the UML diagram with bounding boxes around connected lines that have been identified as belonging to a single UML or English text symbol, and labeling each box with the symbol label that the classifier has identified. Faced with this graphical response, the user can then provide feedback on the segmentation of the symbols (ie. correct the segmentation errors and allow the system to retry classification) and/or correct the classifications made by the system.

Lank et al.'s system goes through several phases of processing once the input has been received. "When the user invokes recognition, a retargetable segmenter performs domain independent segmentation and character recognition," [25]. This first segmentation pass attempts to join line segments into symbols while compensating for noise introduced in the drawing process. The results of the first domain-independent segmentation phase are passed to a third-party character recognizer that attempts to classify all the symbols. The second pass segmenter is applied to the results and using length of strokes per symbol, it distinguishes between text characters and graphic symbols belonging to the UML notation. This is done by assuming that the UML graphical symbols will be drawn using longer strokes than single characters. This assumption does not take into account cursive writing or extremely crowded sentences, but seems to work for the most part in Lank et al.'s system tests. Next, a domain specific UML classifier is allowed to classify the symbols previously identified as UML notation candidates by their size. Finally, a last pass domain specific segmentation process attempts to further refine segmentation and recognition results before reporting the final results to the user's screen. The user can make corrections and invoke the recognition procedure again when segmentation mistakes have been corrected.

Zanibbi's Master thesis presents a system for the online recognition of mathematical expressions [48]. Using a third party character recognition system, the Diagram Recognition Application for Computer Understanding of Large Algebraic Expressions (DRACULAE) parses the recognized characters and mathematical

symbols and produces a baseline structure tree. DRACULAE is based on a parsing algorithm that extracts baseline structure in mathematics expressions from a list of attribute symbols [48]. The implementation of DRACULAE described in [48] represents the extracted baseline structure in a tree structure and can translate it into a mathematical string language such as $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$. Zanibbi produces a representation of the hard conventions of mathematics notation (ie. those that seem to be present across dialects of maths). Using the hard conventions of direction of interpretation (horizontally from left to right) and operator dominance (interpretation always starts at the leftmost symbol), a model of spatial structure in the form of a baseline structure tree is presented. The horizontal adjacency of symbols in mathematics expressions is defined as the baseline of the expression.

Zanibbi's method for parsing mathematics expressions starts by searching from left to right for the baselines of the expression and looking at their spatial neighborhood to find sub-baselines or expressions. Syntax directed scanning is performed to produce the baseline structure tree. Example: a division operator is known to have a numerator and a denominator. This information is exploited by the parsing algorithm (recall that the parsing algorithm has received labeled symbols to look through) in order to produce the final baseline structure tree. Depth-first traversal of the baseline structure tree provides the positional sentence (linear representation of the tree). The positional sentence can be mapped to a $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ string.

DRACULAE assumes that correct recognition of characters has been performed. So in effect, Zanibbi's thesis presents a parsing algorithm for mathematical expressions that can be employed on the back-end of either an online system, as in DRACULAE, or an offline system that performs mathematics recognition.

(d) Offline Recognition Systems

In situations and domains where online processing and recognition is not possible, such as recognition of bank cheques, offline systems are needed. Without the additional information provided by online data capture (temporal or stroke information), the offline systems must do more work in the segmentation stages of the recognition process than their online counterparts. The following paragraphs summarize papers describing offline recognition systems in the fields of logo recognition and handwritten chemical formulas.

Doermann et al. present a logo recognition system which makes use of global invariant features to describe logos and perform recognition tasks [14]. They define invariants as quantities computed from an image, which remain unchanged during changes in geometric conditions such as point of view. By computing a single "signature" for each logo based on the features of interest, they can store these signatures alongside the logos they describe and use them to perform logo matching instead of trying to match the images. In this way, logos created at different scales and rotation can be matched when their signatures are compared.

Logo detection in [14] is performed using prior knowledge about the type of document being processed. Page segmentation algorithms are used to find the logo-like areas of the document for further processing. Invariant shape moments are computed along with other shape descriptors such as circularity and elongation. A method for extending shape invariants to capture similarity, affine invariants and projective invariants is presented as well. Doermann et al. claim good demonstration results when testing their system on a small database of logos.

Ramel et al.'s paper describes a method for finding and recognizing graphical entities in handwritten chemical formulas [39]. The method described is offline and is made up of two phases. The first phase involves global perception of the document, including construction of a precise description of shapes and relationships. The output of the first phase is a structural representation of the image.

The structural representation is constructed using the following steps (occurring in the first phase):

4. Vectorization and smoothing is performed using iterative application of polygonal approximation on the shapes making up the original image.
5. Vectors from opposing sides are paired together to form quadrilaterals. Vectors matching no others correspond to the contours of solid shapes.
6. The quadrilaterals formed are the nodes of the structural graph result and the arcs linking the nodes of the graph represent relationships that exist between primitives. The graph result represents the global structure of the document.

The possible relationships are: T junction, Intersection X, parallel relationships, L junction and two successive lines.

Each node is given attributes to describe the number of connections made at the extremity of the line (represented by the node).

The structural graph is input into the second phase of recognition, which performs incremental extraction of graphical entities. A series of specialist modules examine the structural graph and process it. Each specialist performs a separate kind of analysis and recognition. For example, a polygon specialist finds polygons by recursively looking left and right over the graph, starting from a line and searching neighboring lines.

Ramel et al. implemented a number of specialists which are: liason, polygon and chain, and text extraction and recognition. The authors report adequate text recognition results and good graphics recognition performance.

2.5 Open Issues in Symbol Recognition and Document Recognition Systems in General

As outlined by Chhabra [9], graphic symbol recognition has significant unsolved problems and un-addressed issues that need to be examined. They include:

- Most of the work in the field has addressed low-level recognition [9]. Tombre in [41] agrees and adds that the work that was done on high-level recognition was not complete end-to-end systems in the main, but rather assumed error-free low-level recognition and then went on from there. More work should involve error correction techniques such as using feedback from the syntactic or semantic stages [9] or using one of the methods for handling noise that are reviewed in [5].
- Proposals given of systems and techniques should consider scalability and noise degradation [9].
- More work needs to be done in finding good symbol representations, ones that are compact, incrementally extensible, computationally inexpensive for generating symbol hypothesis, capable of representing all symbol types and similar symbol prototypes with a similar representation or a shared representation.
- More work is needed on the characterization and evaluation of the performances of methods, in a domain independent manner [9][41].
- More work should be done on providing effective user interfaces for error correction [5].
- In [23], Kanungo et al. point to the lack of performance measures and experimental protocols, and recommend that a criterion for evaluation should be agreed upon, and complete experimental protocols using the specified criterion should be developed, used and reported. They offer an approach for conducting controlled experiments which involves generating ground-truth for line-drawings created using a CAD modeler, then degrading the ideal bitmap image of the line-drawing and running the recognition/3-D reconstruction algorithms on them. This

is not only applicable to the field of 3-D reconstruction from scanned engineering drawings, but should be applied in other visual language recognition systems.

- More research needs to be done in the field of model inference. The challenge is to come up with robust learning programs that can infer the syntax of a given world from looking at examples [43].

Chapter 3

In this chapter, we highlight the issues involved in training a classifier with positive and negative data samples. We compare our work with related work done by Anquetil et al. in section 3.2. We describe our method for evaluating classifier performance in section 3.3, and we describe our classifier implementation in section 3.4.

3.1 Definitions

- A *positive sample* is any symbol that the classifier is meant to recognize. In our tests, the positive symbols are hand-drawn circles, triangles, diamonds and rectangles.
- A *negative sample* is any sample that does not belong to the set of positive samples. We distinguish three types of negative samples. An OVER-SEGMENTED sample is a portion of a positive sample, as results when the positive sample is over-segmented. An UNKNOWN sample is a properly segmented symbol that is not known to the classifier. In our experiments, this is any shape that is not a circle, triangle, diamond or rectangle. An UNDER-SEGMENTED sample consists of several positive samples that are not separated during segmentation. This is illustrated in Figure C.
- A *reject option* is a method or rule that a classifier uses to reject a test sample. One common reject option is based on a reject threshold: in a nearest-neighbour classifier, a sample is rejected if the distance to the nearest neighbour exceeds the reject threshold [21].

Figure C: Example of Negative and Positive Symbols

3.2 Considerations

The use of positive and negative samples to train classifiers is not a new idea. It has been used in Machine Learning since Winston's work in 1975 on learning concepts from good and bad examples [48]. Winston describes a program that operates and learns basic building concepts in the domain of three-dimensional structures made of bricks, wedges, and other simple objects. Concepts and relationships between objects are described using a semantic network, as in Figure D. The program learns concepts by examining a carefully chosen and ordered sequence of training samples. The training data is a series of positive examples and "near-miss" examples. A positive example of the "Arch" concept is shown in Figure D, the near-miss example is in Figure E. Winston defines a near-miss example as being a version of a positive example but with a small number of significant differences. For instance, the near-miss figure of the arch has a brick P2 supported by two other bricks P1 and P3. P1 and P3 are not touching in the positive example, but are "married" in the negative

example (married is a term used to refer to the touching of faces or edges relationship). P1 and P3 are touching (married) in the negative example.

The learning program refines the description of Arch when it has examined the samples given to it. For example, it is presented with Figure D, then Figure E, which are labeled positive and near-miss respectively. Figure F shows the resulting structural description of Arch. It has been specialized to exclude the near-miss example.

Figure D: A POSITIVE example of an arch and its semantic net description.

Figure E: A NEAR-MISS example of an arch and its semantic network description.

Figure F: A specialized arch description to exclude the near miss example.

Winston's method can be challenging to apply in other domains. First, there is an infinite range of possible negative samples that can be used, and this possible set of negative samples is made even larger when the domain of recognition is hand-drawn symbols versus type-set symbols. In addition, our definition of a negative sample encompasses the "near-miss" sample definition as well as including UNKNOWN samples. Second, in some domains it is difficult to collect or construct negative data samples. The near-miss examples presented by Winston in [48] are carefully chosen. Winston recognizes that a good "teacher" is important to the progression of learning. This careful planning and selection of negative samples is hard to accomplish in noisy pattern recognition domains (such as our choice of hand-drawn geometric symbols). This is also hard to accomplish in the field of natural language understanding. In fact, statistical approaches for natural language understanding make use of a training corpus consisting of only positive samples [8].

In our experiments, we aim to determine whether negative training data can be beneficial in hand-drawn shape recognition. Anquetil et al. report good results when testing a classifier trained on both positive and negative data samples. In [3], a precise definition of negative samples is not provided, but we deduce that Anquetil et al. share our understanding and definition of negative samples. This is because they use negative sample databases that include unknown symbols and miss-segmented symbols. The negative database they use for digit recognition includes English characters as well as miss-segmented (both under and over segmented) digits.

Some of the important differences between the work presented in this thesis and that of Anquetil et al. are:

- Our classifier is written to recognize only four shape classes. Anquetil et al. demonstrate their method on a digit symbol classifier (10 classes) and a music symbol classifier (23 classes).
- Our classifier has a very simple design, so the performance is adequate but not high. Anquetil et al. implemented a sophisticated classifier and report very high recognition rates.
- Hand-drawn shapes offer different recognition challenges from those of typeset symbol recognition discussed by Anquetil et al. The main difference is in the types of noise we expect to see in the drawings. Human error and drawing irregularity are noise-contributing factors in the hand-drawn symbol recognition domain, but are not factors in the typeset symbol recognition domain. Drawing irregularity is especially apparent when different drawing tools are used. For

example, using a mouse to draw a circle is much harder than the pen-and-paper method. See Appendix A and Appendix B to compare the drawings made by the two methods.

- We did not train our classifier on every kind of negative sample available, nor did we use the whole amount of negative data available at once. We controlled these variables in our tests (ie. the negative data type used in training and testing, and the negative training dataset size). Anquetil et al. simply use all available data. For example: the music recognition classifier they tested was trained on 1,050 positive symbols and 900 negative symbols. A ratio of 1.17 to 1, positive samples to negative samples. It also appears that in addition to the positive symbols, only miss-segmented symbols were included in the negative training dataset used to train and test the music symbol classifier (no mention of unknown symbol types being included in [3]).
- We used two different drawing methods, pen-and-paper and mouse drawing. Our tests were repeated on each type of data.

A brief comparison of our classifier and Anquetil et al.'s classifier(s) is in Table 3-1.

Our results are encouraging: we do achieve an increase in classifier reliability when negative training data is used in addition to positive training data.

Table 3-1: Comparison between our work and Anquetil et al.'s work.

Comparison of	Anquetil et al.'s work	Our Work
Classification Method	RBF Neural Net	Single Nearest Neighbour
Input Description	Typeset images	Two drawing interfaces were used. Mouse-drawn images & pen-and-paper drawn images
Type of Recognition	Offline recognition	Offline recognition
Positive Classes	Music recognition (13 classes) and Digit recognition (10 classes)	Shape recognition (circle/diamond/rectangle/triangle)
Number of Positive samples	Music Symbol Database has 2,100 symbols Digit Symbol Database has 12,000 digits	400 samples (100 each of circles, triangles, diamonds and rectangles) per drawing interface
Number of Negative samples	Music Symbol Database has 1,800 symbols, and Digit Symbol Database has 2,200 printed letters	For each type of negative sample: 4 datasets were used containing 100,150,200,250 per drawing interface
Percentage of Database(s) Used for Training / Testing	50% / 50% of Positive Databases and 50% / 50% of Negative Databases	50% / 50% of Positive Database per drawing interface 50%, 33%, 25%, 20% respectively of Negative Database per drawing interface
Ratio of Positive/Negative Symbols	Music database: approx. 1.17 Digit database: approx. 5.45	Ratios used: {4, 2, 1.34, 1} per drawing interface/method
Types of Negative Symbols Used	Music database = OVER-SEGMENTED, and UNDER-SEGMENTED. Digit database = UNKNOWN, UNDER-SEGMENTED and OVER-SEGMENTED.	Tests conducted using all three types separately per drawing interface. (UNKNOWN, UNDER-SEGMENTED AND OVER-SEGMENTED)

3.3 On Reliability and Correctness

A classifier trained with positive and negative data will make three kinds of classification errors.

1. misclassify a positive sample,
2. misclassify a negative sample (by labeling as a positive sample),
3. reject a (well-segmented) positive sample.

A classifier's correctness is dependent on how many correct classifications are made. Therefore, the fewer the total number of errors made, the higher the correctness value. Correctness is usually a good measure of reliability for classifiers. However, in our tests and in [3], the total number of negative samples used in testing is much less than the total number of positive samples used in testing. In our tests we use a 4 to 1 ratio of positive to negative samples. Anquetil et al. use ratios of 1.17 to 1 for the music recognition, and 5.45 to 1 for the digit recognition. To give equal importance to the performance of the classifier on negative data as its performance on positive data, we developed a reliability measure that uses the rates of error instead of counts of error, and gives each an even weighting. Correctness and reliability together constitute a fair measure of individual classifier performance. These measures are formally defined in section 4.1(c).

3.4 Classifier Design

This section describes the classifier design used in our experiments. We believe that the results would be the same regardless of the type of classifier implemented. The following assumes a certain basic understanding of common pattern recognition techniques.

(a) Classifier Description in Brief

We implemented a single nearest-neighbour classifier for circles, diamonds, triangles and rectangles with a reject option. A reject threshold (Tr) was arrived at by examining results from a subset of the tests performed on the mouse-drawn data ($Tr = 0.17$). Using the Euclidean distance^{*} metric, when a test sample is farther than Tr from its nearest neighbour, a 3NN-like voting system is used. The three closest neighbours are allowed to vote to determine the label of the test sample. If they cannot agree, the test sample is rejected.

The input images are first binarized and thinned. Then simple polygonal approximation is applied. Although some noise-reduction is used in the pre-processing stage, we assume that most shapes are well connected and noise-free. Since our interest is to examine the effects of training on negative data, we do not concern ourselves with developing a complicated shape recognition system.

The features we measure are invariant geometric features involving the convex hull* of the shape and its bounding box* (the smallest axis parallel rectangle that encloses the shape). We use the circularity measure (P_{ch}^2/A_{ch}), rectangularity measure (A_{ch}/A_{bb}), and the triangularity measure (A_{lch}/A_{ch}). P_{ch} is the perimeter of the convex hull. A_{ch} is the area of the convex hull. A_{bb} is the area of the bounding box and A_{lch} is the area of the largest empty triangle within the convex hull. These measures were used with good results in [22] and the circularity and rectangularity measures are well known in the field of shape recognition [11].

Our classifier was written to deal only with loop-based symbols, and ignore all inner or outer spurs. This meant that we could not use in our negative training, miss-segmented symbols made up of loops and spurs because the classifier would simply measure the features around the looped parts of the symbol, ignoring the spurs.

A detailed description of the steps our classifier performs is provided in the following sections of this chapter.

(b) Input Assumptions and Description

- The input is an image file (such as a JPEG format file).
- The shapes are drawn in black “ink” on a white background (such as white paper).
- The shapes are fully connected. If not, any gaps in lines are quite small.

* See the glossary of terms on page 151 for a definition of this term.

- Any spurs (extra lines sticking out of shapes) are small and well separated from other spurs.
- Separate shapes are not intersecting and are drawn well apart from each other.
- The image is drawn by hand.
- Scanned images are scanned at 300 dpi (dots-per-inch).

(c) Output Description

Text labels classifying each object in the image as Circle, Rectangle, Diamond or Triangle. When a reject option is used, rejected shapes are labeled Reject.

(d) Pre-processing Steps

The steps described here are the result of trial-and-error implementations of the classifier. These steps perform simple noise-reduction and prepare the image for further processing.

- The input image is binarized. A global threshold of 128 (on a grayscale of 256 colors) is chosen to perform this in our tests. Any pixels whose color value is greater than or equal to 128 are erased (colored white), and all others are colored black.
- The binarized raster* of the image is formed and used in later steps.

- A dilate operation is performed followed by a erode operation. This noise-removing step removes any small holes* in the image.
- The image is thinned* down to single pixel wide lines. See Figure G for an illustration of thinning.
- A retract operation is performed which removes a couple (exactly 2) pixels from line ends, if there are any. This cleans up spurious pixels from line ends that hide the direction of the line.
- The line ends are extended by 10 pixels in the direction the last two pixels are pointing towards.
- The dilate and erode operations are applied again in sequence. This closes line segments that have been extended but are still not touching because of a small gap between them. For example: parallel lines with only one or two pixels width space between them.
- Thinning is applied again to reduce lines back to single pixel width.
- A simple smoothing operation is performed. Pixels that are “sticking out” in a 3-by-3-pixel filter* are erased and the middle pixel is colored black to prevent line breaks. See Figure H for example of filter used and effect of smoothing.

* See the glossary of terms on page 151 for a definition of this term.

Figure G: Illustration of thinning application on a hand-drawn circle.

Figure H: Simple smoothing operation.

(e) Vectorization and Segmentation Steps

(e) Vectorization and Segmentation Steps

- Perform connected-component finding and labeling. Each connected-component (region or shape) is labeled by a unique shape number starting from Shape 0. This is to aid in classification and result reporting.
- Find holes in the connected regions.
- For each region, create a chain-code* that follows around the outside black border pixels.
- For each region, create a chain-code that follows around the inside (hole-surrounding) black pixels. (Only if the region has holes).
- Form a new chain-code that is the set of pixels that are common in both the boundary-chain-code and the inner-chain-code(s). This new set represents all the points that form the boundary contour while ignoring all spurs in the shapes. See Figure I for an example of the effect of these last 3 steps. This method will also split two loop-based shapes that were connected with a single line into two separate shapes, with the connection line ignored. The current version of our classifier does not check for this and treats the sub-segmented shapes as one “connected” shape. In our future work, we would check for this effect and classify each sub-shape separately. An example of this effect can be seen in Figure J.
- Perform polygonal approximation over the new boundary chain-code. Produce a new vector list that represents the smoothed regions.

* See the glossary of terms on page 153 for a definition of this term.

Figure I: Result of contour finding (and ignoring spurs) on a UML Class box.

Figure J: Result of contour finding method on an under-segmented negative sample. Note that the triangle looks like it has been segmented from the rest of the region, but we treat it as if it were still part of the connected-region.

(f) Feature Extraction Steps

For each region, the following steps were performed:

- Find the convex hull* points of each shape (defined by the smoothed vectors from the last step). Convex hull was found using the Graham Scan method described in [34].

*See the glossary of terms on page 151 for a definition of this term.

- Find the area of the convex hull by summing the triangles formed by a simple application of triangularization* on the convex hull.
- Find the largest empty triangle* within the convex hull.
- Measure the following features: Circularity (perimeter squared of the convex hull divided by the area of the convex hull), Rectangularity (area of the bounding box divided by the area of the convex hull), Triangularity (area of the largest empty triangle within the convex hull divided by the area of the convex hull)
- If the current phase is Training, the user is asked to label the shapes. Then the user is asked for the file name to which the training feature vectors will be saved. If the current phase is Testing, the user is asked to enter the training data set file name that will be used to perform classification.

(g) Classification Steps:

Use the Euclidean distance measure to calculate the distance between the new test feature measurements and the training data. If the nearest labeled point in feature space is more than 0.17 away, the classification of this point is the result of a voting procedure between the 3 closest points in feature space, or Reject if none of the points agree. If the nearest labeled point is less than or equal to 0.17 then the test point is labeled with the same class as its nearest neighbour.

The reject option threshold¹ of 0.17 was chosen by examining a sub-set of the distance measurements made for mouse-drawn test data. Note: the choice of reject option has a significant effect on the reliability of a classifier when the classifier has been trained only on positive data. Our results show that training with positive and negative data has a much greater effect on reliability than the reject option threshold.

¹ The choice of this threshold can be automated for different kinds of classifiers, but not for a single nearest-neighbour classifier.

Chapter 4

In this chapter we describe how we collected our training and test data and the tests that we conducted. The classifier evaluation functions are formally defined in 4.1(c).

4.1 Experimental Setup

(a) Positive Mouse-Drawn Data Collection

We asked ten persons to draw ten repetitions of circles, diamonds, triangles and rectangles (our positive classes). The volunteers were asked to use a paint software package such as MS Paint and draw the shapes at any rotation or scale. The conditions placed on the drawings were that they be drawn hollow, connected and with no intersections between shapes. As well, we asked the volunteers to draw the shapes a reasonable distance from each other (ie. we asked for well-separated shapes). We collected 400 mouse-drawn samples in total.

(a) Positive Pen-and-Paper Drawn Data Collection

Ten persons were asked to draw ten repetitions of circles, diamonds, triangles and circles, using black pens and white paper. The images were then scanned at 300 dpi (dots-per-inch). The conditions placed on the volunteers were the same as those placed on the mouse-drawn images volunteers. One of the main differences in drawing style and data collection between this method and the mouse-drawn

collection method is the orientation of the drawing canvas. The drawing canvas for the mouse-drawn volunteers was electronic and oriented in the direction of the monitor (ie. usually close to perpendicular to the horizontal surface it is placed on) and the drawing canvas was not rotated during the drawing process. The pen-and-paper method allowed the volunteers to rotate and orient the paper in any direction they liked as they drew. Some people were observed drawing at several different orientations, example: portrait and landscape, and other various angles. We did not record the drawing orientation, and the images were scanned in portrait format. See example in Figure K, which shows a clockwise rotation of 90 degrees from landscape to portrait. All images were scanned in portrait format regardless of the original orientation of the drawer's hand, and the images were not necessarily rotated clockwise. This was necessary because various volunteers rotated the pages several times while drawing the images (probably because it is easier to draw rotated shapes by rotating the paper before drawing the images).

Figure K: Example of possible rotation before page is scanned.

(b) Negative Data Collection

Negative data was collected using both mouse-drawn and pen-drawn input. Two sets of twelve files were created for training. Two sets of three files were created for testing. Details are given next.

The negative training data consisted of two sets of twelve files. The first set of files contained samples that we drew in MS Paint for each type of negative sample (OVER-SEGMENTED, UNKNOWN, and UNDER-SEGMENTED). The second set of files contained samples that we drew with black pen on white paper and scanned at 300 dpi. Each set of twelve files contains the following:

Note: Appendix D: and Appendix E: contain a subset of the negative training data we used for mouse-drawn and pen-and-paper drawn samples.

File 1: 50 OVER-SEGMENTED samples.

File 2: 100 samples. 50 OVER-SEGMENTED samples, plus the samples in file 1.

File 3: 150 samples. 50 OVER-SEGMENTED samples, plus the samples in file 2.

File 4: 200 samples. 50 OVER-SEGMENTED samples, plus the samples in file 3.

File 5: 50 UNKNOWN samples.

File 6: 100 samples. 50 UNKNOWN samples, plus the samples in file 5.

File 7: 150 samples. 50 UNKNOWN samples, plus the samples in file 6.

File 8: 200 samples. 50 UNKNOWN samples, plus the samples in file 7.

File 9: 50 UNDER-SEGMENTED samples.

File 10: 100 samples. 50 UNDER-SEGMENTED samples, plus the samples in file 9.

File 11: 150 samples. 50 UNDER-SEGMENTED samples, plus the samples in file 10.

File 12: 200 samples. 50 UNDER-SEGMENTED samples, plus the samples in file 11.

Two sets of negative test data were created, each consisting of three files.

The first set of negative test data was created using a mouse as described earlier. This set contains three test files:

Test File 1: 50 OVER-SEGMENTED samples.

Test File 2: 50 UNKNOWN samples.

Test File 3: 50 UNDER-SEGMENTED samples.

The second set of negative test data was created using pen and paper as described earlier. This set contains three test files.

Test File 1: 50 OVER-SEGMENTED samples.

Test File 2: 50 UNKNOWN samples.

Test File 3: 50 UNDER-SEGMENTED samples.

(c) The Evaluation Functions

In order to compare the performance of the classifiers, we developed a reliability evaluation function (P) based on the costs attributed to various classifier errors, and a function to measure classifier correctness (C). Our reliability function is adapted from [13]; a paper that discusses how to define and evaluate the rejection option on a neural-network.

The reliability measure (a number between 0..100)

$$P = 100 - (Cep * Rep + Crp * Rrp + Cen * Ren) / 2$$

Cep = Cost of error on a single positive sample (ie. misclassification error).

C_{en} = Cost of error on a single negative sample (ie. classifying it as a positive sample).

C_{rp} = Cost of rejecting a single positive sample.

R_{ep} = Rate of errors made on positive samples (ie. misclassification errors) = $(N_{ep} / T_p) * 100$.

R_{en} = Rate of errors made on negative samples (ie. classifying as positive) = $(N_{en} / T_n) * 100$.

R_{rp} = Rate of rejection of positive samples = $(N_{rp} / T_p) * 100$.

The correctness measure (a number between 0..100)

$$C = 100 - ((N_{ep} + N_{rp} + N_{en}) / (T_p + T_n)) * 100$$

N_{ep} = Number of errors (misclassifications) made on positive samples.

N_{en} = Number of errors (misclassifications) made on negative samples.

N_{rp} = Number of rejected positive samples.

T_p = Total number of positive **test** data.

T_n = Total number of negative **test** data.

Note that when all the costs are equal to 1 and $T_p = T_n$, the two functions C and P become equivalent. Recall from section 3.3, that we use P as well as C because T_p is greater than T_n . (In our tests, we set $C_{ep}=C_{en}=C_{rp} = 1$, $T_p = 200$, $T_n = 50$).

(d) Tests Conducted

Tests conducted have the following characteristics:

- Reject Ability. The classifier can either be a “No Reject Classifier” (classifier that never rejects), or a “Classifier with Reject” (classifier described in section 3.4).
- Type of Data. This can be either mouse-drawn or pen-and-paper drawn.
- Number of Positive Training Samples. All our tests use 200 positive training samples.
- Number of Positive Test Samples. All our tests use 200 positive test samples.
- Type of Negative Data. There are three types of negative data. In each test, the type of negative data used to train the classifier is also used to test it (same type, not same data). Recall from section 3.1 that the three types are: UNKNOWN, UNDER-SEGMENTED, and OVER-SEGMENTED.
- Number of Negative Training Samples. Our tests use 0, 50, 100, 150, or 200 negative training samples. These datasets are described previously in this chapter, section 4.1(b) Negative Data Collection.
- Number of Negative Test Samples. All our tests use 50 negative test samples (of the same type as those used in the training phase).

In total, there are $2 * 3 + 2 * 3 * 5$ possible combinations = 36. We tested all of these and report our results in Chapter 5.

The following examples illustrate how we describe a particular test:

No Reject Classifier / Pen Drawn / 200 Positive / 0 Negative OVER-SEGMENTED

This describes a test where the classifier was not operating with a reject option. It is trained and tested on pen-and-paper drawn data. Two hundred positive samples and zero negative samples were used to train the classifier. Two hundred new positive samples and 50 new OVER-SEGMENTED samples were used to test it.

Another example:

Classifier with Reject / Mouse Drawn / 200 positive / 100 negative UNKNOWN

This describes a test where the classifier was operating with a reject option (consisting of a rejection threshold = 0.17, followed by a 3NN-like voting method as described previously). It is trained and tested on mouse drawn data. Two hundred positive samples and 100 negative samples of the UNKNOWN type were used to train the classifier. Two hundred new positive samples and 50 new negative samples of UNKNOWN type were used to test it.

Choices in column 1: "No Reject Classifier" or "Classifier with Reject"

Choices in column 2: "Pen Drawn" or "Mouse Drawn"

Choices in column 3: always "200 positive"

Choices in column 4: "0 negative" or "<number><type>" where <number> is 50, 100, 150, 200 and <type> is UNDER-SEGMENTED, UNKNOWN, OVER-SEGMENTED.

We repeated our tests on both mouse-drawn data and pen-and-paper data to see if the classifier would perform differently when given input that has been collected using different media. In our future work, we also want to try these tests on data drawn on a whiteboard with a marker. The main differences among the three input methods is summarized by examining the input medium and the drawing canvas. The input medium can be a mouse or a pen (markers and ballpoints are pens). The drawing canvas can be written on or simply used as a display (the screen is a display, but the paper and whiteboard are both displays and writing canvases). The drawing canvas

can be manipulated by the drawer (paper can be rotated or moved to accommodate writing style, but whiteboards and screens cannot). The drawing canvas can be horizontal or vertical (screen and whiteboard are vertical, paper is usually horizontal). We believe that the method used to draw the shapes affects the drawing style, which may in turn affect the classifier performance.

The objective of changing the size of the negative training dataset in our experiments is to determine what the ideal ratio of positive to negative symbols should be for good reliability and correctness results.

We created separate test sets for the three kinds of negative symbols instead of combining them into a single set because we wanted to isolate the effects of the different kinds of training, and observe the results separately. (In Anquetil et al.'s test, the negative datasets contained a mixture of the different types of negative symbols).

Chapter 5

This chapter presents an analysis of our results, which we give in detail in section 5.3.

We also discuss the issue of sample quality in section 5.2.

5.1 Analysis of Results

We found that:

- The inclusion of negative training in the design of our classifier in addition to the use of a reject option gives a more reliable classifier than one that relies only on positive training data and a reject option.
- In all our tests, we found that increasing the size of the negative training set relative to the size of the positive training dataset gives increased classifier reliability. (See the graphs of Figure L through Figure Q. Notice the upward trend of the Reliability function).
- Correctness of the classifier usually decreases as the size of the negative training dataset increases with respect to the size of the positive dataset. We surmise that the size of the negative training dataset should not exceed 50% of the size of the positive training dataset. An examination of the graphs in Figure L through Figure Q show that when the size of the negative training dataset increases beyond 50% of the size of the positive training dataset, the increase in reliability tapers off compared to the increase before the 50% point.

5.2 About Sample Quality

Patrick Winston, in his work about a program that learns from positive and near-miss examples, discusses the importance of a “good teacher” that can give “good examples” to aid learning [48].

When creating our training datasets, we did not spend any time choosing good examples. We allowed our classifier to be trained on different users’ drawings of circles, triangles, diamonds and rectangles. We did not weed out shapes that, to our eyes, were inadequate (misshapen) representations of positive shapes. We allowed squares to be drawn at any rotation, which meant that our classifier was being asked to recognize a shape (the square) that might be classified by a human classifier as either a rectangle or a diamond without any additional information. We chose to create negative samples that could also be justifiably misclassified without contextual information (for example: the parallelograms labeled as UNKNOWN negative data samples could very easily pass for diamonds).

Our lack of attention to choosing good samples was based mainly on the expectation that in the hand-drawn UML domain, very few people draw exactly and precisely. Software designers cannot be expected to learn careful drafting techniques, so a robust shape classifier is needed.

We also used very few kinds of negative samples within each type. For instance, in the over-segmented type, we used four kinds of over-segmented shapes.

- 1- The over-segmented circle,

- 2- the over-segmented diamond,
- 3- the over-segmented triangle,
- 4- the over-segmented rectangle.

Some of the shapes within these kinds look exactly like others. A diamond that is missing two of its sides due to over-segmentation, looks a lot like a triangle that is missing one of its sides. In contrast, Anquetil et al. had more than 36 sub-types of negative samples in the negative digit database (at least 26 characters and 10 miss-segmented digits) [3]. For the negative music database, Anquetil et al. used at least 13 different sub-types of negative samples (miss-segmented versions of the 13 positive symbols)[3].

We believe that the issue of sample quality has an effect on the reliability of a classifier. More work needs to be done in this area before we can describe this effect.

5.3 Experimental Results

Figures L to Q summarize the results by showing correctness and reliability of the various classifiers. Detailed results, in the form of confusion matrices, are given in Appendix A.

When reading the graphs in the following pages, note that the numbers at the end of the column labels indicate the negative training dataset size. For instance, in Figure L, Reject_Classifier_0 was trained on 0 negative UNDER-SEGMENTED mouse-drawn data samples, while Reject_Classifier_50 was trained on 50 negative UNDER-SEGMENTED mouse-drawn data samples. All classifiers were trained on 200 positive samples, and

Figure L: Results of training on OVER-SEGMENTED mouse-drawn data.

Figure M: Results of training on UNKNOWN mouse-drawn data.

Figure N: Results of training on UNDER-SEGMENTED mouse-drawn data.

Figure O: Results of training on OVER-SEGMENTED pen-and-paper drawn data.

Figure P: Results of training on UNKNOWN pen-and-paper drawn data.

Figure Q: Results of training on UNDER-SEGMENTED pen-and-paper drawn data.

Chapter 6

6.1 Summary, Conclusions

We set out trying to discover if the good classifier reliability results shown in [3] due to training with positive and negative data would be repeated in the more complex domain of hand-drawn symbol recognition. We found more questions than answers. Here is a discussion of four main issues.

1. When training with negative samples as well as positive samples, what type of negative samples should be used (domain-relevant or non-domain-relevant, under-segmented or over-segmented, or both)? What combination of negative types should be used together in training, and in what proportion to each other?

We are in the process of testing and training our classifier with combinations of all three different types of negative sample data, because such is the type of negative data that our classifier will have to deal with as a result of segmentation error. However, we cannot yet tell what the ideal proportion of the different kinds of negative symbols should be within the negative training dataset. We anticipate that this probably depends on the frequency of the respective segmentation errors. For example, if the segmentation process makes more UNDER-SEGMENTATION errors than the other two types of errors, their frequency within the negative dataset should be proportionately larger.

2. What should the ratio of positive to negative samples be in order to increase classifier reliability?

Based on our tests, we can surmise that the total number of negative samples used in training should not be greater than half of the total number of positive samples used in training, if a balance between reliability and correctness needs to be achieved.

3. What is a good measure of classifier reliability?

The measure of classifier reliability that we used depends equally on the performance on both positive and negative samples (see sections 3.3 and 4.1(c)). By summing the rates of error made on negative and positive data, as well as using a simple count of correct classifications made, we can summarize the performance of a classifier and compare its performance with others.

4. Is the reliability of a classifier made better with inclusion of negative data in training, or is this a more costly way of achieving comparable results to a classifier employing a simple reject threshold?

The inclusion of negative training data in the training phase of the classifier always yields an increase in reliability, but with a certain decrease in correctness. We are also aware that our datasets contain only small amounts of data. Future work includes

applying our tests on much larger datasets to see if our results/conclusions will be repeatable.

6.2 Future Work

- Redo our tests with a larger dataset of positive and negative samples. This will give us more confidence in the conclusions we make based on these results.
- Repeat our tests on data that was drawn using a whiteboard. Pictures of the images would be taken using a digital camera and then passed to our classifier.
- Improve performance of the classifier by augmenting the feature vector with other global invariant shape descriptors, such as shape moments [1].
- Increase the number of positive classes in our test domain to include shapes from UML notation.
- Improve the polygonal approximation by trying more complicated curve-fitting techniques (that are not based on straight-line fitting only). This should improve noise handling and smoothing.
- Automate the selection of thresholds for the following processes: binarization, polygonal approximation, and rejection.
- Implement a corner finding algorithm and use it in a decision tree layer for improved recognition results.
- Add another layer of recognition to the classifier that makes use of contextual information about the UML domain. For example, a possible improvement is to take the results of the current classifier and pass them through a decision tree based process. The process examines line segment connections to correct

classification mix ups between rectangles and diamonds. If the shape is connected to other shapes at its corner points, then it is a diamond. Otherwise, if it is connected from the sides then it is a rectangle (ie. once a shape has been labeled as a diamond or a rectangle, the decision tree can do an extra layer to “check” the result).

- Combine the current offline shape recognition classifier with an online UML input interface such as the retargetable front-end described in [25]. The resulting hybrid recognition system would provide better results than a fully offline system due to the addition of online input data as well as online user feedback.
- Design and implement a method of learning from mistakes. The classifier becomes a dynamically trained classifier. Data from current segmentation errors may be used to augment negative sample databases and refine recognition results. For example: if the user often draws arrows in which the head and tail do not connect, the program could learn this from user feedback on segmentation errors.
- Use concepts from the field of Machine Learning such as those used in [30] to improve recognition features to learn unknown symbols instead of rejecting them. This is especially useful for the domain of UML recognition, because the Unified Modeling Language is extensible.
- Construct a description of UML syntax in order to improve recognition (of a full UML recognition system).

Bibliography

- [1] Alt, F. L., "Digital Pattern Recognition by Moments." *Proc. Journal of the ACM*, 9, 240-258, 1962.
- [2] Arvo, J., "Computer Aided Serendipity: The Role of Autonomous Assistants in Problem Solving." *Proc. Graphics Interface '99*, Kingston, Ontario, 183-192, June 1999.
- [3] Anquetil, E., Couasnon, B. and Dambreville, F., "A Symbol Classifier Able to Reject Wrong Shapes for Document Recognition Systems." *Proc. Third IAPR Intl. Workshop on Graphics Recognition*, Jaipur, India, 195-202, 1999.
- [4] Blostein, D., "General Diagram-Recognition Methodologies." *Graphics Recognition: Methods and Applications*. Lecture Notes in Computer Science, Vol. 1072, Springer, 106-122, 1996.
- [5] Blostein, D., Lank, E. and Zanibbi, R., "Treatment of Diagrams in Document Image Analysis." *Theory and Application of Diagrams: (Proc. of First International Conference, Diagrams 2000)* Lecture Notes in Artificial Intelligence, Vol. 1889, Springer Verlag, New York, 330-344, 2000.
- [6] Booch, G., Rumbaugh, J. and Jacobson, I., *The Unified Modelling Language User Guide*. Addison Wesley Longman Ltd., Reading, Massachusetts, USA, 1999.
- [7] Castleman, K. R., *Digital Image Processing*. Prentice-Hall, New Jersey, USA, 1997.
- [8] Charniak, E., *Statistical Language Learning*. MIT Press, Massachusetts, USA, 1993.
- [9] Chhabra, A. K., "Graphics Symbol Recognition: An Overview." *Graphics Recognition: Algorithms and Systems*, (Selected Papers in Proc. of Second Intl. Workshop, GREC '97, Nancy, France) Lecture Notes in Computer Science, Vol. 1389, Springer, 68-79, Aug. 1997.
- [10] Cockburn, A. and Bryant, A., "Leogo: An Equal Opportunity User Interface for Programming." *Journal of Visual Languages and Computing*, 8, 601-619, 1997.
- [11] Cordella, L. P. and Vento, M., "Symbol Recognition in Documents: A Collection of Techniques?" *International Journal on Document Analysis and Recognition*, 3, 73-88, 2000.
- [12] Costagliola, G. and Orefice, S., et al. "On the Generation of Interactive Iconic Environments." *Intl. Journal of Human-Computer Studies*, 50, 363-390, 1999.

- [13] De Stefano, C., Sansone, C. and Vento, M., "To Reject or Not to Reject: That is the Question – An Answer in Case of Neural Classifiers." *Proc. IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, **30** (1), 84-94, Feb 2000.
- [14] Doermann, D. S., Rivlin, E. and Weiss, I., "Logo Recognition Using Geometric Invariants." *Proc. Second International Conference on Document Analysis and Recognition*, Tsukuba City, Japan, 894-897, Oct. 1993.
- [15] Duda, R. O., Hart, P. E. and Stork, D. G., *Pattern Classification*, Second Edition, John Wiley & Sons, New York, USA, 2001.
- [16] Golin, E. J. and Reiss, S. P., "The Specification of Visual Language Syntax." *Proc. 1989 IEEE Workshop on Visual Languages*, Rome, Italy, 105-110, Oct. 1989.
- [17] Gonzalez, R. C. and Wintz, P., *Digital Image Processing*, Second Edition, Addison-Wesley, Reading, Massachusetts, 1987.
- [18] Gose, E., Johnsonbaugh, R. and Jost, S., *Pattern Recognition and Image Analysis*. Prentice-Hall, New Jersey, USA, 1996.
- [19] Gourlay, J. S., "A Language for Music Printing." *Communications of the ACM*, **29** (5), 388-401, 1986.
- [20] Govindan, V. and Shivaprasad, A., "Character Recognition – A Review." *Pattern Recognition*, **23**, 7, 671-683, 1990.
- [21] Hellman, M. E., "The Nearest Neighbour Classifier Rule with a Reject Option." *Proc. IEEE Transactions on Systems Science and Cybernetics*, **6** (3), 179-185, July 1970.
- [22] Jorge, J. A. and Fonseca, M. J., "A Simple Approach to Recognise Geometric Shapes Interactively." *Proc. Third IAPR Intl. Workshop on Graphics Recognition*, Jaipur, India, 251-258, 1999.
- [23] Kanungo, T., Haralick, R. M. and Dori, D., "Understanding Engineering Drawings: A Survey." *Proc. IAPR Workshop on Graphics Recognition*, Penn. State. U., USA, 119-130, Aug. 1995.
- [24] Lam, L., Lee, S. and Suen, C. Y., "Thinning Methodologies – A Comprehensive Survey." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **14**, 9, Sept. 1992.
- [25] Lank, E., Thorley, J. S. and Chen, S. J., "An Interactive System for Recognizing Hand Drawn UML Diagrams." *Proc. CASCON 2000*, Toronto, Canada, Sept. 2000.
- [26] Lewis, R., *Practical Digital Image Processing*. Ellis Horwood, New York, USA, 1990.

- [27] Lin, X., Shimotsuji, S., Minoh, M. and Sakai, T., "Efficient Diagram Understanding with Characteristic Pattern Detection." *Computer Vision, Graphics, and Image Processing*, **30**, 84-106, 1985.
- [28] Lu, Z., "Detection of Text Regions From Digital Engineering Drawings." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20** (4), 431-439, 1998.
- [29] Marriot, K., Meyer B. and Wittenburg, K. B., "A Survey of Visual Language Specification and Recognition." *Visual Language Theory*, K. Marriott and B. Meyer, Eds., Springer, 5-85, 1998.
- [30] Messmer, B. T. and Bunke, H., "Automatic Learning and Recognition of Graphical Symbols in Engineering Drawings." *Graphics Recognition: Methods and Applications. (Selected Papers from IAPR First Intl. Workshop on GREC, University Park, PA, USA) Lecture Notes in Computer Science*, Vol. 1072, Springer, 1-12, Aug. 1995.
- [31] Myers, B. A., "Taxonomies of Visual Programming and Program Visualization." *Journal of Visual Languages and Computing*, **1**, 97-123, 1990.
- [32] O'Gorman, L., "Basic Techniques and Symbol-Level Recognition - An Overview." *Graphics Recognition: Methods and Applications. (Selected Papers from IAPR First Intl. Workshop on GREC, University Park, PA, USA) LNCS*, Vol. 1072, Springer, 1-12, Aug. 1995.
- [33] Okazaki, A., Takashi K., Kazuhiro, M., Tsunekawa, S. and Kawamoto, E., "An Automatic Circuit Diagram Reader with Loop-Structure-Based Symbol Recognition." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **10** (3), 331-340, 1998.
- [34] O'Rourke, J., *Computational Geometry in C*. Cambridge University Press, New York, USA, 1994.
- [35] Pasternak, B. and Neumann, B., "Adaptable Drawing Interpretation Using Object-Oriented and Constraint-Based Graphic Specification." *Proc. Second International Conference on Diagram Analysis and Recognition*, Tsukuba, Japan, 359-364, Oct. 1993.
- [36] Pavlidis, T., *Graphics and Image Processing*. Computer Science Press, Maryland, USA, 1982.
- [37] Pratt, W. K., *Digital Image Processing*. John Wiley & Sons, New York, USA, 1991.
- [38] Ragnemalm, I., "Fast Edge Smoothing in Binary Images Using Euclidean Metric." *Progress in Image Analysis and Processing II. (Proc. 6th International Conference on*

- Image Analysis and Processing, Como, Italy, Sept. 1991) World Scientific, 83-90, Singapore, 1992.
- [39] Ramel, J., Boissier, G. and Emptoz, H., "A Structural Representation Adapted to Handwritten Symbol Recognition." *Proc. 3rd IAPR International Workshop on Graphics Recognition*, Jaipur, India, 259-266, 1999.
 - [40] Schurmann, J., *Pattern Classification: A Unified View of Statistical and Neural Approaches*. John Wiley & Sons, New York, USA, 1996.
 - [41] Steinherz, T., Rivlin, E. and Intrator, N., "Offline Cursive Script Word Recognition – A Survey," *Intl. Journal on Document Analysis and Recognition*, 2, 2/3, 90-110, 1999.
 - [42] Tombre, K., "Analysis of Engineering Drawing: State of the Art and Challenges." *Proc. Second IAPR Workshop on Graphics Recognition*, Nancy, France, 257-263, Aug. 1997.
 - [43] Tombre, K., "Structural and Syntactic Methods in Line Drawing Analysis: To which Extent do they Work?" *Advances in Structural and Syntactic Pattern Recognition*, (Proc. of Sixth SSPR '96, Leipzig, Germany), LNCS, Vol. 1121, Springer Verlag, 310-321, Aug. 1996.
 - [44] Tombre, K., Ah-Soon, C., Dosch, P., Masini, G. and Tabbone, S., "Stable and Robust Vectorization: How to Make The Right Choices." *Proc. Third IAPR International Workshop on Graphics Recognition*, Jaipur, India, 3-16, Sept. 1999.
 - [45] Toyama, K. and Hager, G. D., "If at First You Don't Succeed..." *Proc. Fourteenth National Conference on Artificial Intelligence and the Ninth Innovative Applications of Artificial Intelligence*, Menlo Park, Calif., AAAI Press, 3-9, 1997.
 - [46] Vaxiviere, P. and Tombre, K., "Knowledge Organization and Interpretation Process In Engineering Drawing Interpretation." *Proc. IAPR International Workshop on Document Analysis Systems*, Kaiserslautern, Germany, 313-321, Oct. 1994.
 - [47] VL Steering Committee (Chang, S., Korfhage, R., et al.), "Ten Years of Visual Languages Research." *Proc. IEEE Symposium on Visual Languages*, St. Louis, Missouri, 196-205, Oct. 1994.
 - [48] Winston, P., "Learning Structural Descriptions from Examples." *The Psychology of Computer Vision*, McGraw-Hill, New York, USA, 1975.
 - [49] Zanibbi, R., "Recognition of Mathematics Notation via Computer Using Baseline Structure." *Master's thesis, Queen's University*, Kingston, Ontario, Canada, Jan. 2000.

Appendix A: Detailed Results of Tests

This appendix shows the detailed results of tests performed.

Table 6-1: Confusion Matrix for Reject_Classifier, 0 negative training samples, mouse-drawn.

		RECOGNIZED							
		Circle	Rectangle	Diamond	Triangle	Over-segmented Negative	Unknown Negative	Under-segmented Negative	Reject
M O U S E D R A W N	Circle	88	0	12	0	0	0	0	0
	Rectangle	2	96	0	2	0	0	0	0
	Diamond	2	8	86	4	0	0	0	0
	Triangle	0	10	0	90	0	0	0	0
	Over-segmented Negative	4	48	16	30	0	0	0	2
	Unknown Negative	24	46	12	8	0	0	0	10
	Under-segmented Negative	10	54	24	4	0	0	0	8

All amounts are percentages

Table 6-2: Confusion Matrix for Reject_Classifier, 50 over-segmented negative training samples, mouse-drawn.

		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Over-segmented Negative	Reject
MOUSE DRAWN	Circle	86	0	10	0	4	0
	Rectangle	0	84	0	2	12	2
	Diamond	2	8	80	4	6	0
	Triangle	0	6	0	80	14	0
	Over-segmented Negative	2	14	10	22	52	0

Table 6-3: Confusion Matrix for Reject_Classifier, 100 over-segmented negative training samples, mouse-drawn.

		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Over-segmented Negative	Reject
MOUSE DRAWN	Circle	84	0	10	0	6	0
	Rectangle	0	80	0	2	16	2
	Diamond	2	8	76	4	10	0
	Triangle	0	4	0	66	30	0
	Over-segmented Negative	0	12	8	16	64	0

Table 6-4: Confusion Matrix for Reject_Classifier, 150 over-segmented negative training samples, mouse-drawn.

All amounts are percentages		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Over-segmented Negative	Reject
MOUSE DRAWN	Circle	84	0	10	0	6	0
	Rectangle	0	76	0	2	18	4
	Diamond	2	6	76	4	12	0
	Triangle	0	4	0	56	40	0
	Over-segmented Negative	0	10	6	16	68	0

Table 6-5: Confusion Matrix for Reject_Classifier, 200 over-segmented negative training samples, mouse-drawn.

All amounts are percentages		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Over-segmented Negative	Reject
MOUSE DRAWN	Circle	72	0	8	0	20	0
	Rectangle	0	74	0	2	18	6
	Diamond	2	6	70	4	18	0
	Triangle	0	2	0	56	42	0
	Over-segmented Negative	0	8	6	16	70	0

Table 6-6: Confusion Matrix for Reject_Classifier, 50 unknown negative training samples, mouse-drawn.

All amounts are percentages		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Unknown Negative	Reject
MOUSE DRAWN	Circle	72	0	10	0	18	0
	Rectangle	0	80	0	2	14	4
	Diamond	2	2	82	4	6	4
	Triangle	0	10	0	90	0	0
	Unknown Negative	18	28	6	8	38	2

Table 6-7: Confusion Matrix for Reject_Classifier, 100 unknown negative training samples, mouse-drawn.

All amounts are percentages		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Unknown Negative	Reject
MOUSE DRAWN	Circle	84	0	10	0	6	0
	Rectangle	0	80	0	2	16	2
	Diamond	2	8	76	4	10	0
	Triangle	0	4	0	66	30	0
	Unknown Negative	0	12	8	16	64	0

Table 6-8: Confusion Matrix for Reject_Classifier, 150 unknown negative training samples, mouse-drawn.

All amounts are percentages		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Unknown Negative	Reject
MOUSE DRAWN	Circle	50	0	10	0	40	0
	Rectangle	0	72	0	2	22	4
	Diamond	2	2	78	4	12	2
	Triangle	0	10	0	88	2	0
	Unknown Negative	12	14	2	2	66	4

Table 6-9: Confusion Matrix for Reject_Classifier, 200 unknown negative training samples, mouse-drawn.

All amounts are percentages		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Unknown Negative	Reject
MOUSE DRAWN	Circle	42	0	10	0	48	0
	Rectangle	0	66	0	0	30	4
	Diamond	2	0	68	4	14	2
	Triangle	0	8	0	88	2	2
	Unknown Negative	10	12	0	2	74	2

Table 6-10: Confusion Matrix for Reject_Classifier, 50 under-segmented negative training samples, mouse-drawn.

		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Under-segmented Negative	Reject
MOUSE DRAWN	Circle	82	0	4	0	14	0
	Rectangle	0	90	0	2	8	0
	Diamond	0	2	76	4	18	0
	Triangle	0	8	0	88	4	0
	Under- segmented Negative	0	24	12	4	32	0

Table 6-11: Confusion Matrix for Reject_Classifier, 100 under-segmented negative training samples, mouse-drawn.

		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Under-segmented Negative	Reject
MOUSE DRAWN	Circle	82	0	4	0	14	0
	Rectangle	0	82	0	2	16	0
	Diamond	0	2	76	4	18	0
	Triangle	0	8	0	86	6	0
	Under- segmented Negative	0	26	16	2	52	4

Table 6-12: Confusion Matrix for Reject_Classifier, 150 under-segmented negative training samples, mouse-drawn.

All amounts are percentages		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Under-segmented Negative	Reject
MOUSE DRAWN	Circle	82	0	4	0	14	0
	Rectangle	0	76	0	2	22	0
	Diamond	0	2	78	4	26	0
	Triangle	0	8	0	82	10	0
	Under- segmented Negative	0	22	12	0	62	4

Table 6-13: Confusion Matrix for Reject_Classifier, 200 under-segmented negative training samples, mouse-drawn.

All amounts are percentages		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Under-segmented Negative	Reject
MOUSE DRAWN	Circle	80	0	2	0	18	0
	Rectangle	0	70	0	0	30	0
	Diamond	0	2	62	4	32	0
	Triangle	0	8	0	80	12	0
	Under- segmented Negative	0	14	10	0	72	4

Table 6-14: Confusion Matrix for Reject_Classifier, 0 negative training samples, pen-and-paper drawn.

All amounts are percentages		RECOGNIZED							
		Circle	Rectangle	Diamond	Triangle	Over-segmented Negative	Unknown Negative	Under-segmented Negative	Reject
P E N D R A W N	Circle	86	4	10	0	0	0	0	0
	Rectangle	0	88	10	2	0	0	0	0
	Diamond	14	14	70	2	0	0	0	0
	Triangle	0	8	2	86	0	0	0	4
	Over-segmented Negative	6	46	18	26	0	0	0	4
	Unknown Negative	50	34	8	8	0	0	0	0
	Under-segmented Negative	8	44	32	16	0	0	0	0

Table 6-15: Confusion Matrix for Reject_Classifier, 50 over-segmented negative training samples, pen-and-paper drawn.

All amounts are percentages		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Over-segmented Negative	Reject
P E N D R A W N	Circle	84	4	10	0	2	0
	Rectangle	0	84	6	2	8	0
	Diamond	12	12	66	2	6	2
	Triangle	0	6	2	78	10	4
	Over-segmented Negative	4	18	14	26	28	10

Table 6-16: Confusion Matrix for Reject_Classifier, 100 over-segmented negative training samples, pen-and-paper drawn.

All amounts are percentages		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Over-segmented Negative	Reject
PEN DRAWN	Circle	84	4	6	0	6	0
	Rectangle	0	76	4	2	16	2
	Diamond	12	10	60	2	16	0
	Triangle	0	2	2	76	20	0
	Over-segmented Negative	4	2	10	8	30	6

Table 6-17: Confusion Matrix for Reject_Classifier, 150 over-segmented negative training samples, pen-and-paper drawn.

All amounts are percentages		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Over-segmented Negative	Reject
PEN DRAWN	Circle	82	2	6	0	10	0
	Rectangle	0	74	6	0	16	4
	Diamond	8	10	60	0	22	0
	Triangle	0	2	2	70	26	0
	Over-segmented Negative	4	10	12	20	44	10

Table 6-18: Confusion Matrix for Reject_Classifier, 200 over-segmented negative training samples, pen-and-paper drawn.

		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Over-segmented Negative	Reject
PEN DRAWN	Circle	82	10	6	0	10	0
	Rectangle	0	72	6	0	18	4
	Diamond	8	10	56	0	26	0
	Triangle	0	2	2	68	28	0
	Over-segmented Negative	4	10	10	14	54	8

Table 6-19: Confusion Matrix for Reject_Classifier, 50 unknown negative training samples, pen-and-paper drawn.

		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Unknown Negative	Reject
PEN DRAWN	Circle	70	4	10	0	16	0
	Rectangle	0	84	8	0	6	2
	Diamond	14	12	64	2	8	0
	Triangle	0	8	2	86	0	4
	Unknown Negative	14	26	6	8	44	2

Table 6-20: Confusion Matrix for Reject_Classifier, 100 unknown negative training samples, pen-and-paper drawn.

All amounts are percentages		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Unknown Negative	Reject
PEN DRAWN	Circle	62	2	10	0	26	0
	Rectangle	0	84	4	0	10	2
	Diamond	12	8	58	2	20	0
	Triangle	0	8	2	86	0	4
	Unknown Negative	8	14	6	6	61	4

Table 6-21: Confusion Matrix for Reject_Classifier, 150 unknown negative training samples, pen-and-paper drawn.

All amounts are percentages		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Unknown Negative	Reject
PEN DRAWN	Circle	62	10	10	0	26	0
	Rectangle	0	4	4	0	12	2
	Diamond	10	58	58	2	22	0
	Triangle	0	2	2	86	2	2
	Unknown Negative	8	6	6	4	70	0

Table 6-22: Confusion Matrix for Reject_Classifier, 200 unknown negative training samples, pen-and-paper drawn.

All amounts are percentages		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Unknown Negative	Reject
PEN DRAWN	Circle	60	2	6	0	32	0
	Rectangle	0	80	0	0	18	2
	Diamond	6	8	50	2	34	0
	Triangle	0	4	2	86	6	2
	Unknown Negative	6	8	6	4	76	0

Table 6-23: Confusion Matrix for Reject_Classifier, 50 under-segmented negative training samples, pen-and-paper drawn.

All amounts are percentages		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Under-segmented Negative	Reject
PEN DRAWN	Circle	84	4	10	0	2	0
	Rectangle	0	74	8	2	12	4
	Diamond	12	10	62	2	14	0
	Triangle	0	6	0	86	2	6
	Under- segmented Negative	4	16	18	12	46	4

Table 6-24: Confusion Matrix for Reject_Classifier, 100 under-segmented negative training samples, pen-and-paper drawn.

		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Under-segmented Negative	Reject
PEN DRAWN	Circle	82	0	8	0	10	0
	Rectangle	0	72	8	0	16	4
	Diamond	10	10	56	2	22	0
	Triangle	0	6	0	84	4	6
	Under-segmented Negative	2	4	12	10	64	8

All amounts are percentages

Table 6-25: Confusion Matrix for Reject_Classifier, 150 under-segmented negative training samples, pen-and-paper drawn.

		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Under-segmented Negative	Reject
PEN DRAWN	Circle	82	0	8	0	10	0
	Rectangle	0	64	6	0	26	4
	Diamond	8	8	44	2	38	0
	Triangle	0	6	0	82	6	6
	Under-segmented Negative	2	2	10	8	74	4

All amounts are percentages

Table 6-26: Confusion Matrix for Reject_Classifier, 200 under-segmented negative training samples, pen-and-paper drawn.

		RECOGNIZED					
		Circle	Rectangle	Diamond	Triangle	Under-segmented Negative	Reject
PEN DRAWN	Circle	82	0	8	0	10	0
	Rectangle	0	58	4	0	39	2
	Diamond	6	6	40	2	36	0
	Triangle	0	6	0	82	8	4
	Under- segmented Negative	2	0	8	6	80	4

All amounts are percentages

Appendix B: Mouse-Drawn Positive Training Samples

Mouse-Drawn Circles

Circle03.jpg

Circle04.jpg

Circle10.jpg

Mouse-Drawn Rectangles

Rectangle04.jpg

Rectangle05.jpg

Rectangle06.jpg

Rectangle07.jpg

Rectangle08.jpg

Rectangle09.jpg

Mouse-Drawn Diamonds

Diamond04.jpg

Diamond05.jpg

Diamond06.jpg

Diamond07.jpg

Diamond08.jpg

Diamond09.jpg

Diamond10.jpg

Mouse-Drawn Triangles

Triangle03.jpg

Triangle04.jpg

Triangle05.jpg

Triangle06.jpg

Triangle07.jpg

Triangle08.jpg

Triangle10.jpg

Appendix C: Pen-and-Paper Drawn Positive Training

Samples

Pen-and-Paper Drawn Circles

Circle03.jpg

Circle04.jpg

Circle05.jpg

Circle06.jpg

Pen-and-Paper Drawn Rectangles

Rectangle05.jpg

Rectangle06.jpg

Rectangle07.jpg

Rectangle08.jpg

Rectangle09.jpg

Rectangle10.jpg

Pen-and-Paper Drawn Diamonds

Diamond03.jpg

Diamond04.jpg

Diamond05.jpg

Diamond06.jpg

Diamond07.jpg

Diamond08.jpg

Diamond09.jpg

Diamond10.jpg

Pen-and-Paper Drawn Triangles

<p>Triangle03.jpg</p> <p>This panel contains five hand-drawn triangles. From top to bottom: a tall, narrow triangle pointing right; a medium triangle pointing right; a small triangle pointing right; a very small triangle pointing right; and a medium triangle pointing right.</p>	<p>Triangle04.jpg</p> <p>This panel contains five hand-drawn triangles. From top to bottom: a medium triangle pointing left; a small triangle pointing left; a medium triangle pointing right; a medium triangle pointing right; and a medium triangle pointing right.</p>
<p>Triangle05.jpg</p> <p>This panel contains five hand-drawn triangles. From top to bottom: a medium triangle pointing right; a large triangle pointing right; a very small triangle pointing right; a medium triangle pointing right; and a medium triangle pointing right.</p>	<p>Triangle06.jpg</p> <p>This panel contains five hand-drawn triangles. From top to bottom: a medium triangle pointing right; a small triangle pointing right; a small triangle pointing left; a medium triangle pointing right; and a large triangle pointing right.</p>

Triangle07.jpg

Triangle08.jpg

Triangle09.jpg

Triangle10.jpg

Appendix D: Mouse-Drawn Negative Training Samples

Mouse-Drawn OVER-SEGMENTED Samples

Mouse-Drawn UNKNOWN Samples

Mouse-Drawn UNDER-SEGMENTED Samples

Appendix E: Pen-and-Paper Drawn Negative Training

Samples

Pen-and-Paper Drawn OVER-SEGMENTED

Samples

Pen-and-Paper Drawn UNKNOWN Samples

Pen-and-Paper Drawn UNDER-SEGMENTED Samples

Appendix F: Illustrations of Visual Languages and Diagrams

Diagrams

Figures Q - U scanned from [11].

Figure R: A utility map.

Figure S: An electronic circuit.

Figure T: Part of a scaled topographic map.

Figure U: Music Notation

Figure V: Various Types of Logos

Figure W: UML Diagram (taken from [6]).

Glossary of Terms

3	2	1
4	P	0
5	6	7

Figure X: Pixel Neighbour Identification

4-Neighbour. Refer to Figure X. The 4-neighbours of the pixel P are those in positions 0, 2, 4 and 6.

8-Neighbour. Refer to Figure X. The 8-neighbours of the pixel P are those in any of the positions marked 0,1,2...7.

Area of an object. The total number of pixels enclosed or used by an object. This includes the border pixels and any enclosed pixels.

Aspect Ratio. The length-to-width ratio of an object.

Binarization. The process of transforming a grayscale or color image into black and white pixels (see also “thresholding”).

Bounding Box. The smallest box enclosing an object and parallel to the principal axes.

Chain-Code. A code that represents the chain formed by a series of connected pixels.

Details of this method can be found in [37].

Circularity Measure. P^2/A (perimeter squared, divided by the area). This is equal to 4π for perfect circles, 16 for squares and numbers larger than 4π for other shapes.

Convex Hull (informal definition). In two dimensions, the convex hull of a set of points is the shape taken by a rubber band when it is stretched around nails pounded into the two-dimensional plane at each point [34].

Connected-Component Labeling. The process of labeling with unique labels the objects (or connected regions) in an image.

Correctness. The correctness measure is the percentage of test patterns that are classified correctly. It is the inverse value of the error rate.

Curve Fitting. A technique for data reduction. Experimental data points are approximated by a curve which is an analytic function. The parameters needed to calculate the curve are used in subsequent processing, rather than the data points themselves [26].

Diagram. A sentence in a visual language. Its meaning can be derived from the spatial configuration of its parts as well as from the individual meanings associated with each part/symbol [29].

Dilation. A morphological operation during which an object is increased in size by the addition of pixels around its boundary [26].

Electronic Whiteboard. An electronic whiteboard is an online interface for data entry or diagram drawing. It provides the co-ordinates of points along the lines being drawn, as well as temporal information about the strokes. From this data, the raster scan of the final image on the whiteboard can be constructed if needed.

Erosion. A morphological operation during which an object is decreased in size by the removal of pixels from its boundary [26].

Euclidean Distance Measure. The distance between two points $a = (a_1, \dots, a_n)$ and $b = (b_1, \dots, b_n)$, defined by

$$d_e(a,b) = \sqrt{\sum_{i=1}^n |b_i - a_i|^2}$$

where n is the number of features.

This is used as a measure of dissimilarity between feature vectors.

Error Rate. The error rate is a very common measure of classifier performance. It is the percentage of test patterns that are assigned to the wrong category or class [15]. It is common to seek minimum-error-rate classification for recognition systems .

Feature. This is a measurement made on an object for use in recognition.

Feature Vector. A set of features measured from the same object.

0	0	0
0	1	0
0	0	0

Figure Y: 3-by-3 filter

Filter. A filter is a window through which pixels may be examined. A pixel and its surrounding region may or may not have good correlation with the filter. For example: Figure Y is a filter that will show greatest response to a black pixel surrounded by white pixels.

Hole. The white space enclosed by a connected series of black points forming a loop.

Invariant Features. Features that when measured under different points of view give the same value. A feature that is invariant under affine transformations, gives approximately the same value when the object is rotated, translated or scaled.

Kernel. A recognition kernel is a software library or collection that makes common recognition algorithms available for use. It can be used by a variety of recognition systems in different domains.

Largest Empty Triangle. A triangle that is found within a set of points, and is the largest triangle (in terms of area) within this set of points.

Lexical Phase. This phase encompasses a number of low-level image processing steps such as noise-reduction, binarization, etc. It is also the recognition phase where the basic constituents of drawings are recognized [23].

Medial Axis. A connected set of lines joining locations which are local maxima in terms of their distance from the object's boundary [26].

Miss-segmented Object. A miss-segmented symbol can either be overly segmented (see *Over-segmented Object* entry) or under segmented (see *Under-segmented Object* entry). Both errors cause classification errors, because the miss-segmented object will not have the same features as the well-segmented object.

Nearest Neighbour Classifier. A classifier that uses a distance measure to find the nearest neighbour(s) to a point in feature space representing a test sample's feature vector. The classifier labels the test point with the same class as the nearest-neighbour, or allows the nearest-neighbours to vote on the classification of the test point.

Negative Samples. Miss-segmented positive symbols and unknown symbols constitute negative samples. (Also referred to as “garbage” in [40]).

Noise. Random variation in the pixel values in an image caused by the scanning/ acquisition process rather than by the scene itself [26]. Examples include: salt-and-pepper noise which looks like small black points on white areas, or vice-versa. Also defined as the additive (or multiplicative) contamination of an image [7].

Object. An object is a two dimensional connected region of the image which has been identified by some previous segmentation step.

Offline. Offline recognition and offline classifiers deal with data that has been gathered offline (ie. drawn by hand on paper or non-electronic interface). This method results in image data as input.

Online. Online recognition and online classifiers deal with data that was gathered in an online electronic environment, usually one that is connected directly to a computer. (Eg. The *electronic whiteboard*) This method results in time-stamped strokes as input.

Over-segmented Object. An over-segmented object is one that has been cut into parts. This over-segmentation causes a single object to be interpreted as two or more symbols. For example, a capital T may be over segmented into two line segments, the top

horizontal line and the bottom vertical line. This may lead a text classifier to interpret the bottom object as a capital I.

Perimeter. The perimeter of an object is obtained by finding the sum of the lengths of the line segments between the centers of a closed-loop of successive black border pixels that are 4-neighbours of the white background [18]. Example: the perimeter of an object made up of two horizontally aligned black pixels is 2, while the perimeter of a single pixel is 0.

Polygonal Approximation. Polygonal approximation is a method for reducing the effects of noise on line segments. Straight lines are fitted to sets of points in the original image. This achieves a smoothing effect on the lines if done properly. (See also “curve fitting”)

Positive Samples. Symbols that are well segmented, and meant to be recognized by the classifier. (In this thesis, positive samples can be circles, diamonds, triangles or rectangles.)

Pre-processing Phase. This phase represents a number of image processing steps which include binarization and thinning. It is the phase that prepares the image for further processing (example: by cleaning up noise in the image) and is included in the Lexical Phase.

Raster Scan. A raster scan is an encoding of an image into a one-dimensional array of integers by reading the pixels from left to right, top to bottom. So `raster_scan[0][0]` is the topmost, leftmost pixel in the image.

Rectangularity Measure. The ratio of the area of the bounding box to the area of the convex hull of the object.

Reject Option. A method or rule used by a classifier in order to reject samples that have a high probability of being misclassified.

Reliability. The reliability of a classifier is determined by how well the classifier is able to classify correctly positive symbols and how well it is able to reject symbols that have a high probability of being misclassified. Formal definition is in section 3.3.

Segmentation. The division of an image into regions corresponding to objects or parts of objects [26].

Semantics. With respect to visual languages, semantics are the rules governing the usage and meaning of various symbols and symbol arrangements.

Skeleton. See medial-axis definition.

Syntax. With respect to visual languages, syntax is the rules that govern the valid placement of symbols in space.

Thinning. Thinning is one technique for retrieving the medial axis of an object, based on removing pixels from the object boundary [26].

Threshold. Refers to a value that can be manually or automatically set. This value acts as a threshold or barrier. Example:

If $x < \text{threshold}$ then { do this };

Else { do that };

Thresholding. Includes binarization. Method to distinguish the foreground from the background. When using a global threshold rule for image binarization, one assigns all pixels at or above the threshold to the color white, and the rest to the color black.

Adaptive thresholding is a method that adapts the threshold with respect to the position in the image.

Triangularity Measure. The ratio of the area of the convex hull of an object to the area of the largest empty triangle enclosed by the convex hull. This number approaches unity if the shape is triangular.

Triangularization. The process of segmenting an object into non-intersecting triangles.

UML. The Unified Modeling Language is a visual language developed by Booch, Rumbaugh and Jacobson to be used in modeling of software systems and processes. The Unified Modeling Language User Guide [6] is the definitive guide to UML.

Under-segmented Object. An under-segmented object is one that has not been separated from connecting lines from another separate object. The new fused shape is not actually a recognizable object but is processed as one object, thus causing an error in classification. An example of this is in segmenting of cursive script. Sometimes two letters written closely together cannot be separated by the segmenter and are therefore seen as one object.

Unknown Object. An unknown object is an object or symbol that the classifier was not trained to recognize and therefore should reject due to inexperience or lack of expertise.

Vectorization. The process of representing an object or an image as a series of lines or vectors.

Visual Language. A language that consists of symbols that can be arranged in two or three dimensions to form valid sentences of that language [29].

Vita

Name: Hanaa Barakat

Place and Year of birth: Abu-Dhabi, U.A.E., 1976

Education: Bachelor of Science,
Honours, Computer Science,
Ryerson Polytechnic University, 1994-1999

Master of Science,
Computing & Information Science,
Queen's University, 1999-2001

Experience: Access Services Analyst
Imperial Oil Ltd., 1996

Co-op Year 2000 Programmer
MIS, SunLife Canada Ltd., 1997-1998

Teaching Assistant
Computing & Information Science
Queens' University, 1999-2000,
Computer Science
Ryerson Polytechnic Univeristy, 1999

Research Assistant,
Computing & Information Science
Queens' University, 1999-2001,
Computer Science
Ryerson Polytechnic University, 1999

Awards: CITO's Research Forum Presentation Award,
2000

Queen's University Graduate Student Award,
1999

Dean's List, 1999

Poster Publications: Barakat, H. and Blostein, D., "Training with
Positive and Negative Data Samples: Effects on a
Classifier for Hand-Drawn Geometric Shapes."
To appear in ICDAR 2001, Seattle, USA. Sept.
2001.