AD-A072 779 PURDUE UNIV LAFAYETTE IN SCHOOL OF ELECTRICAL ENGINEERING F/G 9/2
SYNTACTIC SHAPE RECOGNITION USING ATTRIBUTED GRAMMARS.(U)
AUG 78 K C YOU, K S FU AFOSR-74-2661
UNCLASSIFIED TR-EE78-38 AFOSR-TR-78-1425 NL

1 OF 3
AD
A072779

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered) Unclassified

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFOSR-TR- 78-1425	2. GOVT ACCESSION NO. (19) TR-78-1425
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): SYNTACTIC SHAPE RECOGNITION USING ATTRIBUTED GRAMMARS	5. TYPE OF REPORT & PERIOD COVERED: (9) INTERIM rept.
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): (10) K. C. You and K. S. Fu	8. CONTRACT OR GRANT NUMBER(s): (15) AFOSR-74-2661, MDA903-77-C-0225
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Purdue University, School of Electrical Engineering, West Lafayette, Indiana 47907	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: (16) 61102F 2304/A2 (17) A2
11. CONTROLLING OFFICE NAME AND ADDRESS: Air Force Office of Scientific Research/NM, Bolling AFB, Washington, D.C. 20332	12. REPORT DATE: (11) August 1978
	13. NUMBER OF PAGES: 264
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

LEVEL II

16. DISTRIBUTION STATEMENT (of this Report) (12) 268 p.

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

(14) TR-EE 78-38

18. SUPPLEMENTARY NOTES

D D C RECEIVED 14 AUG 1979

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A syntactic approach is applied to the shape description and recognition. The structure of a shape is described by grammatical rules and the local details by primitives. Four attributes are proposed to describe an open curve segment, and the angle between two consecutive curve segments is used to describe the connection. The property of the attributes and the recognition capability of this method are studies. The primitive extraction and syntax analysis can be performed in the same step by using both semantic and

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

292 000

AD A072779

DDC FILE COPY

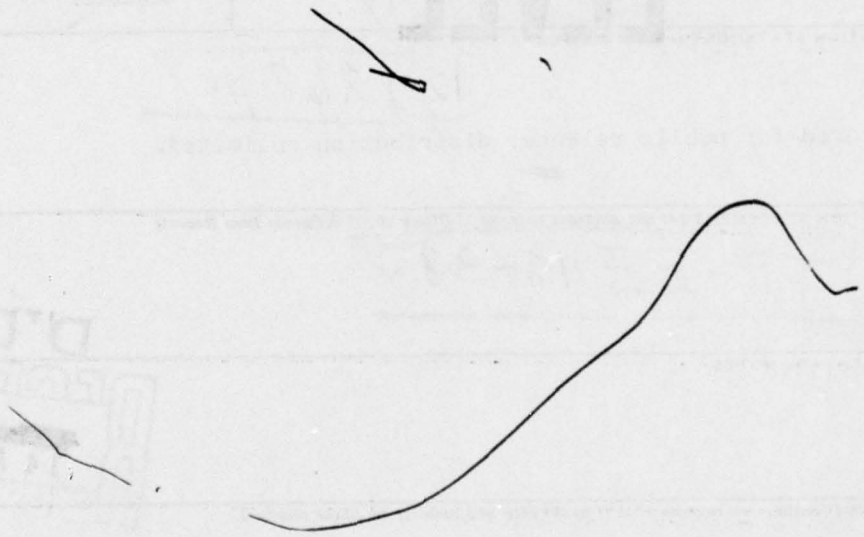
20. Abstract continued.

and syntactic information, anmely, the attributes and production rules.

The recognition system has been implemented and tested on the recognition of airplane shapes. The performance is quite satisfactory with respect to accuracy and computational efficiency. The method is extended to recognize partially distorted shapes. The distorted portion of the shape can be measured in terms of error-weight. The class membership functions of different shapes and the error error-weight estimation of the distorted portions are included in the extended recognition algorithms for recognizing noisy and distorted shape patterns. The success of this extension shows the advantage of the syntactic approach using attributed grammars over other existing shape recognition methods.

The grammatical inference procedures for shape grammars are also developed. The shape grammars can be inferred automatically, interactively or manually directly from the noisy vector patterns.

Although this approach has only been tested on airplane shapes, the results are also applicable to more general shape analysis problems.



AFOSR-TR- 78-1425

SYNTACTIC SHAPE RECOGNITION USING ATTRIBUTED GRAMMARS

K. C. You
K. S. Fu

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

TR-EE 78-38
August 1978

This work was supported by AFOSR Grant 74-2661 and DARPA Grant MDA 903-77-G-1.

SYNTACTIC SHAPE RECOGNITION USING ATTRIBUTED GRAMMARS

K. C. You and K. S. Fu
School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

TR-EE 78-38

This work was supported by AFOSR Grant 74-2661 and DARPA Grant MDA 903-77-G-1

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

A syntactic approach is applied to the shape description and recognition. The structure of a shape is described by grammatical rules and the local details by primitives. Four attributes are proposed to describe an open curve segment, and the angle between two consecutive curve segments is used to describe the connection. The property of the attributes and the recognition capability of this method are studied. The primitive extraction and syntax analysis can be performed in the same step by using both semantic and syntactic information, namely, the attributes and production rules.

The recognition system has been implemented and tested on the recognition of airplane shapes. The performance is quite satisfactory with respect to accuracy and computational efficiency. The method is extended to recognize partially distorted shapes. The distorted portion of the shape can be measured in terms of error-weight. The class membership functions of different shapes and the error-weight estimation of the distorted portions are included in the extended recognition algorithms for recognizing noisy and distorted shape patterns. The success

of this extension shows the advantage of the syntactic approach using attributed grammars over other existing shape recognition methods.

The grammatical inference procedures for shape grammars are also developed. The shape grammars can be inferred automatically, interactively or manually directly from the noisy vector patterns.

Although this approach has only been tested on airplane shapes, the results are also applicable to more general shape analysis problems.

CHAPTER 1

INTRODUCTION

1.1 Shape Recognition

Since the early 1960's, pattern recognition has received increasing attention because of the utilization of digital computers. The recognition techniques can be applied to many pictorial data, such as characters, machine parts, fingerprints, aerial reconnaissance pictures, bubble chamber photographs, chest radiographs, blood cells, chromosome images, etc. The pattern of interest may be an object, a disease, or a phenomenon. The picture may provide information by its shape, texture, and/or color. Shape seems to be the most important of these in many recognition applications. For instance, characters are recognized purely by their shapes. Machine tools and parts can also be recognized by their shapes. In our everyday life experience, we also find that most of the time we can identify an object only by its shape.

Shapes are usually described by their skeleton, contour, symmetry, or other numerical features. The skeleton which describes the structure of an object can be obtained from the contour [17,33]. Symmetry or other numerical features are usually derived from the contour [18,20,25,55]. Sometimes, the region inside the contour is also used to extract information for recognition [6], and the inside region is obvi-

ously dependent on the contour. For patterns which are very similar in structure, the boundary details may be needed for discrimination. The contour provides structural information as well as boundary detail information. The recent papers [50,55] by Davis also suggested describing and understanding a picture by characterizing the angles, sides, and symmetry of the outlines. This implies that the outer boundary is very informative in describing and recognizing an object in the two dimensional image. Therefore, we concentrate our interests on object identification using shape information and we define shape as the outer contour outlining the object in the two dimensional image.

1.2 A Syntactic Approach to General Shape Recognition

Over the past years, many methods for shape description and recognition have been studied. Most of them will be discussed extensively in Chapter 2 under three categories: template matching [3], statistical methods [1,2,3], and syntactic methods [4,5]. To recognize a picture by straightforward template-matching, the machine has to memorize a large number of templates. For shapes which differ in structure, we can store the skeletal templates in the machine and match the skeletons to recognize a shape. But the methods for finding the skeleton are either too sensitive to noise or not sensitive enough to the detailed variations in the boundary.

To apply the statistical methods to picture recognition, we need to select features which can reflect the differences between classes and are not sensitive to noise. The feature space can be partitioned into regions corresponding to classes. An unknown pattern is then recognized by locating its extracted features in the feature space. A pattern

whose extracted features are found in a region in the feature space is recognized as the corresponding class. The statistical methods are effective, if proper features can be selected. Unfortunately, it is usually difficult to select good features. Fourier descriptors and moment invariants are good features for the recognition of rigid-body objects. When parts of the object are covered or touched by other objects, parts of the shape may be badly distorted, while the rest of the shape may be left unaltered. This type of distortion may affect the feature values so badly that the recognition fails.

The syntactic approach essentially breaks a shape into simpler subshapes. The production rules describe the relationships among the subshapes. The simplest shape elements are defined as primitives and described by attributes. An unknown shape pattern is then recognized by recognizing its primitives and analyzing the syntax among the primitives according to the grammatical rules. The syntactic approach seems to be the most promising approach, because it uses grammatical rules to describe the shape structure explicitly and primitives to describe the boundary details. Other existing syntactic methods are developed for particular applications. They use specially-defined primitives for particular pattern classes. The semantic information, the attributes, are used only in the primitive extraction stage, and the syntactic information, the production rules, are used in the parsing or syntax analyzing stage. We intend to develop a method which fully utilizes the syntactic and semantic information to solve a general class of shape analysis problems.

In defining primitives for general shapes, we must try to avoid requiring a context-sensitive grammar to describe a shape, because context-sensitive grammars are very difficult to parse. If the primitives are very simple curve segments and are fixed in length, then we may need context-sensitive grammars to handle the size problem. In fact, the complexity of primitives and production rules are inversely related. We may use complex production rules for simple primitives or vice versa. In order to minimize the overall complexity, we may use primitives which are sophisticated enough to avoid context-sensitive grammars, but are characterized by a rather small number of features, or attributes. The attributes can be considered as carrying the semantic information of a primitive. To fully utilize the syntactic and semantic information to obtain an optimal solution, we employ the attributed grammar to describe and recognize general shapes.

1.3 Summary of the Contents

Chapter 2 is an extensive survey of the existing methods for shape description and recognition. Our proposed syntactic method is described and discussed in Chapter 3. We started our study with the geometrical analysis of curve segments in a continuous case. We found that four attributes are sufficient to describe a simple curve segment. And they can be transformed into four variables which are invariant with respect to translation, rotation, and scaling of the shape. Computation of the attributes in a discrete case are obtained. The relevant properties are discussed. It is interesting that the four attributes can also characterize a nonterminal curve segment. Therefore, every nonterminal or primitive has attributes. Also for each production rule there is a set

of attribute rules to relate the attributes. These properties satisfy the definition of an attributed grammar [5,44]. With attributed grammars, we actually describe and recognize shapes by using both syntactic and semantic information. In fact, both syntactic and semantic information can be used in the same step, the primitive-extraction-embedding parsing, which performs the primitive extraction and parsing together. This integrity reduces the errors in primitive extraction and their consequent difficulties.

Chapter 4 describes an implementation of this method with aircraft shape recognition. This implementation is designed to demonstrate that the proposed syntactic method is capable of discriminating shapes by structure as well as by boundary detail. The process of pictorial pattern recognition usually consists of three steps: preprocessing, description, and recognition. (See Figure 1.1.) In our method, the primitive-extraction-embedding parsing practically combines the latter two steps. In the first step of preprocessing, many operations including digitization, noise reduction, object detection, scene segmentation, etc. may be performed. Since we are interested in using the shape information, we assume that the shape or contour can be easily obtained from the picture. In our experiments, the preprocessing stage includes digitization thresholding, boundary following, and smoothing. It then outputs a shape. The shape, a chain of vectors, is subsequently recognized by a parsing program. All these programs are written in FORTRAN.

The recognition schemes are extended to recognize some badly distorted shapes in Chapter 5. Distorted shape recognition is an important advantage of the syntactic method over other existing methods. We gen-

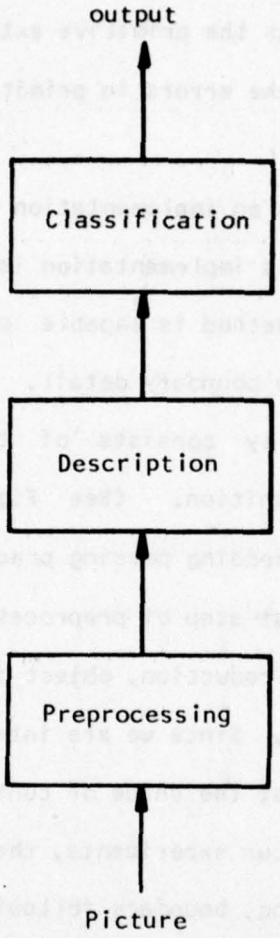


Figure 1.1 Pictorial Pattern Recognition System

eralized the recognition functions for primitives and non-terminals, and utilize the error-correcting technique in the primitive-extraction-embedding parser, so that the parser can fully use the context information to correctly segment the boundary chain and compute the percentage of boundary that is distorted. Several modified versions of Earley's parsing algorithm are described and discussed to show the feasibility of the ideas. Some experimental results demonstrate the power of recognizing distorted shapes. Unfortunately, these modified algorithms have the disadvantages of large storage and long computational time.

Chapter 6 is devoted to grammatical inference. The shape grammars can be constructed manually, interactively, or automatically. The major contents include an automatic learning algorithm ALA, an interactive learning algorithm ILA, and the conversion from context-free shape grammars to finite-state shape grammars. Both ALA and ILA are designed to infer shape grammars from noisy boundary vector chains.

Chapter 7 summarizes the results of this study and proposes suggestions for future research.

CHAPTER 2

SURVEY OF SHAPE DESCRIPTION AND RECOGNITION

Shape has been an interesting research topic for more than twenty years. Attneave [49] focused his attention on the angles of the shapes in 1954. Davis [50,55] studied the methods for detecting angles, sides, and symmetry. Pavlidis [51] and Ramer [52] tried to approximate plane curves by straight lines. We concentrate our interests on using shape information for recognition purposes.

As mentioned in Chapter 1, the description and recognition of shapes are two consecutive steps in the pattern recognition system. The recognition or classification is very much related to the description method. Many methods have been proposed to describe and recognize objects by shapes. Most of them will be discussed in the following sections.

2.1 Template Matching

Template matching [3,7] or prototype matching is one of the most primitive methods of pattern recognition and is still used in many practical applications. This method can be described as follows. Some templates or prototypes with known classification are stored in the machine. The machine matches the unknown input pattern to the prototypes, calculate the differences with a difference function, and then

assigns the unknown to the closest class. For fixed and restricted shapes, such as the recognition of characters printed by the same machine, this method is simple and useful. If the input patterns have many varieties for each class, the machine has to memorize a large number of prototypes. Sometimes one difference function may not work, unless all the possible prototypes are stored in the machine.

For example, Figures 2.1(a) and 2.1(b) are two prototypes of numeral "8" and "3". Let us use the number of different pixels as the difference calculation function. The difference between Figures 2.1(c) and 2.1(a) is 32, while that between Figures 2.1(c) and 2.1(b) is 23. And the difference between Figures 2.1(d) and 2.1(a) is 22, while that between Figures 2.1(d) and 2.1(b) is 28. The misclassification is obvious. It is not easy for this method to handle small changes in the input pattern, even though the change seems trivial to a human. However, this method is practically used in commercialized machines, such as postal sorting machines, credit card bookkeeping machines, etc. The main reason for this is that this method can be implemented optically [7] to achieve very high speeds. In order for the machine to recognize objects more intelligently, more sophisticated matching techniques such as relaxation [56] were developed.

In some cases, classes are different only in structure. The skeleton is used to describe and recognize a shape. Many skeletons are stored in memory as templates. For an unknown pattern, the skeleton is extracted and matched to the template skeletons for classification. There are many techniques of extracting the skeleton of a shape such as medial axis transformation (MAT) [33,34], polygonal approximation [29],

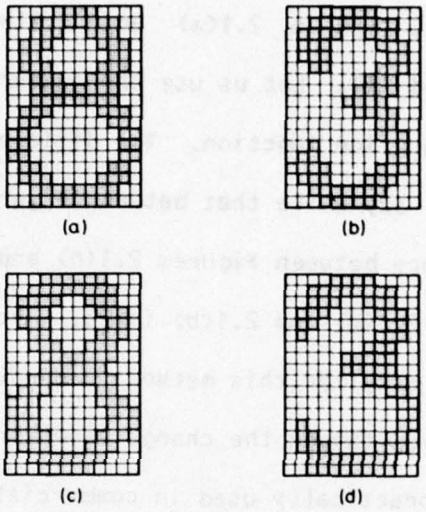


Figure 2.1 Two Different Types of Numerals, 8's and 3's

and Fourier transformation [17]. MAT was the first studied technique but it is computationally expensive and sensitive to noise. Polygonal approximation and Fourier transformation are insensitive to noise, but also insensitive to boundary details. A skeleton is usually represented as a graph.

2.2 Statistical Method

Statistical pattern recognition [1,2,3] has developed rapidly since the utilization of computers in the early 1960's. For processing pictorial data, the statistical pattern recognition system can be described as in Figure 2.2.

We first select the important features which can well characterize the pattern and teach the machine how to extract these features. After the preprocessing, the machine extracts the feature values from the unknown input pattern and uses them for classification. These selected features should be sensitive to different classes yet insensitive to noise and changes of input patterns. But, they usually depend upon the particular application. The features often used will be described in following subsections.

2.2.1 Topological Features [3]

Shape is sometimes defined to be more complicated than simply the outer contour of the object. That is, the shape may contain many regions. If the image is twisted or the object is deformed, the appearance will be different. But there are some topological properties which are invariant with respect to any planary stretching deformation. They are the number of connected components, C, the number of holes, H, and

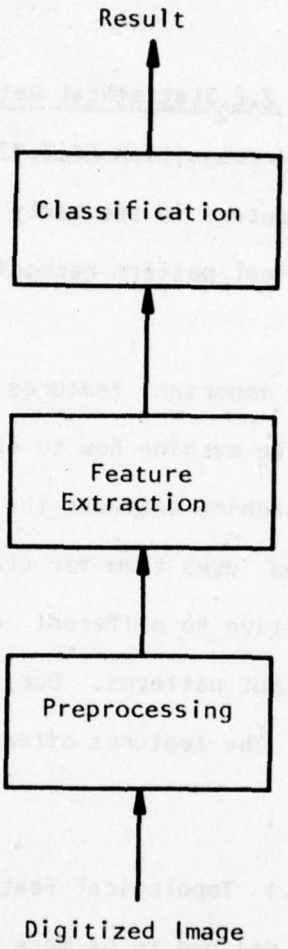


Figure 2.2 A Statistical Pattern Recognition System

hence the Euler number, E = C-H [3].

For pictures of straight line boundaries, known as polygonal networks, the number of vertexes, sides, and faces are also important measurements. Greamas [45], Munson [46] applied these properties to character recognition. Yokoi et. al. [47] did further analysis on binary pictures. Rosenfeld [48] studied the converse to the Jordan curve theorem for digital curves.

The topological properties are useful for stretchable objects and rubber-sheet images, but they cannot fully describe a shape.

2.2.2 Geometrical Features

A quantitative way to describe the irregularity of a shape is to measure the variation of the boundary, which can be represented by a chain of vectors. The most common mechanisms measuring this variation use the length of the vectors, the angles between the vectors, and the distance (radii) from the centroid of the shape to the boundary. For a regular shape, there should be little variation in these measurements, but with increasing irregularity there will be greater variation. Attneave [14] found that the number of turns, the angular variability and the appearance of symmetry are correlated with human judgement of the complexity of a shape. Lee [54] developed a dissimilarity measurement of polygons from the dissimilarity of two triangles. Dissimilarity of two triangles, ΔABC and ΔXYZ, is defined as the minimum of the 6 possibilities of $|A-X| + |B-Y| + |C-Z|$. A, B, C and X, Y, Z are the angles of the two triangles respectively. He used it for chromosomal classification. Young [13] introduced a measure of rate of angular change.

Sklansky [11] suggested a measurement of concavity. This measurement consists of two parts: (1) the concave area and (2) the depth of each of the concave parts. For a shape of 4 concavities, the measurement consists of one value for the area and 4 depths for the 4 concavities. The concavities may imply the overlap of convex shapes [12].

Some quantitative measurements of shape complexity are often used for biological patterns. Kiefer [15] used a measurement D to reflect the compactness. $D = 2\pi \sum_{i=1} d_i^2/A$. For each point i in a shape, the Euclidean distance d_i from the arithmetic mean of all the shape points is squared and summed. In other words, $d_i^2 = (X_i-\overline{X})^2 + (Y_i-\overline{Y})^2$. Rosenfeld [69,70], Bacus and Gose [8], and Green [9] studied another measurement of compactness, $(\text{Perimeter})^2/\text{area}$. The area, spicularity (number of spicules on the boundary) and eccentricity (ratio of eigenvalues of the pixel distribution) were also used to classify red blood cells [53].

For some particular application, special features are used. Kruger et. al. [75] used 8 special features to describe the size and shape of a heart for radiograph diagnosis. The machine is told to find the cardiac area, draw a chest midline which is a vertical line dividing the thoracic cavity into two more or less equal areas, and then measure the 8 features. The chest midline divides the cardiac area into two parts. The 8 features are the maximum lengths, diagonals and areas of the two parts and the whole area. These features are also used by other researchers [68] for cardiac size and shape description.

Chien and Fu [68] used the coefficient of the polynomial which fit the boundary (by the least-square fitting method) to describe the shape of the heart in a chest x-ray picture. Matson et. al. [10] also used

the least square fitting method to find the best fitted ellipse in studying material characteristics. From the ellipse, the orientation and the aspect ratio of a shape can be obtained. Aspect ratio is the ratio of the major axis to the minor axis of the ellipse.

Among the features discussed above, some are based on angles and angular changes while some are based on concavities. Some features reflect the global complexities of a shape. Some features are extracted from the smoothed curve which fits the boundary. Chien and Fu [68] pointed out that there is no general theory for selecting features. Therefore, the heuristic approach is usually taken to select features based on the researchers' knowledge about the picture patterns and the differences between classes of the picture patterns. However, there are two analytic feature families, Fourier descriptors and moments. They will be discussed in the following two sections.

2.2.3 Fourier Descriptor

Fourier transformation has been used for preprocessing [19,22], description and classification [17,18,20]. It is one of the most popular techniques in picture processing [17-20] mainly because (1) it is a complete transformation, and (2) some values in the frequency domain are invariant with respect to translation and rotation in the time domain. The transformation is complete in the sense that no information is lost in the operation, since the original data in the time domain can be obtained by applying the inverse transformation to the frequency domain. The invariant values in the frequency domain can be used as features for classification. In addition to these merits, the Fast Fourier Transform, FFT, reduces the computational cost a great deal [16,21].

Fourier transformation can be applied to a one-dimensional boundary or to a two-dimensional picture. A digital picture can be considered as a two-dimensional array. The two-dimensional Fourier transformation is done by simply applying the one-dimensional Fourier transformation to the array horizontally and vertically in sequence [22]. Since we are concentrating on the shape, which is the boundary of an object on the picture, we will discuss only the one-dimensional Fourier transformation applying to the shape. There are at least three different definitions of a Fourier descriptor:

a. Suppose that the closed contour has length L. $\phi(\ell)$, defined along the contour, is the net amount of angular bend between the starting point $\ell=0$ and ℓ, $0 \leq \ell \leq L$. We then define [18]

$$\phi^*(t) = \phi(\frac{Lt}{2\pi}) + t \quad , \quad 0 \leq t \leq 2\pi \ .$$

$\phi^*(t)$ is invariant with respect to translation, rotation, and scaling. Expanding $\phi^*(t)$ in the Fourier series gives

$$\phi^*(t) = \mu_0 + \sum_{k=1}^{\infty} A_k(kt - \alpha_k)$$

or

$$\phi^*(t) = \mu_0 + \sum_{k=1}^{\infty} (a_k \cos kt + b_k \sin kt)$$

Then, $\{A_k, \alpha_k | k=1 \cdots \infty\}$ or $\{a_k, b_k | k=1 \cdots \infty\}$ is the Fourier descriptor. When the starting point changes, the phase angle of the descriptors changes, while the magnitude of the coefficients remains

invariant. Zahn and Roskies [18] investigated the relationship between Fourier descriptors and the geometry of shapes. They also studied the theories for shape generation from arbitrary Fourier descriptors.

b. If $X(\ell)$ and $Y(\ell)$, with $0 \leq \ell \leq L$ are the coordinate functions of the points on the closed contour, then a complex function, $U(\ell)$, and a complex integration, a_n, can be defined as follows [17]:

$$U(\ell) = X(\ell) + jY(\ell)$$

$$a_n = \frac{1}{L}\int_0^1 U(\ell)\, e^{-j\frac{2\pi}{L}n\ell}\, d\ell$$

Then $\{a_n | n=-\infty \cdots \infty\}$ is the Fourier descriptor. And

$$U(\ell) = \sum_{n=-\infty}^{\infty} a_n\, e^{-jn\frac{2\pi}{L}\ell} .$$

Persoon and Fu [17] discovered that the mean square distance between two Fourier descriptors equals the mean square distance between the two corresponding curves. Also, they found that the skeleton of a shape can be extracted from Fourier descriptors. Granlund [19] applied this technique to character recognition. Richard and Hemami [20] used it for three-dimensional object identification. Wallace and Wintz [21] used interpolation techniques to obtain a desired number of boundary points. The number of boundary points has to be a power of 2, so that FFT can be applied. Wallace and Wintz [21] also studied a procedure for normalizing size, starting point, and

translation.

c. For simple shapes, there is another definition for a Fourier descriptor. Let $\gamma(\theta)$ be the distance function from a contour point at angle θ to the centroid. $\gamma(\theta)$ can be expanded as follows [23]

$$\gamma(\theta) = a_0 + \sum_{k=1}^{\infty} (a_k \cos k\theta + b_k \sin k\theta)$$

Rutovitz [23] found that the first six terms of a_k's and b_k's were adequate to represent a chromosome for discrimination. But this definition is restricted to shapes which can be represented by single valued $\gamma(\theta)$ functions.

2.2.4 Moment Method

Method of moment [24,25,26] is one of the earliest techniques studied. For a binary digital image, the moments with respect to the centroid can be defined as follows [6]:

$$\mu_{pq} = \frac{1}{N} \sum_{i=1}^{N} (U_i - \bar{U})^p (V_i - \bar{V})^q$$

where $\bar{U}$ and $\bar{V}$ are the mean values of the image coordinates U and V, respectively, and the summation is over all the image points of the shape. They can be only the contour points or the body points. Hu [25] derived a set of moment functions which have the desired property of invariance under image translation or rotation. These functions were therefore called moment invariants. Moment invariants were further investigated for 3-dimensional object identification. Dudani [6] selected

7 moment invariants for the contour and the body of the shape respectively. All 14 feature values were used for classifying aircrafts. Liu [27] combined features selected from Fourier descriptors, moments, and geometrical measurements to form a better feature set for white blood cell classification. Mui, Fu, and Bacus [61,62] selected 17 features from a set of 367 features to classify white blood cell neutrophils into band neutrophils and segmented neutrophils.

All the statistical methods discussed in the Section 2.2.1-2.2.4 attempt to find those features which are insensitive to rotation, translation, and scaling of the pattern so that the image pattern can be represented by a set of n features. In other words, the pattern can be represented as a point in the n-dimensional feature space. The points corresponding to patterns of the same class are usually assumed to be close together in the feature space while those of different classes are assumed to not be close together. Thus, the feature space can be partitioned into many regions corresponding to different classes. Then, the classification problem becomes a problem of obtaining discriminant functions which can well partition the feature space. The unknown pattern whose corresponding point falls in one of the regions is assigned to the class corresponding to that region.

Statistical theories [2] prove that the Bayesian classifier, the maximum likelihood classifier, is the minimum error classifier. If the distribution of the patterns in the feature space is parametrical, the classification becomes easier after the parameters of the distribution are obtained. Some applications assuming Gaussian distribution achieved reasonably good results. The learning techniques, or estimation, of

Gaussian parameters can be found in [1]. Unfortunately, the assumption of parametrical distribution lacks theoretical foundations. The assumption may not be true, if a large dimensionality pattern vector is required to describe very complex shapes. The situation may be worse, if the complex shapes do not differ significantly for different classes.

People studied the discriminant functions for non-parametric distributions [1,2,3]. Linear, piecewise linear, or nonlinear discriminant functions can be obtained iteratively from the training patterns. But the learning procedure is time consuming. The training time increases rapidly with the complexity of the function and with the number of training samples. The K-nearest-neighbor method is often used to solve the classification problem of non-parametrical distributions. All the known vector patterns are stored in the computer. For an unknown vector pattern, the computer finds the K nearest known patterns in its neighborhood and then assigns the unknown to the majority class of these K nearest neighbors. Although this method does not require training time, the classification time and storage for known patterns increase with K and the number of known patterns. In fact, the K-nearest-neighbor method is a more intelligent template matching technique. It can also be considered as a combination of the statistical method and template matching method.

2.3 Syntactic Method

The structural property of a contour is usually a deciding factor in recognition. In the statistical method, some features are so designed that they will implicitly reflect the global structure of the shape by numerical values. For example, the Fourier descriptor can ex-

press the global structure in terms of values in the frequency domain. The syntactic method [4,5] can utilize the production rules to explicitly describe the structure of the contour, if appropriate primitives can be found. Generally speaking, syntactic pattern recognition consists of three steps: preprocessing, primitive extraction, and syntax analysis. Figure 2.3 is a diagram of the system.

For shape recognition problems, we can illustrate the syntactic method in terms of an example of chromosome recognition [5]. See Figure 2.4. Let us suppose the smooth boundary of a submedian chromosome is obtained after the preprocessing stage. The machine is given the specifications of the primitives a, b, c, d. The specifications may be numerical attributes such as length, angle, radii, etc. The computer processes the boundary to extract the primitives which fit the specifications and represents the boundary in terms of a concatenated string of primitives. Thus, the pattern is treated as a sentence or a string. Then, the machine analyzes the syntax of the string according to a given grammar. We assume that there is a grammar for each class. The grammar can generate all the possible patterns in the class. A syntax analyzer is called a parser. A parser is used to find a derivation which derives the string through the use of the production rules. The unknown pattern is accepted by a class, if a derivation tree is found with respect to the corresponding grammar. Because of the capability of structural description, the syntactic approach has received more and more attention in recent years [28,29,57].

In addition to chromosome recognition [5,35], syntactic methods have many applications in picture recognition. Shaw [36,37] developed

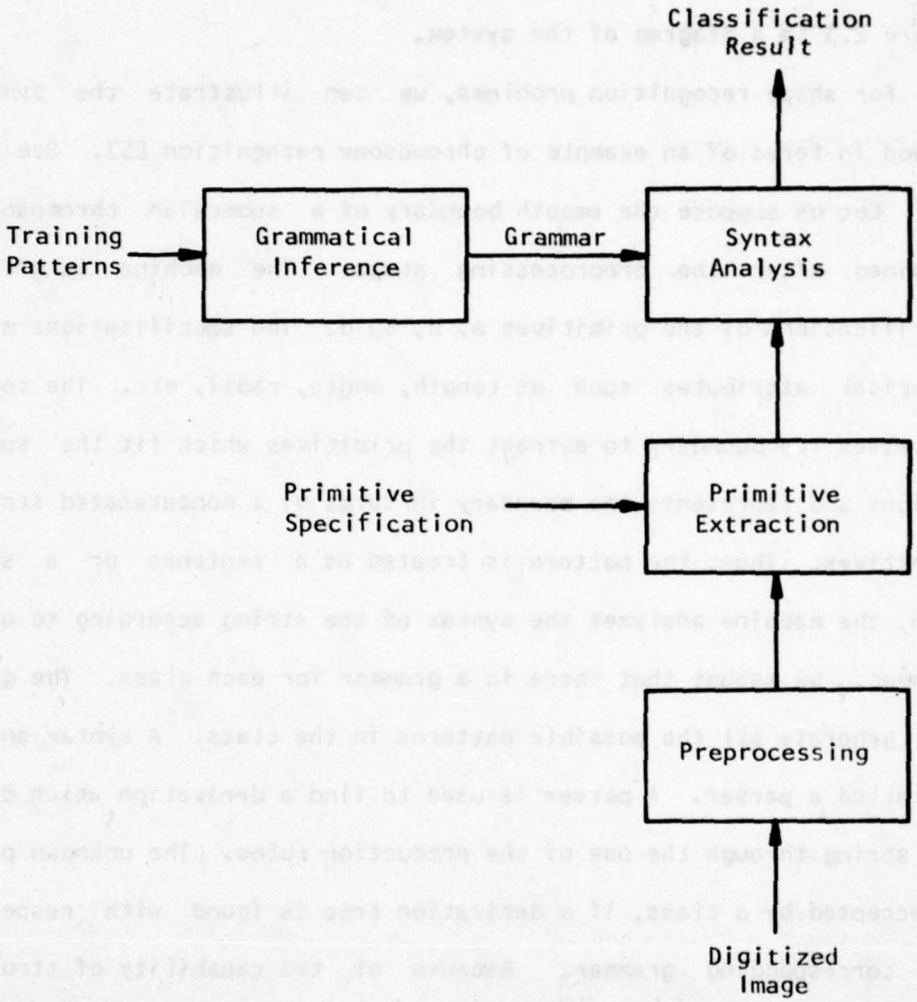


Figure 2.3 A Syntactic Pattern Recognition System

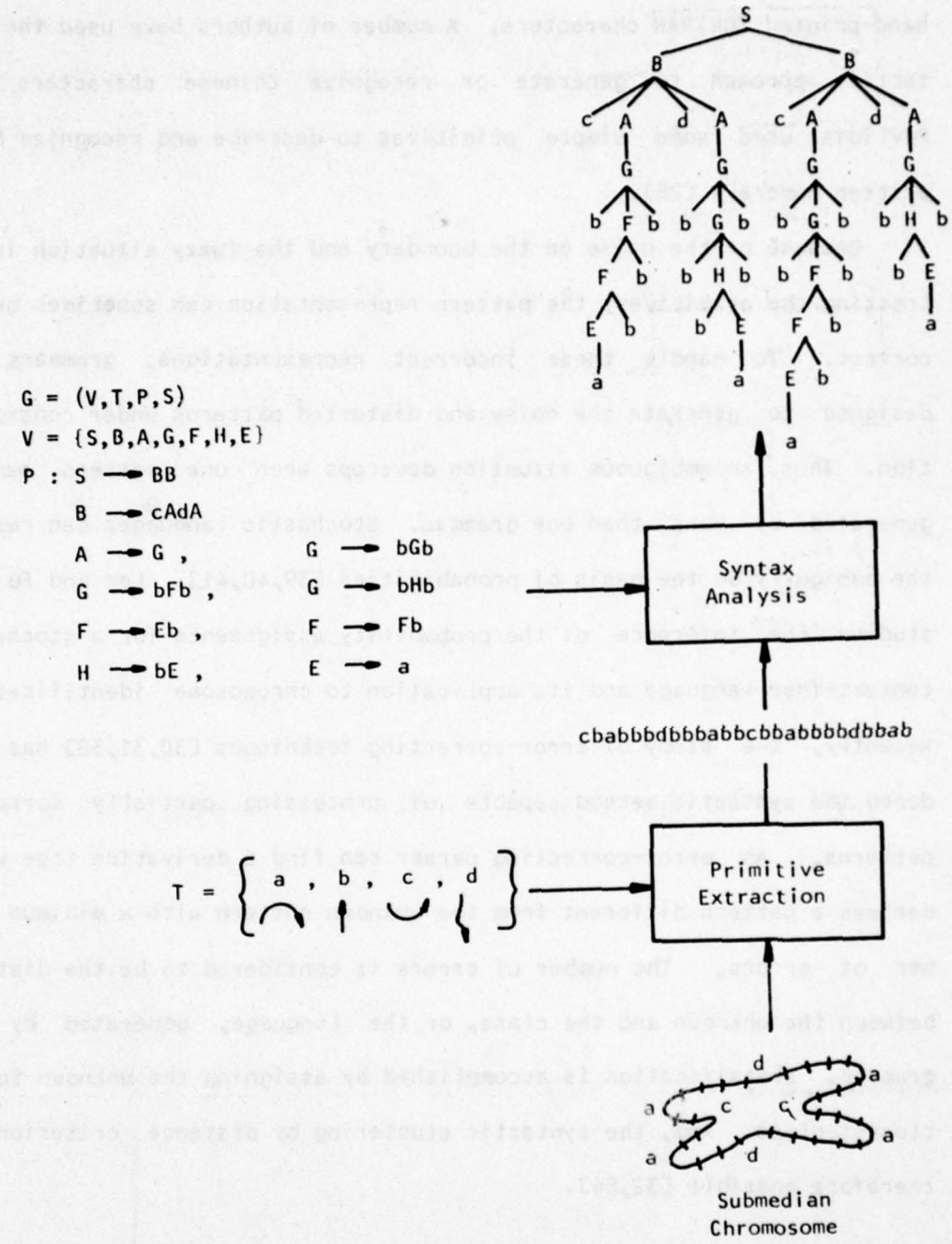


Figure 2.4 A Syntactic Chromosome Recognition System

Picture Description Language. Narasimham [38] proposed a grammar for hand-printed FORTRAN characters. A number of authors have used the syntactic approach to generate or recognize Chinese characters [5]. Pavlidis used some simple primitives to describe and recognize hand-written numerals [28].

Because of the noise on the boundary and the fuzzy situation in extracting the primitives, the pattern representation can sometimes be incorrect. To handle these incorrect representations, grammars are designed to generate the noisy and distorted patterns under consideration. Thus, an ambiguous situation develops when one pattern can be generated by more than one grammar. Stochastic languages can resolve the ambiguity on the basis of probabilities [39,40,41]. Lee and Fu [58] studied the inference of the probability assignments for a stochastic context-free language and its application to chromosome identification. Recently, the study of error-correcting techniques [30,31,32] has rendered the syntactic method capable of processing partially corrupted patterns. An error-correcting parser can find a derivation tree which derives a pattern different from the unknown pattern with a minimum number of errors. The number of errors is considered to be the distance between the unknown and the class, or the language, generated by the grammar. Classification is accomplished by assigning the unknown to the closest class. And, the syntactic clustering by distance criterion is therefore possible [32,84].

pears. This kind of distortion in the picture may demage the Fourier descriptors or moment invariants completely because the distortion may not only affect some boundary details but may also change the overall shape structure. Humans can still recognize the object by seeing only the unaltered part of the shape.

It appears that the syntactic method is the most promising method, since it uses production rules to explicitly describe the structure of a shape and primitives to describe the local boundary details. For a partially demaged shape, error-correcting techniques can be used to recognize the object by the unaltered part of the shape and to potentially correct the distorted part of the shape.

2.4 Conclusion

In the previous sections of this chapter, most of the existing shape recognition methods were discussed. The primitive and restricted prototype matching technique is useful in recognizing machine-printed characters. But it does not satisfy the needs of a higher intelligent recognizer. The more sophisticated skeleton matching method can be used to recognize noise-free and structurally different images. It may not work with the shapes which differ only in some local details for different classes. The statistical method is the most popular technique used. This approach is effective under two conditions: (1) when we can select appropriate features which are sensitive to the class differences and insensitive to noise, and (2) when the discriminant surfaces in the feature space can be easily and correctly obtained. Unfortunately, the features are usually difficult to select and are dependent upon particular applications. The pattern distributions are generally non-parametric. This increases the difficulty of finding the discriminant surfaces. Perhaps, the Fourier descriptors and moment invariants are the best features for rigid-body objects. But they are not so effective when noise is present. We can categorize noise into two kinds by its nature: random noise and non-random distortion. Random noise has been studied for decades. In imagery data, it can be taken care of by using digital filtering techniques [22,90] in the preprocessing stage. For detecting shape boundary, many techniques have been developed [64-67]. Non-random distortion may appear in any application. For example, an airplane may be partially covered by clouds or an automobile may be behind a tree. Some aspect of the shape is badly distorted or disap-

pears. This kind of distortion in the picture may demage the Fourier descriptors or moment invariants completely because the distortion may not only affect some boundary details but may also change the overall shape structure. Humans can still recognize the object by seeing only the unaltered part of the shape.

It appears that the syntactic method is the most promising method, since it uses production rules to explicitly describe the structure of a shape and primitives to describe the local boundary details. For a partially demaged shape, error-correcting techniques can be used to recognize the object by the unaltered part of the shape and to potentially correct the distorted part of the shape.

CHAPTER 3

SHAPE DESCRIPTION AND RECOGNITION USING ATTRIBUTED GRAMMARS

3.1 Introduction

This chapter contains the main portion of our proposed method. The syntactic method can explicitly describe the global structure by grammar rules, and the local boundary details by primitives with numerical attributes or features. These associated attributes carry the semantic information of the primitives. As indicated in the previous chapters, the proposed method attempts to solve general shape recognition problems. In other words, we need to develop a method which can describe arbitrary shapes, and a technique which can recognize the shapes from their descriptions.

In order to avoid the need for a context-sensitive grammar, we try to use more sophisticated primitives. To describe an arbitrary shape, we need to consider a large number of possible primitives, although we may not need all of them for one shape. To extract the primitive from a shape, the machine must know the specifications of every primitive used. It is obvious that we need a systematic way to describe the large number of possible primitives. Though the number of primitives used in any particular applications is finite, the number of possible primitives for arbitrary shapes may be very large unless we restrict ourselves to spe-

cial types of primitives such as unit vectors, pixels, Freeman chain codes [42], etc.

Therefore, in Section 3.2 we proposed a new descriptive method which uses four features to describe an arbitrary curve. The four features range continuously on the real line. That is, there are an infinite number of feature values. The additive property of the primitive descriptors was discovered and will be discussed. In Section 3.3, the exact and efficient equations for computing features in discrete cases are derived by applying the additive property. In Section 3.4, shape grammars are explained and illustrated by examples. In this section we can see the advantage of using our more sophisticated primitives, i.e., the simplicity of the shape grammars.

Following the descriptive method, we will investigate the recognition techniques in subsequent sections. In Section 3.5, we first introduce a transformation which maps the four features in a descriptor to another set of four variables. These four variables constitute a transformed descriptor which is theoretically invariant with respect to translation, rotation, and scaling. Then we will study the effect of digitization noise on the transformed descriptor. In the noise study, we could construct a recognition function of primitives. But, many confusing situations may occur when extracting primitives due to noise or similarity of different parts. These sources of confusion can be resolved by using the context information which is used in parsing. In Section 3.6, we introduce the concept of primitive-extraction-embedding and develop two parsing algorithms which embed the primitive extraction. Because the definition of primitive is flexible and the specification of

primitive is not unique, a classification tree scheme is possible for a multi-class problem. The feasibility of this is discussed in Section 3.7. This chapter concludes with a discussion.

3.2 Primitives and Attributes

Section 2.3 describes syntactic recognition with a chromosome example. It is not difficult to see that the four primitives selected for the submedian chromosome are insufficient for describing an arbitrary shape. In applying the syntactic method to an arbitrary shape problem, we encountered three problems: (1) what are the appropriate primitives? (2) how do we specify the primitives? and (3) how do we describe the shape by production rules with these primitives?

The meaning of a primitive may lead us to use the simplest elements of the shape as primitives. For example, unit vectors, pixels, Freeman chain codes [42], etc., can be used to describe the shape. But, the simpler the primitives are, the more complex the grammars must be. Consequently, more complicated parsing procedure and more computation time are required. If fixed length curve segments are used as primitives, e.g., unit vectors, we may need context-sensitive grammars to handle the scaling problem. The parsing of context-sensitive grammars is difficult and should be avoided whenever possible. Therefore, we propose to use more sophisticated primitives and simpler grammars to tackle shape description and recognition problems.

Since a curve can be characterized by its curvature function [92], we have the following definition. Note, the curvature function $f(\ell)$ of a curve segment is the derivative of the direction along the curve segment with respect to length. That is,

$$f(\ell) = \lim_{\Delta\ell \to 0} \frac{1}{\Delta\ell} \left(\begin{array}{l} \text{angle between the tangent lines to the} \\ \text{curve segment at } \ell - \frac{1}{2}\Delta\ell \text{ and } \ell + \frac{1}{2}\Delta\ell \end{array} \right)$$

Definition 3.1: A curve segment, $\overset{\frown}{X_1 X_2}$, is a directional line with a starting point X_1 and an ending point X_2. The curve segment has a curvature function, $f(\ell)$, along the direction with $0 < \ell < L$, where L is the total length of the curve segment.

To simplify our analysis, we define a simple curve segment as follows.

Definition 3.2: A simple curve segment is a curve segment with either $f(\ell) \geq 0$, or $f(\ell) \leq 0$, for $0 < \ell < L$. See Figure 3.1 for illustrations.

The curvature function of a smooth curve is a continuous function in a continuous case, but it is a summation of pulses in a discrete case. A curve segment is clockwise if $f(\ell) \geq 0$, and counter-clockwise if $f(\ell) \leq 0$.

To allow many possible simple curve segments, we use a finite number of feature values to characterize a simple curve segment. Intuitively, we would use the length and the angular change of the curve segment as features. But sometimes, we would like to distinguish between two curves with the same length and angular change but different lengths of the vector which connects the two ends of the curve. See Figure 3.2 for an illustration.

In addition to the above three features, we need a measure of symmetry to resolve some ambiguities. For example, the shape of a heart is decomposed into two parts. (See Figure 3.3). The two parts are mirror images of each other. If the two parts are rotated and aligned with the vectors as in Figure 3.3(c), we can see their different declinations. A

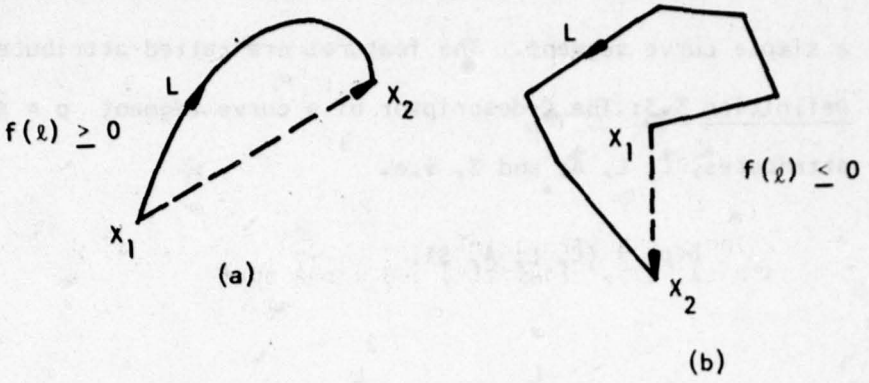


Figure 3.1 Two Simple Curve Segments

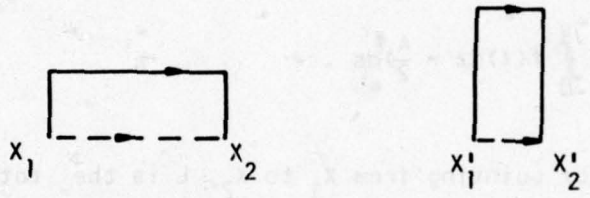


Figure 3.2 Two Curves Have The Same Curve Length, Same Angle But Different Vector Lengths

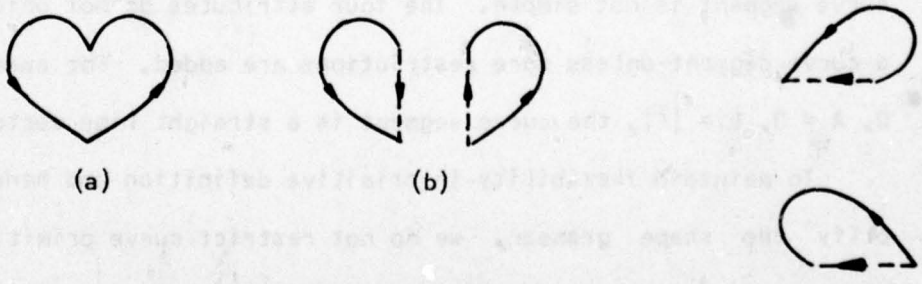


Figure 3.3 Decomposition of a Heart

symmetric measurement, S, is needed to reflect the declination of a curve. We found that these four features can sufficiently characterize a simple curve segment. The features are called attributes.

Definition 3.3: The C-descriptor of a curve segment $p = \overset{\frown}{X_1X_2}$ has four attributes, $\vec{C}$, L, A, and S, i.e.

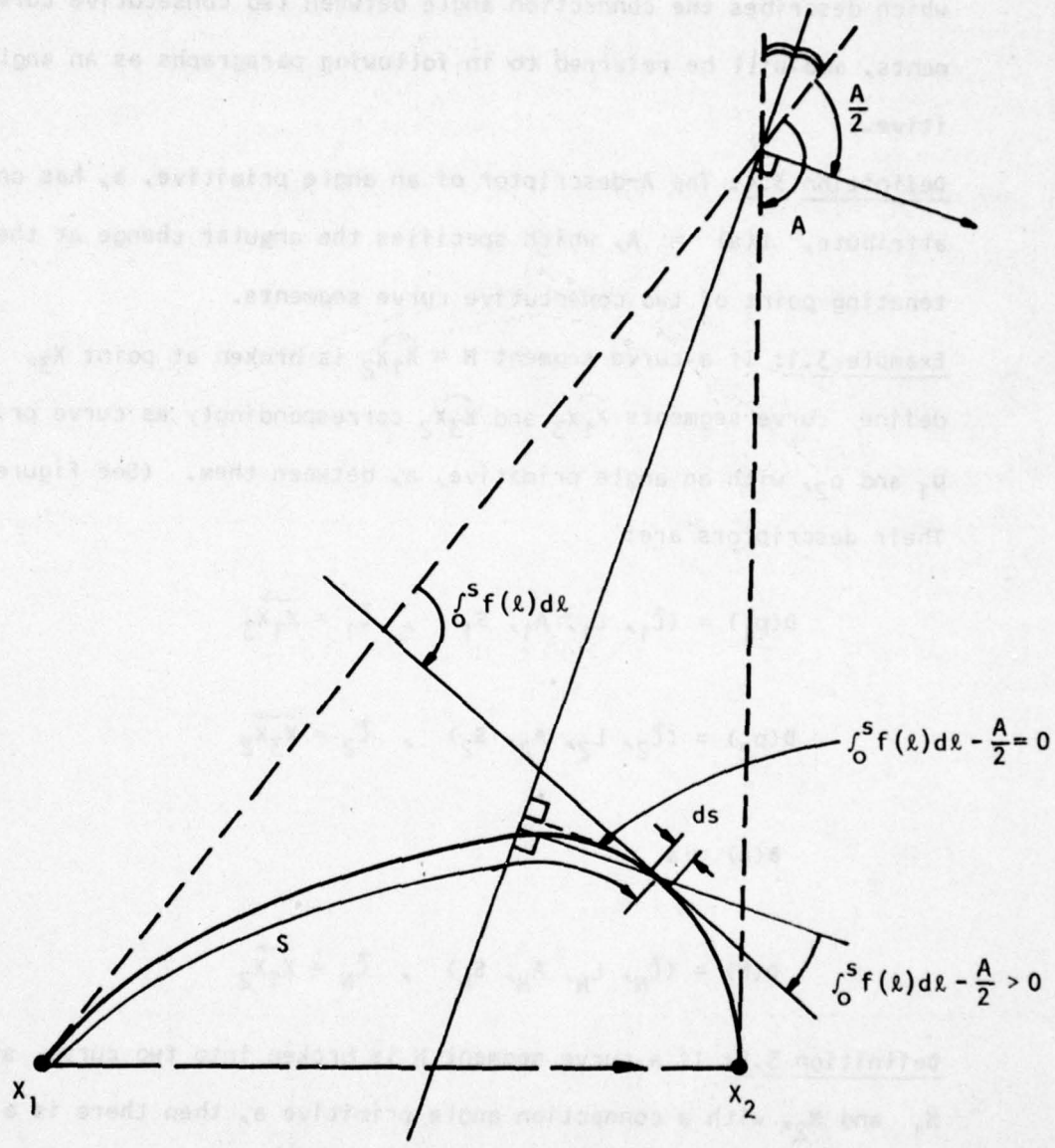
$$D(p) = (\vec{C}, L, A, S).$$

Where $\vec{C} = \overrightarrow{X_1X_2}$, $L = \int_0^L d\ell$, $A = \int_0^L f(\ell)d\ell$,

and $S = \int_0^L (\int_0^s f(\ell)d\ell - \frac{A}{2})ds$.

$\vec{C}$ is the vector pointing from X_1 to X_2, L is the total length of the curve, A is the total angular change, and S measures the symmetry. Figure 3.4 illustrates the function of S. When p is symmetric, S = 0. If p is not symmetric, then S > 0 when p is declined to the left, and S < 0 when p is declined to the right. Somehow, S measures the degree of declination. But, the S measurement becomes less meaningful when the curve segment is not simple. The four attributes do not uniquely define a curve segment unless more restrictions are added. For example, if S = 0, A = 0, L = $|\vec{C}|$, the curve segment is a straight line vector, $\vec{C}$.

To maintain flexibility in primitive definition and hence to simplify the shape grammar, we do not restrict curve primitives to only simple curve segments, although the primitives are essentially simple curve segments. Thus, a curve primitive consists of one or more simple



$$S = \int_0^L \left(\int_0^s f(\ell)d\ell - \frac{A}{2}\right)ds$$

Figure 3.4 An Illustration for Attribute S

curve segments.

In addition to the curve primitives, we need a relation to describe the connection between curve segments. This relation has one attribute which describes the connection angle between two consecutive curve segments, and will be referred to in following paragraphs as an angle primitive.

Definition 3.4: The A-descriptor of an angle primitive, a, has only one attribute, $D(a) = A$, which specifies the angular change at the concatenating point of two consecutive curve segments.

Example 3.1: If a curve segment $N = \widehat{X_1X_2}$ is broken at point X_3, we may define curve segments $\widehat{X_1X_3}$ and $\widehat{X_3X_2}$ correspondingly as curve primitives p_1 and p_2, with an angle primitive, a, between them. (See Figure 3.5). Their descriptors are:

$$D(p_1) = (\vec{t}_1, L_1, A_1, S_1) \quad , \quad \vec{t}_1 = \overrightarrow{X_1X_3}$$

$$D(p_2) = (\vec{t}_2, L_2, A_2, S_2) \quad , \quad \vec{t}_2 = \overrightarrow{X_3X_2}$$

$$D(a) = a$$

$$D(N) = (\vec{t}_N, L_N, A_N, S_N) \quad , \quad \vec{t}_N = \overrightarrow{X_1X_2}$$

Definition 3.5: If a curve segment N is broken into two curve segments, N_1 and N_2, with a connection angle primitive a, then there is a production rule, $N \rightarrow N_1aN_2$.

Their descriptors have an interesting additive property.

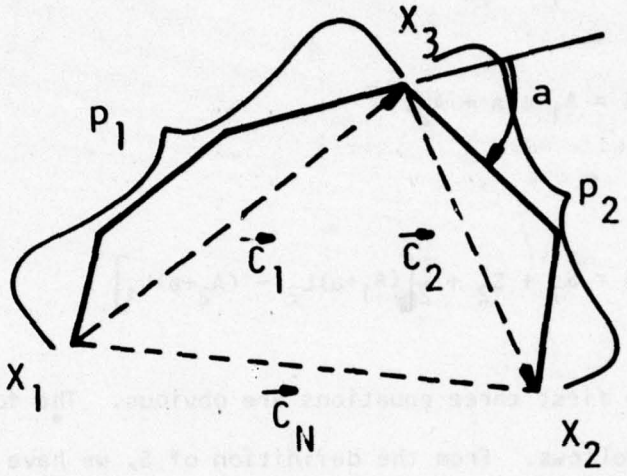


Figure 3.5 A Curve Segment Is Broken Into Two Shorter Curve Segments

Theorem A: Additivity.

If $N \rightarrow N_1 a N_2$ with descriptors $D(N) = (\check{t}, L, A, S)$, $D(N_1) = (\check{t}_1, L_1, A_1, S_1)$, $D(N_2) = (\check{t}_2, L_2, A_2, S_2)$ and $D(a) = a$ then

$$\check{t} = \check{t}_1 + \check{t}_2,$$

$$L = L_1 + L_2,$$

$$A = A_1 + a + A_2,$$

and

$$S = S_1 + S_2 + \frac{1}{2}\left[(A_1+a)L_2 - (A_2+a)L_1\right]$$

Proof: The first three equations are obvious. The fourth equation is proved as follows. From the definition of S, we have

$$S_1 = \int_0^{L_1} \left(\int_0^s f_1(\ell)\, d\ell\right) ds - \frac{1}{2} A_1 L_1$$

$$S_2 = \int_0^{L_2} \left(\int_0^s f_2(\ell)\, d\ell\right) ds - \frac{1}{2} A_2 L_2$$

$$S = \int_0^{L_1+L_2} \left(\int_0^s f(\ell)\, d\ell\right) ds - \frac{1}{2} (A_1+a+A_2)(L_1+L_2)$$

where f_1, f_2, and f are curvature functions of N_1, N_2, and N, respectively, and

$$f(\ell) = \begin{cases} f_1(\ell) & 0 < \ell < L_1 \\ a_1 & \ell = L_1 \\ f_2(\ell - L_1) & L_1 < \ell < L_1 + L_2 \end{cases}$$

$$S = \int_0^{L_1^-} \left(\int_0^s f(\ell)d\ell \right) ds + \int_{L_1^-}^{L_1+L_2} \left(\int_0^s f(\ell)d\ell \right) ds - \frac{1}{2}(A_1+a+A_2)(L_1+L_2)$$

$$= S_1 + \int_{L_1^+}^{L_1+L_2} \left[\left(\int_{L_1^+}^s f_2(\ell)d\ell \right) + A_1 + a \right] ds$$

$$- \frac{1}{2}(L_1A_2 + L_1a + L_2A_1 + L_2A_2 + L_2a)$$

$$\because \int_{L_1^-}^{L_1^+} \left(\int_0^s f(\ell)d\ell \right) ds = 0, \text{ Since} \int_0^s f(\ell)d\ell \text{ is finite.}$$

After substituting S_2 into S, we obtain

$$S = S_1 + S_2 + \frac{1}{2}[(A_1+a)L_2 - L_1(A_2+a)]$$

Q.E.D.

Definition 3.6: $D(N_1) \oplus_a D(N_2)$ denotes the above additivity.

Corollary A.1: If $N \rightarrow N_1 a N_2$ then $D(N) = D(N_1) \oplus_a D(N_2)$.

Theorem B: The addition, $\oplus_a$, is associative.

If $N \rightarrow N_1 a_1 N_2 a_2 N_3$, then

$$D(N) = D(N_1) \oplus_{a_1} D(N_2) \oplus_{a_2} D(N_3)$$

$$= [D(N_1) \oplus_{a_1} D(N_2)] \oplus_{a_2} D(N_3)$$

$$= D(N_1) \oplus_{a_1} [D(N_2) \oplus_{a_2} D(N_3)]$$

The proof is obvious.

3.3 Computation of C-Descriptors in a Discrete Case

The above C-descriptors are defined in a continuous case, but they can be computed efficiently in a discrete case by using only addition and multiplication. We first derive two more corollaries from Theorem A.

Corollary A.2: If N_2, in Theorem A, is a straight line vector, i.e. $A_2 = 0$, $S_2 = 0$, then

$$\vec{t} = \vec{t}_1 + \vec{t}_2 \ , \quad L = L_1 + L_2 \ , \quad A = A_1 + a,$$

$$\text{and } S = S_1 + AL_2 + \frac{1}{2} A_1 L_1 - \frac{1}{2} AL$$

Corollary A.3: If N_1 is also a straight line vector, i.e. $A_1 = 0$, $S_1 = 0$, then

$$\vec{t} = \vec{t}_1 + \vec{t}_2 \ , \quad L = L_1 + L_2 \ , \quad A = a, \text{ and}$$

$$S = aL_2 - \frac{1}{2} a(L_1 + L_2) = AL_2 - \frac{1}{2} AL$$

Theorem C: Recursive Equations for C-Descriptors

$M = (v_1 v_2 \cdots v_m)$ is a vector chain. Let M_i denote $(v_1 \cdots v_i)$, i.e. $M_m = M$, a_i is the angle between v_i and v_{i+1}, $D(v_i) = (\overset{+}{C}_i, L_i, 0, 0)$, $i \leq i \leq m$. Then $D(M_j) = (\overset{+}{C}_{Mj}, L_{Mj}, A_{Mj}, S_{Mj})$, $1 \leq j \leq m$, where

$$\overset{+}{C}_{Mj} = \overset{+}{C}_{M(j-1)} + \overset{+}{C}_j = \sum_{i=1}^{j} \overset{+}{C}_i ,$$

$$L_{Mj} = L_{M(j-1)} + L_j = \sum_{i=1}^{j} L_i$$

$$A_{Mj} = A_{M(j-1)} + a_{j-1} = \sum_{i=1}^{j-1} a_i ,$$

$$S_{Mj} = G_{Mj} - \frac{1}{2} A_{Mj} L_{Mj}$$

$$G_{Mj} = G_{M(j-1)} + A_{Mj} L_j = \sum_{i=1}^{j} A_{Mi} L_i = \sum_{i=1}^{j} \sum_{\ell=1}^{i-1} a_\ell L_i$$

Proof:

(1) When $j = 1$, it is obvious

$$D(M_1) = (\overset{+}{C}_1, L_1, 0, 0)$$

(2) When $j = 2$, according to Corollary A.3

$$D(M_2) = (\check{t}_{M2}, L_{M2}, A_{M2}, S_{M2})$$

$$\check{t}_{M2} = \check{t}_1 + \check{t}_2 = \sum_{i=1}^{2} \check{t}_i$$

$$L_{M2} = L_1 + L_2 = \sum_{i=1}^{2} L_i$$

$$A_{M2} = a_1$$

$$S_{M2} = G_{M2} - \frac{1}{2} a_1 (L_1 + L_2) = G_{M2} - \frac{1}{2} A_{M2} L_{M2}$$

and

$$G_{M2} = a_1 L_2$$

(3) When $j = 3$, according to Corollary A.2, we have

$$D(M_3) = (\check{t}_{M3}, L_{M3}, A_{M3}, S_{M3})$$

$$\check{t}_{M3} = \check{t}_{M2} + \check{t}_3 = \sum_{i=1}^{3} \check{t}_i$$

$$L_{M3} = L_1 + L_2 = \sum_{i=1}^{3} L_i$$

$$A_{M3} = a_1 + a_2 = \sum_{i=1}^{2} a_i$$

$$S_{M3} = G_{M3} - \frac{1}{2} A_{M3} L_{M3}$$

and

$$G_{M3} = S_{M2} + A_{M3} L_3 + \frac{1}{2} A_{M2} L_{M2}$$

$$= G_{M2} + A_{M3} L_3$$

(4) If the equations are true for $j = k < m$, according to Corollary A.2, when $j = k+1$, we have

$$\tilde{c}_{M(k+1)} = \tilde{c}_{Mk} + \tilde{c}_{k+1} = \sum_{i=1}^{k} \tilde{c}_i + \tilde{c}_{k+1} = \sum_{i=1}^{k+1} \tilde{c}_i$$

$$L_{M(k+1)} = L_{Mk} + L_{k+1} = \sum_{i=1}^{k} L_i + L_{k+1} = \sum_{i=1}^{k+1} L_i$$

$$A_{M(k+1)} = A_{Mk} + A_k = \sum_{i=1}^{k-1} A_i + A_k = \sum_{i=1}^{k} A_i$$

$$S_{M(k+1)} = S_{Mk} + A_{M(k+1)} L_{k+1} + \frac{1}{2} A_{Mk} L_{Mk} - \frac{1}{2} A_{M(k+1)} L_{M(k+1)}$$

$$= G_{M(k+1)} - \frac{1}{2} A_{M(k+1)} L_{M(k+1)}$$

By mathematical induction, the theorem is proven true for any m.

Q.E.D.

Theorem C contains the computation equations in a recursive form. The recursive relationships are reversible.

Theorem D: Reverse Relationships

$M = (v_1 v_2 \ldots v_m)$ is a vector chain. Let M_i denote $(v_1 \ldots v_i)$, (i.e., $M_m = M$). a_i is the angle between v_i and v_{i+1}, $D(v_i) = (\vec{C}_i, L_i, 0, 0)$ $1 \leq i \leq m$. Then $D(M_j)$ can be computed from $D(M_{j+1})$ and $D(v_{j+1})$, $1 \leq j \leq m$

$$\vec{C}_{Mj} = \vec{C}_{(Mj+1)} - \vec{C}_{j+1}$$

$$L_{Mj} = L_{M(j+1)} - L_{j+1}$$

$$A_{Mj} = A_{M(j+1)} - a_j$$

$$S_{Mj} = G_{Mj} - \frac{1}{2} A_{Mj} L_{Mj}$$

$$G_{Mj} = G_{M(j+1)} - A_{M(j+1)} L_{j+1}$$

The proof is obvious.

With Theorem C, the attributes can be computed exactly instead of approximately in a discrete case. In addition, Theorem C suggests two ways of computing the C-descriptors. For a boundary chain of m vectors, if there is enough memory to store all the descriptors calculated for later processing, we need to compute at most m(m+1)/2 possible C-descriptors. The recursive equations in Theorem C suggest that each C-

descriptor requires 2 multiplications, 5 additions, and 1 shift. This implies that it takes about m^2 multiplications to compute all the possible C-descriptors. But the equations of total summation in Theorem C suggests that attributes of any curve segment can be computed directly from the lengths and the connecting angles of the vectors of the corresponding chain. That implies another possible implementation. No memory for storing C-descriptors is necessary. The C-descriptor of any nonterminal or curve primitive can be obtained directly from the boundary chain, as long as the corresponding indices of the vector subchain are known. So, Theorem C suggests two different implementations for obtaining C-descriptors. There is a trade-off between the two implementations with regards to time and memory. In our experiment, we use the second implementation in order to have better control of the memory space.

3.4 Shape Grammars

Although a picture is two-dimensional, the outer boundary of an object in the picture is one dimensional. Thus, a one-dimensional string grammar is sufficient to describe shapes. We proposed to use the curve segments as curve primitives and connection angles as angle primitives in previous sections. If a shape is decomposed into proper curve segments and each curve segment is coded as a primitive, then a shape can be represented as a string of primitives, or a sentence. Though a shape is a closed curve, we can break it at some arbitrary point so that it can be described by a curve segment with the same starting and ending point and an angle primitive which specifies the angular change at that point. That arbitrary point can be the first point found in tracing the

boundary.

Because of the attributes associated with the primitives and their additivity, we propose to use attributed grammar [44] for shape description and recognition.

Definition 3.7: An attributed grammar is a grammar where (1) each primitive or nonterminal has a symbol part and a value part which may have several values called attributes, and (2) each symbolic production rule has a corresponding set of attribute rules which process the attributes related to the production rule in parsing.

Let us illustrate the attributed shape grammar with an airplane shape.

Example 3.2: A simple airplane shape (see Figure 3.6) can be decomposed into four parts: nose, left wing, right wing, and fuselage and tail, denoted as N_1, N_2, N_3, and N_4 respectively. N_4 is further decomposed into three parts: left side, tail, and right side, denoted as p_9, N_5, and p_{10} respectively. The nonterminals N_1, N_2, N_3, and N_5 can be further decomposed into primitives. For each decomposition, there is a symbolic production rule and a set of attribute rules. The angle primitives, a_i's, describe the connection angles. α labels the pattern. The shape grammar is as follows

$G = (V, T, P, S_\alpha)$

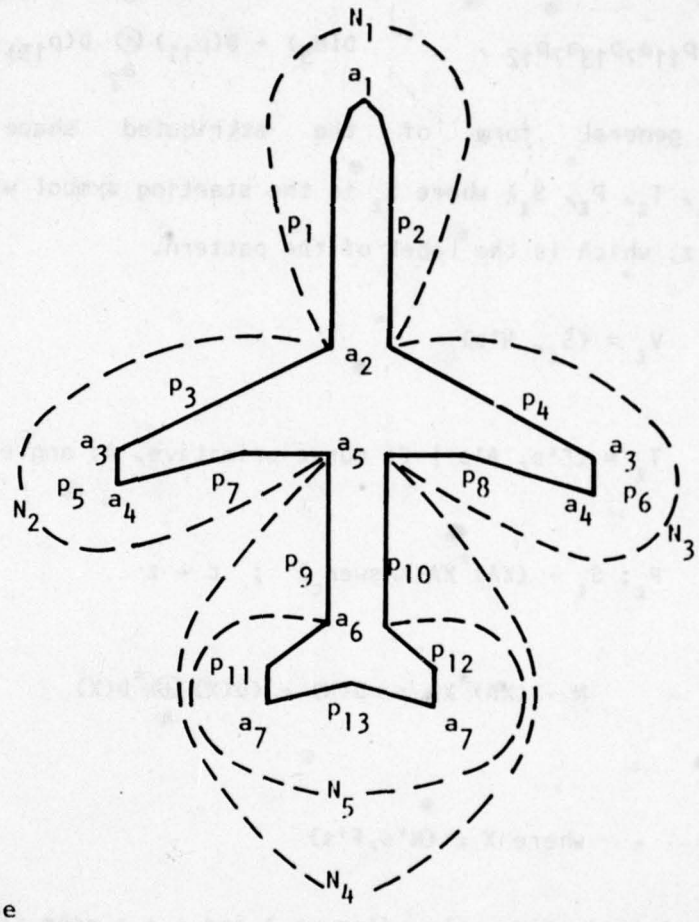
$V = \{S_\alpha, N_k \mid 1 \leq K \leq 5\}$

$T = \{a_i\text{'s}, p_j\text{'s} \mid 1 \leq i \leq 7,\ 1 \leq j \leq 12$

P:

$S_\alpha \rightarrow N_1 a_2 N_2 a_5 N_4 a_5 N_3 a_2$ {Answer$_c$} , $c \leftarrow \alpha$

$N_1 \rightarrow p_2 a_1 p_1$, $D(N_1) \leftarrow D(p_2) \underset{a_1}{\oplus} D(p_1)$



N_1: Nose

N_2: Left Wing

N_3: Right Wing

N_4: Fuselage and Tail

N_5: Tail

Figure 3.6 A Simple Airplane Shape

$N_2 \rightarrow p_3 a_3 p_5 a_4 p_7$, $\quad D(N_2) \leftarrow D(p_3) \underset{a_3}{\oplus} D(p_5) \underset{a_4}{\oplus} D(p_7)$

$N_3 \rightarrow p_8 a_4 p_6 a_3 p_4$, $\quad D(N_3) \leftarrow D(p_8) \underset{a_4}{\oplus} D(p_6) \underset{a_3}{\oplus} D(p_4)$

$N_4 \rightarrow p_9 a_6 N_5 a_6 p_{10}$, $\quad D(N_4) \leftarrow D(p_9) \underset{a_6}{\oplus} D(N_5) \underset{a_6}{\oplus} D(p_{10})$

$N_5 \rightarrow p_{11} a_7 p_{13} a_7 p_{12}$, $\quad D(N_5) \leftarrow D(p_{11}) \underset{a_7}{\oplus} D(p_{13}) \underset{a_7}{\oplus} D(p_{12})$

A general form of the attributed shape grammar is $G_\ell = (V_\ell, T_\ell, P_\ell, S_\ell)$ where S_ℓ is the starting symbol with a special attribute ℓ, which is the label of the pattern.

$$V_\ell = \{S_\ell, N's\}$$

$$T_\ell = \{F's, A's \mid F: \text{curve primitive}, A: \text{angle primitive}\}$$

$$P_\ell: S_\ell \rightarrow (XA)^*XA\{Answer_c\} \quad ; \quad c \leftarrow \ell$$

$$N \rightarrow (XA)^*X \quad ; \quad D(N) \leftarrow (D(X) \underset{A}{\oplus})^* D(X)$$

where $X \in \{N's, F's\}$

For each S-production rule, $\{Answer_c\}$ and $c \leftarrow \ell$ mean that if the parsing is successful, the shape pattern is recognized as being in the class labeled by c. The idea of $\{Answer_c\}$, an action symbol, is borrowed from the translational grammar [43,44]. Because of Theorems A and B, the attribute rules for each N-production rule can be obtained easily from the symbolic rule. Therefore, the attribute rules are omitted in the following examples.

Example 3.3: The shape grammar for airplane BAC 111,

$G_c = (V_c, T_c, P_c, S_c)$

$V_c = \{S_c, N_{ci} \mid 1 \leq i \leq 8\}$

$T_c = \{F_{cj}, A_{ck} \mid 1 \leq j \leq 15, 1 \leq k \leq 7\}$

P_c: (The corresponding segmentations are shown in Figures 3.7 and 3.8.)

(1) $S_c \rightarrow N_{c1}A_{c1}N_{c2}A_{c2}F_{c1}A_{c2}N_{c3}A_{c1}N_{c4}A_{c3}$

(2) $S_c \rightarrow N_{c2}A_{c2}F_{c1}A_{c2}N_{c3}A_{c1}N_{c4}A_{c3}N_{c1}A_{c1}$

(3) $S_c \rightarrow F_{c1}A_{c2}N_{c3}A_{c1}N_{c4}A_{c3}N_{c1}A_{c1}N_{c2}A_{c2}$

(4) $S_c \rightarrow N_{c3}A_{c1}N_{c4}A_{c3}N_{c1}A_{c1}N_{c2}A_{c2}F_{c1}A_{c2}$

(5) $S_c \rightarrow N_{c4}A_{c3}N_{c1}A_{c1}N_{c2}A_{c2}F_{c1}A_{c2}N_{c3}A_{c1}$

(6) $S_c \rightarrow N_{c6}A_{c2}F_{c1}A_{c2}N_{c7}A_{c4}N_{c8}A_{c3}N_{c5}A_{c4}$

(7) $S_c \rightarrow N_{c8}A_{c3}N_{c5}A_{c4}N_{c6}A_{c2}F_{c1}A_{c2}N_{c7}A_{c4}$

(8) $N_{c1} \rightarrow F_{c2}A_{c5}F_{c3}$

(9) $N_{c5} \rightarrow F_{c2}A_{c5}F_{c4}$

(10) $N_{c2} \rightarrow F_{c5}A_{c6}F_{c6}A_{c7}F_{c7}$

(11) $N_{c6} \rightarrow F_{c8}A_{c6}F_{c6}A_{c7}F_{c7}$

(12) $N_{c3} \rightarrow F_{c9}A_{c7}F_{c10}A_{c6}F_{c11}$

(13) $N_{c7} \rightarrow F_{c9}A_{c7}F_{c10}A_{c6}F_{c12}$

(14) $N_{c4} \rightarrow F_{c13} A_{c5} F_{c14}$

(15) $N_{c8} \rightarrow F_{c15} A_{c5} F_{c14}$

In Example 3.3, the S-production rules cover the most probable starting points of the boundary. Due to the rotation of the object, the starting point may be any of the convex points. Instead of looking through the whole boundary chain for a fixed starting point, we use the S-production rules to take care of the most probable starting points so that we only need to look over a short portion of the boundary chain for a sharp convex point, that would serve as the starting point. Because of the noise, sometimes the breaking points can not be found in extracting primitives. For instance, say the corner of angle primitive A_{c1} in Figure 3.7 is smeared so that F_{c5} and F_{c3} cannot be extracted. We can avoid this trouble by finding A_{c4}, see Figure 3.8, to extract F_{c8} and F_{c4}. With this idea, the noise problem at the breaking points can be taken care of by employing different segmentations. This example has essentially two sets of segmentation as seen in Figures 3.7 and 3.8. The non-simple curve segment, F_{c6}, and F_{c10} are used as primitives to reduce the number of primitives, and consequently reduce the difficulties of extracting primitives from the noisy shapes. The assignment of primitives is very flexible.

3.5 Recognition of Primitives

The first three problems mentioned in Section 3.2 are relevant to the description. They are discussed in Sections 3.2, 3.3, and 3.4. The other two problems relevant to the recognition are (1) primitive extraction, and (2) shape recognition by parsing.

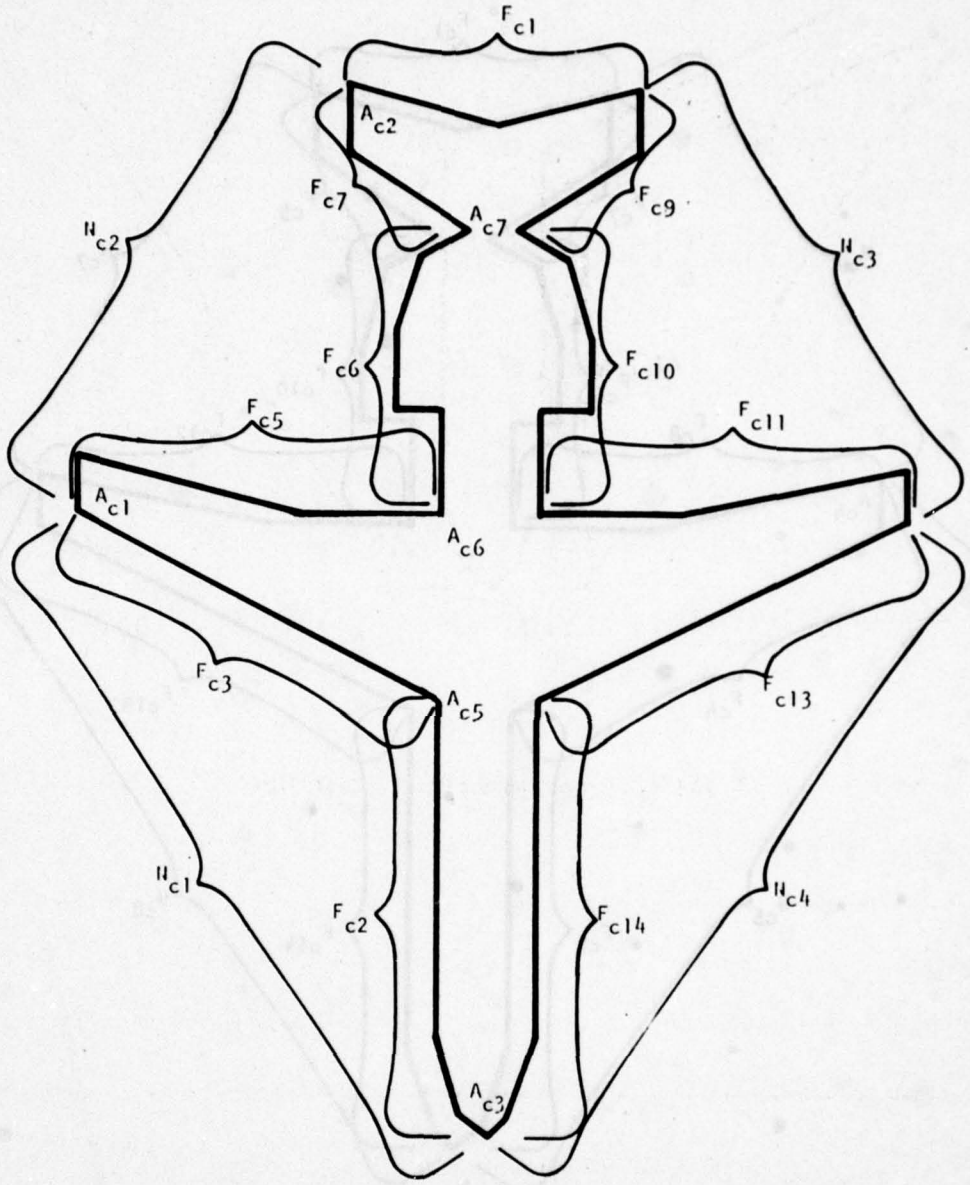


Figure 3.7 Segmentation 1 of BAC 111

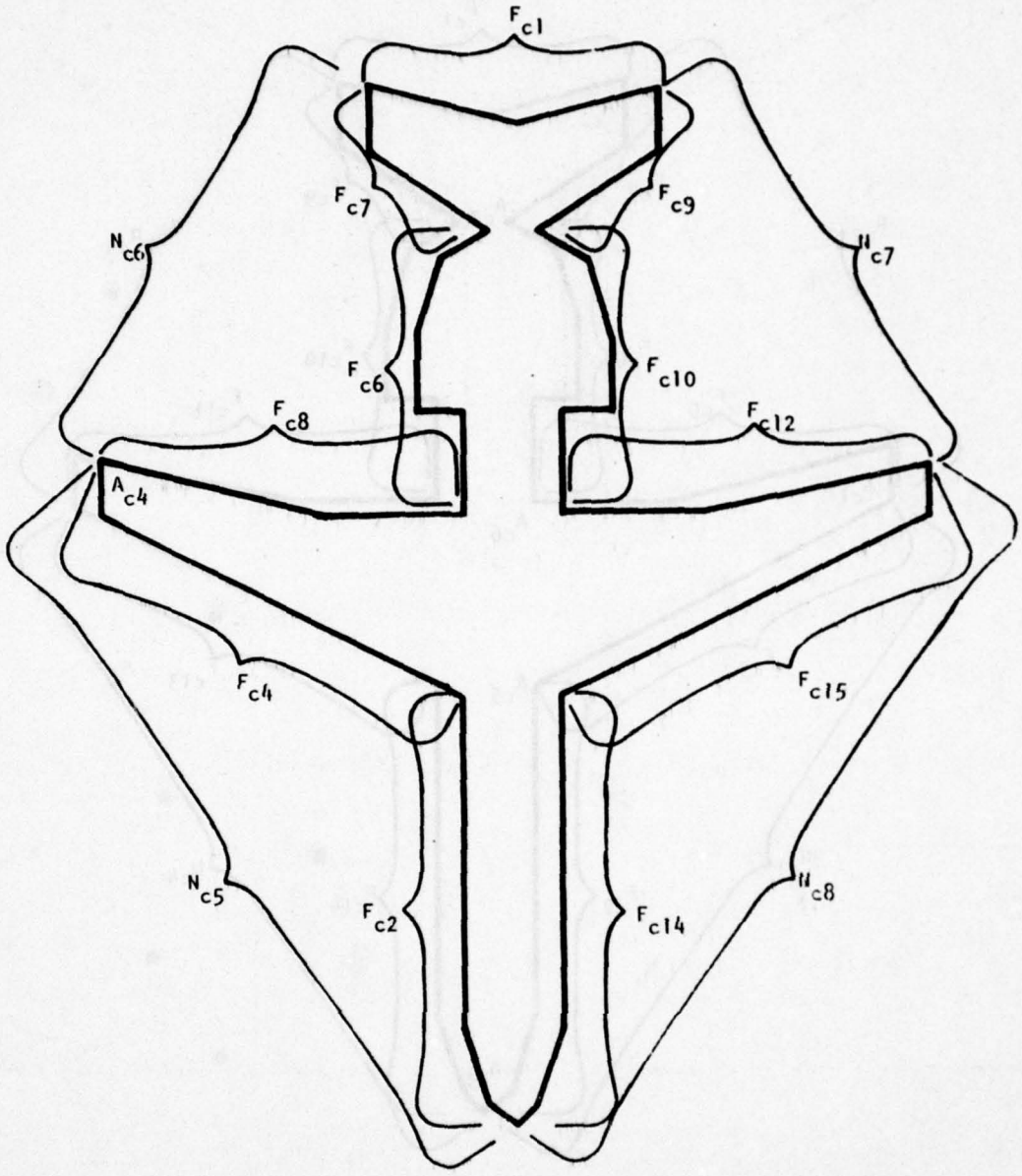


Figure 3.8 Segmentation 2 of BAC 111

As mentioned previously, a curve segment can be described by four attributes. But the translation, scaling, and rotation of the image may create different values for the attributes, and also introduce different noise in digitization. Fortunately, the attributes can be transformed into a multi-dimensional space, in which the transformed descriptors are theoretically invariant under the above operations, if it is noise-free.

Definition 3.8: Transformation $T:D(p) \rightarrow T(p)$, or

$$T:(\vec{C}, L, A, S) \rightarrow (\frac{C}{L}, \frac{L}{L_o}, A, \frac{S}{L}),$$

where $C = |\vec{C}|$, L_o is a scale normalization factor, which could be the total length of the shape pattern.

Theorem E: The C-descriptor transformed by T, T(p), is invariant with respect to translation, rotation, and scaling of the image.

Proof: Let us consider an analog image. The shape does not contain any digitization noise. It is easy to see that the translation does not affect the attributes. $\vec{C}$ changes with scaling and rotation. L, S, and L_o change proportionally with scaling only. Therefore, the divisions eliminate all the scaling factors, and the absolute value of $\vec{C}$ is invariant with respect to rotation.

Q.E.D.

Because of the invariant property, the recognition of the primitives, and hence the whole shape, is based on the transformed descriptors. If two curve segments are mirror images of each other, their transformed descriptors differ only in the sign of S/L. Consequently, if it is necessary, the storage for transformed descriptors of a bisymmetric shape, e.g. top view of airplanes, can be reduced in half by

storing them in pairs.

To recognize a primitive in the boundary, we simply rely on the similarity between the transformed descriptors. But digitization introduces different noise to the descriptors with respect to any operation of translation, rotation, and scaling. Normally, if the digitization resolution is fine enough, the digitization noise introduced by rotation is much more significant than that due to scaling and translation, because rotation influences the angle feature, A, significantly. Although we can apply some smoothing techniques to the boundary vector chain, the boundary cannot completely be recovered in all cases. We need to study the possible distribution of the feature values under various rotations to help construct a proper similarity measurement.

In the following paragraphs, we will concentrate on the noise effect of rotation on the shape information of a curve segment. The shape information is characterized by (C/L, A, S/L), while L/L_o represents the size of the curve segment in proportion to the total length.

For convenience, a normalization function is defined as follows:

Definition 3.9: A *normalization function*, N, of the curve primitive descriptor is:

$$N: (\tilde{C}, L, A, S) \rightarrow (X, Y, Z)$$

where

$$X = C/L \quad 0 \leq X \leq 1$$

$Y = A/2\pi$ (angle in terms of revolution)

$Z = S/AL$, $-0.5 < Z < 0.5$ for a simple curve segment

An experiment was designed and carried out through the following steps to study the distribution of the normalized variables under different rotations.

Experiment 3.1:

1. A picture with a clear boundary was scanned with respect to 8 various rotation angles.

2. Shapes on the digital pictures were traced out and passed through a smoothing procedure. The vector chains were obtained. (Boundary tracing and smoothing will be discussed in Chapter 4.)

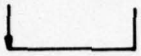
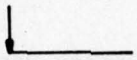
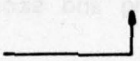
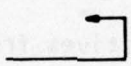
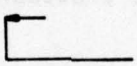
3. Manual extraction of primitives from the chain was performed via an interactive procedure.

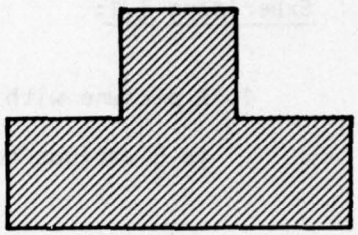
4. The descriptors of the manually extracted primitives were computed and transformed by N.

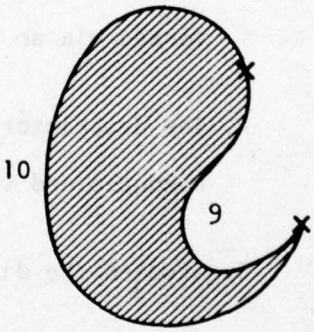
5. Studied the distributions of the normalized variables.

The three normalized variables, X, Y, Z, constitute a three-dimensional space, named 3-D for short. Table 3.1 illustrates the pictures used, the curve primitives, and the corresponding symbols in Figures 3.9-3.11, which show the two-dimensional displays of the distributions of descriptors. Each symbol in the figure indicates the position

Table 3.1 The Pictures and Primitives Selected for Noise Analysis

Cluster	Symbol	Curve Primitive
1	⊙	
2	△	
3	+	
4	×	
5	◇	
6	⏃	
7	⧖	
8	Z	
9	Y	
10	⊠	





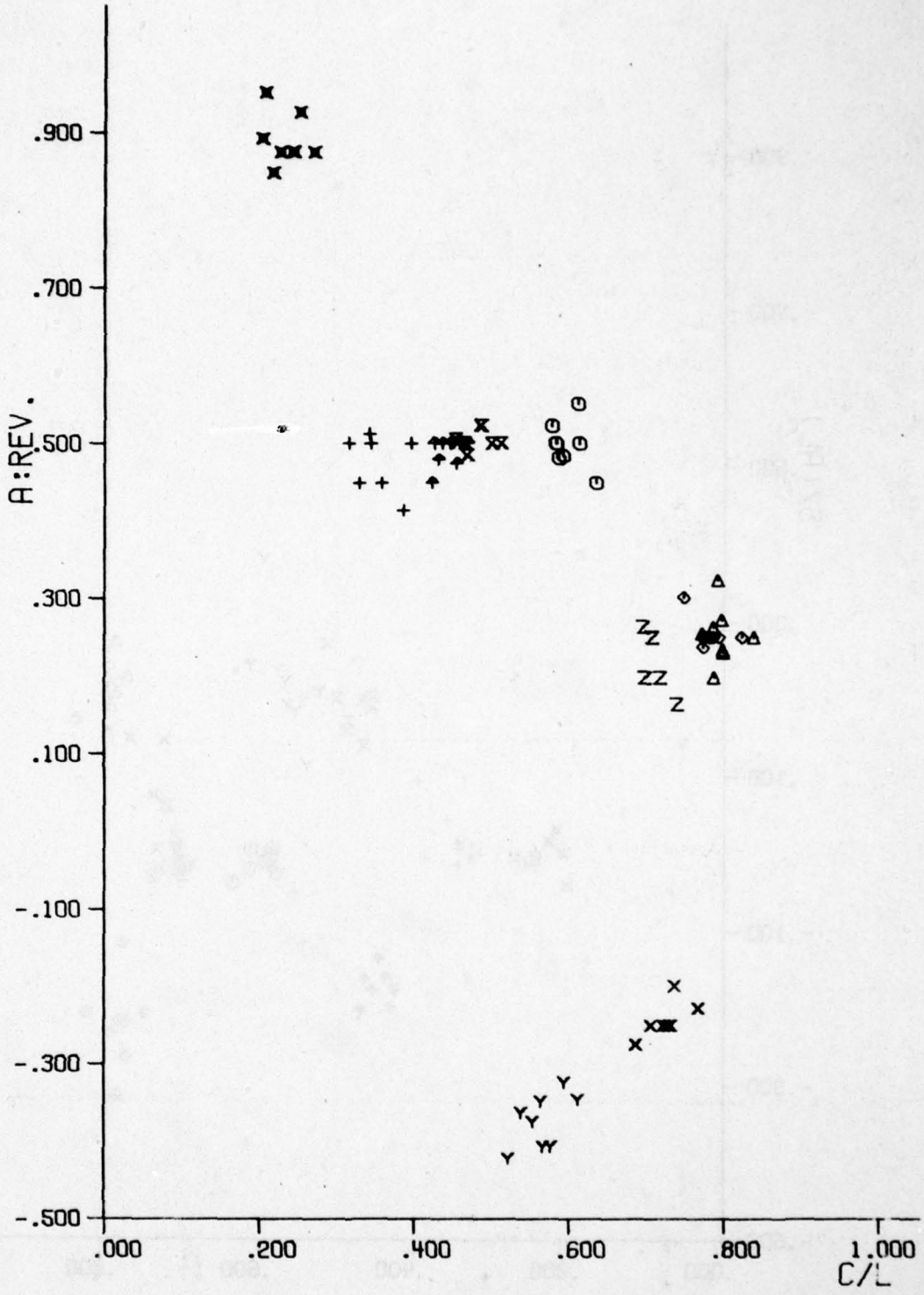


Figure 3.9 The First of the 3 Displays

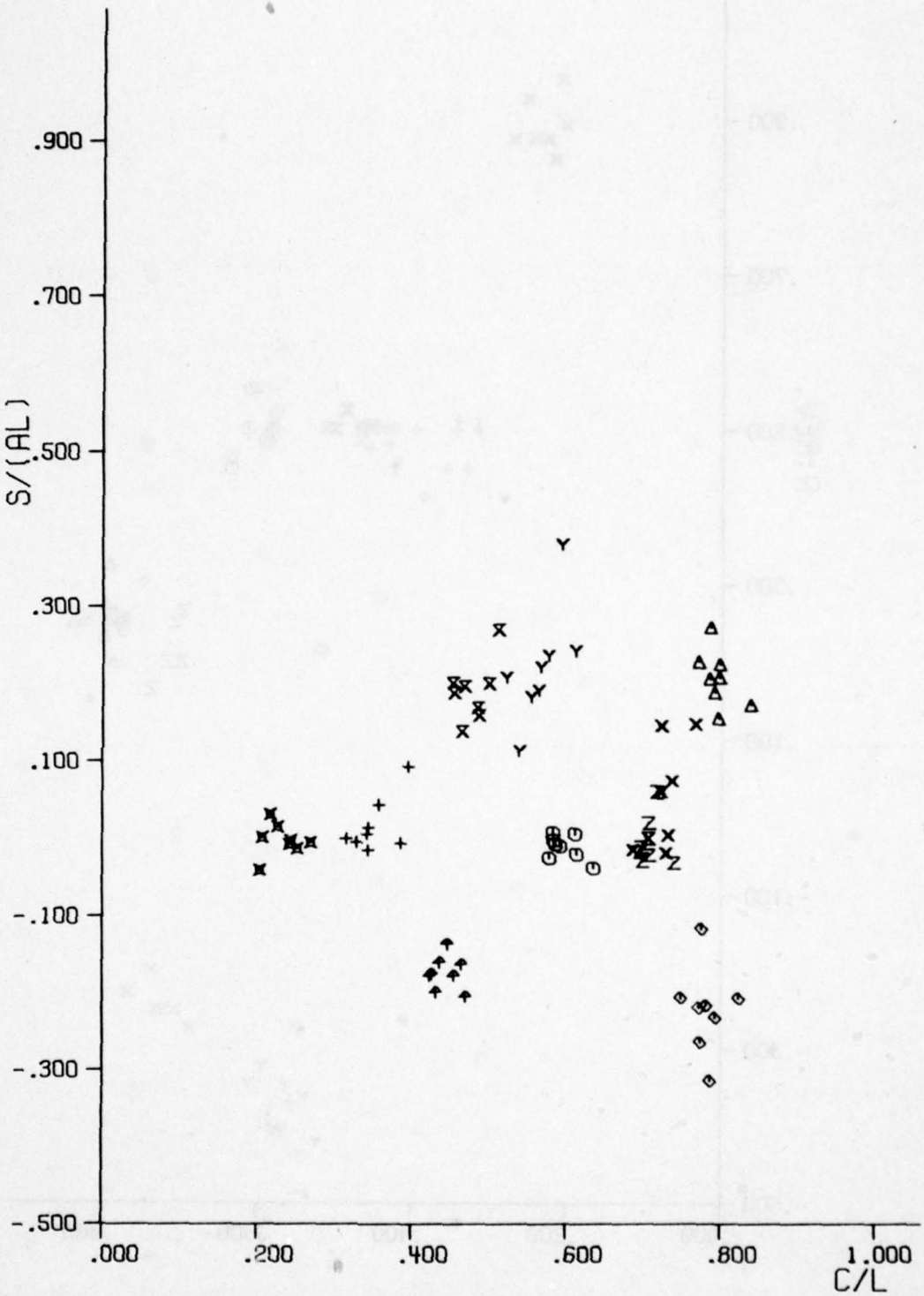


Figure 3.10 The Second of the 3 Displays

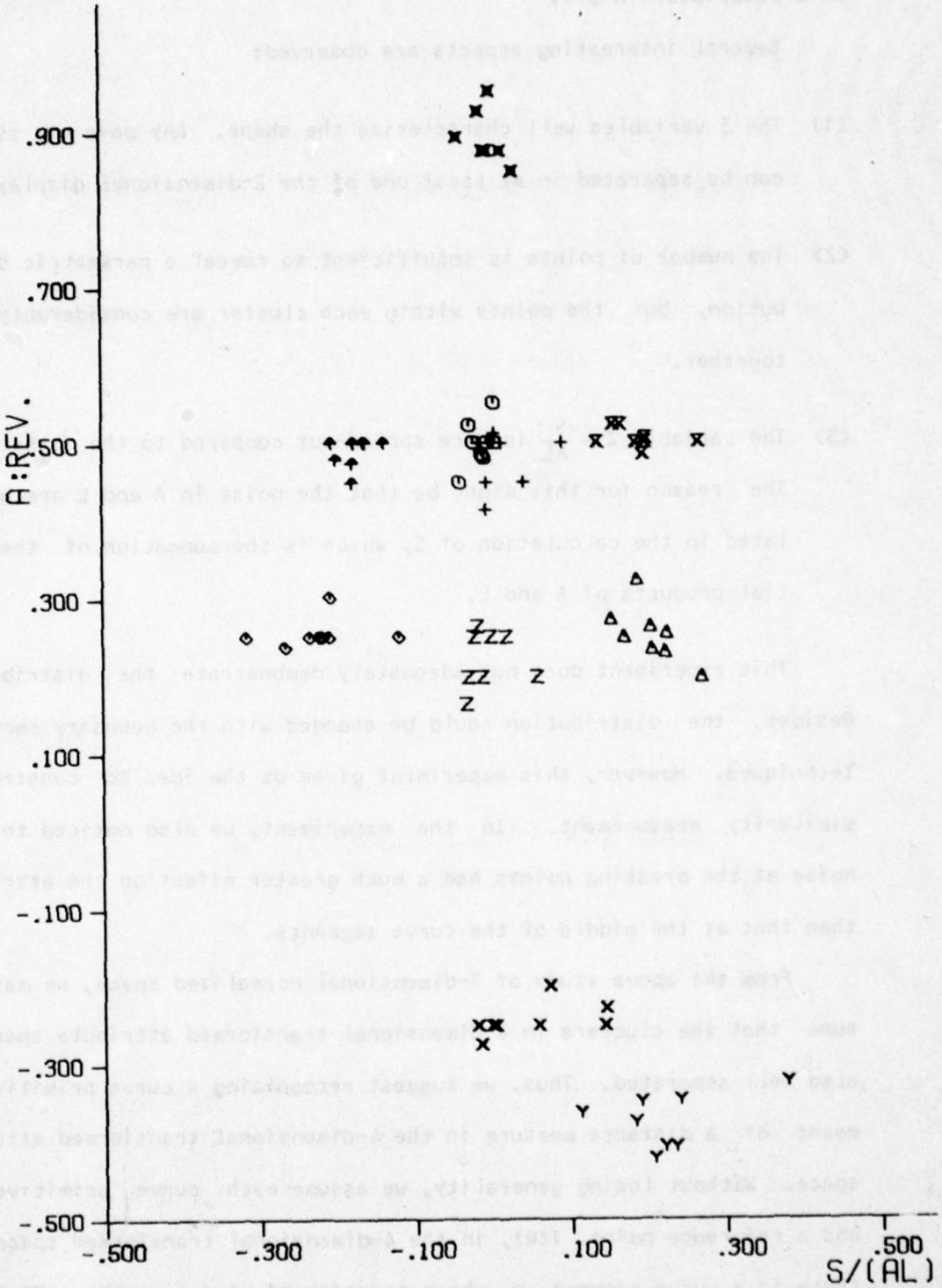


Figure 3.11 The Third of the 3 Displays

of a descriptor in 3-D.

Several interesting aspects are observed:

(1) The 3 variables well characterize the shape. Any pair of clusters can be separated in at least one of the 2-dimensional displays.

(2) The number of points is insufficient to reveal a parametric distribution, but the points within each cluster are considerably close together.

(3) The variable $Z = \frac{S}{AL}$ is more spread-out compared to the other two. The reason for this might be that the noise in A and L are accumulated in the calculation of S, which is the summation of the partial products of A and L.

This experiment does not adequately demonstrate the distribution. Besides, the distribution could be changed with the boundary smoothing techniques. However, this experiment gives us the idea to construct a similarity measurement. In the experiment, we also noticed that the noise at the breaking points had a much greater effect on the attributes than that at the middle of the curve segments.

From the above study of 3-dimensional normalized space, we may assume that the clusters in 4-dimensional transformed attribute space are also well separated. Thus, we suggest recognizing a curve primitive by means of a distance measure in the 4-dimensional transformed attribute space. Without losing generality, we assume each curve primitive, Q, has a reference point, T(Q), in the 4-dimensional transformed space. If there is a curve segment, q, whose transformed C-descriptor, T(q), is

relatively close to T(Q) in the 4-dimensional space, q is recognized as Q. In other words, there is a recognition function R_Q, if $R_Q(q) < t_Q$, t_Q is a threshold, $q \stackrel{r}{=} Q$. In general, R_Q can be a distance, similarity, or probability function dependent on Q. We could rewrite $R_Q(q)$ as R(Q,q).

The recognition of the angle primitive is similar, but simpler. For an angle primitive A, there is a function R_A. For any angle a, $R_A(a) < t_A$, $a \stackrel{r}{=} A$. $R_A(a)$ can be rewritten as R(A,a). Theoretically, the angle primitive has no length. Since sharp corners are often smoothed by noise, we allow a short length for angle primitives as called "corner tolerance". Of course, it is possible to employ the concept of partial recognition, or recognition with probability p, $0 \leq p \leq 1$, instead of "yes" and "no" for both curve and angle primitives.

Definition 3.10: $A \stackrel{r}{=} B$ implies that A and B are recognized as the same. A and B can be primitives, nonterminals, or vector chains.

Let us extend the above definition to a string of primitives, nonterminals, or vectors.

Definition 3.11: $X_1X_2 \ldots X_\alpha \stackrel{r}{=} v_i \ldots v_j$, the vector chain $v_i \ldots v_j$ is recognized as a string of primitives or nonterminals, $X_1X_2 \ldots X_\alpha$, iff there is a feasible segmentation on vector chain at vectors v_{k0}, v_{k1}, ..., $v_{k\alpha}$, where $k0 = i$, $k\alpha = j$, such that $v_{k(p-1)} \ldots v_{kp} \stackrel{r}{=} X_p$, for $2 \leq p \leq \alpha$.

3.6 Parsing Schemes

The primitives can be extracted on the basis of the recognition functions suggested in Section 3.5. If this can be done without knowing the context before parsing, then an input pattern can be represented as a string of primitives and an ordinary parser can be used to analyze the syntax of the primitive string according to a given grammar. This is the normal procedure of the syntactic approach. An ordinary parser constructs a derivation tree which derives the input primitive string using the production rules of the given grammar. (See Figure 3.12.) But, any context information can certainly help to reduce ambiguities in primitive extraction. Let us look at the following examples.

Example 3.4: Figure 3.6 is the top view of a typical airplane. The local descriptions, or the boundary subchains, around the right tail and the top of the right wing are very similar. It is difficult to determine whether the short vertical segment is primitive p_6 or part of primitive p_{12} unless we look ahead to see p_{10}, or the whole p_4.

Example 3.5: Two shape subpatterns in Figure 3.13 can be described as $N_1 \rightarrow p_1 b_1 p_2$ and $N_2 \rightarrow p_3 b_2 p_4$ respectively. The transformed descriptors of p_2 and p_4 are very similar. So are those of b_1 and b_2. Let us try to recognize a noisy vector subchain shown in Figure 3.14. $v_1 v_2$ is very close to p_1. If we extract $v_1 v_2$ as p_1, the recognition procedure may stop at v_2 after a_2 is recognized as b_1. No primitive can be extracted starting from v_3. If we extract $v_1 v_2 v_3$ as p_3, then we can proceed to extract a_3 as b_2 and $v_4 v_5$ as p_4. But the machine will not know that the extraction of $v_1 v_2$ as p_1 is incorrect until v_4 and v_5 are observed. The description of $v_1 v_2$ may be closer to p_1 than $v_1 v_2 v_3$ is to p_3.

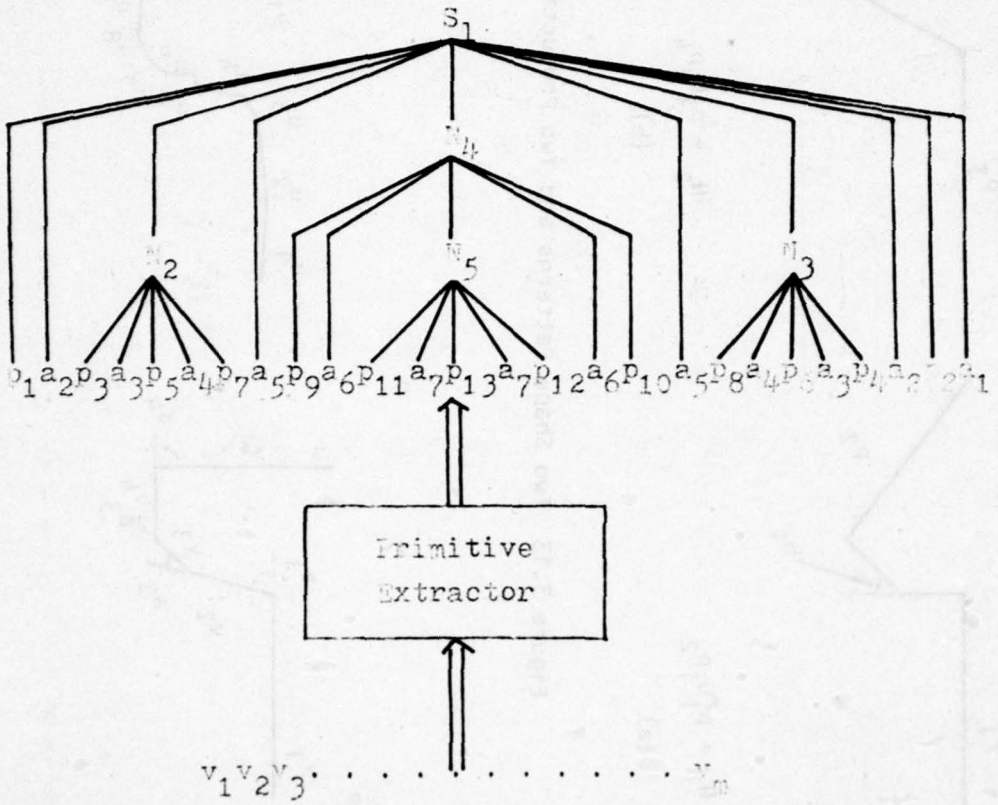
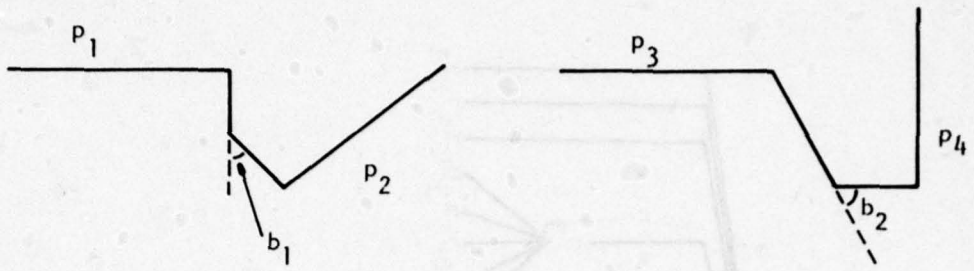


Figure 3.12 The Conventional Parsing



$N_1 \rightarrow p_1 b_1 p_2$

(a)

$N_2 \rightarrow p_3 b_2 p_4$

(b)

Figure 3.13 Two Shape Patterns and Two Productions

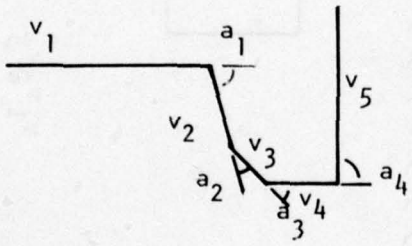


Figure 3.14
Recognition of a Noisy
Vector Subchain

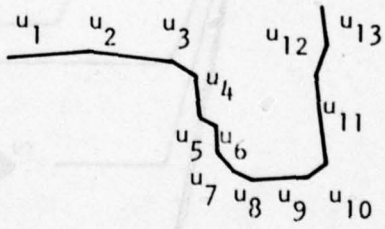


Figure 3.15
Recognition of a Noisier
Vector Subchain

Example 3.6: Suppose that the machine is asked to recognize a noisier pattern such as Figure 3.15 versus Figure 3.13 (a) and (b). The noisy pattern has 13 vectors. In order to allow some noise, the recognition function cannot be very selective. There may be several candidates for each primitive.

Primitive	Candidates
p_1	$u_1 \cdots u_4$, $u_1 \cdots u_6$
p_3	$u_1 \cdots u_5$, $u_1 \cdots u_6$, $u_1 \cdots u_7$, $u_1 \cdots u_8$
p_2 or p_4	$u_9 \cdots u_{13}$, $u_8 \cdots u_{13}$
b_1 or b_2	$u_6 \cdots u_8$, u_8u_9, $u_7 \cdots u_9$

The candidates for angle primitives must have close angle values and must be shorter than corner tolerance in length.

If we check all the candidates with the production rules, we find that there are only three possible combinations, or feasible segmentations. They are $u_1 \cdots u_6 \cdots u_8 \cdots u_{13}$ for N_1, and $u_1 \cdots u_7 \cdots u_9 \cdots u_{13}$, $u_1 \cdots u_8u_9 \cdots u_{13}$ for N_2. A feasible segmentation $u_1 \cdots u_7 \cdots u_9 \cdots u_{13}$ means that $u_1 \cdots u_7$, $u_7 \cdots u_9$, and $u_9 \cdots u_{13}$ are extracted consistently as three primitives of $p_3b_2p_4$. If we check the descriptors of the whole vector chain with those of N_1 and N_2, we find $u_1 \cdots u_6 \cdots u_8 \cdots u_{13}$ cannot be N_1 because of the total angle and the vector length of $\sum_{i=1}^{13} u_i$. Therefore, the unknown noisy subchain is recognized as N_2 through two feasible segmentations, $u_1 \cdots u_7 \cdots u_9 \cdots u_{13}$ and $u_1 \cdots u_8u_9 \cdots u_{13}$.

The above examples show that the ambiguity of primitive extraction can be resolved by using contextual information. The contextual information which is described by production rules is used in parsing. Therefore, if the primitives are extracted during the parsing, the extraction can be improved. Based on this idea, we have developed primitive-extraction-embedding (PEE) parsers. A PEE parser performs the job of recognizing primitives and nonterminals during the parsing. In Figure 3.16, the dashed blocks indicate recognition functions. The idea is, if more than one candidate can pass through the recognition function for a primitive or nonterminal, instead of making the selection immediately, the machine looks ahead to the context, checks the production rules, and then discards inappropriate candidates.

As we can see in Figure 3.15 and Example 3.6, the segmentation of a noisy boundary is fuzzy in the sense that it is difficult to find a definitely correct breaking point. Even a human can hardly break the noisy vector chain exactly. Therefore, the recognition of Figure 3.15 in Example 3.6 succeeded with two feasible segmentations on the noisy boundary corresponding to one segmentation on the true pattern (see Figure 3.13(b)). The feasible segmentations on the noisy boundary are called noisy boundary segmentations (NBS's) and the segmentation on the true boundary is called a reference segmentation. It is difficult to decide which of the two NBS's is more accurate than the other. Therefore, we accept both of them as correct segmentations. There might be more NBS's corresponding to one reference segmentation if the vector chain is very noisy and the shape is complex.

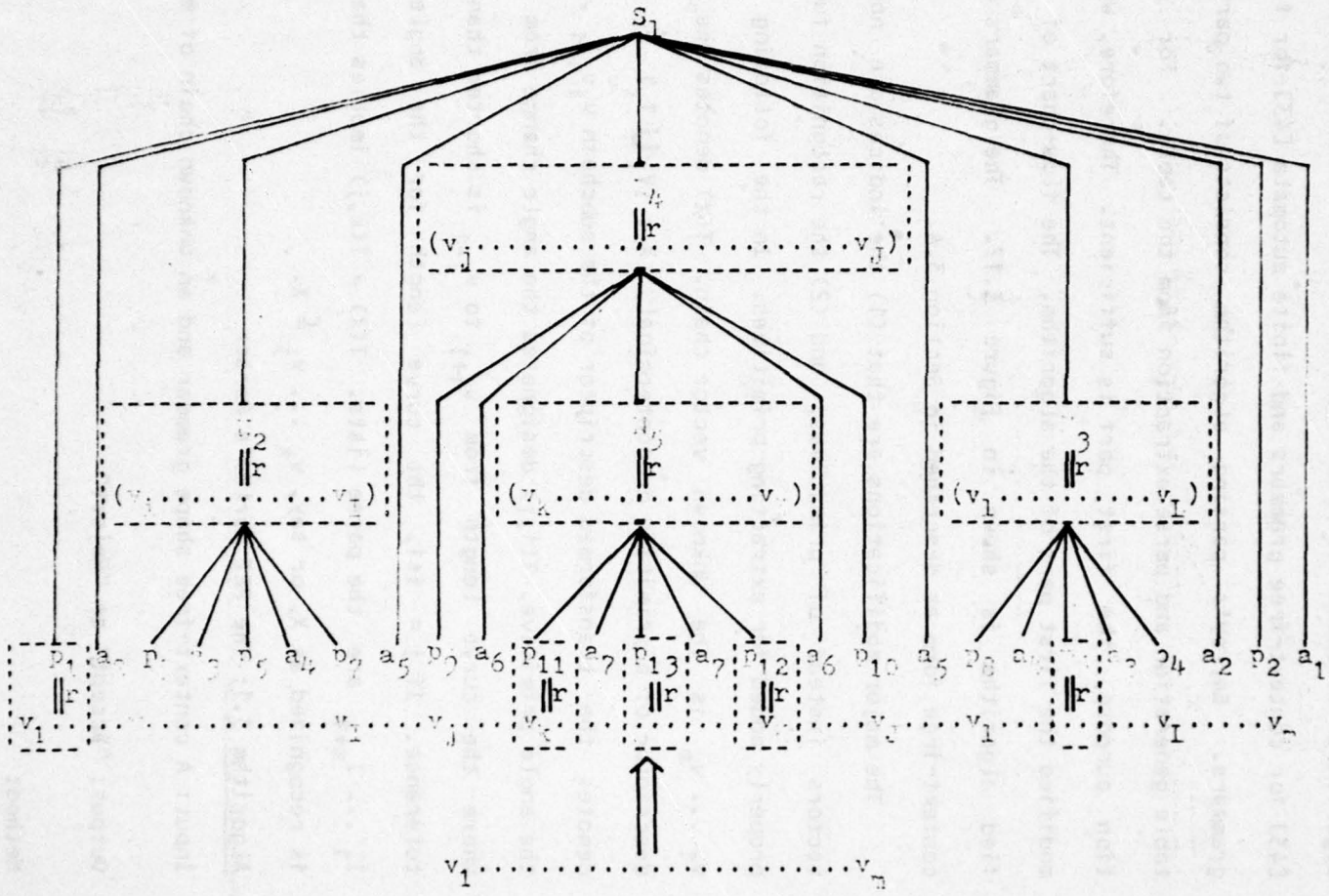


Figure 3.16 The Primitive-Extraction-Embedded (PEE) Parsing

To demonstrate the feasibility of PEE, we have modified two parsing algorithms to embed the primitive extraction. They are Earley's parser [43] for context-free grammars and finite automata [43] for finite-state grammars. Earley's parsing algorithm consists of two parts: parsing table generation and parse extraction from the table. For classification purpose, the first part is sufficient. Therefore, we have only modified the first part of the algorithm. The flow-chart of the modified algorithm is shown in Figure 3.17. The grammars used are in context-free form as described in Section 3.4.

The major modifications are that (1) the indices are now pointing vectors instead of primitives, and (2) the recognition functions are properly added for extracting primitives. In the following algorithm; $v_1 \ldots v_m$ is the unknown vector chain. T(X) denotes the transformed descriptor of a primitive or nonterminal, $X \in (V_\ell \cup T_\ell) - S_\ell$, T(i,j) denotes the transformed descriptor of the subchain $v_i v_{i+1} \ldots v_j$. For the angle primitive, T(i,j) designates the angle change from v_i to v_j where the curve length from v_{i+1} to v_{j-1} is shorter than the corner tolerance. If j = i+1, the curve length for the angle is zero. $I_1 \ldots I_{m+1}$ are the parse lists. $T(X) \simeq T(k,j)$ implies that $v_k \ldots v_j$ is recognized as X, or say, $v_k \ldots v_j \overset{\Gamma}{=} X$.

Algorithm 3.1: The PEE Earley's Parser

Input: A context-free shape grammar and an unknown chain of m vectors.

Output: "Accept" or "Reject"

Method:

(1) Add $[S \rightarrow \cdot\ \alpha, 1]$ to I_1 for all $S \rightarrow \alpha$ in P_ℓ

j = 1

(2) (a) If $[N \rightarrow \alpha \cdot B\beta, i]$ is in I_j and $B \rightarrow \gamma$ in P_ℓ

then add $[B \rightarrow \cdot \gamma, j]$ to I_j

(b) If $[N \rightarrow \alpha \cdot , i]$ is in I_j

then for all $[B \rightarrow \beta \cdot N \gamma, k]$ in I_i

add $[B \rightarrow \beta N \cdot \gamma, k]$ to I_j

(3) j = j+1

if $j > m+1$ goto (5)

For all $[N \rightarrow \alpha \cdot X\beta, i]$ in I_k, $1 \leq k \leq j$

$X \in \{F's, A's\}$

(a) If $\beta \neq \lambda$ and $T(X) \simeq T(k,j)$

then add $[N \rightarrow \alpha X \cdot \beta, i]$ to I_j

(b) If $\beta = \lambda$, $T(X) \simeq T(k,j)$ and $T(N) \simeq T(i,j)$

then add $[N \rightarrow \alpha X \cdot, i]$ to I_j

(4) Go to (2)

(5) If $[S \rightarrow \alpha \cdot, 1]$ in I_{m+1} for some α,

then "Accept", otherwise "Reject"

Remarks: When an ordinary Earley's parser is used to parse a string of symbols $a_1 \ldots a_n$, an item $[A \rightarrow \alpha \cdot \beta, i]$ is in parse list I_j, for $0 \leq i \leq j \leq n$, iff $\alpha \overset{*}{\underset{\ell m}{\Rightarrow}} a_{i+1} \ldots a_j$. But, when a PEE Earley's parser is used to parse a vector chain, $v_1 v_2 \ldots v_m$, an item $[A \rightarrow \alpha \cdot \beta, i]$ in parse list I_j implies that

(1) iff $\alpha \neq \lambda$, then $\alpha \overset{*}{\underset{\ell m}{\Rightarrow}} a_\ell \ldots a_k \overset{\Gamma}{=} v_i \ldots v_j$.

(2) if $\alpha = \lambda$, then $i = j$.

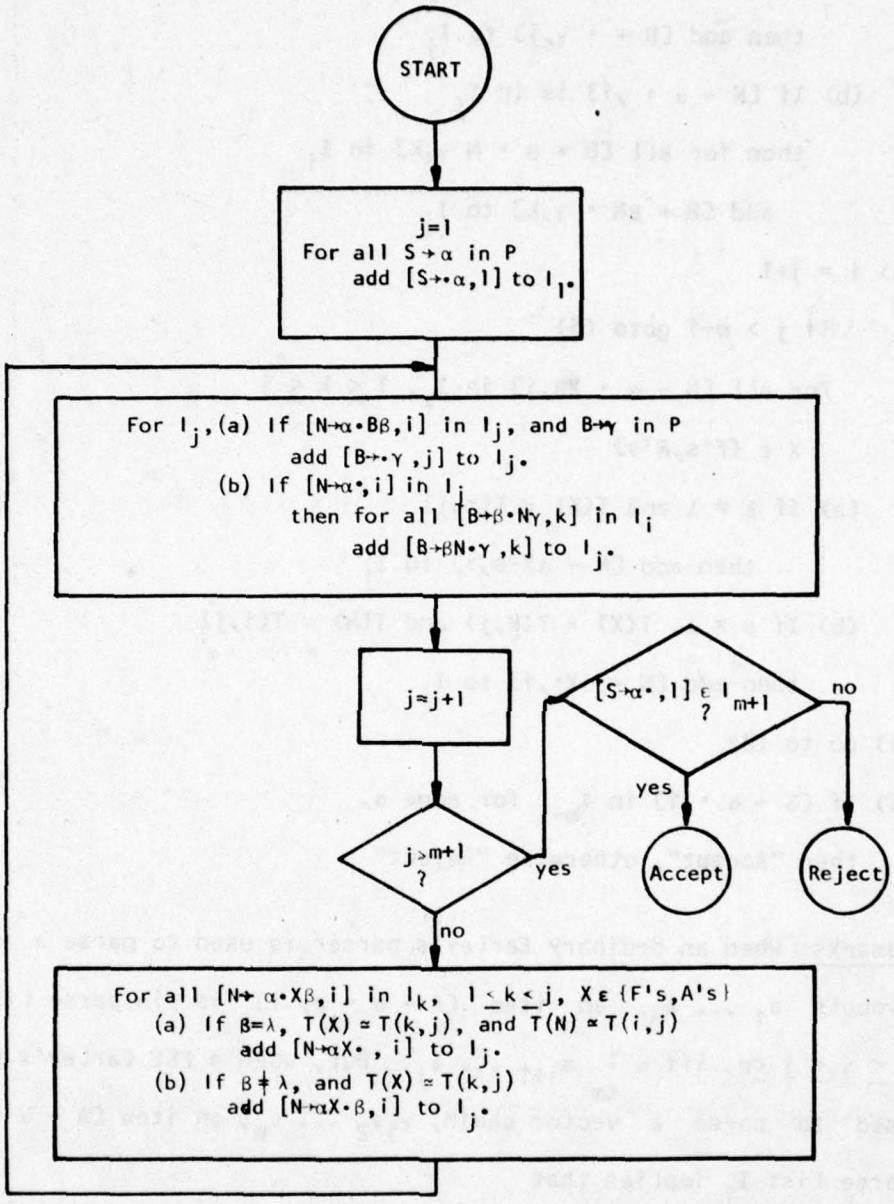


Figure 3.17 The Flow-Chart of PEE Earley's Algorithm

Therefore, if $[S \rightarrow \alpha \cdot, 1]$ is in I_{m+1} for some α, there exists at least one noisy boundary segmentation corresponding to a reference segmentation derived from $S \rightarrow \alpha$.

The following example illustrates how the contextual information is used to help extract primitives in Earley's algorithm.

Example 3.7: Let us look at the step (3.b). In extracting F_1 of $[N \rightarrow \alpha \cdot F_1, i_0]$ in I_{K_0} from the vector chain, $T(F_1) \simeq T(K_0,j)$ and $T(N) \simeq T(i,j)$ may be true for both $j \simeq j_1$ and j_2. In other words, subchains $v_{K_0} -- v_{j_1}$ and $v_{K_0} -- v_{j_2}$ are candidates for F_1, so $[N \rightarrow \alpha F_1 \cdot i_0]$ is in I_{j_1} and $[N \rightarrow \alpha F_1 \cdot, i_0]$ is in I_{j_2}. Suppose that $j_1 < j_2$. After the execution of step (2) for $j = j_2$, $[B \rightarrow \beta N \cdot \gamma, K_1]$ is in I_{j_1} and $[B \rightarrow \beta N \cdot \gamma, K_1]$ is in I_{j_2} where $K_1 \leq i_0 \leq K_0 \leq j_1 < j_2$. Suppose that $\gamma = A_1F_2A_2F_3$. The execution of step (3.a) to extract A_1 of $[B \rightarrow \beta N \cdot A_1F_2A_2F_3, K_1]$ in I_{j_1} and I_{j_2} from the vector chain may reveal that $T(A_1) \not\simeq T(j_1,j_3)$ for any $j_3 > j_1$, but $T(A_1) \simeq T(j_2,j_4)$ for some $j_4 > j_2$. Then only $[B \rightarrow \beta N A_1 \cdot F_2A_2F_3, K_1]$ is added to I_{j_4}. That is, the context information is used to select the subchain $v_{K_0} -- v_{j_2}$ for F_1 and discard $v_{K_0} -- v_{j_1}$. If $T(A_1) \simeq T(j_1,j_3)$ for some $j_3 > j_1$, and $T(A_1) \simeq T(j_2,j_4)$ for some $j_4 > j_2$, then $[B \rightarrow \beta N A_1 \cdot F_2A_2F_3, K_1]$ is in I_{j_3} and $[B \rightarrow \beta N A_1 \cdot F_2A_2F_3, K_1]$ is in I_{j_4}. The execution of step (3.a) to extract F_2 from the vector chain may reveal that $T(F_2) \not\simeq T(j_3,j_5)$ for any $j_5 \geq j_3 > j_2$, but $T(F_2) \simeq T(j_4,j_6)$ for some $j_6 \geq j_4 > j_1$. Then only $[B \rightarrow \beta N A_1F_2 \cdot A_2F_3, K_1]$ is added to I_{j_6}. That is, the lookahead on the information of subchain $v_{j_1} -- v_{j_5}$ selects the candidates $v_{K_0} -- v_{j_2}$ for F_1.

In fact, the extraction of A's and F's embedded in the parsing is different from the pre-extraction of the primitives done without knowing the contextual information. The advantage is that the extraction would be much more accurate in a global sense.

As illustrated in Example 3.1 and Figure 3.6, each nonterminal is semantically significant and is described by the attributes. In PEE Earley's algorithm, the recognition of both nonterminals and primitives is performed. But, the recognition of nonterminals is sometimes unnecessary, when the primitive recognition and syntax analysis are sufficient to discriminate the classes. In general, the languages generated by our context-free shape grammars can also be generated by finite-state grammars. But, the nonterminals in finite-state shape grammars are not very significant in semantics. In other words, we use a context-free grammar to describe a shape, because we like to take advantage of the context-free form, not because the corresponding language has to be generated by a context-free grammar.

Remark: The context-free shape grammar (CFSG) and finite-state shape grammar (FSSG) are referred to the shape grammars in context-free and finite-state forms, respectively.

For problems in which nonterminal recognition is not necessary only finite state grammars are used. Therefore, we also developed a PEE finite automaton. Since we can always find an angle primitive following a curve primitive, we consider that each time the input contains a curve primitive and an angle primitive. Figure 3.18 shows the storage of a finite-state grammar in a structural form. The automaton with its flow-chart shown in Figure 3.19, uses a STACK. Each element in the

STACK contains two fields, state and vtpt, which means the first vtpt-1 vectors of the unknown shape have been accepted through the state. FS is a set of final states.

Algorithm 3.2: The PEE Finite-State Automaton

Input: A finite-state shape grammar in tabular form (Figure 3.18) and an unknown chain of m vectors.

Output: "Accept" or "Reject"

Method:

(1) kp ← 1

STACK (kp) ← (S_{ℓ},1)

(2) If kp = 0 then terminate with "Reject"

otherwise s ← state (STACK (kp))

p ← vtpt (STACK (kp))

kp ← kp - 1

tp ← PTR (s)

(3) If tp = 0 GOTO (2)

(4) nxp ← nxpt (TABLE (tp))

F ← curve (TABLE (tp))

A ← angle (TABLE (tp))

nxs ← nxst (TABLE (tp))

(5) For all x,y, p ≤ x < y ≤ m+1

If (T(p,x) ≃ T(F) and T(x,y) ≃ T(A)) then

if nxs ∈ FS and y = m+1 then GOTO (7)

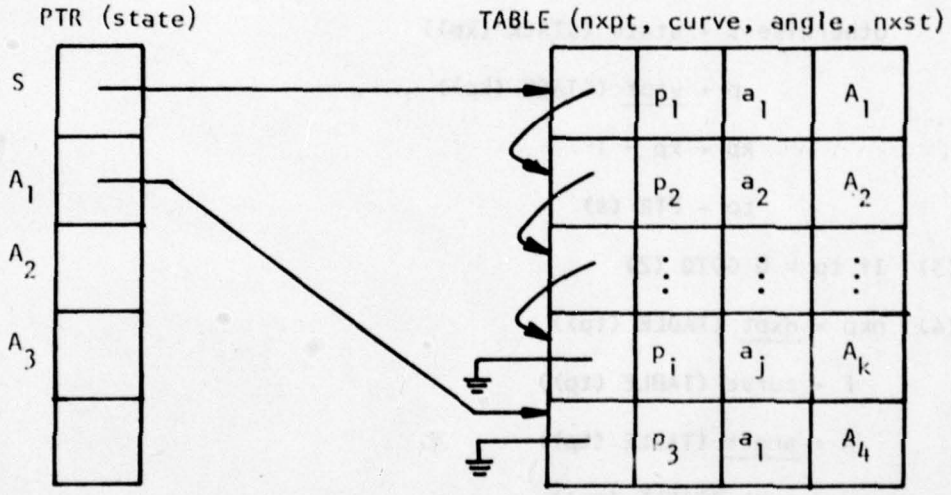
otherwise kp ← kp+1, STACK (kp) ← (nxs,y)

(6) If nxp = 0 then GOTO (2)

$S \rightarrow p_1\ a_1\ A_1 \mid p_2\ a_2\ A_2 \mid \ldots \mid p_i\ a_j\ A_k$

$A_1 \rightarrow p_3\ a_1\ A_4$

(a)



(b)

Figure 3.18 (a) The Production Rules of a Finite State Shape Grammar and (b) Their Structural Form in Storage

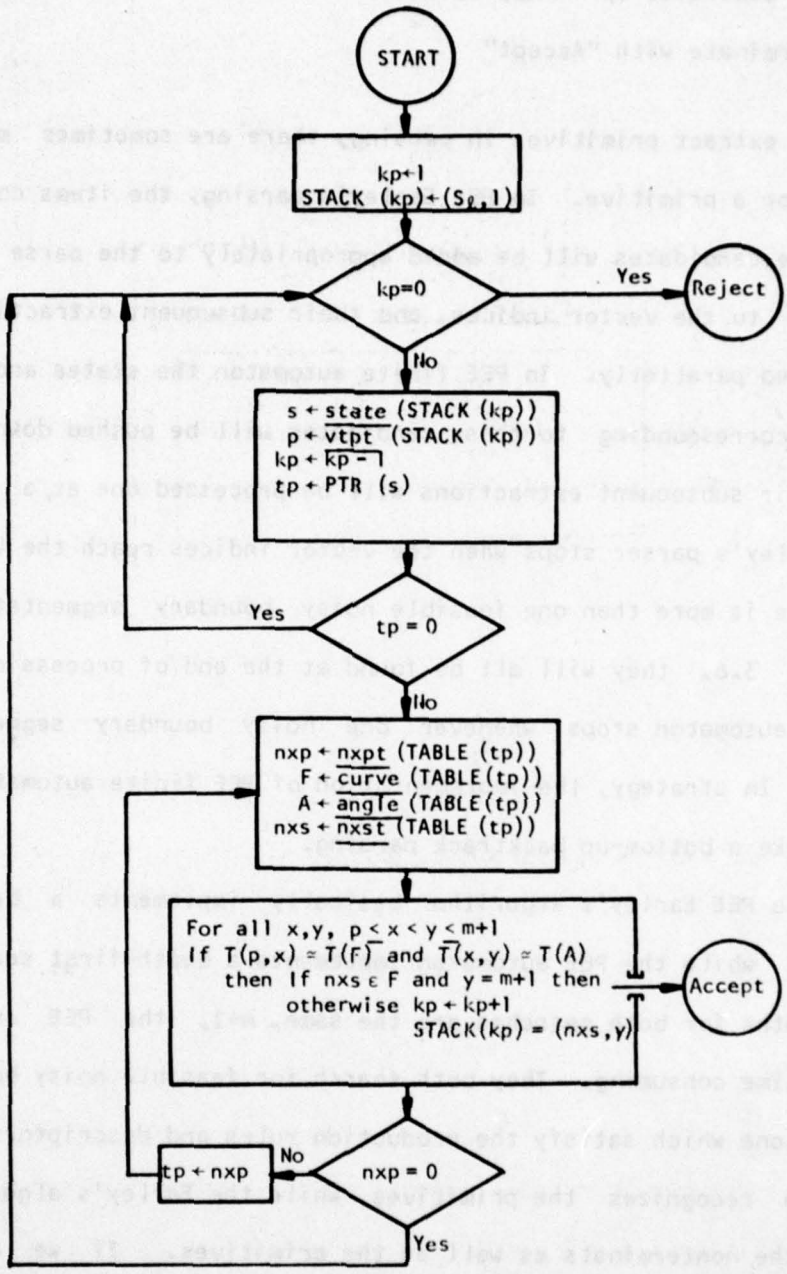


Figure 3.19 The Flow-Chart of PEE Finite Automaton

otherwise tp ← nxp, GOTO (4)

(7) Terminate with "Accept"

To extract primitives in parsing, there are sometimes many candidates for a primitive. In PEE Earley's parsing, the items corresponding to these candidates will be added appropriately to the parse lists according to the vector indices, and their subsequent extractions will be processed parallelly. In PEE finite automaton the states and vector indices corresponding to these candidates will be pushed down the stack, and their subsequent extractions will be processed one at a time. The PEE Earley's parser stops when the vector indices reach the last vector. If there is more than one feasible noisy boundary segmentation, e.g., Example 3.6, they will all be found at the end of processing. The PEE finite automaton stops whenever one noisy boundary segmentation is found. In strategy, the implementation of PEE finite automaton is somewhat like a bottom-up backtrack parsing.

The PEE Earley's algorithm basically implements a breadth-first search, while the PEE automaton implements a depth-first search. Since the depths for both searches are the same, m+1, the PEE automaton is less time consuming. They both search for feasible noisy boundary segmentations which satisfy the production rules and descriptors. The automaton recognizes the primitives, while the Earley's algorithm recognizes the nonterminals as well as the primitives. If we abandon the recognition of nonterminals in the Earley's algorithm, the two algorithms will end up with the same classification result. But, the automaton would be faster, because it stops at the first feasible set of primitives found. However, the recognition of nonterminals upgrades the

discriminating power of the Earley's algorithm.

3.7 Classification Tree

As mentioned before, the four attributes do not uniquely characterize a curve segment. The curve segments in Figure 3.20 (a) and (b) have the same C-descriptor. If the difference between the two figures is caused by noise, then the insensitivity of the descriptor completely ignores that noise. If the difference is significant in discriminating between the two classes, we can decompose Figure 3.20(b) into two shorter curve segments at point X_3'. $\overset{\frown}{X_1X_3'}$ and $\overset{\frown}{X_3'X_2}$ are assigned to be primitives p_1 and p_2 respectively. Figure 3.20(c) shows the decomposition. The recognition of p_1 and p_2 can definitely discriminate Figure 3.20(a) from (b).

If we are only interested in classification, we can use very simple shape grammars which check only some significantly different parts of the pattern for classification. There is no need to describe the complete shape in detail. Let us assign curve $\overset{\frown}{X_1X_2}$ to be primitive p_3. Figure 3.20(b) can be recognized as p_1 and p_2, or only p_3, but Figure 3.20(a) can only be recognized as p_3. This situation suggests a decision hierarchy. We can use p_3 to accept Figure 3.20(a) and (b) at the first decision stage, and use $p_1a_1p_2$ to accept Figure 3.20(b) at the second decision stage. Hence, a decision tree is built as in Figure 3.21.

The above description suggests the possibility of constructing a classification tree for multi-class problems with several simpler discriminating grammars at different stages. From another point of view, a classification tree is a top-down parsing, if only one level of

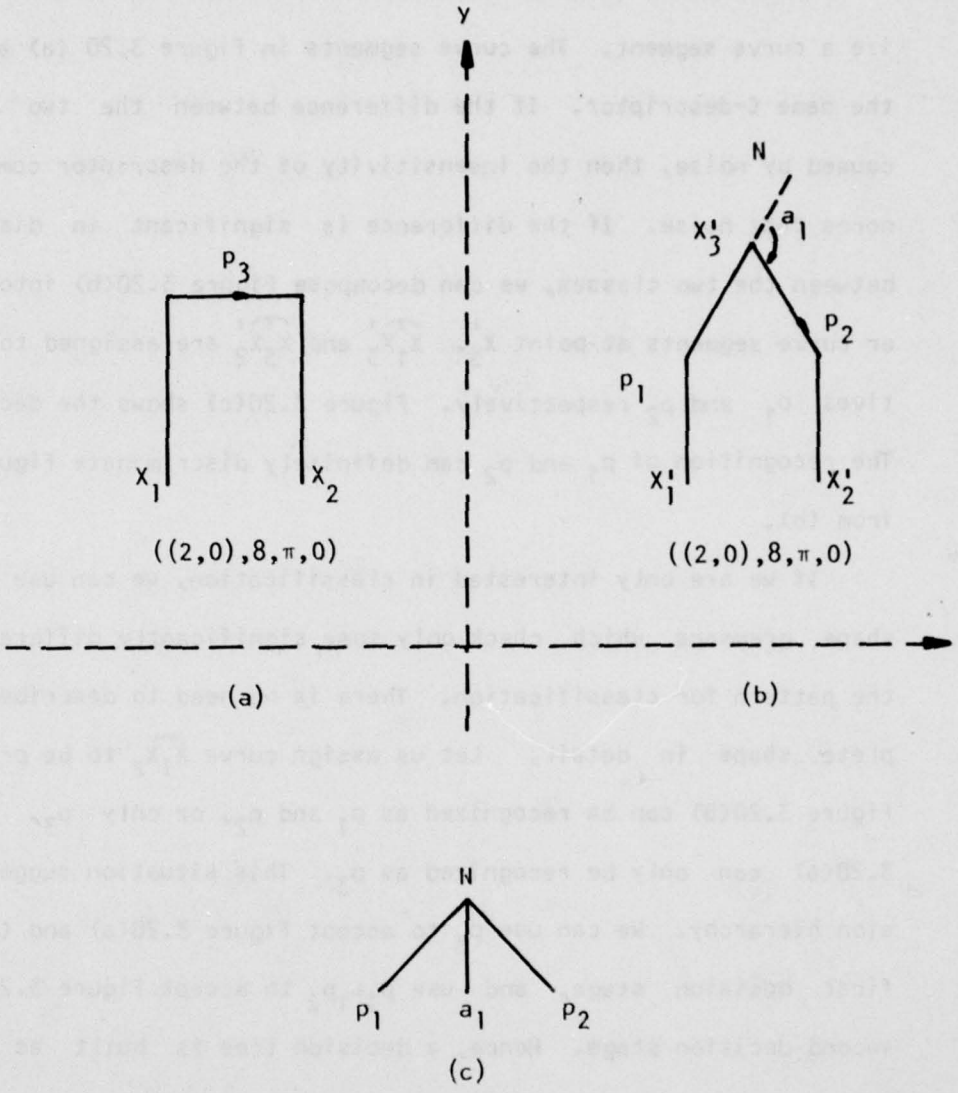


Figure 3.20 Two Different Curve Segments May Have The Same C-Descriptor

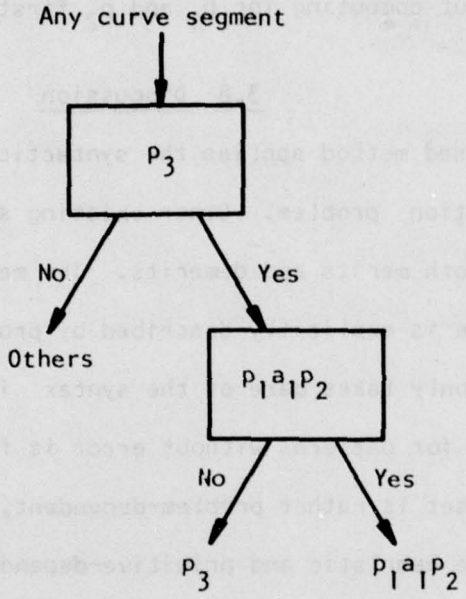


Figure 3.21 A Classification Tree

production rule is used at each discriminating grammar.

Though our shape grammars are formulated as attributed grammars with most of the attributes calculated upwards from the bottom, Theorem C states that it is not necessary to compute the attributes of primitives prior to those of nonterminals. We can always compute the attributes for a curve segment as long as we know the vector chain. For instance, we can compute the attributes for the nonterminal N, in Figure 3.20(b), without computing for p_1 and p_2 first.

3.8 Discussion

The proposed method applies the syntactic technique in the general shape recognition problem. Other existing syntactic shape recognition methods have both merits and demerits. The merits are (1) the global shape structure is explicitly described by production rules, and (2) the parsing stage only takes care of the syntax information, so that the parsing time for patterns without error is fast. The demerits are (1) the primitive set is rather problem-dependent, (2) the specifications of primitives are heuristic and primitive-dependent, (3) the extraction of primitives has to be done before parsing, and therefore, it is heuristic and inaccurate, and (4) nonterminals carry only syntax information.

The attributes we proposed carry the semantic information which gives a clear idea of the curve segment, if it is a simple curve segment. For a more complex curve segment, the first three attributes, $\hat{t}$, L, A, still can roughly characterize it. Thus, our method has the following merits: (1) Production rules describe the shape structure explicitly, (2) The primitive description method can handle a large number of primitives, (3) The primitive description method is systematic, general,

and problem-independent, (4) The primitive extraction utilizes both semantic and syntactic information, namely, attributes and production rules respectively. Hence, the primitive extraction is systematic and accurate in a global sense, (5) The whole system is well structured so that any particular part can be modified for special applications. For example, the attribute set, attribute rules, transformation, and recognition function are modifiable, (6) With context-free shape grammars, CFSG, a shape can be decomposed hierarchically so that each nonterminal represents a part of the shape which is meaningful in human perception. Then, the recognition of nonterminals increases its discriminating power, and (7) With our primitive description method, the primitives can be more sophisticated so that we don't have to use context-sensitive grammars, CSG.

If we use fixed-length curve primitives and delete the S attribute, then our method would become a general form for several existing syntactic applications. For example, in the chromosome recognition problem [5], the primitives a, b, c, d, e, are shown in Figure 3.22. A simpler example is the generation and recognition of squares [5] where the primitives are unit vectors. Since the primitives have equal lengths, they can be characterized by $\hat{C}$ and A. In such cases, the pattern representation would not be size-invariant and we would need a context-sensitive grammar to recognize the same shape with different sizes. In our approach, the S attribute allows various declinations. Size normalization with division by normalization factor L_o makes the descriptor size invariant, and hence, frees us from the need of using CSG in this respect.

But for patterns in which some parts may have different lengths in proportion, division by total length does not solve the problem. For example, a submedian chromosome has two arm pairs, each arm pair having two arms of equal length. Figure 3.23(a) shows the segmentation with conventional primitives and (b) shows one possible segmentation with our proposed shape primitives where α is the angle primitive and s, f, ℓ, c are the curve primitives. One possible sentential form is

$$f\alpha\ell\alpha c\alpha\ell\alpha f\alpha s\alpha c\alpha s\alpha$$

with

$$D(\alpha) = 0$$

$$D(\ell) = (\vec{t}_\ell, L_\ell, A_\ell, S_\ell)$$

$$A_\ell = \pi, \quad S_\ell = 0$$

$$D(f) = (\vec{t}_f, L_f, A_f, 0)$$

$$-\frac{\pi}{4} < A_f \leq 0$$

$$D(s) = (\vec{t}_s, L_s, \pi, 0)$$

$$D(c) = (\vec{t}_c, L_c, -\pi, 0)$$

Since $C_o = |\vec{t}_c|$ is one of values which does not vary with the length of the arms, we may use C_o as the size normalization factor and modify the

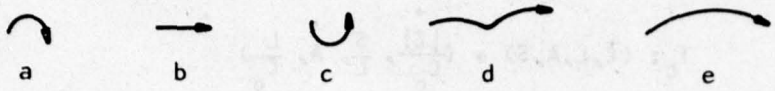
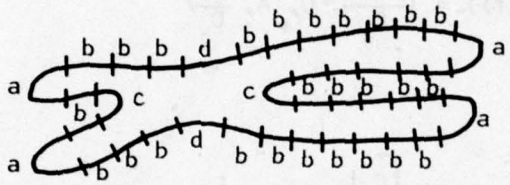
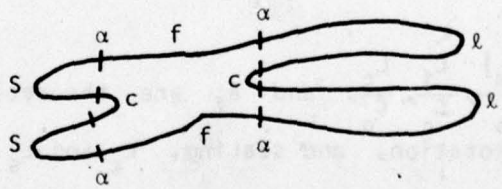


Figure 3.22 Conventional Fixed-Length Primitives for Chromosome



(a)



(b)

Figure 3.23 The Segmentation of a Submedian Chromosome with (a) Fixed-Length Primitives and (b) The Proposed Shape Primitives

T transformation to T_c.

$$T_c: (\vec{C}, L, A, S) \rightarrow \left(\frac{|\vec{C}|}{C_o}, \frac{S}{L}, A, \frac{L}{C_o}\right)$$

So that

$$T_c(D(\ell)) = \left(\frac{|\vec{C}_\ell|}{C_o}, 0, \pi, \frac{L_\ell}{C_o}\right)$$

$$T_c(D(f)) = \left(\frac{|\vec{C}_f|}{C_o}, 0, A, \frac{L_f}{C_o}\right)$$

$$T_c(D(s)) = \left(\frac{|\vec{C}_s|}{C_o}, 0, \pi, \frac{L_s}{C_o}\right)$$

$$T_c(D(c)) = \left(1, \quad 0, -\pi, \frac{L_c}{C_o}\right)$$

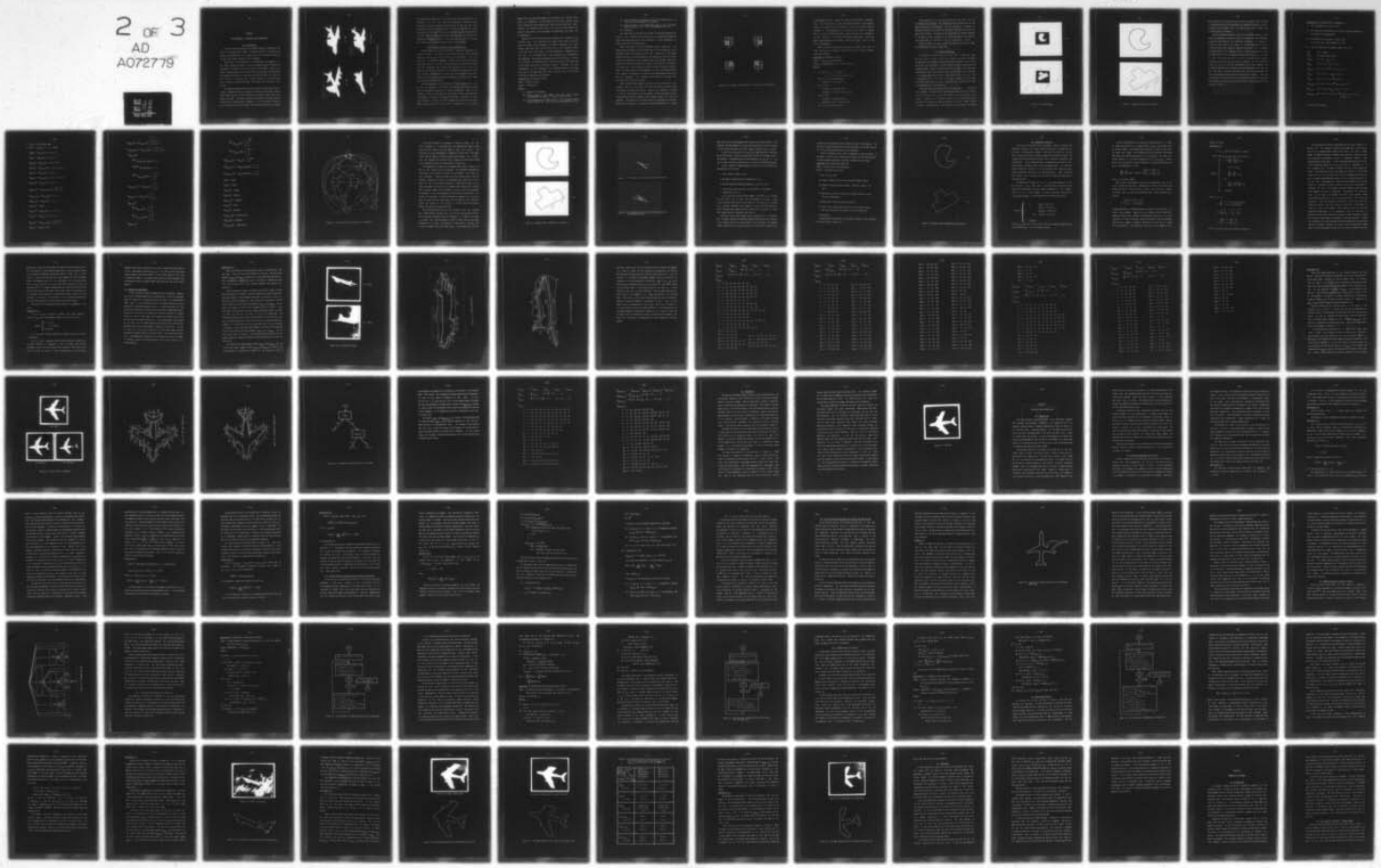
$\frac{|\vec{C}_\ell|}{C_o}$, $\frac{|\vec{C}_f|}{C_o}$, $\frac{|\vec{C}_s|}{C_o}$, $\frac{L_f}{C_o}$, $\frac{L_c}{C_o}$, and A_f are theoretically invariant under translation, rotation, and scaling. L_ℓ and L_s are subjected to change with the length of the arms. If $L_\ell \simeq L_s$, it is median. If $L_\ell \neq L_s$, it is submedian. ($\simeq$ means "nearly equal").

The above discussion implies that we may not need CSG in solving many shape recognition problems, if a proper transformation T can be found. Obtaining the proper transformation seems to be problem depen-

dent. The major purpose of this transformation is to eliminate the size problem. Therefore, the transformation is not difficult to obtain, if a size normalization factor is known. However, the shape grammar we proposed has the potential of solving a general class of shape recognition problems without requiring CSG.

AD-A072 779 PURDUE UNIV LAFAYETTE IN SCHOOL OF ELECTRICAL ENGINEERING F/G 9/2
SYNTACTIC SHAPE RECOGNITION USING ATTRIBUTED GRAMMARS.(U)
AUG 78 K C YOU, K S FU AFOSR-74-2661
UNCLASSIFIED TR-EE78-38 AFOSR-TR-78-1425 NL

2 OF 3
AD
A072779



CHAPTER 4

AN EXPERIMENT OF AIRPLANE SHAPE RECOGNITION

4.1 Introduction

Like other existing methods, the proposed method is designed on the basis of two-dimensional images. Although the objects to identify are usually three-dimensional, our recognition has to be based upon what we can get from economic automatic visual equipment.

The purpose of this experiment is to demonstrate the capability of our shape recognition method. Since the proposed syntactic method can distinguish the shapes by structure as well as by boundary details, we choose airplane models for our experiments because some airplane shapes are completely different in structure, e.g. B52 and F102, and some are very similar in structure but slightly different in such details as tails or wings, e.g. MIG-15 and F86. These models can be found in Figure 4.1.

The computer preprocessing consists of the following steps: digitization, threshold selection, boundary following, and smoothing. Before these steps, however, we have to prepare the airplane models and take analog pictures. The model preparation, analog picture taking and digitization will be discussed in Appendix A. In the following sections, we will discuss the series of steps from threshold selection to recogni-

(a) B52

(c) F86

(b) F102

(d) MIG-15

Figure 4.1 Selected Airplane Models

tion experiments sequentially as we carried out these demonstrative experiments. The first few steps were implemented at Purdue's Pattern Processing and Advanced Automation Laboratory. We used the PDP 11/45 minicomputer in the laboratory mainly for collecting data and for simple preprocessing. The last few steps including parsing were implemented on the CDC 6500 at Purdue University's Computing Center to minimize programming and debugging efforts. In the last section, we will discuss the limitations of the discriminative power of this method.

4.2 Threshold Selection and Boundary Following

As mentioned before, we are interested in the shape information of digital pictures. Our digital pictures have 128 gray levels and good contrast. The gray level value increases with the darkness. The picture size is 180x180 or 200x200. A simple way of obtaining the shape from such a picture is to find a threshold which can separate the object from the background. But selecting the threshold is not trivial. Because of the different characteristics of the analog picture, the threshold finding techniques may be different. The inevitable noise introduced by illumination, background, photograph, and digitization makes the situation more complicated.

An important purpose of thresholding is scene segmentation, i.e., to find the boundary of an object and single out the object. Weszka [77] surveyed many techniques for threshold selection. Most of the threshold selection techniques are based on the information provided by a histogram. For blood cell images, the averaging technique is often used [63,78] to smooth out the histogram until there are only three peaks, and then select the gray level corresponding to the valley

between the first and second peaks as the threshold for finding white blood cell boundaries. For red blood cells, the same technique can be used to smooth out the histogram until there are only two peaks. Unfortunately, this method does not work as well on our airplane images, because the small peak, in the histogram, corresponding to the object may be smoothed out.

Because of the flat black paint on the models, the airplanes look uniformly dark. This characteristic should create a peak in the high gray level region of the histogram. This peak is sometimes very small because some angle view, e.g. the front view, of the airplanes occupies a very small area of the image. The light background is supposedly uniform too. So, the highest peak in the histogram often corresponds to the background. Although the background may not be absolutely uniform, the uniformity of the background is better than that of the object due to the slight reflection of the object surface. Therefore, the peak corresponding to the background in the histogram usually has a steep down slop in the higher gray level side. Thus, we have the following algorithm for obtaining the threshold.

Algorithm 4.1: Threshold Selection

Input: A digital picture.

Output: A threshold t.

Method:

(1) Compute the histogram.

(2) Find the peak of the highest gray level (which usually corresponds to the object). Let k_1 = the gray level.

(3) Find the maximum, the highest peak of the histogram besides the one found in (2). Let k_2 = the corresponding gray level.

(4) Find the lowest valley between the above two peaks and let k_3 = gray level corresponding to the valley.

(5) Find the bottom of the sharpest down slope of the histogram between k_1 and k_3. Let t = the corresponding gray level.

(6) Terminate.

This algorithm is efficient and accurate in selecting threshold for pictures with following two characteristics, (1) good contrast between object and background, and (2) the uniformity of light background is better than that of the dark object.

After a threshold is found, the boundary can be traced out. The boundary may be defined in two different ways: (1) the connection of the outermost pixels of the object, and (2) the connection of the edges between the object and the background. The boundary by the first definition can be coded by the octal Freeman chain codes [42,59]. Under this definition a narrow bar shape with a width of one pixel will be coded as zero pixel wide. Since we hope to extract the correct semantic information, including width from the boundary, we select the second definition. Namely, our boundary is a connection of edges between the object and the background. This boundary can be coded by unit vectors with horizontal and vertical directions.

Sidhu and Bonte [60] proposed to encode a binary picture with containment codes via a 2x2 window. And then, they used the codes to lead the boundary following successfully. Actually, the boundary following is led by the contents in the 2x2 window. We have developed an algorithm which utilizes the contents in the window directly to find the boundary. No conversion to a binary picture nor encoding with containment codes is necessary. Our window is described dynamically in terms

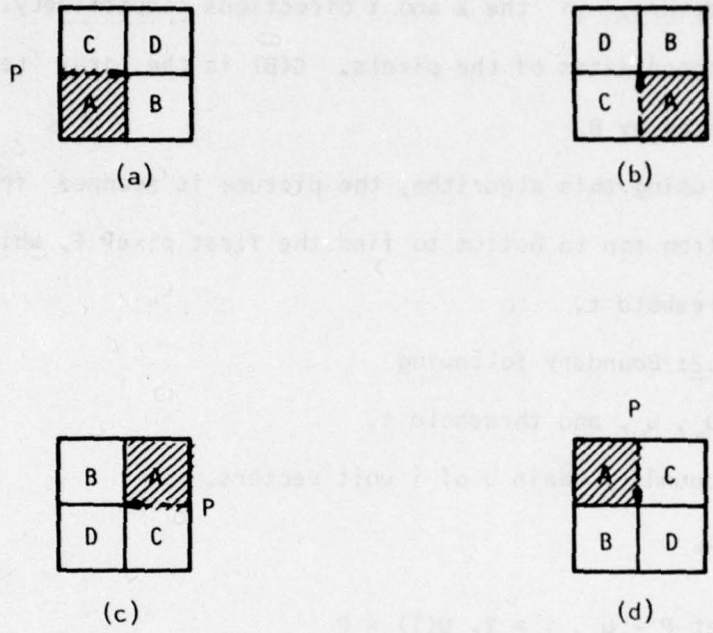


Figure 4.2 4 Possible Configurations of Boundary Following Window

of the boundary vectors. Figure 4.2 shows the four possible configurations. The pixels A, B, C, and D are defined relative to the boundary vector P. The object is to the right of P, so that A is darker than the threshold t. The background is to the left of P, so that C is lighter than t. In the following algorithm, u_x and u_y are the unit movements, or unit vectors, in the X and Y directions respectively. A, B, C, D denote the coordinates of the pixels. G(B) is the gray level of the pixel indicated by B.

Before using this algorithm, the picture is scanned from left to right and from top to bottom to find the first pixel F, which is darker than the threshold t.

Algorithm 4.2: Boundary Following

Input: F, u_x, u_y, and threshold t.

Output: A boundary chain U of i unit vectors.

Method:

(1) Set $P = u_x$, $i = 1$, $U(1) = P$

$A = F$, $C = F-u_y$, $S = C$, go to (3)

(2) If ($S = C$) then terminate

otherwise $i = i+1$, $U(i) = P$

(3) $D = C+P$

If ($G(D) < t$) then go to (4)

otherwise $P = C-A$, $A = D$, go to (2)

(4) $B = A+P$

If ($G(B) < t$) then go to (5)

otherwise $A = B$, $C = D$, go to (2)

(5) $C = B$, $P = B-D$ go to (2)

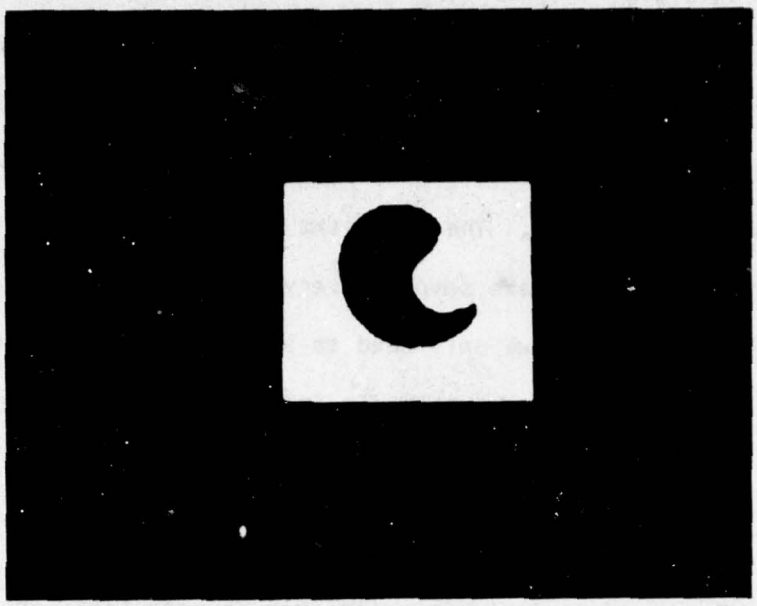
This algorithm is very efficient because only the pixels next to the boundary are processed. The computation time is proportional to the number of vectors in the boundary. The time required for a boundary of 1000 unit vectors is only about 0.6 second on a PDP 11/45 with auxiliary memory for pictures. The algorithm was implemented in FORTRAN language. If we want to take samples every k_x pixels horizontally and every k_y pixels vertically, we only need to set $u_x = (k_x,0)$ and $u_y = (0,k_y)$.

Figure 4.3 shows two simple images, and Figure 4.4 shows the boundaries obtained by using our algorithm.

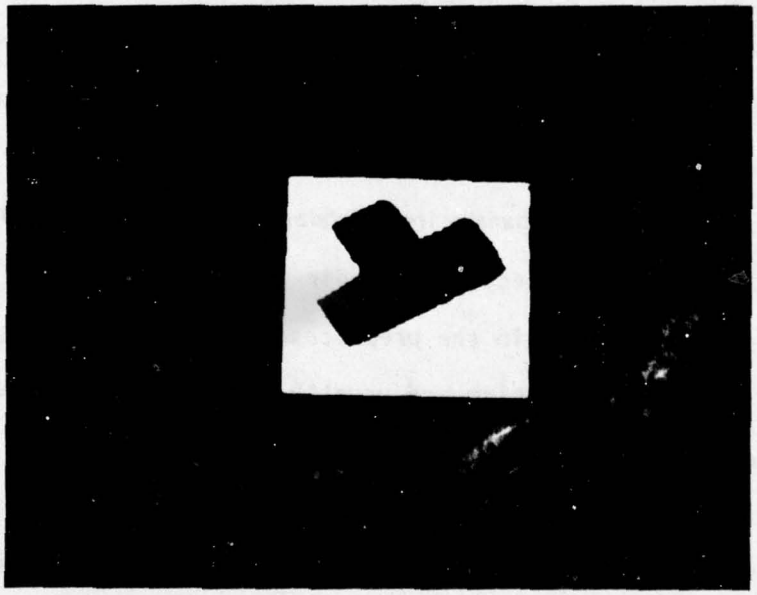
4.3 Boundary Smoothing

Our boundary following algorithm is very fast in computing time. But, the results in Figure 4.4 show that further processing is necessary to smooth out the zig-zag's, because we need more accurate angle information for later processing. In other words, we need to approximate the boundary better. Pavlidis and Horowitz [51], and Ramer [52] studied the algorithms for approximating boundaries. These methods are not suitable in our processing because of their computational cost. We do not like to spend much time in the preprocessing stage. Besides, we hope to keep the sharp corners which are usually meaningful for recognition but sometimes smoothed out in approximation.

The output from our boundary following algorithm is a string of unit vectors. The input to our parsing algorithms discussed in Section 3.6 is a string of vectors which approximates the true shape more accurately. Therefore, we developed smoothing algorithms which use the string of unit vectors as input and produce a string of longer vectors as output. This mechanism can be described as a translation. A machine

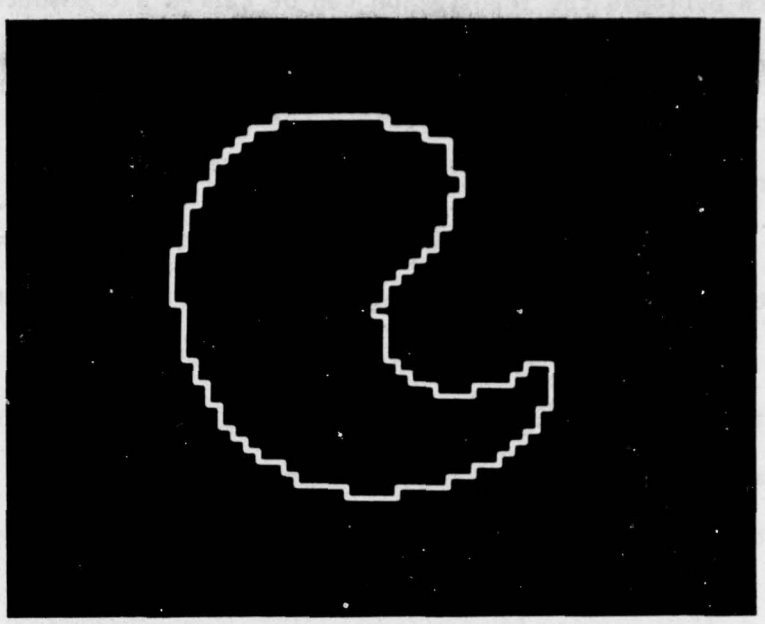


(a)

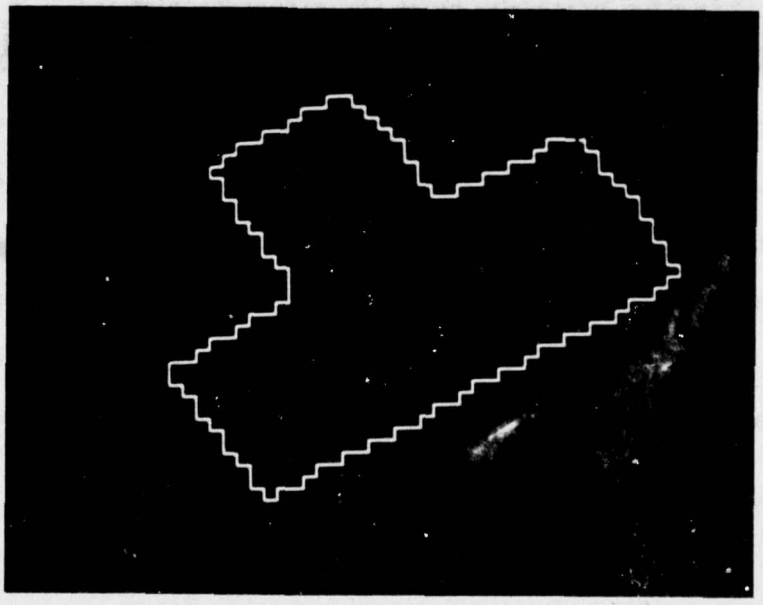


(b)

Figure 4.3 Testing Images



(a)



(b)

Figure 4.4 Boundary Chains of Unit Vectors

which performs the translation can be called a transducer [43]. We used a translation which can be formulated as a pushdown transducer or an attributed finite transducer [44]. For simplicity, we will explain the attributed finite transducer.

In the following definition, qj's are attributed states. Each state qj represents a subchain sj which is accepted but not translated, and which is described by the associated attributes. ℓ v denotes a series of unit vector v, or ℓ times v. -v is the negative of v, i.e., -v and v have the same length but opposite directions. A $\wedge$ B denotes that B follows A. δ is a mapping from Q x I, under condition C, to finite subsets of Q x O^*. The mapping performs when condition C is true. For each state transition, there is a set of attribute rules. The attributes of a state may be unit vectors or numbers. In the transition rules, i, m, n, v, u are unit vectors and p, q, ℓ, k are numbers. The attribute conditions for a transition are described above the right arrow. For each transition, the input unit vector is compared with the vector attributes and the attribute conditions are checked. Then the machine goes to the next state with the appropriate output and transfers the attributes according to the attribute rules. Each expression of the output is a vector ε O.

Definition 4.1: Attributed Finite Transducer A

A is a 6-tuple, (Q, I, O, δ, S, F).

I = the input set consisting of 4 unit vectors and an end marker \$,
{(1,0),(0,-1),(-1,0),(0,1),\$}

O = the output set, $\{(n_1,n_2) | (n_i = 0, n_{3-i} \neq 0)$
or $(n_i = \pm 1,\ n_{3-i}$ = any integer), $i = 1,2\}$

Q = a set of states with attributes, $\{qj | j = 0,\ldots,9\}$

$q0$: s0 = λ, empty

$q1_v$: s1 has one unit vector v

$q2_{\ell,v}$: s2 = ℓv, $\xrightarrow[\ell v]{}$, $\ell > 1$

$q3_{v,u}$: s3 = $v \wedge u$, $\overset{v}{\downarrow} u$ (or $\_\uparrow$)

$q4_{\ell,v,u}$: s4 = $\ell v \wedge u$, $\overset{\ell v}{\longrightarrow\!\!\downarrow} u$ (or $\_\_\uparrow$), $\ell > 1$

$q5_{v,u}$: s5 = $v \wedge u \wedge -v$, $\overset{v}{\underset{\leftarrow}{\sqcup}} u$ (or $\sqsubset$)

$q6_{v,u}$: s6 = $v \wedge u \wedge v$, $\overset{v}{\underset{\rightarrow}{\sqcup}}$ (or $\_\sqcap$)

$q7_{\ell,v,u,k}$: s7 = $\ell v \wedge u \wedge kv$, $\overset{\ell v}{\underset{kv}{\sqcap\!\!\rightarrow}}$ (or $\_\sqcap\!\!\rightarrow$)

$q8_{v,\ell,u}$: s8 = $v \wedge \ell u$, $\overset{v}{\underset{\ell u}{\sqcup\!\!\rightarrow}}$ (or $\sqcap\!\!\rightarrow$), $\ell > 1$

$q9_{\ell,v,uk}$: s9 = $\ell v \wedge u \wedge kv \wedge -u$, $\overset{\ell v}{\underset{kv}{\sqcup\!\uparrow}}$ (or $\_\sqcap\!\downarrow$),
$\ell > 1$, $k > 1$

S = the initial state q0

F = a set of final states, {q0}

δ: $(q0,i) \rightarrow (q1_{v},\phi)$, $v \leftarrow i$, $i \in I-\{\$\}$

$(q1_{m},m) \rightarrow (q2_{\ell,v},\phi)$, $v \leftarrow m$, $\ell \leftarrow 2$

$(q1_{m},i) \rightarrow (q3_{v,u},\phi)$, $v \leftarrow m$, $u \leftarrow i$

$(q2_{p,m},m) \rightarrow (q2_{\ell,v},\phi)$, $v \leftarrow m$, $\ell \leftarrow p+1$

$(q2_{p,m},i) \rightarrow (q4_{\ell,v,u},\phi)$, $\ell \leftarrow p$, $v \leftarrow m$, $u \leftarrow i$

$(q3_{m,n},m) \rightarrow (q6_{v,u},\phi)$, $v \leftarrow m$, $u \leftarrow n$

$(q3_{m,n},n) \rightarrow (q8_{v,\ell,u},\phi)$, $v \leftarrow m$, $u \leftarrow n$, $\ell \leftarrow 2$

$(q3_{m,n},-m) \rightarrow (q5_{v,u},\phi)$, $v \leftarrow m$, $u \leftarrow n$

$(q4_{p,m,n},m) \rightarrow (q7_{\ell,v,u,k},\phi)$, $\ell \leftarrow p$, $v \leftarrow m$, $k \leftarrow 1$, $u \leftarrow n$

$(q4_{p,m,n},-m) \rightarrow (q3_{v,u},pm)$ $v \leftarrow n$, $u \leftarrow v$

$(q4_{p,m,n},n) \rightarrow (q2_{\ell,v},pm)$ $\ell \leftarrow 2$, $v \leftarrow n$

$(q5_{m,n},-n) \rightarrow (q0,\phi)$

$(q5_{m,n},-m) \rightarrow (q2_{\ell,v},m+n)$, $\ell \leftarrow 2$, $v \leftarrow -m$

$(q5_{m,n},n) \rightarrow (q2_{\ell,v},\phi)$, $\ell \leftarrow 2$, $v \leftarrow n$

$(q6_{m,n},-n) \rightarrow (q9_{\ell,v,u,k},\phi)$, $v \leftarrow m$, $\ell \leftarrow 1$, $u \leftarrow n$, $k \leftarrow 1$

$(q6_{m,n},n) \rightarrow (q0,m+n \;/\; m+n)$

$(q6_{m,n,m}) \rightarrow (q7_{\ell,v,u,k,\phi})$ $\ell \leftarrow 1,\ k \leftarrow 2$, $v \leftarrow m,\ u \leftarrow n$

$(q7_{p,m,n,q,m}) \rightarrow (q7_{\ell,v,u,k,\phi})$ $v \leftarrow m,\ \ell \leftarrow p$, $u \leftarrow n,\ k \leftarrow q+1$

$(q7_{p,m,n,q,n})$

$p>2q$
$\rightarrow (q1_{v,(p-1)m/\ 2qm+n}),\ v \leftarrow n$

$q/2<p<2q$
$\rightarrow (q1_{v,(p+q)m+n}),\ v \leftarrow n$

$p<q/2$
$\rightarrow (q4_{\ell,v,u,2pm+n}),$ $\ell \leftarrow q-p$, $v \leftarrow m$, $u \leftarrow n$

$(q7_{p,m,n,q,-n}) \rightarrow (q9_{\ell,v,u,k,\phi})$ $\ell \leftarrow p$, $k \leftarrow q$, $v \leftarrow m$, $u \leftarrow n$

$(q8_{m,q,n,n}) \rightarrow (q8_{v,\ell,u,\phi}),$ $v \leftarrow m$, $u \leftarrow n$, $\ell \leftarrow q+1$

$(q8_{m,q,n,m})$

$q\underline{<}4$
$\rightarrow (q4_{\ell,v,u,m+n})$ $\ell \leftarrow q-1$, $v \leftarrow n$, $u \leftarrow m$

$q>4$
$\rightarrow (q4_{\ell,v,u,m+2n})$ $\ell \leftarrow q-2$, $v \leftarrow n$, $u \leftarrow m$

$(q8_{m,q,n,-m})$

$$(q4_{\ell,v,u,m+n}) \xrightarrow{q\leq 4} \begin{array}{l} \ell \leftarrow q-1 \\ v \leftarrow n \\ u \leftarrow -m \end{array}$$

$$(q4_{\ell,u,v,m+2n}) \xrightarrow{q>4} \begin{array}{l} \ell \leftarrow q-2 \\ v \leftarrow n \\ u \leftarrow -m \end{array}$$

$$(q9_{p,m,n,q,m}) \rightarrow (q2_{\ell,v,\phi}) \quad \begin{array}{l} \ell \leftarrow m+n+1 \\ v \leftarrow m \end{array}$$

$$(q9_{p,m,n,q,-n}) \rightarrow (q2_{\ell,v,(p+q)m+n}), \quad \begin{array}{l} \ell \leftarrow 2 \\ v \leftarrow -n \end{array}$$

$$(q9_{p,m,n,q,-m}) \rightarrow (q3_{v,u,(p+q)m+n}), \quad \begin{array}{l} v \leftarrow -n \\ u \leftarrow -m \end{array}$$

$$(q0,\$) \rightarrow (q0,\phi)$$

$$(q1_{m},\$) \rightarrow (q0,m)$$

$$(q2_{p,m},\$) \rightarrow (q0,pm)$$

$$(q3_{m,n},\$) \rightarrow (q0,m+n)$$

$$(q4_{p,m,n},\$) \rightarrow (q0,pm+n)$$

$$(q5_{m,n},\$) \rightarrow (q0,n)$$

$$(q6_{m,n},\$) \rightarrow (q0,2m+n)$$

$$(q7_{p,m,n,q},\$) \rightarrow (q0,(p+q)m+n)$$

$$(q8_{m,q,n},\$) \rightarrow (q0,m+qn)$$

$$(q9_{p,m,n,q},\$) \rightarrow (q0,(p+q)m)$$

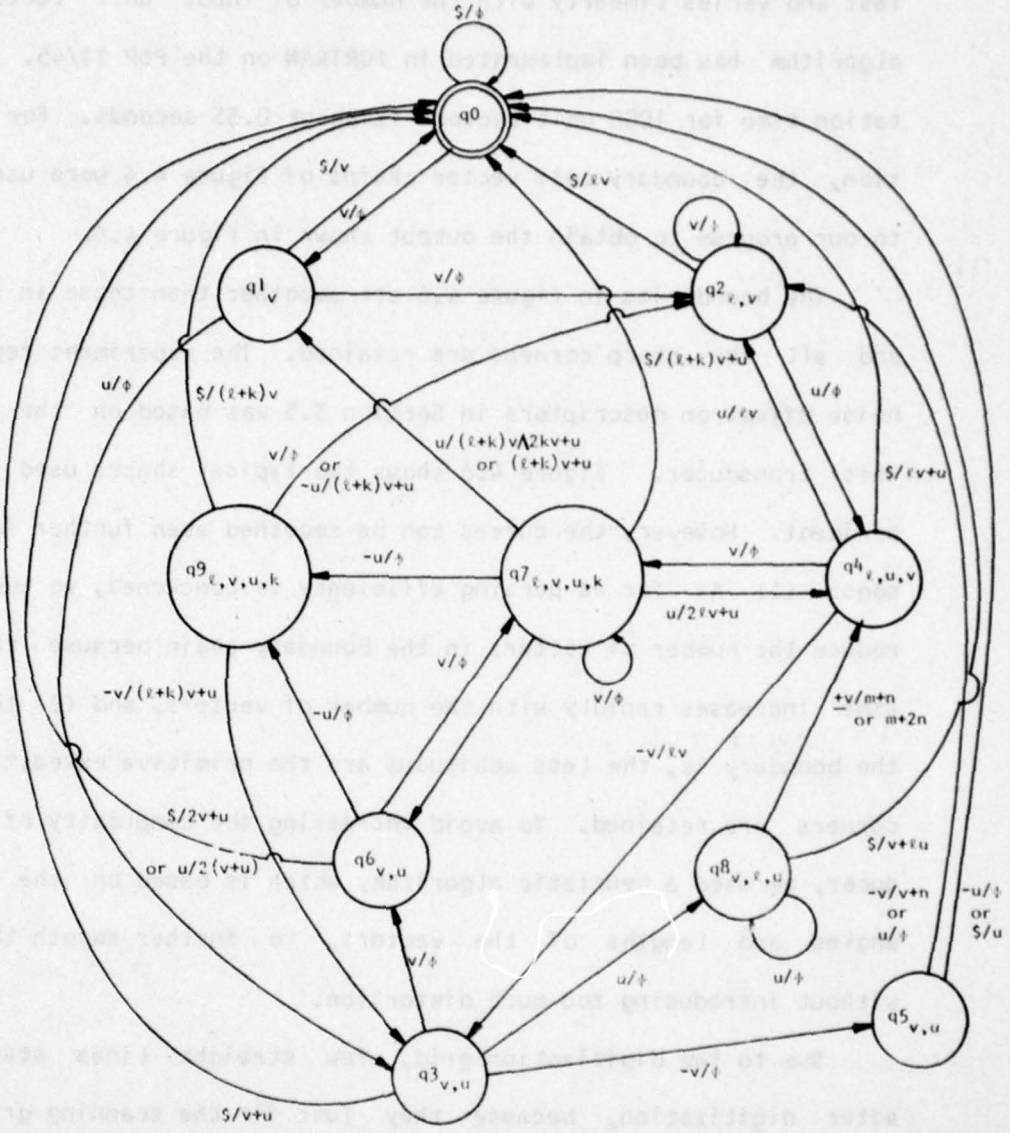
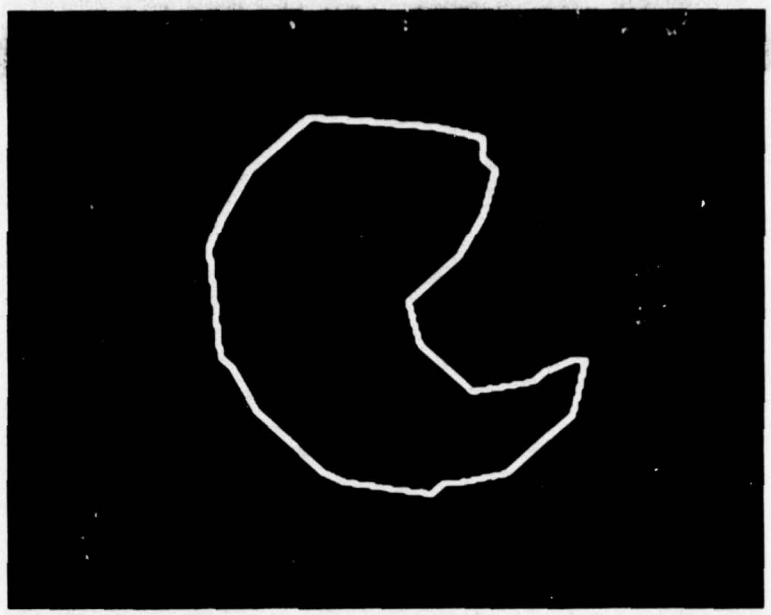


Figure 4.5 The State-Transition Diagram of Transducer A

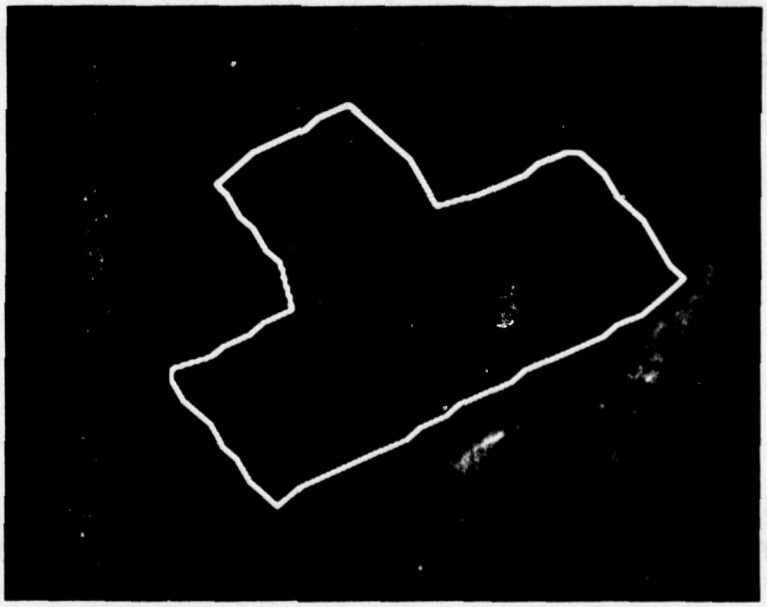
The state flow-chart of Transducer A is shown in Figure 4.5. Because Transducer A is deterministic, the computational speed is very fast and varies linearly with the number of input unit vectors. This algorithm has been implemented in FORTRAN on the PDP 11/45. The computation time for 1000 unit vectors is about 0.35 seconds. For illustration, the boundary unit vector chains of Figure 4.4 were used as input to our program to obtain the output shown in Figure 4.6.

The boundaries in Figure 4.6 are smoother than those in Figure 4.4 and all the sharp corners are retained. The experiment regarding the noise effect on descriptors in Section 3.5 was based on the output of this transducer. Figure 4.6 shows two typical shapes used in that experiment. However, the curves can be smoothed even further for two reasons: (1) As far as parsing efficiency is concerned, we would like to reduce the number of vectors in the boundary chain because the parsing time increases rapidly with the number of vectors, and (2) the smoother the boundary is, the less ambiguous are the primitive extractions if the corners are retained. To avoid increasing the complexity of our transducer, we used a heuristic algorithm, which is based on the connection angles and lengths of the vectors, to further smooth the boundary without introducing too much distortion.

Due to the digitization grid, few straight lines stay straight after digitization, because they just fit the scanning grid [71,72]. Let us consider the output chain of Transducer A. If there is no noise, a straight line could be coded as a chain of small vectors with angle changes up to $r_1 \approx 0.06\pi$. See Figure 4.7. When noise is present, the situation becomes much more complicated. If we assume that the noise



(a)



(b)

Figure 4.6 Boundary Chains Smoothed By Transducer A

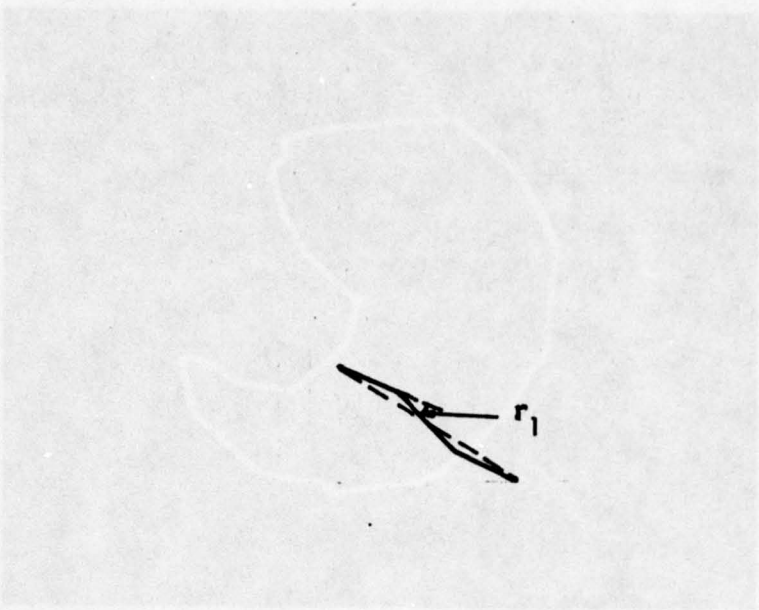


Figure 4.7 A Straight Line Distorted by Digitization Grid

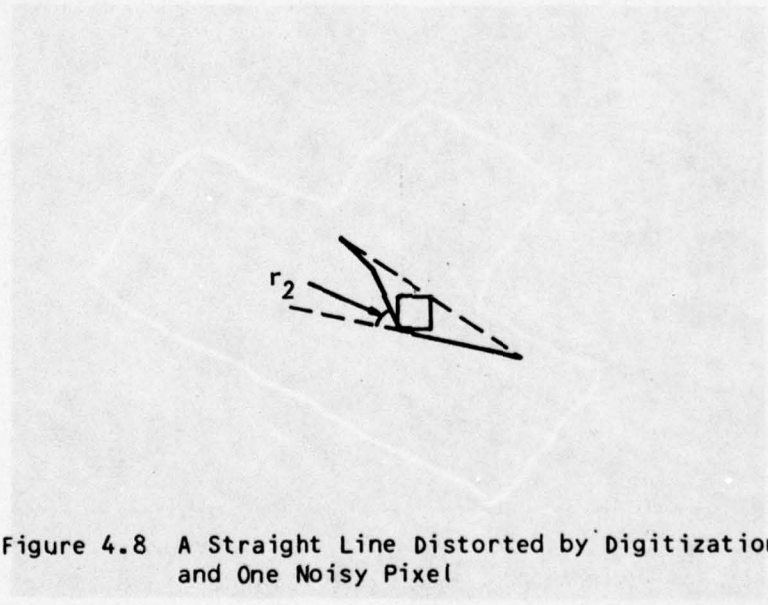


Figure 4.8 A Straight Line Distorted by Digitization Grid and One Noisy Pixel

does not move the false boundary more than one pixel away from the true boundary, and the probability of the noise occurrence is not very large, we can consider one pixel at a time. Figure 4.8 shows that $r_2 \approx 0.25\pi$.

In fact, the noisy pattern is much more complicated than expected. We use the following heuristic algorithm to detect the straight lines in the boundary. The algorithm has three parameters, A_1, A_2, and H. A_1, A_2, are angle distortion maxima and H is a vertical distortion maximum.

A vector subchain is approximated by a vector if the following four requirements are satisfied.

1. Total angular change is $\leq A_1$.

2. No angular change within the subchain is $> A_2$.

3. No two consecutive angular changes are $> A_1$, or $< -A_1$.

4. The distance from any point of the subchain to the approximated vector is $< H$.

A_1, A_2, and H can be any positive number. We used $A_1 = r_1 = 0.06\pi$, $A_2 = 2r_2 = 0.5\pi$, and $H = \sqrt{2}$. $\sqrt{2}$ is the diagonal of a pixel. r_1 and r_2 were obtained from previous noise analysis. We used $A_2 = 2r_2$ instead of r_2 because sometimes the boundary is noisier than one pixel at a time. Although there might be better choices for A_1, A_2, and H, these three numbers gave us reasonably good results.

Noise will make a short smooth curve a concavity or a convexity. This situation may not be detected because there may be only 2 vectors in the subchain and the first requirement is not satisfied. So, another simpler subroutine can be used to detect and reduce small concavities or

convexities by checking only the second and fourth requirements. But the small concavities/convexities can be smoothed out only when they occur in the convex/concave portion of the boundary.

After the boundary vector chain is smoothed, a simple procedure is performed to locally find a sharp convex angle to be the starting point. The whole boundary smoothing algorithm is summarized as follows.

Algorithm 4.3: The Smoothing Algorithm

Input: A vector chain from Transducer A.

Output: A smoothed vector chain.

(1) Set A_1, A_2, and H.

(2) Compute length of each vector and angle between vectors.

(3) Combine consecutive vectors whose connection angles are zero.

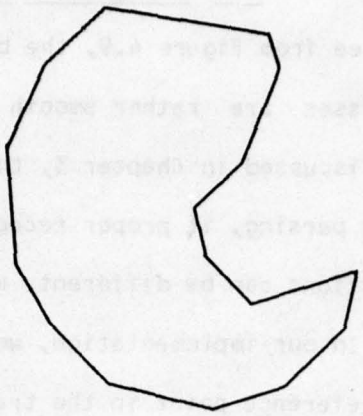
(4) Approximate sub-vector-chains with longer vectors by using the four requirements.

(5) Reduce small concavities and convexities.

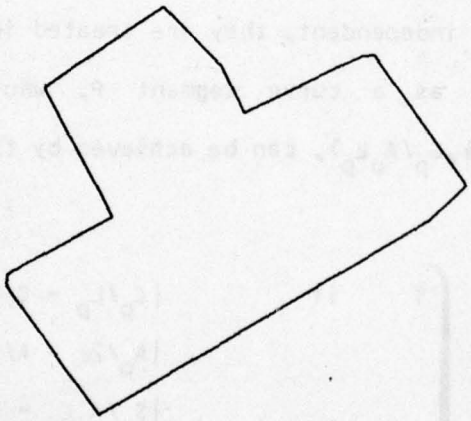
(6) Find a local sharp convex point to be the starting point, and shift the chain with respect to the starting point.

(7) Terminate.

We applied this algorithm to the shapes in Figure 4.6 and obtained the results shown in Figure 4.9.



(a)



(b)

Figure 4.9 Boundary Chains Smoothed By Algorithm 4.3

4.4 Recognition Functions

As we can see from Figure 4.9, the boundary chains following the smoothing processes are rather smooth and accurate, although they are not exact. As discussed in Chapter 3, the primitive extraction can be done during the parsing, if proper recognition functions are used. The recognition functions can be different with different primitives and nonterminals. In our implementation, we assumed each primitive or nonterminal has a reference point in the transformed space. We used the same recognition function for all the primitives for simplicity and a slightly different function for all the nonterminals. These functions are derived from the study of noise effect on normalized descriptors. (See Section 3.5).

A C-descriptor $(\vec{C},L,A,S)$ of $V = v_1 \ldots v_m$ can be normalized by N. N: $(\vec{C},L,A,S) \rightarrow (C/L,\ A/2\pi,\ S/AL)$. Since the three variables are supposed to be independent, they are treated independently. The recognition of V as a curve segment P, whose normalized descriptor is $(C_p/L_p, A_p/2\pi, S_p/A_pL_p)$, can be achieved by the following function.

$$R(P,V) = \begin{cases} 1 & \text{if} \quad |C_p/L_p - C/L| \leq t_1 \\ & \qquad |A_p/2\pi - A/2\pi| \leq t_2 \\ & \qquad |S_p/A_pL_p - S/AL| \leq t_3 \\ & \text{and} \quad |L_p/L_{op} - L/L_o| \leq t_4 \\ 0 & \text{otherwise} \end{cases}$$

where L_p/L_{op} and L/L_o are the sizes of the curve segment proportional to the whole boundary. t's are threshold values.

Further investigation of this function disclosed that R is closer to human perception if t_1 is proportional to C_p/L_p. For an open curve of constant length L, the proportional difference in width of the opening makes more sense in discrimination than the absolute difference. Therefore, the function works better if $t_1 = k_1(C_p/L_p)$. The same reasoning applies to the fourth inequality, so $t_4 = k_4(L_p/L_{op})$. Since A can be zero, we modify the fourth inequality to avoid the possibility of zero in the denominator of S/AL.

$$\left|\frac{S_p}{L_p} - \frac{S}{L}\right| \leq t_3\alpha, \quad \text{where } \alpha = \begin{cases} |A_p|, & \text{if } |A_p| > k_a \\ k_a, & \text{if } |A_p| \leq k_a \end{cases}$$

and k_a is a positive number.

When S_p and S have opposite signs, the corresponding P and V decline to opposite directions. Perceptionally, they decline differently though they may be little different in value. So, we may use a tighter threshold, when S_p and S have different signs. Thus, we modify t_3 by setting

$$t_3 = \begin{cases} 0.5\ k_3 & \text{if } S_p \cdot S \leq 0 \\ k_3 & \text{if otherwise.} \end{cases}$$

As mentioned before, S somehow measures the degree of declination of a simple curve segment. When the curve is complex, the declination becomes less significant. Besides, the experiments show that S is very sensitive to noise occurring close to the ends of the curve segment, when the curve segment is complex. Therefore, S is not used in recognizing nonterminals. The recognition function for curve segments is as-

sumed as follows.

Assumption 4.1:

$$V = v_1 \ldots v_m \stackrel{r}{=} P, \text{ iff } R(P,V) = 1, \text{ where}$$

$$D(P) = (\vec{C}, L_p, A_p, S_p),\ D(V) = (\vec{C}, L, A, S)$$

$$R(P,V) = \begin{cases} 1 & \text{if } \left|\frac{C_p}{L_p} - \frac{C}{L}\right| \frac{L_p}{C_p} \leq k_1 \\ & \\ & |A_p - A| \leq k_2 \\ & \\ & \left|\frac{S_p}{L_p} - \frac{S}{L}\right| f \leq t_3^{\alpha} \\ & \\ & \text{and} \\ & \\ & \left|\frac{L_p}{L_{op}} - \frac{L}{L_o}\right| \frac{L_{op}}{L_p} \leq k_4 \\ & \\ 0 & \text{otherwise} \end{cases}$$

where $k_2 = 2\pi\ t_2$,

$$f = \begin{cases} 1 & \text{if } P \text{ is a curve primitive} \\ 0 & \text{if } P \text{ is a } \textit{nonterminal} \end{cases}$$

$$t_3 = \begin{cases} k_3 & \text{if } S_p \cdot S > 0 \\ 0.5\ k_3 & \text{if } S_p \cdot S \leq 0 \end{cases}$$

$$\alpha = \begin{cases} |A_p| & \text{if } |A_p| > k_a \\ k_a & \text{if } |A_p| \leq k_a \end{cases}$$

and k_1, k_2, k_3, k_4, and k_a are positive constants.

This recognition function is developed from the noise analysis in Section 3.5. The reasons for which we developed such a function are that (1) it is computationally simple and (2) we do not have a large number of samples to obtain a complete noise analysis. Basically, each curve primitive or nonterminal is given a recognition region in the transformed descriptor space. If the transformed description of V falls in the recognition region of P, V is recognized as P. Otherwise, V is not recognized as P. The above k's specify the size of the recognition region. The greater the k's are, the bigger the region is. The extreme cases are (1) if k's are zero's, only the curve segments which have exactly the same transformed descriptor as P can be recognized, and (2) if k's are very large numbers, all possible curve segments can be recognized as P. Therefore, we want to choose k's which are large enough to recognize the noisy curve segments but small enough so that the wrong curve segments will not be recognized. Fortunately, the production rules can be used to eliminate the wrong curve segments whose transformed descriptors fall in the recognition region. This idea was explained in the PEE parsing in Section 3.6. Hence, the size of the recognition region is not very critical, as long as it is big enough to cover the noisy curve segments. But, we don't like to use a very big recognition region even if the production rules are sufficient to describe the structural differences, because a bigger recognition region will accept a larger number of candidates to the parsing table. Though the production rules may eliminate the wrong candidates, the processing time will be longer. Therefore, we like to find the recognition region which is barely big enough to cover the noisy cases to minimize the com-

puting time. From the noise analysis of Section 3.5, we have an idea of the distribution of the normalized descriptors. Here, we simply choose the k's whose corresponding recognition region is about two to three times as wide as the distribution in each dimension of the transformed space. The numbers chosen for our experiments are $0.3 \leq k_1 \leq 0.45$, $0.2 \times \hat{a}r \leq k_2 \leq 0.3 \times \hat{a}r$, $0.4 \leq k_3 \leq 0.6$, $0.2 \leq k_4 \leq 0.3$, $k_a = 0.4$. We used smaller k's to recognize shapes which are less noisy or to discriminate classes which are different only in small details, and greater k's to recognize noisier shapes or to discriminate classes which are very different in structure. The variation of k's is somewhat determined by the relationship between the noise and the recognition region.

The function for recognizing angle primitives is suggested as follows.

Assumption 4.2:

$V = v_1 \ldots v_m \overset{\Gamma}{=} A$, iff $R(A,V) = 1$, where V has angle length, ℓ, from v_2 to v_{m-1}, and total angular change, a, from v_1 to v_m, and

$$R(A,V) = \begin{cases} 1 & \text{if } \ell \leq k_c \\ & \quad |A - a| \leq k_5 \\ 0 & \text{otherwise} \end{cases}$$

k_c is the corner tolerance mentioned in Chapter 3 and k_5 is a positive constant.

$k_5 = r_2 = 0.25\pi$ is obtained from the noise analysis in Section 4.4. An angle primitive is supposed to have no length. But it may be smoothed to cover a few pixels when noise is present. The corner tolerance k_c allows an angle to cover a length up to k_c. Of course, k_c

depends on the noisy situation as well as the resolution of the picture. In our experiments, we used $6 \leq k_c \leq 8$. A k_c smaller than 6 may often cause erroneous rejection, because it is not large enough to take care of some noisy angles. A k_c greater than 8 does not seem reasonable simply because an angle of length longer than 8 does not look like an angle anymore.

4.5 Recognition Experiments

In this section we intend to demonstrate the recognition capabilities of our proposed method with two experiments. The first experiment shows the capability of recognizing arbitrary shapes. The second experiment shows its discriminating ability with shapes which have similar structure but different boundary details. In both experiments, we also discuss the feasibility of applying the two parsing schemes described in Section 3.6. Of course, the recognition functions described in Assumptions 4.1 and 4.2 were used. The context-free grammars used in this section were constructed interactively. Each shape grammar was designed on the basis of one model shape which was obtained by interactively modifying one of the testing shapes to reduce the noise effect. In other words, we first tried to obtain a noise-free model shape, and then designed the grammar based on this model shape. Both steps are interactive. The grammatical inference will be discussed in detail in Chapter 6. Therefore, none of the testing patterns used in this section is a training pattern.

Experiment 4.1:

Figure 4.10 shows two arbitrary angle views of airplane models B52 and F102. They are very much different in structure. We constructed grammars $G_{B52,A}$ and $G_{F102,A}$ interactively to describe them respectively. The corresponding segmentations are illustrated in Figures 4.11 and 4.12. The attribute rules in the following grammars are omitted for simplicity.

Eleven shapes with respect to various rotations of Figure 4.10(a) and nine shapes with respect to various rotations of Figure 4.10(b) were obtained through all the procedures described in Sections 4.2 and 4.3. They can be found in Appendix B. These shapes were processed by the PEE Earley's parser with the above two grammars, $G_{B52,A}$ and $G_{F102,A}$. The 40 tests, 20 shapes for each of the 2 grammars, were all correct. Since some B52's are quite noisy, we use a bigger recognition region. We used $k_1 = 0.45$, $k_2 = 0.3 \times \hat{a}$, $k_3 = 0.6$, $k_4 = 0.3$ in this experiment. The average processing time per test is 0.19 seconds for a chain of 30 vectors and 0.29 seconds for a chain of 36 vectors. We then converted our grammars to a finite-state form by an algorithm, which will be discussed in Chapter 6. And we processed these shapes by the PEE finite automaton with two converted finite-state grammars, $F_{B52,A}$ and $F_{F102,A}$. The 40 tests were also all correct. The average processing time per test is 0.024 seconds for a chain of 30 vectors and 0.03 seconds for a chain of 36 vectors.

The context-free shape grammars (CFSG) $G_{B52,A}$ and $G_{F102,A}$ and the finite-state shape grammars (FSSG) $F_{B52,A}$ and $F_{B52,A}$ are listed in the following pages. As mentioned in Chapter 3, the primitives used to

(a) B52,A

(b) F102,A

Figure 4.10 Two Arbitrary Views

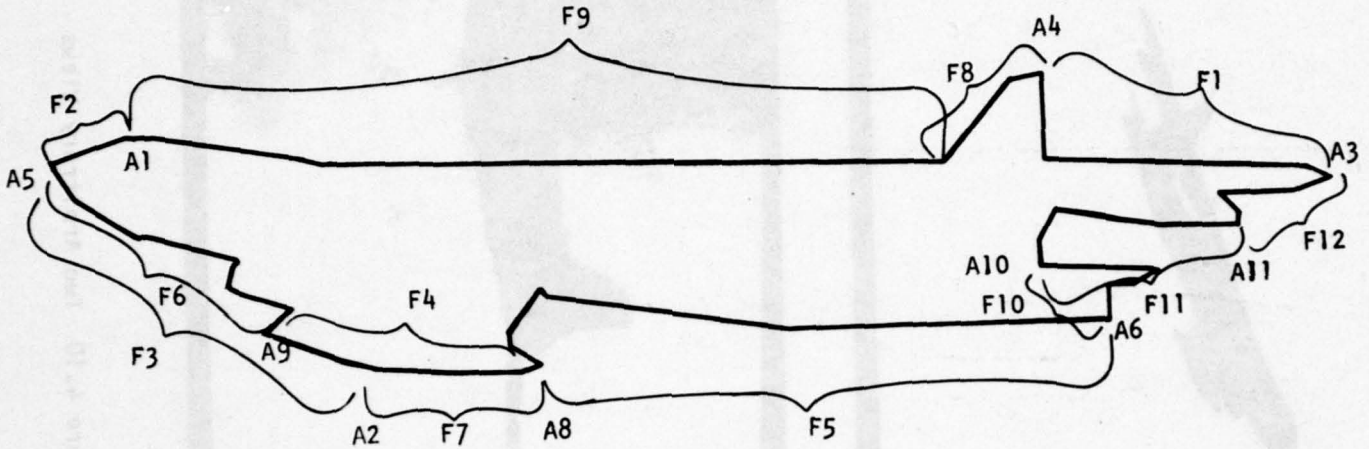


Figure 4.11 The B52,A Segmentation

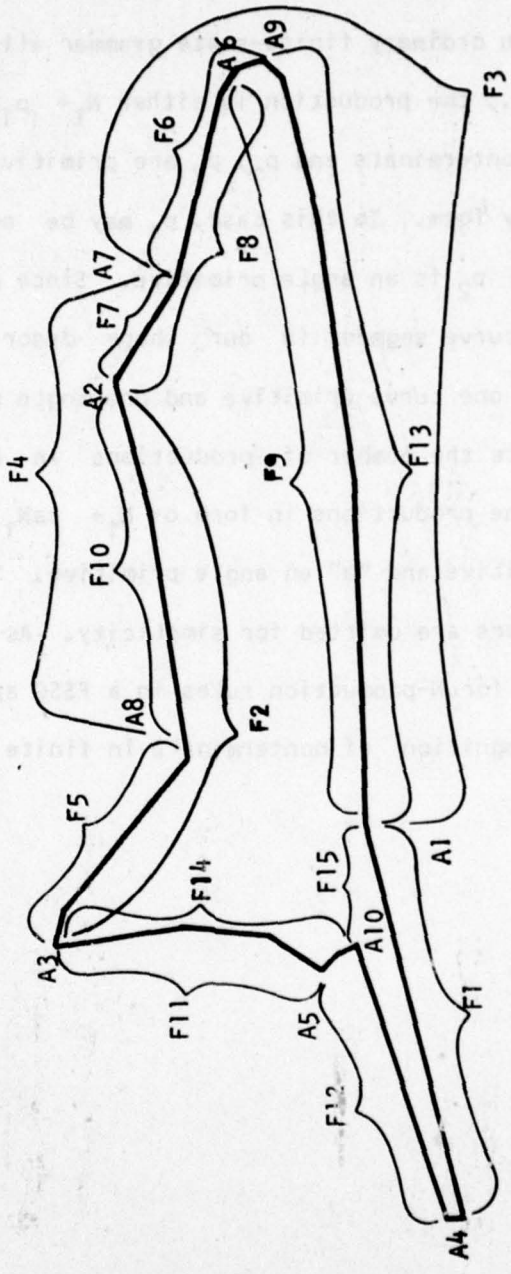


Figure 4.12 The F102,A Segmentation

describe a shape can be flexibly assigned, no matter whether the grammar is a CFSG or a FSSG. For the convenience of comparison, the CFSG and the converted FSSG discussed here have the same primitive set. The definition of an ordinary finite-state grammar allows one primitive in a production, i.e., the production is either $N_1 \rightarrow p_1 N_2$ or $N_1 \rightarrow p_2$, where N_1 and N_2 are nonterminals and p_1, p_2 are primitives. Our FSSG's can be of this ordinary form. In this case, p_1 may be an angle or a curve primitive, and p_2 is an angle primitive. Since an angle primitive always follows a curve segment in our shape descriptions (see Section 3.4), we allow one curve primitive and one angle primitive in each production to reduce the number of productions in half. That is, our FSSG's have the productions in form of $N_1 \rightarrow caN_1$ or $N_1 \rightarrow ca$, where "c" is a curve primitive and "a" an angle primitive. The attribute rules in following grammars are omitted for simplicity. As a matter of fact, the attribute rules for N-production rules in a FSSG are not necessary, because the recognition of nonterminals in finite automaton is not performed.

$G_{B52,A} = (V_{B52,A}, T_{B52,A}, P_{B52,A}, S_{B52,A})$

$V_{B52,A} = \{ S_{B52,A}, NI'S \}, \quad I=1,\ldots, 8$

$T_{B52,A} = \{ FJ'S, AK'S \}, \quad J=1,\ldots, 12, \quad K=1,\ldots, 11$

$P_{B52,A}$

S --> N1 A1 N2 A2 N3 A3

S --> F1 A4 N4 A5 N5 A6 N6 A3

S --> N1 A1 F2 A5 N5 A6 N6 A3

S --> N4 A5 N5 A6 N6 A3 F1 A4

S --> F2 A5 N5 A6 N6 A3 N1 A1

S --> N2 A2 N3 A3 N1 A1

S --> N5 A6 N6 A3 F1 A4 N4 A5

S --> F3 A2 N3 A3 N1 A1 F2 A5

S --> F4 A8 F5 A6 N6 A3 N7 A5 F6 A9

S --> N3 A3 N1 A1 N2 A2

S --> N3 A3 N7 A5 F3 A2

S --> F5 A6 N6 A3 N1 A1 N2 A2 F7 A8

S --> N6 A3 N7 A5 N5 A6

S --> N6 A3 N1 A1 N2 A2 N8 A6

N7 --> N1 A1 F2

N7 --> F1 A4 N4

N2 --> F2 A5 F3

N1 --> F1 A4 F8 A7 F9

N3 --> F7 A8 F5 A6 N6

N5 --> F3 A2 N8

N5 --> F6 A9 F4 A8 F5

N6 --> F10 A10 F11 A11 F12

N8 --> F7 A8 F5

N4 --> F8 A7 F9 A1 F2

$$F_{B52,A} = (V_{B52,A}, T_{B52,A}, P_{B52,A}, S_{B52,A})$$

$$V_{B52,A} = \left\{ S_{B52,A}, NI'S \right\}, \quad I=1,\ldots,84$$

$$T_{B52,A} = \left\{ FJ'S, AK'S \right\}, \quad J=1,\ldots,12, \quad K=1,\ldots,11$$

$P_{B52,A}$:

S	-->	F1	A4	N1	N9	-->	F11 A11 N21
S	-->	F8	A7	N2	N21	-->	F12 A3 N22
S	-->	F2	A5	N3	N11	-->	F2 A5 N23
S	-->	F6	A9	N4	N15	-->	F5 A6 N24
S	-->	F3	A2	N5	N17	-->	F10 A10 N25
S	-->	F4	A8	N6	N22	-->	F1 A4 N26
S	-->	F7	A8	N7	N23	-->	F6 A9 N27
S	-->	F5	A6	N8	N23	-->	F3 A2 N28
S	-->	F10	A10	N9	N24	-->	F10 A10 N29
N1	-->	F8	A7	N10	N10	-->	F9 A1 N30
N2	-->	F9	A1	N11	N16	-->	F5 A6 N31
N3	-->	F3	A2	N12	N13	-->	F4 A8 N14
N3	-->	F6	A9	N13	N14	-->	F5 A6 N32
N12	-->	F7	A8	N14	N19	-->	F11 A11 N33
N4	-->	F4	A8	N15	N20	-->	F11 A11 N34
N5	-->	F7	A8	N16	N26	-->	F8 A7 N35
N6	-->	F5	A6	N17	N25	-->	F11 A11 N36
N7	-->	F5	A6	N18	N27	-->	F4 A8 N37
N18	-->	F10	A10	N19	N29	-->	F11 A11 N38
N8	-->	F10	A10	N20	N28	-->	F7 A8 N37

N31--> F10 A10 N39

N33--> F12 A3 N40

N35--> F9 A1 N41

N30--> F2 A5 N42

N32--> F10 A10 N43

N40--> F1 A4 N44

N34--> F12 A3 N45

N41--> F2 A5 N46

N36--> F12 A3 N47

N37--> F5 A6 N48

N38--> F12 A3 N49

N42--> F6 A9 N50

N42--> F3 A2 N51

N45--> F1 A4 N52

N47--> F1 A4 N53

N48--> F10 A10 N54

N49--> F1 A4 N55

N55--> F8 A7 N56

N39--> F11 A11 N57

N44--> F8 A7 N58

N46--> F3 A2 N59

N46--> F6 A9 N60

N43--> F11 A11 N61

N59--> F7 A8 N62

N50--> F4 A8 N63

N52--> F8 A7 N64

N53--> F8 A7 N65

N54--> F11 A11 N66

N56--> F9 A1 N67

N51--> F7 A8 N63

N58--> F9 A1 N68

N57--> F12 A3 N69

N60--> F4 A8 N62

N61--> F12 A3 N70

N62--> F5 A6

N71--> F12 A3

N63--> F5 A6 N72

N64--> F9 A1 N73

N65--> F9 A1 N74

N66--> F12 A3 N75

N67--> F2 A5

N68--> F2 A5 N76

N69--> F1 A4 N77

N70--> F1 A4 N78

N72--> F10 A10 N79

N73--> F2 A5 N80

N75--> F1 A4

N74--> F2 A5 N81

N76--> F3 A2

N77--> F8 A7 N82

N78--> F8 A7 N83

N79--> F11 A11 N71

N80--> F3 A2 N84

N81--> F6 A9

N84--> F7 A8

N82--> F9 A1 N67

N83--> F9 A1

$G_{F102,A} = (V_{F102,A}, T_{F102,A}, P_{F102,A}, S_{F102,A})$

$V_{F102,A} = \{ S_{F102,A}, NI'S \}, I=1,\ldots, 4$

$T_{F102,A} = \{ FJ'S, AK'S \}, J=1,\ldots, 15, K=1,\ldots, 10$

$P_{F102,A}$:

S --> F1 A1 N1 A2 F2 A3 N2 A4

S --> F2 A3 N2 A4 F1 A1 N1 A2

S --> N2 A4 F1 A1 N1 A2 F2 A3

S --> F1 A1 F3 A7 F4 A8 F5 A3 N2 A4

S --> N2 A4 F1 A1 F3 A7 F4 A8 F5 A3

S --> F6 A7 F7 A2 F2 A3 N2 A4 N3 A9

S --> F6 A7 F4 A8 F5 A3 N2 A4 N3 A9

S --> F8 A2 F2 A3 N2 A4 F1 A1 F9 A6

S --> F10 A8 F5 A3 N4 A1 N1 A2

N2 --> F11 A5 F12

N1 --> F9 A6 F8

N1 --> F3 A7 F7

N3 --> F1 A1 F13

N4 --> N2 A4 F1

N4 --> F14 A10 F15

$$F_{F102,A} = (V_{F102,A}, T_{F102,A}, P_{F102,A}, S_{F102,A})$$

$$V_{F102,A} = \left\{ S_{F102,A}, NI'S \right\}, \quad I=1,\ldots,40$$

$$T_{F102,A} = \left\{ FJ'S, AK'S \right\}, \quad J=1,\ldots,15, \quad K=1,\ldots,10$$

$P_{F102,A}$:

S	-->	F1	A1	N1	N12-->	F5	A3	N21
S	-->	F2	A3	N2	N19-->	F7	A2	N22
S	-->	F11	A5	N3	N19-->	F4	A8	N23
S	-->	F6	A7	N4	N15-->	F2	A3	N16
S	-->	F8	A2	N5	N16-->	F11	A5	N24
S	-->	F10	A8	N6	N17-->	F11	A5	N25
N1	-->	F3	A7	N7	N18-->	F14	A10	N26
N1	-->	F9	A6	N8	N18-->	F11	A5	N27
N2	-->	F11	A5	N9	N21-->	F11	A5	N28
N3	-->	F12	A4	N10	N23-->	F5	A3	
N7	-->	F7	A2	N11	N11-->	F2	A3	N29
N7	-->	F4	A8	N12	N9 -->	F12	A4	N30
N10	-->	F1	A1	N13	N22-->	F2	A3	
N4	-->	F4	A8	N14	N24-->	F12	A4	N31
N4	-->	F7	A2	N15	N25-->	F12	A4	N32
N14	-->	F5	A3	N16	N26-->	F15	A1	N33
N5	-->	F2	A3	N17	N28-->	F12	A4	
N6	-->	F5	A3	N18	N27-->	F12	A4	N34
N13	-->	F3	A7	N19	N20-->	F8	A2	N22
N13	-->	F9	A6	N20	N8 -->	F8	A2	N35

```
N29--> F11 A5  N28
N30--> F1  A1  N33
N31--> F1  A1  N36
N32--> F1  A1  N37
N33--> F3  A7  N38
N33--> F9  A6  N39
N35--> F2  A3  N29
N37--> F9  A6
N34--> F1  A1  N40
N36--> F13 A9
N38--> F7  A2
N39--> F8  A2
N40--> F3  A7  N38
N40--> F9  A6  N39
```

Experiment 4.2:

Figure 4.13 shows three views of two airplane models, F86 and MIG-15. They are very similar in structure. T, V, and U represent different angle views. They are all close to the top view. (a) differs from (b) and (c) by two small missile-tails and a machine gun on the right wing. But the machine gun may not appear in the digitized picture. We can construct a grammar, $G_{F86,T}$, to distinguish it from MIG-15's. The corresponding segmentation can be found in Figure 4.14. As we explained in Section 3.7, the shape of F86,T may be recognized by grammars of MIG-15, but the shapes of MIG-15 cannot be recognized by $G_{F86,T}$ because MIG-15 shapes do not have missile-tails.

MIG-15,V and MIG-15,U are different mainly in the width of the fuselage close to the tail. The whole tail may not be designated as a primitive, since it contains two tail ends which may be the starting points. We need to check the nonterminal which represents the whole tail or the whole shape excluding the tail. A grammar $G_{MIG-15,U}$ is constructed to distinguish the two angle views. The corresponding segmentation can be found in Figure 4.15.

In this experiment, the recognition of these three angle views forms a simple classification tree shown in Figure 4.16. Through all the procedures described in Sections 4.2, 4.3, we obtained 5 F86,T's, 5 MIG-15,V's, and 3 MIG-15,U's with respect to arbitrary rotations. They can be found in Appendix C. At the first stage, the unknown shapes can be processed by the PEE Earley's parser with $G_{F86,T}$ or by the PEE finite automaton with $F_{F86,T}$, which is the converted finite-state shape grammar. Either parsing algorithm can achieve completely correct recogni-

(a) F86,T

(b) MIG-15,V

(c) MIG-15,U

Figure 4.13 Three Views of Two Models

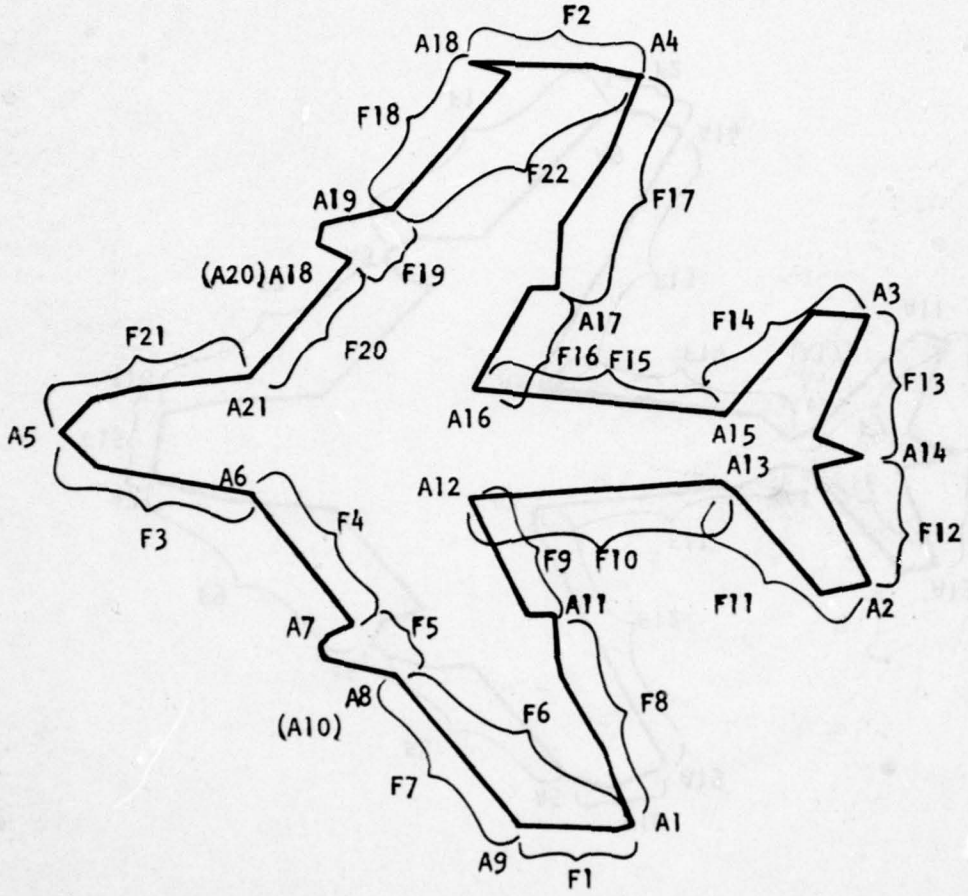


Figure 4.14 The F86,T Segmentation

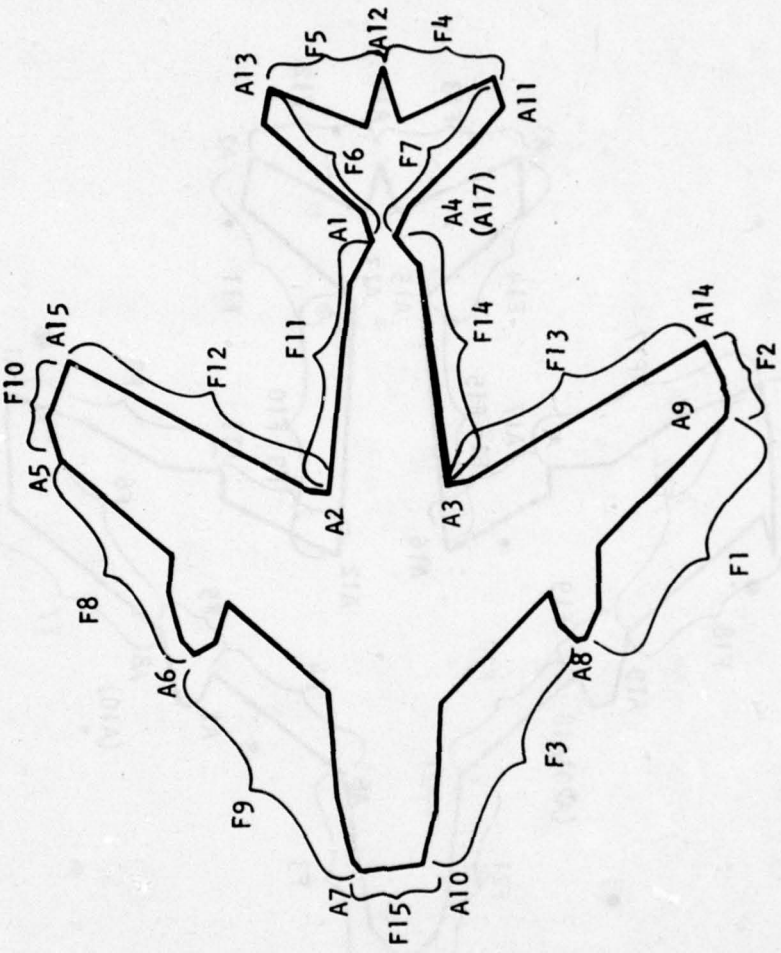


Figure 4.15 The MIG-15,U Segmentation

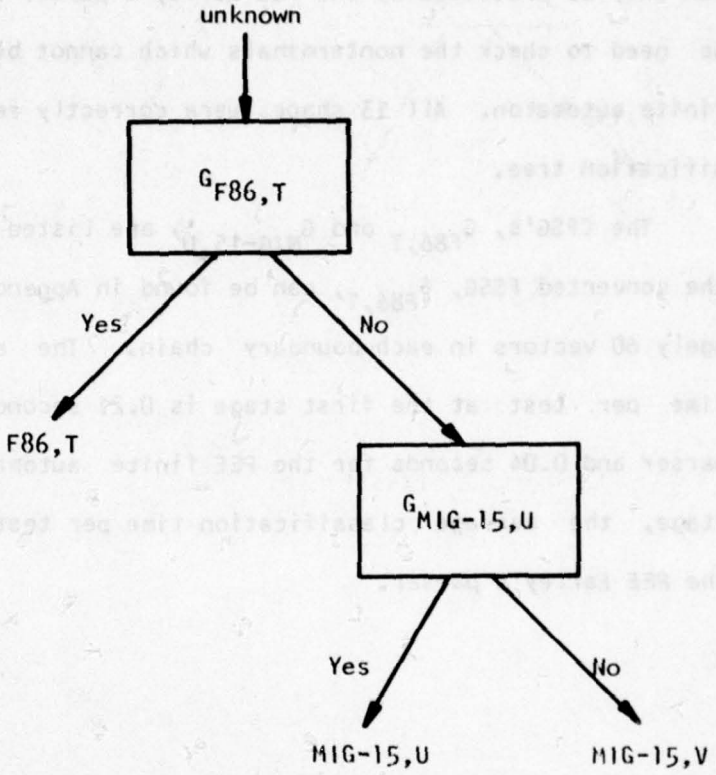


Figure 4.16 A Simple Classification Tree of 3 Classes

tion because the recognition of primitives is sufficient to distinguish F86,T from MIG-15. The recognition functions described in Assumptions 4.1 and 4.2 with smaller constants, k's were used. $k_1 = 0.3$, $k_2 = 0.2 \times 2$, $k_3 = 0.4$, and $k_4 = 0.2$. At the second stage, the shapes can only be processed by the PEE Earley's parser with $G_{MIG-15,U}$, because we need to check the nonterminals which cannot be recognized in the PEE finite automaton. All 13 shapes were correctly recognized via the classification tree.

The CFSG's, $G_{F86,T}$ and $G_{MIG-15,U}$, are listed in following pages and the converted FSSG, $F_{F86,T}$, can be found in Appendix D. There are averagely 60 vectors in each boundary chain. The average classification time per test at the first stage is 0.21 seconds for the PEE Earley's parser and 0.04 seconds for the PEE finite automaton. At the second stage, the average classification time per test is 0.24 seconds using the PEE Earley's parser.

$G_{F86,T} = (V_{F86,T}, T_{F86,T}, P_{F86,T}, S_{F86,T})$

$V_{F86,T} = \{ S_{F86,T}, NI'S \}, \quad I=1,\ldots, 8$

$T_{F86,T} = \{ FJ'S, AK'S \}, \quad J=1,\ldots, 22, \quad K=1,\ldots, 21$

$P_{F86,T}$:

S --> N1 A1 N2 A2 N3 A3 N4 A4 N5 A5

S --> N2 A2 N3 A3 N4 A4 N5 A5 N1 A1

S --> N3 A3 N4 A4 N5 A5 N1 A1 N2 A2

S --> N4 A4 N5 A5 N1 A1 N2 A2 N3 A3

S --> N5 A5 N1 A1 N2 A2 N3 A3 N4 A4

S --> F1 A1 N6 A4 N5 A5 N7 A9

S --> N8 A5 N1 A1 N6 A4 F2 A18

N1 --> F3 A6 F4 A7 F5 A8 F6

N6 --> N2 A2 N3 A3 N4

N1 --> N7 A9 F1

N7 --> F3 A6 F4 A7 F5 A10 F7

N2 --> F8 A11 F9 A12 F10 A13 F11

N3 --> F12 A14 F13

N4 --> F14 A15 F15 A16 F16 A17 F17

N5 --> F2 A18 N8

N8 --> F18 A19 F19 A20 F20 A21 F21

N5 --> F22 A19 F19 A20 F20 A21 F21

$G_{MIG-15,U} = (V_{MIG-15,U}, T_{MIG-15,U}, P_{MIG-15,U}, S_{MIG-15,U})$

$V_{MIG-15,U} = \{ S_{MIG-15,U}, NI'S \}, \quad I=1,\ldots, 8$

$T_{MIG-15,U} = \{ FJ'S, AK'S \}, \quad J=1,\ldots, 15, \quad K=1,\ldots, 17$

$P_{MIG-15,U}$:

S --> N1 A14 N2 A15 N3 A10

S --> F1 A9 F2 A16 N2 A15 N3 A10 F3 A8

S --> F2 A16 N2 A15 N3 A10 F3 A8 F1 A9

S --> N2 A15 N3 A10 N1 A14

S --> F4 A12 F5 A13 F6 A1 N4 A17 F7 A11

S --> F5 A13 F6 A1 N4 A17 F7 A11 F4 A12

S --> F6 A1 N4 A17 F7 A11 F4 A12 F5 A13

S --> N5 A14 N2 A15

S --> F8 A6 F9 A7 N6 A14 N2 A15 F10 A5

S --> F9 A7 N6 A14 N2 A15 F10 A5 F8 A6

S --> N6 A14 N2 A15 N7 A7

N8 --> F7 A11 F4 A12 F5 A13 F6

N4 --> F11 A2 F12 A15 N5 A14 F13 A3 F14

N7 --> F10 A5 F8 A6 F9

N3 --> F10 A5 F8 A6 F9 A7 F15

N1 --> F3 A8 F1 A9 F2

N6 --> F15 A10 F3 A8 F1 A9 F2

N2 --> F13 A3 F14 A4 N8 A1 F11 A2 F12

N5 --> N3 A10 N1

4.6 Discussion

The previous experiments demonstrated that our proposed method can discriminate shapes by their differences in global structure, boundary details, or parts of shapes. They are described in terms of production rules, primitives, and nonterminals respectively. The number of testing patterns, 33, is not large enough to claim one hundred percent accuracy, although these 33 are all correctly recognized. However, this performance appears to be quite good to demonstrate the recognition capability of our proposed syntactic method. In addition, this syntactic method with error-correcting techniques can recognize partially distorted shapes. The distorted shape recognition will be discussed in Chapter 5.

Although the results of these experiments seem satisfactory, further discussion on the following four questions are necessary: (1) What is the limitation of the recognition power?, (2) How do we apply this method to recognize 3-dimensional objects?, (3) Can the grammars be inferred automatically?, and (4) How do we convert a context-free shape grammar to a finite-state shape grammar, if only a finite-state shape grammar is necessary for recognition?

The latter two questions will be discussed in Chapter 6. Using this method to recognize a 3-dimensional object, we must realize that there are often many different angle views for a rigid object. Each view can be described by a subgrammar, whose S symbol has a label indicating the viewing angle. Since the subgrammars are independent, they can be parallelly processed, if a multiprocessor system is used. Otherwise, all these subgrammars can be put together to form a single grammar. Some of the subgrammars have very similar primitives. For in-

stance, some views have similar nose and wings. The combined grammar can be simplified by combining the similar primitives, the nonterminals and the same production rules. The simplification will reduce the storage space required for grammar and speed up the processing.

The second experiment in Section 4.6 shows that this method can distinguish shapes with small differences. Theoretically, any small difference which can be described by descriptors can be used for discrimination. But the practical results show that this may not be true due to the noise, resolution, and so on. We have tried to recognize an angle view, MIG-15,T, (see Figure 4.17) which is between MIG-15,V and MIG-15,U. Three MIG-15,T's were processed through the classification tree in Figure 4.16. They were all classified as MIG-15,V since they are very close to MIG-15,V. Then we tried to distinguish MIG-15,V's and MIG-15,T's. It did not turn out well. Some of the MIG-15,T shapes were misrecognized as MIG-15,V's, and vice versa. The 3 MIG-15,T's can be found in Appendix C. With the limitations of our resolution and boundary smoothing, if we reduce the size of the recognition region, namely tighten the threshold k's in the recognition functions, the noisy patterns will not be recognized. If we do not reduce the size of the recognition region, the recognition functions together with the production rules are not discriminative enough to distinguish all the testing patterns correctly. To improve the recognition performance, we need to improve digitization resolution, boundary approximation, and recognition functions.

Figure 4.17 MIG-15,T

CHAPTER 5

DISTORTED SHAPE RECOGNITION

5.1 Introduction

The syntactic shape recognition method using attributed grammars was proposed and discussed in Chapter 3. The implementation of this method on airplane shapes in Chapter 4 demonstrated that this method is an effective shape recognition method. But the shapes we have dealt with so far were reasonably clear in the sense that only a small amount of random noise was present. To show the advantage of our method over other existing methods, we will discuss in this chapter the generalized version of our method and the utilization of error-correcting techniques to recognize some heavily distorted shapes.

As mentioned previously, the shapes can be distorted by two different types of noise. One type of noise is random in nature and will be referred to as random noise. The other type of noise is not random and will be referred to as nonrandom distortion. The random noise can be filtered to a certain extent by some digital filtering techniques [22,90], while the nonrandom distortion is difficult to remove without knowing the classification of the shape. Normally, random noise is inevitable and nonrandom distortion may often occur. For example, a real airplane picture with a clear background has a small amount of ran-

dom noise, and a picture of an airplane in a cloudy background may have random noise and nonrandom distortion, when the clouds cover some portions of the airplane. In the real world, a shape is often distorted with both types of noise. The conventional filtering techniques or recognition methods have handled the random noise, but still cannot handle the nonrandom distortion well.

In Sections 5.2 and 5.3, the recognition functions and the PEE parsing schemes will be generalized to recognize noisy shapes. An error-correcting technique will be introduced to the parsing scheme in Section 5.4 to recognize heavily noisy or distorted shapes. Several versions of Earley's algorithm with different capabilities will be discussed in Section 5.5 to show the feasibility of these ideas. The Earley's algorithm for context-free grammar is selected for modification and discussion mainly because the context-free shape grammar can describe portions of the shape which are semantically meaningful by nonterminals with attributes.

Some distorted shapes will be processed by our error-correcting PEE Earley's parser to show its recognition capability. Then, a discussion concludes this chapter.

5.2 Generalized Recognition Functions

We have assumed that the recognition functions are 0-1 functions in Section 4.5. (See Assumptions 4.1 and 4.2). The result of such a recognition function is either "accept" or "reject". If the transformed descriptor of an unknown curve segment is within a certain neighborhood of the transformed descriptor of a referenced curve segment, the unknown curve segment is recognized as the referenced curve segment. Otherwise,

the recognition fails. The recognition function for angle primitives is similar. As a matter of fact, the boundary between "accept" and "reject" is not clear. Tsai and Fu [73] proposed a deformational model. In this model, every noise pattern is regarded as transformed from a pure pattern through syntactic deformation and semantic deformation. The syntactic and semantic deformations are symbolic error and attribute deviations respectively. This deformational model is applied to primitive extractions. Each deformation is assigned a probability and each class has a priori probability. A Bayes recognition rule is hence obtained based upon these assigned probabilities.

The probabilistic model is not practical in our case because it is difficult to collect a large enough number of samples to obtain a meaningful probability function for each referenced primitive or nonterminal. Besides, Tsai's deformational model cannot be applied to nonterminal recognition adequately.

For simplicity, we attempt to generalize our 0-1 recognition functions to take care of the so-called semantic deformation, or attribute deviation, and to use error-correcting techniques to solve the syntactic deformation. For a generalized recognition function, we use the idea of membership function from fuzzy set theory [74]. The membership function for a primitive maps from the domain constituted by the primitive and the unknown vector chain to the range of [0,1] on the real line.

Definition 5.1:

Let P and V be a primitive and a vector chain to recognize. The recognition function R maps from P x V to [0,1]. That is, $R(P,V) = r$, $r \in [0,1]$.

If r = 1, the V is recognized as P without a doubt. If r = 0, the recognition fails. If 0 < r < 1, the recognition is partially successful. From another point of view, r can be used to reflect the similarity between P and V. If the recognition is based on the transformed descriptors, we may define the recognition function as follows.

Definition 5.2:

$R(P,V) = R(T(P), T(V)) = r \in [0,1]$, where $T(\cdot)$ denotes the transformed descriptor.

Definition 5.1 can be extended to the case of a primitive string.

Definition 5.3:

Let $\alpha = P_1P_2\ldots P_k$ be a string of k primitives and $V = v_1v_2\ldots v_m$ be a vector chain to recognize. There are n feasible noisy boundary segmentations, or NBS's, each of which segments V into k vector subchains. ϕ_i be a set of vector subchains with respect to one of n NBS's. $\phi_i = \{s_{i1}, s_{i2}, \ldots, s_{ik}\}$ s_i's are vector subchains. The string recognition function R_s of α and ϕ_i is

$$R_s(\alpha,\phi_i) = R_s(P_1, s_{i1}, P_2, s_{i2}, \ldots, P_k, s_{ik})$$

$$= r_i \in [0,1]$$

And the recognition function of α and V is

$$R_s(\alpha,V) = \max_{1\le i\le n} R_s(\alpha,\phi_i) = \max_{1\le i\le n} r_i = r$$

If n = 0, then $R_s(\alpha,V) = r = 0$.

The computation of r_i from P's and s_i's is an interesting topic of study. If we consider that each P_j and its corresponding s_{ij} are relat-

ed by a fuzzy quantity, then the relation between $P_1P_2...P_k$ and $s_{i1}s_{i2}...s_{ik}$ can be considered as a string of concatenated fuzzy quantities. Lee and Zadeh [82] proposed fuzzy languages and fuzzy grammars. Also, they defined the operations of union, intersection, and concatenation of fuzzy languages. Of course, the concatenation of strings was also defined. Recently, Lakshmivarahan and Rajasethupathy [83] followed the same definitions in studying the fuzzification of formal languages and synthesis of fuzzy grammars. However, our attributed shape grammars are not fuzzy grammars. [82] and [83] fuzzify the grammars by giving each production rule a fuzzy membership, while our shape grammars do not have fuzzy membership for the production rules. Besides, the fuzzy quantity obtained through the concatenation of fuzzy quantities defined by [82] does not seem correct semantically for our shape recognition. According to the definition in [82], the membership of P_1P_2 equals the minimum of the membership of P_1 and P_2. But in our shape recognition, if P_1 is perfectly recognized and P_2 is poorly recognized, we would like to get a membership value somewhere between those of P_1 and P_2. A very intuitive idea is to take the weighted average of the membership of P_1 and P_2 as the membership of P_1P_2. This weighted average idea has been suggested by Bellman and Zadeh [89]. The weighting coefficients reflect the relative importance of the corresponding primitives or nonterminals.

The addition and multiplication of fuzzy quantities have been studied and published [80,81]. We may use fuzzy additions and fuzzy multiplications to obtain a good recognition function for a string of primitives. But, for simplicity in computation, we used the idea of weighted average and obtained semantically reasonable membership values. From

another point of view, we assume there is a recognition hierarchy, i.e., the membership value of a string of primitives can be computed based on the membership values of the primitives and the weights associated with the primitives. Taking advantage of the attributes associated with each primitive, we may use the ratio of the curve length of the primitive to the curve length of all the primitives in the string as the weight. Therefore, if P_1 of length ℓ_1 and P_2 of length ℓ_2 are recognized with membership values r_1 and r_2 respectively, the membership value of P_1P_2 can be calculated by $\frac{\ell_1}{\ell_1+\ell_2} r_1 + \frac{\ell_2}{\ell_1+\ell_2} r_2$. So, if P_2 is completely rejected, i.e., $r_2 = 0$, we still have a value of $\frac{\ell_1}{\ell_1+\ell_2} r_1$. In such a case, if P_1 is much longer than P_2, the major part of the total curve can still be recognized with a reasonable membership value. With this hierarchical assumption, the general form of the string recognition function can be written as follows.

Definition 5.4

$$R_s(\alpha,\phi_i) = R_s\{P_1,R(P_1,s_{i1}),P_2,R(P_2,s_{i2}) \ldots, P_k,R(P_k,S_{ik})\}$$

$$= R_s(P_1,r_{i1},P_2,r_{i2},\ldots,P_k,r_{ik}) = r_i \in [0,1]$$

where $r_{ij} = R(P_1,s_{ij})$, $1 \leq j \leq k$, and

$$R_s(\alpha,V) = \max_{1\leq i\leq n} R_s(\alpha,\phi_i) = \max_{1\leq i\leq n} r_i = r \in [0,1]$$

The Definition of 5.2 can easily be plugged into Definition 5.4 to obtain an expression based on transformed descriptors. In the following development, the hierarchical relationship is assumed.

As mentioned previously, the recognition of nonterminals plays an important role in our recognition system. The corresponding recognition functions need to be generalized too. In other words, we would like to have a generalized recognition function which can reflect the similiarity between a nonterminal and a curve segment. Since a nonterminal can be broken into primitives, the similarities of the primitives may affect the similiarity of the nonterminal. Therefore, the recognition function of a nonterminal depends upon the nonterminal and the vector chain as well as the primitives and their corresponding subchains.

$N \rightarrow \alpha_1 | \alpha_2$ indicates two productions $N \rightarrow \alpha_1$ and $N \rightarrow \alpha_2$. Each production rule produces a string of primitives and nonterminals. In the following definition of a nonterminal recognition function, we use R_s to recognize a string of primitives and nonterminals instead of only primitives. This extension is omitted here for simplicity.

Definition 5.5:

Let $N_i \rightarrow \alpha_{i1} | \alpha_{i2} | \ldots | \alpha_{i\ell}$, and $V = v_1 v_2 \ldots v_m$ be a vector chain to recognize. The nonterminal recognition function with respect to $N_i \rightarrow \alpha_{ip}$, $1 \leq p \leq \ell$, can be defined as

$$R(N_i^{\alpha_{ip}}, V) = R\{N_i, V, R_s(\alpha_{ip}, V)\}$$

The nonterminal recognition function for N will be

$$R(N_i, V) = \max_{1 \leq p \leq \ell} R(N_i^{\alpha_{ip}}, V) = r \in [0,1]$$

For our particular interest in transformed descriptors we may define the nonterminal recognition function as follows.

Definition 5.6:

Let $N_i \rightarrow \alpha_{i1}|\alpha_{i2}|\ldots|\alpha_{i\ell}$, and $V = v_1 v_2 \ldots v_m$. Then

$$R(N_i^{\alpha_{ip}}, V) = R\{T(N_i), T(V), R_s(\alpha_{ip}, V)\}$$

for $N_i \rightarrow \alpha_{ip}$, and

$$R(N_i, V) = \max_{1 \le p \le \ell} R(N_i^{\alpha_{ip}}, V) = r \in [0,1]$$

for nonterminal N.

In the above derivations, R denotes the recognition function for a primitive or a nonterminal, while R_s denotes the recognition function for a string of primitives and nonterminals. The recognition functions are defined such that, in the general case, it is more convenient to recognize the primitives before the nonterminals. We can see from Definition 5.5 that we need to recognize the strings, α's, (the right side of the production rules) in recognizing the nonterminal N. This situation favors bottom-up parsing. And, it can fit our PEE parsing very well. (See Figure 3.16.)

5.3 The PEE Parser Using Generalized Recognition Functions

In Section 3.4, we explained the concept of primitive-extraction-embedding. A PEE parser attempts to find a derivation tree and a corresponding feasible segmentation of the input vector chain. As illustrated by Example 3.6, there may be more than one feasible noisy boundary segmentation (NBS) corresponding to a reference segmentation. With 0-1 recognition functions, these feasible NBS's are equally recog-

nized or accepted by the grammar. With generalized recognition functions, the membership values are computed along with parsing until the starting symbol is reached. There may be many possible membership value between the noisy boundary chain and the shape grammar with respect to different NBS's and different reference segmentations. We could select the NBS which corresponds to the highest membership value as the best NBS and this membership value reflects the closest relationship between the vector chain and the shape grammar. Let us define the membership of a vector chain with respect to a shape grammar as follows. Let the input vector chain be $V = v_1 \ldots v_m$. $V_{1,m+1}$ denotes $v_1 v_2 \ldots v_m v_{m+1}$, with $v_{m+1} = v_1$. In the following discussion, $V_{i,j}$ denotes a vector subchain $v_i v_{i+1} \ldots v_j$.

Definition 5.7:

Let $G = (V, T, P, S)$ be a shape grammar and $V = v_1 v_2 \ldots v_m$ be an unknown vector chain. The membership of V with respect to G is $s = R(G, V_{1,m+1})$. If P has k S-production rules,

$$S \rightarrow \alpha_1 | \alpha_2 | \ldots | \alpha_k$$

then

$$R(G, V_{1,m+1}) = \max_{1 \le i \le k} R_s(\alpha_i, V_{1,m+1})$$

From the view point of attributed grammars, we may consider the membership value as an attribute associated with each primitive and nonterminal. With this idea, the general form of the attributed shape grammar in Section 3.4 can be modified as follows.

$G_\ell = (V_\ell, T_\ell, P_\ell, S_{\ell,r(S)})$

$V_\ell = \{S_{\ell,r(S)}, N_{r(N),D(N)}\text{'s}\}$

$T_\ell = \{F_{r(F),D(F)}\text{'s}, A_{r(A),D(A)}\text{'s}\}$

P_ℓ: $S_{\ell,r(S)} \rightarrow \alpha = (xA)^*XA\{Answer_{c,s}\}$;

$r(S) \leftarrow$ Recognition function of D(X), r(X), D(A), r(A) for X's and A's in α

$s \leftarrow r(S)$

$c \leftarrow \ell$

$N_{r(N),D(N)} \rightarrow \beta = (XA)^*X$

$D(N) \leftarrow (D(X) \underset{A}{\oplus})^*D(X)$

$r(N) \leftarrow$ Recognition function of D(N), D(X), r(X), D(A), r(A) for X's and A's in β

The r(X) and D(X) denote the membership value and the descriptor of X respectively. And, $X \in \{N\text{'s } F\text{'s}\}$.

To understand clearly how the membership functions are computed and how the production rules are selected in the generalized PEE parser, we have to consider the input vector chain and its subchains. During parsing, the recognition functions are performed and production rules are selected according to the following rules.

(a) *A S-production rule*

$S_{\ell,r(S)} \rightarrow \alpha = X_1A_2X_3\cdots X_{2n+1}A_{2n+2}$ $\{Answer_{c,s}\}$,

$x_k \in \{F\text{'s},N\text{'s}\}$, is selected and

$r(S) \leftarrow R_s(\alpha, V_{1,m+1})$

$s \leftarrow r(S)$

$c \leftarrow \ell$

if and only if the following conditions are satisfied.

(1) For any A_k in α, there is a corresponding subchain $V_{ik,jk}$, and $r(A_k) \leftarrow R(A_k, V_{ik,jk})$

(2) For any $X_k \in \{F's\}$ in α, there is a corresponding subchain $V_{ik,jk}$, and $r(X_k) \leftarrow R(X_k, V_{ik,jk})$

(3) $i1 = 1$, $jk = i(k+1)$, for $1 \leq k \leq 2n+1$ and $j(2n+2) = m+1$

(b) A N-production rule

$N_{r(N),D(N)} \rightarrow \beta = X_1 A_2 X_3 \cdots X_{2n+1}$, $X_k \in \{F's, N's\}$

is selected corresponding to a vector subchain $V_{i,j}$ and

$$D(N) \leftarrow D(X_1) \underset{D(A_2)}{\oplus} D(X_3) \cdots \underset{D(A_{2n})}{\oplus} D(X_{2n+1})$$

$r(N) \leftarrow R(N^{\beta}, V_{i,j})$

if and only if the following conditions are satisfied

(1) For any A_k in β, there is a corresponding subchain $V_{ik,jk}$, and $r(A_k) \leftarrow R(A_k, V_{ik,jk})$

(2) For any $X_k \in \{F's\}$ in β, there is a corresponding subchain $V_{ik,jk}$ and $r(X_k) \leftarrow R(X_k, V_{ik,jk})$

(3) $i1 = i$, $jk = i(k+1)$ for $1 \leq k \leq 2n$ and $j(2n+1) = j$

In both (a) and (b), the conditions (1) and (2) state how the angle primitives and curve primitives are extracted from the noisy boundary chain, and condition (3) guarantees that the vector subchains corresponding to the A's, F's, and N's in the production rules selected are feasible or consistent. Therefore, if a S-production rule is selected, then (1) there is a derivation tree which derives a syntactically correct symbolic pattern in the language by using the production rules, (2) the derivation tree corresponds to a reference segmentation of a noise-free shape, and (3) there exists at least one feasible noisy boundary segmentation of the vector chain corresponding to the reference segmentation. (Refer to Section 3.6 for the meaning of reference segmentation and noisy boundary segmentation.)

Since each recognition function outputs a membership value, it is easy to set a threshold to reduce the number of NBS's and hence to speed up the processing time. The NBS's denied by the thresholding will not be processed any further. But if we do not set a threshold, all possible NBS's will be processed even if some of them correspond to very small membership values. The idea and implementation of membership thresholding are obvious, so they are not discussed in detail here.

If the generalized recognition functions are used in our PEE parser, whenever a S-production is used, an answer consisting of c and s will be obtained. It indicates that the shape is recognized by the grammar labelled c with membership value s. If there is more than one answer, we can select an answer with the highest value of s which supposedly corresponds to the feasible NBS that fits a derivation tree the

best.

5.4 Error-Correcting Techniques and Generalized PEE Parsing

In the previous section, the parsing continues even if zero membership values are obtained. That is, if some parts of the shape are so badly distorted that they cannot be recognized by primitive or nonterminal recognition functions, the parsing will still continue, but with smaller membership values. In a sense, this idea is similar to the error-correcting technique discussed in [30,31,32]. The error-correcting parsing considers the noisy symbolic patterns containing three types of errors. They are substitution, deletion, and insertion errors. We will discuss these error types together with the special nature of shape boundaries, since shape is our major interest in this study.

If some parts of the shape are so distorted that they are misrecognized as wrong primitives, these wrong primitives can be considered as substitution errors by error-correcting techniques. In such a case, the error-correcting technique will measure the distortion by the number of substitution errors and the generalized PEE parsing will measure the distortion by membership functions. They both can handle this kind of distortion.

In fact, the two techniques attack the problem of noisy patterns by different approaches. The generalized PEE (GPEE) parsing can compute the degree of distortion for each primitive or nonterminal via the membership function. Since the membership functions deal with the boundary chain and the descriptors of the primitives and nonterminals directly, it is easier to obtain a membership value which reflects accurately the

similarity between the noisy shape and the shape it is supposed to be. The number of errors counted by the error-correcting (EC) parsing is not accurate enough to reflect their similarity. Recently, Lu and Fu [87] studied the stochastic error-correcting (SEC) technique for recognizing noisy symbolic patterns. Through the probabilities assigned to the production rules, the SEC parser can compute the likelihood function of a noisy pattern. This was an important advance in recognizing noisy symbolic patterns.

Example 5.1

Figure 5.1 shows the top view of a simple airplane distorted by a cloud on the right wing. The true shape can be found in Figure 3.6. The cloud on the right side creates an extra area attached to the airplane. The boundary of this extra area is very noisy. Although some filtering techniques may be used to smooth out the zig-zags of the boundary, the shape of this extra area is unpredictable because it changes with the cloud and the shape of a cloud is unpredictable. In such a case, the primitive extraction on the boundary of this extra area becomes a difficult problem. One way of solving this difficulty is to use an extra error primitive to represent the noisy curve segments which cannot be recognized as any legal primitive. And then, add the error production rules for this extra error primitive in all possible cases. But, the length of this extra error primitive is still undefined, since the boundary length of the extra area of the cloud is unpredictable. A primitive without any specification on its attributes will cause confusion in extractions. For instance, the noisy boundary of that extra area may be considered as one or several consecutive noisy curve seg-

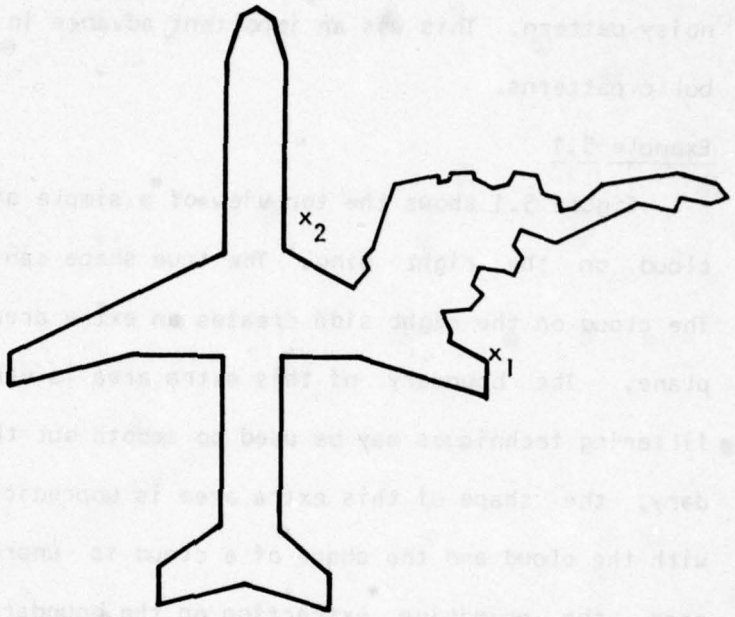


Figure 5.1 Top view of a Simple Airplane with a Distorted Right Wing

ments or error primitives. If the distorted boundary ($\widehat{X_1X_2}$) is extracted as one error primitive, the erroneous symbolic pattern can be accepted by an EC parser or SEC parser which allows only substitution errors. But there is no existing method which can systematically achieve such an extraction without using contextual information. If the distorted boundary is extracted as more than one error primitive, the erroneous symbolic pattern has to be accepted by an EC parser or SEC parser which allows substitution and insertion errors. If $\widehat{X_1X_2}$ is very long, as shown in Figure 5.1, due to distortion, it may be extracted to be k error primitives where k may be 3 or more. Then, the number of errors counted by an EC parser may change with k, and the probability computed by the SEC parser of a given grammar generating the erroneous symbolic pattern may change inversely with k. But, the length of $\widehat{X_1X_2}$ or the noisy condition of $\widehat{X_1X_2}$ should not change the probability or the error count, which may affect the classification. Since a human can recognize Figure 5.1 by the unaltered portion of the shape, the similarity between Figures 3.6 and 5.1 should closely approximate the percentage of the unaltered portion of the shape. The GPEE parser can segment the shape in Figure 5.1 at the same positions as the true shape in Figure 3.6, but the distorted portion, $\widehat{X_1X_2}$, will have a small membership value, and hence, the membership value of the whole distorted shape will also be smaller. If the distorted $\widehat{X_1X_2}$ has a zero membership, then the membership value of the whole shape will remain unchanged, no matter how badly $\widehat{X_1X_2}$ is distorted. In other words, the GPEE parser can fully use the contextual information to extract the primitives, so that a distorted shape like Figure 5.1 can be handled by only one substitution error.

Besides, the associated attribute information can be used to obtain a membership value which is close to human perception.

This example explains the advantages of GPEE parsing over the EC or SEC parsing in extracting primitives and reasonable membership values. But the error-correcting parsing is more powerful than the generalized PEE parsing in some cases, such as when the shape is distorted in such a way that some parts disappear. For instance, an airplane tail is missing because it is outside the visible field. The boundary is shorter and the number of vectors is smaller. There may not be a feasible segmentation of the vector chain corresponding to any possible derivation tree. In such a case, the generalized PEE parsing will stop, but the error-correcting parsing can recognize the bad shape by allowing deletion errors. In other words, if a whole primitive disappeared from the boundary, the GPEE parser could not handle such a deletion error.

However, these two techniques can be combined together to obtain a Generalized Error-Correcting PEE (GECPEE, for short) parsing which has all of the merits. As mentioned before, a membership threshold can be set to deny recognition of some primitives and nonterminals. Without losing any generality, we set the threshold to r_o, i.e., all the recognitions with membership less than r_o will be rejected. As discussed previously, the error-correcting technique can be used to take care of these rejections as well as the deletion errors.

The error-correcting technique with weighted error has been recently studied by Fu and Lu [84]. Since we have the attributes associated with each primitive and nonterminal, we can select proper weights for the errors. For instance, we can use the fourth attribute, the propor-

tional length, of a curve primitive as the error weight. If a primitive extraction is rejected by membership thresholding or a curve primitive is found deleted, the error-correcting technique is applied and the error weight indicates length of the distorted or deleted portion in proportion to the whole shape.

Therefore, any arbitrary vector chain can be processed by a GECPEE parser. Given an arbitrary shape grammar, the parser will terminate with two values, a membership value and an error weight. The error weight estimates the portions of the shape which are too badly distorted to recognize. The membership reflects the similarity between the other recognized portions of the shape and the corresponding portions of a shape generated by the grammar. There may be many sets of membership values and error weights. The GECPEE parser can always select the one with high membership and low error weight according to criteria set by the user. The GECPEE parser attempts to find a derivation tree which may have flaws and a segmentation on the unknown vector chain which fits the tree the best. Figure 5.2 shows a possible derivation tree obtained by the GECPEE parser. The X's indicate the flaws of the derivation tree. Namely, X's indicate where the recognitions are rejected or the primitives are deleted.

5.5 Modified Versions of Earley's Parser

Using a Valiant's parser [91], the time needed to recognize a string of length n generated by a context-free grammar is at most a constant times $n^{2.81}$. But, the grammar has to be in Chomsky normal form (CNF) and the constant is so large that the Valiant's parser is of only theoretical interest. (A CFG G = (N,T,P,S) is in CNF, if each produc-

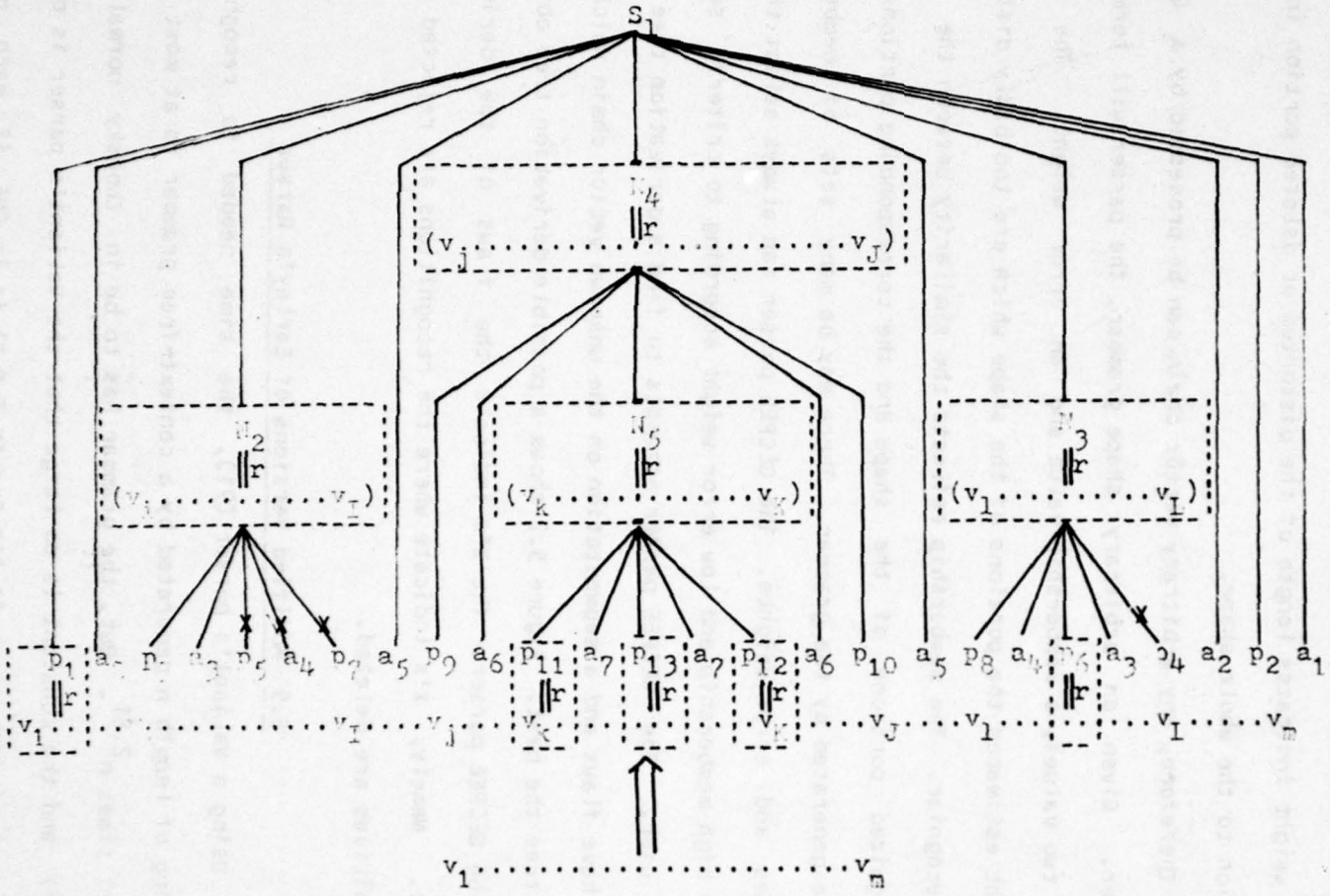


Figure 5.2 The GECPEE Parsing

tion in P is of one of the forms (1) $A \rightarrow BC$, $A,B,C, \in N$, (2) $A \rightarrow a$, $a \in T$, or (3) If $\lambda \in L(G)$ then $S \rightarrow \lambda$ is in P and S does not appear on the right side of any production [43,91].) The Cocke-Younger-Kasami parser and Earley's parser both result in a time bound proportional to n^3 [91]. The Cocke-Younger-Kasami parser also requires the context-free grammar in Chomsky normal form.

Earley's parsing algorithm has been the most practical and efficient algorithm for context-free grammars in general form. The modified first half of this algorithm was the PEE Earley's parser for our shape recognition in Section 3.6. Aho and Peterson introduced the error-correcting technique to Earley's algorithm in 1972 [30]. In this section, we will discuss several modified versions of Earley's algorithm for shape recognition. We will describe the GPEE Earley's algorithm, our version of error-correcting Earley's algorithm, and the combined GECPEE Earley's algorithm. The meaning of each item in the parse lists of different versions will be explained as well. Again, only the part of parsing table generation will be modified and discussed.

5.5.1 The Generalized PEE Earley's Algorithm

The algorithm in Section 3.6 is modified to use the generalized recognition function described in Section 5.2. In the following algorithms, the subscript r of a nonterminal or a primitive denotes that the corresponding recognition function is performed and the membership r is obtained. The flow-chart of the generalized PEE Earley's algorithm is shown in Figure 5.3. Each item in the parse lists, I's, has the same implication as that in Section 3.6.

Algorithm 5.1: The Generalized PEE Earley's Parser

Input: A shape grammar G, recognition functions R's, and an unknown chain of m vectors, $V_{1,m}$.

Output: Membership $s = R(G, V_{1,m+1})$.

Method:

(1) Add $[S \rightarrow \cdot\ \alpha, 1]$ to I_1 for all $S \rightarrow \alpha$ in P_ℓ

$j = 1$

(2) (a) If $[N \rightarrow \alpha \cdot B\beta, i]$ is in I_j and $B \rightarrow \gamma$ in P_ℓ

then add $[B \rightarrow \cdot\ \gamma, j]$ to I_j

(b) If $[N_r \rightarrow \alpha \cdot\ , i]$ is in I_j

then for all $[B \rightarrow \beta \cdot N\ \gamma, k]$ in I_i

add $[B \rightarrow \beta N_r \cdot\ \gamma, k]$ to I_j

(3) $j = j+1$

if $j > m+1$ goto (5)

For all $[N \rightarrow \alpha \cdot X\beta, i]$ in I_k, $1 \leq k \leq j$

$X \in \{F's, A's\}$

(a) If $\beta \neq \lambda$ then $r = R(X, V_{k,j})$

and add $[N \rightarrow \alpha X_r \cdot \beta, i]$ to I_j

(b) If $\beta = \lambda$, then $r1 = R(X, V_{k,j})$, $r2 = R(N^{\alpha X}, V_{i,j})$

and add $[N_{r2} \rightarrow \alpha X_{r1} \cdot\ , i]$ to I_j

(4) Go to (2)

(5) Find $[S_s \rightarrow \alpha \cdot\ , 1]$ in I_{m+1} with maximum s.

If none can be found, then $s = 0$

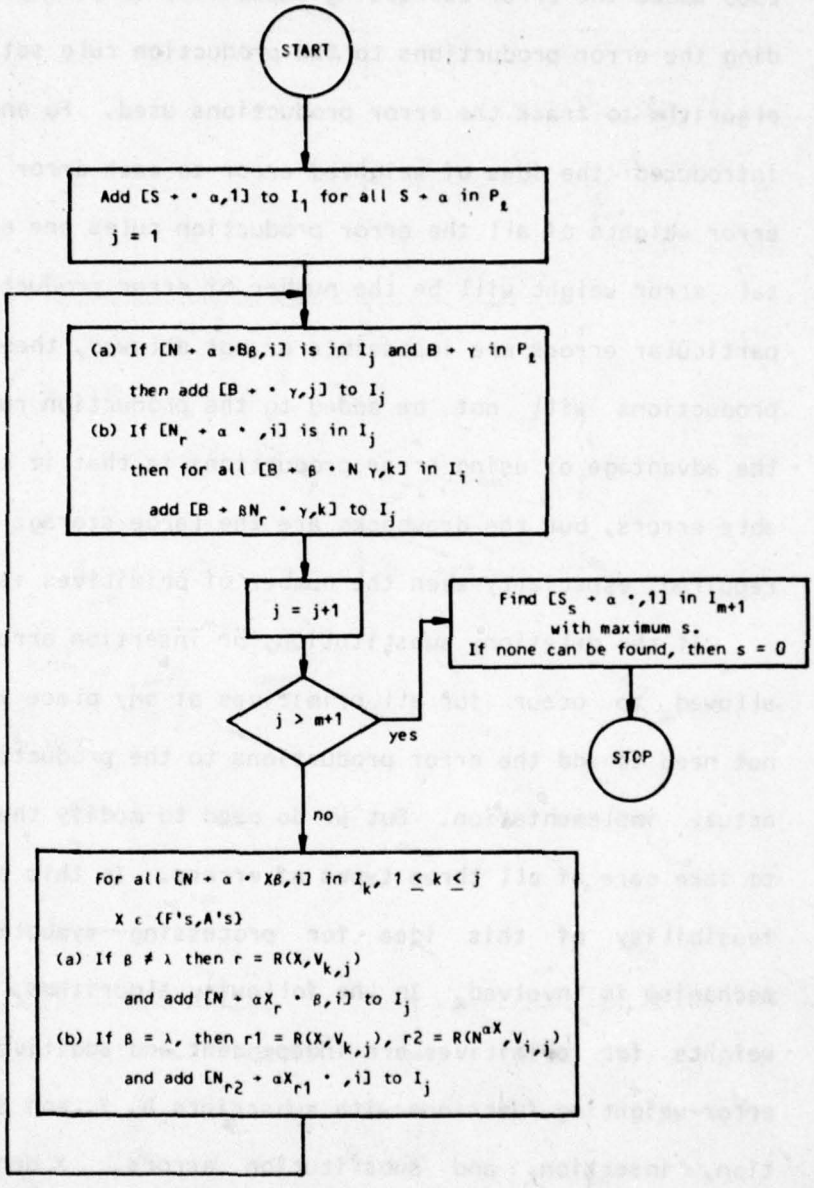


Figure 5.3 The Flow-Chart of Generalized PEE Earley's Algorithm

5.5.2 The Modified Error-Correcting Earley's Algorithm

In Section 5.4, we mentioned that the error-correcting technique can be applied to recognize badly distorted shapes. Aho and Peterson [30] added the error-correcting capability to Earley's algorithm by adding the error productions to the production rule set, and modifying the algorithm to track the error productions used. Fu and Lu [84] further introduced the idea of weighted error to each error production. If the error weights of all the error production rules are equal to 1, the total error weight will be the number of error productions used. If some particular errors are impossible or not allowed, the corresponding error productions will not be added to the production rule set. Therefore, the advantage of using error productions is that it controls the allowable errors, but the drawbacks are the large storage and processing time required, especially when the number of primitives is large.

If the deletion, substitution, or insertion errors are possible or allowed to occur for all primitives at any place in the string, we do not need to add the error productions to the production rule set in the actual implementation. But we do need to modify the Earley's algorithm to take care of all three types of errors. In this section, we show the feasibility of this idea for processing symbolic patterns. No PEE mechanism is involved. In the following algorithms, we assume the error weights for primitives are independent and additive. $\overline{W}(\cdot)$ denotes the error-weighting functions with subscripts D, I, and S indicating deletion, insertion, and substitution errors. X denotes the primitive. e.g., $\overline{W}_S(X,b_j)$ means the error weight of b_j substituting X. Step (2.c) in the following algorithm takes care of the deletion errors, while step

(3.b) takes care of the insertion and substitution errors. The corresponding flow-chart is in Figure 5.4.

An item $[A \rightarrow \alpha \cdot \beta, i, \omega]$ is in I_j for input string $b_1 \ldots b_n$, $0 \leq i \leq j \leq n$, if and only if

(1) $\alpha \overset{*}{\underset{\ell m}{\rightarrow}} a_{\ell+1} \cdots a_k$

(2) The difference between $a_{\ell+1} \cdots a_k$ and $b_{i+1} \cdots b_j$ is $\{a_{d1}, \ldots, a_{dD}, b_{i1}, \ldots, b_{iI}, (a_{s1}, b_{s1}), \ldots (a_{sS}, b_{sS})\}$, where

$a_{d1} \cdots a_{dD}$ are the primitive deleted,

$b_{i1} \cdots b_{iI}$ are the primitive inserted, and

$(a_{s1}, b_{s1}), \ldots (a_{sS}, b_{sS})$ indicate the substitution errors

(3) $$\omega = \sum_{x=1}^{D} \overline{W}_D(a_{dx}) + \sum_{x=1}^{I} \overline{W}_I(b_{ix}) + \sum_{x=1}^{S} \overline{W}_S(a_{sx}, b_{sx})$$

Algorithm 5.2: Modified Error-Correcting Earley's Parser

Input: An ordinary context-free grammar $G = (V, T, P, S)$, error-weighting functions $\overline{W}(\cdot)$'s and an unknown input string, $b_1 \ldots b_n$.

Output: Error-weight, ω.

Method:

(1) Add $[S \rightarrow \cdot\alpha, 0, 0]$ to I_0 for all $S \rightarrow \alpha$ in P

$j = 0$

(2) (a) If $[A \rightarrow \alpha \cdot B\beta, i, \omega]$ is in I_j and $\beta \rightarrow \gamma$ is in P

then add $[B \rightarrow \cdot\gamma, j, 0]$ to I_j

(b) If $[A \rightarrow \alpha \cdot, i, \omega_1]$ is in I_j,

then for all $[B \rightarrow \beta \cdot A\gamma, k, \omega_2]$ in I_i

add $[B \rightarrow \beta A \cdot \gamma, k, \omega_1+\omega_2]$ to I_j

(c) If $[A \rightarrow \alpha \cdot X\beta, i, \omega]$ is in I_j

and $X \neq b_{j+1}$, $j+1 \leq n$,

then add $[A \rightarrow \alpha X \cdot \beta, i, \omega+\overline{W}_D(X)]$ to I_j

(3) $j = j+1$, if $j > n$ go to (5)

For all $[A \rightarrow \alpha \cdot X\beta, i, \omega]$ in I_{j-1}

(a) If $X = b_j$ then add $[A \rightarrow \alpha X \cdot \beta, i, \omega]$ to I_j

(b) If $X \neq b_j$ then add $[A \rightarrow \alpha \cdot X\beta, i, \omega+\overline{W}_I(b_j)]$

and $[A \rightarrow \alpha X \cdot \beta, i, \omega+\overline{W}_S(X, b_j)]$ to I_j

(4) Go to (2)

(5) Find $[S \rightarrow \alpha \cdot, 0, \omega]$ in I_n with minimum ω.

In the above algorithm, all the possible errors are considered. If we know beforehand that a shape of total error weight higher than a threshold will not be accepted, we can threshold the error when an item is added to the item lists to eliminate a large number of possibilities. For instance, if we do not accept a shape of error higher than $\omega_o = 0.4$, then we only need to consider items of form $[A \rightarrow \alpha \cdot \beta, i, \omega]$, in which $\omega \leq \omega_o$. In such a case, step (5) may end up with no $[S \rightarrow \alpha \cdot, 0, \omega]$ in I_n for $\omega \leq \omega_o$. That means the input pattern is rejected by the grammar because no derivation with error smaller than ω_o can be found.

As mentioned in Section 5.4, we may use the proportional length of a primitive to the whole shape attribute as the error weight for each curve primitive in our shape attribute as the error weight for each curve primitive in our shape recognition. In this case, an item $[S \rightarrow \alpha \cdot, 0, \omega]$ in I_n implies ωx100% of the shape is badly distorted. The selection of threshold ω_o may affect the computational time and

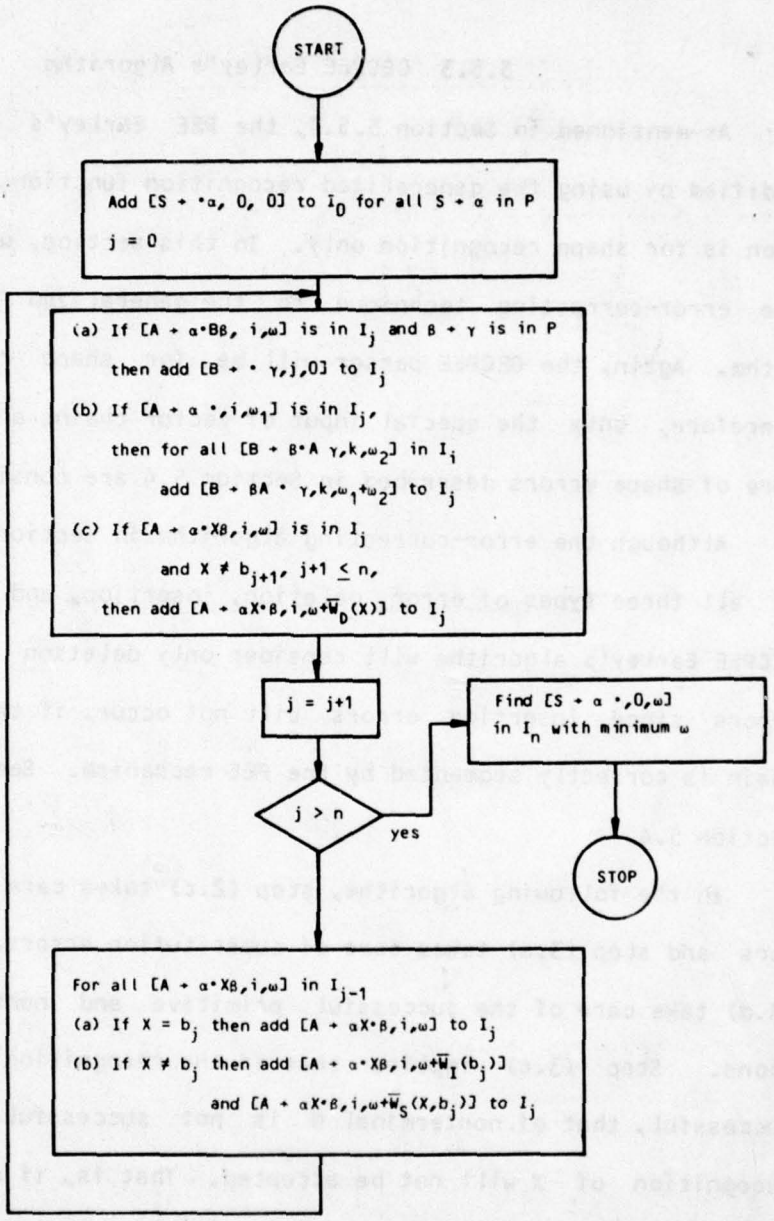


Figure 5.4 The Flow-Chart of Modified Error-Correcting Earley's Algorithm

recognition result. The smaller ω_o is, the shorter is the computation time. But a shape with distortion greater than ω_ox100% of what the shape is supposed to be will be rejected.

5.5.3 GECPEE Earley's Algorithm

As mentioned in Section 5.5.1, the PEE Earley's parser has been modified by using the generalized recognition functions. This modification is for shape recognition only. In this section, we will introduce the error-correcting technique to the generalized PEE Earley's algorithm. Again, the GECPEE parser will be for shape recognition only. Therefore, only the special input of vector chains and the special nature of shape errors described in Section 5.4 are considered.

Although the error-correcting algorithm in Section 5.5.2 takes care of all three types of error, deletion, insertion, and substitution, our GECPEE Earley's algorithm will consider only deletion and substitution errors since insertion errors will not occur, if the boundary vector chain is correctly segmented by the PEE mechanism. See Example 5.1 in Section 5.4.

In the following algorithm, step (2.c) takes care of deletion errors and step (3.a) takes care of substitution errors. Steps (3.b) and (3.d) take care of the successful primitive and nonterminal recognitions. Step (3.c) implies that if the recognition of primitive X is successful, that of nonterminal N is not successful and $\omega = 0$, the recognition of X will not be accepted. That is, if all the successors of N are successfully recognized, the recognition of N must succeed. The recognition of primitive or nonterminal is successful if and only if the membership value r is greater or equal to threshold r_o.

An item $[N \rightarrow \alpha \cdot \beta, i, \omega]$ in I_j for input vector chain $v_1 \ldots v_m$, $1 \leq i \leq j \leq m+1$, implies that

(1) iff $\alpha \neq \lambda$,

then $\alpha \overset{*}{\underset{\ell m}{\rightarrow}} a_\ell \ldots a_k \overset{\Gamma}{=} v_i \ldots v_j$

but $\{a_{d1} \ldots a_{dD}\}$ are deleted,

and $\{(a_{s1}, V_{i1,j1}), \ldots, (a_{sS}, V_{iS,jS})\}$ are substitution errors

and $\omega = \sum_{x=1}^{D} \overline{W}_D(a_{dx}) + \sum_{x=1}^{S} \overline{W}_S(a_{sx}, V_{ix,jx})$

(2) if $\alpha = \lambda$, then $i = j$

The flow-chart of this GECPEE Earley's algorithm is shown in Figure 5.5.

Algorithm 5.3: The GECPEE Earley's Algorithm

Input: A CFSG G, recognition function R's, membership threshold r_o, error-weighting functions $\overline{W}(\cdot)$'s, and an unknown chain of m vectors $V_{1,m}$.

Output: Membership $s = R(G, V_{1,m+1})$, and error-weight ω = summation of weights of deletion and substitution errors.

Method:

(1) Add $[S \rightarrow \cdot\alpha, 1, 0]$ to I_1 for all $S \rightarrow \alpha$ in P

$j = 1$

(2) (a) If $[N \rightarrow \alpha \cdot B\beta, i, \omega]$ is in I_j and $B \rightarrow \gamma$ in P

then add $[B \rightarrow \cdot\gamma, j, 0]$ to I_j

(b) If $[N_r \rightarrow \alpha \cdot, i, \omega_1]$ is in I_j

then for all $[B \rightarrow \beta \cdot N\gamma, k, \omega_2]$ in I_i

add $[B \rightarrow \beta N_r \cdot \gamma, k, \omega_1+\omega_2]$ to I_j

(c) If $[N \rightarrow \alpha \cdot X\beta, i, \omega]$ in I_j, $X \in \{F's, A's\}$

then add $[N \rightarrow \alpha X_0 \cdot \beta, i, \omega + \overline{W}_D(X)]$ to I_j

(3) $j = j+1$

if $j > m+1$ goto (5)

For all $[N \rightarrow \alpha \cdot X\beta, i, \omega]$ in I_k, $1 \leq k \leq j$, $X \in \{F's, A's\}$

(a) If $\beta = \lambda$, $r = R(X, V_{k,j}) < r_o$

then add $[N \rightarrow \alpha X_0 \cdot, i, \omega + \overline{W}_S(X, V_{k,j})]$ to I_j

(b) If $\beta = \lambda$, $r1 = R(X, V_{k,j}) \geq r_o$ and $r2 = R(N^{\alpha X}, V_{i,j}) \geq r_o$

(c) If $\beta = \lambda$, $\omega > 0$, $r1 = R(X, V_{k,j}) \geq r_o$

and $r2 = R(N^{\alpha X}, V_{i,j}) < r_o$

then add $[N_{r2} \rightarrow \alpha X_{r1} \cdot, i, \omega]$ to I_j

(d) If $\beta = \lambda$, and $r = R(X, V_{k,j}) \geq r_o$

then add $[N \rightarrow \alpha X_r \cdot \beta, i, \omega]$ to I_j

(e) If $\beta = \lambda$, and $r = R(X, V_{k,j}) < r_o$

then add $[N \rightarrow \alpha X_0 \cdot \beta, i, \omega + \overline{W}_S(X, V_{k,j})]$ to I_j

(4) go to (2)

(5) Find $[S_s \rightarrow \alpha \cdot, 1, \omega]$ in I_{m+1} with high s and low ω.

5.6 Experimental Results

In Section 5.5, several versions of Earley's algorithm were developed and discussed. We have implemented the GPEE Earley's algorithm and the GECPEE Earley's algorithm for our particular interests, distorted shape recognition. Like other existing error-correcting techniques, the above two algorithms both have the disadvantages of large storage and long computational time. In GPEE parsing we need generalized recognition functions for primitives and nonterminals. The 0-1

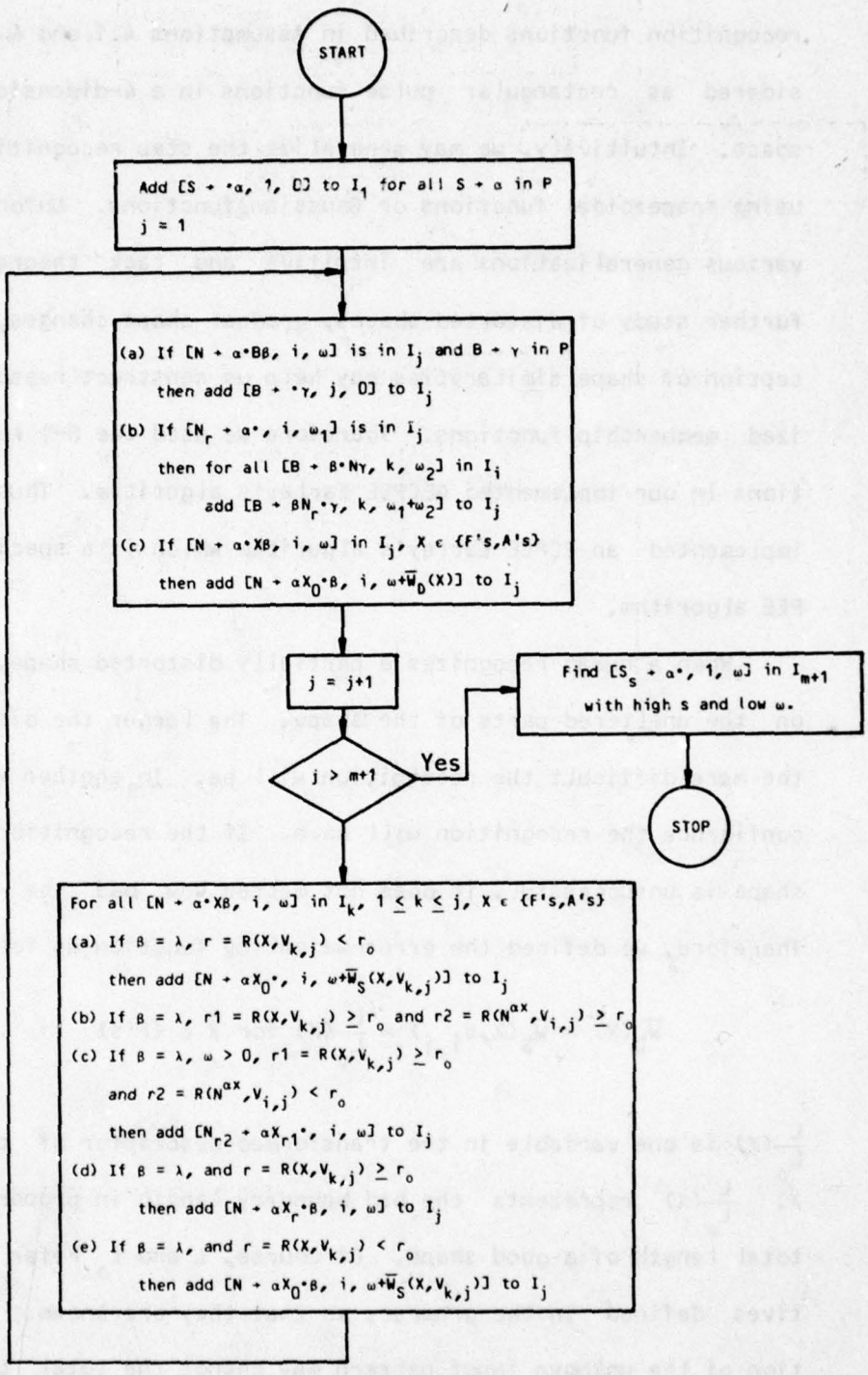


Figure 5.5 The Flow-Chart of GECPEE Earley's Algorithm

recognition functions described in Assumptions 4.1 and 4.2 can be considered as rectangular pulse functions in a 4-dimensional transformed space. Intuitively, we may generalize the step recognition functions by using trapezoidal functions or Gaussian functions. Unfortunately, these various generalizations are intuitive and lack theoretical support. Further study of distorted shapes, gradual shape changes, and human perception of shape similarities may help us construct reasonable generalized membership functions. Therefore we used the 0-1 recognition functions in our implemented GECPEE Earley's algorithm. Thus, we actually implemented an ECPEE Earley's algorithm which is a special case of GECPEE algorithm.

When a human recognizes a partially distorted shape, he/she relies on the unaltered parts of the shape. The larger the distorted part is, the more difficult the recognition will be. In another words, the less confidence the recognition will have. If the recognition of a part of a shape is unsuccessful, it does not matter how bad the distortion is. Therefore, we defined the error-weighting function as follows.

$$\overline{W}_D(X) = \overline{W}_S(X, V_{i,j}) = \frac{L}{L_o}(X) \text{ for } X \in \{F's\}$$

$\frac{L}{L_o}(X)$ is one variable in the transformed descriptor of curve primitive X. $\frac{L}{L_o}(X)$ represents the bad boundary length in proportion to L_o, the total length of a good shape. Of course, L and L_o refer to the primitives defined in the grammar, so that they are known. But the distortion of the unknown input pattern may change the total boundary length considerably and unpredictably. The total boundary is required by the Transformation T to perform the size normalization, as discussed in

Chapter 3. In our experiment, we assume that the total boundary length can be obtained by other means. This assumption is reasonable because the distance from the object to the camera can usually be estimated accurately. For instance, the distance between a flying airplane and an observer can be estimated by a radar. Distances under water can be estimated by a sonar. Once the distance of a given object is known, the total boundary length of a shape can be calculated using a proportional relation. Therefore, we assume that the total boundary length is known and use it as a size normalization factor.

Since ideally the angle primitive has no length, we defined $\overline{W}_D(A) = \overline{W}_S(A, V_{i,j}) = 0$. But, a curve primitive is recognized only if one of its two adjacent angle primitives shown in the production rule is recognized. So, the angle primitive recognition itself has no error weight, but it is used as a condition for adjacent curve primitives in our implementation. With these error-weighting functions, the final error weight ω of $[S_s \rightarrow \alpha^{\bullet}, 1, \omega]$ in I_{m+1} implies ωx100% of the shape is badly distorted.

The 0-1 membership functions we used for primitive and nonterminal recognition are the same as those described in Section 4.5. (See Assumptions 4.1 and 4.2.) The recognition functions are based upon transformed descriptors only. Therefore, the membership value is always 1 if the recognition succeeds. As mentioned in Section 5.5.2, we set an error threshold ω_o to reduce the number of items in parse lists and hence reduce the computation time.

In fact, we use two other strategies in the implementation to reduce the number of possibilities and computing time. One is deduced

from the error threshold ω_o. Since ω_o implies an error percentage greater than ω_ox100% will not be accepted, we may as well say the boundary recognized must be greater than $(1-\omega_o)$x100%. Therefore, when we are processing the i-th vector of chain $V_{1,m}$, the boundary recognized in $V_{1,i}$ plus the boundary to be processed in $V_{i,m+1}$ must be greater than $(1-\omega_o)$x100% of the whole shape. The other strategy uses the lookahead information to reduce the number of wrong segmentations when processing the distorted portion of the shape. That is, in Algorithm 5.3, step 3(e) says,

"For all $[N \rightarrow \alpha \cdot X\beta, i, \omega]$ in I_k, $1 \leq k \leq j$ $X \in \{F's, A's\}$,
if $\beta \neq \lambda$ and $r = R(X, V_{k,j}) < r_o$
then add $[N \rightarrow \alpha X_0 \cdot \beta, i, \omega + \overline{W}_S(X, V_{k,j})]$ to I_j."

We put in one more condition, i.e., if $\beta = Y\gamma$, $Y \in \{F's, A's\}$, $r = R(X, V_{k,j}) < r_o$ and $r1 = R(Y, V_{j,\ell}) \geq r_o$ for some $\ell \geq j$ then add $[N \rightarrow \alpha X_0 \cdot \beta, i, \omega + \overline{W}_S(X, V_{k,j})]$ to I_j. All these strategies make the following experiments possible under the limitations of memory space and computing time.

As a matter of fact, we implemented two versions of the ECPEE Earley's parser. Version 1 allows only substitution errors, i.e. step 2(c) of Algorithm 5.3 is taken away to not allow deletion errors. In this version, the latter one of the above two strategies is not implemented in the program. Version 2 allows both deletion and substitution errors. Both strategies are implemented in version 2.

Experiment 5.1

Figure 5.6 is a picture of a B52 in a cloudy sky. It is obtained by adding a picture of clouds to a picture of an arbitrary angle view of the B52 shown in Figure 4.10(a). Because the cloud on the right side is smoothly connected to and is as dark as the airplane, there is no way to delineate the boundary between the cloud and the airplane without knowing the model and viewing angle of the airplane first. Through the procedures of threshold finding, boundary following, and smoothing we obtained a shape shown in Figure 5.7. The vector chain in Figure 5.7 has 120 vectors.

Two versions of ECPEE Earley's parser were implemented on Purdue University's CDC 6500. Both versions can have no more than 8000 items in the parsing table. Each item requires 3 60-bit words. Thus, the whole parsing table occupies 24K words, which is about $60K_8$ words. The program itself occupies about $26K_8$ words. So, a total of about $106K_8$ 60-bit words is required to run on the CDC 6500.

The chain of 120 vectors (see Figure 5.7) was fed into both versions of the ECPEE Earley's parser with maximum error weight $\omega_o = 0.10$ and shape grammar $G_{B52,A}$ (see Experiment 4.1). Version 2 which allowed both substitution and deletion errors ran out of parsing table space and could not get an answer. Version 1 recognized Figure 5.7 as B52,A with error weight 0.09 in 140 seconds. The same vector chain was fed into version 1 with $\omega_o = 0.10$ and shape grammar $G_{F102,A}$ (see Experiment 4.1). The vector chain was rejected by $G_{F102,A}$ in 34 seconds. Then, we fed the vector chain to version 1 again with $\omega_o = 0.15$ and shape grammar $G_{B52,A}$. The parsing table ran out of space before an answer could be

Figure 5.6 A B52 in a Cloud Sky

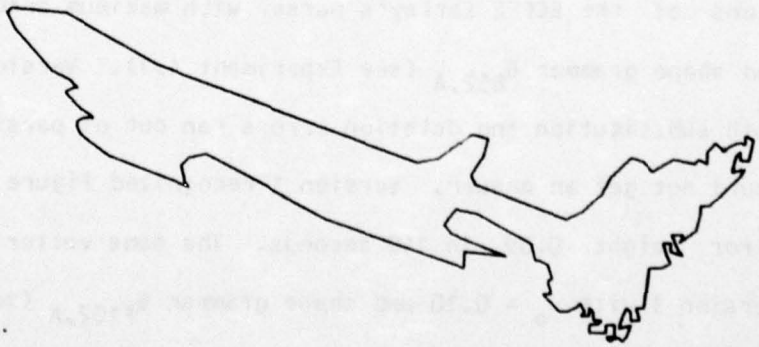


Figure 5.7 The Shape of a Distorted B52 After Preprocessing

obtained. Since the shapes of B52,A and F102,A are structurally different, the shape in Figure 5.7 can be accepted by $G_{F102,A}$ only if a small portion of the outer boundary is recognized as a one or two primitives of $G_{F102,A}$ and the rest of the shape is processed as distorted portion. Very possibly, Figure 5.7 can never be accepted by $G_{F102,A}$ because no primitive of $G_{F102,A}$ can be extracted from Figure 5.7.

This experiment demonstrated the recognition capability of our ECPEE parsing on a badly distorted shape such as that in Figure 5.7. Also, this experiment revealed that the number of items in the parsing table increased with ω_o.

Experiment 5.2

This experiment demonstrates the recognition capability on another type of distortion. When some parts of the shape are missing, we may need to allow both deletion and substitution errors. In this experiment, we will deal with some shapes of which some parts have disappeared. Therefore we need to use version 2 of the ECPEE Earley's parser.

Figures 5.8 and 5.10 are two F86,T's with a broken nose and a broken right wing respectively. After the preprocessing, Figures 5.9 having 65 vectors and 5.10 having 72 vectors are obtained. The starting points of the shapes are the leftmost points in the figures. The vector chains describing Figures 5.9 and 5.10 are fed into version 2 of ECPEE Earley's parser with ω_o = 0.15 and shape grammar $G_{F86,T}$ and $G_{MIG-15,U}$. (See Experiment 4.2.) Figures 5.9 and 5.11 are accepted by $G_{F86,T}$ with error weights 0.10 and 0.14 respectively and are both rejected by $G_{MIG-15,U}$. The processing times for $G_{F86,T}$ accepting Figure 5.9 (having

Figure 5.8 A F86,T Without Nose

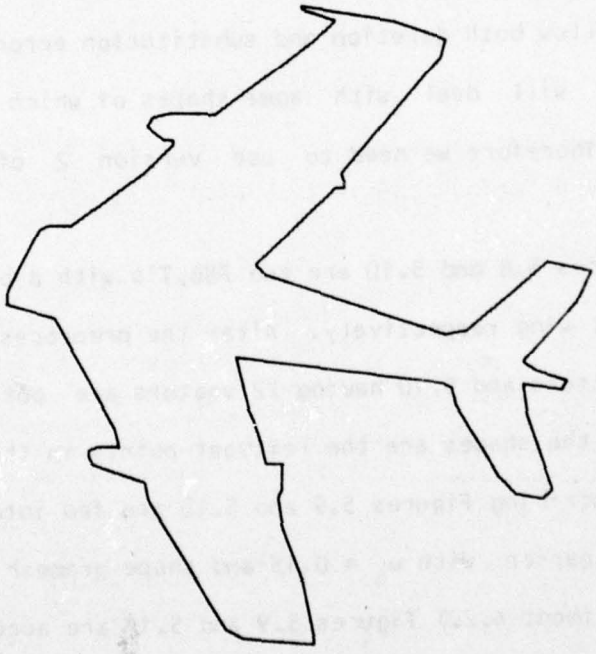


Figure 5.9 The Shape Obtained After Preprocessing Figure 5.8

Figure 5.10 A F86,T With a Broken Right Wing

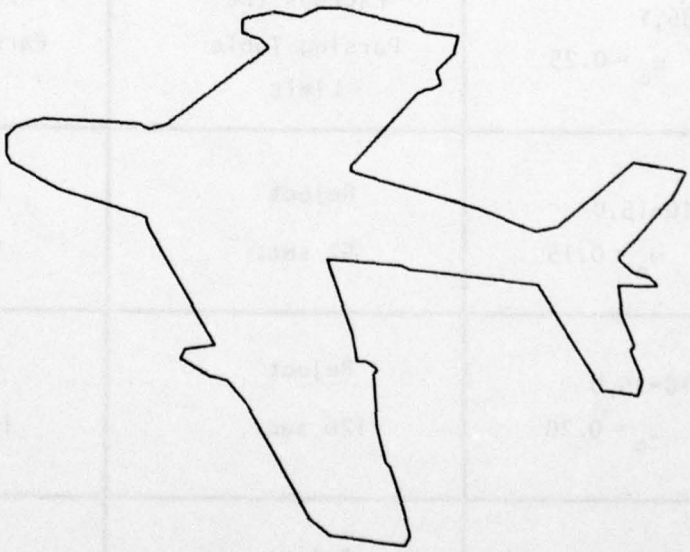


Figure 5.11 The Shape Obtained After Preprocessing Figure 5.10

Table 5.1 Experimental Results of Processing Distorted F86,T's with ECPEE Earley's Parser against $G_{F86,T}$ and $G_{MIG-15,U}$ with Different Error Threshold ω_o's

Error Weight and Processing Time / Shape / Grammar	Figure 5.9: F86,T with a Broken Nose (65 vectors)	Figure 5.11: F86,T with a Broken Wing (72 vectors)
$G_{F86,T}$ $\omega_o = 0.15$	$\omega = 0.10$ 143 sec.	$\omega = 0.14$ 117 sec.
$G_{F86,T}$ $\omega_o = 0.20$	$\omega = 0.10$ 375 sec.	$\omega = 0.14$ 370 sec.
$G_{F86,T}$ $\omega_o = 0.25$	Exceeds the Parsing Table Limit	Exceeds the Parsing Table Limit
$G_{MIG-15,U}$ $\omega_o = 0.15$	Reject 52 sec.	Reject 64 sec.
$G_{MIG-15,U}$ $\omega_o = 0.20$	Reject 126 sec.	Reject 165 sec.
$G_{MIG-15,U}$ $\omega_o = 0.25$	Reject 346 sec.	Reject 503 sec.

65 vectors) and Figure 5.11 (having 72 vectors) are 143 seconds and 117 seconds (CDC 6500) respectively. And the times for $G_{MIG-15,U}$ rejecting Figure 5.9 and Figure 5.11 are 52 seconds and 64 seconds respectively. We have tried the experiment with different error threshold ω_o's. The result can be found in Table 5.1. The experimental results show that the selection of ω_o affects the computation time, but does not affect the recognition results at all. Of course, a very small ω_o may reject some recognizable shapes and a very large ω_o may take a long computing time and a large memory for parsing table to recognize or reject a shape.

Experiment 5.3

We mentioned in Experiment 5.1 that the processing time and the number of items increase with ω_o. Figure 5.12 shows the rear half of a MIG-15,U. The preprocessed shape having 40 vectors can be found in Figure 5.13. The starting point, the leftmost point in the figure, is in the middle of the distorted region. It is processed by version 2 of the parser with ω_o = 0.5. It is rejected by $G_{F86,T}$ in 466 seconds, and accepted by $G_{MIG-15,U}$ with error weight 0.48 in 267 seconds. We can see that the processing time goes up with ω_o even though the number of vectors is less.

Figure 5.13 can still be recognized as MIG-15,U instead of F86,T although the distorted portion of the shape is large. Because, the important characters of the narrow width at the fuselage end and the straight edge of the rear ends of the wings are still shown in the shape for classification. Of course, the maximum error weight ω_o has to be reasonably set, so that the badly distorted shape can be recognized

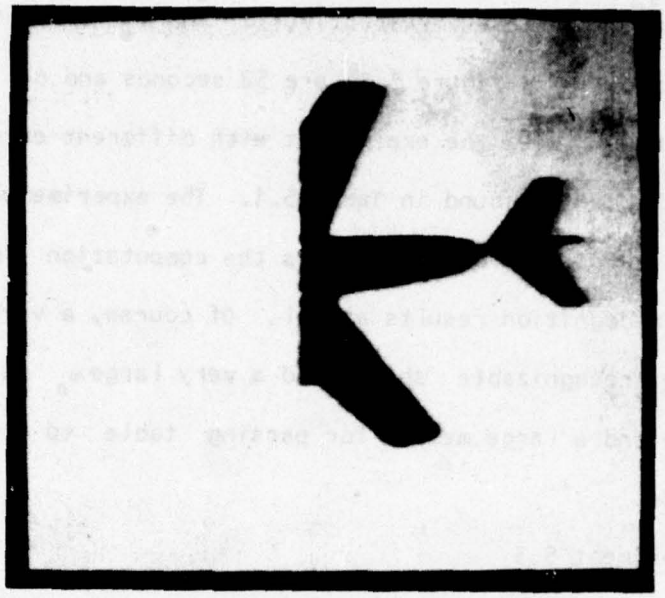


Figure 5.12 The Rear Half of an MIG-15,U

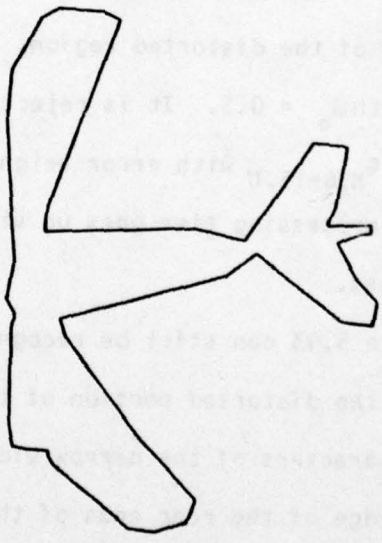


Figure 5.13 The Shape Obtained After Preprocessing Figure 5.12

within the limitation of time and memory.

5.7 Discussion

As mentioned before, none of the other existing methods can recognize shapes which are partially distorted. McKee and Aggarwal [76] developed an automatic system to learn and recognize entire shapes and partial shapes. By partial shapes, we mean only part of the shape is visible. McKee and Aggarwal utilized a library to store the edges, approximated and described by straight lines, of each shape and then, recognized the unknown shapes by matching the edges in the library. This matching method can recognize the partial shape only if the set of edges belonging to the object is known. As a matter of fact, this kind of partial shape can be considered a special case of distorted shapes.

A very straightforward extension of our distorted shape recognition involves identification of touching or overlapping objects. In biomedical images, touching objects, such as blood cells are very difficult to separate. If two touching objects are in different classes, the whole shape should be recognized by the two corresponding grammars with two error weights respectively. If both error weights are lower than a threshold, the touching assumption is correct. The same argument is also true for more than two touching objects, all of which are different. If two touching objects are in the same class, then we need to do further processing to make sure there are two objects touching each other. This further processing involves the second part of parsing, parse construction from the parsing table.

In constructing parse from the parsing table, we can find all the possible segmentations of the vector chain. If there are two segmenta-

tions Ψ_1 and Ψ_2, $\Psi_1 = \{s_{11},\ldots,s_{1n}\}$ and $\Psi_2 = \{s_{21},\ldots,s_{2n'}\}$, Ψ_{j1} and Ψ_{j0} are sets of subchains s_{ji}'s which are recognized and rejected, respectively. So, $\Psi_j = \Psi_{j0} \cup \Psi_{j1}$ and $\Psi_{j0} \cap \Psi_{j1} = \phi$ empty set for $j = 1,2$. If there are no overlapping subchains between Ψ_{11} and Ψ_{21}, Ψ_1 and Ψ_2 should be two objects of the same class. If the two error weights corresponding to Ψ_1 and Ψ_2 are both lower than the error threshold, the whole vector chain should contain portions of the boundaries where the two objects touch each other. This idea can be extended to more than two touching objects.

Another extension of distorted shape recognition is the reconstruction of distorted parts. This topic will be discussed in Chapter 7.

The GECPEE parsing is a time-consuming scheme because it has to consider many possible cases and finally obtain the segmentation that best fits a derivation tree. Two thresholds, membership threshold and error-weight threshold, can be employed to reduce the possibilities and computational time. However, this method can solve the distorted shape recognition problem in addition to all the other recognition problems which can be solved by other existing methods.

In 1975, Persoon and Fu [88] proposed a sequential classification algorithm (SCA) for stochastic context-free languages. The SCA is a modified Earley's algorithm. In 1977, Lu and Fu [87] further studied the SCA for noisy and distorted patterns. This algorithm is designed for symbolic patterns. Of course, it is possible to use the sequential classification idea to process a string of vectors which describe a shape. Besides, setting a stopping rule and a decision rule based on the membership values and error-weights obtained in processing the GEC-

PEE parser is very easy. But, as mentioned before, a shape may have different starting points due to the different scanning directions used in finding the boundary. If the starting point happens to be in the distorted portion, or the distorted portion is immediately adjacent to the starting point, then the sequential classification may produce false results. This situation can be improved by shifting the starting point or scanning along different directions, and finding the longest recognizable portion of the boundary chain. In addition, low-level language programming, such as assembly language programming [97], microprogramming [98], and parallel processing techniques [99,100] can be used to speed up the processing.

CHAPTER 6

GRAMMATICAL INFERENCE

6.1 Introduction

It is shown in Chapter 4 that the recognition capability of our proposed method is high, when proper shape grammars are provided. Our shape grammar can only describe 2-dimensional shapes, while actual 3-dimensional objects, e.g. airplanes, may have a number of different 2-dimensional views. Each view may require a different set of production rules to describe it. We can manually construct a shape grammar for some simple object which has only a few different views. But for complicated objects, it is much more convenient to have an algorithm which can automatically or interactively construct a grammar from the shapes extracted from digitized pictures.

Grammatical inference is an interesting research topic. Fu and Booth [78] surveyed various techniques of grammatical inference from symbolic sentences. If the primitives can be extracted correctly by machine without knowing the contextual information described by production rules, all of the patterns can be represented by symbolic sentences. The inference techniques described in [78] can then be used to obtain the grammar. For the shape recognition problem, we would like to infer grammars as well as primitives with attributes from the boundary

vector chains, since the production rules and primitives are flexible and dependent on the particular shapes. With human interference, interactive inference can be done easily. In the following sections, we describe an automatic and an interactive inference procedures which are basically designed for the shapes of a rigid body with a fixed appearance at a certain viewing angle.

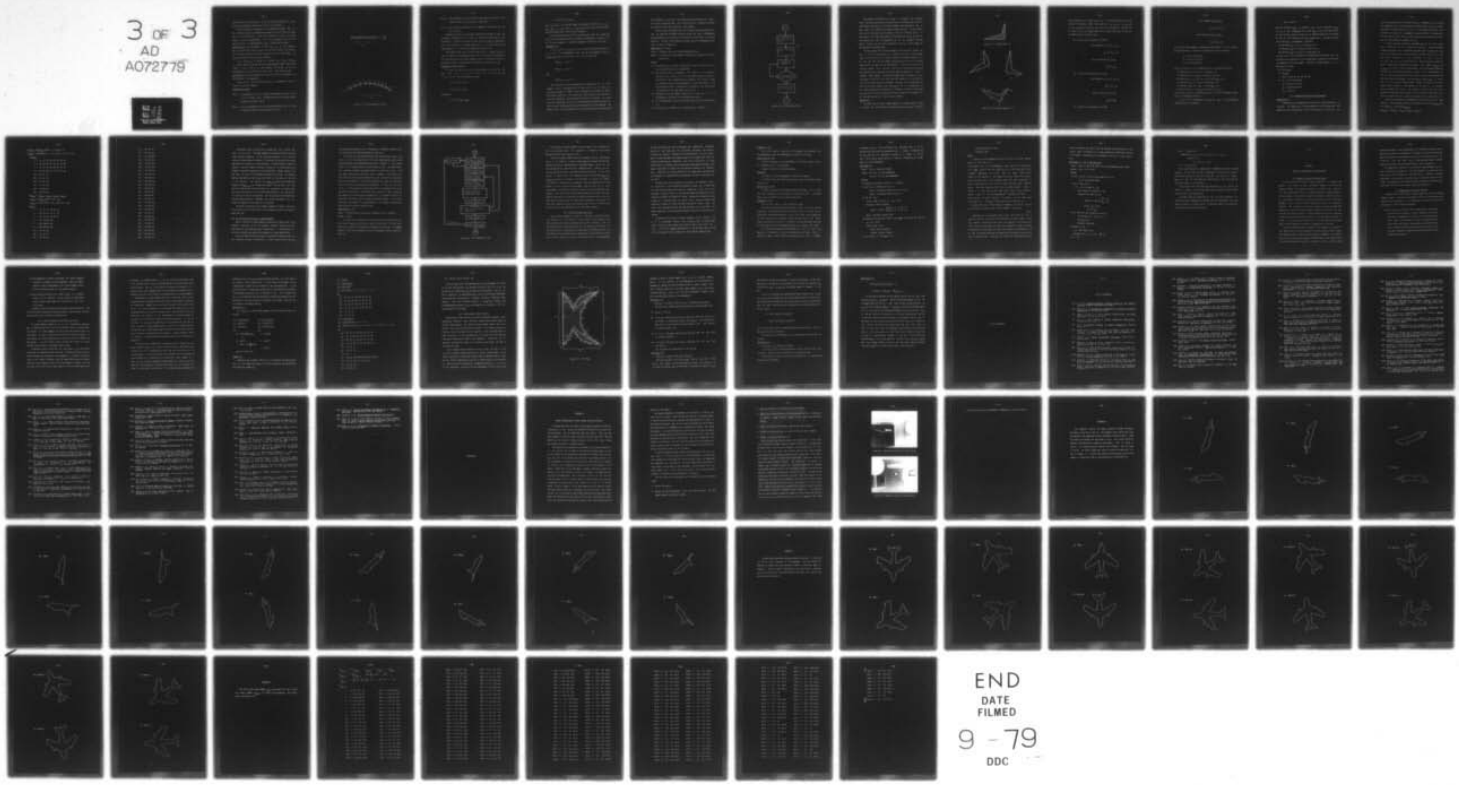
In Section 6.2, we will discuss an automatic learning algorithm, ALA, which infers shape grammars directly from the noisy shapes whose classifications are known. The ALA is actually implemented in FORTRAN and an experiment will be described in Section 6.3. But the shape grammars inferred automatically may not use the nonterminals to represent semantically meaningful and discriminative portions of the shape as properly as those inferred manually and interactively. Therefore, an interactive procedure is developed and discussed in Section 6.4. As described in Chapter 4, using finite-state shape grammars (FSSG) is more efficient in recognition time and is sometimes as accurate as using context-free shape grammars (CFSG). Thus, the conversion from CFSG to FSSG is sometimes worthwhile. This conversion will be discussed in Section 6.5.

6.2 The Automatic Inference of Shape Grammar

Since the shapes obtained from digitized images usually contain a certain amount of noise, the ALA tries to infer the shape grammars such that the attributes associated with the primitives and nonterminals are affected by the noise as little as possible. Based on the random characteristic of the noise, the ALA accepts more than one boundary chain for one view and uses the "averaging" technique to reduce the

AD-A072 779 PURDUE UNIV LAFAYETTE IN SCHOOL OF ELECTRICAL ENGINEERING F/G 9/2
SYNTACTIC SHAPE RECOGNITION USING ATTRIBUTED GRAMMARS.(U)
AUG 78 K C YOU, K S FU AFOSR-74-2661
UNCLASSIFIED TR-EE78-38 AFOSR-TR-78-1425 NL

3 OF 3
AD
A072779



END
DATE
FILMED
9-79
DDC

noise effect on the attributes. In the following paragraphs, the techniques and assumptions employed in ALA will be explained.

The ALA provides a systematic way of decomposing shapes. That is, the training shapes are decomposed systematically until each curve segment is a curve primitive. To do so, we need some criteria to decide whether a curve is decomposable. If it is, we decompose the curve into shorter curves. An indecomposable curve is designated as a NDC.

Definition 6.1: A curve segment, $C = \widehat{AB} = v_1 v_2 \ldots v_n$, is not decomposable, if the distance from any point on the curve to $\overline{AB}$ is less than E_h, or if the angle change at any point along the curve is less than E_a. E_h and E_a are arbitrary small positive numbers.

If $E_h = 0$ and $E_a = 0$, the NDC is a straight line. If $E_a = 0$ and E_h is a small number, then the NDC can be approximated by $\overline{AB}$. (See Figure 6.1(a)). If $E_h = 0$ and E_a is a small number, then the NDC can be approximated by a very smooth curve from A to B. (See Figure 6.1(b)).

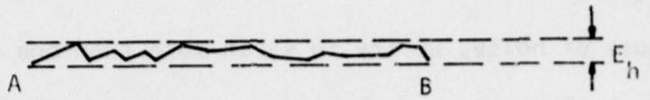
Definition 6.2: A curve segment which does not meet the requirements of definition 6.1 is decomposable.

There are four rules, according to which a decomposable curve is broken into shorter segments.

Decomposition Rules:

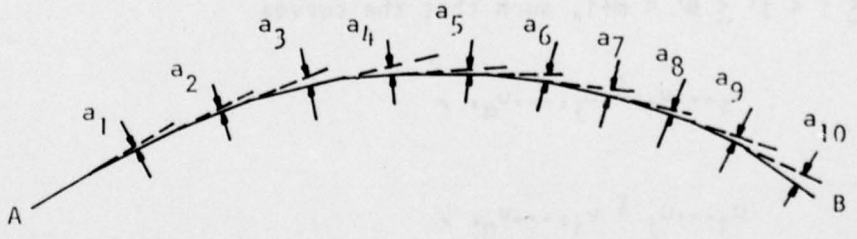
Rule 1: A decomposable curve can always be decomposed at a point which is the farthest away from $\overline{AB}$ among the points whose angular changes are greater than E_a.

Rule 2: For each decomposition, an associated production rule is constructed and added to the production rule set.



$E_a = 0$, $E_h > 0$

(a)



$E_h = 0$, a_i's $< E_a$

(b)

Figure 6.1 The Indecomposable Curves

Rule 3: The decomposable curves and NDC's obtained are defined as non-terminals and curve primitives, respectively.

Rule 4: Any connection between two curve segments is defined as an angle primitive.

Because of noise, we try to match more than one shape, so that the averaging techniques can be used to reduce the noise effect in the descriptors. Therefore, the ALA is input with more than one shape boundary with respect to the same angle view. If the input has only one shape boundary, it is indirectly assumed to be noise-free.

The boundaries may start from different points. A MATCH routine is used to search, in one boundary chain, for a point which corresponds to the starting point of the other boundary chain. The problem can be formulated as follows. The notation $\overset{r}{=}$ was defined in Definition 3.10.

Problem 6.1: Match Problem

$V = v_1 v_2 \ldots v_n$ and $U = u_1 \ldots u_m$ are two boundary vector chains. We must find i, i', n', j, j', m' with $1 \leq i < i' \leq n' < n+1$ and $1 \leq j < j' \leq m' < m+1$, such that the curves

$$v_1 \ldots v_i \overset{r}{=} u_{j'} \ldots u_{m'} ,$$

$$u_1 \ldots u_j \overset{r}{=} v_{i'} \ldots v_{n'} ,$$

and angles

$$v_i \ldots v_{i'} \overset{r}{=} u_{m'} \ldots u_{m+1} ,$$

$u_j \ldots u_{j'} \overset{r}{=} v_{n'} \ldots v_{n+1}$,

$u_{m+1} = u_1$, $v_{n+1} = v_1$, and the length of the angle primitives $v_i \ldots v_{i'}$, $u_{m'} \ldots u_{m+1}$, $u_j \ldots u_{j'}$ and $v_{n'} \ldots v_{n+1}$ must be less than a certain small positive number which is the corner tolerance.

Since the above curves may not be primitives, and the production rules are not constructed yet, we cannot use Assumption 4.1 directly for all the curve segments. A modified assumption is made for this case.

Assumption 6.1:

$V = v_x \ldots v_{x'} \overset{r}{=} U = u_y \ldots u_{y'}$, if (1) V and U are NDC's and $R(V,U) = 1$ or (2) V is decomposable, such that we can decompose V into $V = V_1 A_1 V_2$, and there are z, z' with $y \leq z < z' \leq y'$ such that

$$R(A_1, u_z \ldots u_{z'}) = 1$$

$$R(V_1, u_y \ldots u_z) = 1$$

and

$$R(V_2, u_{z'} \ldots u_{y'}) = 1$$

The R functions can be implemented as in Assumption 4.1 for curve segments and as in Assumption 4.2 for angle primitives. If V and U, or V_1 and V_2 are NDC's or simple curves, Assumption 6.1 performs a reasonably good recognition. If V_1 and V_2 are rather complex, they may be incorrectly recognized. This may lead to incorrect matching. To avoid incorrect matching, we can alter the thresholds by decreasing the k's. With stricter thresholds, some noisy boundaries may not be matched. The shapes which are not matched will be treated separately, because they

are considered so noisy that they need different primitives and production rules to describe them. This situation simply increases the number of production rules and primitives in the grammar.

There are many techniques that can be used to solve the match problem. We implement the MATCH routine by using "DO" loops in FORTRAN and using Assumption 6.1 to search for the match for every pair of training patterns. The ALA is described as follows, and the corresponding flowchart is shown in Figure 6.2.

Algorithm 6.1: The ALA

Input: n boundary chains as learning sample patterns.

Output: A context-free shape grammar consisting of production rules and descriptors.

Method:

(1) The MATCH routine is used to match $\frac{n}{2}$ (n-1) pairs of chains and all the corresponding points are memorized.

(2) Each boundary chain is decomposed with respect to the possible starting points found in (1). An associated S-production rule is constructed. The corresponding curve segments in different chains are defined using the same nonterminal or primitive symbols. (The associated descriptor is obtained by averaging all the descriptors corresponding to the same symbol at step (5)).

(3) Decompose each curve segment obtained in (2) and (3), if it is decomposable, according to the Decomposition Rules.

(4) If no decomposable curve can be found, then goto (5) otherwise goto (3).

(5) Calculate the attributes for all descriptors. Terminate.

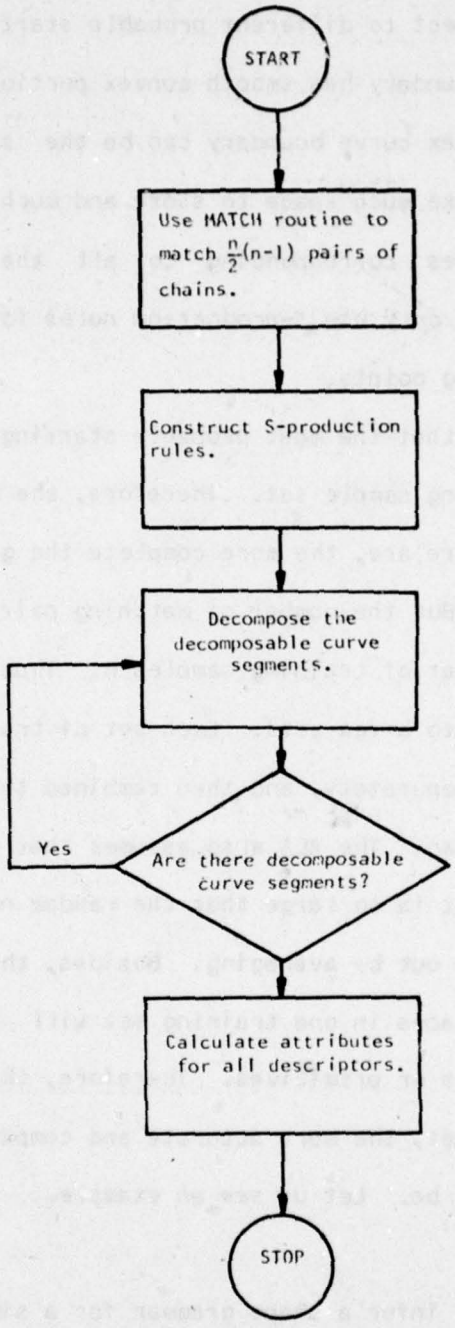


Figure 6.2 The Flow-Chart of ALA

As discussed in Section 3.4, in order to recognize the boundary chain starting from different convex points, the S-production rules are constructed with respect to different probable starting points. But in some cases, the boundary has smooth convex portions. In other words, any point on the convex curve boundary can be the starting point. In such cases, it may take much space to store and much time to process all the S-production rules corresponding to all the probable starting points. Actually, we only use S-production rules for a finite number of most probable starting points.

The ALA assumes that the most probable starting points should appear in the training sample set. Therefore, the more training sample inputs to the ALA there are, the more complete the grammar outputs will be from the ALA. But the number of matching pairs $n(n-1)/2$ increases rapidly with the number of training samples n. Thus, we may divide the training samples into a few sets. Each set of training patterns would be processed by ALA separately, and then combined together to obtain a complete shape grammar. The ALA also assumes that the number of training samples in one set is so large that the random noise on the descriptors can be cancelled out by averaging. Besides, the corresponding portions of different shapes in one training set will be designated with the same nonterminals or primitives. Therefore, the more training samples we have in one set, the more accurate and compact the grammar output from the ALA will be. Let us see an example.

Example 6.1:

We would like to infer a shape grammar for a simple shape "L" shown in Figure 6.3. With respect to different rotation angles, we obtain 3

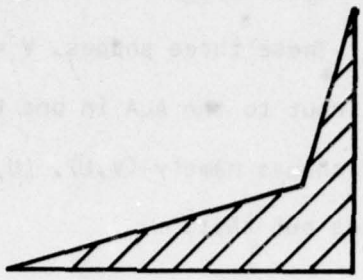
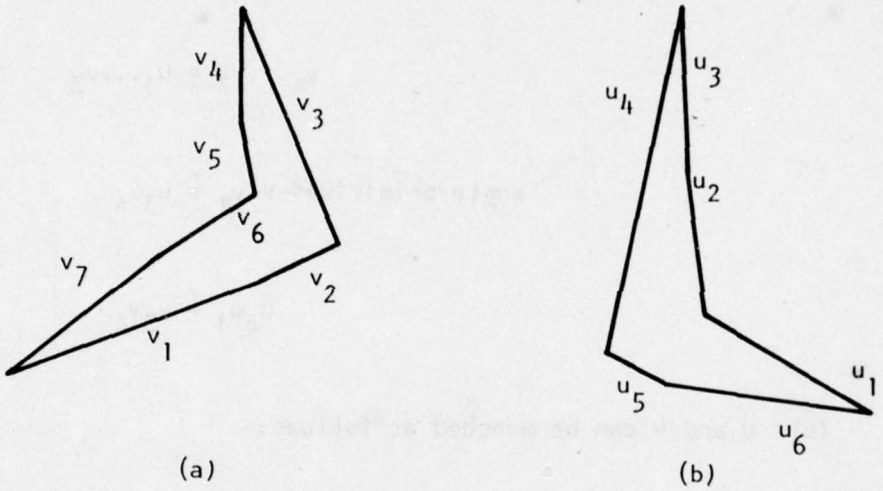


Figure 6.3 A Simple Shape "L"



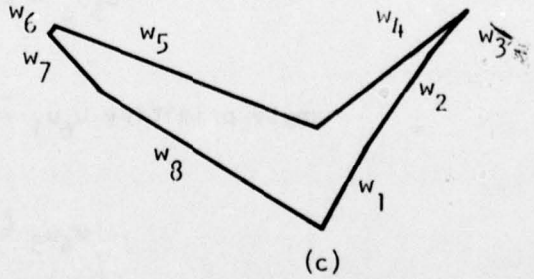


Figure 6.4 Three Noisy Shapes of "L"

noisy shapes shown in Figure 6.4(a)-(c). The starting points are at the bottom of the shapes. These three shapes, $V = v_1 \ldots v_7$, $U = u_1 \ldots u_6$, and $W = w_1 \ldots w_8$ serve as input to the ALA in one training set. The ALA has to match 3 pairs of shapes namely (V,U), (U,W), and (V,W) in step (1). The MATCH routine finds out that:

(a) V and U can be matched as follows:

curve segments $v_1 \ldots v_3 \overset{r}{=} u_4 \ldots u_6$

$v_4 \ldots v_7 \overset{r}{=} u_1 \ldots u_3$

angle primitives $v_7 v_1 \overset{r}{=} u_3 u_4$

$u_6 u_1 \overset{r}{=} v_3 v_4$

(b) U and W can be matched as follows:

curve segments $u_1 \ldots u_4 \overset{r}{=} w_4 \ldots w_8$

$u_5 u_6 \overset{r}{=} w_1 \ldots w_3$

angle primitive $u_6 u_1 \overset{r}{=} w_3 w_4$

$u_4 u_5 \overset{r}{=} w_8 w_1$

(c) V and W can be matched as follows:

curve segments $v_1v_2 \overset{r}{=} w_7w_8$

$v_3 \ldots v_7 \overset{r}{=} w_1 \ldots w_5$

angle primitives $v_2v_3 \overset{r}{=} w_8w_1$

$v_7v_1 \overset{r}{=} w_5 \ldots w_7$

In step (2), each boundary is decomposed with respect to the starting points found above. So we have 3 S-production rules as follows:

$S_L \rightarrow$ F1 A1 F2 A2 N1 A3

$S_L \rightarrow$ N1 A3 F1 A1 F2 A2

$S_L \rightarrow$ F2 A2 N1 A3 F1 A1

where F's and N's are curve segments and A's are angle primitives.

F1 indicates v_1v_2 in V, u_4 in U and w_7w_8 in W.

F2 indicates v_3 in V, u_5u_6 in U and $w_1 \ldots w_3$ in W.

N1 indicates $v_4 \ldots v_7$ in V, $u_1 \ldots u_3$ in U and w_4w_5 in W.

A1 indicates v_2v_3 in V, u_4u_5 in U and w_8w_1 in W.

A2 indicates v_3v_4 in V, u_6u_1 in U and w_3w_4 in W.

A3 indicates v_7v_1 in V, u_3u_4 in U and $w_5 \ldots w_7$ in W.

In step (3), the N1 in each boundary chain is further decomposed according to the Decomposition Rule.

$v_4 \ldots v_7$ is first decomposed into v_4v_5 and v_6v_7. A corresponding production rule is obtained.

$N1 \rightarrow F3\ A4\ F4$

where F3 indicates v_4v_5, F4 indicates v_6v_7 and A4 indicates v_5v_6. $u_1 \ldots u_3$ is then decomposed into u_1 and u_2u_3. Since their descriptors are very close to those of F3 and F4, the same N1 production is used and F3 and F4 also indicate u_1 and u_2u_3 respectively. The same situation occurs when w_4w_5 is decomposed. Therefore,

F3 indicates v_4v_5 in V, u_1 in U and w_4 in W.

F4 indicates v_6v_7 in V, u_2u_3 in U and w_5 in W.

A4 indicates v_5b_6 in V, u_1u_2 in U and w_4w_5 in W.

In step (5), all the attributes of the descriptors associated with the primitives and nonterminals can be computed by averaging the descriptors corresponding to the same symbol. Therefore, a shape grammar of L is obtained as follows:

$G_L = \{V, T, P, S_L\}$

$V = \{S_L, N1\}$

$T = \{F1, F2, F3, F4, A1, A2, A3, A4\}$

P: $S_L \rightarrow F1\ A1\ F2\ A2\ N1\ A3$

$S_L \rightarrow N1\ A3\ F1\ A1\ F2\ A2$

$S_L \rightarrow F2\ A2\ N1\ A3\ F1\ A1$

$N_1 \rightarrow F3\ A4\ F4$

6.3 Experimental Results and Discussion

Experiment 6.1:

To demonstrate the proposed ALA, two views of F102,A and B52,A are selected. The F102,A has 4 training patterns in one training set. The B52,A has 6 training patterns in two training sets, 3 in each set. All

ten training patterns can be found in group L in Appendix B. The reason for separating the 6 training patterns of B52,A into 2 sets is to reduce the number of matching pairs from 15 to 6. The grammars resulting from different training set arrangements may be different in compactness. The more patterns there are in one set, and the less training sets there are, the more compact the grammar will be.

To avoid incorrect matching by the MATCH routine, we used rather strict thresholds for the recognition function. The corresponding recognition region in the transformed attribute space is about the same size as the distribution of the descriptors studied in Section 3.5. We set corner tolerance $k_c = 6$ and thresholds $k_1 = 0.15$, $k_2 = 0.1$, $k_3 = 0.2$, and $k_4 = 0.1$ (see Section 4.4). These thresholds caused two pairs of B52,A training shapes to not match. The output $GL_{B52,A}$ and $GL_{F102,A}$ are shown in the following pages. The corresponding segmentation will not be shown here, because it takes a lot of programming effort to add the plotting facilities to ALA to track each decomposition and each primitive assignment. We used $GL_{B52,A}$ and $GL_{F102,A}$ to recognize the 10 training samples shown in group L and another 10 testing samples shown in group T. (See Appendix B.) All 40 tests, 20 patterns for each grammar, are correct. The recognition functions we used in the recognition experiment are the same as those we used in Experiment 4.1. Also the recognition time is about the same as that in Experiment 4.1.

The ALA was implemented in FORTRAN on Purdue University's CDC 6500 computer. The processing time was 3.9 seconds for inferring $GL_{B52,A}$ and 3.4 seconds for inferring $GL_{F102,A}$.

$GL_{F102,A} = (V_{F102,A}, T_{F102,A}, P_{F102,A}, S_{F102,A})$

$V_{F102,A} = \{S_{F102,A}, NI's\}, \quad I = 1,2,...,3$

$T_{F102,A} = \{FJ's, AK's\}, \quad J = 1,...,14, \quad K = 1,...,13$

$P_{F102,A}$:

S → F1 A1 F2 A2 N1 A3 N2 A4

S → N2 A4 N3 A1 F2 A2 N1 A3

S → F2 A2 N1 A3 N2 A4 F1 A1

S → N1 A3 N2 A4 F1 A1 F2 A2

N1 → F3 A5 F4

N2 → F5 A6 F6

N3 → F7 A8 F8

N1 → F9 A9 F10

N2 → F11 A11 F12

N2 → F13 A13 F14

$GL_{B52,A} = (V_{B52,A}, T_{B52,A}, P_{B52,A}, S_{B52,A})$

$V_{B52,A} = \{S_{B52,A}, NI'S\}, \quad I=1,...,25$

$T_{B52,A} = \{FJ'S, AK'S\}, \quad J=1,...,39, \quad K=1,...,39$

$P_{B52,A}$:

S → N1 A1 F1 A2 N2 A3

S → N2 A3 N1 A1 F1 A2

S → F1 A2 N2 A3 N1 A1

S → N3 A15 N4 A16

S → N4 A16 N3 A15

S → N5 A21 N6 A22

N1 → F2 A4 N7

N2 → F3 A5 F4

N2 → F5 A6 F6

N1 → N8 A7 F7

N1 → F8 A9 N9

N7 → F9 A10 N10

N8 → F10 A11 F11

N9 → F12 A12 N11

N10 → F13 A13 F14

N11 → F15 A14 F16

N3 → N12 A17 F17

N4 → F18 A18 F19

N3 → N13 A20 F20

N12 → N14 A23 N15

N13 → N16 A24 N17

N5 → N18 A25 N19

N6 → N20 A26 N2

N14 → N21 A27 F21

N15 → F22 A28 F23

N16 → F24 A29 F25

N17 → N22 A30 N23

N18 → F26 A31 F27

N19 → F28 A32 N24

N20 → F29 A33 F30

N21 → F31 A35 F32

N22 → F33 A36 F34

N23 → F35 A37 N25

N24 → F36 A38 F37

N25 → F38 A39 F39

The design of ALA in Section 6.2 is based upon the training patterns of one view. The shape grammar is constructed without knowing other classes of patterns. If the classes of patterns to be discriminated are significantly different in structure, the ALA may infer grammars which are effective in discrimination. If the different classes of patterns are only slightly different in detail or in certain parts, the grammars inferred by ALA may not be discriminative enough because the ALA does not know the differences between the classes. For example, MIG-15,U and MIG-15,V (see Figure 4.13) differ in the width at the end of the fuselage. The ALA may infer $GL_{MIG-15,V}$ based on the MIG-15,V shapes, and $GL_{MIG-15,U}$ on the MIG-15,U shapes. The ALA does not know where the significant difference should be. Therefore, its output grammars may not be effective in discriminating MIG-15,U's and MIG-15,V's. Of course, the ALA can be improved. But the present ALA implementation requires 1200 statements in FORTRAN. Any further improvement will increase the complexity and length of the program.

To obtain discriminative and compact grammars for shape classification without too much human effort, we developed an Interactive Learning Algorithm, ILA.

6.4 The Interactive Inference of Shape Grammars

Manual inference of shape grammars requires more human effort than automatic inference, while the grammars obtained manually are more discriminative than those obtained automatically. A man-machine interactive procedure may improve performance yet reduce human effort.

There are many different versions of interactive procedures due to the differing extents of interaction. In this section we will discuss

one developed procedure which is implemented in FORTRAN at Purdue's Pattern Processing and Advanced Automation Laboratory.

The interactive learning algorithm, ILA, processes one model pattern at a time. The algorithm is first fed a shape pattern which may be obtained through the preprocessing described in Chapter 4. The original source of this pattern may be a noisy analog picture. The human operator is allowed to modify this model pattern to reduce the noise. Then the human operator may use ALA to infer a grammar or interactively assign the curve primitives and production rules. We used the interactive way in our experiments. The ILA will automatically check the consistencies or conflicts of the primitive assignments. If there are any conflicts or inconsistencies, these wrong primitive assignments need to be corrected interactively. The ILA will also assign the angle primitive and compute all the descriptors automatically. The grammar inferred interactively can come close to a manually designed grammar. If the classification performance is unsatisfactory, the grammar can always be modified again through a feed-back loop.

Algorithm 6.2: ILA

Input: Training patterns and operator commands from the keyboard.

Output: A CFSG

For simplicity, the details of this algorithm are omitted here, but the block diagram of this ILA is shown in Figure 6.5. The (I)'s mark the blocks in which the interactive techniques are employed. The RAMTEK display is utilized to show the primitive assignments, pattern modification, etc.

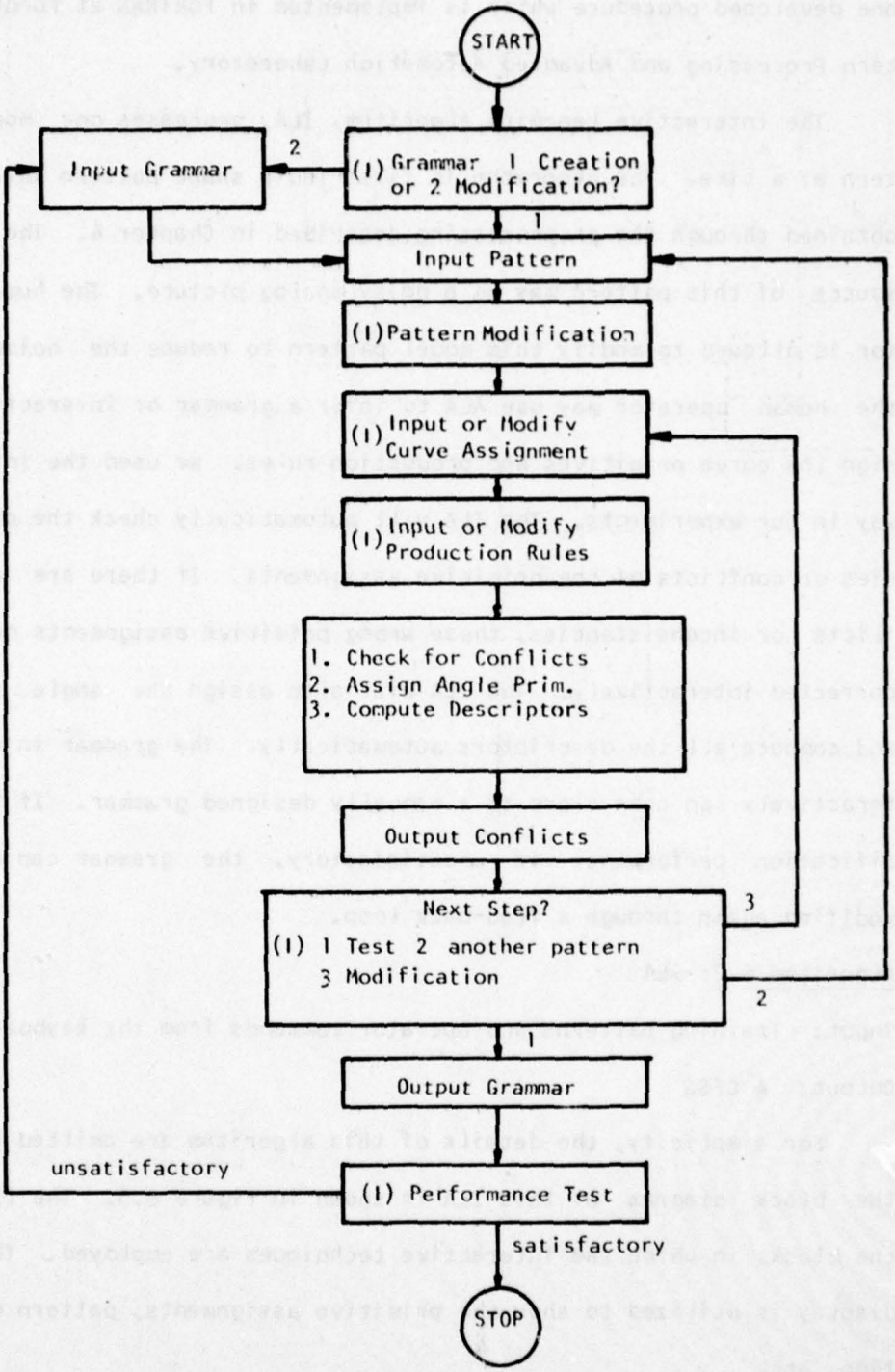


Figure 6.5 The Flow-Chart of ILA

The context-free shape grammars used in Chapter 4 are produced by the ILA discussed here. The experiments in Chapter 4 have already demonstrated the effectiveness of ILA.

Since ILA requires human efforts and feedback from the performance test, it is difficult to make a good comparison between ILA and ALA. Besides, the ALA was implemented on a CDC 6500 computer while the ILA was implemented on a PDP 11/45 with a RAMTEK display. However, we still can get some idea from our experiences using ILA and ALA. As we mentioned in Section 3.4, the computing time for inferring $GL_{B52,A}$ or $GL_{F102,A}$ was about 3.7 seconds on the CDC 6500. Of course, the computing time will increase with the number of training samples and with the number of vectors in each training sample. When we used ILA to infer the grammars on the PDP 11/45, about 2 hours of time was needed for each grammar. This 2 hour time includes human observation, design and man-machine interaction. Of course, the central processing unit of the computer was idle and waiting for input from the keyboard during most of this time. Needless to say, the inferring time of ILA is related to human skill and to the smoothness of the input model patterns.

6.5 The CFSG to FSSG Conversion

The attributed grammars are used for shape description and recognition because (1) the production rules can describe the structure, (2) the primitives describe the boundary details with attributes, and (3) the nonterminals can describe portions of the shape with attributes. If the primitives of a shape pattern are defined, the boundary details are described by the attributes and the production rules describe the structure in either context-free form or finite-state form. The nonterminals

in CFSG can properly be used to represent some semantically meaningful portion of the shape and the associated attributes can usefully represent the semantic information for discrimination. But the nonterminal in FSSG represents the remaining part of the shape after some consecutive primitives are recognized. There is no flexibility of using nonterminals to represent some perceptionally meaningful portions of the shape. Experiment 4.2 shows the discriminative capability of nonterminals. Therefore, if certain portions of the shape need to be described by attributes or represented by nonterminals, the CFSG is preferred over the FSSG.

In some cases, the differences in structure and boundary details are sufficient in discriminating different classes, e.g. in experiment 4.1, both CFSG and FSSG can be used for recognition. Since the finite automaton is usually faster in recognizing shapes (see Section 4.5), the inference of FSSG is worthwhile to study. Because we already have algorithms for inferring CFSG, we would rather study the CFSG to FSSG conversion instead of a FSSG inference algorithm to obtain FSSG. Besides, the converted FSSG has the same primitive set as the CFSG. This makes comparison of the performance between CFSG and FSSG much more convenient.

According to the theory of formal languages, for an arbitrary CFG G, it is undecidable whether L(G) is regular [43]. In other words, there is no way to say whether there exists a FSG F, such that L(F) = L(G). L(F) is the language generated by F. But we may be able to find F's for a subset of CFG G's because of the following lemma in [43].

Lemma 6.1: [43]

L(G) is not regular if and only if all grammars that generate L are self-embedding, where "self-embedding" is defined as follows:

Definition 6.3: [43]

A CFG G = (N,T,P,S) is self-embedding if $A \overset{+}{\Rightarrow} uAv$ for some u and v in T^+ (neither u nor v can be empty).

Lemma 6.1 implies the following lemma.

Lemma 6.2:

If A CFG G is not self-embedding, then L(G) is regular.

To further study the conversion we need to know the Greibach normal form of CFG.

Definition 6.4: [43]

A CFG G is said to be in Greibach normal form (GNF) if G is λ-free and each production is of the form $A \rightarrow a\alpha$ with $a \in T$, $\alpha \in N^*$. λ indicates empty and λ-free means G has no empty productions.

Theorem 6.1: [43]

If L is a CFL, then L = L(G) for some G in GNF.

The proof of this theorem can be found in [43]. There are several algorithms in [43] which convert an arbitrary CFG into its GNF. We are interested in determining whether or not a CFG in GNF is self-embedding. If not, we can try to construct a FSG F such that L(F) = L(G). Otherwise, the existence of such a F is unknown. We have developed Algorithm 6.3 for detecting the self-embedding property of a general CFG in GNF.

For a better understanding of Algorithm 6.3 and its proof, the meaning of functions P, F, and S are explained as follows. F(p,A) = 1 means that symbol A can be found in production rule p with a nonempty

succeeding string γ. P(A) containing (i,X) indicates that X can be derived from A with the last production rule i. S(A,B) = 1 implies that $A \overset{*}{\Rightarrow} \alpha B\gamma$, with $|\gamma| > 0$. Therefore, if S(X,X) = 1 is found, it implies that $X \overset{+}{\Rightarrow} \alpha X\gamma$ can be found with $|\gamma| > 0$ and $|\alpha| > 0$ because G is in GNF. Hence, G is self-embedding.

Algorithm 6.3:

Input: CFG G = (N,T,P,S) in GNF.

Output: (1) Yes: G is self-embedding

(2) No: G is not self-embedding.

Method:

(1) Number the production rules p = 1 through n

(2) Let $P(A) = \phi$ (empty set) $\forall A \in N$

$F(p,A) = 0$ (Logical False) $\forall A \in N$, $1 \leq p \leq n$

$S(X,A) = 0$ (Logical False) $\forall X,A \in N$

(3) For all rules

if $p: A \rightarrow a\beta X\gamma$ $A,X \in N$, $a \in T$, $\beta,\gamma \in N^*$

then $P(A) = P(A) \cup \{(p,X)\}$

$$S(A,X) = F(p,X) = \begin{cases} F(p,x) + 1 & \text{if } |\gamma| > 0 \\ F(p,x) + 0 & \text{if } |\gamma| = 0 \end{cases}$$

where + indicates Logical 'OR'

(4) Repeat this step until there is no change in the P's, F's, and S's

If $(i,Y) \in P(A)$

then $\forall (j,B) \in P(Y)$

$P(A) = P(A) \cup \{(j,B)\}$

$S(A,B) = F(j,B) + S(A,Y)$

(5) If S(X,X) = 1 for some $X \in N$

then terminate with "Yes",

otherwise "No".

Proof:

To detect the self-embedding property, we have to find out whether there is $A \overset{+}{\rightarrow} uAv$ with $v \neq \lambda$.

(I) To prove: If G is self-embedding, there must be a production k: $X \rightarrow d\beta B\delta$, and $B \overset{*}{\rightarrow} \eta' A\alpha' d\varepsilon T$, $\delta\varepsilon N^+$, β,η', α, $\alpha'\varepsilon N^*$, such that $A \overset{*}{\rightarrow} \eta X\alpha \rightarrow \eta d\beta B\delta\alpha \overset{*}{\rightarrow} \eta d\beta\eta' A\alpha'\delta\alpha$, $\eta\varepsilon(T \cup N)^*$. Step (3) makes $(k,B) \varepsilon P(X)$, $F(k,B) = 1$, and $S(X,B) = 1$. The repeat of (4) will cause all the possible X's derived from A to be included in P(A) with some production i. That is, there must be $(i,X) \varepsilon P(A)$ for some i and also $(j,A) \varepsilon P(B)$ for some j. Then, with step (4), $P(A) = P(A) \cup \{(k,B)\}$, $S(A,B) = S(A,X) + F(k,B) = 1$. And since $(j,A) \varepsilon P(B)$, $P(A) = P(A) \cup \{(j,A)\}$, $S(A,A) = S(A,X) + F(j,A) = 1$. So, the algorithm will terminate with "Yes".

(II) To prove: If there is $S(X,X) = 1$ for some $X \varepsilon N$, then there must be $X \overset{+}{\rightarrow} uXv$, $v \neq \lambda$. This part is similar but the reverse of (I).

Because P(X) has a finite number of elements, F's and S's, step (4) will certainly terminate after a finite number of iterations.

Q.E.D.

Algorithm 6.3 is for general CFG's in GNF. Our CFSG's are only a subset of the CFG. The CFSG's for rigid objects are very simple in syntax. Because a shape within the vision of an observer can always be divided into a finite number of simple curve segments, a shape can always be syntactically described by an FSG. But, whether the nonterminals of an FSG can properly describe the semantically meaningful portions of a shape is another story. Therefore, our CFSG's for rigid objects can al-

ways be converted into FSSG's, but the FSSG may not distinguish well the classes whose differences are in some semantically meaningful portions of the shapes. Algorithm 6.4 is developed for the CFG to FSG conversion.

Algorithm 6.4: CFG to FSG Conversion

Input: A CFG $G = \{N,T,P,S\}$ which is not self-embedding and in GNF.

Output: FSG F, $L(F) = L(G)$.

Method:

(1) $N_z = \{Z_i\text{'s}\}$ is a set of new symbols not in N.

N_z is initialized empty.

(2) $\forall\ A \to aB\gamma,\ |\gamma| > 0$

If $Z_i \to B\gamma$ for some $Z_i \in N_z$,

then replace $A \to aB\gamma$ by $A \to aZ_i$.

Otherwise for some $Z_j \notin N_z$

$$\text{replace } A \to aB\gamma \text{ by } \begin{cases} A \to aZ_j \\ Z_j \to B\gamma \end{cases}$$

$$\text{and } N_z = N_z \cup \{Z_j\}$$

$$N = N \cup N_z.$$

(3) If there are some production rules not

in the form of $A \to aX$, or $A \to a$,

then goto (4).

Otherwise terminate.

(4) Order $<$ on N.

If $A \to B\beta$, then $A < B$,

We obtain $A_1 < \ldots\ldots < A_n \quad |N| \geq n$

(5) $i = n-1$

(6) If i = 0 goto (2),

Otherwise for $A_i \rightarrow A_j \alpha$, $j > i$, $A_j \rightarrow \beta_1 |.....| \beta_m$

replace it by

$$A_i \rightarrow \beta_1\alpha |.....| \beta_m\alpha$$

(7) i = i-1, goto (6).

This algorithm has been implemented in FORTRAN on Purdue's CDC 6500 computer. All the FSSG F's used in Experiment 4.1 and 4.2 were converted from the CFSG G's by using this algorithm. The success of Experiments 4.1 and 4.2 have shown the effectiveness of this algorithm for converting CFSG's of rigid objects.

Since the ALA can infer CFSG's and Algorithm 6.4 can convert the CFSG's to FSSG's, the two algorithms can be cascaded to obtain FSSG's from the noisy training patterns.

The output FSSG from Algorithm 6.4 has not been minimized yet. Since many authors [85,86] have studied machine minimization, we are not going to discuss it in detail here. A minimized FSSG will be economical in both storage and processing time.

CHAPTER 7

RESULTS, CONCLUSIONS, AND SUGGESTIONS

7.1 Summary of Results and Conclusions

A syntactic method for general shape recognition is proposed and studied. This method utilizes attributed grammars to describe and recognize shapes. The attributed grammar is a powerful and convenient tool when designing a highly intelligent descriptive method because it utilizes symbolic productions and numeric attributes to describe syntactic and semantic information. We have successfully used the attributed grammars to solve a class of general shape description and recognition problems. The primitive extraction has been a difficult problem when applying the syntactic method to general nonsymbolic patterns. The PEE idea which has been successful in shape recognition provides a systematic and accurate way to extract primitives. The experimental results in Chapter 4 have shown the feasibilities of this PEE idea and its advantages in both recognition accuracy and computational efficiency.

Since the proposed syntactic method is not designed for any particular application, it has the potential of solving general shape recognition problems without requiring context-sensitive grammars. By employing the error-correcting techniques, the ECPEE parser can recognize badly distorted shapes which cannot be handled by other existing shape

recognition methods. Taking advantage of the available attributes which describe the semantic information of the shape, our error-correcting PEE parser can obtain an error-weight very close to a human's perception of distortion. Just like other error correcting algorithms, the ECPEE parser unfortunately has the disadvantage of costly computation.

The grammatical inference of our particular shape grammars was studied and reported in Chapter 6. The shape grammar can be obtained from noisy vector chains automatically using the ALA, an automatic learning algorithm, or interactively using the ILA, an interactive learning algorithm. Our experiments have shown that both algorithms can obtain grammars which can be used to recognize shapes satisfactorily.

7.2 Suggestions for Further Research

Although results from syntactic shape recognition using attributed grammars appear to be quite satisfactory, there are some aspects about which we need to know more. Let us summarize the topics for further investigation in the following paragraphs.

(1) The recognition functions and the generalized recognition functions can be improved. The 0-1 recognition functions, described in Assumptions 4.1 and 4.2, are derived from the noise study of descriptors. Further investigations into noisy primitives, gradual shape changes, and human perception of shape similarity and dissimilarity may help us obtain more reasonable recognition functions and generalized recognition functions.

(2) The grammatical inference algorithms for shape grammars discussed in Chapter 6 can be improved. Also, the combination of several subgrammars into a single grammar may be worth an extensive study for very complex objects.

(3) The parsing efficiency of an ECPEE parser or a GECPEE parser can be improved. Further research in processing vector strings using sequential classification will also be fruitful.

In addition, there are some other related interesting topics. They are vector pattern generation and moving object identification. We will discuss them in the following two subsections.

7.2.1 Pattern Generation and Error Correction

The shape grammars studied so far are for recognition purposes. But, to generate patterns is one of the capabilities of a pattern grammar [38,93]. Our shape grammars can surely generate patterns which are represented in terms of primitives and their associated attributes. In other words, the patterns generated are strings of primitive symbols and descriptors. In this section, we will discuss the feasibility of generating vector patterns or boundary vector chains.

To generate vector chains from the descriptors, the descriptors describing the primitives must be unique. An angle primitive, the connection between two consecutive curve segments, is uniquely characterized by the angle attribute. But the C-descriptor does not uniquely describe a curve segment, even if the curve segment is simple. We can put some restrictions on the C-descriptors, so that the descriptors will

be unique. For example, $|\hat{C}|/L = 1$, i.e., all the curve primitives have to be straight lines. Then, it becomes possible to map the descriptors to the vectors. That is, if we restrict the semantic information to a certain type of curve segment which can be uniquely described by the attributes, then the vector chains can be generated by the shape grammars.

Unfortunately, a shape grammar which has restricted curve primitives and which can generate the vector chain is not generally effective for recognition. Suppose that we use only straight lines as curve primitives. If the true boundary is a continuous and smooth curve, it can be approximated by a series of straight lines. With different sizes and rotations, the boundary can be approximated by different number of straight lines of different lengths and connecting angles. It may be difficult to recognize the primitives, the straight lines, and the angles. Therefore, a shape grammar which is effective in recognizing vector patterns and in generating symbolic patterns with attributes may not be as effective in generating vector patterns. A shape grammar, which is effective in generating vector patterns, may not be as effective in recognition.

In case we need a grammar for both generation and recognition, we would suggest having two levels of primitive sets. The higher level primitive set is for recognition, and the lower level primitive set is for generation. To avoid confusion, such a grammar can be considered as a dual grammar D_ℓ, a combination of a recognition shape grammar $G_\ell = (V_\ell, T_\ell, P_\ell, S_\ell)$ and a generation shape grammar $H_\ell = (U_\ell, E_\ell, Q_\ell, S_\ell)$. Since H_ℓ is for generation, and the primitives in E_ℓ are uniquely defined by associated descriptors, the attribute rules associated with

production rules in Q_ℓ are useless and hence omitted. G_ℓ is the same as the general form in Section 3.4. ℓ is the label of the shape. Q_ℓ has F-productions in addition to S-production and N-productions. E_ℓ has curve primitives e's and angle primitives a's which describe the connections between consecutive e's, in addition to A's. The F-productions may not be obtained from the descriptors, because the mapping from the descriptor to the vector pattern is generally not unique. Thus, H_ℓ cannot generally be deduced from G_ℓ.

Definition 7.1

$D_\ell = (G_\ell, H_\ell)$ is a dual shape grammar for both recognition and generation.

$G_\ell = (V_\ell, T_\ell, P_\ell, S_\ell)$

$V_\ell = \{S_\ell, N's\}$

$T_\ell = \{F's, A's\}$

P_ℓ:

$S_\ell \rightarrow (XA)^*XA\{Answer_C\}$

$C \leftarrow \ell$

$N \rightarrow (XA)^*X$

$D(N) \rightarrow (D(X) \underset{A}{\oplus})^* D(X)$

where $X \in \{N's, F's\}$

$H_\ell = (U_\ell, E_\ell, Q_\ell, S_\ell)$

$U_\ell = \{S_\ell, N's, F's\}$

$E_\ell = \{e's, A's, a's\}$

Q_ℓ:

$S \rightarrow (XA)^*XA$

$N \rightarrow (XA)^*X$

$F \rightarrow (ea)^*e$

Example 7.1

Suppose we use straight lines as e's, a continuous and smooth boundary of a "K" shape (see Figure 7.1) can be recognized and generated by following dual grammar D_K.

$D_K = (G_K, H_K)$

$G_K = (V_K, T_K, P_K, S_K)$

$V_K = \{S_K, N1, N2\}$

$T_K = \{Fi's, Aj's\}$, $i = 1 \ldots 6$, $j = 1 \ldots 6$

P_K:

$S_K \rightarrow$ N1 A1 F1 A2 N2 A3 F2 A4

$S_K \rightarrow$ F1 A2 N2 A3 F2 A4 N1 A1

$S_K \rightarrow$ N2 A3 F2 A4 N1 A1 F1 A2

$S_K \rightarrow$ F2 A4 N1 A1 F1 A2 N2 A3

N1 → F3 A5 F4

N2 → F5 A6 F6

$H_K = (U_K, E_K, Q_K, S_K)$

$U_K = \{S_K, N1, N2, Fi's\}$, $i = 1 \ldots 6$

$E_K = \{Aj's, ek's, a\ell's\}$, $j = 1 \ldots 6$, $k = 1 \ldots 18$, $\ell = 1 \ldots 12$.

Q_K:

$S_K \rightarrow$ N1 A1 F1 A2 N2 A3 F2 A4

$S_K \rightarrow$ F1 A2 N2 A3 F2 A4 N1 A1

$S_K \rightarrow$ N2 A3 F2 A4 N1 A1 F1 A2

$S_K \rightarrow$ F2 A4 N1 A1 F1 A2 N2 A3

N1 → F3 A5 F4

N2 → F5 A6 F6

F1 → e7

F2 → e14 a9 e15 a10 e16 a11 e17 a12 e18

F3 → e1 a1 e2 a2 e3 a3 e4

F4 → e5 a4 e6

F5 → e8 a5 e9

F6 → e10 a6 e11 a7 e12 a8 e13

As we can see, the S- and N-productions in Q_ℓ correspond to those in P_ℓ, and F-productions in Q_ℓ produce e's and a's for all F in G_ℓ. We can use G_ℓ for recognition, H_ℓ for generation, and both G_ℓ and H_ℓ for error correction. As mentioned before, the error-correcting techniques can be applied in the parsing to recognize partially distorted shape patterns. Once the distorted pattern is recognized using P_ℓ, the corresponding productions in Q_ℓ can be applied to reconstruct the distorted portion of the shape.

7.2.2 Moving Object Identification

Moving object identification is another interesting research topic [94,95,96], especially when different objects have similar shapes at certain viewing angles. For instance, some airplane models have very similar front views but different top views. If an unknown airplane is flying in the sky, we would like to classify it with a minimum amount of time and effort. If the view observed is the front view, then one more view may be necessary to resolve any ambiguity. Therefore, two interesting questions are that: (1) is one more view needed? (2) what is the next best view?

Let us suppose that each object C_i has n_i views. We refer a view to a set of probable shapes observed at a certain viewing angle. Due to noise, resolution, or some other reasons such as removable parts, there may be several appearances, P_{ij}'s for a viewing angle V_j of the object C_i. Each view can be described by a subgrammar $G(C_i,V_j)$, $j = 1,\ldots,n_i$. It was mentioned in Section 4.6 that subgrammars for one class can be

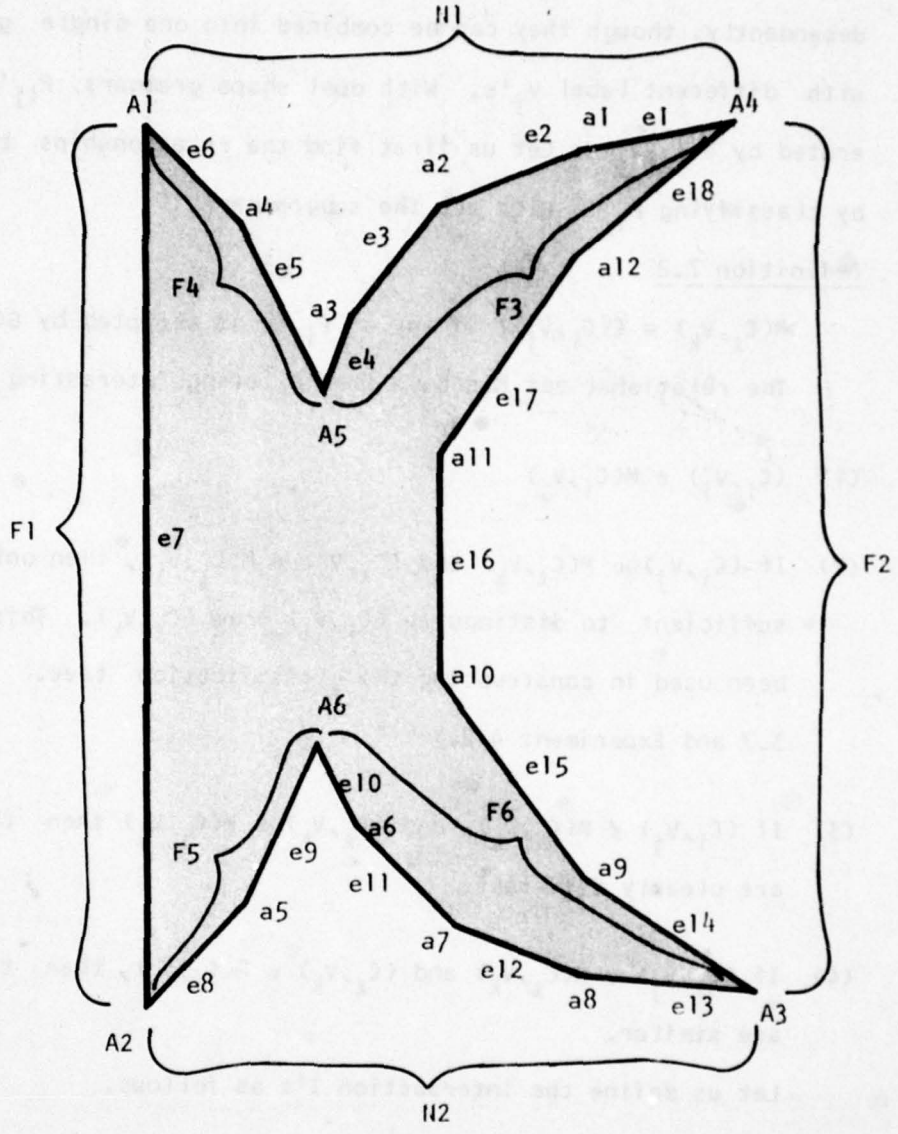


Figure 7.1 A "K" Shape

combined to obtain a single grammar $G(C_i)$, or can be treated independently. In this section, all the subgrammars will be considered independently, though they can be combined into one single grammar $G(C_i)$ with different label V_j's. With dual shape grammars, P_{ij}'s can be generated by $G(C_i,V_j)$. Let us first find the relationships between views by classifying P_{ij}'s with all the subgrammars.

Definition 7.2

$M(C_\ell,V_k) = \{(C_i,V_j) |$ if any of P_{ij}'s is accepted by $G(C_\ell,V_k)\}$.

The relational set M's have the following interesting properties:

(1) $(C_i,V_j) \in M(C_i,V_j)$

(2) If $(C_i,V_j) \in M(C_\ell,V_k)$ and $(C_\ell,V_k) \notin M(C_i,V_j)$, then only $G(C_i,V_j)$ is sufficient to distinguish (C_i,V_j) from (C_ℓ,V_k). This property has been used in constructing the classification tree. (See Section 3.7 and Experiment 4.2.)

(3) If $(C_i,V_j) \notin M(C_\ell,V_k)$, and $(C_\ell,V_k) \notin M(C_i,V_j)$ then the two views are clearly different.

(4) If $(C_i,V_j) \in M(C_\ell,V_k)$ and $(C_\ell,V_k) \in M(C_i,V_j)$, then the two views are similar.

Let us define the intersection I's as follows.

Definition 7.3

$I[(C_i,V_j), (C_\ell,V_k)] = M(C_i,V_j) \cap M(C_\ell,V_k)$

$I[(C_i,V_j),(C_\ell,V_k)] \supset \{(C_i,V_j),(C_\ell,V_k)\}$ implies that both views (C_i,V_j) and (C_ℓ,V_k) can be recognized by $G(C_i,V_j)$ or by $G(C_\ell,V_k)$. If $i \neq \ell$ and the viewing angle of the object is unknown, or known as $V_{j=k}$,

then another view may be necessary to identify the object. If the viewing angle of the object can be obtained, the next best view would be V_b, at which $I[(C_i,V_b), (C_\ell,V_b)]$ has the minimum number of elements in the set.

If we are classifying an unknown moving object against two classes C_i and C_j, we may assume that the viewing angle function of time, $V(t)$, can be obtained by estimating the velocity and trajectory by other means such as radar. The best picture shooting time for classification will be t_0 at which

$$N = |I[(C_i,V(t_0)),(C_j,V(t_0))]|$$

$$= \min_t |I[(C_i,V(t)),(C_j,V(t))]|$$

If $N = 0$, the object can be recognized without the next shot. If $N > 0$, one more shot may be necessary.

The union of the intersection of all angle views can be defined as follows.

Definition 7.4

$$II\ (C_i,C_\ell) = \bigcup_{j,k} I\{(C_i,V_j),(C_\ell,V_k)\}$$

If $II\ (C_i,C_j) = \phi$, then the two objects differ clearly at any viewing angle. Only one view is sufficient to distinguish them.

Definitions 7.3 and 7.4 for two-class problems can be generalized to multi-class problems.

Definition 7.5

$$I[(C_i,V_j),(C_\ell,V_k),(C_m,V_n) \ldots]$$

$$= M(C_i,V_j) \cap M(C_\ell,V_k) \cap M(C_m,V_n) \cap \ldots$$

For an N-class problem, the best angle view will be V_1 such that $I[(C_1,V_1),(C_2,V_1),\ldots,(C_N,V_1)]$ contains the minimum number of C's. Let us suppose they are C_1^1, C_2^1 ..., C_M^1. Then, the best second view will be V_2, such that $I[(C_1^1,V_2),(C_2^1,V_2),\ldots,(C_M^1,V_2)]$ contains the minimum number of C^1's. The view can be selected beforehand from the relational sets. For moving objects, the problem is a little more complicated. We have to find the velocity and the trajectory of the objects, and we have to calculate the time at which we can get the first view V_1 for classification. If the I function of V_1 contains more than one class, then we have to make sure that the subsequent views selected can be obtained after V_1. For this reason, V_1 is the first view of the best view series $V_1 \ldots V_M$ for classification instead of the best view for which the minimum number of C's can be obtained. The best view series $V_1 \ldots V_M$ may have the minimum number of views or the views within the most minimal time with respect to a given velocity and a given trajectory.

LIST OF REFERENCES

LIST OF REFERENCES

[1] Fu, K. S., Sequential Methods in Pattern Recognition and Machine Learning, Academic Press, New York, 1968.

[2] Andrews, H. C., Introduction to Mathematical Techniques in Pattern Recognition, Wiley, New York, 1972.

[3] Duda, R. O. and P. E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1972.

[4] Fu, K. S. (ed.), Syntactic Pattern Recognition Applications, Springer-Verlag, 1977.

[5] Fu, K. S., Syntactic Methods in Pattern Recognition, Academic Press, 1974.

[6] Dudani, S. A., K. J. Breeding, and R. B. McGhee, "Aircraft Identification by Moment Invariants," IEEE Trans. on Computers, Vol. C-26, No. 1, Jan. 1977, pp. 39-45.

[7] Ullman, J. R., Pattern Recognition Techniques, London Butterworths, 1973.

[8] Bacus, J. W. and E. E. Gose, "Leukocycte Pattern Recognition," IEEE Trans. on SMC, No. 6, Sept. 1972.

[9] Green, J. E., "IEEE Conference Records, Symposium on Feature Extraction and Selection in Pattern Recognition," (Argonne, IL), pp. 100-109, IEEE, New York.

[10] Matson, W. et. al., "Computer Processing of SEM Images by Contour Analysis," Pattern Recognition 1970, Vol. 2, pp. 303-312.

[11] Sklansky, J., "Measuring Concavity on a Rectangular Matrix," IEEE Trans. on Computers, Vol. 21, No. 12, Dec. 1972, pp. 1355-1364.

[12] Arcelli, C. and S. Levialdi, "Picture Processing and Overlapping Blobs," IEEE Trans. on Computers, Vol. 20, No. 9, Sept. 1971, pp. 1111-1115.

[13] Young, I. T., J. E. Walker, and J. E. Bowie, "Analysis Techniques for Biological Shape," Information and Control, Vol. 25, 1974, pp. 357-370.

[14] Attneave, F., "Physical Determinants of the Judged Complexity of Shapes," J. of Experimental Psychology, Vol. 53, No. 4, 1957, pp. 221-227.

[15] Kiefer, G. et. al., "Nuclear Images of Cells in Different Functional States," J. of Histchem. and Cytochem., Vol. 22, No. 7, July 1974, pp. 569-577.

[16] Singleton, R. C., "An Algorithm for Computing the Mixed Radix Fast Fourier Transform," IEEE Trans. on Audio and Electroacoustic, Vol. AU-17, No. 2, June 1969, pp. 93-103.

[17] Persoon, E. and K. S. Fu, "Sequential Decision Procedures with Prespecified Error Probabilities and Their Applications," Purdue Univ., TR-EE 74-30, Aug. 1974.

[18] Zahn, C. T. and R. Z. Roskies, "Fourier Descriptors for Plane Close Curves," IEEE Trans. on Computers, Vol. C-12, No. 3, March 1972, pp. 269-281.

[19] Granlund, G. H., "Fourier Preprocessing for Hand Print Character Recognition," IEEE Trans. on Computers, Feb. 1972, pp. 195-201.

[20] Richard, C. W. Jr. and H. Hemani, "Identification of Three Dimensional Objects Using Fourier Descriptors of the Boundary Curves," IEEE Trans. on SMC, Vol. SMC-4, No. 4, July 1974, pp. 371-377.

[21] Wallace, T. and P. A. Wintz, "Shape Information Extraction," in "Image Understanding and Information Extraction," Purdue University, TR-EE 77-16 and TR-EE 77-41, March 1977 and Nov. 1977.

[22] Rosenfeld, A. and A. C. Kak, Digital Picture Processing, Academic Press, 1976.

[23] Rutovitz, D., "Centromere Finding: Some Shape Descriptors For Small Chromosome Outlines," Machine Intelligence, Vol. 5, 1970, pp. 435-462.

[24] Pavlidis, T., "A Review of Algorithms for Shape Description," Proc. of Engineering Foundation Conf. on "Algorithms for Image Processing," Franklin Pierce College, Ringe, NH, Aug. 1976.

[25] Hu, M. K., "Pattern Recognition by Moment Invariants," Proc. of IRE, Sept. 1961, pp. 1428-1429.

[26] Alt, F. L., "Digital Pattern Recognition by Moment," J. of ACM, 1962, pp. 240-258.

[27] Liu, H. C. R., "Shape Description and Characterization of Continuous Change," Ph.D. Thesis, SUNY at Stony Brook, Aug. 1976.

[28] Pavlidis, T., "Syntactic Pattern Recognition on the Basis of Functional Approximation," Pattern Recognition and Artificial Intelligence (C. H. Chen, Ed.), Academic Press, 1976.

[29] Feng, H. Y. F. and T. Pavlidis, "Decomposition of Polygons into Simpler Components: Feature Generation for Syntactic Pattern Recognition," IEEE Trans. on Computers, Vol. C-24, No. 6, June 1975.

[30] Aho, A. V. and T. G. Peterson, "A Minimum Distance Error-Correcting Parser for Context-Free Languages," SIAM J. Comput., Vol. 1, No. 4, Dec. 1972.

[31] Fung, L. W. and K. S. Fu, "Syntactic Decoding for Computer Communication and Pattern Recognition," Purdue University, TR-EE 74-47, Dec. 1974.

[32] Lu, S. Y. and K. S. Fu, "Error-Correcting Parsing for Syntactic Pattern Recognition," Purdue University, TR-EE 77-31, August 1977.

[33] Blum, H., "A Model for Extracting New Descriptors of Shape" in Models for the Perception of Shape and Visual Form, W. Wathen-Dunn (ed.), MIT Press, Cambridge, MA (1977).

[34] Nagel, R. N. and H. Blum, "A Symmetric Axis Basis for Object Recognition and Description," IEEE Conf. on Decision and Control, Clearwater, FL, Dec. 1976.

[35] Ledley, R. S., L. S. Rotolo, T. J. Golab, J. D. Jacobson, M. D. Ginsburg, and J. B. Wilson, "FIDAC: film input to digital automatic computer and associated syntax-directed pattern recognition programming system," Optical and Electo-Optical Information Processing (J. T. Tippett et. al, eds.), Chapter 33, pp. 591-614, MIT Press, Cambridge, MA, 1965.

[36] Shaw, A. C., "The Formal Description and Parsing of Pictures," Rep. SLAC-84 Stanford Linear Accelerator Center, Stanford Univ., Stanford, California, 1968.

[37] Shaw, A. C., "A Formal Picture Description Scheme as a Basic for Picture Processing Systems," Information and Control 14, 9-52 (1969).

[38] Narasimhan, R., "On the Description, Generation, and Recognition of Classes of Pictures," Automatic Interpretation and Classification of Images (A. Grasselli, ed.), Academic Press, New York, 1969.

[39] Fu, K. S., "Stochastic Automata, Stochastic Languages and Pattern Recognition," IEEE Sym. Decision and Control, Austin, Texas, Dec. 1970, published in J. Cybernet. 1, pp. 31-49 (1971).

[40] Swain, P. H. and K. S. Fu, "Stochastic Programmed Grammars for Syntactic Pattern Recognition," Pattern Recognition 4, (1972), Special issue on syntactic pattern recognition.

[41] Fu, K. S. and T. Huang, "Stochastic Grammars and Languages," Int. J. Computers and Information Science 1, (1972).

[42] Freeman, H., "Techniques for the Digital Computer Analysis of Chain-Encoded Arbitrary Plane Curves," Proc. Natl. Elec. Conf. 18, pp. 312-324 (1961).

[43] Aho, A. V. and J. D. Ullman, Theory of Parsing, Translation, and Compiling, Vol. 1, Parsing, Prentice-Hall, 1972.

[44] Lewis, P. M. II, D. J. Rosenkrantz, and R. E. Stearns, Compiler Design Theory, Addison Wesley, 1976.

[45] Greanias, E. C., P. F. Meagher, R. J. Norman, and P. Essinger, "The Recognition of Handwritten Numerals by Contour Analysis," IBM Journal 7, (Jan. 1963).

[46] Munson, J. H., "Experiments in the Recognition of Hand-Printed Text: Part I - Character Recognition," Proc. FJCC, pp. 1125-1138 (Dec. 1968).

[47] Yokoi, S., J. Toriwaki, and T. Fukumura, "An Analysis of Topological Properties of Digitized Binary Pictures Using Local Features," Computer Graphics and Image Processing 4 (1975), pp. 63-73.

[48] Rosenfeld, A., "A Converse to the Jordan Curve Theorem for Digital Curves," Information and Control, 29 (1975), pp. 292-293.

[49] Attneave, F., "Some Information Aspects of Visual Perception," Psych. Rev., Vol. 61, pp. 183-193, 1954.

[50] Davis, L. S., "Understanding Shape: Angles and Sides," IEEE Trans. on Computers, Vol. C-26, No. 3, March 1977, pp. 236-242.

[51] Pavlidis, T. and S. Horowitz, "Segmentation of Plane Curves," IEEE Trans. on Computers, Vol. C-23, pp. 860-870, 1974.

[52] Ramer, U., "An Iterative Procedure for the Polygonal Approximation of Plane Curves," Computer Graphics and Image Processing, Vol. 1, pp. 244-256, 1972.

[53] Bacus, J. W., M. G. Belanger, R. K. Aggarwal, and F. E. Trobaugh, Jr., "Image Processing for Automated Erythrocyte Classification," J. of Histochem. and Cytochem., Vol. 24, No. 1, pp. 195-201, 1976.

[54] Lee, E. T., "The Shape-Oriented Dissimilarity of Polygons and its Application to the Classification of Chromosomes Images," Pattern Recognition, Vol. 6, pp. 47-60, 1974.

[55] Davis, L. S., "Understanding Shape: II. Symmetry," IEEE Trans. on SMC, Vol. SMC-7, No. 3, pp. 204-212, March 1977.

[56] Davis, L. S., "Shape Matching Using Relaxation Techniques," TR-180, Computer Science Center, University of Maryland, Sept. 1976.

[57] Shapiro, L. G., "Inexact Pattern Matching in a Syntactic Pattern Recognition System."

[58] Lee, H. C. and K. S. Fu, "Stochastic Linguistic for Picture Recognition," Purdue University, TR-EE 72-17, June 1972.

[59] Freeman, H., "On the Encoding of Arbitrary Geometric Configurations," IRE Trans. on Elec. Comp., June 1961, pp. 260-268.

[60] Sidhu, G. S. and R. T. Bonte, "Property Encoding: Application in Binary Picture Encoding and Boundary Following," IEEE Trans. on Comp., Vol. C-21, No. 11, Nov. 1972, pp. 1206-1216.

[61] Mui, J. K., K. S. Fu, and J. W. Bacus, "Feature Selection in Automated Classification of Blood Cell Neutrophils," Proc. on IEEE Pattern Recognition and Image Processing Conference, May 30, 1978 at Chicago.

[62] Mui, J. K., K. S. Fu, and J. W. Bacus, "Automated Classification of Blood Cell Neutrophils," J. of Histochemistry and Cytochemistry, Vol. 25, No. 7, pp. 633-640, 1977.

[63] Mui, J. K., J. W. Bacus, and K. S. Fu, "A Scene Segmentation Technique for Microscopic Cell Images," Proc. of the Symposium on Computer Aided Diagnosis of Medical Images (J. Sklansky, ed.), San Diego, CA, 1976.

[64] Nahi, N. E. and M. H. Jahanshahi, "Image Boundary Estimation," IEEE Trans. on Computers, Vol. C-26, No. 8, August 1977.

[65] Rosenfeld, A., "A Nonlinear Edge Detection Techniques," Proc. IEEE, May 1970, p. 814.

[66] Rosenfeld, A. and M. Thurston, "Edge and Curve Detection for Visual Scene Analysis," IEEE Trans. on Computers, Vol. C-20, May 1971, pp. 562-569.

[67] Griffith, A. K., "Edge Detection in Simple Scenes using a Priori Information," IEEE Trans. on Computers, Vol. C-22, 1973, p. 371.

[68] Chien, Y. P. and K. S. Fu, "Preprocessing and Feature Extraction of Picture Patterns - With Application to Chest X-Ray Images," Purdue University, TR-EE 74-20, April 1974.

[69] Rosenfeld, A., Compact Figures in Digital Pictures," IEEE Trans. on SMC, March 1974.

[70] Rosenfeld, A., Picture Processing by Computer, Chapter 9, Academic Press, New York, 1969.

[71] Rosenfeld, A., "Digital Straight Line Segments," IEEE Trans. on Computers, Vol. C-23, No. 12, Dec. 1974.

[72] Tang, G. Y. and T. S. Huang, "Digital Straight Edges," in "Image Understanding and Information Extraction" (by T. S. Huang and K. S. Fu), ARPA Report, School of Electrical Engineering, Purdue University, Feb.-Apr. 1976.

[73] Tsai, W. H. and K. S. Fu, "A Pattern Deformational Model and Bayes Error-Correcting Recognition System,"

[74] Zadeh, L. A., "Fuzzy Sets," Information and Control, Vol. 8, 1965, pp. 338-353.

[75] Kruger, R. P., J. R. Townes, D. L. Hall, S. J. Dwyer III, and G. s. Lodwick, "Automated Radiographic Diagnosis via Feature Extraction and Classification of Cardiac Size and Shape Descriptors," IEEE Trans. on Biomedical Eng., Vol. BME-19, No. 3, May 1972.

[76] McKee, J. W. and J. K. Aggarwal, "Computer Recognition of Partial Views of Curved Objects," IEEE Trans. on Computers, Vol. C-26, No. 8, Aug. 1977, pp. 790-800.

[77] Weszka, J. S., "Survey: A Survey of Threshold Selection Techniques," Computer Graphics and Images Processing 7, 1978, pp. 259-265.

[78] Prewitt, J. M. S. and M. L. Mendelrohn, "The Analysis of Cell Images," Ann. N. Y. Acad. Sci 128, 1966.

[79] Fu, K. S. and T. L. Booth, "Grammatical Inference: Introduction and Survey, Part I and Part II, IEEE Trans. on SMC, Vol. 5, Jan. and July 1975.

[80] Jain, R., "Tolerance Analysis Using Fuzzy Sets," Int. J. Systems Sci., Vol. 7, No. 12, 1976, pp. 1393-1401.

[81] Dubois, D. and H. Prade, "Operations on Fuzzy Numbers," Int. J. Systems Sci., Vol. 9, No. 3, 1978.

[82] Lee, E. T. and L. A. Zadeh, "Note on Fuzzy Languages," Inf. Sci. 1, pp. 421-434.

[83] Lakshmivarahan, S. and K. S. Rajasethupathy, "Considerations for Fuzzfying Formal Languages and Synthesis of Fuzzy Grammars," J. of Cybernetics, 8:1978, pp. 83-100.

[84] Fu, K. S. and S. Y. Lu, "A Clustering Procedure for Syntactic Patterns," IEEE Trans. on SMC, Vol. SMC-7, No. 10, Oct. 1977, pp. 734-742.

[85] Booth, T. L., Sequential Machines and Automata Theory, Wiley, 1967.

[86] Kohavi, Z., Switching and Finite Automatic Theory, McGraw-Hill, 1970.

[87] Lu, S. Y. and K. S. Fu, "Stochastic Error-Correcting Syntax Analysis for Recognition of Noisy Patterns," IEEE Trans. on Computers, Vol. C-26, No. 12, Dec. 1977, pp. 1268-1276.

[88] Persoon, E. and K. S. Fu, "Sequential Classification of Strings Generated by SCFG's," Int. J. of Computer and Information Sciences, Vol. 5, No. 3, 1975, pp. 205-217.

[89] Bellman, R. E. and L. A. Zadeh, "Decision-Making in a Fuzzy Environment," Manag. Sci., Vol. 17. 1970, pp. B-141 - B-164.

[90] Mitchell, O. R., E. J. Delp, and P. L. Chen, "Filtering to Remove Cloud Cover in Satellite Imagery," IEEE Trans. on Geoscience Electronics, Vol. GE-15, No. 3, July 1977, pp. 137-141.

[91] Graham, S. L. and M. A. Harrison, "Parsing of General Context-Free Languages," Advances in Computers, Vol. 14, 1976, (Ed. by Rubinoff and Yovits).

[92] Pavlidis, T., Structural Pattern Recognition, Springer-Verlag, 1977 (pp. 164-168).

[93] Freeman, H., "Computer Processing of Line-Drawing Images," Computing Surveys, Vol. 6, No. 1, (March 1974), p. 57.

[94] Uno, T., T. Tokunaga, and E. Ejiri, "Method of Real-time Recognition of Moving Objects and its Application," Pattern Recognition, Vol. 8, No. 4, Oct. 1976, pp. 201-208.

[95] Potter, J. L., "Velocity as a Cue to Segmentation," IEEE Trans. SMC, Vol. SMC-5, No. 3, May 1975, pp. 390-394.

[96] Vasylyer, V. I., V. V. Razwayer, and V. Yu Reutskyy, "Classification of Movable Objects With Use of the Previous History," Soviet Automatic Control, Vol. 5, No. 2, Mar.-Apr. 1972, pp. 11-17.

[97] Knuth, D. E. The Art of Computer Programming, Vol. 1 Fundamental Algorithms. Addison-Wesley, 1975, pp. 120-227.

[98] Salisbury, A. B. Microprogrammable Computer Architectures.

[99] Chang, N. S. and K. S. Fu, "Parallel Parsing of Tree Grammar," Proc. of Conf. on Pattern Recognition and Image Processing, Chicago, IL, May 31 - June 2, 1978, pp. 262-268.

[100] Stone, H. S. (ed.) Introduction to Computer Architecture. Science Research Associates, 1975.

APPENDICES

APPENDIX A

MODEL PREPARATION, PICTURE TAKING, AND DIGITIZATION

To demonstrate that our method can distinguish shapes by structural differences as well as by tiny boundary differences, we selected four airplane models. They are F102, B52, F86, and MIG-15. F102 and B52 have completely different shape structures from any angle view. F86 and MIG-15 are very similar in shape structure in most of the angle views, but slightly different in boundary details.

For demonstrative purposes, we have made the following assumptions: (1) the picture can be taken at any arbitrary viewing angle, (2) the picture taking is fast enough that the shape in the image looks stationary, and (3) the pictures have relatively low background noise. The first assumption indicates that no particular angle view can be hidden. To satisfy this assumption, we cannot use any visible support for the models. Dudani [6], in his experiment of moment invariants, used an apparatus to hold the airplane in front of the camera. The apparatus controlled the viewing angle. But it blocked a portion of the aircraft from the camera view in a certain range of viewing angles. He used such an apparatus because he had to take pictures at every 5^{o} to collect enough training samples. In our experiment, we do not need to control the exact angle of the view. Therefore, we tied three very thin white threads to the airplane at three widely spaced and balanced positions. These three threads were then hung to the ceiling. Any arbitrary angle could be obtained by changing the lengths of the three threads and the

position of the camera.

The second assumption is reasonable but difficult to directly obtain from our TV camera. After we hung the airplane at a desired angle, it took at least 15 minutes to decrease the pendulum motion to where it was almost unnoticable. But it still kept moving back and forth slowly. The digitization of a 200x200 picture through the TV camera took about 10 seconds. If we took the digital picture directly from the TV camera, the small pendulum motion would distort the picture so much that even man would not know what was in the picture. But, this assumption is not difficult to satisfy, if we first take the analog picture through an ordinary camera using a 1/10 second or faster exposure time, and then digitize the analog picture via the TV camera.

The third assumption simply reduces the background noise to an extent that it is not too difficult to find the boundary. We satisfied this assumption by increasing the contrast between object and background and adjusting the illumination. We used a paper board covered with a large sheet of white paper as the background. And we painted the airplane models with a flat black color spray. The relative distance between the airplane and the background is adjustable.

The first part of our experiment can be summarized in the following steps:

1. Collect the models.

2. Prepare the white background. Paint the models black. Tie the models properly with white threads.

3. Hang up the model at a desired position and angle.

4. Adjust the relative position of the background and the illumination to obtain a good contrast without strong reflections and dark shades.

5. Adjust the camera to the proper position and take pictures.

6. Attach the digital picture to a wall in front of the TV camera.

7. Adjust illumination and digitize.

The collected four airplane models were painted with a flat black color to avoid reflection. They are shown in Figure 4.1. The B52, F86, and MIG-15 were tied with white threads at the two wings and the tail, while the F102 was tied at the two wings and the nose. These threads may be visible in the analog picture, but will not be seen in the digital picture because of digitization resolution. Figure A.1 shows the set-up for taking analog pictures. The set-up for digitization is simpler and is shown in Figure A.2. The whole experiment was designed to be movable and was set up in the Laboratory of Pattern Processing and Advanced Automation, directed by Professor K. S. Fu. The digitization process was controlled interactively through a PDP 11/45 computer in the laboratory. In order to get an adequate resolution with minimal picture size to save storage, we did not restrict ourselves to a fixed picture size. Before digitization, we adjusted the relative distance and focus the TV camera to obtain a reasonably clear picture on a TV monitor. Then, we gave commands through the computer to start digitizing the picture and to save the digital picture on a disc or a magnetic tape. Most

Figure A.1 Laboratory Set-up for Picture Taking

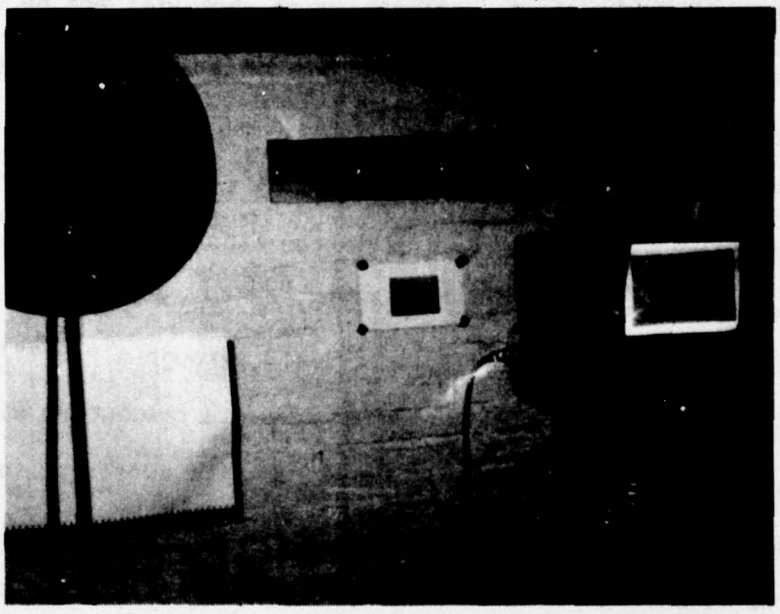


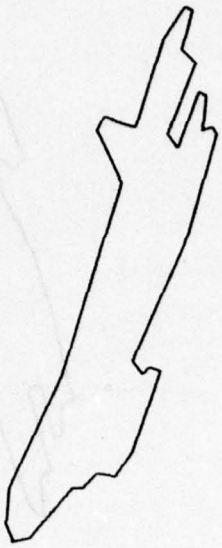
Figure A.2 Laboratory Set-up for Digitization

of our pictures have sizes 180x180 or 200x200 and 7-bit gray levels.

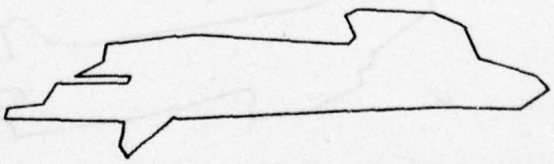
APPENDIX B

This appendix contains 20 shapes obtained through procedures described in Section 4.2 and 4.3. The boundary vector chains start from the bottom of the shape and outline the objects counterclockwise. These 20 shapes are divided into two groups, L and T. The L group contains 6 shapes of view B52,A and 4 shapes of view F102,A. They are shown in (a)-(j). The T group contains 5 B52,A's and 5 F102,A's. They are shown in (k)-(t). All twenty shapes were used for testing in Experiment 4.1. The 10 shapes in L group were used as training patterns and the 10 shapes in T group were used as testing patterns in Experiment 6.1.

(a) B52,A



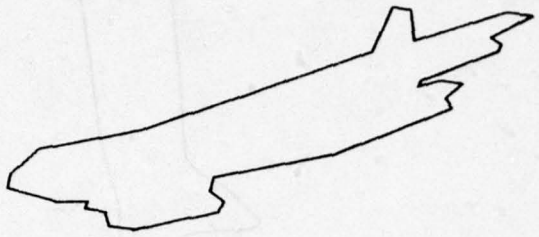
(b) B52,A



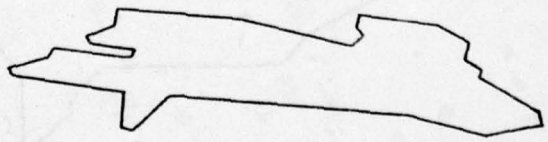
(c) B52,A

(d) B52,A

(e) B52,A

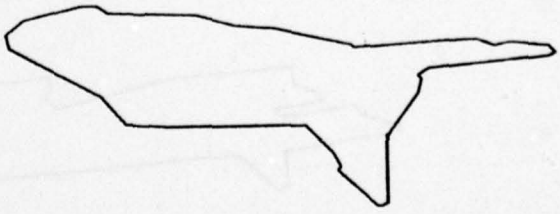


(f) B52,A



(g) F102,A

(h) F102,A

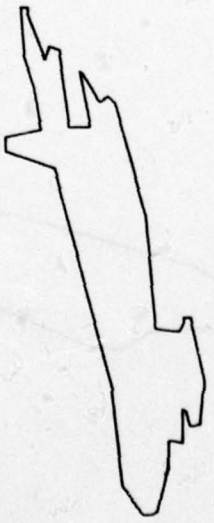


(i) F102,A

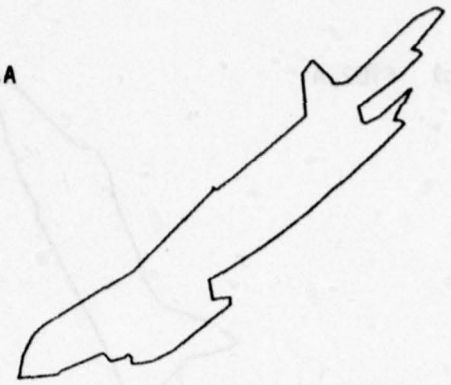
(j) F102,A

(k) B52,A

(l) B52,A

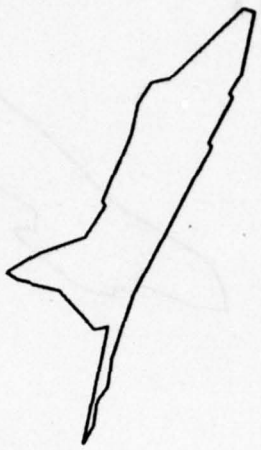


(m) B52,A

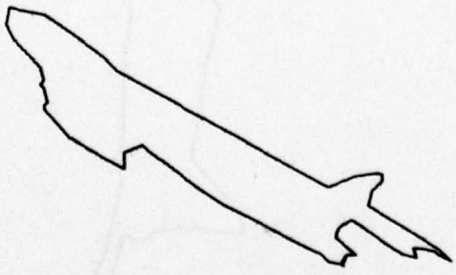


(n) F102,A

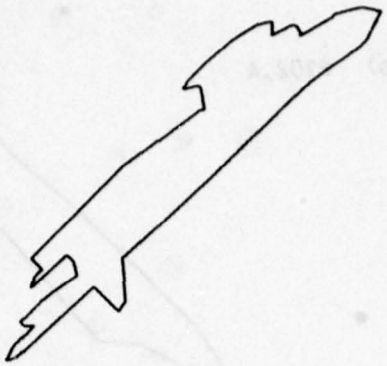
(o) F102,A



(p) B52,A

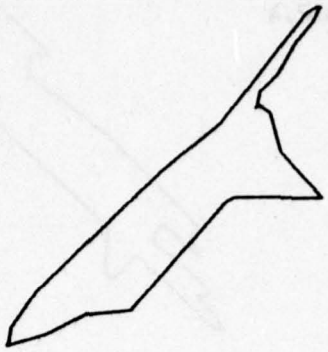


(q) B52,A

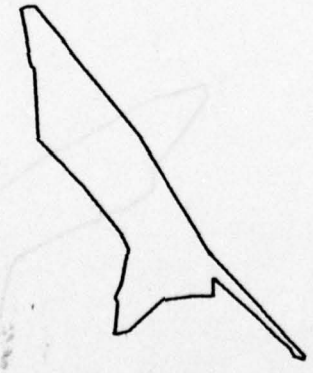


(r) F102,A

(s) F102,A



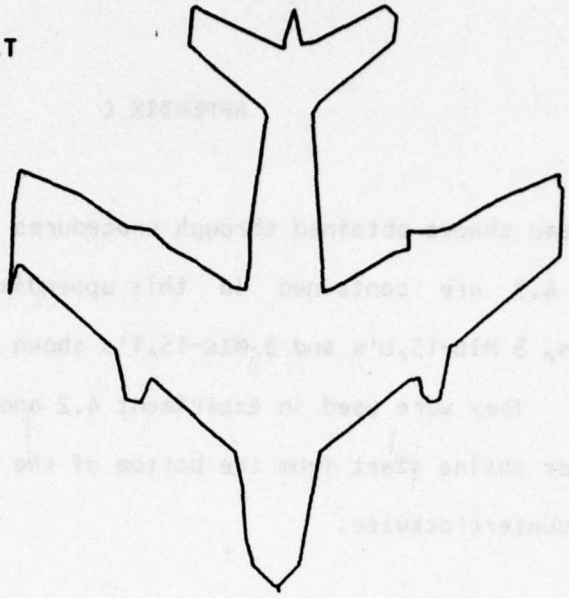
(t) F102,A



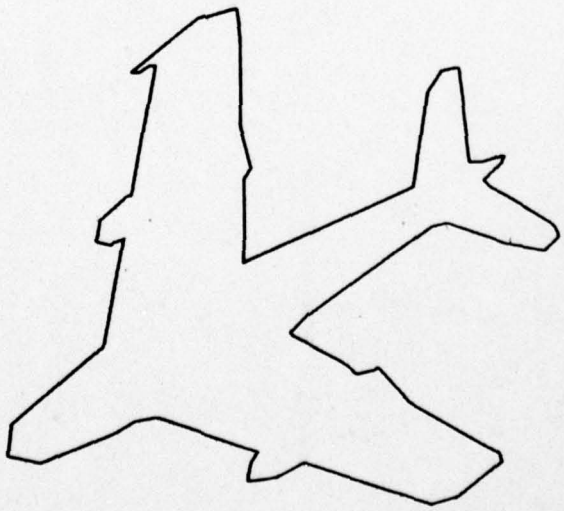
APPENDIX C

Sixteen shapes obtained through procedures described in Sections 4.2 and 4.3 are contained in this appendix. They are 5 F86,T's, 5 MIG-15,V's, 3 MIG-15,U's and 3 MIG-15,T's shown in following pages in sequence. They were used in Experiment 4.2 and Section 4.6. The boundary vector chains start from the bottom of the shape and outline the objects counterclockwise.

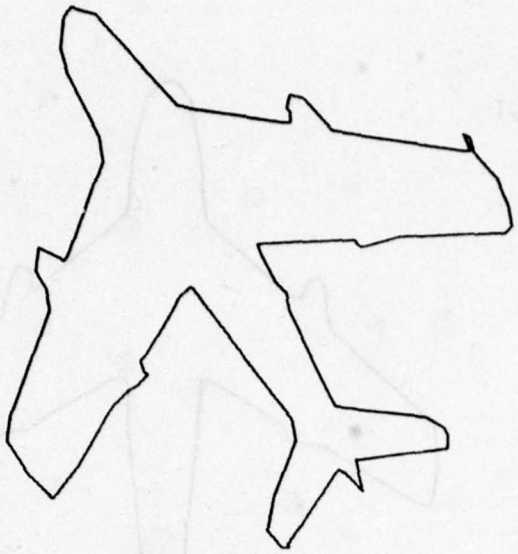
(a) F86,T



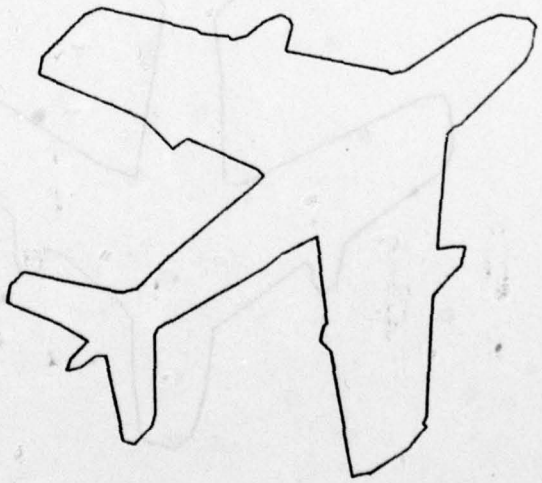
(b) F86,T



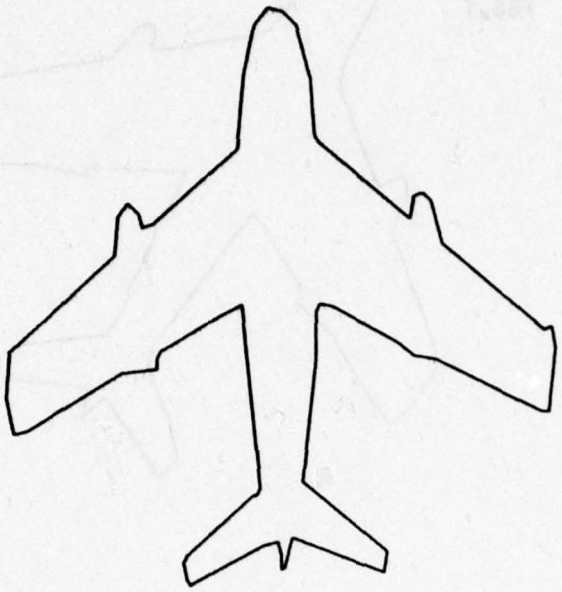
(c) F86,T



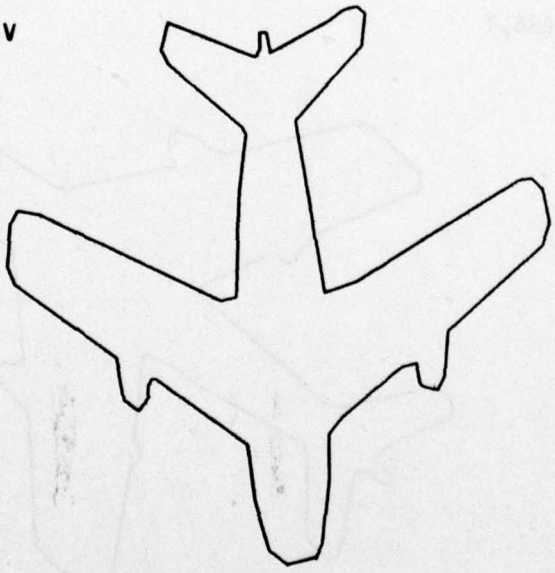
(d) F86,T



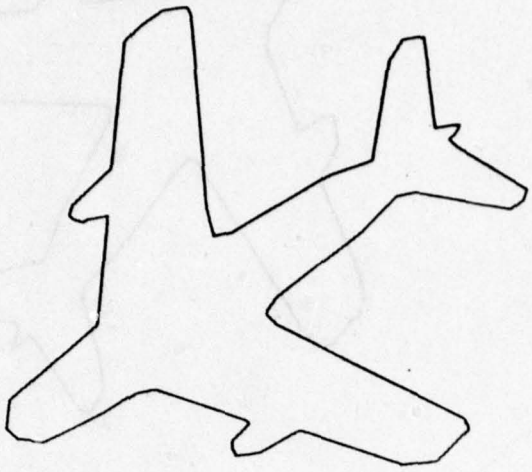
(e) F86,T



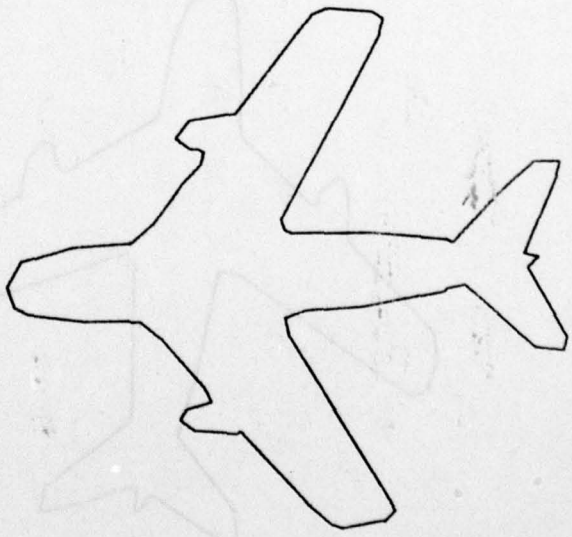
(f) MIG-15,V



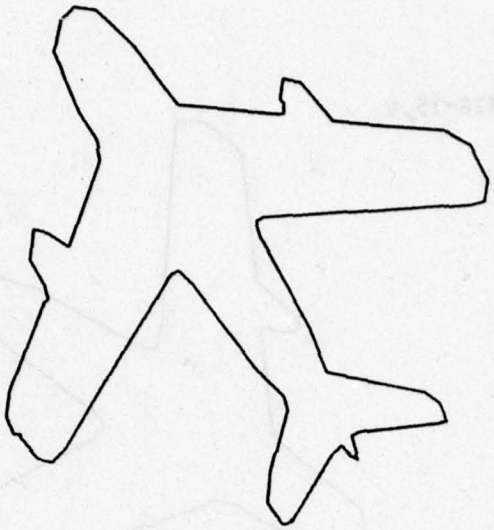
(g) MIG-15,V



(h) MIG-15,V

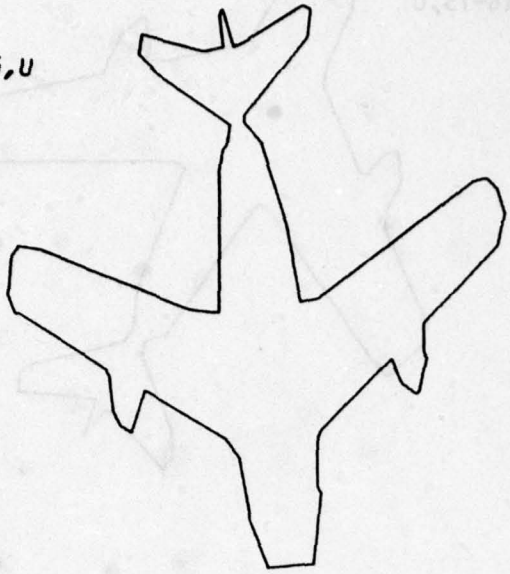


(i) MIG-15,V



(j) MIG-15,V

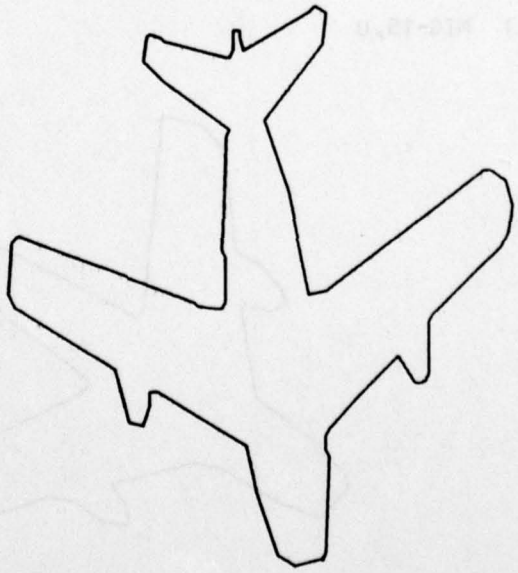
(k) MIG-15,U



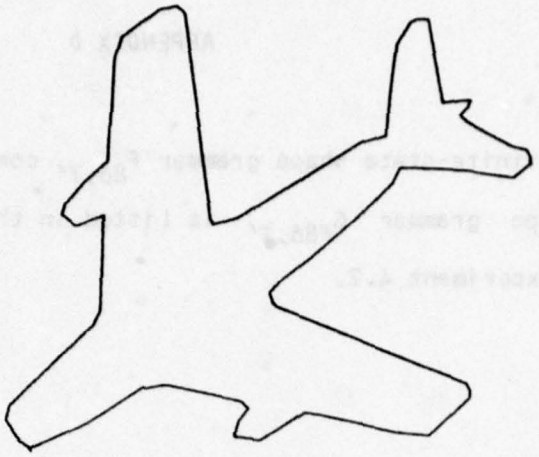
(l) MIG-15,U

(m) MIG-15,U

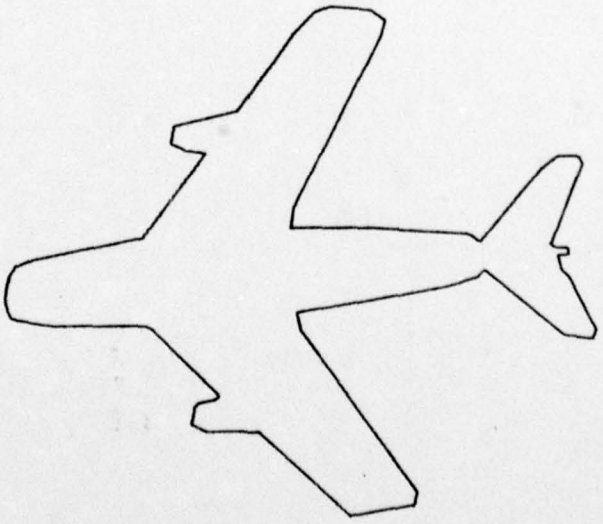
(n) MIG-15,T



(o) MIG-15,T



(p) MIG-15,T



APPENDIX D

The finite-state shape grammar $F_{86,T'}$ converted from the context-free shape grammar $G_{F86,T'}$ is listed in this appendix. This FSSG is used in Experiment 4.2.

$$F_{F86,T} = (V_{F86,T}, T_{F86,T}, P_{F86,T}, S_{F86,T})$$

$$V_{F86,T} = \{S_{F86,T}, NI'S\}, \quad I=1,\ldots,234$$

$$T_{F86,T} = \{FJ'S, AK'S\}, \quad J=1,\ldots,22, \quad K=1,\ldots,21$$

$P_{F86,T}$:

S	-->	F3	A6	N1	N11	-->	F14	A15 N21
S	-->	F8	A11	N2	N12	-->	F16	A17 N22
S	-->	F12	A14	N3	N13	-->	F20	A21 N23
S	-->	F14	A15	N4	N15	-->	F20	A21 N24
S	-->	F22	A19	N5	N17	-->	F6	A1 N25
S	-->	F1	A1	N6	N19	-->	F1	A1 N25
S	-->	F18	A19	N7	N20	-->	F11	A2 N26
S	-->	F2	A18	N8	N22	-->	F17	A4 N27
N1	-->	F4	A7	N9	N23	-->	F21	A5 N28
N2	-->	F9	A12	N10	N24	-->	F21	A5 N29
N3	-->	F13	A3	N11	N25	-->	F8	A11 N30
N4	-->	F15	A16	N12	N26	-->	F12	A14 N31
N5	-->	F19	A20	N13	N27	-->	F22	A19 N32
N6	-->	F8	A11	N14	N27	-->	F2	A18 N33
N7	-->	F19	A20	N15	N28	-->	F3	A6 N34
N9	-->	F5	A10	N16	N29	-->	F3	A6 N35
N9	-->	F5	A8	N17	N14	-->	F9	A12 N36
N8	-->	F18	A19	N18	N18	-->	F19	A20 N37
N16	-->	F7	A9	N19	N21	-->	F15	A16 N38
N10	-->	F10	A13	N20	N30	-->	F9	A12 N39

N31 --> F13 A3 N40
N32 --> F19 A20 N41
N34 --> F4 A7 N42
N35 --> F4 A7 N43
N33 --> F18 A19 N44
N40 --> F14 A15 N45
N36 --> F10 A13 N46
N37 --> F20 A21 N47
N38 --> F16 A17 N48
N39 --> F10 A13 N49
N41 --> F20 A21 N50
N42 --> F5 A10 N51
N42 --> F5 A8 N52
N43 --> F5 A10 N53
N43 --> F5 A8 N54
N44 --> F19 A20 N55
N45 --> F15 A16 N56
N46 --> F11 A2 N57
N47 --> F21 A5 N58
N48 --> F17 A4 N59
N49 --> F11 A2 N60
N50 --> F21 A5 N61
N51 --> F7 A9 N62
N53 --> F7 A9 N63
N55 --> F20 A21 N64
N56 --> F16 A17 N65

N54 --> F6 A1 N66
N52 --> F6 A1 N67
N58 --> F3 A6 N68
N59 --> F22 A19 N69
N59 --> F2 A18 N70
N60 --> F12 A14 N71
N61 --> F3 A6 N72
N66 --> F8 A11 N73
N67 --> F8 A11 N74
N57 --> F12 A14 N75
N62 --> F1 A1 N76
N63 --> F1 A1 N77
N64 --> F21 A5 N78
N65 --> F17 A4 N79
N68 --> F4 A7 N80
N69 --> F19 A20 N81
N71 --> F13 A3 N82
N72 --> F4 A7 N83
N73 --> F9 A12 N84
N74 --> F9 A12 N85
N70 --> F18 A19 N86
N76 --> F8 A11 N87
N77 --> F8 A11 N88
N78 --> F3 A6 N89
N79 --> F22 A19 N90
N79 --> F2 A18 N91

N82 --> F14 A15 N92
N75 --> F13 A3 N93
N80 --> F5 A10 N94
N80 --> F5 A8 N95
N81 --> F20 A21 N96
N83 --> F5 A10 N97
N83 --> F5 A8 N98
N84 --> F10 A13 N99
N85 --> F10 A13 N100
N86 --> F19 A20 N101
N87 --> F9 A12 N102
N88 --> F9 A12 N103
N89 --> F4 A7 N104
N90 --> F19 A20 N105
N92 --> F15 A16 N106
N91 --> F18 A19 N107
N93 --> F14 A15 N108
N94 --> F7 A9 N109
N96 --> F21 A5 N110
N97 --> F7 A9 N111
N99 --> F11 A2 N112
N100 --> F11 A2 N113
N101 --> F20 A21 N114
N102 --> F10 A13 N115
N103 --> F10 A13 N116
N104 --> F5 A10 N117
N104 --> F5 A8 N118
N105 --> F20 A21 N119
N106 --> F16 A17 N120
N107 --> F19 A20 N121
N98 --> F6 A1 N122
N95 --> F6 A1 N123
N110 --> F3 A6 N124
N113 --> F12 A14 N125
N122 --> F8 A11 N126
N123 --> F8 A11 N127
N108 --> F15 A16 N128
N109 --> F1 A1 N129
N111 --> F1 A1 N130
N112 --> F12 A14 N131
N114 --> F21 A5 N132
N115 --> F11 A2 N133
N116 --> F11 A2 N134
N117 --> F7 A9 N135
N119 --> F21 A5 N136
N120 --> F17 A4 N137
N121 --> F20 A21 N138
N124 --> F4 A7 N139
N125 --> F13 A3 N140
N126 --> F9 A12 N141
N127 --> F9 A12 N142
N118 --> F6 A1 N130

N129 --> F8 A11 N143
N130 --> F8 A11 N144
N132 --> F3 A6 N145
N133 --> F12 A14 N146
N136 --> F3 A6 N147
N137 --> F22 A19 N148
N137 --> F2 A18 N149
N140 --> F14 A15 N150
N128 --> F16 A17 N151
N131 --> F13 A3 N152
N134 --> F12 A14 N153
N135 --> F1 A1 N154
N138 --> F21 A5 N155
N139 --> F5 A10 N156
N139 --> F5 A8 N157
N141 --> F10 A13 N158
N142 --> F10 A13 N159
N143 --> F9 A12 N160
N144 --> F9 A12 N161
N145 --> F4 A7 N162
N146 --> F13 A3 N163
N147 --> F4 A7 N164
N148 --> F19 A20 N165
N150 --> F15 A16 N166
N149 --> F18 A19 N167
N154 --> F8 A11 N168

N155 --> F3 A6 N169
N163 --> F14 A15 N170
N151 --> F17 A4 N171
N152 --> F14 A15 N172
N153 --> F13 A3 N173
N156 --> F7 A9 N174
N158 --> F11 A2 N175
N159 --> F11 A2 N176
N160 --> F10 A13 N177
N161 --> F10 A13 N178
N162 --> F5 A10 N179
N162 --> F5 A8 N180
N164 --> F5 A10 N181
N164 --> F5 A8 N182
N165 --> F20 A21 N183
N166 --> F16 A17 N184
N167 --> F19 A20 N185
N168 --> F9 A12 N186
N169 --> F4 A7 N187
N170 --> F15 A16 N188
N157 --> F6 A1 N189
N171 --> F22 A19 N190
N171 --> F2 A18 N191
N175 --> F12 A14 N192
N176 --> F12 A14 N193
N189 --> F8 A11 N194

N172 --> F15 A16 N195
N173 --> F14 A15 N196
N174 --> F1 A1 N197
N177 --> F11 A2 N198
N178 --> F11 A2 N199
N179 --> F7 A9 N200
N181 --> F7 A9 N201
N183 --> F21 A5
N184 --> F17 A4
N185 --> F20 A21 N183
N186 --> F10 A13 N202
N187 --> F5 A10 N203
N187 --> F5 A8 N182
N188 --> F16 A17 N184
N190 --> F19 A20 N204
N192 --> F13 A3
N193 --> F13 A3 N205
N194 --> F9 A12 N206
N191 --> F18 A19 N207
N182 --> F6 A1
N180 --> F6 A1 N197
N197 --> F8 A11 N208
N198 --> F12 A14 N209
N199 --> F12 A14 N192
N205 --> F14 A15 N210
N195 --> F16 A17 N211

N196 --> F15 A16 N212
N200 --> F1 A1 N213
N201 --> F1 A1
N202 --> F11 A2 N214
N203 --> F7 A9 N201
N204 --> F20 A21 N215
N206 --> F10 A13 N216
N207 --> F19 A20 N217
N208 --> F9 A12 N218
N209 --> F13 A3 N219
N210 --> F15 A16 N220
N213 --> F8 A11 N221
N214 --> F12 A14 N192
N219 --> F14 A15 N222
N211 --> F17 A4 N223
N212 --> F16 A17 N224
N215 --> F21 A5 N225
N216 --> F11 A2
N217 --> F20 A21 N226
N218 --> F10 A13 N216
N220 --> F16 A17 N184
N221 --> F9 A12 N227
N222 --> F15 A16 N228
N223 --> F2 A18
N225 --> F3 A6 N229
N224 --> F17 A4 N223

```
N226 -->  F21   A5   N230
N227 -->  F10   A13  N216
N228 -->  F16   A17  N184
N229 -->  F4    A7   N231
N230 -->  F3    A6   N232
N231 -->  F5    A10  N233
N232 -->  F4    A7   N234
N233 -->  F7    A9
N234 -->  F5    A10  N233
```