Hierarchical Constraint Processes for Shape Analysis

LARRY S. DAVIS, MEMBER, IEEE, AND THOMAS C. HENDERSON

Abstract—A major application of syntactic pattern recognition is the analysis of two-dimensional shape. This paper describes a new syntactic shape analysis technique which combines the constraint propagation techniques which have been so successful in computer vision with the syntactic representation techniques which have been successfully applied to a wide variety of shape analysis problems. Shapes are modeled by stratified shape grammars. These grammars are designed so that local constraints can be compiled from the grammar describing the appearance of pieces of shape at various levels of description. Applications to the analysis of airplane shapes are presented.

Index Terms—Artificial intelligence, relaxation procedure, shape recognition, syntactic pattern recognition.

I. INTRODUCTION

THIS paper presents a new approach to shape representation and recognition based on a union of constraint propagation procedures and syntactic pattern analysis procedures. Constraint propagation procedures have been applied to a wide variety of problems in computer vision, especially low-level vision. They can, to a great degree, overcome the intrinsic ambiguity to assigning descriptions (such as "horizontal edge," "vertical edge") to image parts (e.g., pixels). They operate by initially assigning a set of labels (descriptions) to each picture part and then iteratively updating each labeling based on labelings at "adjacent" parts and a model of label compatibility (usually expressed as a relation). Their advantages include locality, since only "adjacent" parts interact, and speed, since each iteration is performed in parallel to each picture part. Davis and Rosenfeld [1] contain a recent survey.

Syntactic pattern analysis techniques have been extensively applied to problems in planar shape recognition and analysis (Fu [2]). In order to apply the syntactic paradigm, a shape must be segmented into pieces which we shall call *primitives*. Each primitive must then be assigned a terminal symbol name from the grammar at hand. The resulting string (or cycle) can then be parsed by an appropriate parsing mechanism, which can even be designed to account for a limited number of missing symbols, extraneous symbols, and incorrect symbols. The advantages of the syntactic approach include its flexibility in analyzing noisy and distorted data and the organization it imposes on the data by generating a parse tree. Although it is possible in applying syntactic techniques to a recognition problem to design a specific grammar for each recognition class (such as a Boeing 747 or a DC10), the more general approach, and the one adopted in this paper, is to design a single grammar for the largest class of shapes (e.g., airplanes) and then to recognize specific shapes based on an analysis of the resulting parse trees (the engine position, e.g., can distinguish between a Boeing 747 and a DC10). This leads to much larger grammars and often less efficient parsers (especially when the number of recognition classes is small).

There are, however, several shortcomings of this syntactic approach (some of which are partially overcome by top-down parsing; see, e.g., Stockman [3]). They include

1) the necessity for computing a unique *segmentation* of the shape into primitives, and

2) the requirement of assigning a *single* terminal symbol name to each primitive.

In computer vision it has been recognized that any single segmentation based on low-level analysis will be bound to contain errors, i.e., missing segments, extraneous segments, etc., because of the many scales of events in an image (Marr and Hildreth [4]). One way to overcome the problem of missing segments is to allow for multiple, overlapping primitives on parts of the shape which are difficult to segment.

Furthermore, it is also the case that a premature commitment to a single description of each piece will be highly erroneous. These errors can be catastrophic as processing reaches higher levels. A more conservative approach assigns *all* plausible descriptions or labels to each piece and allows higher level processes to disambiguate the labelings of each piece.

We will therefore adopt the position that the shape to be analyzed will be decomposed into many, possibly overlapping, primitives, and that each primitive will be labeled with *all* plausible terminal symbol names. A *level-0 hypothesis* will denote the association of a single primitive with a single terminal symbol.

The goal of the syntax analysis procedure is to discover all subsets of the set of level-0 hypotheses which can be used to form a parse of the shape. Note that many (probably most) of the level-0 hypotheses will not be part of any such grammatical subset—it is the contention of this paper that the sooner it is discovered that a level-0 hypothesis (or, as we shall see, a level-n hypothesis) cannot be part of a grammatical description of the shape and can therefore be discarded from further consideration, the more efficient, in space *and* time, the grammatical analysis will be. Such superfluous hypotheses can be discov-

Manuscript received November 30, 1979; revised April 3, 1980. This work was supported in part by the National Science Foundation under Grant ENG-7904037.

L. S. Davis is with the Department of Computer Sciences, University of Texas, Austin, TX 78712.

T. C. Henderson is with the Institute of Communication Theory, German Aerospace Research Establishment, Oberpfaffenhofen, West Germany.

0162-8828/81/0500-0265$00.75

ered by constraint propagation procedures which apply local constraints on the appearance of a shape to a hierarchical network of hypotheses about the shape. We describe how such local contextual constraints can be generated from a particular class of shape grammars, called *stratified context-free shape grammars*. These constraints are not only derived for terminal symbols, but for symbols at *all* levels of description.

Thus, the grammar plays two important roles, as follows:

1) as a source of constraints on the local appearance of the shape, and

2) as a model for imposing organization on larger and larger pieces of the shape.

The remainder of this paper is devoted to the description of a shape analysis process, called a *hierarchical constraint process* for integrating these two roles of grammars into one bottom-up, constraint-based parsing procedure. They are an extension of the hierarchical relaxation process described in Davis and Rosenfeld [5]. The thrust of this paper is primarily how this integration is accomplished, and only secondarily is it concerned with grammar design, except to the extent that the grammars be of a form from which constraints can be derived.

The hierarchical constraint process can be compared with other recently proposed syntactic shape analyzers. Pavlidis and Ali [6] suggest that many shape analysis problems can be solved using relatively simple grammars (i.e., finite-state) whose terminal symbols are themselves syntactic objects (corners, arcs, etc.) and whose input is a piecewise linear approximation to the shape being analyzed. They describe applications to character recognition and printed circuit board analysis. They do not deal explicitly with the problem of superfluous hypotheses, although their system could be extended to handle this problem. In fact, the type of constraints described in Section III could be applied to a network of hypotheses about the locations at "corners," "arcs," etc., so that the parser of Pavlidis and Ali can be extended to include constraint propagation techniques.

Tang and Huang [7] do explicitly consider the problem of superfluous hypotheses and propose a solution to the problem based on the design of what they call a "creation machine." A creation machine is an abstract mechanism which applies formal language theory to the construction of strings which are prefixes of sentences of the language for the grammar at hand. The syntactic-semantic grammars described by Tang and Huang are similar to the stratified grammars presented in Section II. In contrast to the creation machine, the hierarchical constraint process finds locally *ungrammatical* fragments at *all* positions in the shape (rather than just grammatical fragments at the beginning) and propagates the effects of deleting those fragments from the current set of hypotheses about the shape. Thus, the creation machine of [7] and the hierarchical constraint process described in this paper represent different approaches to solving the same problem. Again, it would be possible to integrate the constraint propagation procedures employed by the hierarchical constraint process into the design of Tang and Huang's creation machine.

The remainder of this paper is organized as follows. Section II discusses the grammatical formalism adopted-stratified context-free shape grammars. They are similar to other existing grammatical formalisms, e.g., the geometrical grammars of Vamos [8] and Gallo [9], as well as the more recent syntactic-semantic grammars of Tang and Huang [7] or You and Fu [10]. The class of grammars discussed in Section II has no more formal power than any of these other formalisms, but is designed so that local contextual constraints can be *compiled from* them for use by the hierarchical constraint process. Section III discusses how local constraints are compiled from the grammar, while Section IV discusses how the hierarchical constraint process applies the constraints in conjunction with the grammar to the analysis of a shape. Section V includes experiments in airplane shape analysis and discusses both storage and time requirements of the hierarchical constraint process. Section VI introduces an extension of hierarchical constraint process to uncertain hypotheses; finally, Section VII contains a summary and conclusions.

II. The Grammatical Shape Model

This section introduces the grammatical formalism used by the hierarchical constraint process. The principal design criterion of this formalism is that local constraints on the appearance of the shape can be compiled from the grammars. The shape grammars which we will consider are an extension of the geometrical grammars suggested by Vamos [8] and Gallo [9]; they are similar to other formalisms in the literature (see Section I).

We define a *stratified context-free grammar* G as a 4-tuple (T, N, P, S), where

T is the set of terminal symbols,
N is the set of nonterminal symbols,
P is the set of productions, and
S is the set of start symbols.

Let $V = (N \cup T)$ be the set of vocabulary symbols.

Associated with every symbol $v \in V$ is a level number, $\ln(v)$. For each terminal symbol v, $\ln(v) = 0$. The set of terminal symbols corresponds to the smallest segments of the shapes modeled by the grammar, e.g., short straight-edges of the shape boundary.

Each nonterminal symbol has a level number from 1 to n associated with it. A start symbol has level number n, and for any rule $v := v_1 v_2 \cdots v_r$, if $\ln(v) = k$, $1 \leqslant k \leqslant n$, then $\ln(v_i) = k - 1$, $i = 1, \cdots, r$. Unlike conventional string grammars, vocabulary symbols have a nontrivial structure. A vocabulary symbol v is composed of a ⟨name part⟩, {attachment part}, and a [semantic part], where

1) ⟨name part⟩ is a unique name by which the symbol v is known,

2) {attachment part} is a set of attachment points of the symbol, which are required to specify how the symbol can be combined with other symbols to form higher level symbols, and

3) [semantic part] is a set of predicates which describe certain properties of the symbol, such as its axis, length, etc.

Each production in the grammar is of the form $(v := v_1 v_2 \cdots v_r, A, C, G_a, G_s)$, where

1) $v := v_1 v_2 \cdots v_r$ is the rewrite part which indicates that the symbol v is constructed from the group of symbols $v_1 v_2 \cdots v_r$,

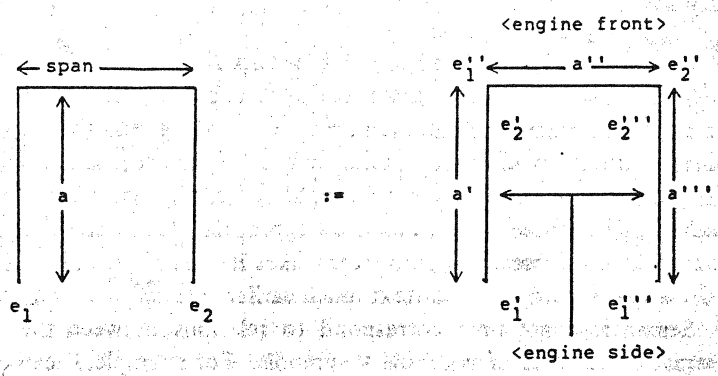


Fig. 1. Example of a production.

2) A is an applicability condition on the syntactic arrangement of the v_i; A specifies how the various attachment points of the v_i must be attached in order to produce a v,

3) C contains semantic constraints on the v_i, and consists of various predicates describing geometric and other properties and relations of the v_i which must be satisfied in order to produce a v,

4) G_a contains rules for generating the attachment part for v, and

5) G_s contains rules for generating the semantic part of v.

As an example of a production, consider how engines are formed (see Fig. 1) in a grammar for an airplane.

⟨engine⟩ {e1, e2} [a, span] :=
⟨engine side⟩ {e1', e2'} [a']
+ ⟨engine front⟩ {e1'', e2''} [a'']
+ ⟨engine side⟩ {e1''', e2'''} [a''']

A : [Join (e1' or e2', e1'') and Join (e1''' or e2''', e2'')
or Join (e1''' or e2''', e1'') and Join (e1' or e2', e2'')]

C : [Parallel (a', a''') and Length (a') = Length (a''')
and Perpendicular (a', a'')
and Parallel (a'', Vector (Midpt (a'), Midpt (a''')))]

G_a : [Set (e1, Unjoined (e1', e2')) and
Set (e2, Unjoined (e1''', e2''')) or
Set (e1, Unjoined (e1''', e2''')) and
Set (e2, Unjoined (e1', e2'))]

G_s : [a := (a' + a''')/2 and span := a''].

This rule specifies that an "engine" is composed of two "engine side" symbols and an "engine front" symbol. A, C, G_a, and G_s can be viewed as a program for producing "engine" from symbols on the right-hand side of the rewrite rule. A specifies the physical connections of the symbols on the right-hand side, i.e., that each end of the "engine front" has an "engine side" attached to it, but the "engine side" symbols are not connected to each other (see Fig. 1). The predicate Join (x, y) is true if x and y correspond to the same point in the shape. C indicates that the two "engine side" symbols should be parallel, of the same length, perpendicular to the "engine front" symbol, and on the same side of the "engine front." G_a and G_s describe the derivation of the attachment points and semantic features for "engine"; the unjoined endpoints of the "engine side" symbols can be given either attachment point name due to the symmetry of the symbol. The function Unjoined (x, y) computes the endpoint which did not satisfy the Join condition in the applicability condition of the production. The function Set (x, y) assigns the physical attributes of the existing endpoint y to the endpoint x of the symbol being constructed. The main axis is the "average" of those of the "engine side" symbols, and the span is exactly that of "engine front."

As a simple example of a complete grammar, consider a house grammar $G = (T, N, S, P)$, where

T = {roof, wall, floor},
N = {top, bottom, house},
S = {house}, and
P = {

(production 1) :
⟨top⟩ {e1, e2} [a] :=
⟨roof⟩ {e1', e2'} [a']
+ ⟨roof⟩ {e1'', e2''} [a'']
A : [Join (e1', e1'') or Join (e1', e2'') or
Join (e2', e1'') or Join (e2', e2'')]
C : [Perpendicular (a', a'') and
Equal(Length (a'), Length (a''))]
G_a : [Set (e1, Unjoined (e1', e2')) and
Set (e2, Unjoined (e1'', e2'')) or
Set (e1, Unjoined (e1'', e2'')) and
Set (e2, Unjoined (e1', e2'))]
G_s : [Set (a, Vector (e1, e2))].

(production 2) :
⟨bottom⟩ {e1, e2} [a] :=
⟨wall⟩ {e1', e2'} [a']
+ ⟨floor⟩ {e1'', e2''} [a'']
+ ⟨wall⟩ {e1''' e2'''} [a''']
A : [Joined (e1', e1'') and Joined (e2'', e1''') or
Joined (e1', e1'') and Joined (e2'', e2''') or
Joined (e1', e2'') and Joined (e1'', e1''') or
Joined (e1', e2'') and Joined (e1'', e2''') or
Joined (e2', e1'') and Joined (e2'', e1''') or
Joined (e2', e1'') and Joined (e2'', e2''') or
Joined (e2', e2'') and Joined (e1'', e1''') or
Joined (e2', e2'') and Joined (e1'', e2''')]
C : [Parallel (a', a''') and Perpendicular (a', a'') and
Equal(Length (a'), Length (a''), Length (a'''))]
G_a : [Set (e1, Unjoined (e1', e2')) and
Set (e2, Unjoined (e1''', e2''')) or
Set (e1, Unjoined (e1''', e2''')) and
Set (e2, Unjoined (e1', e2'))]
G_s : [Set (a, a'')].

(production 3) :
⟨house⟩ { } [a] :=
⟨top⟩ {e1', e2'} [a']
+ ⟨bottom⟩ {e1'', e2''} [a'']
A : [Joined (e1', e1'') and Joined (e2', e2'') or
Joined (e1', e2'') and Joined (e2', e1'')]
C : [Equal (a', a'') and Parallel (a', a'')]
G_a : [ ]
G_s : [Set (a, a')]}.

Fig. 2 shows a typical example of the shapes described by G.

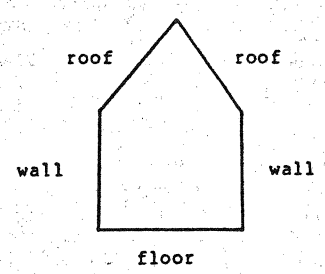


Fig. 2. Typical house shape.

The grammars utilized by the hierarchical constraint process are ordinarily quite large (the grammar for the airplanes analyzed in Section V fills 28 typed pages); it is important, therefore, to discuss the automatic generation of these grammars even though all grammars used so far have been completely hand-generated. It is doubtful, at this time, that such grammars could be inferred automatically and without human intervention for the following reasons.

1) Many nonterminal symbols correspond to meaningful, rather than arbitrary, pieces of the shape (for airplanes, e.g., wings, engines, tails, etc.). It is doubtful that an automatic technique would choose meaningful constituents.

2) The semantic information specified in each production is also meaningful. One could imagine an inference system computing axes for a set of arbitrary pieces and then searching for relationships (such as parallel) between such axes, but it is unlikely that any useful relationships would be discovered if the pieces themselves were not meaningful.

3) The semantic generation part G_s for a rule, presents a difficult inference problem by itself, let alone in the context of a grammatical inference process.

The most realistic approach to the generation of such grammars at this time then seems to be an interactive one, where the human specifies meaningful shape pieces at various levels of description, as well as potentially useful semantic features, and the computer performs the laborious bookkeeping chores, as well as confirmation/refutation/refinement of the human's hypotheses about useful semantic features.

III. Compilation of Constraints

Two types of constraints, syntactic and semantic, can be compiled from a stratified shape grammar. Syntactic constraints describe the possible neighbors a symbol may have at a specific attachment point. If v is a symbol in a grammar G, then let $\mathrm{Nei}(v,a)$ denote the set of ordered pairs of symbols and attachment points which can be attached to v at attachment point a in some sentential form of G. That is, $(v',a') \in \mathrm{Nei}(v,a)$ if and only if v can be attached to v' using point a of v and a' of v'. Now, suppose during the analysis of an actual shape a shape segment s is hypothesized to be an instance of v. Then some actual point of s, say p, is associated with a by the hypothesis. Of course, other segments have been hypothesized as corresponding to other vocabulary symbols. A necessary condition for the hypothesis relating v to s to be part of a grammatical description of the shape is that some other hypothesis relates symbol v' to a segment s' and a point p' in s' to attachment point a' of v' such that

1) $(v',a') \in \mathrm{Nei}(v,a)$, and

2) p' is actually attached to p in the shape.

The sets $\mathrm{Nei}(v,a)$ represent the syntactic constraints and they can be utilized to discard extraneous hypotheses. If these constraints are not applied to the analysis of a shape, then several levels of vocabulary symbols might be built before it is discovered that some hypothesis lacks appropriate context. The use of such constraints, however, makes it possible to detect the lack of appropriate context much earlier.

Semantic constraints correspond to relations between the semantic features of vocabulary symbols. For example, it can be determined from the grammar for airplanes that the main axis of a ⟨plane⟩ is parallel to the main axis of an ⟨engine⟩. Semantic constraints facilitate the incorporation of high-level information into the analysis of a shape, e.g., if the orientation of the fuselage of a plane is known (possibly due to prior image analysis which has discovered runways), then this information can be automatically compiled into constraints on the orientation of the wings through the productions of the grammar.

A. Compiling Syntactic Constraints

Let $G = (T,N,P,S)$, and let v, w, and $x \in V$. Let $at(v)$ denote the set of attachment points of v, and let $av \in at(v)$. We define the three binary relations as follows.

1) v *ancestor:av,aw* w iff there exists a production p such that the rewrite rule of p is $v := \cdots w \cdots$ and there exists an $aw \in at(w)$ such that aw is identified with av in G_a of p. That is, the attachment point aw of the right-hand side symbol w becomes the attachment point av of the left-hand side symbol v. So, for example, in the house grammar we have

⟨bottom⟩ ancestor : $e1, e1'$ ⟨wall⟩

since

a) there is a production ⟨bottom⟩ := ⋯ ⟨wall⟩ ⋯

b) there exists an $aw \in at$ (⟨wall⟩)–namely $e1'$–that is identified with $av \in at$ (⟨bottom⟩) in G_a of the production in (*a*).

2) w *descendent:aw,av* v iff v ancestor:*av,aw* w.

3) v *neighbor:av,aw* w iff

a) there exists a production p such that the rewrite rule of p is $x := \cdots v \cdots w \cdots$ and $aw \in at(aw)$ is specified as being joined to $av \in at(v)$ in the applicability condition A of p, or

b) there exists $x \in V$ with $ax \in at(x)$, and $y \in V$ with $ay \in at(y)$ such that x ancestor:*ax,av* v, and y neighbor:*ay,ax* x, and w descendent:*aw,ay* y. Note that computing the neighbor relation for level k symbols assumes knowing the neighbor relation for all levels greater than k.

So, for example, to illustrate 3a) with respect to the house grammar we have

⟨wall⟩ neighbor:$e1'$, $e1''$ ⟨floor⟩

since

1) there is a production bottom := ⋯ ⟨wall⟩⟨floor⟩ ⋯ ,

2) $e1' \in at$ (⟨wall⟩) is joined to $e1'' \in at$ (⟨floor⟩) in the applicability condition A of that production.

Condition 3b) is more complicated and enables us to compute the constraint that a ⟨roof⟩ is adjacent to a ⟨wall⟩, even though ⟨roof⟩ and ⟨wall⟩ are not mentioned in the same pro-

duction. Specifically, we have

⟨roof⟩ neighbor: $e1'$, $e1'$ ⟨wall⟩.

Since, letting x = ⟨top⟩, $ax = e1'$ in production 3, y = ⟨bottom⟩, and $ay = e1''$ again in production 3, we see that

1) ⟨top⟩ ancestor: $e1'$, $e1'$ ⟨roof⟩ from production 1,
2) ⟨bottom⟩ neighbor: $e1''$, $e1$ ⟨top⟩ from production 3,
3) ⟨wall⟩ descendent: $e1'$, $e1''$ ⟨bottom⟩ from production 2.

Using matrix representations for these relations, the descendents and neighbors of a symbol at a particular attachment point can be computed (see Gries [11] for an introduction to binary relations, their representation using matrices, and their manipulation). The notation $w\ R{:}aw,av\ v$ indicates that w is in relation R to v through attachment point aw of w and av of v. Given k attachment points per vocabulary symbol, the neighbor: i,j relation (which is equivalent to the sets Nei (v,a) discussed above) is computed by iterating the following matrix computation $n-1$ times:

$$\text{neighbor}{:}\,i,j := \text{neighbor}{:}\,i,j + \sum\left\{\text{descendent}{:}\,i,m * \left[\sum(\text{neighbor}{:}\,m,n * \text{ancestor}{:}\,n,j)\right]\right\}.$$

As an example, consider the house grammar given in Section II. An ancestor matrix $M_{aw,av}$ is a square matrix whose order is the number of vocabulary symbols in the grammar, and for which aw specifies the attachment point of the vocabulary symbol for the rows, and av specifies the attachment point of the vocabulary symbol for the columns. Since there are two attachment points for each symbol of the grammar, there should be four matrices specifying the ancestor relation; however, as the attachment conditions and the attachment part generator are symmetric, all four ancestor matrices are equal

$$A = A_{11} = A_{12} = A_{21} = A_{22} = \begin{array}{c|cccccc} & r & w & f & t & b & h \\ r & 0 & 0 & 0 & 0 & 0 & 0 \\ w & 0 & 0 & 0 & 0 & 0 & 0 \\ f & 0 & 0 & 0 & 0 & 0 & 0 \\ t & 1 & 0 & 0 & 0 & 0 & 0 \\ b & 0 & 1 & 0 & 0 & 0 & 0 \\ h & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

where the rows and columns 1 through 6 correspond to the vocabulary symbols ⟨roof⟩, ⟨wall⟩, ⟨floor⟩, ⟨top⟩, ⟨bottom⟩, and ⟨house⟩, respectively. Note that the ⟨house⟩ symbol could be left out as it has no attachment points. The descendant matrix D_{ij} is just the transpose of A_{ij}

$$D = D_{11} = D_{12} = D_{21} = D_{22} = \begin{array}{c|cccccc} & r & w & f & t & b & h \\ r & 0 & 0 & 0 & 1 & 0 & 0 \\ w & 0 & 0 & 0 & 0 & 1 & 0 \\ f & 0 & 0 & 0 & 0 & 0 & 0 \\ t & 0 & 0 & 0 & 0 & 0 & 0 \\ b & 0 & 0 & 0 & 0 & 0 & 0 \\ h & 0 & 0 & 0 & 0 & 0 & 0 \end{array}.$$

Finally, the explicit neighbor relation is given as

$$N = N_{11} = N_{12} = N_{21} = N_{22} = \begin{array}{c|cccccc} & r & w & f & t & b & h \\ r & 1 & 0 & 0 & 0 & 0 & 0 \\ w & 0 & 0 & 1 & 0 & 0 & 0 \\ f & 0 & 1 & 0 & 0 & 0 & 0 \\ t & 0 & 0 & 0 & 0 & 1 & 0 \\ b & 0 & 0 & 0 & 1 & 0 & 0 \\ h & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

which is determined by the grammar. The full neighbor relation is computed as

$$N = N + DNA = \begin{array}{c|cccccc} & r & w & f & t & b & h \\ r & 1 & 1 & 0 & 0 & 0 & 0 \\ w & 1 & 0 & 1 & 0 & 0 & 0 \\ f & 0 & 1 & 0 & 0 & 0 & 0 \\ t & 0 & 0 & 0 & 0 & 1 & 0 \\ b & 0 & 0 & 0 & 1 & 0 & 0 \\ h & 0 & 0 & 0 & 0 & 0 & 0 \end{array}.$$

The full neighbor relation includes the relation between ⟨roof⟩ and ⟨wall⟩, which was not directly represented in the grammar. The "one's" in each row specifies the set of vocabulary symbols which can possibly neighbor the symbol associated with that row.

B. Compiling Semantic Constraints

Semantic constraints can be generated in exactly the same way as syntactic constraints, i.e., by defining binary relations and compiling their transitive closure. This approach is analogous to the syntactic neighbor case; now a relation is defined between every two symbols whose semantic features are related and the closure contains relations not explicitly mentioned in the grammar.

As an example, consider the parallel relation. The parallel relation can occur between the axes of two vocabulary symbols in a variety of ways:

1) they can be explicitly defined as parallel in the semantic consistency part of a production,
2) the semantic generation part of a production may set an axis of the new vocabulary symbol equal to an axis of one of the vocabulary symbols being used to produce the new symbol, or
3) they may be indirectly parallel if there exists a third vocabulary symbol to which they are both parallel.

These relations are computed using a binary-valued matrix, whose rows and columns correspond to the axes of the vocabulary symbols.

Again, consider the grammar for the simple house shape. Let the matrix P denote the parallel relation between the axes of the vocabulary symbols (ordered as in Section III-A). Since only one axis exists per vocabulary symbol, only one matrix is necessary to define the parallel relation. The matrix given di-

rectly by the grammar is

P'	r	w	f	t	b	h
r	0	0	0	0	0	0
w	0	1	0	0	0	0
f	0	0	0	0	1	0
t	0	0	0	0	1	1
b	0	0	1	1	0	0
h	0	0	0	1	0	0.

The transitive closure of P' is

P	r	w	f	t	b	h
r	0	0	0	0	0	0
w	0	1	0	0	0	0
f	0	0	0	1	1	1
t	0	0	1	0	1	1
b	0	0	1	1	0	1
h	0	0	1	1	1	0,

except that elements on the diagonal of P' remain unaltered, where $P = (P' \vee \neg I) \wedge P'!$, where I is the Boolean identity matrix and $P'!$ is the transitive closure of P'. This must be done since by transitivity a vocabulary symbol would be parallel to itself through any other element to which it is parallel. For example, any ⟨wall⟩ hypothesis must have a distinct ⟨wall⟩ hypothesis parallel to it. However, this constraint was already explicit in the productions; a less obvious constraint is that any ⟨top⟩ symbol must be parallel to a ⟨floor⟩ symbol. Since ⟨floor⟩ is a lower level symbol, all ⟨floor⟩ symbols will already have been built by the time ⟨top⟩ symbols are being built, and this can be used to delete ⟨top⟩ hypotheses which are not parallel to any ⟨floor⟩ hypotheses.

In order to add other semantic constraints, a matrix to represent the constraints is needed. The matrix can be computed from the grammar once the relation has been defined in terms of the predicates which appear in the productions. Parallel is a transitive relation, and other transitive relations can be computed in much the same way. Relations which are not transitive, e.g., perpendicular, require special procedures for their computation.

Some applications may prohibit the precompilation of all constraints, e.g., due to the size of the grammar. In such cases a possible alternative is to compute relations only when necessary. Of course, once the relations between the features of two vocabulary symbols have been computed, they can be stored for future reference and need not be recomputed every time.

IV. The Hierarchical Constraint Process

As discussed previously, a major problem associated with applying syntactic techniques to shape analysis is segmenting a shape into pieces which correspond to the terminal symbols of the grammar required to parse the shape. In general, in order to obtain all the required segments, the segmentation procedures must have a high false alarm rate. This results in a large search space for parses of the shape. To overcome this problem, the *hierarchical constraint process* (HCP) uses hierarchical models of objects and model derived constraints to eliminate inconsistent hypotheses at all levels. In particular, using the stratified context-free grammars already described, syntactic (e.g., spatial concatenation) and semantic (e.g., symmetry, collinearity, etc.) constraints can be automatically generated to guide the analysis of the shape.

The hierarchical constraint process computes a bottom-up parse of the shape by applying the constraints to a network of low-level hypotheses about the pieces of the shape and constructing a layered network of hypotheses containing alternative parses of the shape. The processing of this network can be easily described by specifying three simple procedures and two sets which the procedures manipulate. The three procedures are as follows.

1) BUILD: Given level k of the network, BUILD uses the productions of the grammar to construct nodes corresponding to level $k+1$ hypotheses. Any level k symbols which are used to generate a node at level $k+1$ are associated with that level $k+1$ node as supporting it, and it, in turn, is recorded as supported by them. After all nodes on a given level are generated, node pairs corresponding to adjacent boundary segments are linked only if the constraints allow the symbols hypothesized for that pair to be adjacent. Building level 0 involves applying the segmentation strategy to the shape to generate the level 0 nodes.

2) CONSTRAIN: Since each node corresponds to a single hypothesis and since nodes are only linked to compatible nodes, the within layer application of syntactic constraints simply involves removing a node if it has no neighbor at some attachment point. Likewise, a node is removed if any semantic constraints are not satisfied.

3) COMPACT: Given a node n at level k, if level $k+1$ has been built and n does not support a level $k+1$ node, then n is deleted from the network. If any of the nodes which produced n have been deleted, then n is deleted, too.

These procedures operate on two sets of nodes, R_x and R_c, both of which are initially empty. When at level k with R_x and R_c empty, BUILD produces the level $k+1$ hypotheses (or stops if k is the highest level) and puts them into R_x, while putting all level k nodes into R_c. CONSTRAIN examines nodes from R_x. Let n be a node from R_x. If n satisfies all syntactic and semantic constraints, then CONSTRAIN simply deletes n from R_x. Otherwise, CONSTRAIN deletes node n from the network and puts its same level neighbors in R_x (since n might have been their only neighbor at some attachment point) and its supporting and supported nodes in R_c. COMPACT removes nodes from R_c, taking no action if all of the node's original supporting nodes still exist at level $k-1$ and the node still supports at least one level $k+1$ node (if level $k+1$ has been built); otherwise, COMPACT deletes the node from the network and puts its same level neighbors in R_x and its across level neighbors in R_c.

HCP does not eliminate any hypothesis which can be part of a complete parse. This can be seen as follows: BUILD simply generates the next level symbols, and if used without CONSTRAIN and COMPACT will produce all possible hypoth-

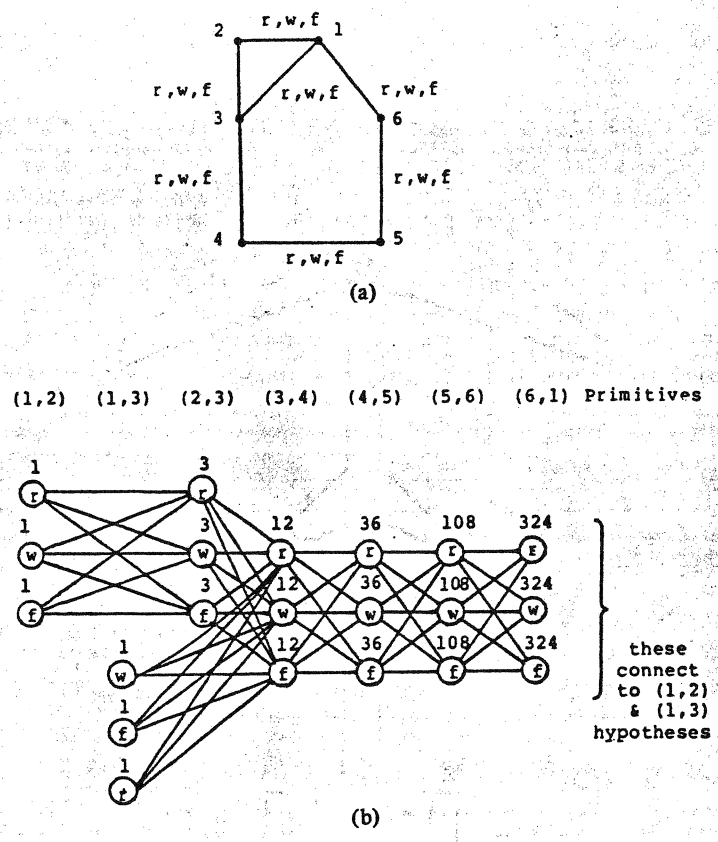


Fig. 3. House hypotheses. (a) Primitives (1, 2), (1, 3), etc., and their hypotheses. (b) Network of hypotheses.

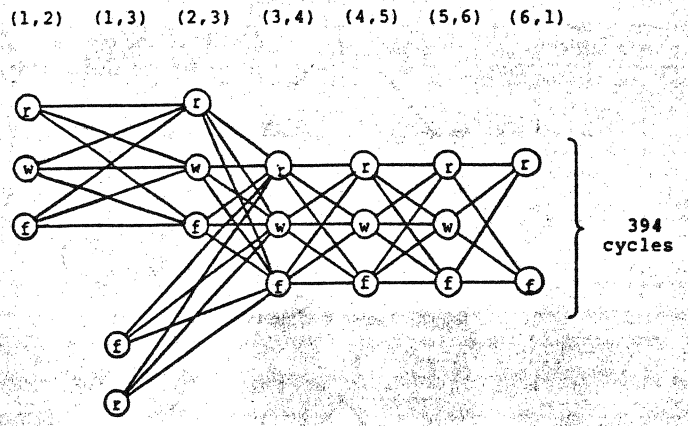


Fig. 4. Hypotheses after semantic constraints.

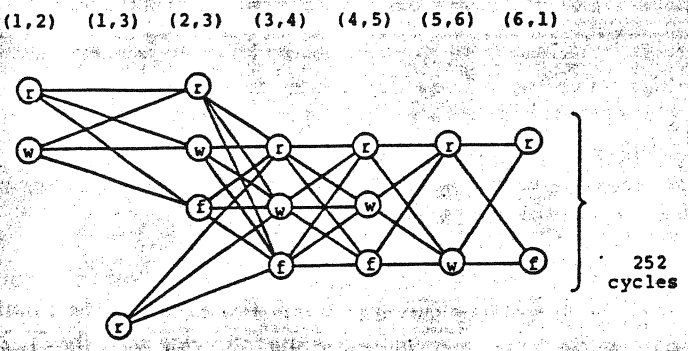


Fig. 5. Hypotheses after syntactic constraints.

eses at every level. CONSTRAIN deletes a hypothesis h only if one endpoint of h has no neighboring hypothesis which can be joined to it, or if it fails to satisfy a semantic constraint. Thus, either h cannot be used to build a higher level symbol, or any symbol which h can be used to build will lack appropriate context at the next higher level. As for COMPACT, there are two cases to consider. First, if a level k hypothesis is not used to produce any level $k+1$ hypothesis, then due to the stratification that level k hypothesis will never produce any higher level hypothesis; thus, it cannot be part of a parse. Finally, if a level k hypothesis h loses the support of one or more of the hypotheses which produced it, then it cannot be part of a complete parse because if it were, then its supporting nodes would be too.

To illustrate the application of HCP, consider once more the house grammar. Suppose the level 0 hypotheses for the seven primitives are as given in Fig. 3(a), where r, w, f, t, b, and h stand for ⟨roof⟩, ⟨wall⟩, ⟨floor⟩, ⟨top⟩, ⟨bottom⟩, and ⟨house⟩, respectively, and every possible hypothesis has been made for each primitive. The syntactic and semantic constraints are as given in Sections III-A and B.

Fig. 3(b) shows the set of level-0 hypotheses organized as a network which we refer to as an *adjacency graph*. The nodes correspond to level-0 hypotheses, and given two nodes, n_1 and n_2, n_1 is connected by a directed arc to n_2 if the primitive corresponding to n_1 in the shape "precedes" and is adjacent to the primitive corresponding to n_2 (precedes is well defined if we adopt a specific sense for following the shape boundary). The number next to each node is the number of distinct paths (from left to right) to that node from a leftmost node. This number is computed for nodes from left to right by assigning the leftmost nodes a value of one, and continuing to the right assigning to each node the sum of those nodes immediately to the left and adjacent to it. Each rightmost node connects to each leftmost node for a closed shape. The goal of HCP is to find all the cycles in the network that correspond to shapes in the language defined by the grammar. In the diagram the ordered pair (i,j) indicates a hypothesis which describes the segment of the shape from point i counterclockwise to point j. Thus, a hypothesis concerning (i_1,j_1) is connected to a hypothesis concerning (i_2,j_2) if $j_1 = i_2$ or $j_2 = i_1$. The networks in Figs. 3-5 are columnated, and the ordered pair at the top of each column indicates the shape segment associated with each hypothesis in that column. Thus, when we speak of the (i,j) hypotheses below, we are referring to the set of hypotheses in the column headed by (i,j).

HCP begins by applying CONSTRAIN to the network of hypotheses. All the syntactic constraints are satisfied since all primitives have been labeled with all level 0 vocabulary symbols. However, the ⟨wall⟩ hypotheses of primitives (1, 3) and (6, 1) are deleted since neither of them has a parallel ⟨wall⟩ hypotheses, thus failing to satisfy a semantic constraint. The result is shown in Fig. 4. The syntactic constraints now cause several hypotheses to be deleted: the ⟨floor⟩ hypotheses are deleted from (1, 2), (1, 3), and (5, 6) since one attachment point fails to have a neighboring ⟨wall⟩ hypothesis. Fig. 5 shows the resulting network.

At this point, all syntactic and semantic constraints are satisfied and all possible level 1 vocabulary symbols can be built. Each hypothesis is checked against all applicable pro-

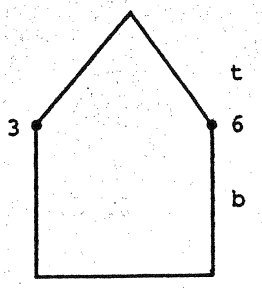


Fig. 6. Level 1 of house.

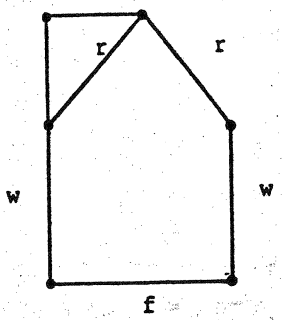


Fig. 7. Final level 0 hypotheses.

ductions in building the next level. For example, the primitives in Fig. 5 can be combined in the following way: the (1, 2) ⟨roof⟩ hypothesis can be joined with the (2, 3) ⟨roof⟩ hypothesis to form a ⟨top⟩ hypothesis; however, the axis of the resulting ⟨top⟩ hypothesis has no ⟨floor⟩ hypothesis parallel to it, and this ⟨top⟩ hypothesis is therefore discarded. The (1, 2) ⟨roof⟩ hypothesis and the (1, 6) ⟨roof⟩ hypothesis, although connected properly, fail the semantic consistency part of the production as they are not perpendicular. The other level 0 hypotheses are matched to productions in a similar manner, and the resulting level 1 hypotheses are shown in Fig. 6. COMPACT is now applied to the level 0 hypotheses and the result is shown in Fig. 7. At this point all incorrect hypotheses have been eliminated and the level 2 start symbol house is directly built. Even this simple example shows the advantage of using syntactic and semantic constraints.

V. Experiments

A set of Pascal programs implementing HCP has been developed. Input to HCP consists of a stratified shape grammar defining the class of shapes to be analyzed and a set of primitives computed from a shape to be analyzed. The primitives are a set of line segments whose descriptions provide information including orientation, length, and endpoints. HCP produces a (possibly empty) network of hypotheses relating primitives to the vocabulary symbols at each level of the grammar. Thus, any highest level hypothesis corresponds to a complete shape in the grammar.

A grammar describing the top view of airplane shapes (down to the level of detail of engines) has been developed. The grammar consists of 37 productions and has seven levels of vocabulary symbols. Note that the grammar was not designed to describe a particular airplane (such as a 747), but rather to model a wide class of airplanes.

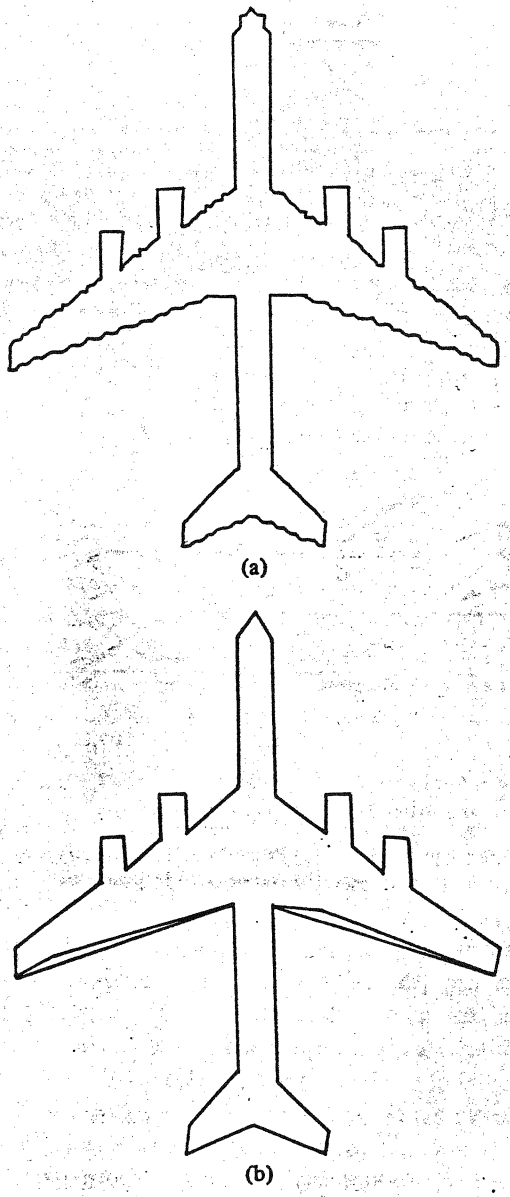


Fig. 8. (a) Shape 1. (b) Segmentation of shape 1.

We will describe the application of HCP to the top view of airplanes. The shapes used in this study were obtained from the literature (shape 1 [10]) and from model airplanes (shapes 2 through 5). Figs. 8–12 display all of the shapes.

The split-and-merge algorithm [12] was used to obtain piecewise linear approximations to the shape. The algorithm was applied at several thresholds of goodness of fit. For these shapes two thresholds were used, i.e., both a close fit and a loose fit were obtained. The close fit picks out small pieces of the shape, while the loose fit picks out longer pieces.

Once the primitives have been found, the initial hypotheses for each primitive must be made. HCP was run with several different numbers of hypotheses per primitive. When only one hypothesis was associated with each primitive, the correct one was associated with each primitive that formed part of a grammatical description of the shape; a "reasonable" hypothesis was chosen for each other primitive which was not part of a grammatical description of the shape (e.g., if the primitive were short, then it might be labeled as an engine side). In the

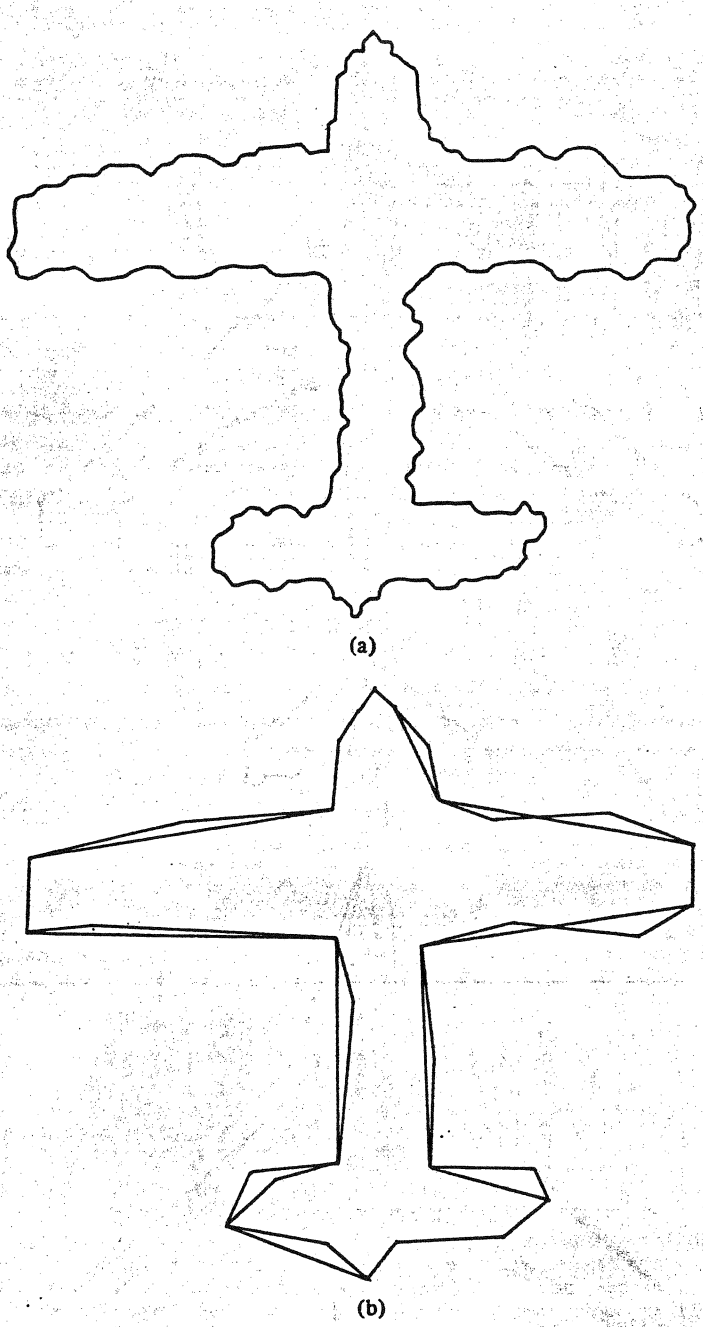


Fig. 9. (a) Shape 2. (b) Segmentation of shape 2.

other experiment described, three hypotheses were associated with each primitive. In general, every terminal symbol should be associated with each primitive, unless some prior information on size or orientation is available which can eliminate some of those guesses.

For each of these sets of initial hypotheses, HCP was run with full constraints and with no constraints. Running HCP with no constraints means that procedure CONSTRAIN is not applied, i.e., procedure BUILD simply builds level after level until all start symbols are created. A measure of space efficiency was defined in terms of the number of hypotheses produced at each level versus the number of hypotheses actually necessary to parse the shape. Given a shape and a level i there is some fixed number of hypotheses $N_a(i)$ which are required at that level to construct all parses of the shape. Let $N_0(i)$ be the number of nodes produced at level i when no constraints

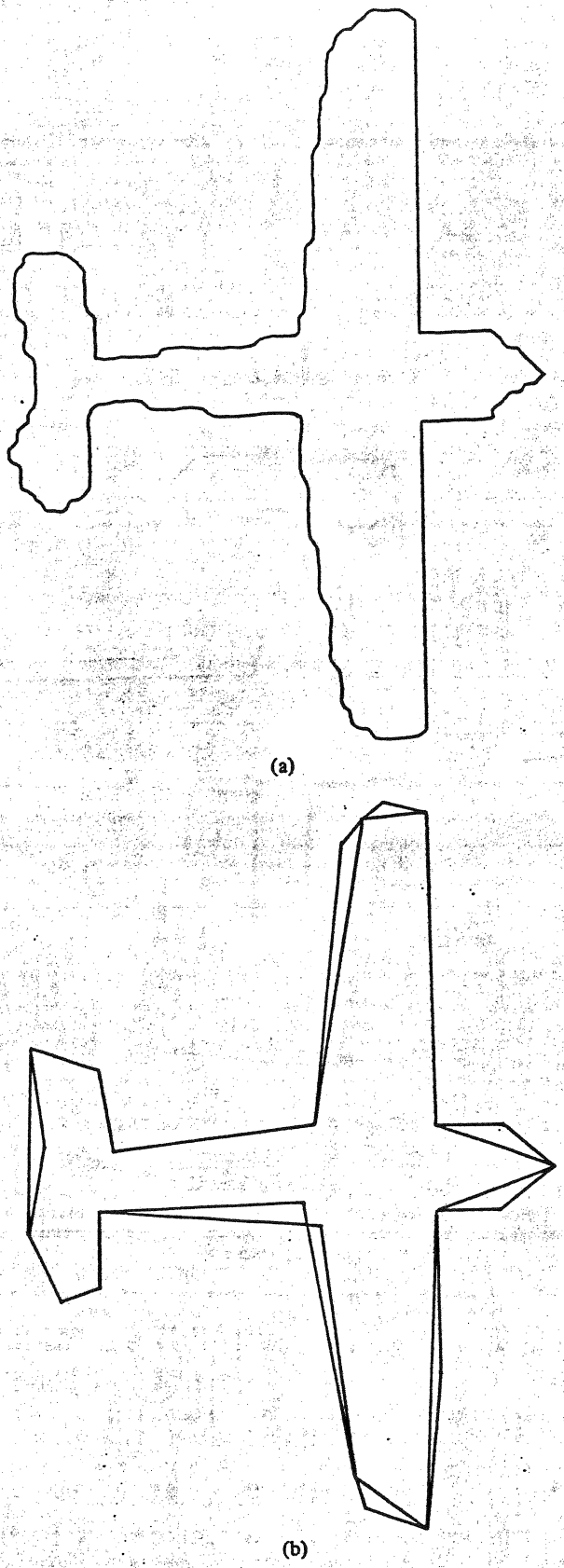


Fig. 10. (a) Shape 3. (b) Segmentation of shape 3.

were used, and let $N_1(i)$ be the number of nodes produced at level i when the constraints were used. Then the efficiency of each process can be given as

$$e_0(i) = N_a(i)/N_0(i) \quad \text{and} \quad e_1(i) = N_a(i)/N_1(i).$$

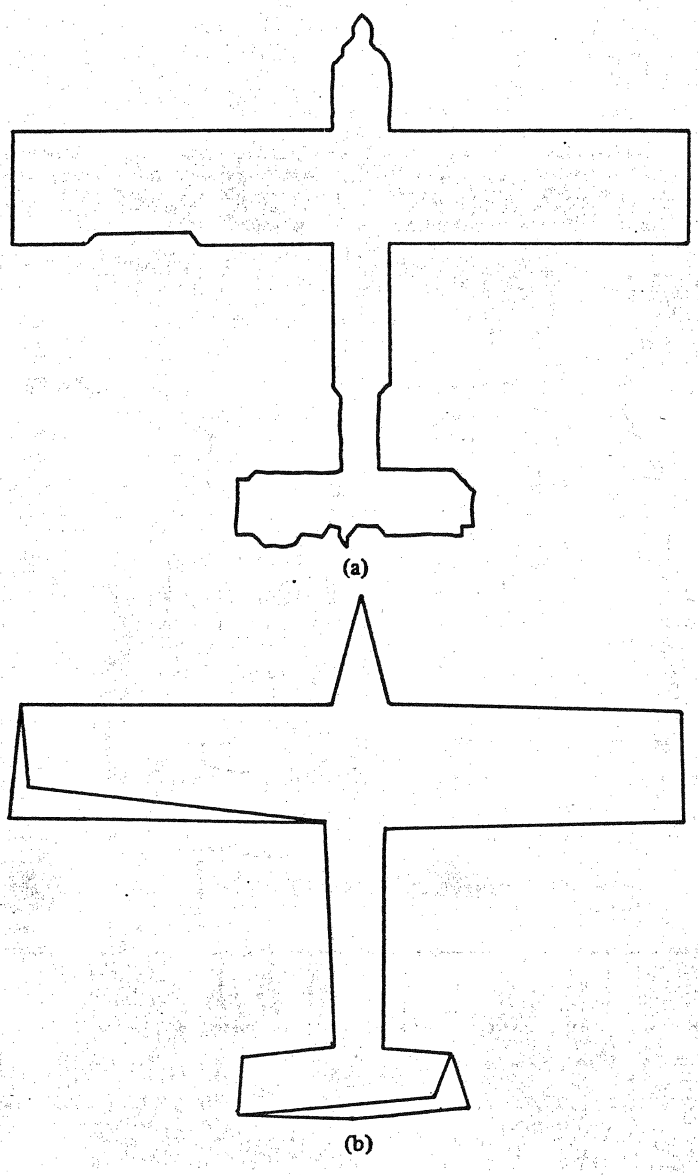


Fig. 11. (a) Shape, 4. (b) Segmentation of shape 4.

TABLE I
NODE EFFICIENCY WITH ONE HYPOTHESIS PER PRIMITIVE

Shape	Node Level 0	1	2	3	4	5	6	
1	.91	.94	.92	.93	.56	.83	1	(No constraints)
	.95	.94	.92	1	1	1	1	(All constraints)
2	.50	.50	.50	.45	.32	.50	1	
	.63	.63	.63	.90	1	1	1	
3	.59	.59	.59	.6	.5	.5	1	
	.70	.70	.70	.75	.75	1	1	
4	.76	.76	.76	.80	.60	1	1	
	.89	.89	.89	1	1	1	1	
5	.67	.67	.67	.62	.48	.80	1	
	.67	.67	.67	1	1	1	1	
Average	.69	.69	.69	.68	.49	.73	1	
	.77	.77	.77	.93	.95	1	1	

These measurements reflect the efficiency of the processes in terms of storage space used, where a value of one means that only as many nodes were used at level i as were needed. Tables I and II give the results for the experiments.

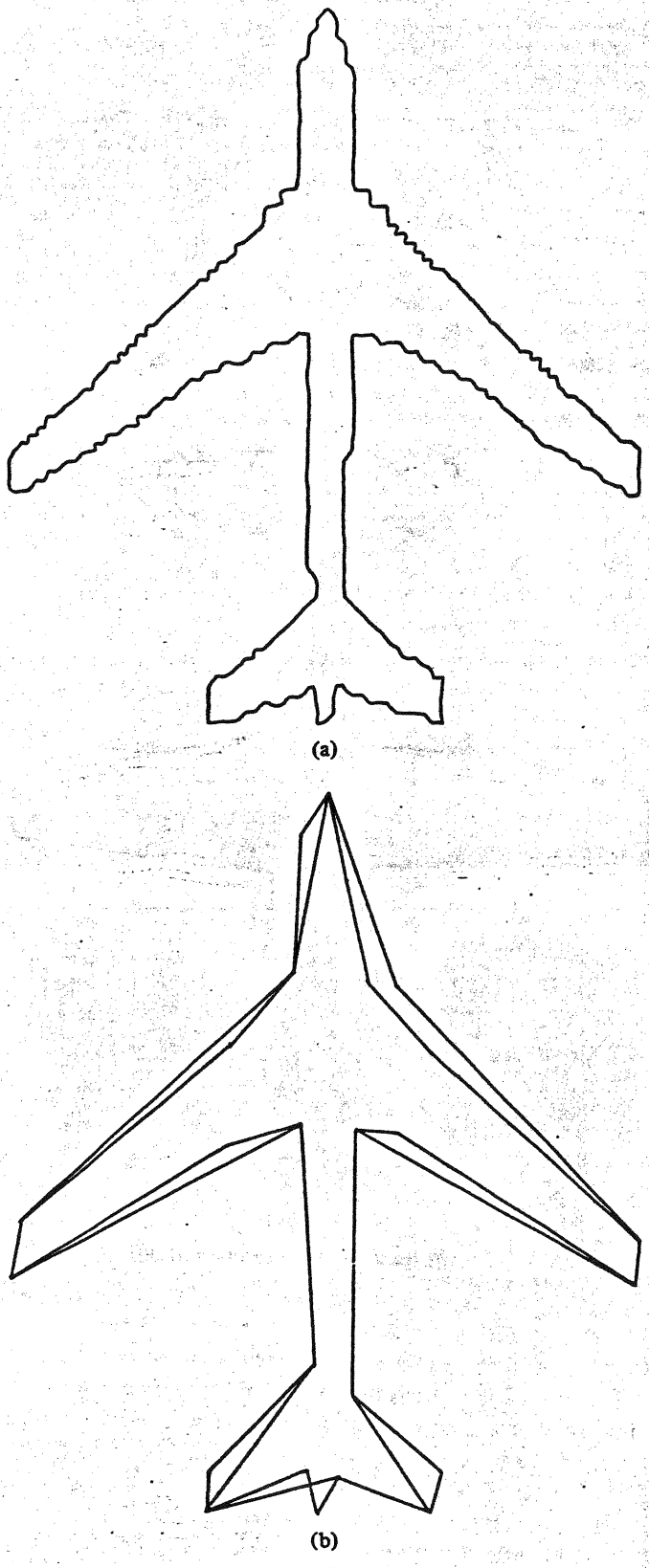


Fig. 12. (a) Shape 5. (b) Segmentation of shape 5.

Tables I and II give a comparison of the node efficiency of HCP for each shape at each level. The first row gives the node efficiency when no constraints are applied to eliminate hypotheses. The second row gives the node efficiency of HCP with all constraints applied. For several shapes, the node efficiency remains fairly constant over the first three levels. This is due to the fact that the first two levels are involved

TABLE II
NODE EFFICIENCY WITH THREE HYPOTHESES PER PRIMITIVE

	Node Level							
Shape	0	1	2	3	4	5	6	
1	.30	.26	.21	.21	.30	.50	1	(No constraints)
	.50	.44	.39	.50	.75	1	1	(All constraints)
2	.17	.25	.50	.45	.32	.50	1	
	.18	.63	.63	.90	1	1	1	
3	.22	.33	.64	.85	.75	1	1	
	.33	.33	.78	.92	1	1	1	
4	.25	.25	.25	.22	.24	1	1	
	.30	.30	.30	.29	1	1	1	
5	.19	.29	.58	.53	.38	.50	1	
	.19	.29	.58	1	1	1	1	
Average	.23	.27	.44	.45	.40	.70	1	
	.30	.39	.53	.72	.96	1	1	

TABLE III
COMPUTATION TIMES FOR HCP WITH (w) AND WITHOUT (wo) CONSTRAINTS. NOTE: AVERAGE FOR 3w DOES NOT INCLUDE SHAPE 1 FOR WHICH HCP wo DID NOT COMPLETE EXECUTION

	Number of Hypotheses					
	1		3		7	
Experiment	w	wo	w	wo	w	wo
1	1:29	1:23	5:22	--	--	--
2	1:20	1:35	3:15	3:29	--	--
3	2:25	2:49	2:54	2:34	8:44	12:03
4	:41	:39	2:39	2:55	9:18	14:35
5	1:23	1:28	3:12	2:56	13:08	19:46
Average	1:27	1:35	3:00	2:59	10:23	15:28

in the description of airplane engines, and if the shape has no engines, then each symbol usually gives rise to a single higher level counterpart. It should be observed that HCP with constraints is consistently more node efficient at all levels and converges much more rapidly to the correct solution than HCP without constraints. As a matter of fact, with constraints HCP always found the correct solution by level 5.

Table III lists computation time for running HCP both with and without constraints, and with 1, 3, and 7 hypotheses per primitive. Several points should be made with respect to this table.

1) The entries which are left blank indicate that HCP exceeded available storage, and execution at that time was aborted.

2) The computation times only grossly reflect the time spent by the programs analyzing the data because of the significant amount of I/O, monitoring, and tracing which the programs perform. Our primary concern throughout is with the design and organization of HCP.

3) The previous remarks nonwithstanding, HCP shows clear computational advantages over a no constraint-based parser, especially when the number of hypotheses/primitives is high. For a low number of hypotheses/primitives, time with and without constraints are quite close, while the storage requirements of HCP with constraints for these problems was generally much lower than for HCP without constraints.

VI. USING HCP WITH UNCERTAIN HYPOTHESES

In the preceding discussions all hypotheses of vocabulary symbols for shape segments were considered to be equally likely. In many situations though, some hypotheses should be regarded with more confidence than others. In what follows we present a generalization of the discrete HCP described above to an HCP which associates likelihoods with hypotheses and applied continuous relaxation-like operators to update the likelihoods. We also discuss embedding HCP into a state-space search procedure for finding the most likely parse of a shape.

Let $G=(P,N,T,S)$ be a stratified context-free grammar, and let $V=N\cup T$. A hypothesis consists of a vocabulary symbol and a likelihood. For hypothesis h, let $L(h)$ be the likelihood of h: If h is a level $k+1$ hypothesis formed from the level k hypotheses $h_1,\cdots,h_n$, then the likelihood of h is obtained as follows:

$$L(h)=\min\{L(h_i)\},\quad i=1,\cdots,n.$$

Hypotheses relating terminal symbols to primitives are constructed if certain features of the primitive satisfy numerical constraints specified in the definition of the terminal symbol. For example, the length of a primitive might be measured, and if the length is found to be less than some value, then it may be possible for that primitive to play the role of an ⟨engine side⟩ in the current shape being processed. The degree to which these numerical constraints are satisfied determines the likelihood of the associated hypothesis.

If only the most likely start symbol hypothesis is desired, then HCP can be embedded in a search algorithm in such a way as to find the best (i.e., most likely) start symbol hypothesis first. The search algorithm employed here is a modified version of the state-space search algorithm (called M^*) described by Barrow and Tennenbaum [13]. The state-space search is organized as a tree. Let $T(r)$ be a finite tree with root node r. For every $n\in T(r)$, let $T(n)$ designate the subtree with root node n: If t is a terminal node of $T(r)$, then $v(t)$ denotes the value of t. A best terminal node t is a terminal node such that $v(t)>v(n)$ for every $n\in$ terminal nodes. If n is a nonterminal node, let $v(n)$ be the value of the best terminal node in $T(n)$. Finding the best parse can now be formulated as finding the best terminal node in a search tree $T(r)$ and can be accomplished by the $A+$ algorithm [13]. This is just one form of the ordered search algorithm.

Algorithm A+

Let f be an evaluation function for estimating the value of nodes in $T(r)$. That is, $f(n)$ is an estimate of $v(n)$ and $v(n)$ is bounded by $f(n)$. Search algorithm $A+$ is defined as follows.

1) Put node r in a set called OPEN.

2) Select $n\in$ OPEN such that $f(n)>f(m)$, for any m distinct from n in OPEN. Break ties arbitrarily, but in favor of terminal nodes.

3) If n is a terminal node, then terminate; else continue.

4) Expand n. Put the successors of n on OPEN. Remove n from OPEN.

5) Go to 2).

It is shown in [13] that $A+$ is admissible and optimal.

We next describe how HCP can be embedded in the $A+$ algorithm. The nodes (or states) of the tree represent multilayer networks of hypotheses. Nodes having start symbol hypotheses are terminal nodes. Each nonterminal node has either

1) one successor corresponding to the result of applying BUILD to the highest level hypotheses of that node, or

2) two successors: one representing the assertion of the most likely hypothesis (called the instantiation hypothesis) for a previously ambiguous piece of the boundary, and the other representing the denial of that assertion.

A level k hypothesis is asserted for a piece of the boundary if no other level k hypothesis which concerns that piece of the boundary is allowed to remain in the network. Likewise, a hypothesis is denied by being deleted from the network. The evaluation function used to order OPEN, f, is the maximum of the likelihoods of the highest level hypotheses of a state.

The constraints between pieces of a shape are no longer used simply to delete a hypothesis, but rather to change its likelihood. The likelihood of a hypothesis is dependent not only on the hypotheses which produced it, but also on the likelihoods of its neighboring hypotheses. For a hypothesis to contribute to a complete parse, it must be joined to a neighboring hypothesis at each endpoint. Thus, given a hypothesis h, an upper bound on the likelihood of any hypothesis produced from h is the minimum of $L(h)$ and the maximum likelihood of any neighboring hypothesis. Then in addition to applying CONSTRAIN and COMPACT to sets of hypotheses, we define a constraint operator (called CONSTRAIN*) to be applied to the likelihood of a hypothesis. The constraint operator assigns likelihoods as follows:

$$L^{t+1}(h) := \min\{L^t(h), \max\{L^t(h_i): h_i \in \mathrm{Nei}(h)\}\}$$

where $L^t(h)$ is the likelihood of hypothesis h after the tth iteration of CONSTRAIN* and $\mathrm{Nei}(h)$ is the set of hypotheses which neighbor h. $L^0(h)$ is the initial likelihood of hypothesis h computed when the level containing h is built. This operator is applied iteratively until no changes in likelihood occur. The M^* algorithm of Barrow and Tennenbaum [13] can be modified to perform search in conjunction with HCP. We call this new algorithm HCP*.

*Algorithm HCP**

0) Generate the initial set of hypotheses and compute the corresponding likelihood of each hypothesis. Apply CONSTRAIN* and CONSTRAIN to the network. Save the result on OPEN.

1) Select the current globally best node s from OPEN, and remove s from OPEN. In case of a tie, choose any terminal node; if none, choose the node with the highest level hypothesis. If s is a terminal node, then halt.

2) If s has no ambiguous pieces of boundary, then

a) build the next level of the network for s, put the resulting node on OPEN and go to 3), otherwise

b) disambiguate a piece of the boundary by instantiating the best hypothesis of s, i.e., generate a branch corresponding to asserting and denying of the instantiation hypothesis, setting up a new node for each.

3) Apply CONSTRAIN, COMPACT, and CONSTRAIN* to the new nodes.

4) Evaluate the global score of each node by computing f, the maximum likelihood of the highest level hypotheses in the network for the node. (If all possible primitive hypotheses are deleted, set the score to 0.)

5) Update the likelihoods of the hypotheses associated with new nodes and put the new nodes on OPEN.

6) Go to 1).

HCP is the above algorithm with 2b) removed and with likelihoods $\{0, 1\}$.

We will now show that the application of such an operator during the search is admissible, i.e., the start symbol produced using HCP* is the same as the start symbol which is produced by using HCP and then choosing the most likely start symbol.

Let $H = \{h_{00}, h_{01}, \cdots, h_{0n}\}$ be the level 0 hypotheses for some shape, and let $S = \{S_1, S_2, \cdots, S_m\}$ be the start symbol hypotheses which can be constructed from H according to G. That is,

$$S_1 \rightarrow h_{11} h_{12} \cdots h_{1r1}$$
$$S_2 \rightarrow h_{21} h_{22} \cdots h_{2r2}$$
$$\vdots$$
$$S_m \rightarrow h_{m1} h_{m2} \cdots h_{rm}$$

and let $L(S_i) = \min\{L(h_j): 1 < j < ri\}$. Suppose that S_b is the best start symbol hypothesis, i.e., $L(S_b) \geq L(S_i)$ for $i \neq b$ and $1 \leq i < m$. We first show that if h_{k_i} is a level k hypothesis used in the production of any S, then even if the constraint operator is applied, $L(h_{k_i}) \geq L(S)$.

Lemma: Let h_{k_i} be a level k hypothesis used in the production of a start symbol S_c. Then $L(h_{k_i})$ is never lowered below $L(S_c)$ by CONSTRAIN*.

Proof: For $k = 0$ it is true, since initially $L(h_{c_i} \geq L(S_c)$, for $1 \leq i \leq rc$, by definition. Suppose $L^t(H_{c_i}) > L(S_c)$ for all $i, 1 \leq i \leq rc$. Then $L^{t+1}(h_{c_i}) = \min\{L^t(h_{c_i}), \max\{L^t(h_j): h_j \in \mathrm{Nei}(h_{c_i})\}\}$; but $L^t(h_{c_i}) \geq L(S_c)$ by assumption, and since $h_{c_{i+1}} \in \mathrm{Nei}(h_{c_i})$ and $L^t(h_{c_{i+1}}) \geq L(S_c)$ by assumption, then $L^{t+1}(h_{c_i}) \geq L(S_c)$.

Now suppose that for $k < j$, $L(h_{k_i}) \geq L(S_c)$, for all hypotheses in level $0, \cdots, j-1$ used to produce S_c. Then initially, all level j hypotheses h_{j_i} have $L(h_{j_i}) \geq L(S_c)$ since they are produced from level $j-1$ hypotheses whose likelihoods are greater than or equal to $L(S_c)$. But then, by an induction similar to above, $L^t(h_{j_i}) \geq L(S_c)$, for all t.

Thus, CONSTRAIN* never lowers the likelihood of any hypothesis used to construct S_c below $L(S_c)$.

Proposition 1: If $L(S_b) > L(S_c)$ for all c such that $c \neq b$, then S_b is the first start symbol hypothesis produced by HCP*.

Proof: Suppose not. Let v be the first node removed from OPEN containing a start symbol hypothesis $S_c \neq S_b$. Then there must be some node m an OPEN containing all of the hypotheses required to construct S_b up to level $k < n$ (node m could not yet contain S_b, since otherwise m would have been picked from OPEN). But, from the lemma the likelihood of those hypotheses must be greater than or equal to $L(S_b)$ and thus, $f(m) \geq L(S_b)$. But, $f(v) = L(S_c)$ because

S_c is the only level n hypothesis in the network of node v. Thus, $L(S_c) = f(v) > f(m) = L(S_b)$, contradicting the assumption that $L(S_b) > L(S_c)$.

VII. Conclusions

This paper has discussed the design of hierarchical constraint processes and presented their application to shape recognition. The particular class of shapes considered was the silhouettes of airplanes viewed from above. We have not considered the more general problem of recognizing a three-dimensional object based on an arbitrary two-dimensional projection.

There are a variety of issues concerning the design of a generally useful shape analysis system which this paper has not addressed. For example, the construction of the airplane grammar was a painful, time-consuming process. Here, it would have been useful to have an interactive tool for constructing such models. Completely automatic grammatical inference mechanisms do exist, but they tend to produce unwieldy and unnatural grammars.

Another important question which deserves further consideration is the way in which hypothesis formation is integrated into the constraint application system. This will have a major impact on the efficiency and performance of the system. For example, instead of assuming that only level 0 symbols of the grammar have semantic descriptions which can be directly compared with the descriptions of the primitives, it may be that there are several levels of the grammar at which this is possible. HCP would now begin by detecting primitives at some suitably high level in the grammar, and applying CONSTRAIN, COMPACT, and BUILD to the resulting layered network. Once HCP has stabilized on this network (all higher levels constructed and all constraints satisfied), the surviving lowest level hypotheses can serve to guide the search for still lower level, and probably even less reliably detected, pieces of the shape.

Many claims have been made [14]-[16] about the relative efficiency of constraint processes when compared with conventional search strategies, but very little effort has been devoted to substantiating or invalidating these claims (exceptions include Gaschnig [17] and Haralick and Gordon [18]). As another research goal, the computational complexity of HCP needs to be investigated by both analytical and empirical (e.g., simulation) studies on abstractions of the pattern analysis problem. Only through such studies can we hope to assess the real significance and practical importance of such systems.

References

[1] L. Davis and A. Rosenfeld, "Cooperative processes for low-level vision: A survey," Dep. Comput. Sci., Univ. of Texas, Austin, Tech. Rep. 123, Jan. 1980.

[2] K. S. Fu, "Introduction to syntactic pattern recognition," in *Syntactic Pattern Recognition Applications*, K. S. Fu, Ed. Berlin: Springer-Verlag, 1977, pp. 1-31.

[3] G. Stockman, "A problem-reduction approach to the linguistic analysis of waveforms," Univ. of Maryland, Tech. Rep. 538, May 1977.

[4] D. Marr and E. Hildreth, "The theory of edge detection," Mass. Inst. Technol., Cambridge, Art. Int. Memo, 1979.

[5] L. Davis and A. Rosenfeld, "Hierarchical relaxation for waveform parsing," in *Computer Vision Systems*, Hanson and Riseman, Eds. New York: Academic, 1978, pp. 101-109.

[6] T. Pavlidis and F. Ali, "A hierarchical syntactic shape analyzer," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-1, pp. 2-10, 1979.

[7] G. Tang and T. Huang, "A syntactic-semantic approach to image understanding and creation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-1, pp. 135-144, 1979.

[8] T. Vamos and Z. Vassy, "Industrial pattern recognition experiment-A syntax aided approach," in *Proc. 1973 IJCPR*, Washington, DC, pp. 445-452.

[9] V. Gallo, "A program for grammatical pattern recognition based on the linguistic method of the description and analysis of geometrical structures," in *Proc. 1975 IJCAI*, Tbilisi, Georgia, USSR, 1975, pp. 628-634.

[10] K. You and K. S. Fu, "Syntactic shape recognition," in *Image Understanding and Information Extraction*, Summary Report of Research for the Period Nov. 1, 1976-Jan. 31, 1977, T. S. Huang and K. S. Fu, Co-Principal Investigators, Mar. 1977, pp. 72-83.

[11] D. Gries, *Compiler Construction for Digital Computers*. New York: Wiley, 1971.

[12] T. Pavlidis and S. Horowitz, "Segmentation of plane curves," *IEEE Trans. Comput.*, vol. C-23, pp. 860-870, 1974.

[13] H. Barrow and M. Tennenbaum, "MSYS: A system for reasoning about scenes," SRI, Menlo Park, CA, AI Tech. Rep. 121, 1976.

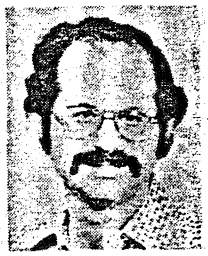
[14] R. Haralick and L. Shapiro, "The consistent labeling problem," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-1, pp. 173-183, Apr. 1979.

[15] R. Haralick, L. Davis, A. Rosenfeld, and D. Milgram, "Reduction operators for constraint satisfaction," *Inform. Sci.*, vol. 14, pp. 199-219, 1978.

[16] A. K. Mackworth, "Consistency in networks of relations," *Artificial Intell.*, vol. 8, pp. 99-118, 1977.

[17] J. Gaschnig, "Experimental case studies of backtrack vs. waltz-type vs. new algorithms for satisficing assignment problems," in *Proc. 2nd Nat. Conf. Canadian Soc. for Comput. Studies of Intell.*, Toronto, Canada, July 19-21, 1978, pp. 268-277.

[18] R. Haralick and G. Elliot, "Increasing tree search efficiency for constraint satisfaction problems," presented at 5th Int. Joint Conf. on Artificial Intell., Tokyo, Japan, Aug. 1979.



Larry S. Davis (S'74-M'77) was born in New York, NY, on February 26, 1949. He received the B.S. degree in mathematics from Colgate University, Hamilton, NY, in 1970, and the M.S. and Ph.D. degrees in computer science from the University of Maryland, College Park, in 1972 and 1976, respectively.

From 1972 to 1977 he was a member of the Computer Vision Laboratory at the Computer Science Center, University of Maryland. Since September 1977 he has been an Assistant Professor in the Department of Computer Sciences at the University of Texas, Austin.

Thomas C. Henderson received the B.S. degree from Louisiana State University in 1973 and the Ph.D. degree in computer science from the University of Texas, Austin, in 1979.

He served as a Research Assistant in the Applied Research Laboratories at the University of Texas from 1975 to 1979. In 1979 he joined the Institute of Communication Theory at the German Aerospace Research Establishment in Oberpfaffenhofen, West Germany, where he is currently a Research Associate. He has conducted research and published in the areas of shape analysis, artificial intelligence, and physiological psychology.