

A STRUCTURAL MODEL OF SHAPE\*

by

Linda G. Shapiro  
Department of Computer Science  
Virginia Polytechnic Institute  
Blacksburg, VA 24061

Technical Report #CS79003-R

October 1978  
revised April 1979

\*This work was supported by NSF Grant Number MCS77-23945

A STRUCTURAL MODEL OF SHAPE\*

Linda G. Shapiro  
Department of Computer Science  
Virginia Polytechnic Institute  
Blacksburg, VA 24061

October 1978  
revised April 1979

\*This work was supported by NSF Grant Number MCS77-23945

## TABLE OF CONTENTS

|   | <u>Page</u> |
|---|-------------|
| ABSTRACT . . . . .  | i           |
| KEY WORDS . . . . .   | i           |
| I. INTRODUCTION . . . . .   | 1           |
| II. RELATED WORK . . . . .  | 3           |
| III. THE MODEL . . . . .  | 8           |
| The General Model and Some Examples . . . . .   | 8           |
| A Structural Model Based on Simple Parts and Intrusions . . . . .                             | 13          |
| Example . . . . .   | 16          |
| IV. SHAPE MATCHING . . . . .  | 19          |
| Mappings on Weighted Relations . . . . .  | 23          |
| Mappings That Can Assign Several Candidate Primitives to One<br>Prototype Primitive . . . . . | 24          |
| Measuring the Goodness of a Match . . . . .   | 24          |
| The Shape Matching Procedure . . . . .  | 25          |
| V. EXPERIMENTAL RESULTS . . . . .   | 28          |
| Example of Tests with Equal Weights, Fixed Mergeability, and $\epsilon = 0$ . . . . .         | 32          |
| Examples of Tests Where $\epsilon$ Was Varied . . . . .                                       | 32          |
| Examples of Tests Where M Was Varied . . . . .  | 36          |
| Examples of Tests Where Weights Were Varied . . . . .   | 41          |
| VI. DISCUSSION AND CONCLUSIONS . . . . .  | 45          |
| REFERENCES . . . . .  | 48          |

## LIST OF FIGURE CAPTIONS

|           |   | <u>Page</u> |
|-----------|---|-------------|
| Figure 1  | illustrates the decomposition of a shape into simple parts and intrusions.  | 7           |
| Figure 2  | illustrates a set of rectangular shape primitives.  | 11          |
| Figure 3  | illustrates a shape description for a block letter E. The primitives are directed, oriented rectangles, and the relation C consists of triples of the form (primitive 1, primitive 2, operation) where the operator determines the way in which the two primitives connect. The idea of directed primitives and the operators + (head-to-tail connection) and x (tail-to-tail) connection are from the work of Shaw [32]. | 12          |
| Figure 4  | illustrates a second shape description for a block letter E. The primitives, which are line segments, have one attribute-- their relative length. The relation J consists of triples of the form (segment1, segment2, angle) where segment1 and segment2 join and angle is the angle between them. For the purpose of calculating the angles, the line segments are considered to be directed clockwise.                  | 14          |
| Figure 5  | illustrates the protrusion relation $R_p$ and the intrusion relation $R_i$ of a block letter 'E'.   | 17          |
| Figure 6  | illustrates decompositions of 5 different block letters and the $R_p$ and $R_i$ relations for each letter.  | 18          |
| Figure 7  | illustrates the decomposition of three shapes into simple parts and intrusions. Shape E1 was hand-printed and is considered a prototype letter 'E'. Shapes E2 and E3 were produced by adding noise to shape E1.   | 21          |
| Figure 8  | illustrates the prototype shapes and a set of corresponding imperfect versions.   | 30          |
| Figure 9  | illustrates the tree search to find all homomorphisms from letter E1 of Figure 7 to letter E2.  | 33          |
| Figure 10 | illustrates the simple parts and intrusions for the prototype shape I1 and the imperfect shape I3.  | 34          |
| Figure 11 | illustrates the tree search for two non-matches.  | 35          |
| Figure 12 | illustrates an inexact match of I1 to a portion of E2 at $\epsilon = .5$ .  | 37          |
| Figure 13 | illustrates the relations used in an inexact matching experiment where no homomorphisms were found for $\epsilon = 0, .05, \text{ or } .25$ .   | 38          |
| Figure 14 | illustrates the relations and tree searches for an inexact matching experiment where two homomorphisms were found for $\epsilon = .25$ .  | 39          |

LIST OF FIGURE CAPTIONS (continued)

|  | <u>Page</u> |
|--|-------------|
| Figure 15 illustrates a matching experiment on letter C's where weights were varied. The functions $h_1$ and $h_2$ are two of the .2-homomorphisms found when the unequal weights were used. | 42          |
| Figure 16 illustrates a matching experiment on two letter T's where weights were varied. The function $h$ is a .2-homomorphism found when the unequal weights were used.                     | 44          |

## ABSTRACT

Shape description and recognition is an important and interesting problem in scene analysis. Our approach to shape description is a formal model of a shape consisting of a set of primitives, their properties, and their interrelationships. The primitives are the simple parts and intrusions of the shape which can be derived through the graph-theoretic clustering procedure described in [31]. The interrelationships are two ternary relations on the primitives: the intrusion relation which relates two simple parts that join to the intrusion they surround and the protrusion relation which relates two intrusions to the protrusion between them. Using this model, a shape matching procedure that uses a tree search with look-ahead to find mappings from a prototype shape to a candidate shape has been developed. An experimental SNOBOL4 implementation has been used to test the program on hand-printed character data with favorable results.

## KEY WORDS

decomposition, matching, relational description, relaxation, shape description, shape recognition, tree search.

## I. INTRODUCTION

Shape description and characterization is an important problem in scene analysis. There have been many different analytic methods used in shape analysis including transformations and decompositions of the boundary of the shape, and partitions of the interior of the shape into simple pieces. Shape descriptions have included numeric values, feature vectors, character strings, trees, and graphs. The major problem seems to be the lack of a mathematical model for characterizing a shape.

Our approach to shape description is to construct a topological model of a shape which consists of a set of primitives, their properties, and their interrelationships. The primitives we use are the simple parts (near-convex pieces) and intrusions of the shape which can be derived through a decomposition procedure described in a previous paper [31]. The interrelationships are two ternary relations on the primitives: the intrusion relation and the protrusion relation. The intrusion relation is a set of triples of the form  $(s_1, i, s_2)$  where the two simple parts  $s_1$  and  $s_2$  touch or nearly touch and form part of the boundary of intrusion  $i$ . The protrusion relation is a set of triples of the form  $(i_1, s, i_2)$  where simple part  $s$  protrudes between the two intrusions  $i_1$  and  $i_2$ . These two relations seem to satisfactorily characterize a shape in a manner that agrees with human intuition on familiar shapes such as hand-printed characters. Our topological approach to shape matching is that one shape matches another if there is a mapping from the primitives of the first shape to the primitives of the second shape that preserves the interrelationships. In this approach, the metric and statistical properties of the shape and its parts are suppressed.

In this paper, we define a formal shape model and a shape matching procedure that uses a tree search with look-ahead to find mappings from a prototype shape to a candidate shape. Section II discusses related work, Section III describes

a general structural model for shape and defines our particular relational model, Section IV discusses shape matching, and Section V describes an experimental shape matching system.



## II. RELATED WORK

There have been several different approaches to the shape recognition problem. (For a complete survey, see Pavlidis [26]). On the statistical side shapes have been described by such features as area, perimeter, moments, and coefficients of Fourier series. Description by moments includes the work of Alt [1] and Hu [19] among others. Description using coefficients of Fourier series includes the work of Zahn and Roskies [35], Richard and Hemami [28], Persoon and Fu [27], and Granlund [13]. Agrawala and Kulkarni [2] present a sequential one-pass algorithm for extracting simple shape features such as perimeter, area, and moments. All of these approaches work with a discrete representation of the boundary of the shape.

Much of the recent work has been structural in nature. Blum [4] suggests the application of the medial axis transformation which transforms a shape into a line drawing representing its "skeleton". The skeleton can be used to derive properties of the shape and, together with information about the distance of the boundary from the skeleton, allows reconstruction of the shape. A second structural approach involves the decomposition of the boundary of the shape into a sequence of line segments. Work in this area includes the Freeman chain code [11], the analysis of convex blobs by computing dominant points [20,23,29,30], and Davis' work with angles, sides, and symmetry [6,7,8].

Also in the structural domain is the work on syntactic shape recognition. This includes Pavlidis and Ali's syntactic analysis of shapes [24] using regular expressions over the terminal symbols QUAD (arcs that can be approximated by quadratic curves), TRUS (sharp protrusions or intrusions), LINE (long linear segments), and BREAK (short segments). Also using the syntactic approach are Horowitz [18] and Lozano-Perez [21], both of whom parse piecewise linear approximations of one-dimensional waveforms to determine the structure of the peaks

and valleys and Fu and Lu [12] who encode line patterns as strings and use error-correcting parsers to determine the distance from an input pattern to grammars describing each of the possible pattern classes. A clustering algorithm then determines which class the input pattern belongs to.

Another recent effort in syntactic shape recognition is the work of You and Fu [35] on attributed shape grammars. In an attributed shape grammar, each curve segment (primitive or nonterminal) is described by four attributes: the vector spanning its end points, its length, its angular change, and a symmetry measure. Grammar rules specify the generation of curve segments from smaller curve segments connected by angle primitives and are of the context free form  $N \rightarrow (XA)^*X$  where  $N$  is a nonterminal,  $X$  can be a primitive curve segment or a nonterminal, and  $A$  is an angle primitive. In this case, the description of the curve segment  $N$  can be obtained directly from the descriptions of the curve segments composing  $N$  and the angles between them. Two recognition algorithms have been developed which accomplish parsing and primitive extraction simultaneously, using high-level knowledge to drive low-level processes.

Syntactic pattern recognition has the main advantage of having well-defined often linear parsing algorithms. The main disadvantage of this approach to shape recognition is that the shape is usually first encoded as a one-dimensional string of symbols which is the input to the parser. The You and Fu work is an exception in that the parser directs the low-level processes that extract and encode primitives. However, the parsing is still inherently a one-dimensional process due to the nature of context free grammars. While the attributed shape grammar and associated recognition algorithms seem to be more powerful than previous shape grammars, they typically handle only adjacency relationships on the boundary of the shape.

It is not clear to us that all the information in a two-dimensional shape can be preserved in a one-dimensional representation. In particular, the intrusion and protrusion relations defined in this paper cannot be expressed naturally by one-dimensional grammar rules since they capture non-linear information about the shape.

The approaches that are most related to our work are those that attempt to decompose the whole shape into two-dimensional parts. This includes the work of Pavlidis [25], Feng and Pavlidis [10], Maruyama [22], and Eden [9]. In the Feng and Pavlidis work, a shape is decomposed into convex parts, T-shaped parts, and spirals and described by a labeled graph indicating connections between pairs of parts. Maruyama suggests a decomposition of shapes into angularly simple regions where each such region has at least one interior point that can "see" its entire boundary. Eden decomposes script characters into meaningful primitive strokes.

In [31] we presented an algorithm for the decomposition of a two-dimensional shape into simple parts. The input to this algorithm is an ordered sequence of  $(x,y)$  coordinates representing the vertices of a polygonal approximation to the boundary of a shape. From these points the interior line segment relation  $LI$  is computed.  $LI$  is a binary relation consisting of all pairs of vertices such that the straight line segment joining them lies wholly within the shape. A graph-theoretic clustering algorithm on the relation  $LI$  yields a set of clusters, each consisting of a subset of the original set of vertices. These clusters are the interior clusters or simple parts of the shape.

The decomposition algorithm has three important properties:

- (1) The algorithm is independent of the starting point on the boundary of the shape.
- (2) The relation  $LI$  is invariant under linear transformations and perspective transformations; thus a transformation of a shape yields the same decomposition as the original shape.

- (3) The clusters produced by the graph-theoretic clustering algorithm can be controlled by numerical parameters; they are not necessarily convex or even disjoint. Thus the definition of simple part remains flexible.

Figure 1a shows the decomposition of a hand-printed letter E by this method.

In [31] we defined the exterior line segment relation LE, consisting of all pairs of vertices such that the straight line joining them lies exterior to the shape, and suggested that LE would be useful in obtaining a structural description of the shape. The clusters of the LE relation, obtained with the same graph-theoretic clustering algorithm, correspond to intrusions into the boundary of the shape. Figure 1b shows these exterior clusters for the letter E of Figure 1a.

In Section III we will describe a structural model for a shape based on the simple parts, the intrusions, their properties, and their interrelationships.

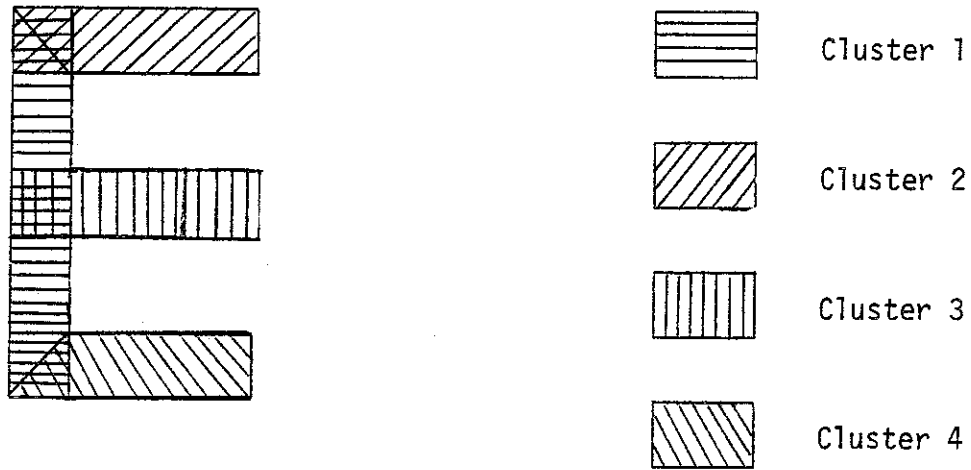


Figure 1a shows the simple parts or interior clusters of a hand-printed letter E.

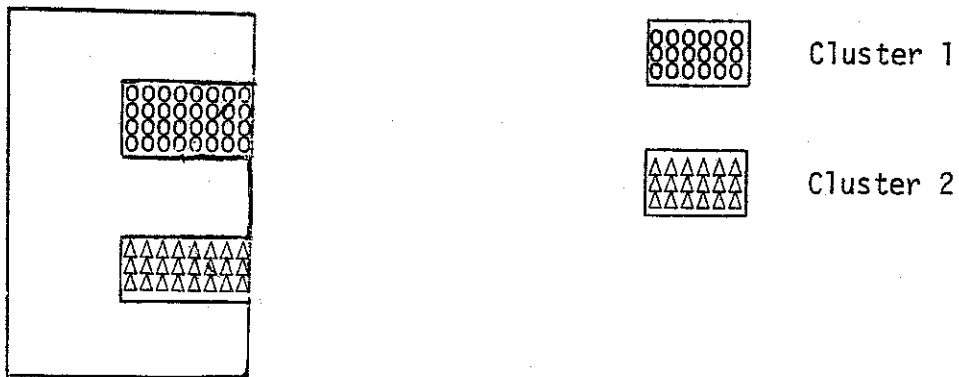


Figure 1b shows the intrusions or exterior clusters of the hand-printed letter E.

Figure 1 illustrates the decomposition of a shape into simple parts and intrusions.

### III. THE MODEL

A structural description of a shape must include a set of shape primitives, their properties, and their interrelationships. In this section we will define a general, formal, structural model for the description of two-dimensional shapes. We will then define a specific structural description that we have been using in our shape matching experiments.

#### The General Model and Some Examples

We define a shape description  $D$  to be a 5-tuple  $D = (N, T, G, P, R)$  where  $N$  is the name of the shape being described,  $T$  is the type of the shape,  $G$  is an attribute-value table containing the global attributes of the shape,  $P$  is a set of shape primitives obtained from a structural analysis of the shape, and  $R$  is a set  $\{R_1, \dots, R_k\}$  of relations on  $P$ . For each  $k$ , there is a positive integer  $N_k$  and a possibly singleton label set  $L_k$  such that  $R_k \subseteq P^{N_k} \times L_k$ ,  $k = 1, \dots, K$ . Thus, each  $R_k$  is a labeled  $N_k$ -ary relation. Haralick and Kartus [14] call  $R_k$  an "arrangement".

The name  $N$  and the type  $T$  are identifying information. The name might identify a region in an image being analyzed or it might be a character string that the experimenter wants to associate with a particular input file. The type identifies a class that the shape belongs to. For instance, if zip codes are being analyzed, the type might be "numeral". The purpose of the type is to limit the number of stored prototypes that are considered for a match. (See Section V.) Of course in the worst case, where the origin of the shape is completely unknown, no type information can be input.

A structural description of a shape includes a table of attribute-value pairs pertaining to the shape as a whole, a set of primitives that the shape can be broken down into, and a set of relations showing the interrelationships among the primitives. While the primitives  $P$  and the relations  $R$  define

the structure of complex shapes, we feel that it is also important to record any global attributes that are available. If the shape is associated with a region of an image, then it might be convenient to consider intensity or texture information as global attributes of the shape. The global attributes may also include statistical descriptors such as circularity, elongation, convexity, crenation (notched boundary), and so on ([17]). For simpler shapes, the global attributes may be more important than the primitives and relations, which may be missing altogether. The primitives and relations will differ with each particular shape model. We will illustrate with a few examples borrowed from the literature.

Shaw [32] defined a picture description language (PDL) in which pictures could be defined hierarchically in terms of primitives. Each primitive in PDL is a two-dimensional entity having two connection points, a head and a tail. The operators +, -, x, and \* indicate the head-to-tail, head-to-head, tail-to-tail, and head-to-head with tail-to-tail connection of two primitives. For example, the expression "A + B" indicates that the head of A is to be connected to the tail of B. The resultant entity has its tail at the tail of A and its head at the head of B and can be connected to other primitives or higher entities. PDL sentences are expressions consisting of primitives, operators, and parentheses indicating the order of applying the operators. For example, "A\*(B + C)" indicates that the head of B is connected to the tail of C to form a new entity. Then the new entity is connected to A, head-to-head and tail-to-tail.

We will give an example of a simple shape description using Shaw's operators, but not using the PDL expression concept. The primitives of our example shape are rectangles one unit long and one-half unit wide of various orientations. Figure 2 illustrates a set of such primitives. The primitives can be connected head-to-tail, head-to-head, and tail-to-tail. (Since they

are not curved, the head-to-head with tail-to-tail connection is not possible). We will assume that for each pair of primitives, the method for joining them in each of the possible connections has been specified. We can describe certain simple shapes such as block letters in terms of these primitives and their pairwise connections.

A description for a block letter would be a 5-tuple  $D1 = (N, T, G, P, R)$  where  $N$  is the name of the letter,  $T$  is the type "block letter",  $G$  contains global attributes of the letter such as the number of primitives,  $P$  is the subset of the primitives of Figure 2 that the letter is composed of, and  $R$  contains a single labeled binary connection relation  $C \subseteq \{(p_1, p_2, 0) \mid p_1, p_2 \in S, 0 \in \{+, -, x\}\}$ . Figure 3 illustrates this type of description for the block letter E.



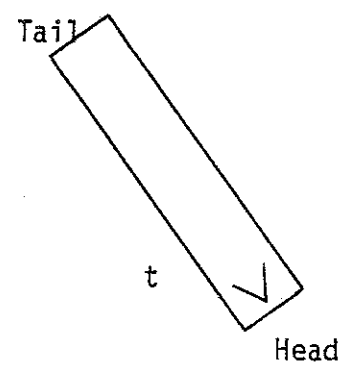
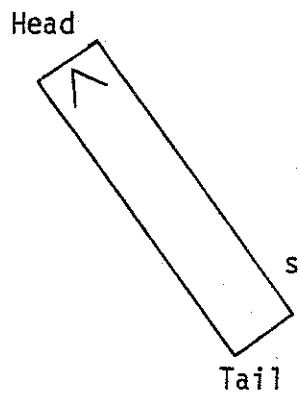
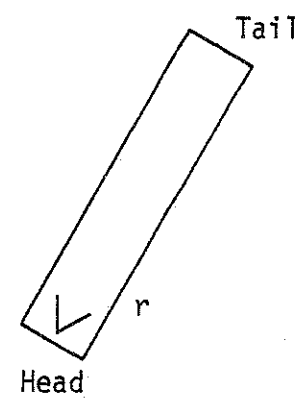
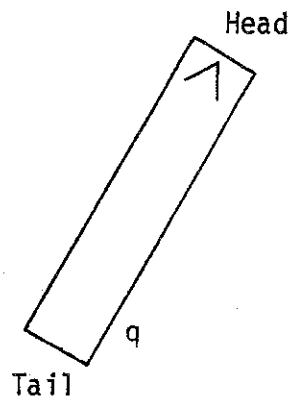
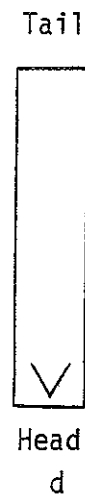
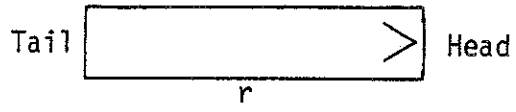
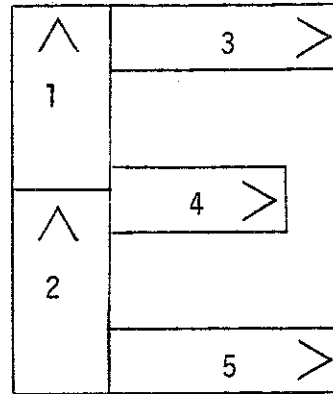


Figure 2 illustrates a set of rectangular shape primitives.



$D1 = ('E', \text{block letter}, G, P, \{C\})$

$G = \{(\text{number-of-primitives}, 5)\}$

$P = \{1, 2, 3, 4, 5\}$

$C = \{(2, 1, +)(1, 3, +)(1, 4, x)(2, 4, +)(2, 5, x)\}$

$1 = u$

$2 = u$

$3 = r$

$4 = r$

$5 = r$

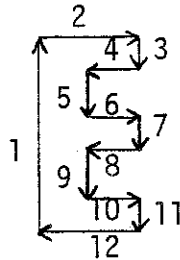
Figure 3 illustrates a shape description for a block letter E. The primitives are directed, oriented rectangles, and the relation C consists of triples of the form (primitive 1, primitive 2, operation) where the operator determines the way in which the two parameters connect. The idea of directed primitives and the operators + (head-to-tail connection) and x (tail-to-tail) connection are from the work of Shaw [32].

Another approach to shape description is in terms of an ordered sequence of line segments comprising the boundary of the shape. The relative lengths of each line segment and the relative angles between successive segments complete the model. A description for a block letter in this system would be a 5-tuple  $D2 = (N, T, G, P, R)$  where  $N$ ,  $T$ , and  $G$  are as above,  $P$  is a set of line segments whose attributes include length, and  $R$  contains the labeled binary relation  $J = \{(l_1, l_2, a) \mid l_1, l_2 \in P, l_1 \text{ connects to } l_2 \text{ and } a \text{ is the angle between } l_1 \text{ and } l_2\}$ . Figure 4 illustrates this type of description for the block letter E.

Both description D1 and D2 are size invariant since all lengths specified are also relative. In description D1, the primitives are defined according to their orientation. Thus, the structural descriptions are not rotation invariant. This sort of description can work very well in applications like character recognition where the text is assumed to be right side up, but descriptions using orientations cannot handle a general shape recognition problem. Description D2 is rotation invariant since the only orientation information is the set of relative angles between adjacent pairs of line segments. A description like D2 which concentrates on the boundary of the shape is useful in shape matching problems where the boundary is of primary importance. Map data is an example of this class of shapes.

#### A Structural Model Based on Simple Parts and Intrusions

One aspect of shape is its size and orientation invariance. This is the aspect that we attempt to capture with the structural model presented here. Let  $S = \{s_1, \dots, s_n\}$  be the set of simple parts of a shape and let  $I = \{i_1, \dots, i_m\}$  be the set of intrusions.  $S$  and  $I$  are assumed to be obtained from the shape by a decomposition procedure such as the one described in [31]. We will treat each simple part and each intrusion as a two-dimensional region having area and a



D2 = ('E', block letter, G, P, {J})

G = {(number-of-segments, 12)}

P = {(1, 2, 3, 4, ..., 12)}

J = {(1, 2, -45), (2, 3, -45), (3, 4, -45)  
 (4, 5, 45), (5, 6, 45), (6, 7, -45)  
 (7, 8, -45), (8, 9, 45), (9, 10, 45)  
 (10, 11, -45), (11, 12, -45), (12, 1, -45)}

|             |            |            |              |
|-------------|------------|------------|--------------|
| 1 = 4 units | 4 = 1 unit | 7 = 1 unit | 10 = 1 unit  |
| 2 = 2 units | 5 = ½ unit | 8 = 1 unit | 11 = 1 unit  |
| 3 = 1 unit  | 6 = 1 unit | 9 = ½ unit | 12 = 2 units |

Figure 4 illustrates a second shape description for a block letter E. The primitives, which are line segments, have one attribute-- their relative length. The relation J consists of triples of the form (segment1, segment2, angle) where segment1 and segment2 join and angle is the angle between them. For the purpose of calculating the angles, the line segments are considered to be directed clockwise.

closed boundary. Let  $P = S \cup I$ .  $P$  consists of the simple parts and intrusions. Let  $\delta: P \times P \rightarrow [0, \infty)$  be a distance function which gives a relative measure of the distance between two simple parts, two intrusions, or a simple part and an intrusion with the property that if  $p_1$  and  $p_2$  touch or overlap, then  $\delta(p_1, p_2) = 0$ . Let  $\Delta$  be a non-negative real number. Define the Boolean function  $p: P^3 \rightarrow \{\text{true}, \text{false}\}$  by

$$p(p_1, p_2, p_3) = \begin{cases} \text{true if } \delta(p_1, p_2) \leq \Delta, \\ \quad \delta(p_2, p_3) \leq \Delta, \text{ and} \\ \text{it is possible to draw a straight line from} \\ \text{a boundary point of } p_1 \text{ through } p_2 \text{ to a} \\ \text{boundary point of } p_3 \\ \text{false otherwise} \end{cases}$$

Thus,  $p(p_1, p_2, p_3)$  is true if both  $p_1$  and  $p_3$  touch or almost touch  $p_2$  and a part of  $p_2$  lies between  $p_1$  and  $p_3$ . A shape is characterized by a 5-tuple  $M = (N, T, G, P, R)$  where  $N$  is the name of the shape,  $T$  is the type of the shape,  $G = \{(\text{number-of-simple-parts}, \#S), (\text{number-of-intrusions}, \#I)\}$ ,  $P = S \cup I$ , and  $R = \{R_p, R_i\}$ .  $R_p = \{(i_1, s, i_2) \in I \times S \times I \mid p(i_1, s, i_2) = \text{true}\}$  and  $R_i = \{(s_1, i, s_2) \in S \times I \times S \mid p(s_1, i, s_2) = \text{true and } \delta(s_1, s_2) \leq \Delta\}$ . Thus,  $R_p$  consists of triples of the form (intrusion 1, simple part, intrusion 2) where the simple part protrudes between the two intrusions, and  $R_i$  consists of triples of the form (simple part 1, intrusion, simple part 2) where the two simple parts touch or nearly touch and form part of the boundary of the intrusion. Note that although the relations  $R_i$  and  $R_p$  are currently defined using the predicate  $p$  which returns true or false, alternate definitions could be formulated where  $p$  would return a numeric value. So far no provision has been made for describing each simple part and each intrusion of a shape. The decomposition algorithm was designed to

produce near-convex primitives, and it does not seem useful to decompose these primitives any further. However, a list of the properties of each primitive would add to the completeness of the description of the entire shape. Some of the properties of a primitive that might be useful are listed below:

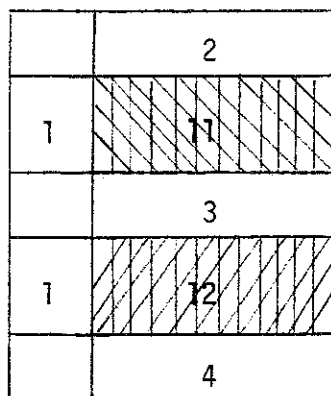
circularity  
 elongation  
 convexity  
 crenation  
 relative length  
 relative width  
 relative area

### Example

Figure 1 illustrates the decomposition of a block letter E into four simple parts and two intrusions. We will name the simple parts 1, 2, 3, and 4 corresponding to clusters 1, 2, 3, and 4 in Figure 1a and the intrusions 11 and 12 corresponding to clusters 1 and 2, respectively, in Figure 1b. Then the structural description of the letter E for any value of  $\Delta$  is

```
( 'E', letter, G, P, {Rp, Ri} )
G = {(number-of-simple-parts,4) (number-of-intrusions,2)}
P = {1,2,3,4,11,12}
Rp = {(11,3,12)}
Ri = {(1,11,2),
      (1,11,3),
      (1,12,3),
      (1,12,4)}
```

Figure 5 illustrates this symbolically and Figure 6 compares the decompositions of five block letters E, G, I, C, and B into simple parts and intrusions and the  $R_p$  and  $R_i$  relations for each letter. (Note that for brevity, we list only triples of the form  $(p_1, p_2, p_3)$ ,  $p_1 < p_3$  and omit their symmetric counterparts.)



$$R_p = \{(11,3,12)\}$$

$$R_i = \{(1,11,2), \\ (1,11,3), \\ (1,12,3), \\ (1,12,4)\}$$

Figure 5 illustrates the protrusion relation  $R_p$  and the intrusion relation  $R_i$  of a block letter 'E'.

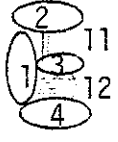
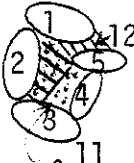
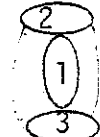
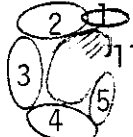

|       | E   | G   | I   | C  | B   |
|-------|---|---|---|--|---|
|       |  |  |  |  |  |
| $R_p$ | 11,3,12   | 11,5,12   | 11,1,12   | empty  | empty   |
| $R_i$ | 1,11,2  | 1,12,2  | 1,11,2  | 1,11,2   | 1,11,1  |
|       | 1,11,3  | 2,11,3  | 1,12,2  | 2,11,3   |   |
|       | 1,12,3  | 3,11,4  | 1,11,3  | 3,11,4   |   |
|       | 1,12,4  | 4,11,5  | 1,12,3  | 4,11,5   |   |

Figure 6 illustrates decompositions of 5 different block letters and the  $R_p$  and  $R_i$  relations for each letter.



#### IV. SHAPE MATCHING

For our purposes, two shapes match if there is a function that maps the primitives of the first shape to the primitives of the second in such a way that primitives map to like primitives and the interrelationships among the primitives are preserved. Such a structure preserving function is called a homomorphism. Haralick and Kartus [14] define a relational homomorphism as follows.

Let  $A$  be a finite set of objects, let  $L$  be a finite set of labels, let  $R \subseteq A^N \times L$  be a labeled  $N$ -ary relation, and let  $h:A \rightarrow B$  be a mapping from  $A$  to a second set  $B$ . The composition of the relation  $R$  with the function  $h$  is the labeled  $N$ -ary relation  $R \circ h$  defined by

$$R \circ h = \{(b_1, \dots, b_N, \ell) \in B^N \times L \mid \text{there exists } (a_1, \dots, a_N, \ell) \in R \\ \text{with } h(a_i) = b_i, i = 1, \dots, N\}$$

Suppose  $R \subseteq A^N \times L$  and  $R' \subseteq B^N \times L$ . A relational homomorphism from  $R$  to  $R'$  is a function  $h:A \rightarrow B$  such that  $R \circ h \subseteq R'$ .

Thus, the composition of an  $N$ -ary relation with a function produces a second  $N$ -ary relation. The relational homomorphism is a structure preserving function that maps a labeled  $N$ -ary relation onto a subset of a second labeled  $N$ -ary relation. If shapes are represented by labeled relations, then two shapes match if there is a relational homomorphism from one relation to the other.

The structural description of a shape defined in Section III includes two ternary relations:  $R_p$ , the protrusion relation, and  $R_i$ , the intrusion relation. In order to simplify our description of the shape matching process, we will combine  $R_i$  and  $R_p$  to form a labeled ternary relation  $R(R_i, R_p)$  defined by

$$R(R_i, R_p) = \{(p_1, p_2, p_3, \ell) \in P^3 \mid \text{either } (p_1, p_2, p_3) \in R_i \text{ and } \ell = i \text{ or } (p_1, p_2, p_3) \in R_p \text{ and } \ell = p\}.$$

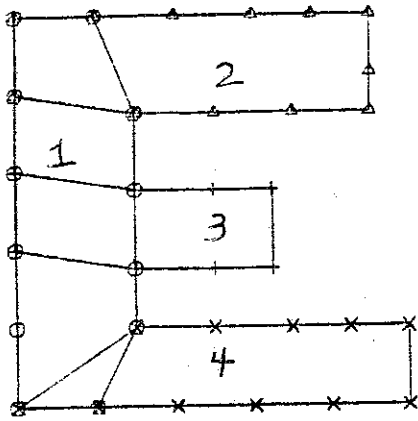
Each element of  $R(R_i, R_p)$  is a triple of either  $R_i$  or  $R_p$  with a label indicating whether it is from  $R_i$  or  $R_p$ .

The elements of  $R(R_i, R_p)$  contain information about the structure of the shape and indicate which primitives are simple parts and which are intrusions. One form of shape matching is to compare candidate shapes to a prototype shape and determine whether the candidate is similar enough to the prototype to be considered a match. Let  $S_1 = (N_1, T_1, G_1, S_1 \cup I_1, \{R_{p1}, R_{i1}\})$  be a prototype shape and  $S_2 = (N_2, T_2, G_2, S_2 \cup I_2, \{R_{p2}, R_{i2}\})$  be a candidate shape. Let  $R_1 = R(R_{i1}, R_{p1})$  and  $R_2 = R(R_{i2}, R_{p2})$ . The candidate shape  $S_2$  matches the prototype shape  $S_1$  if there is a relational homomorphism  $h$  from  $R_1$  to  $R_2$ .

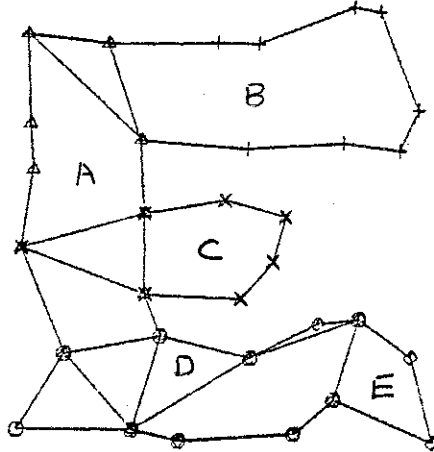
For example, consider the decomposition of shapes  $E_1$ ,  $E_2$ , and  $E_3$  in Figure 7. Suppose  $E_1$  is a prototype shape for the letter 'E' and  $E_2$  and  $E_3$  are candidate shapes. The relations  $R_1$ ,  $R_2$ , and  $R_3$  corresponding to  $E_1$ ,  $E_2$ , and  $E_3$  are given below. Very thin exterior clusters (13, 14, 15, EE, DD, FF, TT, UU, and VV in Figure 7) which correspond to almost straight parts of the boundary rather than true intrusions, have been eliminated from the descriptions. These "false intrusions" will be eliminated by the program that calculates  $R_i$  and  $R_p$ . They are currently produced by the algorithm used to find the LE relation and have not been eliminated earlier because we think they may prove useful in future shape analyses.

| $R_1$       | $R_2$       | $R_3$       |
|-------------|-------------|-------------|
| (1,11,2,i)  | (A,AA,B,i)  | (Q,QQ,R,i)  |
| (1,11,3,i)  | (A,AA,C,i)  | (Q,QQ,S,i)  |
| (1,12,3,i)  | (A,BB,C,i)  | (Q,RR,S,i)  |
| (1,12,4,i)  | (A,BB,D,i)  | (Q,RR,T,i)  |
| (11,3,12,p) | (A,BB,E,i)  | (T,SS,U,i)  |
|             | (D,BB,E,i)  | (QQ,S,RR,p) |
|             | (D,CC,E,i)  | (RR,U,SS,p) |
|             | (AA,C,BB,p) | (RR,T,SS,p) |
|             | (AA,C,CC,p) |             |

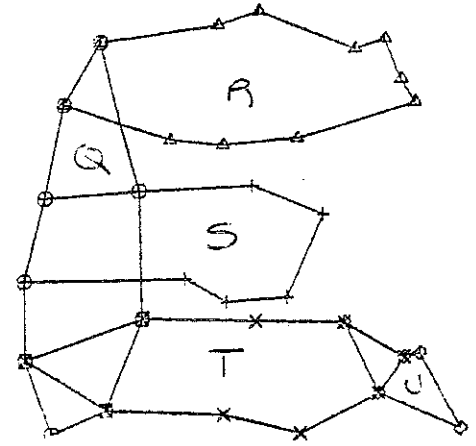
E1



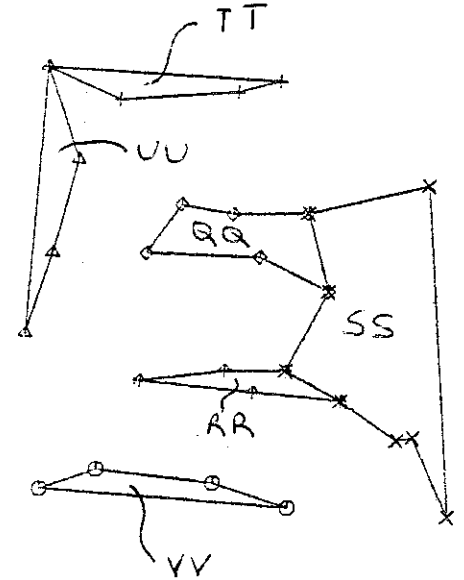
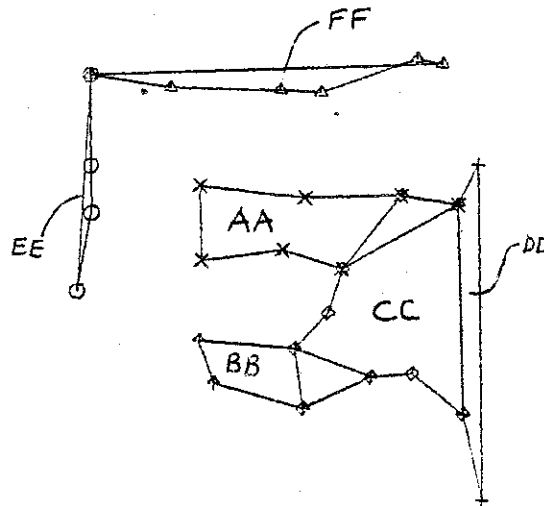
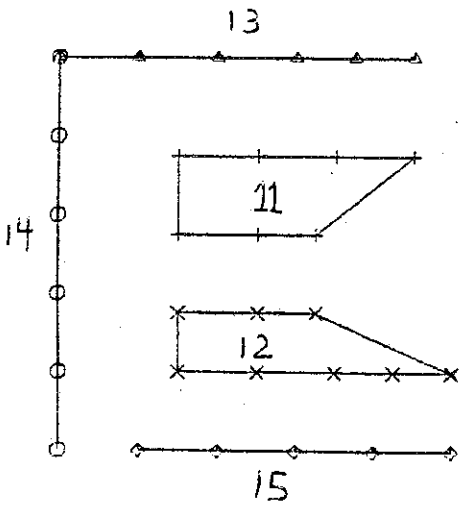
E2



E3



Decomposition into simple parts



Decomposition into intrusions

Figure 7 illustrates the decomposition of three shapes into simple parts and intrusions. Shape E1 was hand-printed and is considered a prototype letter 'E'. Shapes E2 and E3 were produced by adding noise to shape E1.

Now for  $E_2$  to match  $E_1$  there must be a homomorphism  $h_{12}:(S_1 \cup I_1) \rightarrow (S_2 \cup I_2)$  from  $R_1$  to  $R_2$ . Four such homomorphisms are given below.

| $p \in (S_1 \cup I_1)$ | $\langle 1 \rangle$<br>$h_{12}(p)$ | $\langle 2 \rangle$<br>$h_{12}(p)$ | $\langle 3 \rangle$<br>$h_{12}(p)$ | $\langle 4 \rangle$<br>$h_{12}(p)$ |
|------------------------|------------------------------------|------------------------------------|------------------------------------|------------------------------------|
| 1                      | A                                  | A                                  | A                                  | A                                  |
| 2                      | B                                  | B                                  | D                                  | E                                  |
| 3                      | C                                  | C                                  | C                                  | C                                  |
| 4                      | D                                  | E                                  | B                                  | B                                  |
| 11                     | AA                                 | AA                                 | BB                                 | BB                                 |
| 12                     | BB                                 | BB                                 | AA                                 | AA                                 |

For each  $i$ ,  $1 \leq i \leq 4$ , it is easy to show that  $R_1 \circ h_{12}^{\langle i \rangle} \subseteq R_2$ . For example,  $R_1 \circ h_{12}^{\langle 1 \rangle} = \{(A, AA, B, i), (A, AA, C, i), (A, BB, C, i), (A, BB, D, i), (AA, C, BB, p)\} \subseteq R_2$ . Similarly, there are homomorphisms  $h_{13}:(S_1 \cup I_1) \rightarrow (S_3 \cup I_3)$  from  $R_1$  to  $R_3$ . Thus, both candidate  $E_2$  and candidate  $E_3$  can be said to match prototype  $E_1$ . Notice that some of the homomorphisms from a model to a candidate shape really represent the mappings from the model to the mirror image of the candidate. In the above example,  $h_{12}^{\langle 3 \rangle}$  and  $h_{12}^{\langle 4 \rangle}$  are the mirror image mappings.

A shape matching procedure must find a homomorphism from a prototype shape to a candidate shape. Several important considerations pertain to the design of such a procedure:

(1) Some of the relationships in the prototype shape may be more important than others. Thus, it is desirable to assign weights to tuples in the description of the prototype.

(2) It is possible for two similar shapes to decompose into different numbers of primitives. Thus, the mapping should be able to assign the same primitives in the candidate to several primitives in the prototype if they

are physically close enough to each other. (If we choose to look for a binary relation instead of a mapping, then several primitives in the candidate could be associated with one primitive in the prototype).

(3) A homomorphism corresponds to a match, but does not necessarily indicate a good match. We need a measure of the goodness of a match. In the remainder of this section we will discuss each of these considerations in more detail.

### Mappings on Weighted Relations

Suppose  $R_1 \subseteq P_1^3 \times \{i,p\}$  is a prototype relation and  $R_2 \subseteq P_2^3 \times \{i,p\}$  is a candidate relation. Let  $W$  be the set  $[0,1]$  of real numbers between 0 and 1. A weighting function for  $R_1$  is a mapping  $w:R_1 \rightarrow W$  that assigns a weight to each triple of  $R_1$  such that  $\sum_{r \in R_1} w(r) = 1$ . Let  $h$  be a mapping

from  $P_1$  to  $P_2$  and  $w$  a weighting function for  $R_1$ . A triple  $r$  of  $R_1$  is satisfied by  $h$  with respect to  $R_2$  if  $h(r) \in R_2$ . If  $h$  is a homomorphism from  $R_1$  to  $R_2$ , then  $\sum_{\substack{r \in R_1 \\ [r \text{ not satisfied by } h]}} w(r) = 0$  since  $R_1 \circ h \subseteq R_2$ . In order

to allow imperfect matches, we define an  $\epsilon$ -homomorphism to be a mapping  $h:P_1 \rightarrow P_2$  such that  $\sum_{\substack{r \in R_1 \\ [r \text{ not satisfied by } h]}} w(r) \leq \epsilon$ . Thus, in an  $\epsilon$ -homomorphism

the sum of the weights of those triples that are not satisfied is not greater than  $\epsilon$ . A homomorphism, as defined earlier, is a 0-homomorphism.

Using these ideas, a human or a computer can assign weights to the triples of the prototype relation. One method of obtaining the weights would be to let a program induce them from imperfect matches ranked as to goodness of match. Then the shape matching problem is to find  $\epsilon$ -homomorphisms from the prototype to the candidate.

### Mappings That Can Assign Several Candidate Primitives to One Prototype Primitive

We would like to allow mappings that are not one-one, but with the restriction that only primitives that are near enough to each other can map to a single primitive. This leads to the mergeability relation  $M \subseteq (S_1 \times S_1) \cup (I_1 \times I_1)$ . If  $(p_1, p_2) \in M$  then  $p_1$  and  $p_2$  may map to the same candidate primitive. One possible definition for  $M$  is

$$M = \{(p_1, p_2) \mid \delta(p_1, p_2) \leq \Delta\}$$

A mapping  $h: P_1 \rightarrow P_2$  is consistent with  $M$  if  $h(p) = h(q)$  implies  $(p, q) \in M$ .

Thus, we can expand the shape matching problem to find all  $\epsilon$ -homomorphisms that are consistent with  $M$ .

### Measuring the Goodness of a Match

The quantity  $\sum_{r \in R_1} w(r)$  measures the relational goodness of a mapping  $h$ .  
[ $r$  satisfied by  $h$ ]

Two other measures are of interest:

- (1) A measure of the similarity between each primitive  $p$  of the prototype to the corresponding primitive  $h(p)$  of the candidate.
- (2) A measure of the size of the matched subset of the candidate primitives.

The measure (1) would take into account properties of a primitive such as those mentioned in Section III. The strictness or lenience of the enforcement of this measure in the matching criteria will determine how close to identical two shapes must be in order to match. The measure (2) might simply be the ratio of the area of the matched primitives (or just the simple parts) of the candidate to the total area of the primitives (or simple parts) of the candidate.

### The Shape Matching Procedure

The problem of finding homomorphisms has been with us for some time. (See Barrow, Ambler, and Burstall [3], Ullman [33], and Corneil and Gottlieb [5]). Finding homomorphisms can be accomplished by means of a tree search incorporating a "look-ahead" or "relaxation" operator. (See Haralick and Shapiro [15]). Finding  $\epsilon$ -homomorphism can be accomplished by a similar procedure. We now define a simple procedure for finding  $\epsilon$ -homomorphisms that are consistent with a mergeability relation. The following high-level algorithm describes the procedure for finding all  $\epsilon$ -homomorphisms that are consistent with mergeability relation  $M$  from a weighted  $N$ -ary relation  $R_1$  on a set  $U$  of primitives of a prototype to an  $N$ -ary relation  $R_2$  on a set  $L$  of primitives of a candidate. The primitives of  $U$  are referred to as units and the primitives of  $L$  as labels.

```

procedure MAIN ( );
comment LABELS is a table
          LABELS (u) contains a list of those elements
          of L that U may be mapped to by the homomorphism
          to be found;

read ( $\epsilon$ );
read (M);
read (U);
read (L);
read ( $R_1$ );
read ( $R_2$ );
R :=  $R_1 \times R_2$ ;
for each  $u \in U$  do
  read (LABELS( $u$ ));
call TREESEARCH (R1,R,LABELS);
stop
end MAIN;

comment error threshold;
comment mergeability relation;
comment primitives of the prototypes;
comment primitives of the candidate;
comment N-ary relation on U;
comment N-ary relation on L;
comment used in relaxation;

```

```
procedure TREESEARCH(R1,R,LABELS);
```

```
comment Recursively perform a tree search to find mappings from the set
        U of primitives of the prototype to the set L of primitives of
        the candidate. The mappings found will be  $\epsilon$ -homomorphisms from
        R1 to R2 where  $R = R1 \times R2$ ;
```

```
THISUNIT := an element of U that has the least number of labels;
```

```
NEXTLABEL: if (there are no more labels in LABELS(THISUNIT))
            then return;
```

```
THISLABEL := next label in LABELS(THISLABEL);
```

```
R' := RESTRICTION(R,THISUNIT,THISLABEL);
```

```
LABELS' := EPSILON PROJECTION(R1,R',LABELS);
```

```
if (LABELS' =  $\emptyset$ ) then goto NEXTLABEL
```

```
comment the following 4 statements are included if the tree search is to
        use a relaxation procedure on R;
```

```
RELAXR: R' := EPSILON RELAX(R1,R',LABELS');
```

```
LABELS' := EPSILON PROJECTION(R1,R',LABELS');
```

```
if (LABELS' =  $\emptyset$ ) then goto NEXTLABEL;
```

```
if (EPSILON RELAX removed any N-tuples from R')
```

```
then goto RELAXR;
```

```
comment If relaxation is not being used, the partial homomorphism defined
        by those elements of U that have only one label left in LABELS'
        must be checked to see that it meets the requirements of an  $\epsilon$ -
        homomorphism;
```

```
if ( $\neg$  EPSILON CONSISTENT(LABELS')) then goto NEXTLABEL;
```

```
comment Now if all the elements of U have only a single label left in LABELS',
        we have a homomorphism, otherwise continue the tree search;
```

```
if (LABELS' is single-valued)
```

```
then print ('HOMOMORPHISM',LABELS')
```

```
else call TREESEARCH(R1,R',LABELS');
```

```
comment Continue looking for more mappings;
```

```
goto NEXTLABEL;
```

```
end TREESEARCH
```

```
procedure EPSILON_PROJECTION(R1,R,LABELS);
```

```
comment Produce a new table containing for each unit u those labels that an
         $\epsilon$ -homomorphism may still map u to, given the current state of R;
```

```
for each unit UNIT that has only one label in LABELS do
```

```
begin
```

```
    EPSILON_PROJECTION(UNIT) := LABELS(UNIT);
```

```
    PARTIAL(UNIT) := LABELS(UNIT)
```

```
end;
```

```
for each unit UNIT that has more than one label in LABELS do
```

```
    for each label LAB in LABELS(UNIT) do
```

```
        begin
```

```
            PARTIAL(UNIT) := LAB;
```

```
            if EPSILON CONSISTENT(PARTIAL,R1,R)
```

```
            then add LAB to EPSILON_PROJECTION(UNIT)
```

```
        end;
```

```
return;
```

```
end EPSILON_PROJECTION;
```



```

procedure EPSILON_CONSISTENT(LABELS,RI,R)
comment determine if the partial function defined by the single-valued
           entries in LABELS is an  $\epsilon$ -homomorphism;
ERROR SUM := 0;
for each N-tuple  $(u_1, \dots, u_N) \in RI$  | such that each of  $u_1, \dots, u_N$  has just
           one label left in LABELS do
  begin
    if  $((u_1, LABELS(u_1)), \dots, (u_N, LABELS(u_N)))$  is
           not in R
    then ERROR SUM := ERROR SUM + weight( $u_1, \dots, u_N$ );
    if ERROR SUM >  $\epsilon$  then goto FAIL
  end;
  EPSILON_CONSISTENT = true;
  return;
FAIL: EPSILON_CONSISTENT = false;
  return
end EPSILON_CONSISTENT;

```

The algorithm described by the above procedures was used for shape matching. The relations  $R_i$  and  $R_p$  were used in place of the general relation  $R$ .  $R_i$  and  $R_p$  were stored, restricted, projected, and relaxed on separately because the program ran faster using the two smaller separate relations and combining their results than by merging them into one larger relation. The program was tested with and without a relaxation operator. Because the shape relations are relatively small, the overhead involved in setting up and carrying out the relaxation was greater than the amount of time required to perform the tree-search with the restriction and projection, but without the relaxation. The weighted discrete relaxation procedure, which does improve time significantly on larger relations, will be described in a future paper.

## V. EXPERIMENTAL RESULTS

The shape matching procedure has been tested on character data. The data consists of a set of hand-printed characters which are considered to be prototype shapes and for each prototype shape, a set of imperfect versions of the shape. The imperfect versions were obtained by starting with a list of the boundary points of the prototype shape and calling on a uniform random number generator to produce horizontal and vertical displacements which were then added to the coordinates of the boundary points. By using different seeds for the random number generator, each prototype shape was used to generate several imperfect shapes with the same number of points, but with distorted boundaries. The amount of distortion was controlled by the range of displacements. Figure 8 shows the prototype shapes and a set of corresponding imperfect versions.

The characters were first decomposed into simple parts and intrusions using the graph-theoretic clustering procedure described in [31]. Then the  $R_p$  and  $R_i$  relations were hand-coded for each character. In the hand-coding, we followed the definitions of  $R_p$  and  $R_i$  given in Section III as closely as possible using a distance function  $\delta$  defined by  $\delta(p_1, p_2) = 0$  if  $p_1$  and  $p_2$  have a point in common or if a point of  $p_1$  is adjacent (along the boundary of the shape) to a point of  $p_2$  and  $\delta(p_1, p_2) = \infty$  otherwise. (The software to automatically produce the relations is under current development). Several extra test shapes were constructed totally by hand in order to have some shapes which were not just imperfect versions of the prototype shapes, but were "near-misses" as used in Winston's concept-learning experiments [34]; that is, some important property of these shapes was missing. These shapes were used to further test the inexact matching.

In each test, the matching program was given the  $R_p$  and  $R_i$  relations, a weighting function, and a mergeability relation for the prototype shape; the  $R_p$  and  $R_i$  relations for the candidate shape, and the inexact matching

threshold  $\epsilon$ . We will first describe some results where the weighting function assigned equal weights to each triple in the prototype relations, the mergeability relation  $M$  was empty or fixed, and  $\epsilon$  was set to 0 for exact matching. We will then describe the effects of varying  $\epsilon$ , the mergeability relation, and the weights. We would like to emphasize at this point that our goals in running these experiments were not to test a new character recognition algorithm, but instead to learn about the concept of shape. Thus, for each test run, our interest was not in whether two shapes matched, but how much they matched and how much work the program had to do to decide how much they matched.





Imperfect Versions of Prototype Shapes

Figure 8 (continued).

### Example of Tests with Equal Weights, Fixed Mergeability, and $\epsilon = 0$

We will illustrate this kind of test with a few characteristic examples. In one such test letter E1 of Figure 7 was used as the prototype shape and letter E2 as the candidate shape. The program found the four legal 0-homomorphisms from E1 to E2 with a tree search in which 39 nodes were processed.

Figure 9 shows the tree search for this run. It is of interest to note that when the program used relaxation at each node of the tree search, only sixteen nodes were processed, but the program performed so many more operations due to the overhead of the relaxation that it executed four times as long.

As examples of non-matches, letter E1 was matched against letter I3, and letter I1, the prototype I, was run against letter E2. The decompositions of letter I1 and letter I3 are shown in Figure 10. In both cases the program reported no homomorphisms found while searching only six nodes of the tree. The tree searches are illustrated in Figure 11.

### Examples of Tests Where $\epsilon$ Was Varied

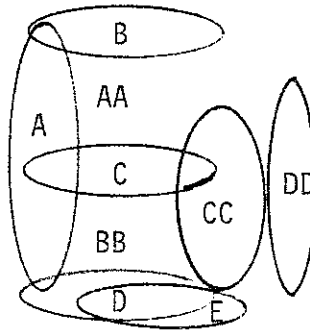
A number of tests were performed where  $\epsilon$  was varied in order to watch the behavior of the program as it went from exact matching to inexact matching and to see what kind of inexact matches occurred at higher values of  $\epsilon$ . The E1-E2 experiment which processed 39 nodes for  $\epsilon = 0$ , processed 31 nodes for  $\epsilon = .25$ , finding the four 0-homomorphisms plus four .25-homomorphisms. The additional .25 homomorphisms are given below.

| 1 | 2 | 3 | 4 | 11 | 12 |
|---|---|---|---|----|----|
| A | D | C | E | AA | BB |
| A | E | C | D | AA | BB |
| A | D | C | E | BB | AA |
| A | E | C | D | BB | AA |

E1

|   |    |
|---|----|
|   | 2  |
| 1 | 11 |
|   | 3  |
|   | 12 |
|   | 4  |

E2

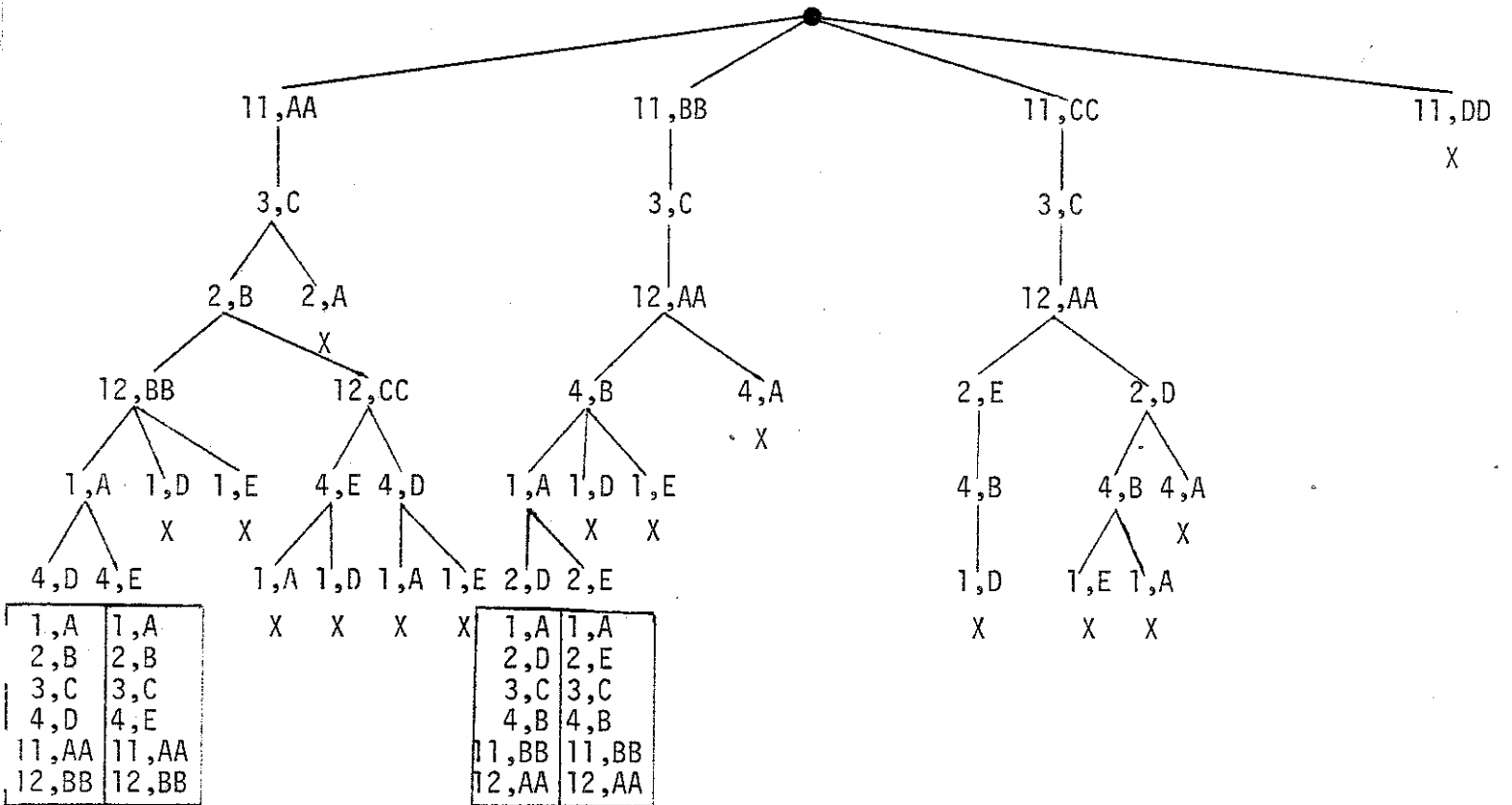


equal weights

$M = \emptyset$

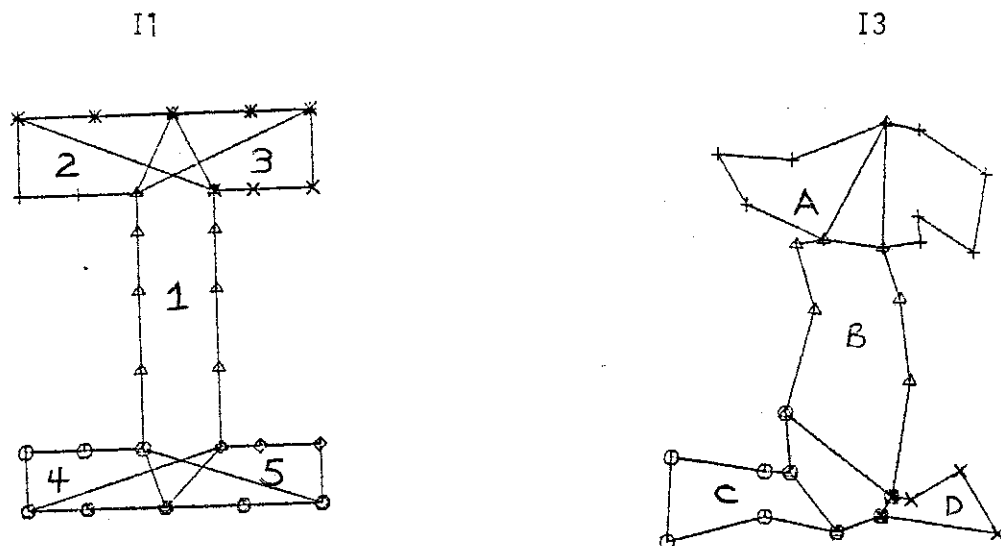
$\epsilon = 0$

39 nodes

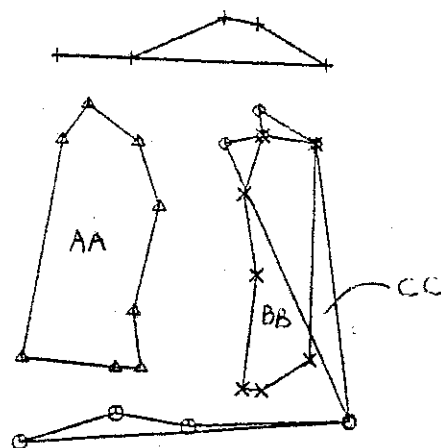
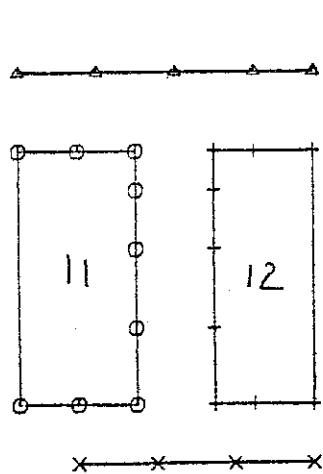


X indicates failure

Figure 9 illustrates the tree search to find all homomorphisms from letter E1 of Figure 7 to letter E2.



Simple Parts



Intrusions

Figure 10 illustrates the simple parts and intrusions for the prototype shape I1 and the imperfect shape I3.



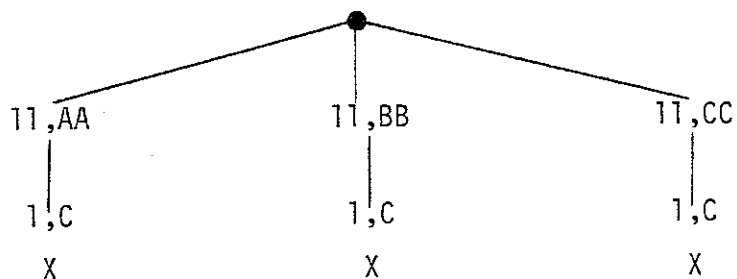
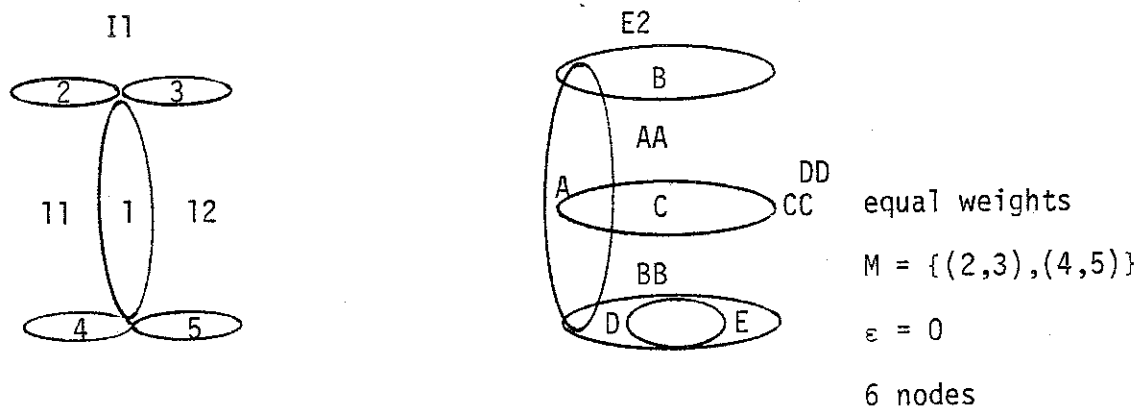
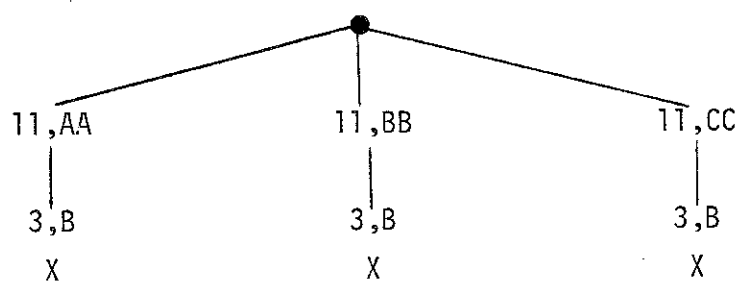
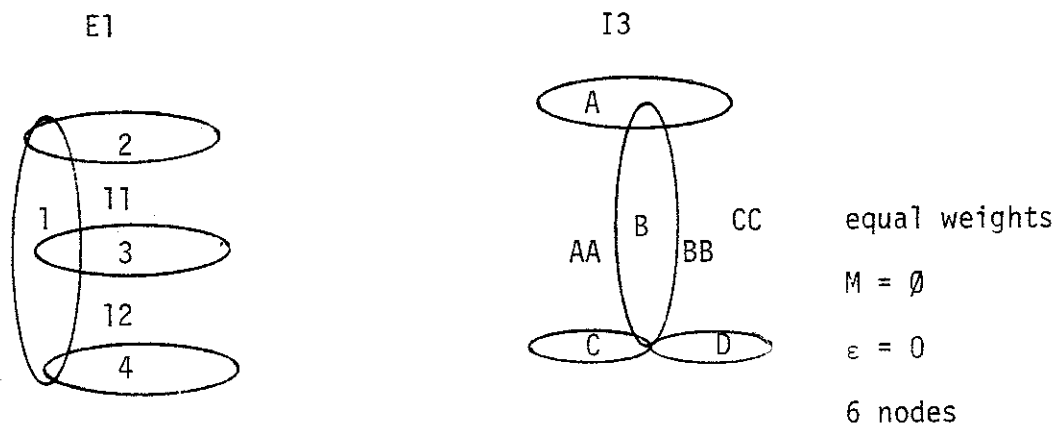


Figure 11 illustrates the tree search for two non-matches.

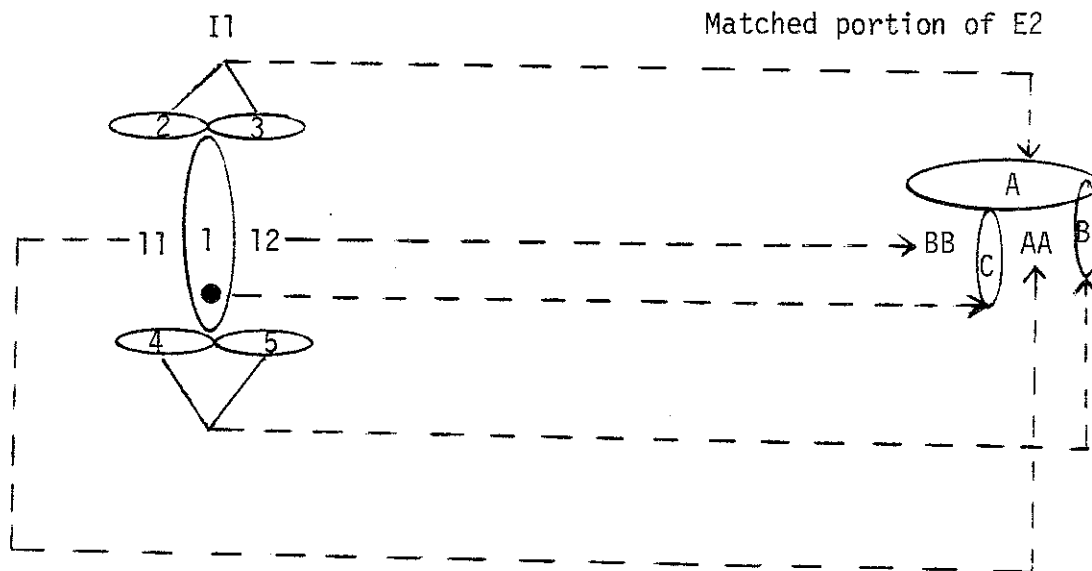
The E1-I3 experiment which processed 6 nodes for  $\epsilon = 0$ , processed 45 nodes for  $\epsilon = .5$ , again reporting no homomorphisms. The I1-E2 experiment which processed 6 nodes and found no matches for  $\epsilon = 0$ , processed 69 nodes in the time it was allowed to run and had found 13 .5-homomorphisms before it reached its time limit. One such .5-homomorphism was  $\{(1,C),(2,A),(3,A),(4,B),(5,B),(11,AA),(12,BB)\}$ . In this mapping, the triples  $(1,11,2)$  and  $(1,12,3)$  of  $R_i$  were satisfied and the triples  $(1,11,4)$  and  $(1,12,5)$  of  $R_i$  were not satisfied. Since half the triples of  $R_i$  were satisfied, the mapping was a valid .5-homomorphism on  $R_i$ .  $R_p$  contained only the triple  $(11,1,12)$  which was satisfied. Figure 12 illustrates this inexact match.

Figure 13 shows the relations used in another inexact matching experiment. The prototype shape was letter E1 and the candidate a constructed relation NOT\_E2. The program found no homomorphism for  $\epsilon = 0$  processing 10 nodes, no homomorphisms for  $\epsilon = .05$  processing 17 nodes, and no homomorphisms for  $\epsilon = .25$  processing 31 nodes. The reason for no homomorphisms in this example is that no mapping could satisfy the  $R_p$  relation, which forces one of the mappings  $\{(11,AA),(3,C),(12,BB)\}$  or  $\{(11,BB),(3,C),(12,AA)\}$ , and simultaneously satisfy the  $R_i$  relation.

Figure 14 shows the relations used and the tree searches in a similar experiment with prototypes E1 and candidate NOT\_E3. This time two inexact matches were found when  $\epsilon$  was set to .25. The program processed 10 nodes for  $\epsilon = 0$ , 17 nodes for  $\epsilon = .05$ , and 31 nodes for  $\epsilon = .25$ .

#### Examples of Tests Where M Was Varied

The mergeability relation M was the empty set for prototype E1 and was fixed at  $\{(2,3),(4,5)\}$  for prototype I1 in the above examples. In this section we will describe some experiments using prototype I1 where M was varied.



$R_i$   
 $\left. \begin{array}{l} 1,11,2 \\ 1,12,3 \end{array} \right\}$  satisfied  
 $\left. \begin{array}{l} 1,11,4 \\ 1,12,5 \end{array} \right\}$  not satisfied

$R_p$   
 11,1,12 satisfied

$R_i$   
 $C,AA,A$   
 $C,BB,A$   
 $B,AA,A$

$R_p$   
 AA,C,BB

Figure 12 illustrates an inexact match of I1 to a portion of E2 at  $\epsilon = .5$ .

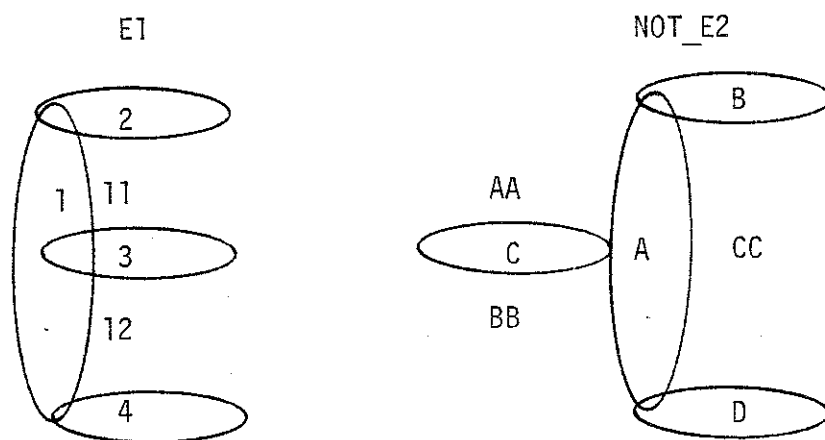
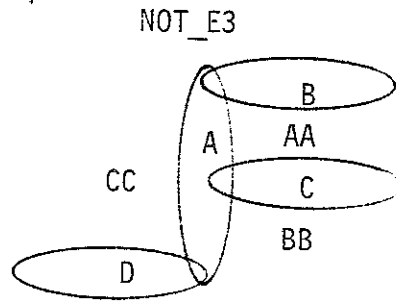
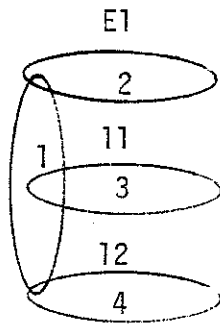
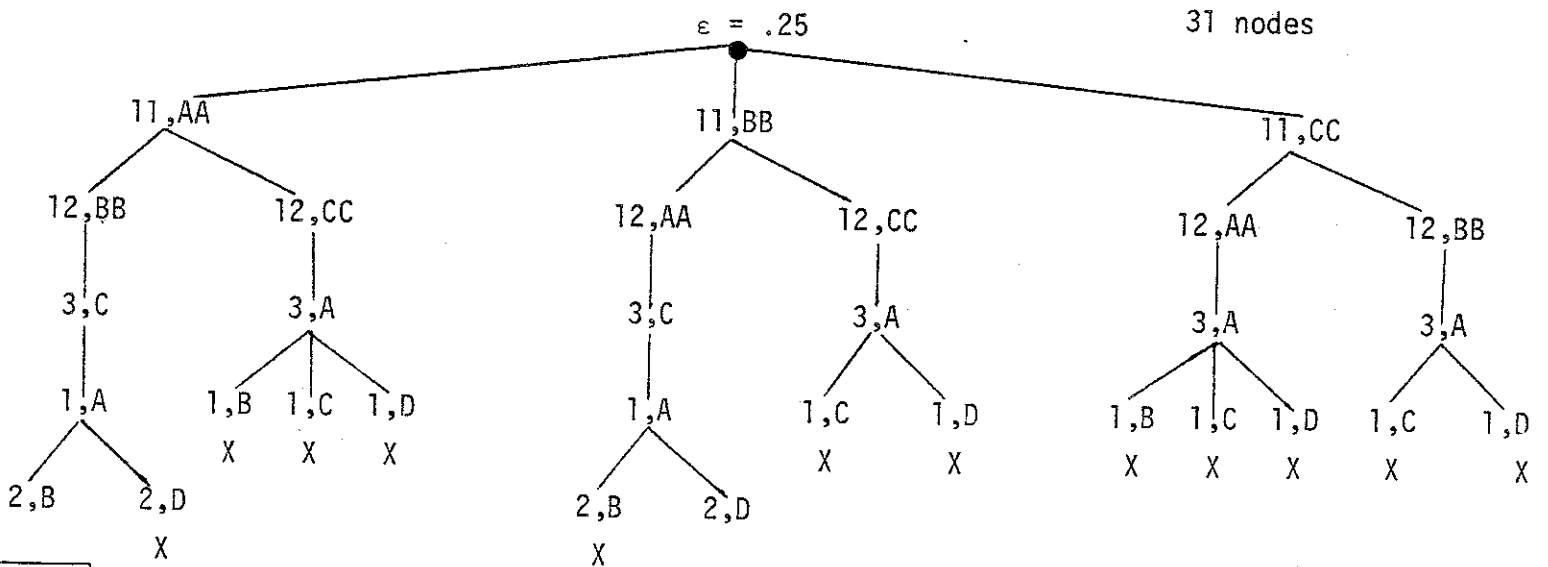
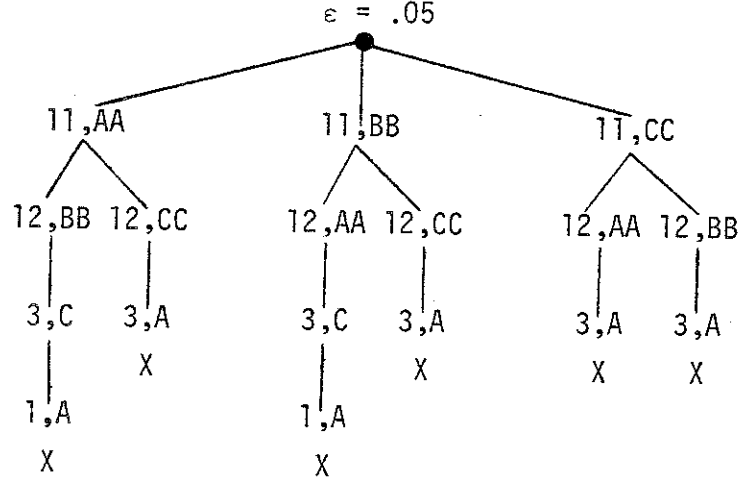
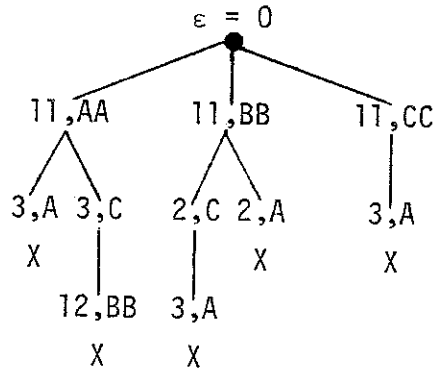


Figure 13 illustrates the relations used in an inexact matching experiment where no homomorphisms were found for  $\epsilon = 0, .05, \text{ or } .25$ .



10 nodes

17 nodes



- 1,A
- 2,B
- 3,C
- 4,D
- 11,AA
- 12,BB

- 1,A
- 2,D
- 3,C
- 4,B
- 11,BB
- 12,AA

Two .25-homomorphisms

Figure 14 illustrates the relations and tree searches for an inexact matching experiment where two homomorphisms were found for  $\epsilon = .25$ .

Prototype I1 was matched against candidate I3 (both shown in Figure 10) with  $M = \emptyset$ ,  $M = \{(2,3)\}$ ,  $M = \{(1,2),(1,3),(1,4),(1,5)\}$ , and  $M = \{(2,3)(4,5), (1,2),(1,3),(1,4),(1,5)\}$ . Each value of  $M$  was tested with  $\epsilon = 0$  and  $\epsilon = .25$ . For  $M = \emptyset$ , the program found no homomorphisms either with  $\epsilon = 0$  or with  $\epsilon = .25$ . For  $M = \{(2,3)\}$ , the program found two homomorphisms with  $\epsilon = 0$  and eight homomorphisms with  $\epsilon = .25$ . The 0-homomorphisms are listed below.

| 1 | 2 | 3 | 4 | 5 | 11 | 12 |
|---|---|---|---|---|----|----|
| B | A | A | C | D | AA | BB |
| B | A | A | D | C | BB | AA |

The .25-homomorphisms include the 0-homomorphisms and the six listed below.

|   | 1 | 2 | 3 | 4 | 5 | 11 | 12 |
|---|---|---|---|---|---|----|----|
| * | B | C | C | A | D | AA | BB |
| * | B | D | D | C | A | AA | BB |
|   | B | A | A | C | D | AA | CC |
| * | B | C | C | D | A | BB | AA |
| * | B | D | D | A | C | BB | AA |
|   | B | A | A | D | C | CC | AA |

Note that the mappings marked with "\*" are valid .25-homomorphisms with respect to  $R_i$  and  $R_p$  but are intuitively invalid because they map primitives 4 and 5 which touch to A and C or A and D which do not. This is caused by the fact that 4 and 5 join but border no intrusion and therefore were not included in the  $R_i$  relation. This problem can be solved by defining a null intrusion  $i_0$  and including  $(4, i_0, 5)$  and  $(2, i_0, 3)$  in the  $R_i$  relation. Of course, for higher values of  $\epsilon$ , these extra constraints could be ignored.

For  $M = \{(1,2),(1,3),(1,4),(1,5)\}$ , the program found no 0-homomorphisms and had discovered the three .25-homomorphisms listed below in its allotted time.

| 1 | 2 | 3 | 4 | 5 | 11 | 12 |
|---|---|---|---|---|----|----|
| B | B | A | D | C | BB | AA |
| B | B | C | D | A | BB | AA |
| B | D | A | B | C | BB | AA |

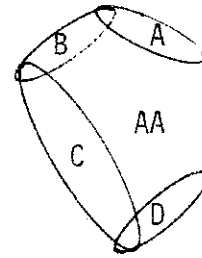
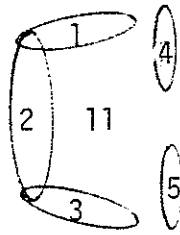
Finally for  $M = \{(2,3), (4,5), (1,2), (1,3), (1,4), (1,5)\}$  the program found four 0-homomorphisms and in its allotted time had found 22 .25-homomorphisms.

#### Examples of Tests Where Weights Were Varied

The weighting function was added to the shape matching procedure to add flexibility to the experiments. In this section we discuss the effects of varying the weights.

A simple experiment was run using the prototype letter 'C' and imperfect version shown in Figure 8. The decomposition of the two shapes is illustrated symbolically in Figure 15. The prototype, C1, has five simple parts and one intrusion while the candidate, C2, has only four simple parts and one intrusion. In the experiment the mergeability relation  $M$  was fixed at  $\{(1,2), (2,3), (3,5), (1,4)\}$  so that any two simple parts that touched or nearly touched could map to a single simple part. The experimental weighting function assigned high weights (.45) to the triples (1,11,2) and (2,11,3) which represent the joining of the three main primitives of the shape and low weights (.05) to the triples (1,11,4) and (3,11,5) which represent the joining of the two optional end primitives to the remainder of the shape. The experiment was also run with equal weights as a comparison. The error threshold  $\epsilon$  was fixed at .2.

As expected, the program reported no .2-homomorphisms when the weights were equal, since at least one of the four triples was not satisfied. With unequal weights a number of .2-homomorphisms were found. Figure 15 gives two of these homomorphisms,  $h_1$  and  $h_2$ , the triples that were not satisfied by



| $R_{i1}$ | Unequal Weights | Equal Weights | $R_{i2}$ |
|----------|-----------------|---------------|----------|
| 1,11,2   | .45             | .25           | A,AA,B   |
| 2,11,3   | .45             | .25           | B,AA,C   |
| 1,11,4   | .05             | .25           | C,AA,D   |
| 3,11,5   | .05             | .25           |          |

$R_{p1} = \emptyset$

$R_{p2} = \emptyset$

$M = \{(1,2),(2,3),(3,5),(1,4)\}$

|       | 1 | 2 | 3 | 4 | 5 | 11 | Not Satisfied        | Error With Unequal Weights | Error With Equal Weights |
|-------|---|---|---|---|---|----|----------------------|----------------------------|--------------------------|
| $h_1$ | A | B | C | D | A | AA | (1,11,4)<br>(3,11,5) | .1                         | .5                       |
| $h_2$ | B | C | D | A | D | AA | (3,11,5)             | .05                        | .25                      |

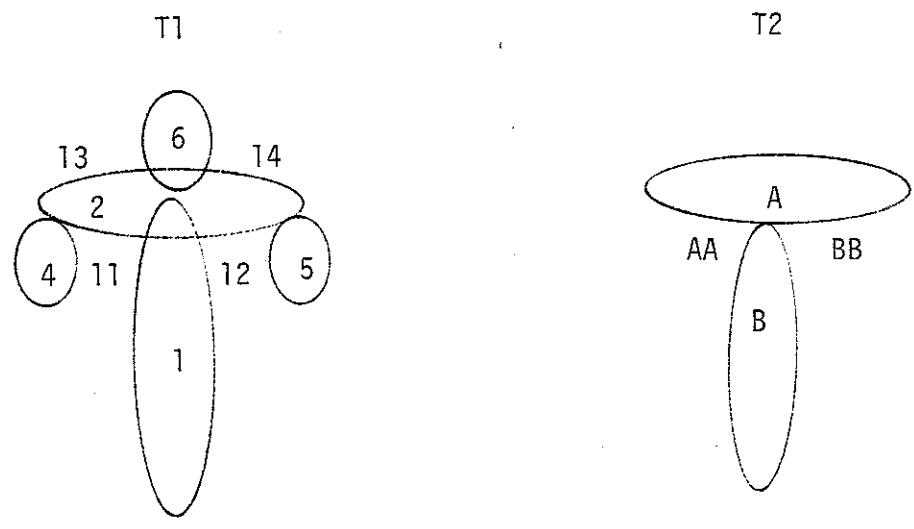
Figure 15 illustrates a matching experiment on letter C's where weights were varied. The functions  $h_1$  and  $h_3$  are two of the .2-homomorphisms found when the unequal weights were used.



each function, and the corresponding error for each function for the unequal weights experiment and the equal weights experiment.

A second experiment in which weights were varied was run on two letter 'T' relations which were created expressly for the experiment. The relations for these shapes is shown in Figure 16. The prototype shape T1 has several optional extra parts that may or may not be present on a letter 'T'. The candidate shape T2 is a standard 'T' shape. In this experiment, instead of allowing two primitives in the prototype to map to a single primitive in the candidate, the mergeability relation was empty and primitives were allowed to map to the symbol NL meaning "no label". Thus, the effect was to look at partial mappings. Although the approach is impractical (there are too many partial mappings), the experiment does show the effect of the weights on the mappings found.

Figure 16 shows the experimental weighting function and the equal weights function. The experimental weighting function assigns the highest weights to the relationships that make a shape 'T'-shaped and lower weights to the optional relationships. The mapping  $h$  in Figure 16 is one of the .2-homomorphisms that was found with the unequal weights, but did not exist when the equal weights were used. The two experiments show that varying the weights can be used to set up a model with many low priority optional parts and relationships.



| $R_{i1}$ | Unequal Weights | Equal Weights | $R_{i2}$ |
|----------|-----------------|---------------|----------|
| 6,13,2   | .05             | .167          | A,AA,B   |
| 6,14,2   | .05             | .167          | A,BB,B   |
| 2,11,4   | .2              | .167          |          |
| 2,12,5   | .2              | .167          |          |
| 1,11,2   | .25             | .167          |          |
| 1,12,2   | .25             | .167          |          |

| $R_{p1}$ | Unequal Weights | Equal Weights | $R_{p2}$ |
|----------|-----------------|---------------|----------|
| 11,1,12  | .7              | .25           | AA,B,BB  |
| 11,2,13  | .1              | .25           |          |
| 12,2,14  | .1              | .25           |          |
| 13,6,14  | .1              | .25           |          |

$M = \emptyset$

|   | 1 | 2 | 4  | 5  | 6  | 11 | 12 | 13 | 14 | Triples Not Satisfied  | Error With Unequal Weights | Error With Equal Weights |
|---|---|---|----|----|----|----|----|----|----|------------------------|----------------------------|--------------------------|
| h | B | A | NL | NL | NL | AA | BB | NL | NL | (11,2,13)<br>(12,2,14) | .2                         | .5                       |

Figure 16 illustrates a matching experiment on two letter T's where weights were varied. The function h is a .2-homomorphism found when the unequal weights were used.

## VI. DISCUSSION AND CONCLUSIONS

We have discussed a general model for a structural description of a shape and defined a specific model based on decomposing the shape into simple parts and intrusions. In our model two relations, the intrusion relation and the protrusion relation, characterize the shape. The descriptions can be enhanced by adding properties of the simple parts and intrusions.

We have defined shape matching as the problem of finding relational homomorphisms from a prototype shape to a candidate shape. We have discussed homomorphisms on weighted relations and homomorphisms that can assign several candidate primitives to one prototype primitive. We have also discussed some measures for the goodness of a match. Finally, we have described an experimental shape matching procedure that uses a tree search to find homomorphisms from prototype shapes to candidate shapes. At this point, we would like to comment on the results so far and the work yet to be done.

The main contribution of this work is the relational shape model and specialized shape matching procedure. The intrusion and protrusion relations appear to be sufficient to characterize a shape. The model is both simple and powerful. The shape matching procedure incorporating weights, thresholds, and a mergeability relation is a highly flexible matching procedure. Because not all of the parts have to be distinct and not all of the relationships have to be satisfied, inexact matches and partial matches are possible. Thus, shapes that are distorted or obstructed can still match their prototypes. This important characteristic is needed in scene analysis systems.

The entire shape recognition system at present consists of a set of FORTRAN programs that perform the decomposition of a shape and a SNOBOL program that performs the shape matching procedure. The automation of extracting the  $R_i$  and  $R_p$  relations and other information from the results of the decomposition is

currently being tackled. For a shape represented by  $N$  points, the FORTRAN programs perform  $O(N^3)$  operations for computing relations and  $O(N^2)$  operations for graph-theoretic clustering. We hope to reduce the complexity of the  $O(N^3)$  program by improving the algorithm used. (The current one is just brute force). The SNOBOL program, in the worst case, would search the entire tree since finding homomorphisms is an NP-complete problem. However, in our experiments, the program searched a relatively small portion of the tree. As mentioned previously, the program executed factor on these simple shapes without the relaxation procedure. However, when relaxation was used, the program rarely made a mistake requiring back-up. Thus, for very complex shapes with many primitives to be matched, a relaxation procedure should be used. In practice, the number of nodes searched seems to be proportional to the number of primitives of the prototype shape times the number of homomorphisms found. As the error threshold  $\epsilon$  increases, the program tends to search more nodes and to find more homomorphisms. Occasionally, the program will search fewer nodes with a higher value of  $\epsilon$  because the order of the search has changed.

The current SNOBOL shape matching procedure is very slow; approximately 2 1/4 nodes per second are searched. This is mostly due to the fact that the program uses character strings to represent most data objects (triples, lists, units, and labels) and character string operators such as pattern matching and concatenation for manipulating the data objects. The program was initially written in SNOBOL4 because the powerful data structures and facilities of the language made for a short program and an equally short debugging time. More efficient matching programs are currently being developed in FORTRAN and PASCAL.

The results reported here are only initial experimental results. We expect to perform many more experiments using more extensive data and studying further the effects of different weighting schemes and mergeability relations. We also

wish to study the effects of adding properties of the primitives to the model (putting statistical and structural descriptions together), and we will be adding symmetry to the model. Another goal is to develop a shape matching procedure that constructs a third shape from which there are homomorphisms to each of two shapes being tested for similarity. This will take care of the problem of a primitive of the prototype shape mapping to two different primitives of the candidate shape. In summary, the shape recognition procedure as a whole seems to be a promising tool in scene analysis.

## REFERENCES

1. Alt, F.L., "Digital Pattern Recognition by Moments," JACM, Vol. 11, 1962, pp. 240-258.
2. Agrawala, A.K. and A.V. Kulkarni, "A Sequential Approach to the Extraction of Shape Features," Computer Graphics and Image Processing, Vol. 6, No. 6, December 1977, pp. 538-557.
3. Barrow, H.G., A.P. Ambler, and R.M. Burstall, "Some Techniques for Recognizing Structures in Pictures," Frontiers of Pattern Recognition, S. Watanabe (Editor), Academic Press, 1972, pp. 1-29.
4. Blum, H., "A Transformation for Extracting New Descriptions of Shape," Symposium on Models for the Perception of Speech and Visual Form.
5. Corneil, D.G. and C.C. Gottlieb, "An Efficient Algorithm for Graph Isomorphism," JACM, Vol. 17, No. 1, January 1970, pp. 51-64.
6. Davis, L.S., "Shape Matching Using Relaxation Techniques," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-1, No. 1, pp. 60-72.
7. Davis, L.S., "Understanding Shape: Angles and Sides," IEEE Transactions on Computers, Vol. C-26, No. 3, March 1977, pp. 236-242.
8. Davis, L.S., "Understanding Shape II: Symmetry," IEEE Transactions on Systems, Man, and Cybernetics, March 1977, pp. 204-212.
9. Eden, M., "Handwriting and Pattern Recognition," IRE Transactions on Information Theory, Vol. IT-8, 1962, pp. 160-166.
10. Feng, H.F. and T. Pavlidis, "Decomposition of Polygons into Simpler Components: Feature Generation for Syntactic Pattern Recognition," IEEE Transactions on Computers, Vol. C-24, No. 6, June 1975, pp. 636-650.
11. Freeman, H., "Computer Processing of Line Drawing Images," Computing Surveys, Vol. 6, No. 1, March 1974, pp. 57-97.
12. Fu, K.S. and S.Y. Ly, "A Clustering Procedure for Syntactic Patterns," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-7, 1977, pp. 734-742.
13. Granlund, G.H., "Fourier Preprocessing for Hand Print Character Recognition," IEEE Transactions on Computers, February 1972, pp. 195-201.
14. Haralick, R.M. and J. Kartus, "Arrangements, Homomorphisms, and Discrete Relaxation," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-8, No. 8, August 1978, pp. 600-612.
15. Haralick, R.M. and L.G. Shapiro, "The Consistent Labeling Problem," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-1, No. 2, 1979.

16. Haralick, R.M. and L.G. Shapiro, "Decomposition of Polygonal Shapes by Clustering," Proceedings of the First IEEE Conference on Pattern Recognition and Image Processing, June 1977, pp. 183-190.
17. Haralick, R.M., "Statistical Shape Descriptors for Simple Shapes," Unpublished work.
18. Horowitz, S.L., "Peak Recognition in Waveforms," Syntactic Pattern Recognition Applications, K.S. Fu (Editor), Springer-Verlag, Berlin, 1977.
19. Hu, Ming-Kuei, "Visual Pattern Recognition by Moment Invariants," IRE Transactions on Information Theory, February 1962, pp. 179-187.
20. Langridge, D., "On the Computation of Shape," Frontiers of Pattern Recognition, S. Watanabe (Editor), Academic Press, New York, 1962, pp. 347-365.
21. Lozano-Perez, T., "Parsing Intensity Profiles," Computer Graphics and Image Processing, Vol. 6, 1977, pp. 43-60.
22. Maruyama, K., A Study of Visual Shape Perception, VIVDCS-R-72-533, Department of Computer Science, University of Illinois, October 1972.
23. O'Callaghan, J.F., "Recovery of Perceptual Shape Organization from Simple Closed Boundaries," Computer Graphics and Image Processing, Vol. 3, No. 4, December 1974, pp. 300-312.
24. Pavlidis, T. and F. Ali, "A Hierarchical Syntactic Shape Analyzer," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-1, No. 1, January 1979, pp. 2-9.
25. Pavlidis, T., "Representation of Figures by Labelled Graphs," Pattern Recognition, Vol. 4, 1972, pp. 5-17.
26. Pavlidis, T., "A Review of Algorithms for Shape Analysis," Computer Graphics and Image Processing, Vol. 7, No. 2, April 1978, pp. 243-258.
27. Persoon, E. and K.S. Fu, "Shape Discrimination Using Fourier Descriptors," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-7, 1977, pp. 170-179.
28. Richard, C.W. and H. Hemami, "Identification of Three-Dimensional Objects Using Fourier Descriptors of the Boundary Curve," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-4, No. 4, July 1974, pp. 371-377.
29. Rosenberg, B., "The Analysis of Convex Blobs," Computer Graphics and Image Processing, Vol. 1, 1972, pp. 183-192.
30. Rosenberg, B., "Computing Dominant Points on Single Shapes," Int. J. Man-Machine Studies, Vol. 6, 1975, pp. 1-12.

31. Shapiro, L.G. and R.M. Haralick, "Decomposition of Two-Dimensional Shapes by Graph-Theoretic Clustering," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-1, No. 1, January 1979, pp. 10-20.
32. Shaw, A.C., "Parsing of Graph-Representable Pictures," JACM, Vol. 17, No. 3, July 1970, pp. 453-481.
33. Ullman, J.R., "An Algorithm for Subgraph Isomorphism," JACM, Vol. 23, No. 1, January 1976, pp. 31-42.
34. Winston, P.H., "Learning Structural Descriptions from Examples," The Psychology of Computer Vision, P.H. Winston (Editor), McGraw-Hill, 1975, pp. 157-209.
35. You, K.C. and K.S. Fu, "Syntactic Shape Recognition Using Attributed Grammars," 1978 EIA Symposium on Emerging Patterns in Automatic Imagery Pattern Recognition, April 3-4, Gaithersburg, Maryland.
36. Zahn, C.T. and R.Z. Roskies, "Fourier Descriptors for Plane Closed Curves," IEEE Transactions on Computers, Vol. C-21, 1972, pp. 269-281.