Shape Recognition Using Hierarchical Constraint Analysis *

by

Thomas C. Henderson

and

Larry S. Davis

May 1979 TR-96

*This paper will appear in the 1979 Pattern Recognition and Image Processing Conference to be held in Chicago, IL, August 6-8, 1979.

This research was supported in part by the National Science Foundation undergrant ENG 74-04986 and by the Applied Research Laboratories, University of Texas at Austin.

1.0 Introduction

A major application of syntactic pattern recognition is the analysis of two-dimensional shape. In order to adopt the syntactic approach, the shapes to be analyzed must be segmented into pieces which correspond to the terminal symbols of some grammar, and these pieces must subsequently be analyzed by a parsing mechanism. Many syntactic methods assume that the pieces can be found easily (top-down methods provide a wide class of exceptions, e.g., see Stockman [S1]). However, in most real problems, the design of a segmentation procedure that can find (almost) all of the pieces will require the acceptance of a high false alarm rate - i.e., many of the hypothesized pieces may not, in fact, be part of a "grammatical" description of the shape.

This paper discusses a general parsing procedure which has been designed specifically to overcome this problem. Shapes are modeled by hierarchical, or stratified grammars. These grammars are designed in such a way that local contextual constraints can be automatically compiled from the grammar at all levels of description of the shape. These constraints can then be iteratively applied to an initial set of hypotheses by a relaxation procedure (see Davis [D1] or Davis and Rosenfeld [D3]). In what follows, we will describe algorithms designed to compile these constraints and to employ the constraints to analyze shapes.

We will first introduce a class of shape grammars, called stratified context-free shape grammars, which provide a strict hierarchical structure for vocabulary symbols. Both syntactic and semantic contextual constraints for all the vocabulary symbols can be

generated automatically from such grammars.

These contextual constraints can be exploited by a hierarchical relaxation process. Such a process constitutes a bottom-up, constraint-based parsing method and attempts to overcome the combinatorial explosion in parsing the shape implied by the segmentation.

Examples of the application of this hierarchical system to airplane recognition are described.

2.0 Grammatical Shape Models

Grammatical models for shape analysis have been developed and investigated by Fu [F2] among others. With some simple modifications, these models can be integrated in a natural way with relaxation techniques. An extension of the geometrical grammars of Vámos [V1] and Gallo [G1] will be used to model shapes.

A STRATIFIED CONTEXT-FREE GRAMMAR, G, is a quadruple (T, N, P, S), where

T is the set of terminal symbols,

N is the set of non-terminal 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, $ln(v) : V \rightarrow \{0,1,...n\}$. For every $v \in T$, $ln(v) = 0$.

1) T - corresponds to relatively large pieces of the shapes modeled by the grammar, e.g., straight-edge approximations to the boundary of the shape.

2) N - consists of a set of symbols each of which has a level number from 1 to n associated with it. A start symbol has level number n, and in any rule v := a (the rewrite part of a production), if ln(v) = k, $1 \leq k \leq n$, then every symbol in the string a is at level k-1. Furthermore, for every $v \in V$

v := <name part> {attachment part} [semantic part], where

a) <name part> is a unique name by which the symbol v is known

b) {attachment part} is a set of attachment points of the symbol,

c) [semantic part] is a set of predicates which describe certain aspects of the symbol.

3) P - consists of productions of the form (v:=a,A,C,Ga,Gs), where

a) v:=a is the rewrite part that indicates the replacement of the symbol v by the group of symbols a, where

$v \in N$ and
$a = v1v2...vk$ ($vi \in V$ and ln(vi) = ln(v)-1), i = 1,k)

b) A - set of applicability conditions on the syntactic arrangement of the vi, i = 1,k.

c) C - semantic consistency of the vi, i = 1,k, and consists of various predicates describing geometric and other properties of the vi.

d) Ga - rules for generating the attachment part for v.

e) Gs - rules for generating the semantic part of v.

As an example of the productions of the grammar, consider how engines are formed:

```
< engine > { e1,e2 } [ a,span ] :=
          < engine side > { e1',e2' } [ a' ] +
          < engine front > { e1'',e2'' } [ a'' ] +
          < engine side > { e1''',e2''' } [ a''' ]
```

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

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

Ga : [ el := Unjoined(el',e2') and e2 := Unjoined(el''',e2''')
    or el := Unjoined(el''',e2''') and e2 := Unjoined(el',e2') ]

Gs : [ 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, Ga and Gs 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). 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". Ga and Gs describe the derivation of the attachment points and semantic features for "engine"; the unjoined end points of the "engine side" symbols can be given either attachment point name due to the symmetry of the symbol. The main axis is the average of those of the "engine side" symbols, and the span is exactly that of "engine front".

Stratified grammars naturally give rise to a large set of contextual constraints on the organization of a shape. It is these constraints which the hierarchical relaxation process will utilize to analyze shapes.

3.0 Hierarchical Relaxation

As discussed in the introduction, a major problem associated with syntactic pattern recognition is the segmentation of the object into pieces which correspond to the terminal symbols of the grammar. A high false alarm rate implies that many primitives will be generated, and correspondingly many terminal symbols hypothesized from them, thus implying a large search space. In order to overcome these difficulties, a hierarchical relaxation process (HRP) uses hierarchical models of objects and uses model derived constraints to eliminate inconsistent hypotheses at each level of the model. 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.

Primitives for the grammatical analysis are generated by computing several piecewise linear approximations to the boundary of the shape. A modified split-and-merge algorithm (Pavlidis [P1] and Horowitz [H2]) fits straight edges to the boundary using the cornerity measure proposed by Freeman and Davis [F1] to choose break points. For each point on the original boundary, an error measure defined as the minimum distance from that boundary point to the line segment which approximates a boundary segment containing that boundary point is computed. Then, an error measure for the line segment is defined to be the sum of the errors of each underlying boundary point. Primitives are generated at various error thresholds; even though stricter thresholds are applied to segmentations already generated,

this is not the same as applying the stricter thresholds to each segment which would result in a hierarchical set of primitives. Neighboring primitives from different thresholds are not guaranteed to have endpoints that meet exactly, but primitives are now defined as neighbors if they have endpoints that lie within some disk of fixed size. By computing several segmentations, it is hoped that all the necessary primitives will be found. The search will be made feasible by the constraints implied in the grammar and imposed by the relaxation techniques.

The association of terminal symbols with primitives will (in the limit) be to hypothesize every terminal for each primitive. However, methods for reducing the number of hypotheses are being studied and include using a more global analysis to derive indications of appropriate scale, orientation, etc. from simple global properties, e.g., histogramming selected features of the primitives themselves and using the model to infer properties of particular terminal symbols.

Before discussing the organization of the hierarchical relaxation system, we must discuss the procedures for deriving the local constraints from the shape grammar. Let G = (T,N,P,S), let v, w and $x \in V$, let at(v) denote the attachment points of v, and let $av \in$ at(v). We define,

1) v ancestor:av,aw w iff there exists $p \in P$ such that the rewrite rule of p is v := ...w... and there exists an $aw \in$ at(w) such that aw is identified with av in Ga of p. That is, the attachment point av of the left-hand side symbol, v, is associated with endpoint

aw of the right-hand side symbol w. For example, in Fig. 1 the attachment points for the symbol "engine" are associated with the unjoined attachment points of the "engine side" symbols.

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

3) v neighbor:av,aw w iff

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

b) there exists $x \in V$ with $ax \in at(x)$, and there exists $y \in V$ 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.

Using matrix representations for these relations, the descendents and neighbors of a symbol at a particular attachment point can be computed (see Gries [G1] 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 endpoint aw of w and av of v. Given k attachment points per vocabulary symbol, the neighbor:i,j relation (assuming an ordered set of attachment points and i,j are in the range 1 to k) is computed by iterating the following computation n-1 times:

$$\text{neighbor:i,j} = \text{neighbor:i,j} + \sum_{m=1}^{k} \{\text{descendent:i,m} * \sum_{n=1}^{k} (\text{neighbor:m,n} * \text{ancestor:n,j})\}.$$

Semantic constraints can be generated in exactly the same way,i.e., by defining binary relations and compiling their transitive closure. For example, the axes of two symbols are parallel if a production states this explicitly, or if each symbol has an ancestor parallel to itself and these ancestors are explicitly parallel. Such semantic constraints have not yet been incorporated.

The hierarchical relaxation system computes a bottom-up parse of the shape by applying the constraints to a network of low-level hypotheses about pieces of the shape. The processing of this network can be easily described by specifying three simple procedures and two sets which these procedures manipulate.

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 are generated, nodes corresponding to boundary segments sharing an endpoint are linked only if the constraints allow the symbols hypothesized for each node to be adjacent at that endpoint. Building level 0 involves applying the segmentation strategy to the shape to generate the level 0 nodes.

RELAX - since each node corresponds to a single hypothesis, and since nodes are only linked to compatible nodes, the within layer relaxation simply involves removing a node if it has no neighbor at some endpoint.

REDUCE - after BUILD generates level k+1, any level k node which does not support a level k+1 node will be removed. For any level k node which is removed, all level k-1 nodes which support only it are removed. If a level k node is removed, any level k+1 node it supports is removed.

These procedures operate on two sets of nodes, Rx and Rc, both of which are initially empty. When at level k with Rx and Rc empty, BUILD produces the level k+1 hypotheses (or stops if k =n), and puts them into Rx while putting all level k nodes into Rc. RELAX then removes nodes from Rx, taking no action if the node has a neighbor at all endpoints, but otherwise deleting the node from the network and putting its same level neighbors in Rx and its across level neighbors in Rc. REDUCE removes nodes from Rc, taking no action if all 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, REDUCE deletes the node from the network and puts its same level neighbors in Rx and its across level neighbors in Rc. Of course, constraints can be generated from grammars that are not stratified, but the application of the constraints will not prevent the repeated production of symbols which fail to satisfy the constraints, whereas stratification insures a symbol will be built only once.

4.0. HRP vs. Top-down parsing

An important decision facing the designer of a syntactic pattern recognition system is the choice of a parsing mechanism. In this paper, we have proposed a constraint-based, bottom-up parsing mechanism which compiles local contextual constraints from a stratified shape grammar. In this section we would like to discuss two important dimensions, data complexity and model constraint level, which should affect the choice of HRP vs. some other parsing mechanism. A more detailed discussion of these problems, along with results on the comparative complexity of HRP and top-down parsers can be found in Davis and Korner[D2].

The first dimension is what we call data complexity and is meant to reflect the relative difficulty of computing the correct segmentation of the input pattern into the segments needed for the syntax analysis. Clearly, if this segmentation can be done with no errors, then one of the major motivations for adopting HRP is lost. As the reliability of primitive detection decreases, it seems reasonable to assume that the desirability for a parsing mechanism like HRP should increase. However, this depends on yet another factor.

We call the second dimension which determines the choice of parsers the model constraint level. Very simply, it reflects how much the presence of an arbitrary symbol associated with some part of the pattern constrains the possible symbols that can be associated with other, nearby parts of the pattern. Again, if the model has a low

contraint level, then HRP will have only very weak constraints to apply through RELAX, and so, even if the data has high complexity, it might still be advantageous to apply a top-down parsing mechanism. Clearly, if the model has a very high level of constraint, then HRP will be more efficient in analyzing patterns, but then of course, the predictive power of the top-down parser will also increase.

The important question is : Are there areas in this two-dimensional space of data reliability vs. model constraint level where it is clearly advantageous to adopt HRP as a parsing mechanism, and are there areas where it is clearly better to choose a top-down parser? We do not yet have clear-cut answers to this question, but it is obviously a central issue in the evaluation of HRP (or, indeed, of any parsing algorithm).

5.0 Examples

A PASCAL program (4600 lines, 50k) implementing HRP has been written and runs on the DEC-10 computer at the University of Texas. Input to HRP consists of a stratified shape grammar defining the class of shapes to be analyzed and a set of primitives, i.e., line segment descriptions including orientation, length and endpoints. HRP produces a (possibly empty) network of hypotheses relating primitives to the vocabulary symbols at each level of the grammar. Thus, any level n 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 33 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. We do not view parsing as a recognition procedure but rather as a process which imposes organization on the shape (by forming engines, wings, etc.) Recognition is subsequently performed by analyzing the organization.

We will describe the application of HRP to the top view of the airbus in Fig. 2 (traced from You and Fu [Y1]). The split-and-merge algorithm was used to obtain piecewise linear approximations to the boundary of the airbus at several thresholds of goodness of fit. For this shape using three thresholds, a total of 35 primitives were found, of which only 27 were needed to define the shape of the airbus.

Once the primitives are generated, each one must be associated with an initial set of level zero hypotheses. Analysis of the histograms of segment lengths and orientations allows many hypotheses to be rejected, e.g., the very longest segments, if substantially longer than the rest, are not very likely to be wing tips. When the initial set of labels was restricted to only the correct hypotheses, a total of 87 nodes were generated during a complete parse, and HRP required 1 minute and 11 seconds of CPU time to analyze the shape. Table 1 describes the results of runs with 1, 2 and 3 initial hypotheses per primitive. At each level Relax and Reduce eliminated unsupported hypotheses, and Table 1 shows the complete stable network. Run times for these sets of initial hypotheses were 1 minute 11 seconds, 2 minutes 22 seconds, and 4 minutes 51 seconds, respectively.

6.0 Conclusions

HRP has been successfully used to recognize silhouettes of airplanes. A system of programs has been developed which automatically generates syntactic constraints for a given stratified shape grammar and applies them and the grammar to analyze a set of low level hypotheses about a shape. For a more detailed discussion of HRP and its application to shape recognition, see Henderson [H1].

The design and debugging of shape grammars is one major difficulty in using HRP. Errors in the grammar are often confused with errors in HRP itself. Furthermore, there are no strict criteria for a "best" or even a "good" grammar, e.g., no indication of the trade off between the number of symbols in the right-hand side of a rewrite rule vs. the number of levels in the grammar. Automated grammatical inference can certainly be helpful in these respects, but the desirability of decomposing a shape into natural pieces may require an interactive approach.

An area under active investigation is the use of derived semantic constraints which are implicit in the grammar. A set of procedures to compute the semantic constraints will include facilities for describing relative orientation, length and nearness of vocabulary symbols. One would like to know, for example, when it is possible to determine (at any given level of the grammar) the possible relative orientations of the vocabulary symbols. Is it possible to derive across level relations between vocabulary symbols? One possible approach is to construct transitive relations that are known to exist

(by way of the applicability condition and Gs, the semantic generation part of a production) and then take the transitive closure of the relation to determine the implicit relations. For example, parallel is a transitive relation, and if v1 is parallel to v2, and v2 is parallel to v3, where v1, v2 and v3 are vocabulary symbols, then it can be deduced that v1 is parallel to v3. Across level relations can also be generated in this manner since axis orientation is independent of level. Semantic information of this type can be used to aid in hypothesis formation if any specific knowledge is available about the shape to be parsed, for instance, the orientation of the main axis of the plane.

Another goal to be realized is the use and specification of strategy trees to control the application of HRP to the data. A strategy tree will function much as a hierarchical classifier (see Kulkarni and Kanal [K1]). Each non-leaf node of the tree is comprised of three parts. First is a method of generating semantic constraints in the given context, i.e., location in the tree. This will provide a basis for the second part which describes the application of the grammar to the remaining pieces of the shape. The third component is the decision rule to be used in choosing the next node of the strategy tree. A leaf of the tree corresponds to a complete analysis of the shape. Such an approach allows a coarser primary analysis of the data and subsequent directed search for detail in the shape. Another possibility is to search for specific pieces of the shape, e.g., search for wings present in the data. The derived semantic constraints would then play a key role in predicting the size and location of other pieces of the shape. HRP could then take advantage

of this information to direct the search.

References

D1. Davis, L., "Hierarchical Relaxation for Shape Analysis," Pattern Recognition and Image Processing Conference, Chicago, Ill., June 1978, pp. 275-279.

D2. Davis, L. and K. Korner, "Comparative Complexity of Syntactic Parsing Procedures," University of Texas Computer Sciences Dept. TR, [in preparation.]

D3. Davis, L. and A. Rosenfeld, "Hierarchical Relaxation for Waveform Parsing," in Computer Vision Systems, ed. Hanson and Riseman, Academic Press, pp. 101-109, 1978.

F1. Freeman, H. and L. Davis, "A Corner-Finding Algorithm for Chain-Coded Curves," IEEEC, March, 1977, pp. 297-303.

F2. Fu, K.S., "Introduction to Syntactic Pattern Recognition," in Syntactic Pattern Recognition Applications (ed. K.S. Fu), Springer-Verlag, Berlin, 1977, pp. 1-31.

G1. Gallo, V., "A Program for Grammatical Pattern Recognition Based on the Linguistic Method of the Description and Analysis of Geometrical Structures," IJCAI-75, Tbilisi, Georgia, USSR, 1975, pp. 628-634.

G2. Gries, D.,Compiler Construction for Digital Computers, John Wiley, N.Y., 1971.

H1. Henderson, T., Ph.D. Dissertation, U. of Texas, [in preparation].

H2. Horowitz, S., "Peak Recognition in Waveforms," in Syntactic Pattern Recognition Applications (ed. K.S. Fu), Spinger-Verlag, Berlin, 1977, pp.1-31.

K1. Kulkarni, K. and L. Kanal, "An Optimization Approach to Hierarchical Classifier Design,", IJCPR-76, Cal., pp. 459-466.

P1. Pavlidis, T. and S. Horowitz, "Piecewise Approximation of Plane Curves," Pattern Recognition, 1973, pp.346-405.

S1. Stockman, G. "A Problem-Reduction Approach to the Linguistic Analysis of Waveforms," U. of Maryland, TR-538, May 1977.

V1. Vámos, T. and Z. Vassy, "Industrial Pattern Recognition Experiment - A Syntax Aided Approach," IJCPR-73, Washington, pp. 445-452.

Y1. You, K. 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, March 1977, pp.72-83.

Level generated	0	1	2	3	4	5	6	Total	Time
A. 1.0 hyp/prim									
New hypotheses	35	27	23	15	8	2	1	111	1:11
After Relax and Reduce									
0	31	31	31	27	27	27	27		
1		27	27	23	23	23	23		
2			23	19	19	19	19		
3				9	9	9	9		
4					6	6	6		
5						2	2		
6							1		
B. 2.3 hyp/prim									
New hypotheses	81	55	48	33	8	2	1	228	2:22
After Relax and Reduce									
0	59	59	44	27	27	27	27		
1		55	40	23	23	23	23		
2			36	19	19	19	19		
3				9	9	9	9		
4					6	6	6		
5						2	2		
6							1		
C. 3.2 hyp/prim									
New hypotheses	114	74	58	32	9	2	1	290	4:51
After Relax and Reduce									
0	80	75	50	29	27	27	27		
1		71	46	25	23	23	23		
2			42	21	19	19	19		
3				11	9	9	9		
4					6	6	6		
5						2	2		
6							1		

Table 1 - Hypothesis generation and deletion (by level)

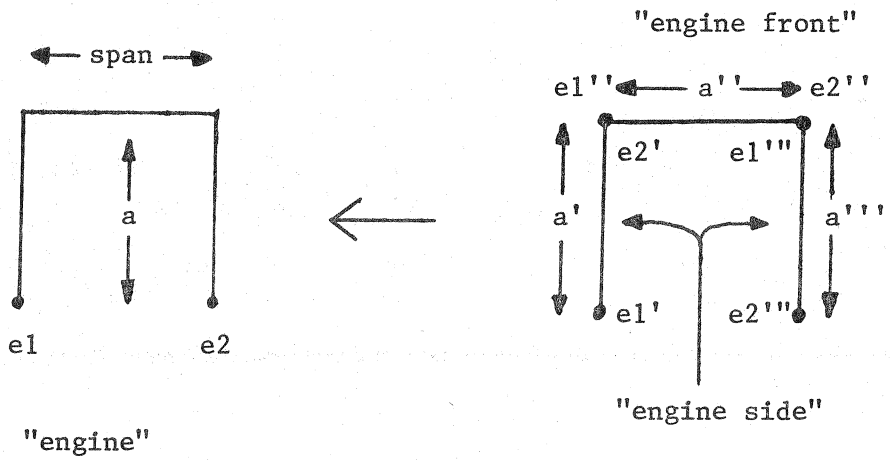

Figure 1 - Example of a production

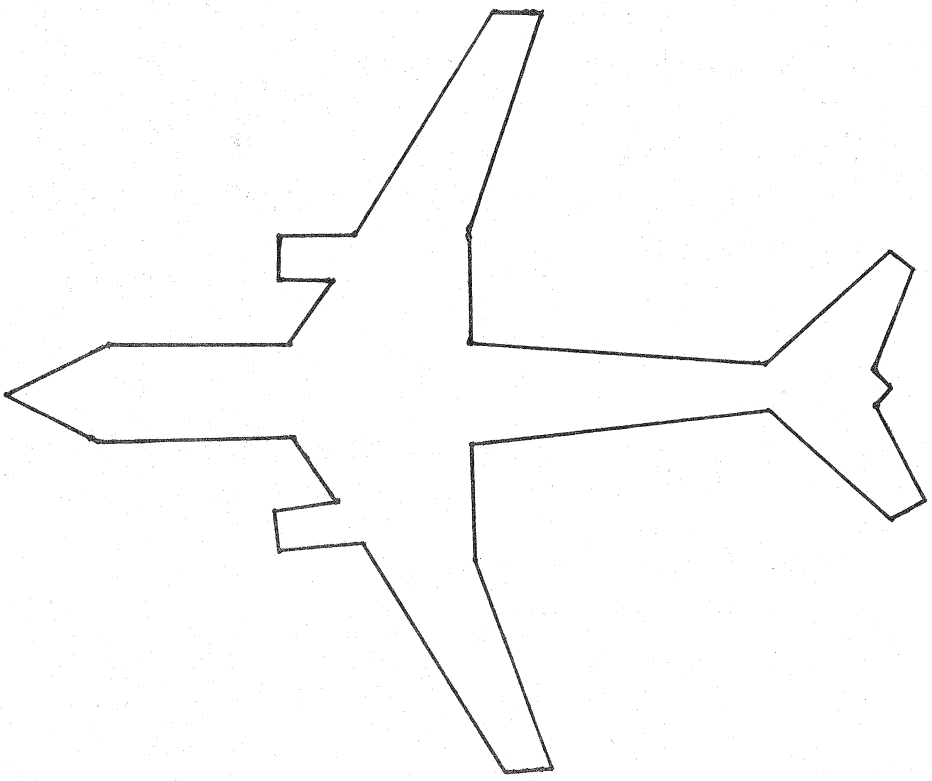

Figure 2 - Airbus