FINAL REPORT: NASA-87
GRANT # NAG 1-412/JFR

FUZZY AUTOMATA AND PATTERN MATCHING

PRINCIPAL INVESTIGATORS: C. B. SETZER
N. A. WARST

MATHEMATICAL AND COMPUTER SCIENCES
ATLANTA UNIVERSITY, ATLANTA, GEORGIA 30314

OCTOBER 1, 1983 - SEPTEMBER 30, 1986

NASA LANGLEY RESEARCH CENTER
TECHNICAL MONITOR: MICHAEL GOODE

RC AC6/MF AC1 CSCL 09B G3/61 0085446 Unclas

N87-27429
FINAL REPORT: NASA-87
GRANT # NAG 1-412/JFR

FUZZY AUTOMATA AND PATTERN MATCHING

PRINCIPAL INVESTIGATORS: C. B. SETZER
N. A. WARST

MATHEMATICAL AND COMPUTER SCIENCES
ATLANTA UNIVERSITY, ATLANTA, GEORGIA 30314
OCTOBER 1, 1983 - SEPTEMBER 30, 1986

NASA Langley Research Center
Technical Monitor: Michael Goode
# TABLE OF CONTENTS

## A. SUMMARY OF WORK

| a. A Survey of Pattern Recognition: An Annotated Bibliography | 1 |
| b. Edge Detection and Image Processing Using a Gradient and Edge Following Method | 1 |
| c. Minimal Finite Automata From Finite Training Sets | 2 |

## B. APPENDICES (Details)

| I. A SURVEY OF PATTERN RECOGNITION: AN ANNOTATED BIBLIOGRAPHY | 5 |
| II. EDGE DETECTION AND IMAGE PROCESSING USING A GRADIENT AND EDGE FOLLOWING METHOD | 50 |
| III. MINIMAL FINITE AUTOMATA FROM FINITE TRAINING SETS | 108 |
FINAL REPORT

1. Project Period:
   October 1, 1983 to September 30, 1986

2. Personnel:
   Principal Investigators:
   C. Bennet Setzer (Oct. 1, 1983 to Sept. 30, 1985)
   Graduate Students:
   I. O'Nour, P. Sparrow, F. Brown, Linda Adams and E. Placide
   The first three have finished their graduate theses and the last two are in the process of doing so.

3. Summary of Completed Work
   This report contains the work accomplished through three main project. Two more projects started during this period will finish shortly. The corresponding report will be included in that of the second project.
   (a) A Survey of Pattern Recognition: An Annotated Bibliography (See Appendix I)
      The work consisted of a wide-ranging search for articles and books concerned with fuzzy automata and syntactic pattern recognition. Also, a number of survey articles on image processing and feature detection have been collected. Although the work is basically survey, it lays down the foundation of further work through its annotated bibliography. The complete work is included as Appendix I.
   (b) Edge Detection and Image Processing Using a Gradient and Edge Following Method (See Appendix II)
Hough's algorithm illustrates one way in which knowledge about the image can be used to interpret the details of the image. Since the used syntactic methods are more local, the performance of standard local techniques are studied. To this end, an edge following algorithm based on comparison of the gradients at adjacent pixels is implemented. In general, the closest match is considered an extension of an edge element. It was found that in hand generated pictures, the algorithm worked well on following straight lines, but had great difficulty turning corners. The connectivity of polygons was not evident in the resulting global edge trace. This experimental work although poor in performance has helped other related efforts. If an edge is being followed stepwise, some history would be useful in picking a continuation. The goal would be to be able to continue across apparent gaps in edges, given that the algorithm was already following a "strong edge". For details see Appendix II.

(c) Minimal Finite Automata From Finite Training Sets

The basic result of this research is an algorithm which produces a minimal finite automaton recognizing a given finite set of strings. Minimality is in the number of states and compared among those automata that recognizes the given string and no other up to a given length. One difficulty of the construction is that, in some cases, this minimal automaton is not unique for a given set of strings and a given maximum length. Numerous examples show that it correctly deduces the "correct" automaton for sets of strings drawn from a given regular set. Further,
these examples show that non-unicity of the minimal automaton disappears for large samples. However, the convergence of the algorithm in the limit has not been demonstrated.

This algorithm compares favorably with other inference algorithms. It seems to produce smaller automata than other methods (e.g. [1]). It produces a deterministic automaton directly comparing with [1]. More importantly, the algorithm produces an automaton with a rigorously described relationship to the original set of strings that does not depend on the algorithm itself. In general, the study of abstract objects is more tractable if they have invariant relationships among themselves. See Appendix III for details).

APPENDIX 1. A Survey of Pattern Recognition: An Annotated Bibliography
CHAPTER ONE
THE STATISTICAL APPROACH TO PR

1.1. INTRODUCTION

In a paper on pattern recognition, one would like to make the first statement: a definition of pattern recognition.

"Mathematically, pattern recognition is a classification problem. We wish to design a system such that a handwritten symbol will be recognized as an 'A', a 'B' etc. The handwritten characters are often ambiguous, and there will be misclassified characters. The major goal in designing a pattern recognition machine is to have a low probability of misclassification." [Young 1974].

"Pattern recognition may be regarded as a process in which classes of patterns are directly mapped on a set of discrete elements." [Mohanan 1988].

Neither I nor any of my colleagues have found a succinct definition of pattern recognition (PR) which does justice to our personal understanding. PR means different things to each of us." [Batchelor 1979].

Batchelor states the purpose of PR research as follows:

"The purpose of PR research is to build machines that will perform tasks which are essentially 'human'. PR is motivated partially by our desire to free human beings from tedious, boring, repetitive, or dangerous work."

The problem of pattern recognition usually denotes classification and/or description of a set of processes or events. The set of processes or events to be classified could be a set of physical objects or a set of more abstract ones such as mental states. The processes or events with some similar properties are grouped into a class. The total number of pattern classes in a particular problem is often determined by the particular application in mind. For example, consider the problem of English character recognition: we should have a
problem of 24 classes. On the other hand, if we are interested
in discriminating English characters from Arabic characters, we
have only a two-class problem. In some problems, the exact
number of classes may not be known initially, and it may have to
be determined from the observation of many representative
classes. In this case, we would like to detect the possibility
of having new classes of patterns as we observe more and more
patterns. Human beings perform the task of pattern recognition
in almost every instant of their daily lives.

We could also think of a pattern according to one of the
dictionary definitions: as a model, guide, or plan used in making
things. This can be a physical entity or an abstraction. Thus
almost anything which is within the reach of our senses can be
chosen as a pattern-a character, a photograph, a speech sample,
odors, tastes, an idea, etc.

A pattern class is a group of patterns with certain
properties. In real patterns, the pattern classes may be various
rock types: acidic, basic, volcanic, sedimentary, etc.
The problem of pattern recognition is that of classifying a
pattern into one of the pattern classes on the basis of certain
measurements or observations (subjectively).

Conceptually, the simplest form of recognition is probably
"template matching". The sentence describing an input pattern is
matched against sentences representing each prototype or
reference pattern. [Fu'82]. Based on a selected "matching" or
"similarity" criterion, the input pattern is classified in the
same class as the prototype pattern which is the "best" to match

6
the input. The structural information is not recovered. If a complete pattern description is required for recognition, a parsing or syntax analysis is necessary. Recently, the use of discriminant grammars has been proposed for the classification of syntactic patterns.

Surveying the many different mathematical techniques used to solve pattern recognition problems, one can group them into two general approaches. These are the statistical or decision-theoretic (sometimes called discriminant) approach and the syntactic (or structural) approach.

Let us consider an m-class pattern recognition problem. If each pattern is considered as a single entity, then a set of n characteristic measurements (features) can be used to represent each pattern under study. In such a case, each pattern is represented by an n-dimensional feature vector and the recognition of patterns can be accomplished by applying various techniques in discriminant analysis and statistical decision theory. [Fu '82]. Such an approach is often called decision-theoretic (or statistical) approach.

On the other hand, if the patterns are very complex or if the number of pattern classes m is very large (in fingerprint identification or scene analysis), the number of features n required for recognition could also become very large. In such problems, the statistical approach often becomes ineffective or computationally infeasible in providing a solution. To overcome this difficulty, one could use syntactic methods or a hybrid of syntactic/semantic combination.
1.0 STATISTICAL PATTERN RECOGNITION

In the decision-theoretic approach, the classification is based on a set of selected features extracted from the input pattern. These features' measurements are supposed to be invariant or less sensitive with respect to the commonly encountered variations and distortions and also contain less redundancies. The first subproblem is what measurements should be taken from the input patterns. Usually, the decision of what to measure is rather subjective and also dependent on the practical situations (e.g., the availability of measurements, the cost of measurements, etc.). The criterion of feature selection or ordering is often based on either the importance of the features in characterizing the patterns or the contribution of the features to the performance of recognition (i.e., the accuracy of recognition).

The second subproblem in pattern recognition is the problem of classification (or making a decision on the class assignment to the input patterns) based on the measurements taken from the selected features. The template-matching approach may be interpreted as a special case of the decision-theoretic approach, where the templates are stored in terms of feature measurements and a special classification criterion (matching) is used for the classifier.

Statistical pattern recognition is one of the mathematical theories underlying the design and analysis of recognition machines. By properly incorporating an input device (e.g., a scanner) in the digital, analog, or hybrid computer system, a statistical recognition machine is formed. In addition to the
characters and biomedical waveforms, statistical recognition methods have been successfully applied to patterns in weather prediction, photo-interpretation, multispectral crop classification, and many other areas. [Chen'73].

1.3 DISCRIMINANT FUNCTIONS AND FORMULATION OF PATTERNS

Let us consider patterns which are represented as vectors in the multidimensional Euclidean space \( R^d \). One of the most fundamental approaches in pattern recognition is the classification of patterns by means of discriminant functions [Kohonen'94]. Assume that the representation vectors are grouped into a finite number of clusters (each of which corresponds to a particular class). How can one define (mathematically) the equations of those hypersurfaces which optimally separate all clusters from one another? The simplest alternative is to try the linear hyperplane (the lowest degree surface). If the linear separation fails, one may take polynomial forms of successively higher degree. One optimality criterion could be the distance of all points from the separating surfaces. One frequently used criterion is the sum of squares of Euclidean distances.

Pattern classification is a process that is related to decision making, detection theory, etc., and may be discussed in these settings [Kohonen'94].
1.4 REPRESENTATION OF PATTERNS:

We must distinguish between a pattern (e.g., a visual image) and its representation within a computer — a process during which a lot of information was destroyed (because pattern description causes destruction of information).

If the input to a pattern recognition machine is a set of \( N \) measurements (e.g., an \( N \)-dimensional vector), the output is the classification \( C \). [Young'74]. Since \( C \) depends on the input vector \( X \), the classification is written as:

\[ C = d(X) \]

where \( d(X) \) is the decision function.

The selection of measurements is very important in the design of these machines. Because the values of \( X \) are determined by the measurements, the measurement selection defines the pattern space \( S_X \).

The pattern recognition machine may be divided into two parts, a feature extractor and a classifier. The feature extractor reduces the dimensionality of the input vectors to the classifier. Hence feature extraction is a transformation

\[ Y = T(X) \]

which transforms a pattern vector \( X \) in the pattern space \( S_X \) into a feature vector \( Y \) in a feature space \( S_Y \). The classifier then classifies \( X \) based on \( Y \). Because \( S_Y \) is of lower dimensionality than \( S_X \), the transformation is singular and some information is lost. The feature extraction should reduce the dimensionality but at the same time maintain a high machine performance.
A special case of feature extraction is feature selection. It selects a subset of the given measurements.

1.5 Feature Extraction

"Feature selection (or extraction) is generally considered a process of mapping the original measurements into more effective features. If the mapping is linear, the mapping function is well-defined and our task is simply to find the coefficients of the linear function so as to maximize or minimize the criterion. Therefore, if we have the proper criterion for evaluating the effectiveness of features, we can use the well-developed techniques of linear algebra for simple criteria; or, in the case of a complex criterion, we can apply optimization techniques to determine these mapping coefficients. Unfortunately, in many applications of pattern recognition, there are important features which are not linear functions of original measurements, but are highly nonlinear functions. Then, the basic problem is finding a proper nonlinear mapping function for the given data. Since we do not have any general theory to generate mapping functions systematically and to find the optimum one, the selection of features becomes very much a problem oriented." Fukunaga 1972.

One process frequently used in data reduction is feature extraction. Patterns (ordered sets of values) in practice may have dimensionalities that are intolerably high. Therefore it is common to preprocess these values, by forming various functionals over selected subsets of the pattern elements [Kohonen'84]. Such functionals are then named features, and in many cases they contain important correlations or other relationships which comprise the intrinsic information. The feature may be chosen statistically or heuristically.

Heuristics derive its name from the famous exclamation of Archimedes: "Eureka" - 'I have found'. It studies creative activity [Peckelis'74].

Heuristics, according to Judea Pearl, 1984, are criteria.
methods, or principles for deciding which among several alternative courses of action promises to be the most effective in order to achieve some goal. They represent compromises between two requirements: the need to make such criteria simple and, at the same time, the desire to see them discriminate correctly between good and bad choices.

In a pattern recognition problem, a number of pattern features can be extracted from a pattern, but only a few of them are often necessary to the recognition. To determine what variables (or attributes) are to be measured is the most important step towards designing an efficient recognition machine [Chen '77]. The task of determining the variables is called feature selection or extraction. Feature selection is usually preceded by preprocessing (data compaction) of patterns, which refers to the processes of sorting, normalization, filtering, etc. of the input patterns. The desirable requirements of the features selected are:

1. they properly describe the pattern,

2. they are easy to process, and

3. they are invariant to translation and rotation of the pattern.

The selectivity in the choice and the use of pattern features seems to belong to the fundamentals of human recognition. [Pik]. In the statistical approach to pattern recognition, this natural procedure results in sequential methods.
CLUSTERING

Another approach for the reduction of data to be stored in a recognition is to perform a classification of primary data (e.g., pattern vectors) into representative subsets. These subsets could be structurally related, ignoring the individual data points.

There are many instances where classification must and can be performed without a priori knowledge [Fukunaga 1972]. The clustering problem is not well defined unless the resulting classes of samples are required to exhibit certain properties. The choice of properties (or the definition of a cluster) is the fundamental issue in the clustering problem. Given a suitable definition of a cluster, it is possible to distinguish between good and bad classifications of samples.

The clustering problem may be set up [Kohonenu ‘84], as follows: assume that

\[ A = \{ a_i : i = 1, 2, ..., n \} \]

is a finite set of representations of items. This set has to be partitioned into disjoint subsets

\[ A_j, j = 1, 2, ..., k \]

such that with respect to this division, some functional describing the 'distance' between items attains an extremum value. This functional ought to describe the quality of groupings in the sense that the mutual distance between all \( a_i \) belonging to the same \( A_j \) are as small as possible while the distances between different \( A_j \) are large. The functional describing the grouping may contain, e.g., sums of some powers of the distances.
CHAPTER TWO

THE SYNTACTIC APPROACH TO PR

II SYNTACTIC PATTERN RECOGNITION:

"In some recognition problems, the structural information that describes each pattern is important, and the recognition process includes not only the capability of assigning the pattern to a particular class (to classify it), but also the capacity to describe aspects of the pattern that make it ineligible for assigning to another class." [Fu'92].

"The syntactic methods begin with the extraction of primitives from the patterns. With regard to line drawings, the primitives can be line segments defined, e.g., by regions of certain curvature, or segments between distinct points such as intersections or branchings. In principle, any subpattern in the original image, when suitably segmented and labelled, could also be defined as a primitive." [Kohonen'94].

"The study of syntactic pattern recognition has intensified during the last few years. In order that more experience shall be gained with the approach it is important that it can be applied by researchers whose principal interest is the application rather than the syntactic pattern recognition method. Thus there is a need for a suite of programs that enables a researcher skilled in neither language theory nor computing to make progress with the method." [Blake'91].

A typical example of the class of recognition mentioned above is scene analysis (in particular picture recognition). Here, the patterns under consideration are usually quite complex and the number of features required is often very large. In this kind of problems, the syntactic approach to pattern recognition is more appropriate. This approach draws an analogy between the (tree-like) structure of patterns and the syntax (or grammar) of languages. Patterns are specified as being built up out of subpatterns in various ways of composition, just as phrases and sentences are built up by concatenation of words (and words by concatenating characters). The simplest subpatterns (pattern
primitives' selected, should be much easier to recognize than the patterns themselves.

2.2 PRIMITIVE SELECTION AND PATTERN GRAMMARS

Though pattern primitives are the basic components of a pattern, they are not easy to recognize. For example, strokes are considered good primitives for script handwriting, and so are phonemes for continuous speech; however, neither strokes nor phonemes can easily be extracted by machine. A compromise between its use as a basic part of the patterns and its easiness for recognition is often required in the process of selecting pattern primitives. [Fu'92].

So far there is no general solution for the primitive selection problem. For line patterns or patterns described by boundaries of skeletons, line segments are often suggested as primitives. The information characterizing the primitives can be considered as their associated semantic information or as features used for primitive recognition. Through the structural description and the semantic specification of a pattern, the semantic information associated with its subpatterns or the pattern itself can then be determined.

After pattern primitives are selected, the next step is the construction of a grammar which will generate a language to describe the patterns under study. It is known that increased descriptive power of a language is paid for in terms of increased complexity of the syntax analysis system (recognizer or acceptor). The selection of a particular grammar for pattern description is affected by the primitives selected and by the
The tradeoff between the grammar's descriptive power and analysis efficiency.
Attributed grammars were first formulated by Knuth to assign semantics (or meanings) to context-free languages from the computational point of view. Each production rule of an attributed grammar consists of a syntactic rule (to specify language syntax) and a semantic rule (to add contextual semantic).

The following is a definition of attributed grammars according to the formalism of Knuth:

An attributed context-free grammar is a 9-tuple

$$G = ( \mathcal{U}_N, \mathcal{U}_T, \mathcal{P}, S)$$

where

- $\mathcal{U}_N$ set of nonterminals,
- $\mathcal{U}_T$ set of terminals,
- $S \in \mathcal{U}_N$ start symbol,

for each $X \in \mathcal{U}_N \cup \mathcal{U}_T$, there exists a finite set of attributes $A(X)$, each attribute $\alpha$ of $A(X)$ having a set, either finite or infinite, of possible values $D_\alpha$; and $\mathcal{P}$ is a set of productions each of which is divided into two parts: a syntactic rule and a semantic rule.

In practical applications, pattern classes can often be divided into groups, each group consisting of several pattern classes which are similar in structure but different in attributes. In such cases, it is appropriate to construct an attributed grammar for each group of pattern classes. The discrimination of within-group pattern classes is left to statistical classification on attributes.
CHAPTER THREE

A COMBINED SYNTACTIC/SEMANTIC APPROACH TO PR

11.1 THE COMBINED SYNTACTIC/STATISTICAL APPROACH

"Thus the controversy between geometric and structural approaches for problems of pattern recognition seems to me historically inevitable, but temporary. There are problems to which the geometric approach is ideally suited. Also, there are some well-known problems which, though solvable by the geometric method, are more easily solvable by the structural approach. But any difficult problems require a combination of these approaches, and methods are gradually crystallizing to combine them: the structural approach is the means of construction of convenient space; the geometric is the partitioning in it "[Aizerman'69].

Combining syntactic and statistical pattern recognition approaches, has been advocated by several investigators in the past decade. [Tsai'80]. The motivation arises from the fact that neither the syntactic approach nor the statistical approach alone is adequate for some practical applications. The syntactic is weak in handling noisy patterns and numerical semantic information. Another characteristic drawback of structural methods is the arbitrariness in the selection of primitives (e.g., there exists an infinite number of ways for the construction of grammars for pictures).

The statistical is incapable of describing complex pattern structures and subpattern relations. Since the advantages of one seem to be the drawbacks of the other, a hybrid model that incorporates the advantages of both is desirable in dealing with real applications. The use of attributed grammar was suggested by Tsai and Fu, 1980, as such a tool.
authors have formulated a quite general and powerful scheme for pattern analysis which can be viewed as a combination of syntactic and statistical approaches to pattern recognition.
attributed grammars. The semantic computation is performed simultaneously to obtain all required nonterminal (subpattern) attributes according to the semantic rules. It is possible, during the computation of semantics using the semantic rules, that some nonterminal (subpattern) attributes may not be obtainable by a mapping or computation from lower level terminal attributes. When that occurs, it is necessary to go back to the input pattern, to find out the subpattern corresponding to the nonterminal, and to perform the necessary subpattern attribute extractions as specified in the semantic rule. The authors would like to emphasize here that subpattern attributes cannot be extracted before syntax analysis and semantic computation because, without the guidance of syntax analysis, the system would not know which terminals (primitives) should be grouped into a nonterminal (subpattern). This is indeed an advantage of using attributed grammars because subpattern attribute extraction now becomes more effective with the guidance of syntax analysis [Tsai'80]. Such an advantage is not obtainable by using the statistical approach alone. After this stage, the result is a sense of the pattern representation if the input is syntactically correct, together with its total attribute vector. The parser will reject any syntactically incorrect representation corresponding to a structurally erroneous pattern. The hierarchical syntactic and semantic description of the input pattern is also available now except its class assignment.

The final stage is the decisionmaking performed on the total attribute vector to classify the input pattern. By combining syntax analysis, semantic computation, and decisionmaking, the
From the statistical classification point of view, the incorporation of syntax analysis into statistical decision-making has several advantages over the classical statistical approach:

1. utilization of structural information for pattern description.
2. effective extraction of subpattern attributes, and
3. description and generation of patterns.

On the other hand, from the syntactic point of view, the injection of primitive and subpattern attributes into the grammatical analysis offers the following advantages over the conventional syntactic methods:

1. flexibility in choosing primitives and subpatterns,
2. improvement of recognition accuracy,
3. capability of recognizing noisy patterns, and
4. reduction of grammatical complexity.

3.2.3 A PATTERN ANALYSIS SYSTEM USING ATTRIBUTED GRAMMARS

The block diagram of a syntactic pattern recognition system using attributed grammars is shown in Fig. 2. Given an input pattern for classification, after preprocessing, all necessary primitives and their attributes are extracted according to some prespecified procedures [Tzai'80]. No subpattern attributes are to be extracted at this stage because they are still unknown.

The next step is to transform the primitive set into some structural representation (a string, a tree, etc.) by assigning symbols to primitives, selecting concatenating directions, or any other prespecified relations. The resulting representation is then analyzed syntactically by using the syntactic rules of the
CHAPTER FOUR

PRACTICAL APPLICATIONS OF PR

1.1 BIOMEDICAL APPLICATIONS

The medical applications of pattern recognition are quite numerous. and among the most interesting is electroencephalography.

1.1.1 ELECTROENCEPHALOGRAPHY

The electroencephalogram (EEG) is a complex electrical signal which reflects generalized brain activity. The EEG is utilized in the clinical assessment of many neurological and psychiatric disorders and offers promise for monitoring of patients undergoing anesthesia and operation (Sanderson '80). Automated analysis and classification of EEG signals is hampered by the enormous amounts of data which are generated by detailed descriptors of the nonstationary waveforms.

In order to detect abnormalities, one needs to have a standard domain of the 'normal'. EEG have helped in dealing with epilepsy. The literature contains several systems for the automatic recognition of abnormalities associated with epilepsy. Virtually most of the systems developed have dealt with the detection of spikes and sharp waves (Thorne '91). One of the most frequently used approaches has been to compare the amplitudes, derivatives and durations of EEG wave forms with respect to preset thresholds to indicate the presence of abnormalities.
1.1.2 BRAIN ELECTRICAL POTENTIALS

Few problems are more challenging than decoding the mass electrical activity of the human brain [Gevins'80]. Although this goal is distant, there are currently several useful applications of a correlative approach based on pattern recognition. These include investigation of the neural concomitants of higher cognitive functions, diagnostic screening, differential diagnosis and prognostic assessment of neurologic disorders, classification of the stages of ordinary disturbed sleep, and classification of the neuroelectric patterns associated with different types of psychotronic drugs. Since brain electrical potentials (BEP) are correlated with a variety of behavioral and clinical variables, especially tight experimental designs are necessary.

If features are not neurophysiologically interpretable, the utility of even successful classification is likely to be limited in most applications. Features should expressed in a form which is insensitive to irrelevant anatomic and metabolic variables. Sample size determination is application-specific since intra- and interperson BEP variability is different for different problems [Gevins'80].

Most pattern classification algorithms have been applied to BEP's including decision functions, trainable classification networks, distance functions, syntactic methods, and hybrids of the proceeding.
Experimental psychologists have been interested in the perception of symmetry since the last century [Schaeffer’s1]. The interest has been mainly with the perception of bilateral symmetry. At first glance translational symmetry would appear to be the simplest of the symmetry operations, yet, it is bilateral symmetry that is detected more quickly and with greater accuracy than the other symmetries. A computational model of human symmetry detection was developed by Schaefer for detecting the presence of the various symmetries found in two-dimensional shapes. This model utilizes search strategies and a comparison process for detection. The search strategies consist of the successive selection of feature pairs to compare for the presence or absence of symmetry. A basic search strategy was specifically designed to reduce the number of feature pair comparisons carried out by capitalizing on the differences in the distances between symmetrically related features, that are found in the various symmetries. The results indicate a strong relationship between the behavior of the model and that of human subjects on symmetry detection tasks.

4.1.1.5 APPLICATION TO CANCER

"Breast cancer is the leading cause of death in accidental women. According to statistics, the death rate is 22 per 100,000 female population in the United States for the past 40 years. Early detection increases the five-year survival rate to 90-95% in contrast to 10-50% where detection of breast cancer is delayed."

Silverberg ’71.

According to Fong et. al. (1984), thermography is regarded as a potential screening tool for early detection of breast
cancer. 'but its value is still in doubt'. They conducted a study designed to critically evaluate the validity of thermographic images as an aid to the early detection of breast cancer.

The following is a brief description of the procedure they followed. A set of 24 elemental features was extracted from thermographic images and combined into 14 compound features. These features were used as input to the pattern recognition system using generalized inverse approach. From the result of the pattern recognition, it is found that thermography is not suitable for early breast cancer detection.

4.1.5 ee APPLICATION TO CAROTID WAVES

Methods of syntactic pattern recognition and knowledge based systems for signal understanding have recently been proposed for application in biomedical engineering [Arduino'82]. Among the application of such methods, a system for the interpretation of carotid pulse waves using a general waveform parsing system has been developed by Stockman, Kanal and Kyle.

The motivation for analyzing the carotid signal resides in the fact that carotid artery in the neck is the point in the arterial system close to the heart where external (non-invasive) sensing can be carried out. The carotid pulse wave exhibits wider structural variations than pulse waves from other body points. It is believed that more valuable information, related to age and cardiovascular disease can be gathered from the carotid pulse wave.

Though the work proceed in the Arduino paper follows the lines of Stockman, Kanal and Kyle, important innovations
characterize the Arduino interpretation. One of these is that a functional approximation of the waveform of the search for important features on it is performed after an automatic extraction of the fundamental period. This allows one to make a "best-synchronous" analysis with advantages already tested in systems developed for the automatic processing of electrocardiograms (ECG).

4.2.1 FS APPLICATION TO CHARACTER RECOGNITION

The problem of machine intelligence attracted scientists from the moment the first computer was born [Verschueren '94]. Especially the automatic recognition of data generated by humans (e.g. alphanumerical characters) was a favourite problem in those early days. Some big computer companies spent a great amount of money into this research. Later on the problem was abandoned a little because of the large amount of computing power needed.

Today, with the enormous amount of information produced by men and machine, the problem of automatic recognition of both handprinted and machine printed characters is again up to date.

Many companies try to find a solution to the amount of administration they must treat. Although many years have been spent on this research, no general solution exists. The methods developed "are all partially empirical and suited for one particular problem" [Verschueren '94]. One of the most difficult problems in this area is the recognition of handwritten characters.
"The central problem of cursive script interpretation is that of determining the identity of connected spatial symbols that correspond to words" [Bozinovic '94].

The problem of cursive script differs from optical character recognition which predominantly deals with discrete letters, and its akin to speech recognition dealing with continuous strings of phonemes. Research on word-level recognition of off-line cursive script started almost twenty years ago and has been revisited only occasionally [Amin '94]. The recognition method depends on the type the characters to be recognized. For the Arabic language, e.g., the shape of handwritten Arabic characters depends on their position in the word (at the beginning, in the middle, at the end or isolated characters) and, moreover, some characters are provided with group of dots. These characteristics were used by Amin et. al. to elaborate "an efficient and original method for the recognition of isolated words". (Currently, the same group of authors is involved in a new system using syntactic methods to recognize words within sentences). In their research they combined statistical and syntactic methods.

In general, a cursive word is recognized through a hierarchical analysis: a word is decomposed into letters, letters into strokes and strokes into primitive elements [Edan'81]. The hierarchical analysis can also be made with the aid of geometrical graph representation that models the line structures in a given cursive word [Havas'79].
IPAC (Interactive Recognition of Arabic Characters), developed by Amin et al., is a system capable of translating simple handwritten Arabic sentences into French. Two methods were experimented for the recognition of the words: The former is a syntactical method based on the segmentation of words into primitives like curves and strokes. An automaton transforms the range of primitives into the list of characters constituting the word. The latter method uses a global approach: each word is identified according to a vector of some pre-determined parameters. Then a syntactical and semantical analyzer verifies the grammatical structure and the meaning of the Arabic sentence which is translated word by word into French. The French sentence is pronounced by a speech synthesizer.

4.3 INDUSTRIAL APPLICATIONS OF IPAC

The industrial applications of pattern recognition are so enormous that one can not even attempt to enumerate them. In the following sections only some of these applications will be mentioned.

4.3.1 MACHINE VISION AND ROBOTICS

Although robot vision is a new technology, it has been applied successfully to a variety of industrial needs in inspection, manufacturing, and material handling (Trombly '94). New developments in robot technology are creating practical, cost effective solutions for a variety of industrial needs.

A year or two ago, researchers and robot manufacturers interested in implementing a robot vision application could take one of two
approaches [Trombly and Hudson '94]. The first approach was to purchase all the necessary vision components from various sources. That meant buying an image processor from one company, a camera from another, and lens and light sources from yet another. The user then had to assemble the pieces, and in most instances he had to write all of his own software to test, analyze and process the vision application. The second and most common approach was to contract with the vision equipment vendor for the development and installation of a turnkey inspection or manufacturing system. The robot user and his company paid a premium for their vision system in an effort to assure the success of the system.

Since 1991, emphasis on robotics has skyrocketed [Trombly-'91]. New groups have been formed in many manufacturing companies with the determination to learn about, test and initially apply new robot and automation technologies. Machine vision is one of the new technologies being tested and applied. This focused interest has created a need for a robot vision system that makes it easy for manufacturing engineers to learn about, test, and implement a robot vision application. Trombly and Hudson addressed these needs in a newly developed vision system:

Vision Development System (VDS) is a complete hardware and software product for the development and testing of robot vision applications. A complimentary, low-cost Target Application System (TASK) runs the application program developed with the VDS. An actual robot vision application that demonstrates inspection and
pre-assembly for kiosk manufacturing is used to illustrate the VDS/TASK approach. A detailed description of each stage of development and implementation was elaborated, including:

a. defining the requirements,
b. matching the application to the vision system,
c. developing the application software, and
d. building and installing the Vision System.

1.3.2 DEFECT DETECTION IN TEXTILES

Ade et al. 1984 described a general system for automatic inspection of textured surfaces. The system is applicable, for example, to the industrial automatic inspection of textile fabrics. A normal such web is characterized by a fairly regular and visually homogeneous texture. Any deviation of the structure from normal, exceeding a certain tolerance threshold, is to be detected on-line. The main difficulty in such an application, according to Ade, is to select a set of texture measures capable of adequate representation of the texture and of efficient computability. Measurement of textural neighborhood properties can be achieved using local matches. The resulting system can be looked upon as a bank of filters. A relevant thorough investigation was carried out by Laws previous to the work of Ade. He proposed the characterization of a texture by a set of "texture energy measures" computed by squaring and combining the outputs of a set of filters. Ade extended this approach by the proposal to use first order statistics associated with the coefficients of a sliding local linear transform for texture analysis and characterization. From the results of Ade approach,
the filter sets obtained compare, in general, favorably with the set of empirical filters introduced by Laws.

A RECOGNITION SYSTEM FOR THE REIDENTIFICATION OF MOTOR VEHICLES

In order to obtain reliable data for traffic management and control, it is necessary to re-identify vehicles after passing a specific section of a road network. This allows the measurement of journey times, traffic density, etc.

For this purpose a pattern recognition system was developed by Pfannerstill [1984]. Vehicles are characterized by signals of an inductive loop detector. Information contained in these is compressed on a few features by means of the Karhunen-Loève expansion. This compression is necessary in order to minimize the capacity required for data paths to transmit these features and computer power to correlate the objects with each other.

A simple metric is used to re-identify single vehicles. Correlation analysis is performed for the identification of platoons of vehicles in order to measure the above-mentioned traffic data. The result is a pattern recognition system with only minimal requirements for (1) data paths to transmit the selected features and (2) computer capacity for the correlation.


ORIG1NAL PAGE IS
OF POOR QUALITY

[ 111. Blacke, R.E., "A Weak Algorithm For The Inference Of
String And Tree Grammars.", Proc. Of The 5th Int. Conf. On
Pattern Recognition., Miami Beach, F1. USA. New York. USA,

Recognition Approach To Quantitative Electro-
encephalographic Analysis.". Electroencephalography. And
pp. 57-64.

[ 131. Bosinovic. Radmilo et al., "Knowledge-Based Cursive
Script Interpretation.". Proc. 7th. Int. Conf. On Pattern
Recognition., Montreal. Canada. IEEE. Vol.2. 1984, pp.774-
778.

[ 141. Braccini. C. Gambardella. G. Grattarola. A. "Shift-
Variant Image Processing For Scale-Invariant Recognition"

Utterances.". Proc. Of ICASSP 83. IEEE Int. Conf. On
Acoustics, Speech And Signal Processing. Boston. MA.
14-16 April 1983.

[ 161. Bunke, H., "Experience With Several Methods For The

[ 171. Bunke, H., "Attributed Programmed Graph Grammars And
Their Application To Schematic Diagram Interpretation.",

Straight Lines.". Proc. 7th Int. Conf. On Pattern
Recognition., Montreal. Canada. IEEE. Vol.1. 1984, pp.482-
485.

Rosenfeld. A., "A Study On Parallel Parsing Of Tree
Languages And Its Application To Pattern Recognition.",
Proc. Of Part Of The Japan-United States Seminar On
Research Towards Real-time Parallel Image Analysis And


ORIGINAL PAGE IS OF POOR QUALITY
Sampled Recursive Multidimensional Filtering.", Inf. And

[401] Fu, K.S., "VLSI Architectures For Syntactic Pattern
Recognition Algorithms.", IEEE Workshop On Languages For
Automation, Chicago, IL, USA., Silver Spring, MD, USA.,

Grammars For Image Analysis.", Identification And System
Parameter Estimation, Proc. Of The 6th IFAC Symposium,

[421] Fu, K.S., "A Step Towards Unification Of Syntactic
And Statistical Pattern Recognition.", IEEE Trans. Pattern
pp.200-5.

[431] Fu, K.S., "Pattern Recognition For Automatic Visual

[441] Fu, K.S., "Attributed Grammars For Pattern Recognition -
A General (syntactic-Semantic) Approach.", Proceedings
Recognition And Image Processing., Las Vegas, NV, USA.,

[451] Fu, K.S., Kanal, L.N., Rosenfeld, A., "Recent
Progress In Syntactic Pattern Recognition.", Progress In
Pattern Recognition., Vol.1 Amsterdam, Netherlands., 1991,

[461] Fu, K.S., Gelbana, E.S., Kanal, L.N., "Syntactic
Models And Pattern Recognition And Applications.", Pattern
Recognition In Practice., Proc. Of An Int.Workshop,
Amsterdam, Netherlands., Amsterdam, Netherlands. XIII:552,

[471] Fu, K.S., "Recent Developments In Pattern Recognition."
pp. 945-56.


ORIGINAL PAGE IS OF POOR QUALITY.


ORIGINAL PAGE IS OF POOR QUALITY


BOOK REFERENCES


ORIGINAL PAGE IS OF POOR QUALITY
APPENDIX II. Edge Detection and Image Processing Using a Gradient and Edge Following Method
CHAPTER ONE

INTRODUCTION

A most significant branch of recognition technology is "image processing". It involves transforming pictures into forms that facilitate analysis by machines and/or humans.

Edge detection is generally an important step in automatic image processing. For the most part, edge detection is a two-step process. The initial step consists of determining the local edges (e.g., gray level discontinuities) of an image. The second step in this process involves a method of connecting local edges into global edges. To accomplish this, we shall use an edge following technique.

This paper is an investigation of one method of edge detection. Programs are provided which illustrate the two step process involved in the problem of edge detection. Generally, the detection of edges is started by performing local operations on picture neighborhoods through the use of an edge operator. These operations, sometimes regarded as local edge operations compare intensity values within small regions of the picture. Among the many edge operators which
may be used for detecting edges, the gradient will be used in this paper. It is one of the most commonly used edge operators. The result of the gradient is a vector attached to each picture point indicating the direction of maximum gray level change. The picture is similar to the magnetic field in the space around a magnetic bar where lines of force are used to show the direction a particle would tend to move. The lines of force, like the orientations from the gradient, form a convenient way to create a visual image of the situation.

The gradient operator is applied to each pixel or element in the input picture, thus transforming the picture into an array of gradient vectors. Each gradient magnitude is compared to some threshold value. Those magnitudes less than the threshold value are, for all practical purposes, disregarded as edges. Gradient magnitudes greater than or equal to the threshold values represent significant edges in the image. The technique of thresholding is a way of segmenting or separating the object in a picture from its background, assuming that the picture is simply an object-background image. The process described above is generally referred to as a preprocessing or early processing technique.
A method for edge following will also be presented in this paper. The edge following technique computes a "goodness of fit" measurement for each pair of adjacent pixels. The goodness measure is based on the gradient of each of the two pixels. Combining pairs of pixels that fit well produces longer edges. The outcome of this method is in the form of a chain-coded representation which illustrates the direction of most change in the picture.
CHAPTER TWO

DESCRIPTION OF THE PROBLEM

This paper focuses upon two types of image processing problems. The first, referred to as local edge detection, basically involves locating discontinuities of gray levels in a given picture. The second problem, edge following, involves linking local edges to form global edges. The following paragraphs will be a discussion of these concerns.

2.1 THE EDGE DETECTION PROBLEM

Central to work involving picture processing by a computer is representation of the picture. Since the actual object cannot be manipulated in the computer, a model which can be used is generally constructed to represent the real object.

A picture is typically recorded as a vector or array of fixed dimensions. It consists of components which may be accessed arbitrarily by specifying an index which gives the position of the component within the array. For example, the picture may be given as a rectangular array of $M \times N$ dimensions, where $x = \{0 \ldots M\}$ and $y = \{0 \ldots N\}$.
Before a picture can be analyzed by the digital computer, it must be converted to a discrete form or binary image. This is usually achieved by recording the picture photographically and then transforming it based on intensity values within local regions of the picture. In the case of the black and white picture, these values are referred to as gray levels.

A point, \((x,y)\), in the digitized picture is referred to as a pixel. Each pixel can be said to represent a square, with the horizontal and vertical neighbors of \((x,y)\) sharing a common boundary, and its diagonal neighbors touching it only at a corner. (See fig. 1)

![Diagram of a pixel on a 3x3 grid and its eight adjacent neighbors.](image)

fig. 1. Representation of a pixel on a 3 x 3 grid and its eight adjacent neighbors.
Except for those pixels in the first or last row or column, each pixel in a rectangular grid representation has eight immediate neighbors associated with it as shown in fig. 1. The neighbors are identified as

\[(x,y)\rightarrow (x-1,y-1),(x-1,y),(x-1,y+1),(x,y-1)
\[(x,y+1),(x+1,y-1),(x+1,y),(x+1,y+1)\].

An edge element \((x,y)\) defines the common boundary between two adjacent elements. An edge is a sequence of edge elements \((x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\) such that the sequence of boundary segments they define on the rectangular grid is connected. The direction of a segment is determined by moving clockwise around the boundaries of the edge element.

Each pixel has a gray level value which can be modified by applying an operation to it. There should be an abrupt change (edge) in gray level at the boundary between black areas and white areas. For instance, by applying an edge detecting operator to a black and white picture, edges can be detected. A simple operator for detecting edges is the gradient. This operation on a picture computes a value (gradient magnitude) which reflects the amount of variation in gray level at a pixel. That is, high values are in areas
where edges are located and low values elsewhere. It also computes a direction which corresponds with the direction of maximum change in gray level.

Thresholding these magnitudes is a way of segmenting the picture into two regions of gray levels. Points where the gradient magnitude is greater than or equal to a certain threshold is determined to be an edge element candidate and is set to the number 1, 0's are assigned elsewhere. This new image is a two-valued image sometimes referred to as a binary image or discrete values.

2.2 THE EDGE FOLLOWING PROBLEM

Edge following is implemented after the set of edge element candidates have been isolated. The focus of the edge following technique is the location of connected edge elements and a process for linking these components as an indication of the "goodness" of an edge. Goodness here refers to a measure of how well two adjacent edge elements "fit together". A local neighborhood of a pixel is defined as the pixel and its eight adjacent neighbors on a 3 x 3 grid. A check in the direction of each adjacent neighbor for one which has the largest "goodness" in relation to the central element, yields the next element. All edge element
candidates (those pixels marked with the number 1) would be processed in the same manner. The goodness operator, a locally applied operator, computes a goodness of fit measure in the local neighborhood of each central pixel, which allows these comparisons within the neighborhood to be successfully implemented.

As aforementioned, when we view a pixel as a square on a rectangular grid, it can have eight adjacent neighbors. To determine which neighbor has been chosen to have the best fit to the central pixel in a local neighborhood, a direction code represented by the numbers \{1,2,3,\ldots,8\} is assigned to each of the eight neighboring positions. The coding procedure used to identify a neighbor is straightforward, it assigns the adjacent pixel vertically upward from the central pixel the code 1, and other codes can be determined by making assignments progressively while moving in a clockwise direction. An illustration of the coding scheme is provided in figure 2.
fig. 2. Coding scheme used in the assignment of direction codes.

In generally, we can say that the codes represent arrows or orientations as indicated by the coding scheme. As the picture is scanned, a direction code is assigned which corresponds with the neighbor having the "best fit" to the central pixel. The codes generated from this can be said to represent a "coded picture" when displayed in the form of a matrix, as shown in fig. 3.
Here, the coded picture illustrates the direction in which the maximum amount of change is occurring in the picture.
CHAPTER THREE

IMPLEMENTATION NOTES

The thrust of this study is edge detection and edge following. The proceeding is an explanation of several programming considerations. The program was implemented in standard PASCAL on the VAX 11/780.

3.1 INPUT DATA

The picture will be in the form of a black and white representation. Each pixel in the picture provide some detail about the picture. In this case, a description is needed which represents both the image (black area) and the background (white area) of the picture. By using an asterisk (*) at each pixel where the image is located, representation of the desired image is generated. The background is made up of any other character. Using this arrangement, a number of different images may be created and processed, respectively. This picture will represent the original or input picture, as illustrated in fig. 4.
A representation of the input picture.

The picture is placed into a data file which is a way of assuring that it will not be destroyed or altered in any way. However, for the purpose of analysis, the picture is copied into an array, of a size equal to the data file.

3.2 INTERNAL CODE

In order to further process the picture it is necessary to encode the input picture numerically. For this purpose, descriptive values are assigned to each pixel of the picture. This is accomplished by scanning the picture from left to right, top to bottom, whenever an asterisk is encountered the number 10 is assigned to that pixel, otherwise a 0 is assigned. The values are descriptive of
the data located at each pixel. In this case, the values represent the two regions of gray levels. The number 10 represents the black area and 0 is for the white area. The choice of the numbers is arbitrary, but must be different enough to distinguish the two regions. This numerical description of the picture is stored in the new array, NUMPIC[x,y]. It is obvious that before the picture is completely analyzed, it will undergo a number of transformations. In fact NUMPIC[x,y] will be used in subsequent algorithms for further manipulation and gray level interpretations. Using the input picture in fig. 4, the data in the new array would be represented as shown in fig. 5.

```
0 0 0 0 0 0 0
0 10 10 10 10 0
0 10 0 0 10 0
0 10 0 0 10 0
0 10 10 10 10 0
0 10 10 10 10 0
0 0 0 0 0 0
```

fig. 5. The numerically encoded picture, NUMPIC.
4.1 EDGE DETECTION TECHNIQUE

It is known, in general, that an edge represents a location in the picture where an abrupt change occurs. To determine where these changes are occurring, a gradient operator is applied to the picture. Since the gradient operator is a continuous operator, it must be converted to a difference operation for use on a discrete picture.

We use:

\[ \text{Diffx} = \text{Numpic}[x,y] - \text{Numpic}[x+1,y] \]
\[ \text{Diffy} = \text{Numpic}[x,y] - \text{Numpic}[x,y+1] \]

Using data from the numerically encoded picture, this process would generate data similar to that shown in figs. 6a and 6b.
fig. 6.

(a) Diffx

\[
\begin{array}{cccccc}
0 & -5 & -5 & -5 & -5 & 0 \\
0 & 5 & 10 & 10 & 5 & 0 \\
0 & 5 & 0 & 0 & 5 & 0 \\
0 & 5 & -5 & -5 & 5 & 0 \\
0 & 5 & 0 & 0 & 5 & 0 \\
0 & 10 & 10 & 10 & 10 & 0 \\
\end{array}
\]

(b) Diffy

\[
\begin{array}{cccccc}
0 & 0 & 0 & 0 & 0 & 0 \\
-5 & 10 & 0 & -5 & 10 & 0 \\
-5 & 5 & 5 & 5 & 10 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
-5 & 5 & 5 & 5 & 10 & 0 \\
-5 & 10 & 0 & -5 & 10 & 0 \\
\end{array}
\]

fig. 6. (a) and (b) Represent data in the two difference matrices, Diffx and Diffy.
Using Diffx and Diffy as components of the gradient, this operator is now represented as:

\[
\text{Gradient} = \text{Sqr.}(\text{Diffx}[x,y] + \text{Diffy}[x,y])
\]

Gradient magnitudes are generated by applying the operator to each element in the numeric picture starting in the first row and ending in the last one. (See fig. 7a) With this data, edges can be located by thresholding the magnitudes. The threshold is supplied at runtime.

The magnitudes generated from this operation which are greater than the threshold value are recorded as edge data, and are marked with the number 1, otherwise a value of 0 is assigned. The 1's and 0's are stored into an array which represents the picture as discrete values. (See fig 7b)
fig. 7.

(a) Picture of gradient magnitudes.

(b) Image of discrete values (local edges).
4.2 EDGE FOLLOWING METHOD

Given the local edges found in the proceeding operation, we use the goodness operator to determine how well adjacent pixels are connected to the central pixel. The goodness operator uses gradient magnitudes and information from the two differenced matrices, \((\text{Diff}_x[x,y], \text{Diff}_y[x,y])\). The goodness operator is defined as:

\[
\text{GOODN} = \frac{(\text{Diff}_x[x,y] \cdot \text{Diff}_x[x,y]) \cdot \text{Grad}[x,y]}{\text{Diff}_y[x,y] \cdot \text{Diff}_y[x,y]} + \text{Grad}[x,y]
\]

In this formula, \((x,y)\) locates data at the position of an adjacent neighbor and \((x,y)\) locates data at the central pixel in a local neighborhood. Using this equation, a goodness measure is computed for each of the neighbors in the local neighborhood of the central pixel. The goodness value for the eight neighbors are compared to each other. The "best goodness" is recorded together with the position of the adjacent pixel with the best fit, the position of the central pixel, and its gradient magnitude.
Recall, the gradient computes an orientation of discontinuity, which are basically directed lines indicating the direction of maximum change in the picture. By knowing the orientation of the line formed from the x and y coordinates of the central pixel in terms of its gradient (Diffx, Diffy) and that of its adjacent neighbor of best fit within the local neighborhood, an angle can be formed. The angle is formed when an adjacent point is projected onto the central point. The two vectors can be used to reflect how parallel or perpendicular the gradients of two pixels are to each other. The more parallel the gradients are, the smaller the goodness measure. The more perpendicular the gradients are, the larger the goodness measure. More parallel gradients indicate a more likely (smoother) continuation of an edge.

Each of the eight neighbors in the region is assigned a code {1, 2, ..., 8} moving in a clockwise fashion. The code represents the direction in which the edge is moving. It is also recorded as data from this procedure. Here the edge pixel represents the central pixel in the local neighborhood and the neighbor with the best goodness reflects the adjacent neighbor with a magnitude most related to the magnitude of the central pixel. Local edges are generated
by applying the goodness operator to each local neighborhood where a pixel at an edge exists.

The edge following method is controlled by a procedure which produces a record of local 'edges', a gradient magnitude, a goodness measure, and a chain-coded direction. An illustration of a chain-coded edge image for our example is shown in fig. 8.

```
0 0 0 0 0 0 0
0 0 3 7 1 0 0
0 1 0 0 1 0 0
0 5 0 0 5 0 0
0 3 3 7 5 0 0
0 0 0 0 0 0 0
```

fig. 8. The chain-coded image for sample input picture.

In this way, the chain codes can be used to link the local edges together to form a global edge.
A method for edge detection is discussed in this section which allows for simple patterns to be conveniently represented and analyzed for image processing. The strategy used in solving the edge detection/edge following problem is outlined below.

5.1 MAJOR DATA STRUCTURES

RSIZE: row size
CSIZE: column size
UBORDERX and LBORDERX: upper and lower boundaries of the x coordinate
UBORDERY and LBORDERY: upper and lower boundaries of the y coordinate
DIFFX and DIFFY: components of the gradient
GRADIENT: length of the gradient
THRESHOLD: a value initiated at run time
CTX,CTY : the central pixel of the local neighborhood

ORIGPIC[ctx,cty] : a storage location for the picture

NUMPIC[ctx,cty] : an array holding the numeric picture

ADJPTARY : an array of adjacent neighbors

ADJX,ADJY : the adjacent pixel with the best fit

GOODN : a measure of how well an adjacent pixel fits together with a central pixel

BESTGOODN : the goodness measure of the adjacent pixel with the best fit to the central pixel

CODE[ctx,cty] : array of direction codes

RESULT[ctx,cty] : array of 1's and 0's

5.2 INPUT FILE DESCRIPTION

PICTURE is a text file where the input picture is stored. Using asterisks (*) to outline the image (black area) and any other character to represent the background (white area), the picture is created. When a new picture is
to be processed, the existing input picture would need to be
discarded in order to create a new one in the file.

5.3 MAJOR PROCEDURES

A. MAIN PROGRAM

1. Read the picture into array ORIGPIC.
   (PICMATRX)

2. Encode the picture numerically, by scanning
   the picture and replacing the asterisks
   encountered with the number 10, otherwise
   assign 0. Store the new image in array
   NUMPIC. (CHANGEPIC)

3. Develop the components (DIFFx and DIFFy) of
   the gradient operator. (DIFFERENCE)

4. Apply the gradient operator to the picture.
   Compare the gradient magnitudes to the
   threshold; if the gradient[ctx,cty] is greater
   than or equal to the threshold assign a 1
   to pixel, otherwise assign a value of 0.
   Store this data in array RESULT. (GRADTHRSHL)

5. Search and produce the chaincodes. Store the
   codes into array CODE. (NEXTSEARCH)

6. Display array CODE. (FINALMTRX).
B. NEXTSEARCH

For each pixel with a gradient magnitude greater than the threshold do the following:

1. Find the adjacent neighbors of the pixel and store this data in array ADJPTARY.
   (FINDADJACENT)

2. Compute a goodness of fit measure for each adjacent neighbor in the local neighborhood of the central pixel. Compare the values within the local neighborhood to determine which has the largest "goodness" relative to the central pixel. Output the coordinates of the central pixel, the coordinates of the adjacent neighbor with the best goodness, the gradient magnitude of the central pixel and the goodness measure. (GOODNESS)

3. Direction code is computed and stored in array CODE.

C. FINALMTRX

Output the array of direction codes.
CHAPTER SIX

CONCLUSION

This paper provides a system of picture processing for the detection of edges in gray level pictures. To illustrate the effectiveness of program EDGEDETCT several images have been processed. Images tested in this study are size 10 x 10. In this program, the user must create the input picture in a text file before the processing begins and must also supply a threshold value at run time. By entering the threshold interactively, it has been determined that for the examples used, threshold values between 8 and 14 are considered appropriate. That is, any number greater than 14 will cause a loss of relevant edge data and values less than 8 will pickup undesirable data.

An edge following technique is also presented which illustrates the concept of edge "goodness". The edge following technique uses a 3x3 grid representation, which includes a central pixel and its eight immediate neighbors. Using a local goodness operator in local neighborhoods of the picture, yields a list of connected elements. A coded direction scheme is employed which plots these relationships.
in a chain-coded matrix. The coded picture may be interpreted as a direction matrix where the codes represent orientations of directionality.

Basically, the mechanisms involved in the edge detection/edge following technique include a series of transformations of an input picture. Starting with an original image, a new image is produced. In a sense, our beforehand knowledge of the original picture is limited, we know more about the new image since each transformation provides additional information which can be better interpreted by us. For example, the comparison between the original picture (figs. 4 and 5) and the chain-coded image (fig. 8) produces knowledge about the original picture that we can better understood. In the instance of our example, we can interpret the codes to present the direction of maximum change in the picture.

One of the difficulties with the program described in its present form is the problem it has in interpreting the corners of an object during the scanning of the pictures to locate local edges. There was also a limitation in the types of pictures which could be created. Simple patterns were primarily used in this investigation.
Despite these drawbacks, the pictures sampled indicated that program EDGEDETCT was effective in the extraction of local edges. In addition, the program was capable of growing an edge and determining the shape of objects.

Although the program sends the output data to a standard data file, it can also be captured in the Virtual Memory Storage (VMS) using a Define/User command. A copy of the program is provided in the APPENDIX section of the paper, along with printouts of the sampled data.
PROGRAM EDGEDETCT(INPUT,OUTPUT,PICTURE);
const RSIZE=9;
  CSIZE=9;
  UBDORDX=9;
  LDLORDX=0;
  UBDORDY=9;
  LDLORDY=0;

type
  GRADARRAY=ARRAY0..CSIZE,0..RSIZE1DF REAL;
  ORIGINAL=ARRAY0..CSIZE,0..RSIZE1DF CHAR;
  ORIG=ARRAY0..CSIZE,0..RSIZE1DF REAL;
  DELTAY=ARRAY0..CSIZE,0..RSIZE1DF REAL;
  DELTAX=ARRAY0..CSIZE,0..RSIZE1DF REAL;
  TEMTYPE=ARRAY0..CSIZE,0..RSIZE1DF INTEGER;
  DIRECTION=ARRAY0..CSIZE,0..RSIZE1DF INTEGER;
  GOODVALS=ARRAY0..71F REAL;
  POSITION=RECORD
    X: INTEGER;
    Y: INTEGER;

END;
ADJARRAY=ARRAY1D1F POSITION;

var PICTURE:TEXT;
  ORISPIC:ORIGINAL;
  NUMPIC:ORIG;
  DIFFX:DELTAX;
  DIFFY:DELTAY;
  ADJPTARY:ADJARRAY;
  GRAD:GRADARRAY;
  RESULT,TEMP:TEMTYPE;
  CTX,CTY:INTEGER;
  THRESHOLD:REAL;
  BESTGOOD:REAL;
  PX, PY: INTEGER;
  CODE:DIRECTION;
This procedure copies the picture of asterisks and blanks from the data file PICTURE.dat where it is stored, into matrix array ORIGPIC[x,y].

PROCEDURE PICMATX(ORIGPIC:ORIGINAL);
VAR X, Y: INTEGER;
BEGIN (** PICMATX **)
  RESET(PICTURE);
  WRITELN;
  BEGIN
    FOR I:=0 TO PIMAGE DO
      BEGIN
        FOR J:=0 TO CIMAGE DO
          BEGIN
            READLNP(PICTURE, ORIGPIC[I,J]);
            REND(PICTURE);
          END;
          WRITELN;
        END;
      END;
    WRITELN;
  END;
END; (** END OF PICMATX **)
$ Procedure CHANGEPIC, scans the picture and assigns a number to each point in the picture. The number 10 is assigned to points where an asterisk resides and 0's elsewhere. This new picture is stored in array NUMPIC.[x,y].$

PROCEDURE CHANGEPIC(ORIGPIC:ORIGINAL;VAR NUMPIC:ORIG);
VAR X,Y:INTEGER;

BEGIN (## CHANGEPIC ##)
   WRITELN(' THIS IS THE PICTURE');
   WRITELN;
   FOR X:=0 TO ASIZE DO  
      BEGIN
         FOR Y:=0 TO CSIIE DO
            BEGIN
               IF ORIGPIC[X,Y]= '8'   
                  THEN NUMPIC[X,Y]:=10
               ELSE
                  NUMPIC[X,Y]:=0;
               WRITE(TRUNC(NUMPIC[X,Y]):4);
               WRITELN;
            END;
            WRITELN;
         END;
      WRITELN;
END; ( END OF CHANGEPIC )
PROCEDURE DIFFERENCE(NUMPIC:ORIG;VAR DIFFX:DELTA;VAR DIFFY:DELTA); 
VAR X,Y:INTEGER;
BEGIN (## DIFFERENCE ##)
   WRITELN(' DIFF(Y)'); 
   FOR X:=0 TO RSIZE DO 
      BEGIN 
         WRITELN;
         FOR Y:=0 TO CSIZE DO 
            BEGIN 
               DIFFYX,Y1:=(NUMPIC[X,Y])-(NUMPIC[X+1,Y])/2;
               WRITE(TRUNC(DIFFYX,Y1):4);
               END;
               WRITELN;
      END;
   WRITELN;
   WRITELN;
   WRITELN(' DIFF(X)');
   WRITELN;
   FOR X:= 0 TO (RSIZE - 1) DO 
      BEGIN 
         WRITELN;
         FOR Y:=0 TO CSIZE DO 
            BEGIN 
               DIFFX,Y1:=(NUMPIC[X,Y])-(NUMPIC[X,Y+1])/2;
               WRITE(TRUNC(DIFFX,Y1):4);
               END;
               WRITELN;
      END;
   WRITELN;
   WRITELN;
END; ( END OF DIFFERENCE )

Procedure SRADTHRSHLD computes a gradient magnitude at edge pixel which is compared to a threshold value that the user designates. Points where the gradient magnitude is greater than the threshold are marked with 1; the number 0 is assigned. The 1's and 0's are stored in array RESULT[ctx,cty].

PROCEDURE SRADTHRSHLD(CTX, CTY: INTEGER; VAR RESULT: TEMPTYPE; VAR SRAD: GRADARRAY);

VAR THRESHOLD: REAL;

BEGIN

WRITELN;
WRITELN('PLEASE ENTER THE THRESHOLD VALUE');
READLN(THRESHOLD);
FOR CTX:=0 TO RSIZE DO
  FOR CTY:=0 TO CSIZE DO
    IF SRAD[CTX,CTY] != THRESHOLD THEN
      ELSE
      BEGIN
        RESULT[CTX,CTY]:=1;
        RESULT[CTX,CTY]:=0;
      END;
WRITELN;
WRITELN;
WRITELN;
WRITELN('GRA2 RESULT:');
WRITELN;
FOR CTX:=0 TO RSIZE DO
  BEGIN
    FOR CTY:=0 TO CSIZE DO
      WRITE(RESULT[CTX,CTY]:4);
    WRITELN;
  END;
WRITELN;
WRITELN;
END; (END OF SRADTHRSHLD)
PROCEDURE FINDADJACENT (CTX, CTY: INTEGER; VAR ADJARRAY: ADJARRAY);
VAR I: INTEGER;
BEGIN (* FINDADJACENT *)
FOR I := 0 TO 7 DO
BEGIN
    ADJARRAY[I].I := -99;
    ADJARRAY[I].Y := -99;
END;
IF CTY = LBORDERY
THEN IF CTX = LBORDERX
    THEN BEGIN
        ADJARRAY[0].I := CTX + 1;
        ADJARRAY[0].Y := CTY;
        ADJARRAY[1].I := CTX + 1;
        ADJARRAY[1].Y := CTY + 1;
        ADJARRAY[2].I := CTX;
        ADJARRAY[2].Y := CTY + 1;
    END
ELSE IF CTX = UBORDERX
    THEN BEGIN
        ADJARRAY[3].I := CTX - 1;
        ADJARRAY[3].Y := CTY + 1;
        ADJARRAY[4].I := CTX - 1;
        ADJARRAY[4].Y := CTY;
    END
ELSE BEGIN
    ADJARRAY[0].I := CTX + 1;
    ADJARRAY[0].Y := CTY;
    ADJARRAY[1].I := CTX + 1;
    ADJARRAY[1].Y := CTY + 1;
    ADJARRAY[2].I := CTX;
    ADJARRAY[2].Y := CTY + 1;
    ADJARRAY[3].I := CTX - 1;
    ADJARRAY[3].Y := CTY + 1;
    ADJARRAY[4].I := CTX - 1;
    ADJARRAY[4].Y := CTY;
END
END.
ELSE IF CTY=UBORDERY
THEN IF CTI=LBORDERX
THEN BEGIN
  ADJPTARY[0].X:=CTI+1;
  ADJPTARY[0].Y:=CTY;
  ADJPTARY[1].X:=CTI;
  ADJPTARY[1].Y:=CTY-1;
  ADJPTARY[2].X:=CTI+1;
  ADJPTARY[2].Y:=CTY+1;
  ADJPTARY[3].X:=CTX;
  ADJPTARY[3].Y:=C?Y-1;
END
ELSE IF CTI=UBORDERX
THEN BEGIN
  ADJPTARY[4].X:=CTX+1;
  ADJPTARY[4].Y:=CTY;
  ADJPTARY[5].X:=CTX;
  ADJPTARY[5].Y:=CTY-1;
  ADJPTARY[6].X:=CTX;
  ADJPTARY[6].Y:=CTY+1;
  ADJPTARY[7].X:=CTX+1;
  ADJPTARY[7].Y:=CTY-1;
END
ELSE BEGIN
  ADJPTARY[4].X:=CTX;
  ADJPTARY[4].Y:=CTY;
  ADJPTARY[5].X:=CTX;
  ADJPTARY[5].Y:=CTY-1;
  ADJPTARY[6].X:=CTX;
  ADJPTARY[6].Y:=CTY+1;
  ADJPTARY[7].X:=CTX+1;
  ADJPTARY[7].Y:=CTY-1;
END
ELSE IF CTI=LBORDERI
THEN BEGIN
  ADJPTARY[0].X:=CTX+1;
  ADJPTARY[0].Y:=CTY;
  ADJPTARY[1].X:=CTX+1;
  ADJPTARY[1].Y:=CTY+1;
  ADJPTARY[2].X:=CTX;
  ADJPTARY[2].Y:=CTY+1;
  ADJPTARY[3].X:=CTX;
  ADJPTARY[3].Y:=CTY+1;
  ADJPTARY[4].X:=CTX+1;
  ADJPTARY[4].Y:=CTY-1;
  ADJPTARY[5].X:=CTX+1;
  ADJPTARY[5].Y:=CTY+1;
  ADJPTARY[6].X:=CTX+1;
  ADJPTARY[6].Y:=CTY-1;
END
ELSE IF CTI=UBORDERI
THEN BEGIN
  ADJPTARY[4].X:=CTX;
  ADJPTARY[4].Y:=CTY;
  ADJPTARY[5].X:=CTX;
  ADJPTARY[5].Y:=CTY-1;
  ADJPTARY[6].X:=CTX;
  ADJPTARY[6].Y:=CTY+1;
  ADJPTARY[7].X:=CTX+1;
  ADJPTARY[7].Y:=CTY-1;
END
ELSE BEGIN
  ADJPTARY[4].X:=CTX;
  ADJPTARY[4].Y:=CTY;
  ADJPTARY[5].X:=CTX;
  ADJPTARY[5].Y:=CTY-1;
  ADJPTARY[6].X:=CTX;
  ADJPTARY[6].Y:=CTY+1;
  ADJPTARY[7].X:=CTX+1;
  ADJPTARY[7].Y:=CTY-1;
END
ELSE IF CTX=UBORDER1
THEN BEGIN
  ADJPTARY[2].I:=CTX;
  ADJPTARY[2].Y:=CTY+1;
  ADJPTARY[3].I:=CTX-1;
  ADJPTARY[3].Y:=CTY+1;
  ADJPTARY[4].I:=CTX-1;
  ADJPTARY[4].Y:=CTY;
  ADJPTARY[5].I:=CTX-1;
  ADJPTARY[5].Y:=CTY-1;
  ADJPTARY[6].I:=CTX;
  ADJPTARY[6].Y:=CTY-1;
END

ELSE BEGIN
  ADJPTARY[0].I:=CTX+1;
  ADJPTARY[0].Y:=CTY;
  ADJPTARY[1].I:=CTX+1;
  ADJPTARY[1].Y:=CTY+1;
  ADJPTARY[2].I:=CTX;
  ADJPTARY[2].Y:=CTY+1;
  ADJPTARY[3].I:=CTX-1;
  ADJPTARY[3].Y:=CTY+1;
  ADJPTARY[4].I:=CTX-1;
  ADJPTARY[4].Y:=CTY;
  ADJPTARY[5].I:=CTX-1;
  ADJPTARY[5].Y:=CTY-1;
  ADJPTARY[6].I:=CTX;
  ADJPTARY[6].Y:=CTY-1;
  ADJPTARY[7].I:=CTX;
  ADJPTARY[7].Y:=CTY-1;
END; (END OF IF)

WRITEY;
(WRITELN('POSITION', CTX, CTY);)
WRITEY;
(FOR I:=0 TO 7 DO
  WRITELN(ADJPTARY[II].X, ADJPTARY[II].Y);)

END; (END OF FINDADJACENT)
The following procedure is used to compute a goodness of fit measure for each pair of adjacent pixels. This measure determines whether an adjacent neighbor to a central pixel fits well together.

```
PROCEDURE GOODNESS(CTX, CTY: INTEGER; ADJPARY: ARRAY; VAR BESTGOODN: REAL;
VAR ADJX, ADJY: INTEGER; GRAD: GRADARRAY; CODE: DIRECTION);

VAR I, K: INTEGER;
GOODN: REAL;

BEGIN (### GOODNESS ###)

BESTGOODN:= -999;
ADJX:= 0;
ADJY:= 0;
FOR I:= 0 TO 7 DO
  IF (ADJPARY[I].X=0) THEN
    BEGIN
      IF (GRAD[ADJPARY[I].X, ADJPARY[I].Y]=0) THEN
        GOODN:= 0
      ELSE
        GOODN:=(ABS((DIFFX[CTX, CTY]+DIFFY[ADJPARY[I].X, ADJPARY[I].Y])
        +(DIFFY[CTX, CTY]+DIFFY[ADJPARY[I].X, ADJPARY[I].Y]))/
        (GRAD[ADJPARY[I].X, ADJPARY[I].Y]));

    IF BESTGOODN<GOODN
      THEN BEGIN
      BESTGOODN:= GOODN;
      ADJX:= ADJPARY[I].X;
      ADJY:= ADJPARY[I].Y;
      END;
      END;

WRITELN;
WRITELN(CTX, CTY: 2, (TRUNC(Grad[CTX, CTY])), ADJX, ADJY: 2, (TRUNC(BESTGOODN)));
WRITELN;
WRITELN;
END: (END OF GOODNESS)
```
Procedure NEXTSEARCH calls procedures FINDADJACENT and GOODNESS, then it assigns a direction code which corresponds to each respective adjacent point. The code is generated in this procedure in a clockwise direction. The codes are stored in array CODE[ctx,cty].

PROCEDURE NEXTSEARCH (CTX, CTY: INTEGER; VAR CODE: DIRECTION);
BEGIN
  FINDADJACENT(CTX, CTY, ADJPTARY);
  GOODNESS(CTX, CTY, ADJPTARY, BESTGOODN, ADJX, ADJY, GRAD, CODE);
  BEGIN
    IF (ADJX=ADJPTARY[O].X) AND (ADJY=ADJPTARY[O].Y) THEN CODE[CTX, CTY]:=1;
    IF (ADJX=ADJPTARY[1].X) AND (ADJY=ADJPTARY[1].Y) THEN CODE[CTX, CTY]:=2;
    IF (ADJX=ADJPTARY[2].X) AND (ADJY=ADJPTARY[2].Y) THEN CODE[CTX, CTY]:=3;
    IF (ADJX=ADJPTARY[3].X) AND (ADJY=ADJPTARY[3].Y) THEN CODE[CTX, CTY]:=4;
    IF (ADJX=ADJPTARY[4].X) AND (ADJY=ADJPTARY[4].Y) THEN CODE[CTX, CTY]:=5;
    IF (ADJX=ADJPTARY[5].X) AND (ADJY=ADJPTARY[5].Y) THEN CODE[CTX, CTY]:=6;
    IF (ADJX=ADJPTARY[6].X) AND (ADJY=ADJPTARY[6].Y) THEN CODE[CTX, CTY]:=7;
    IF (ADJX=ADJPTARY[7].X) AND (ADJY=ADJPTARY[7].Y) THEN CODE[CTX, CTY]:=8;
    CODE[CTX, CTY]:=-;
  END;
END; (* END OF NEXTSEARCH *)
PROCEDURE FINALMATRIX(TEMP;TEMPTYPE;VAR CODE;DIRECTION);
VAR X,Y:INTEGER;
BEGIN (** FINALMATRIX **) 
  WRITELN;
  WRITELN(' DIRECTION CODE MATRIX:');
  WRITELN;
  FOR X:=0 TO ASIZE DO
    BEGIN
      FOR Y:=0 TO CSIZE DO
        WRITE(CODE(X,Y));
        WRITELN;
    END;
  WRITELN;
END; ( END OF FINALMATRIX )
BEGIN (II MAIN PROGRAM II)

PICMATX(ORIGPIC);
CHANGEPIC(ORIGPIC,NUMPIC);
DIFFERENCE(NUMPIC,PICX,PICY);
GRADTHRESH(CTX,CTY,RESULT,GRAD);

WRITELN('START POSITION', ', ' GRADI', ', ' ADJ-POSITION', ', ' BEST SNESs');
FOR CTX:=O TO RSIZE DO
  FOR CTY:=O TO CSIZE DO
    IF RESULT[CTX,CTY]=1
      THEN BEGIN
        NEXTSEARCH(CTX,CTY,CODE);
        END;
  WRITELN;
  WRITELN;
FINALMTRX(TMP,CODE);
END. (END OF MAIN)

ORIGINAL PAGE 1
OR POOR QUALITY
THE NUMERICALLY ENCODED PICTURE:

- The block letter C

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 10 10 10 10 10 10 10 0 0 0 0 0 0 0 0 0 0 0
0 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 10 10 10 10 10 10 10 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

DIFF(Y)

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 -5 5 5 5 5 5 5 10 0 0 0 0 0 0 0 0 0 0 0
0 -5 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 -5 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 -5 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 -5 5 5 5 5 5 5 10 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

```

DIFF(X)

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 -5 -5 -5 -5 -5 -5 -5 0 0 0 0 0 0 0 0 0 0 0
0 0 5 10 10 10 10 10 10 0 0 0 0 0 0 0 0 0 0 0
0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 5 -5 -5 -5 -5 -5 -5 0 0 0 0 0 0 0 0 0 0 0
0 0 10 10 10 10 10 10 10 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

91
**GRAD/THRESHOLD RESULT:**

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 1 1 0
0 0 1 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 0 1 1 1 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

**CHAIN-CODED IMAGE:**

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 3 3 3 7 7 7 0
0 0 1 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 0 5 0 0 0 0 0 0 0
0 0 3 3 3 3 7 7 7 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```
THE NUMERICALLY ENCODED PICTURE:

- The letter X

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 10 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 10 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 10 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

DIFF(Y)

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -5 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | -5 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | -5 | 10 | 0 | 0 | 0 | 0 | -5 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | -5 | 10 | 0 | -5 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | -5 | 5 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | -5 | 10 | 0 | -5 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | -5 | 10 | 0 | 0 | 0 | 0 | -5 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -5 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | -5 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

DIFF(X)

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0 | -5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 10 | -5 | 0 | 0 | 0 | 0 | -5 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 10 | -5 | 0 | 0 | -5 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 10 | -5 | -5 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | -5 | 10 | 10 | -5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | -5 | 10 | 0 | 0 | 10 | -5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -5 | 10 | 0 | 0 | 0 | 0 | 0 | 10 | -5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

93
**GRAD/THRESHOLD RESULT:**

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**CHAIN-CODED IMAGE:**

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

94
THE NUMERICALLY ENCODED PICTURE:

- The number 8 in block form.

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 10 10 10 10 10 10 10 0 0 0 0 0 0
0 0 10 0 0 0 0 0 0 10 0 0 0 0 0
0 0 10 0 0 0 0 0 0 10 0 0 0 0 0
0 0 10 10 10 10 10 10 10 0 0 0 0 0 0
0 0 10 0 0 0 0 0 0 10 0 0 0 0 0
0 0 10 10 10 10 10 10 10 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

DIFF(Y)

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 -5 5 5 5 5 5 5 10 0 0 0 0 0 0
0 -5 10 0 0 0 0 -5 10 0 0 0 0 0 0
0 -5 10 0 0 0 0 -5 10 0 0 0 0 0 0
0 -5 5 5 5 5 5 5 10 0 0 0 0 0 0
0 -5 10 0 0 0 0 -5 10 0 0 0 0 0 0
0 -5 10 0 0 0 0 -5 10 0 0 0 0 0 0
0 -5 5 5 5 5 5 5 10 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

DIFF(X)

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 -5 -5 -5 -5 -5 -5 -5 0 0 0 0 0 0
0 0 5 10 10 10 10 5 0 0 0 0 0 0 0
0 0 5 0 0 0 0 5 0 0 0 0 0 0 0 0
0 0 5 -5 -5 -5 -5 -5 5 0 0 0 0 0 0
0 0 5 10 10 10 10 5 0 0 0 0 0 0 0 0
0 0 5 0 0 0 0 5 0 0 0 0 0 0 0 0
0 0 5 -5 -5 -5 -5 -5 5 0 0 0 0 0 0
0 0 10 10 10 10 10 10 0 0 0 0 0 0 0 0
```

95

\( \mu = -2 \)
### GRAD/THRESHOLD RESULT:

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### CHAIN-CODED IMAGE:

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
**THE NUMERICALLY ENCODED PICTURE:**

- The block letter A.

```
<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

**DIFF(Y)**

```
<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>-5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>-5</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-5</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>-5</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-5</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>-5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>-5</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-5</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>-5</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-5</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

**DIFF(X)**

```
<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-5</td>
<td>-5</td>
<td>-5</td>
<td>-5</td>
<td>-5</td>
<td>-5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>5</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>5</td>
<td>-5</td>
<td>-5</td>
<td>-5</td>
<td>-5</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>5</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

97
GRAD/THRESHOLD RESULT:

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

CHAIN-CODED IMAGE :

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 3 | 3 | 3 | 7 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 5 | 0 | 0 |
| 0 | 0 | 5 | 0 | 0 | 0 | 0 | 8 | 0 | 0 |
| 0 | 0 | 0 | 3 | 3 | 7 | 4 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 5 | 0 | 0 | 0 | 0 | 5 | 0 | 0 |
| 0 | 0 | 5 | 0 | 0 | 0 | 0 | 5 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
THE NUMERICALLY ENCODED PICTURE:

-The number 8 with squared corners.

```
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 10 10 10 10 0 0 0
0 0 10 0 0 0 0 10 0 0
0 0 10 0 0 0 10 0 0 0
0 0 0 10 10 10 10 0 0 0
0 0 10 0 0 0 0 10 0 0
0 0 0 10 10 10 10 0 0 0
0 0 0 0 0 0 0 0 0
```

**DIFF(Y)**

```
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 -5 5 5 5 10 0 0 0
0 -5 10 0 0 0 -5 10 0 0
0 -5 10 0 0 0 -5 10 0 0
0 0 -5 5 5 5 10 0 0 0
0 -5 10 0 0 0 -5 10 0 0
0 -5 10 0 0 0 -5 10 0 0
0 0 -5 5 5 5 10 0 0 0
0 0 0 0 0 0 0 0 0
```

**DIFF(X)**

```
0 0 0 0 0 0 0 0 0
0 0 0 -5 -5 -5 -5 0 0 0
0 0 -5 10 10 10 10 -5 0 0
0 0 5 0 0 0 0 5 0 0
0 0 10 -5 -5 -5 -5 10 0 0
0 0 -5 10 10 10 10 -5 0 0
0 0 5 0 0 0 0 5 0 0
0 0 10 -5 -5 -5 -5 10 0 0
0 0 0 10 10 10 10 0 0 0
```
### GRAD/THRESHOLD RESULT:

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 0 0 0
0 0 1 0 0 0 0 1 0 0
0 0 1 0 0 0 0 1 0 0
0 0 0 1 1 1 1 0 0 0
0 0 1 0 0 0 0 1 0 0
0 0 1 0 0 0 0 1 0 0
0 0 0 1 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0
```

### CHAIN-CODED IMAGE:

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 3 3 7 2 0 0 0
0 0 1 0 0 0 0 1 0 0
0 0 2 0 0 0 0 8 0 0
0 0 0 3 3 7 4 0 0 0
0 0 1 0 0 0 0 1 0 0
0 0 2 0 0 0 0 8 0 0
0 0 0 3 3 7 4 0 0 0
0 0 0 0 0 0 0 0 0 0
```
THE NUMERICALLY ENCODED PICTURE:

- The letter B.

\[
\begin{array}{cccccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 10 & 10 & 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 \\
0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 \\
0 & 0 & 10 & 10 & 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 \\
0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 \\
0 & 0 & 10 & 10 & 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[\text{DIFF}(Y)\]

\[
\begin{array}{cccccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -5 & 5 & 5 & 5 & 5 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -5 & 10 & 0 & 0 & 0 & -5 & 10 & 0 & 0 & 0 & 0 & 0 \\
0 & -5 & 10 & 0 & 0 & 0 & -5 & 10 & 0 & 0 & 0 & 0 & 0 \\
0 & -5 & 5 & 5 & 5 & 5 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -5 & 10 & 0 & 0 & 0 & -5 & 10 & 0 & 0 & 0 & 0 & 0 \\
0 & -5 & 10 & 0 & 0 & 0 & -5 & 10 & 0 & 0 & 0 & 0 & 0 \\
0 & -5 & 5 & 5 & 5 & 5 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[\text{DIFF}(X)\]

\[
\begin{array}{cccccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -5 & -5 & -5 & -5 & -5 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 5 & 10 & 10 & 10 & 10 & -5 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 5 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 5 & -5 & -5 & -5 & -5 & 10 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 5 & 10 & 10 & 10 & 10 & -5 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 5 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 5 & -5 & -5 & -5 & -5 & 10 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 10 & 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]
GRAD/THRESHOLD RESULT:

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

CHAIN-CODED IMAGE:

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
THE NUMERICALLY ENCODED PICTURE:

- The letter A.

\[
\begin{array}{cccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 10 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 10 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 \\
0 & 0 & 10 & 10 & 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\
0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 \\
0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[\text{DIFF}(Y)\]

\[
\begin{array}{cccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -5 & 5 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -5 & 10 & 0 & -5 & 10 & 0 & 0 & 0 & 0 & 0 \\
0 & -5 & 10 & 0 & 0 & 0 & -5 & 10 & 0 & 0 & 0 & 0 \\
0 & -5 & 5 & 5 & 5 & 5 & 5 & 10 & 0 & 0 & 0 & 0 \\
0 & -5 & 10 & 0 & 0 & 0 & -5 & 10 & 0 & 0 & 0 & 0 \\
0 & -5 & 10 & 0 & 0 & 0 & -5 & 10 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[\text{DIFF}(X)\]

\[
\begin{array}{cccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -5 & -5 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -5 & 10 & 0 & 0 & 10 & -5 & 0 & 0 & 0 & 0 \\
0 & 0 & 5 & -5 & -5 & -5 & -5 & 5 & 0 & 0 & 0 & 0 \\
0 & 0 & 5 & 10 & 10 & 10 & 10 & 5 & 0 & 0 & 0 & 0 \\
0 & 0 & 5 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\
0 & 0 & 5 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\
0 & 0 & 5 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\
0 & 0 & 5 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\
0 & 0 & 5 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\
\end{array}
\]

103
### GRAD/THRESHOLD RESULT:

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0 0 0
0 0 1 0 0 1 0 0 0 0
0 0 0 1 1 1 1 1 0 0
0 0 1 0 0 0 1 0 0 0
0 0 1 0 0 0 1 0 0 0
0 0 1 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0
```

### CHAIN-CODED IMAGE:

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 3 2 0 0 0 0
0 0 0 4 0 6 0 0 0 0
0 0 1 0 0 0 1 0 0 0
0 0 0 3 3 7 1 0 0 0
0 0 1 0 0 0 1 0 0 0
0 0 5 0 0 0 1 0 0 0
0 0 5 0 0 0 5 0 0 0
0 0 0 0 0 0 0 0 0 0
```
THE NUMERICALLY CODED PICTURE:

- A diamond shaped figure.

```
0 0 0 0 0 0 0 0 0
0 0 0 0 10 10 0 0 0
0 0 0 10 0 0 10 0 0
0 0 10 0 0 0 0 10 0
0 10 0 0 0 0 0 0 10
0 0 10 0 0 0 0 10 0
0 0 0 10 0 0 10 0 0
0 0 0 0 10 10 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
```

```
DIFF(Y)
```

```
0 0 0 0 0 0 0 0 0
0 0 0 -5 5 10 0 0 0
0 0 -5 10 0 -5 10 0 0
0 -5 10 0 0 0 -5 10 0
-5 10 0 0 0 0 -5 10 0
0 -5 10 0 0 0 -5 10 0
0 0 -5 10 0 -5 10 0 0
0 0 0 -5 5 10 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
```

```
DIFF(X)
```

```
0 0 0 0 -5 -5 0 0 0
0 0 0 -5 10 10 -5 0 0
0 0 -5 10 0 0 10 -5 0
0 -5 10 0 0 0 0 10 -5
0 10 -5 0 0 0 0 -5 10
0 0 10 -5 0 0 -5 10 0
0 0 0 10 -5 -5 10 0 0
0 0 0 0 10 10 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
```
### GRAD/THRESHOLD RESULT:

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### CHAIN-CODE IMAGE:

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 3 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 8 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 6 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 3 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

106
REFERENCES


APPENDIX III.
Minimal Finite Automata from Finite Training Sets

Department of Mathematical and Computer Sciences
The Atlanta University
Atlanta, GA 30314

Introduction

This paper describes a solution to the following problem: Given a finite set of strings over an alphabet $A$ and a positive integer $n$, find a deterministic finite automaton (DFA) with a minimal number of states that recognizes the strings in the given set but does not recognize any other strings of length less than $n$. Clearly such an automaton exists, indeed several minimal automata exist for some combinations of $A$ and $n$. The chief difficulty in solving this problem was determining a structure that is uniquely determined by the problem data.

This paper begins by expounding standard results on regular sets in a form clearly parallel to the new construction. The construction has been implemented in a version of LISP. Examples mentioned in this paper were derived with this implementation.

Notation

$A$ ....... a finite alphabet
$A^*$ ....... the set of strings over $A$
Every regular set is associated with a unique set of regular sets, from which the minimal automaton recognizing that regular set can be easily recovered.

Definition 1: A pointed set of regular sets (PSRS) is a set $(F_0 : F_1, \ldots, F_n)$ of subsets $F_i$ of $A^*$. $F_0$ is the distinguished member of the PSRS. A PSRS must satisfy this condition:

For all $i, 0 \leq i \leq n$, and all $a$ in $A$, there is a $j, 0 \leq j \leq n$ such that $\text{del}(F_i, a) = F_j$.

Definition 2: A PSRS $(F_0 : F_1, \ldots, F_n)$ is connected if the following condition holds:

For all $j, 0 \leq j \leq n$, there is a string $s$ in $A^*$ such that $\text{del}(F_0, s) = F_j$.

Lemma 3: Given a PSRS $(F_0 : F_1, \ldots, F_n)$, let $A = \{a_1, \ldots, a_m\}$. Given $i, j, 0 \leq i \leq n$, $0 \leq j \leq m$ let $\text{del}(F_i, a_j) = F_{d(i,j)}$. Then $F_i = F_1 + \ldots + F_m F_{d(i,m)}$.

Proof: Omitted. //

Lemma 4: Let $F' = (F_0 : F_1, \ldots, F_n)$ and $G' = (G_0 : G_1, \ldots, G_m)$
be PSRS's. Suppose also that $F_0 = G_0$. Then:

(a) if $F'$ is connected, then $F'$ is a subset of $G'$.

(b) if $F'$ and $G'$ are both connected then $F' = G'$.

Proof: (a) $F_0 \subseteq G'$ by hypothesis, suppose that for some $p \geq 1$ that the following is true:

(*) For any $s \in A^*$, length$(s) < p$, del$(F_0, s) \subseteq G'$.

Let $s \subseteq A^*$ be of length $p-1$ and $a \in A$. Then, by (*), $F_i = \text{del}(F_0, s)$ for some $i$ and some $j$. Then $\text{del}(F_0, sa) = \text{del}(\text{del}(F_0, s), a) = \text{del}(G_j, a) = G_k$ for some $k$. Thus, (*) is true even if length$(s) = p$. By induction, $\text{del}(F_0, s) \subseteq G'$ for any $s \in A^*$. Since $F'$ is connected, $F' \subseteq G'$.

(b) Clear from (a).

Lemma 5: Let $F'' = (F_0 : F_1, \ldots, F_r, F_{r+1}, \ldots, F_n)$ be a PSRS. Suppose that for each $1 \leq i \leq r$ there is an $s \in A^*$ such that $F_i = \text{del}(F_0, s)$ and for $r+1 \leq i \leq n$ there is no such $s$. Then $F'' = (F_0 : F_1, \ldots, F_n)$ is a connected PSRS.

Proof: Suppose that $\text{del}(F_i, a) = F_j$ for some $0 \leq i \leq r$ and $r+1 \leq j \leq n$. Then $F_j = \text{del}(\text{del}(F_0, s), a) = \text{del}(F_i, sa)$ for some $s \in A^*$, contradicting the hypothesis. Thus, $F''$ is a PSRS. $F''$ is clearly connected.

Note: A regular set is defined by a regular grammar. We use the definition of a regular grammar as a context-free grammar with productions of the form $X \rightarrow aY$ and $X \rightarrow \epsilon$, where $X$ and $Y$ are non-terminals, $a \in A$ and $\epsilon$ represents the empty string.

Theorem: Let $S$ be a regular subset of $A^*$. There is a unique
connected PSRS with $S$ its distinguished member.

Proof: Let $S$ be defined by a grammar over $A$ with non-terminals
$(X_0, \ldots, X_n)$ and start symbol $X_0$. For each $i$, let $R_i$ be the regular
set defined by the same grammar, but with start symbol $X_i$. So, $S = R_0$.
Let $F' = \{F_0, \ldots, F_m\}$ be the set of all unions of the sets $R_i$, numbered so that $S = R_0 = F_0$. The $F'$ is a PSRS. This is seen as
follows. For a given $X_i$ and $a \in A$, let $X_i \rightarrow aX_g(i,j)$ be all the
productions with left hand side $X_i$ and right hand side a followed by a
non-terminal. If there are no such productions then $\text{del}(R_i, a)$ is
empty, which is one of the unions $F_k$. Otherwise,
$\text{del}(R_i, a) = R_g(i,1) \cup \cdots \cup R_g(i,p)$, which is also one of the
$F_j$'s. Since $\text{del}(C \cup D, a) = \text{del}(C, a) \cup \text{del}(D, a)$, $F'$ is seen to
be a PSRS. A connected PSRS can be derived by Lemma 5. By Lemma 4, it
is unique. */

The Main Problem

Another way of stating the problem we solve here is this. Given a
finite set $S$ of strings over $A$ and a positive integer $n$, we wish to
determine a PSRS $(F_0; \ldots, F_m)$ with $m$ as small as possible, subject to
the condition that $F_0 \cap A(n) = S \cap A(n)$. As mentioned in the
introduction, $S$ and $n$ do not uniquely determine such a PSRS. However,
the following similar structure is uniquely determined, if one
condition is added. From it, all PSRS's that satisfy the conditions
can be easily determined.

Definition: $I$: A pointed set of finite sets (PSFS) is a finite set of
pairs \((C_0, n_0); (C_1, n_1), \ldots (C_m, n_m)\), with distinguished element \((C_0, n_0)\), satisfying the following conditions:

(a) \(n_i\) is an integer, \(C_i \subseteq A(n_i)\).

(b) For all \(0 \leq i \leq m\) and \(a \in A\), there is a \(j\) such that 
\[
\text{del}(C_i, a) = C_j \cap A(n_i - 1)\text{ and } n_j \geq n_i - 1.
\]

(c) For all \(0 < j \leq m\), there are \(i\) and \(a\) such that 
\[
\text{del}(C_i, a) = C_j\text{ and } n_j = n_i - 1.
\]

Lemma 9: For a PSFS \(((C_0, n_0); \ldots), n_i < n_0\) for all \(i > 0\).

Proof: Given a pair \((C_i, n_i)\) with \(i > 0\), we can build a chain of pairs, by successively applying Definition 7(c), having strictly increasing \(n_j\). As no pair may be repeated in this chain, it must be finite in length. Therefore, it must end with the pair \((C_0, n_0)\). //

Condition (c) is parallel to the connected condition on PSRS's. However, a further condition is needed to fully minimize a PSRS, given its distinguished pair.

Definition 8: A PSFS \(((C_0, n_0); \ldots)\) is minimal if for every \(i\) and \(j\), \(C_i \cap A(n_j) = C_j \cap A(n_i)\) implies \(i = j\).

Theorem 10: Suppose that \(C'\) and \(D'\) are PSFS's with the same distinguished pair.

(a) If \(C'\) is minimal, then \(C' \leq D'\).

(b) If both \(C'\) and \(D'\) are minimal, then \(C' = D'\).

Proof: Let \(C' = ((C_0, n_0); \ldots)\) and \(D' = ((D_0, m_0); \ldots)\). Suppose for some \(i < n_0\) the following is true:

\((*)\) For every \((C_i, n_i)\) such that \(n_i = p\), there is a \((D_k, m_k)\) such
that $C_i = A(n_i) \cap D_k$ and $m_k \geq n_i$.

Note that since $C_0 = D_0$ and $n_0 = m_0$ by hypothesis, (*) is true for $p = n_0 - 1$. Let $(C_j, n_j)$ be given with $n_j = p$. Choose $(C_i, n_i)$ and $a \in A$ such that $\text{del}(C_i, a) = C_j$ and $n_j = n_i - 1$. Now, $n_i > p$, so there is a pair $(D_k, m_k)$ as in (*). There is also a $(D_1, m_1)$ such that $\text{del}(D_k, m_k) = D_1 \land A(m_k - 1)$ and $m_1 \geq m_k - 1$. Then

$$D_1 \land A(n_j) = D_1 \land A(m_k - 1) \land A(n_j) = \text{del}(D_k, a) \land A(n_j)$$

since $n_j = n_i - 1 \leq m_k - 1$.

Thus

$$D_1 \land A(n_j) = \text{del}(D_k \land A(n_j + 1), a) = \text{del}(C_i, a) = C_j.$$  Further, $n_j \leq m_k - 1 \leq m_1$. So, if (*) is true for some $p < n_0$, it is true for $p - 1$. By the remark following (*), it is true for all $p < n_0$. Thus, for any $(C_i, n_i)$ there is a $(D_k, m_k)$ such that $C_i = D_k \land A(n_i)$ and $m_k \geq n_i$. Since the minimality of $C'$ has not been used, there is, by symmetry, a $(C_j, n_j)$ such that $D_k = C_j \land A(m_k)$ and $n_j \geq m_k$. Substituting for $D_k$, $C_j = C_j \land A(n_j) \land A(m_k)$ and $n_i \leq m_k \leq n_j$.

Thus, $C_i \land A(n_j) = C_i = C_j \land A(n_i)$. So, $i = j$ by the minimality of $C'$. This implies that $n_i = m_k = n_j$ and $D_k = C_i$. That is, every pair $(C_i, n_i)$ is in $D'$.

(b) From (a), by symmetry. //

As a PSFS can be reduced to a connected PSRS by removing some of its members, so a PSFS can be reduced to a minimal one. However, the removal must be done one pair at a time, as detailed in the following lemma.

Lemma 1: Let $C' = \{ (C_0, n_0); \ldots \}$ be a PSFS. Suppose there are one or more pairs $i \neq j$ such that $D_i \land A(n_i) = C_i \land A(n_j)$ and $n_j \leq n_i$.

Pick the pair with the smallest $n_j$. If $(C_j, n_j)$ is removed from $C'$,
the result is a PSFS with the same distinguished pair.

Proof: Call the new set of pairs $C''$. By Lemma 8, $j > 0$, so the distinguished pair would not be removed. Note that $C_j = C_i \cap A(n_j)$ and $n_j \geq n_k$. Condition (a) of Definition 7 is clearly satisfied by $C''$.

Suppose that, for some $k$ and $a$, $\text{del}(C_k, a) = C_j \cap A(n_j - 1)$ and $n_j \geq n_k - 1$. Then $\text{del}(C_k', a) = C_j \cap A(n_j - 1) = C_i \cap A(n_j) \cap A(n_k - 1) = C_i \cap A(n_k - 1)$ and $n_i \geq n_k - 1$. Clearly, then, 7(b) is satisfied by $C''$.

Suppose that for some $k$ and $a$, $\text{del}(C_j, a) = C_k$ and $n_k = n_j - 1$. Pick 1 so that $\text{del}(C_i, a) = C_1 \cap A(n_i - 1)$ and $n_1 \geq n_i - 1$. Now, $n_1 \geq n_i - 1 > n_j - 1 = n_k$, so $C_i \cap A(n_k) = C_1 \cap A(n_i - 1) \cap A(n_k) = \text{del}(C_i, a) \cap A(n_k) = \text{del}(C_j, a) = C_k$. However, this contradicts the minimality of $n_j$, so no such $k$ exists. Condition (c) of Definition 7 is thus satisfied by $C''$, it is a PSFS.

Corollary 10: Any PSFS contains a (unique) minimal PSFS, with the same distinguished pair, as a subset.

Proof: Repeated applications of Lemma 11 will result in the condition of Lemma 8 being satisfied. Unicity follows from Theorem 10(b).

We can now describe an algorithm for constructing the minimal PSFS with given distinguished pair $(C, n)$, $C \subseteq A(n)$. This algorithm maintains two lists of pairs: PROC and UNPROC. Initially, PROC is empty and UNPROC contains only the initial pair $(C, n)$. The algorithm is this:
while UNPROC is not empty do
    Fetch (D, m) from UNPROC and put it in PROC
    for each a ∈ A do
        if there is no pair (E, k) in PROC or UNPROC that
        satisfies \( \text{del}(D, a) = E \cap A(n-1) \) and
        \( k \leq n-1 \) then
            add the pair (\( \text{del}(D, a) \), m-1) to UNPROC
    Reduce PROC by using Lemma 11.

The final value of PROC, with \((C, n)\) as distinguished pair, is
the desired result.

We will show later that the next to the last step (reducing PROC)
is unnecessary if UNPROC is treated as a queue. To show that the
algorithm actually terminates, we must examine a pair, call it \((F, 1)\)
= (\( \text{del}(F, a) \), m-1), added to UNPROC. Assuming \( D \subseteq A(m) \), then\n\( F \subseteq A(1) \) since the strings in \( F \) are one shorter than those in \( D \).
Since UNPROC begins with just \((C, n)\) and \( C \subseteq A(n), F \subseteq A(1) \) indeed.
Suppose that \( D \) were empty. Then \( \text{del}(D, a) = D \cap A(n-1) \) and \( n \geq n-1 \)
so no new pair would need to be added. If \( C \) itself is empty, then the
algorithm only produces one pair, the original \((C, n)\). Otherwise, if \( F \)
is empty, we may assume that \( D \) is not empty and so \( m \geq -1 \), that is\( 1 \geq -1 \). This shows that the integer part of pairs added to UNPROC are
bounded below (by -1 or by n). The integer parts are clearly bounded
above by n. Since the strings in any pair are shorter than the longest
string in \( F \), there are only finitely many pairs that could be added to
UNPROC. The algorithm must eventually terminate.

The algorithm assures that the final set of pairs satisfies \( \gamma(b) \),
by construction. Condition (c) of Definition 7 is satisfied by the manner in which new pairs are introduced.