Pattern Recognition Pergamon Press 1974. Vol. 6, pp. 189–199. Printed in Great Britain

A SYNTAX-DIRECTED PROGRAM THAT PERFORMS A THREE-DIMENSIONAL PERCEPTUAL TASK*

JAMES GIPS

260 South Sycamore Avenue, Los Angeles, California 90036, U.S.A.

(*Received* 14 *February* 1974)

Abstract—A syntax-directed program that performs a three-dimensional perceptual task is described. The task, in a slightly simpler form, was used originally in a psychological study of mental rotation.[1] The task consists of determining whether two line drawings portray (different views of) identical objects, mirror image objects, or structurally different objects, where the objects are composed of linear strings of attached cubes. The program is syntax-directed in the sense that it uses a fixed set of syntactic rules to analyze the line drawings. This is the first use of formal syntactic techniques in the analysis of pictures (in this case, line drawings) of three-dimensional objects.

Syntactic pattern recognition Shape grammars Graph grammars Picture grammars
Scene analysis Perception Mental rotation

INTRODUCTION

This paper describes a program that performs a task originally administered to human subjects in a perceptual psychology study.[1] The task involves the analysis and comparison of line drawings portraying three-dimensional objects.

The program is syntax-directed in the sense that the program uses a fixed set of syntactic rules to analyze the line drawings. These rules originally were formulated[2] in terms of shape grammars[2,3] but could be expressed, with minor alterations, in terms of other grammar formalisms, including web grammars,[4,5] graph grammars,[6] plex grammars[7] and picture description grammars.[8] Formal syntactic techniques have been widely used in pattern recognition to analyze pictures of a predominantly two-dimensional nature. This paper reports the first use of formal syntactic techniques in the analysis of pictures (in this case, line drawings) of three-dimensional objects.

THE TASK

The perceptual task performed by the program was developed by Roger Shepard and Jacqueline Metzler of the Department of Psychology at Stanford and is reported in Ref. (1). The task can be described as follows: given a pair of perspective line drawings such as those in Figs. 1(a–c), determine whether the drawings portray objects that are identical to each other in terms of three-dimensional shape (i.e. the drawings can be considered different views of the same object) or whether the drawings portray objects that are three-dimensional mirror-images of each other. The line drawings in Figs. 1(a, b) portray different views of the same objects. The line drawings in Fig. 1(c) portray objects that are three-dimensional mirror images of each other.

Shepard and Metzler administered this task to human subjects and recorded the amount of time required by the subjects to decide whether the portrayed objects were "the same" or "mirror image". Two kinds of "same" pairs were used: the drawings portrayed two views of the same object either rotated in the picture plane (as in Fig. 1a) or rotated in depth (as in Fig. 1b). Their (surprising) results are summarized as follows:

> The time required to recognize that two perspective drawings portray objects of the same three-dimensional shape is found to be (i) a linearly increasing function of the angular difference in the portrayed orientations of the two objects and (ii) no shorter for differences corresponding simply to a rigid rotation of one of the two-dimensional drawings in its own picture plane than for differences corresponding to a rotation of the three-dimensional objects in depth.[1]

These results seem to confirm the very subjective notion of solving the task by mentally rotating one of the portrayed three-dimensional objects at a fixed rate to determine if it matches the other portrayed object. The results suggest that it is just as easy to mentally rotate an object in depth as it is in the picture plane

* This research was performed at the Stanford Artificial Intelligence Laboratory and was supported in part by the Advanced Research Projects Agency of the Department of Defense under Contract No. SD-183.

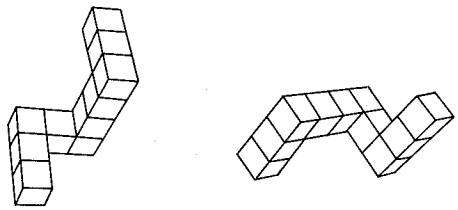


Fig. 1(a). Line drawings portraying identical objects rotated in picture plane.

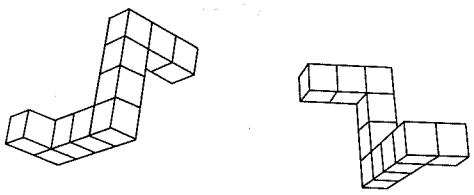


Fig. 1(b). Line drawings portraying identical objects rotated in depth.

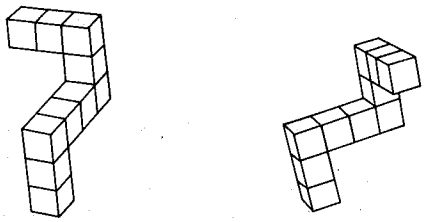


Fig. 1(c). Line drawings portraying mirror image objects.

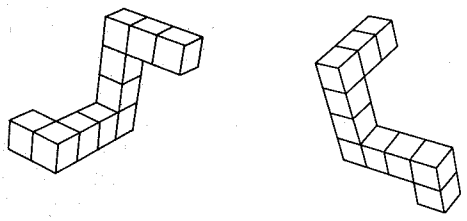


Fig. 1(d). Line drawings portraying structurally different objects.

and that these rotations can be done at roughly 60°/sec.[1]

The task was expanded slightly for the computer program. The program is required to determine whether the pairs of drawings portray: (1) identical objects (as in Figs. 1a and b); (2) mirror images (as in Fig. 1c); or (3) structurally different objects (as in Fig. 1d). If the pairs are reported as identical or mirror image, the program also gives the three-dimensional axis equivalences. Shepard and Metzler always used line drawings portraying objects consisting of strings of exactly ten cubes; there is no restriction on the number of cubes for the program. The program successfully analyzed the pair of line drawings shown in Fig. 2 as well as those in Fig. 1. It should be noted at the outset that there was no intention of simulating the processes

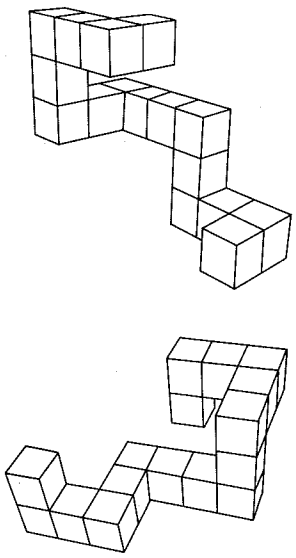


Fig. 2. A pair of more complicated line drawings successfully analyzed and compared by the program.

used by people in solving this task. The program was written to explore the uses of syntax-directed methods in analyzing pictures of three-dimensional objects.

PROGRAM OVERVIEW

The program is written in SAIL,[9] an extended ALGOL, and runs on the Stanford Artificial Intelligence Laboratory PDP-10. Input to the program are two files prepared using Geomed,[10] a geometric editor. Each file is a specification of a perspective line drawing of an object. Namely, each file is a list of the two-dimensional coordinates of the endpoints of each line segment occurring in the line drawing. There are four possible outputs for the program: (1) a statement that the objects portrayed by the line drawings are identical and an indication of the three-dimensional axis equivalences of the objects; (2) a statement that the objects portrayed by the line drawings are mirror images and an indication of the three-dimensional axis equivalences of the objects; (3) a statement that the objects portrayed by the line drawings are completely different; or (4) a statement indicating the failure of the program to successfully analyze one of the line drawings.

The program has three parts: (1) preprocessing, in which the vertices of the line drawing are classified, a data structure is constructed, and the three-dimensional axes are determined; (2) analysis and model building, in which the syntactic rules are applied to the line drawing and a model of the object is constructed; and (3) comparison of models, in which the models

constructed for the two objects are compared. The first two parts are applied to each line drawing independently, the idea being to construct a model of the three-dimensional structure of each portrayed object. In the third part the two models are compared to determine the relationship between the portrayed objects.

The process of analysis and model building in the program is the process of extracting the three-dimensional structure of the portrayed object from the two-dimensional line drawing. The model is constructed during application of the syntactic rules to the line drawing. Each time a rule is applied, something new is added to the model. This process can be likened to the syntactic analysis component of some compilers. The preprocessing is similar to the lexical scan. Applying the rules to a line drawing is similar to parsing a symbol string using a phrase structure grammar. The model constructed for the portrayed object is not dissimilar to the tree representation that might be constructed by a compiler for an arithmetic expression. Just as the tree embodies the structure of the expression, the model embodies the structure of the object.

In the next three sections, a detailed description of the program is given.

PREPROCESSING

The preprocessing stage of the program includes building the data structure, classifying the vertices, and determining the three-dimensional axes.

The LEAP associative data structures[11] of SAIL are used to represent the line drawing in the program. Associations have the general form "Attribute ⊗ Object ≡ Value", which conventionally is read as "Attribute of Object is Value". The three elements of an association are called items. For each line segment of the line drawing, two associations of the form

$$\text{ENDPOINT} \otimes L_i \equiv V_j$$

and

$$\text{ENDPOINT} \otimes L_i \equiv V_k$$

are added where L_i is an item representing the line segment and V_j and V_k are items representing the vertices of the line segment. Each vertex has a one-dimensional array of length two as a datum. The array contains the x and y coordinates of the vertex in the line drawing. As the associations are added, the coordinates of the vertices are compared with the coordinates of previously added vertices. Vertices that are located within a certain threshold distance are considered identical and the data structure is constructed accordingly. A discussion of the use of LEAP associations in representing scenes of polyhedral objects is given in Ref. (12).

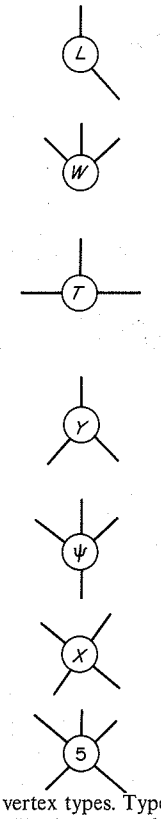


Fig. 3. Definition of vertex types. Type L—a vertex where two lines meet; type W—A vertex where three lines meet with one angle $>180°$; type T—a vertex where three lines meet, one pair colinear (see Fig. 4 for further classification); type Y—a vertex where three lines meet with all angles $<180°$; type ψ—a vertex where four lines meet, one pair colinear; type X—a vertex where four lines meet, two pairs colinear; type 5—a vertex where five lines meet.

After the associations for each line segment are added, each vertex is classified in terms of the number of lines that meet at the vertex and the angles between the lines. Vertex classification roughly follows that of Guzman.[13] Seven vertex types are allowed: L, W, T, Y, ψ, X, and 5. The definition of vertex types is given in Fig. 3. Vertices of type T are further classified as either $T1$ or $T3$ using local information. If a T vertex is contained in a parallelogram of vertices (see Fig. 4)

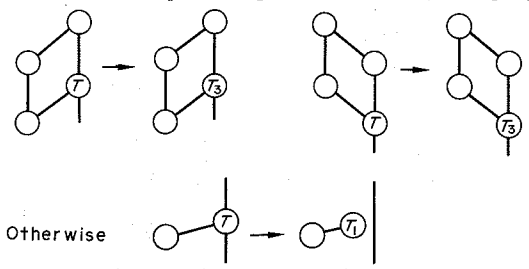


Fig. 4. Classification of type T vertex into type $T3$ or type $T1$.

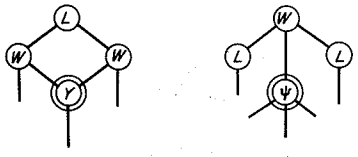


Fig. 5. The two allowable vertex configurations for the initial double-circled vertex.

it is classified as type $T3$ and is still considered a vertex of three lines. If a T vertex is not contained in a parallelogram of vertices, it is classified as type $T1$, considered a vertex of one line (see Fig. 4), and the ENDPOINT associations are changed accordingly. For each vertex, an association of the form

$$\text{VTYPE} \otimes V_i \equiv \text{vertex\_type}$$

is added, where V_i is the item for the vertex and vertex_type is the item for its vertex type.

One of the vertices of the line drawing is distinguished by surrounding it with an extra circle. Intuitively, this vertex is the central vertex of one of the end

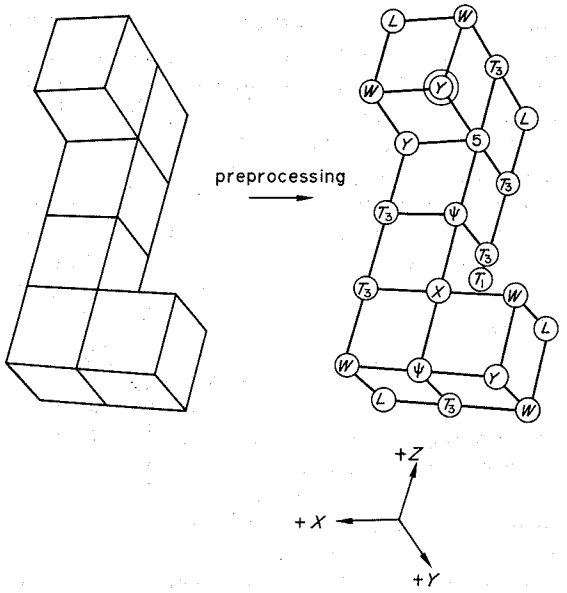


Fig. 6. The effect of preprocessing on a simple line drawing.

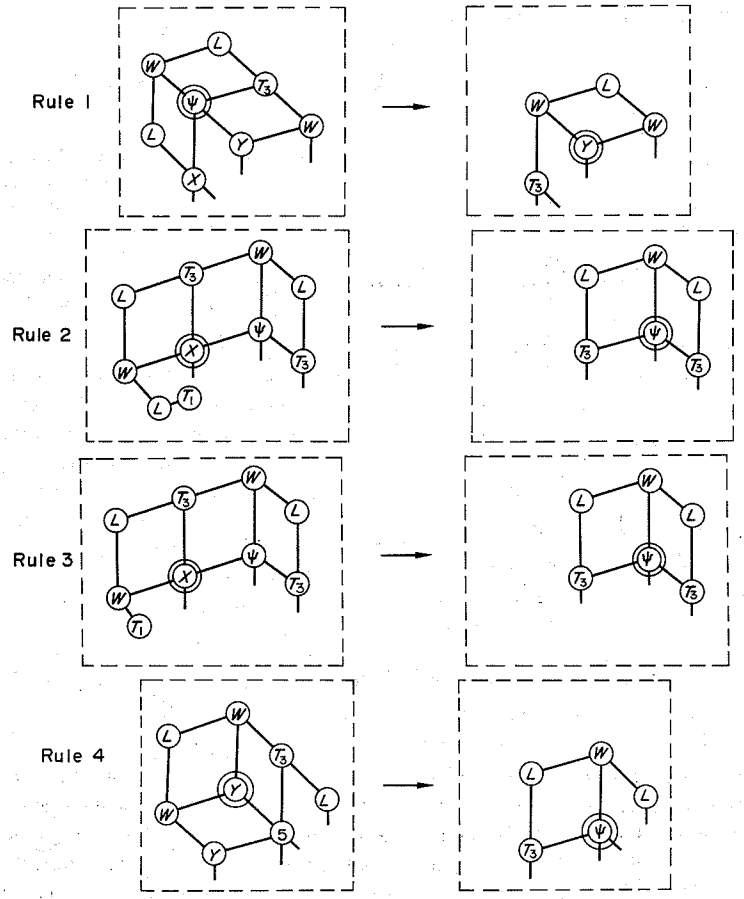


Fig. 7. The syntactical rules used to analyse the line drawings.

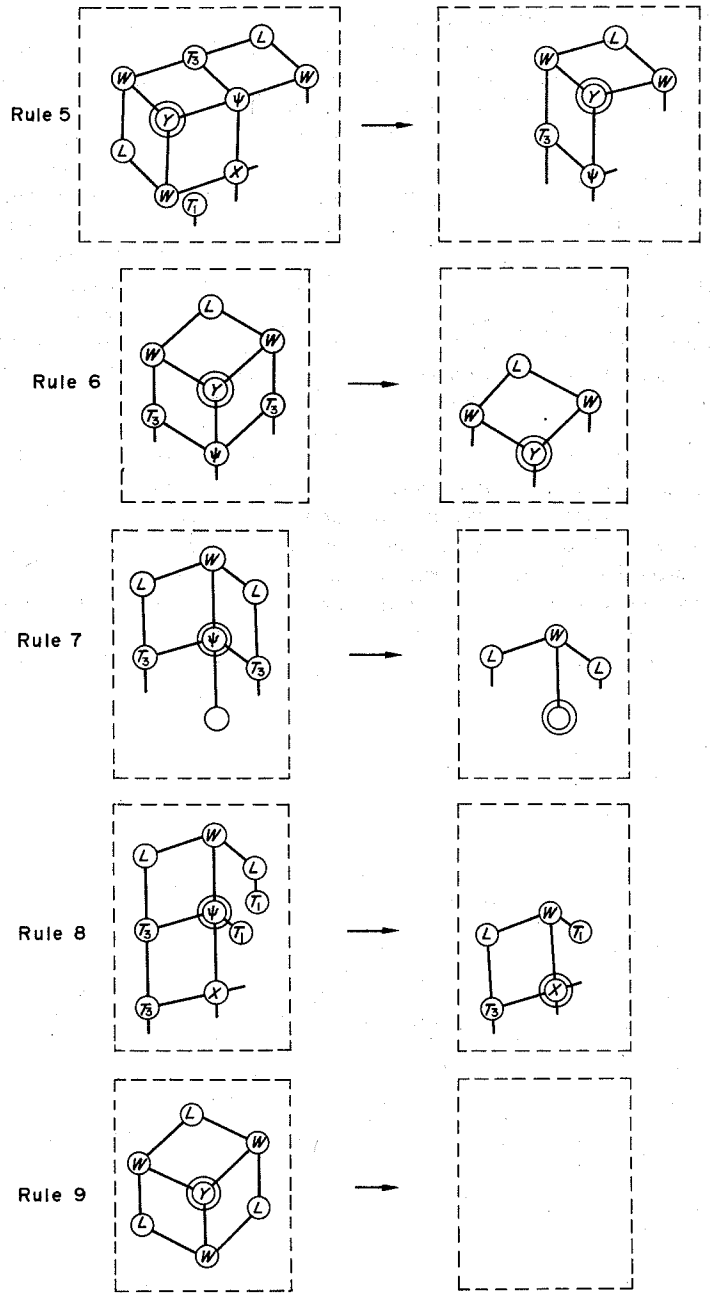


Fig. 7. Cont.

cubes of the object. The vertex is either of type Y or type ψ. The possible vertex configurations for this distinguished vertex are shown in Fig. 5. If, as normally occurs, two different vertices of the line drawing qualify, i.e. one on each end cube, one of these vertices is chosen arbitrarily.

After the vertices are classified, a three-dimensional axis system for the line drawing is determined using the distinguished vertex. If this vertex is of type Y, the orientations of the three line segments radiating from the vertex are calculated. If the vertex is of type ψ, the orientations of the three line segments that would form

a type Y vertex are calculated. These three orientations are assigned directions $+x$, $+y$ and $+z$ in a counterclockwise order. This insures that the axis system is a left handed system. The axis system will be used in specifying the model constructed for the portrayed object.

The effect of preprocessing on a line drawing portraying an object composed of six cubes is shown in Fig. 6.

ANALYSIS AND MODEL BUILDING

The heart of the program is the part that applies the syntactic rules to the line drawing with classified vertices and constructs the model.

The nine rules used to analyze the line drawings are shown in Fig. 7. The left sides of the rules contain different vertex configurations that can portray an end cube. These different vertex configurations result from the various possible three-dimensional structures of the objects and the various viewpoints from which these objects can be seen. The right sides of the rules are the vertex configurations that result from the removal of the end cube.

The analysis process begins with the line drawing (with classified vertices and double-circled vertex) from preprocessing, and consists of the repeated application of the rules to the line drawing until no rules are applicable. A rule is applicable to a line drawing only if the line drawing contains a vertex configuration identical to the left side of the rule. The application of the rule results in the substitution of the right side of the rule for the part of the line drawing that was identical to the left side of the rule. In the application of the rules, the exact angles between the lines at each vertex are ignored within the bounds of the definition of the vertex type. All that matters is the vertex configurations. The transformations of rotation, scale, translation, and mirror image may be applied to the left and right sides of the rules during rule application.

During the process of rule application, the line drawing is erased one cube at a time. In the preprocessing stage, one of the vertices of one of the end cubes was marked with an extra circle. The left side of each of the rules contains a double-circled vertex. Thus each rule is only applicable to the portion of the line drawing containing the double-circled vertex. Each application of a rule results in the elimination of the vertex

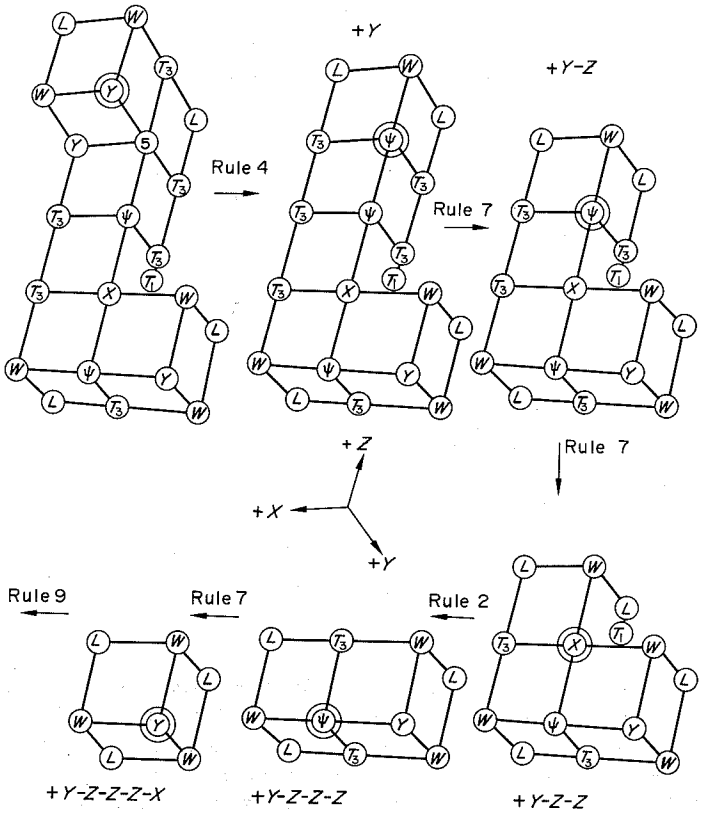


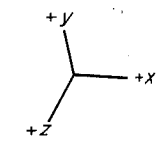
Fig. 8. Rule application and model construction for the line drawing of Fig. 6.

configuration which portrays the end cube with a double-circled vertex, the placement of an extra circle around a vertex of the next cube, and the filling in of edges and vertices which were obscured by the erased cube.

The rules are embodied in SAIL code in the program. First the code establishes which rule is applicable by determining which vertex configuration of the left side of a rule exists as specified by the current associations. The rule is applied by changing the associations to match the right side of the rule. This may result in the addition, as well as the deletion, of line segment and vertex items and associations. This section of code is repeated until no rules are applicable. If at this point, no line segment or vertex items and associations remain, the program has succeeded in analyzing the line drawing. If no rules are applicable but some items and associations remain, the program has failed.

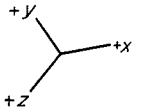
If this part of the program consisted solely of an implementation of the analysis rules, the program could be used only as an acceptor. Namely, the program could make a binary choice: it could determine whether or not a line drawing does indeed portray a Shepard–Metzler object (as defined by rules of Fig. 7) by classifying the vertices and checking if the rules could be applied until nothing remained. But the task involves extracting the three-dimensional structure of the objects portrayed by the line drawings. To do this, a model of the portrayed object is constructed during rule application.

The model used consists of a one-dimensional array of characters which specifies the skeleton of the portrayed object. The skeleton can be thought of as the sequence of line segments connecting the centers of the cubes. The model consists of the sequence of directions of the skeleton, where the directions are the axis directions determined during preprocessing. There are six possible directions: $+x$, $-x$, $+y$, $-y$, $+z$, $-z$. The



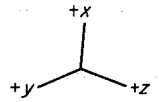
I(a) left

-y-y+z+z+z-x-x-y-y



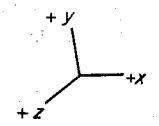
I(a) right

+z+z+y+y-x-x-x+z+z



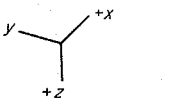
I(b) left

-z-z-x-x-x+y+y+y-z



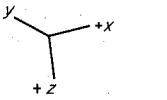
I(b) right

+x+x-y-y-y-z-z-z+x



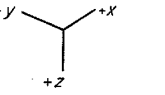
I(c) left

-y-y+z+z-x-x-x+z+z



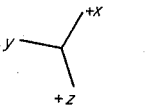
I(c) right

+y+y+z+z-x-x-x+z+z



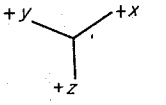
I(d) left

+y+y+z+z+z-x-x-x+y



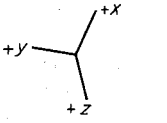
I(d) right

-x-x+z+z+z-y-y-y+z



2 top

-x+y+y+z+z+x+x-y-y-y+z+z-y-y-x



2 bottom

+z-y-y+x+x-y-y-y-z-z+x+x+y+y+z

Fig. 9. The three-dimensional axes and models constructed for the line drawings of Figs. 1 and 2.

model is constructed by recording the direction of the extra circle. Each time a rule is applied (except for the final application of rule 9) the axis direction closest to the direction of motion of the extra circle is added to the model. The length of the model is one less than the number of cubes in the portrayed object.

An example of this process is shown in Fig. 8 for the line drawing of Fig. 6. The model constructed for the object portrayed by this line drawing is $+y\ -z\ -z\ -z\ -x$. The models constructed for the objects portrayed by the line drawings shown in Figs. 1 and 2 are shown in Fig. 9. In the construction of the models in this figure, the convention that the uppermost of two eligible vertices is distinguished as the initial double-circled vertex is used in preprocessing.

COMPARISON OF MODELS

After the preprocessing and model building routines are applied to each line drawing independently, the models that were constructed for the portrayed objects are compared. The model comparison has two stages.

The first stage determines whether the two models represent similar (i.e. identical or mirror image) objects or completely different objects. Recall that each model is a one-dimensional array of axis directions where the possible directions are $+x$, $-x$, $+y$, $-y$, $+z$ and $-z$. First, the lengths of the two models are compared. If the lengths are unequal, the objects are declared to be different and the model comparison routine is terminated. If the lengths are equal, the program attempts to find axis equivalences for the $+x$, $+y$ and $+z$ directions of the first model.

An axis direction d_1 of the first model is considered equivalent to an axis direction d_2 of the second model (written $d_1 \equiv d_2$) if

(1) the positions in which d_1 occurs in the first model are identical to the positions in which d_2 occurs in the second model and

(2) the positions in which $-d_1$ occurs in the first model are identical to the positions in which $-d_2$ occurs in the second model.

For example, in the models shown in Fig. 9 derived for the line drawings of Fig. 1(b), the axis equivalences for the $+x$, $+y$ and $+z$ directions of the first model are $+x \equiv +y$, $+y \equiv -z$ and $+z \equiv -x$. Note that by the definition of axis equivalence, if $d_1 \equiv d_2$ then $-d_1 \equiv -d_2$.

If an equivalence can be found for each of the axis directions then the portrayed objects must be similar (identical or mirror image) and the second stage of the model comparison routine is entered. If an equivalence for each of the directions cannot be found, no final conclusions can be drawn. The objects still could be similar if the initial double-circled vertices were located in preprocessing at different ends of the objects. This occurred, for example, with the construction of the models in Fig. 9 for the line drawings in Fig. 1(a). To determine if the portrayed objects are similar or different, the second model is reversed and axis equivalences for the $+x$, $+y$ and $+z$ directions of the first model again are searched for. To reverse the second model, each entry in the model is first negated and the sequence of directions is reversed. For example, the reverse of the model $+z+z+y+y-x\ -x-x+z+z$ for the right line drawing of Fig. 1(a) (see Fig. 9) is $-z\ -z\ +x\ +x\ +x\ -y\ -y\ -z\ -z$. This new model is the model that would have been constructed if the initial double-circled vertex had been located at the other (lower) end of the object and the model had been constructed in reverse order. If, again, an axis equivalence cannot be found for each of the directions of the first model, the objects are declared different and the model comparison routine is terminated. If equivalences are found, the objects must be either identical or mirror image and the second stage is entered.

The second stage of the model comparison determines whether models with axis equivalences portray identical or mirror image objects. In the program $+1$

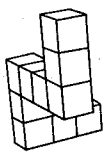


Fig. 10(a). A view in which a substantive part is obscured.

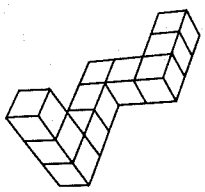


Fig. 10(b). A view in which two different lines and two different vertices appear to coincide.

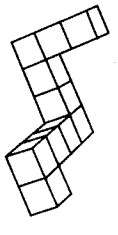


Fig. 10(c). A degenerate view.

Fig. 10. Three line drawings which the program cannot analyze.

represents the $+x$ direction, -1 represents $-x$, $+2$ represents $+y$, -2 represents $-y$, $+3$ represents $+z$ and -3 represents $-z$. Integer array EQUIV[1:3] is defined. The value of EQUIV[1] is the axis equivalence of the $+x$ direction of the first model, etc. By the definition of axis equivalence, the array EQUIV can have $6*4*2 = 48$ different permutations of assigned values. For the models of Fig. 1(b), EQUIV[1] = +2, EQUIV[2] = −3, and EQUIV[3] = −1. Half of the 48 permutations represent axis equivalences that would occur if the second line drawing portrayed simply a three-dimensional rotation of the object portrayed by the first line drawing. The other half of the 48 permutations occur if the object portrayed by the second line drawing is a mirror image of the object portrayed by the first. The following SAIL subroutine determines whether the line drawings portray objects which are identical or mirror image:

```
INTEGER PROCEDURE SAME_OR_MI (INTEGER ARRAY EQUIV);
  BEGIN
  INTEGER I, MINUSAXES, DIFFAXES;
  MINUSAXES ← DIFFAXES ← 0;
  FOR I ← 1, 2, 3 DO
    BEGIN
    IF ABS(EQUIV[I]) ≠ I THEN DIFFAXES ← DIFFAXES + 1;
    IF EQUIV[I] < 0 THEN MINUSAXES ← MINUSAXES + 1;
    END;
  IF (MINUSAXES = 1) v (MINUSAXES = 3)
  THEN RETURN (IF DIFFAXES = 2 THEN SAME ELSE MI)
  ELSE RETURN (IF DIFFAXES = 2 THEN MI ELSE SAME);
  END;
```

The subroutine determines whether the axis equivalences can be effected with just a three-dimensional rotation of the first object or whether an axis inversion (mirror image) is required. Recall that the method for determining the axis directions in preprocessing insures that both axis systems are left handed. The subroutine determines whether after transforming the first model into the second model the resulting axis system would be left handed or right handed. If the axis system would still be left handed, the line drawings portray identical objects from possibly different viewpoints. If the axis system would become right handed, the line drawings portray objects that are mirror images of each other.

PROGRAM RESULTS

The range of line drawings that the program can analyze is defined effectively by the rules of Fig. 7. The program has successfully compared over a hundred pairs of line drawings of objects.

It may be of interest to compare the performance of the program with the performance of people on the same task.

The program cannot analyze many line drawings that are analyzable by people. In particular, there are three types of views of an object for which the program fails: (1) a view of the object in which a substantive part of the object is obscured (as in Fig. 10a); (2) a view of the object in which two different lines meet and appear to be one, or two different vertices appear to coincide (as in Fig. 10b); and (3) a degenerate view of the object in which only one face of a particular cube is observable (as in Fig. 10c). Some of these problems could be fixed by not too complicated program patches. In particular, most degenerate views (of type 3) probably could be analyzed successfully with the addition of a few rules.

The program succeeds in analyzing some line drawings which are very difficult for people, namely line drawings of objects containing many cubes, such as the line drawings in Fig. 2. There is essentially no limit to the number of cubes in a line drawing that can be analyzed by the program. As Shepard and Metzler only used line drawings of objects containing ten cubes, no information was collected as to the limit of the number of cubes in line drawings analyzable by people. Personally, the solution of the task for the pair of line drawings in Fig. 2 seems very much more difficult than the solution for the pairs in Fig. 1. Mentally solving the task for line drawings portraying twice as many cubes as those portrayed in Fig. 2 seems out of the question for me using the normal, non-analytic, mental rotation technique.

Shepard and Metzler were interested in the amount of time needed to solve the task. As noted previously, the amount of time required by people varies linearly with the angular difference of the portrayed

orientations of the two objects. The amount of time taken by the program is independent of the angular difference of the portrayed orientations and is relatively constant for line drawings of objects of a fixed number of cubes. Indeed, where people would immediately recognize that two identical line drawings portray identical objects, the program would take about the same amount of time to solve this particular task as any other.

The input format for the program is different than that used for people. People are shown the line drawings. The program is given a digitized encoding of the line drawing made in terms of the coordinates of the endpoints of the line segments. A front end program could have been written that made use of a television camera and a line finder algorithm. It is interesting to note that the program would take substantially longer to solve the task if shown the actual line drawing rather than given the encoded version while a person would certainly take longer to solve the task given the encoding rather than shown the line drawing itself. Shown the line drawing, the program would first encode it in terms of the coordinates of the vertices. Given the encoding, a person would first draw the picture.

All of this discussion must be tempered by the obvious fact that the program can solve only this one perceptual task while a person is a very general perceptual system.

LIMITATIONS AND POSSIBLE EXTENSIONS

The task domain of the program was carefully chosen. The task has some important restrictions that facilitate its solution. For example: The program is given encodings of perfect line drawings. Each line drawing portrays exactly one object. The range of objects the line drawings can portray is limited to objects composed of linear strings of cubes. The task can be solved using only structural information extractable from the line drawings.

A first extension to the program might be to add rules that would allow cubes to be attached to more than two other cubes. The objects to be analyzed would no longer be restricted to strings of cubes but could be arbitrary assemblies of attached cubes. A one-dimensional array of directions would no longer suffice as a model of an object. The new model constructed might be a connected graph where the nodes would represent the centers of the cubes and the arcs of the graph would indicate adjacency or attachment of the cubes and be labelled as to the axis direction between the cubes. This model would still be a representation of the skeleton of the object. The model used in the program could be considered a special case of this connected graph model. More generally, the objects could be composed of not just cubes but a variety of primitive objects, e.g. wedges, prisms, octahedrons. The models constructed would be connected graphs as above but with each node labelled to indicate the type of object represented. Using this scheme, the process of determing whether two pictures portray different views of identical objects would consist of analyzing the pictures using the rules, constructing the graph models, and determining whether the two graphs represent skeletons of identical objects. To help insure that shape rules are not applicable in unintended situations, a labelling technique such as the one described by Waltz[14] might be used in preprocessing to label the portrayed edges of the object as concave, convex, etc. The lines between vertices in the rules would be labelled accordingly. Perhaps, lines between vertices could be allowed to be curves as well as straight lines. How large a class of objects can be analyzed using this type of technique is an open question.

The major advantage of this technique is that line drawings portraying a large number of objects can be analyzed, including line drawings portraying objects which may never have been previously seen, using a fixed set of rules. Instead of requiring an object to be analyzed to be a member of a small, fixed class of objects, it is only required that the object be decomposable into instances of a fixed set of primitive objects attached in certain ways.

SUMMARY

This paper describes a syntax-directed program that performs a three-dimensional perceptual task. The task, in a slightly simpler form, was used originally in a psychological study of mental rotation.[1] The task for the program consists of determining whether two line drawings portray (different views of) identical objects, mirror image objects, or structurally different objects, where the objects are composed of linear strings of attached cubes. No attempt was made to simulate the processes used by people in solving this task.

The program is written in SAIL, an extended ALGOL. Input to the program are two files prepared using a geometric editor. Each file is a specification of a perspective line drawing of an object. Namely, each file is a list of the two-dimensional coordinates of the endpoints of each line segment occurring in the line drawing. The output of the program indicates whether the portrayed objects are: (1) identical; (2) mirror images; or (3) structurally different; or (4) the output of the program could indicate that the program failed to

analyze one of the line drawings. In cases (1) and (2) the program also gives the three-dimensional axis equivalences of the portrayed objects.

The program is syntax-directed in the sense that it uses a fixed set of syntactic rules to analyze the line drawings. The program has three parts. The first two operate on each line drawing independently. The first part is preprocessing, in which a data structure is constructed, the vertices of the line drawing are classified, and a three-dimensional axis system is determined. The second part, analysis and model building, is the heart of the program. In the second part, nine syntactic rules are applied repeatedly to the line drawing with classified vertices and a model of the structure of the portrayed object is constructed. The syntactic rules can be expressed, with minor alterations, in several of the formalisms for picture grammars that have been developed. The effect of applying each of the rules is to erase the vertices and lines of one of the end cubes of the portrayed object. During the process of rule application, a model of the three-dimensional structure of the portrayed object is constructed. The model consists of a one-dimensional array of axis directions and can be considered a representation of the skeleton of the portrayed object. In the third part, the models that have been constructed are compared to determine whether the line drawings portray identical, mirror image or structurally different objects.

In the final two sections of the paper, the performance of the program is compared to the performance of people in solving the task, the limitations of the program are discussed, and some possible extensions are suggested.

Formal syntactic techniques have been widely used to analyze pictures of a predominantly two-dimensional nature. This is the first use of this type of technique in the analysis of pictures (in this case, line drawings) of three-dimensional objects. The major advantage of the technique used in the program is that line drawings portraying a large number of objects can be analyzed, including line drawings portraying objects which may never have been previously seen, using a fixed set of rules.

REFERENCES

1. R. Shepard and J. Metzler, Mental rotation of three-dimensional objects. *Science* **171,** 701 (1971).
2. J. Gips, Shape grammars and their uses, Ph.D. Dissertation, Stanford Computer Science Report, No. 413 (1974).
3. G. Stiny and J. Gips, Shape grammars and the generative specification of painting and sculpture, *Proc. IFIP Congress* 71. North Holland, Amsterdam (1972); also in *The Best Computer Papers of* 1971. O. Petrocelli ed., Auerbach (1972).
4. J. Pfaltz and A. Rosenfeld, Web grammars, *Proc. 1st Int. Joint Conf. on Artificial Intelligence*, Washington, D.C., p. 609 (1969).
5. J. Pfaltz, Web grammars and picture description. *Computer Graphics and Image Processing* **1,** 193 (1972).
6. T. Pavlidis, Linear and context-free graph grammars. *J. ACM* **19,** 11 (1972).
7. J. Feder, Plex languages. *Inf. Sci.* **3,** 225 (1971).
8. A. Shaw, Parsing of graph-representable pictures. *J. ACM* **17,** 453 (1970).
9. K. Vanlehn (ed.), SAIL user manual, Stanford Computer Science Report, No. 373 (1973).
10. B. Baumgart, Winged edge polyhedron representation, Stanford Computer Science Report, No. 320 (1972).
11. J. Feldman and P. Rovner, An Algol-based associative language, *Commun. ACM* **12,** 439 (1969).
12. R. Paul, G. Falk and J. Feldman, The computer representation of simply described scenes, M. Faiman and J. Nievergelt eds., *Pertinent Concepts in Computer Graphics*. University of Illinois Press, Chicago (1969).
13. A. Guzman, Decomposition of a visual scene into three-dimensional bodies, *Proc. FJCC* **33,** 299 (1968).
14. D. Waltz, Generating semantic descriptions from drawings of scenes with shadows, M.I.T. A.I. Memo *TR*-271 (1972).