Rule-based geometrical reasoning for the interpretation of line drawings

Elizabeth T. Whitaker

Center for Machine Intelligence, University of South Carolina,
Columbia, SC 29208

and

Michael N. Huhns

Microelectronics and Computer Technology Corporation,
9430 Research Blvd., Austin, TX 78759

Abstract

The design of a knowledge-based system for inferring the three-dimensional structure of an object based only on three orthogonal views of it is presented. The three views are line drawings obtained from digitized two-dimensional images of the front, top, and one side of the object. The system attempts to make the same deductions as would be made by a draftsman observing these line drawings. Although the information in only three views is insufficient to uniquely characterize an object, a draftsman is still able to infer a reasonable three-dimensional interpretation by making appropriate assumptions about hidden surfaces and using intuition about the structure of objects in three-space. The system incorporates similar intuitive knowledge and reasoning techniques which enable it to make the same assumptions. The knowledge which is used by the system is encoded as productions in the rule-based language OPS5. The system successfully produces a correct three-dimensional description for many simple polyhedral objects. It can be easily expanded to include new polyhedral objects simply by adding new production rules.

I. Introduction

The problem discussed here is the construction of an expert system to produce the three-dimensional representation of a polyhedral object using information about lines and vertices obtained from line drawings of three orthogonal, two-dimensional views of the object. The line drawings are derived from orthographic projections of the object. These are produced by projecting points on the object into the plane of projection along lines which are perpendicular to the plane. For simplicity, the objects are limited to polyhedra, so that the line drawings and the resultant three-dimensional description consist solely of straight line segments. It is assumed that the three views contain sufficient information to construct this three-dimensional description.

An attempt has been made to capture the reasoning techniques used by humans in understanding the shape of an object when given the top, front, and right side views. The information from the three views is combined to generate lines in three dimensions.

A. Understanding line drawings of polyhedra

There have been several attempts to construct a system that can automatically generate an interpretation of a line drawing [1,2,3]. Waltz [1], using labelling methods similar to those introduced by Huffman and Clowes [4], developed a technique for interpreting line drawings of polyhedral scenes. Mackworth [5] described a method for producing an interpretation of a line drawing of a polyhedral scene by using physical rules which govern the relationships among the edges and surfaces in such a scene. Sugihara [6,7] discussed the mathematical conditions under which a line drawing represents a valid polyhedral scene.

B. Manual blueprint reading

An orthographic projection of an object is a drawing produced by projecting points on the object into the plane of projection along lines which are perpendicular to the plane. Each line in a drawing represents a change in the direction of a surface, but a second view must be given in order to tell what kind of change it is. A line can be an edge view, the meeting of two surfaces, or the limit of a curved nonpolyhedral surface as shown in Figures 1 and 2. In the drawing of a polyhedron, only the first two possibilities exist.

Many objects are described by presenting their top, front, and side views on an engineering drawing or blueprint. This is the most common combination of views used for description.

The reader of a drawing attempts to determine the shape of the object by quickly surveying all of the given views [8,9]. He imagines himself moving around the object by looking at the features of the object, beginning with the most prominent features. A person studying drafting often tries to develop skill at reading drawings through exercises in sketching objects and modeling them with clay. The process of reading

drawings depends heavily on the reader's intuitive ability to determine the three-dimensional shape of the object.

C. Characteristics of production systems

The system discussed here is a production system implemented in OPS5 [10]. A production is a statement of the form

IF C1, C2, . . ., Cn
THEN A1, A2, . . ., Am

where C1, C2, . . ., Cn are conditions and A1, A2, . . ., Am are actions to be taken if conditions C1, C2, . . ., Cn are met. The conditions are called the antecedent of the production and the actions are called its consequent.

OPS5 is a general purpose production language. A program in OPS5 consists of a set of productions which operate on data in a database called working memory. During the execution of a program, the conditions of each production are tested to see if they are satisfied by the data in working memory. This is termed matching. Then one of the productions whose conditions are satisfied by working memory is selected, and its actions are performed. This mode of inference is forward chaining. OPS5 incorporates a conflict resolution strategy which selects a particular production to be executed when more than one is executable. Execution continues until there is no production whose antecedent is satisfied.

II. Representation of information

The input data for this program are descriptions of line drawings of the top, front, and side views of the object. Each line segment of each drawing is input by the system as a pair of endpoints in the appropriate plane, where an endpoint consists of a pair of coordinates. A coordinate system is chosen so that the top view is described by x and y coordinates, the front view by x and z coordinates, and the side view by y and z coordinates. The origin of each plane is in the lower left corner, as shown in Figure 3.

The results for a given object are the descriptions of a set of lines in three dimensions, where each line is specified by its endpoints, and each endpoint is specified by its x, y, and z coordinates. The state of the system at any time is determined by the contents of working memory, where the data in working memory are represented as instances of a set of objects. Each object has a set of attributes associated with it, and an instance of the object may have a value associated with each attribute.

III. Geometrical reasoning

The goal of the system is to find a correct three-dimensional interpretation for all line segments which matches all of the data. The reasoning strategy for this system is therefore data-driven, which is implemented readily with the match-act control strategy of OPS5. Forward reasoning works well for this problem since, for the most part, the system starts with the given two-dimensional line segments and vertices and tries to derive three-dimensional line segments. (It would be impossible to use a backward reasoning strategy, because there are an infinite number of possible objects which could be represented [11].) The rules attempt to match lines and points in the three views which may be projections of the same edge of the object. Backtracking is used in cases where three-dimensional points or lines which are inferred are later shown to be incorrect. The system terminates when it finds a three-dimensional interpretation which is consistent with all line segments.

A. Program execution

The operation of the system is described by the following sequence of contexts and their associated actions:

- Read 2D line segments
- Generate 2D vertices, calculate slopes of 2D lines, and extend 2D lines
- Generate 3D vertices and lines
- Remove incorrect 3D vertices and lines
- Print resultant 3D lines

A predicate named GOAL maintains and controls the current value of this context.

For example, in the context for reading data, the attribute TYPE of the predicate GOAL alternates between values read-code and read-data. When TYPE has value read-code, a production causes a value to be read which determines whether the data corresponds to a line segment in the top, front, or side view. This production also changes the value of the attribute TYPE of GOAL to read-data. An appropriate production can then fire, causing the data to be read and changing the GOAL back to read-code. This sequence is repeated until an end-of-file is reached. The attribute TYPE of GOAL then has its value changed to the context for generating two-dimensional vertices.

When GOAL has TYPE generate-points, the slopes of all two-dimensional line segments in the three views are calculated; also, all vertices in the three views are generated. The GOAL TYPE generate-points also activates a set of productions which fire when a view contains line segments from point A to point B and from point B to point C with the same slope. A longer line from point A to point C is then generated for that view. These new lines provide information needed for the

generation of lines in the three-dimensional description of the object.

After all vertices have been generated and all slopes have been calculated, the GOAL TYPE is changed to add-points. In this context, productions are activated which generate instances of LINE-3D and POINT-3D and add them to working memory.

After all instances of LINE-3D and POINT-3D have been added to working memory, the value of GOAL TYPE is changed to remove-points. Productions which check for and remove invalid lines and points in the three-dimensional description are then allowed to fire.

The value of the TYPE attribute of GOAL is then changed to print, and a list of the lines which make up the three-dimensional description of the object is written to a file.

B. Rules for adding three-dimensional lines

There are twelve productions which generate the three-dimensional lines whose projections onto each of the three views are lines. These productions all have the following form: if there is a line in the top view with endpoints (x1, y1) and (x2, y2), a line in the front view with endpoints (x1, z1) and (x2, z2), and a line in the side view with endpoints (y1, z1) and (y2, z2), then an instance of LINE-3D with endpoints (x1, y1, z1) and (x2, y2, z2) is generated. Consider the rectangular wedge of Figure 4. The line from (20, 0, 0) to (20, 5, 10) is projected onto the top view as a line from (x=20, y=0) to (x=20, y=5), onto the front view as a line from (x=20, z=0) to (x=20, z=10), and onto the side view as a line from (y=0, z=0) to (y=5, z=10). One of the twelve productions generates an instance of LINE-3D which represents the line from (20, 0, 0) to (20, 5, 10).

There are six productions which generate the three-dimensional lines whose projections onto one of the views are points and whose projections onto the other two views are lines. In fact, these lines have to be perpendicular to the view which shows the line as a point. To illustrate this, consider the cube shown in Figure 5. The three-dimensional line from (10, 0, 10) to (10, 10, 10) is projected onto the front view as a point (x=10, z=10), onto the top view as a line from (x=10, y=0) to (x=10, y=10), and onto the side view as a line from (y=0, z=10) to (y=10, z=10). Notice that the lines in the top and side views are perpendicular to the front view. One of the productions generates the correct instance of this line.

Another set of productions is used for the case where the projection of an edge of the object is a line in each of two views, but does not appear at all in the third view. Consider the case where the top view contains a line from (x1, y1) to (x2, y2), the side view contains a line from (y1, z1) to (y2, z2), but the line from (x1, z1) to (x2, z2) is hidden and does not appear in the front view. The system must look for other lines in the top and side views with the same y coordinates. An example of this case is shown in Figure 6. The lines with the same x coordinate endpoints in the top and side views need to be matched in order of descending x with ascending z. The program must also consider the possibility that a line with different x or z coordinates could prevent the matching.

Two productions generate instances of POINT-3D when a point occurs as an end point in an instance of LINE-3D, but was not previously instantiated as a three-dimensional vertex.

C. Rules for deleting incorrect or redundant lines and points

Consider Figure 7 which shows a cube with one of its corners missing. From the point (x=10, z=10) in the front view and the lines from (x=10, y=0) to (x=10, y=3) in the top view and from (y=0, z=10) to (y=3, z=10) in the side view, one of the productions generates a line in three dimensions from (10, 0, 10) to (10, 5, 10). This line is incorrect since (10, 0, 10) is not part of the figure. There are 64 productions which correct errors of this type by removing the incorrectly inferred vertices from working memory. Once a point has been removed from working memory, then all lines which contain that point are removed. The 64 productions for removing incorrect points have the following general format:

> If there is a line L from point A to point B,
> and there is a line M from point B to point C
> and line M has the same slope as line L,
> and there is a line N from point D to point E,
> and there is a line O from point E to point C,
> and line O has the same slope as line N,
> and there is a line P from point F to point G,
> and there is a line Q from point G to point C,
> and line Q has the same slope as line P,
>
> Then remove point C from working memory.

Inferred instances of three-dimensional lines which contain this point are removed by productions of the form

> If a three-dimensional point C has been removed from working memory,
> and there is a three-dimensional line L from(to) A to(from) C,

Then remove line L from working memory.

The productions which generate instances of three-dimensional lines can generate lines which render existing lines redundant, as follows:

If there is a line L from point A to point B,
and there is a line M from point B to point C,
and line M has the same slope as line L,

Then generate a three-dimensional line from point A to point C.

Lines M and L have obviously become redundant. There are eight productions which remove redundant lines of this form.

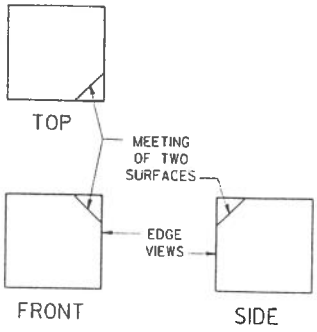
IV. Results and conclusions

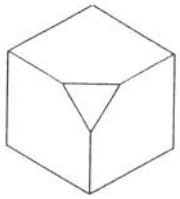
Figures 5 through 11 show examples of results from several different objects. The program successfully produces the three-dimensional representation for many simple polyhedral objects, including those of Figures 5 through 9. It can be expanded to include new polyhedral objects by adding new production rules.

The results for the object of Figure 10 contain an extra line from (0,5,5) to (5,5,5) as if there were two objects, a wedge and a parallelepiped. The problem could be eliminated by a procedure to check to see if the surfaces on either side of a line are in the same plane. If they are, then the line should be eliminated. The pyramid of Figure 11 has the same type of result. The three-dimensional description given by the system contains two extraneous lines in the base plane. There is a line given from (0,0,0) to (20,20,0) and one from (20,0,0) to (0,20,0). The problem here would also be solved by checking to see if all nonboundary lines represent the meeting of two nonplanar surfaces.

A preprocessor for this system which analyzes real two-dimensional images of polyhedra to determine their edges and vertices has been constructed [12]. Further improvements could be made by considering surfaces as well as points and lines so that the program would attempt to match a surface in one view with a surface or line in another view. The program could also be enhanced by a procedure to check to see if the result of an object reconstruction represents a valid polyhedron.

A system such as this could be used for "reverse drafting", i.e., converting an existing object into a computer representation in terms of three-dimensional surfaces, lines, and vertices. This representation is very compact compared to the amount of storage required for three two-dimensional images of the object. More importantly, the representation can then be easily modified and used to control machines for manufacturing similar objects. The system could also lead to insights that could be used for instructing draftsmen in the interpretation of drawings.

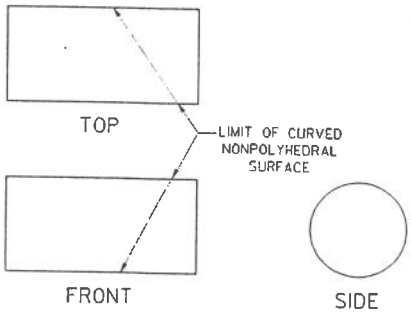


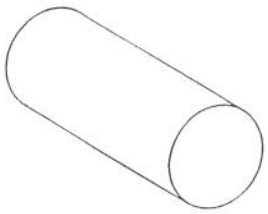


ISOMETRIC

MODIFIED CUBE

Figure 1

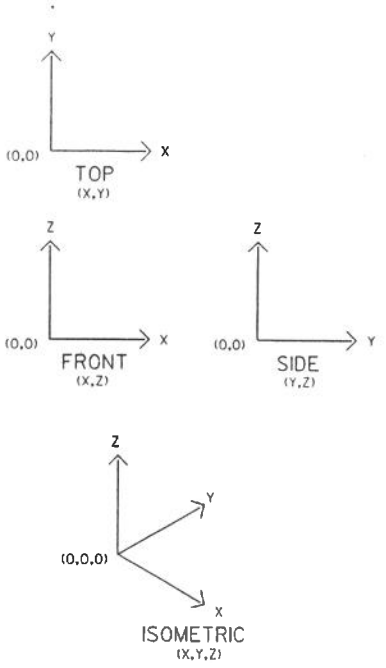




ISOMETRIC

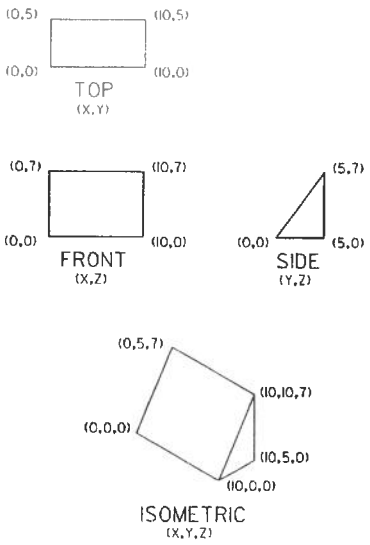
CYLINDER

Figure 2



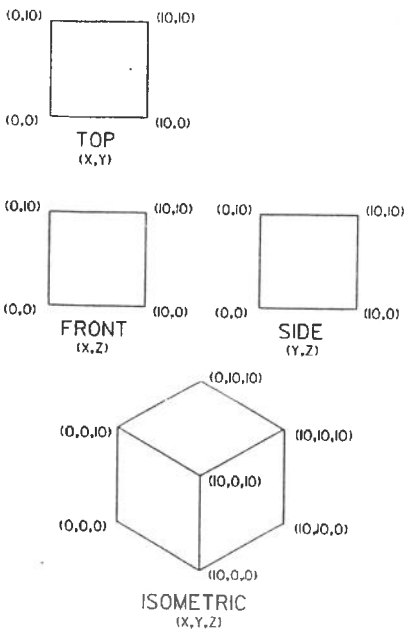
THE COORDINATE SYSTEM

Figure 3



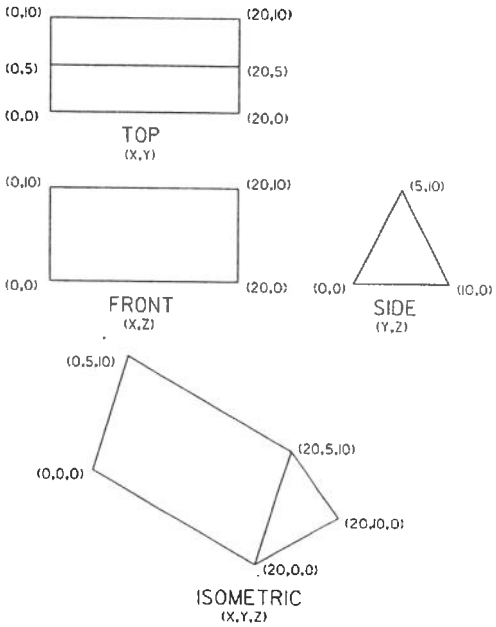
RECTANGULAR WEDGE

Figure 4



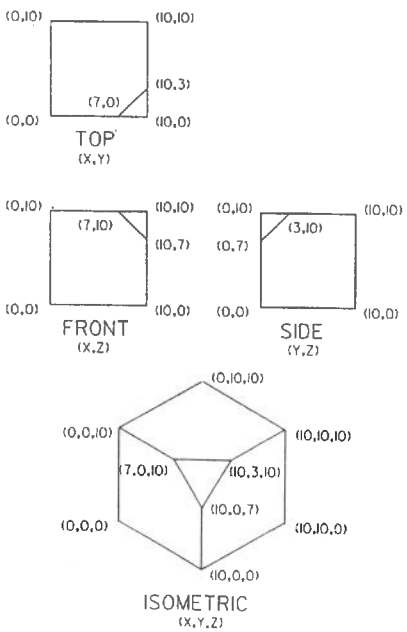
SIMPLE CUBE

Figure 5



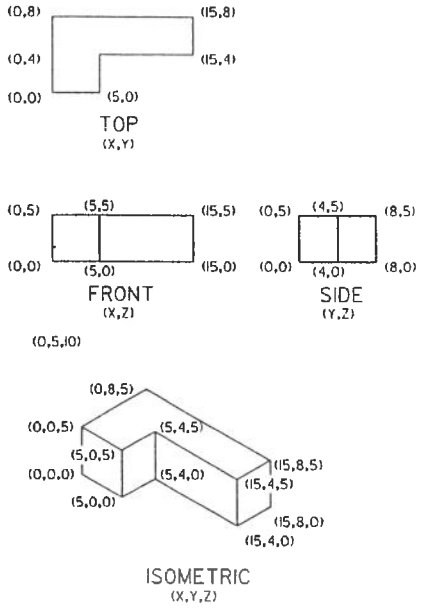
NONRECTANGULAR WEDGE

Figure 6



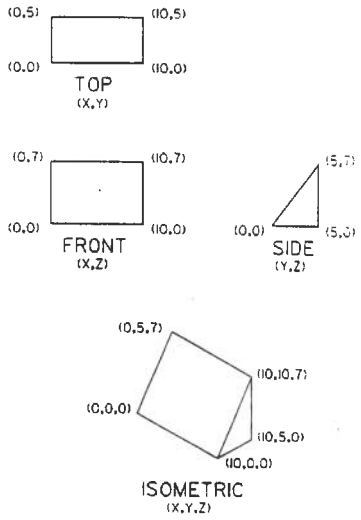
MODIFIED CUBE

Figure 7



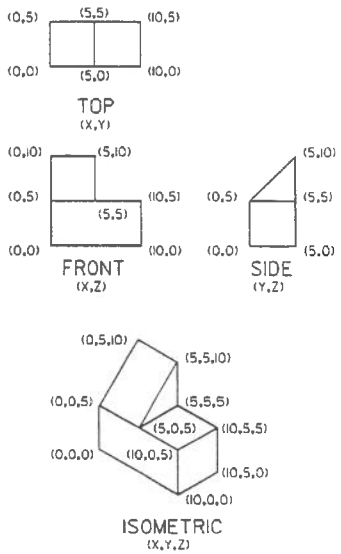
L-SHAPED OBJECT

Figure 8



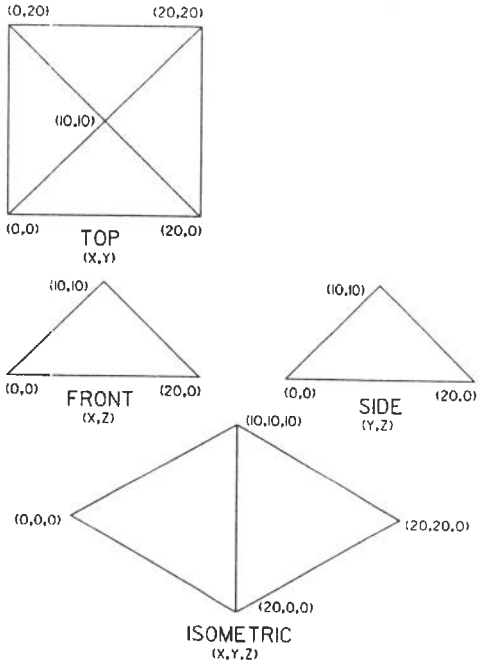
RECTANGULAR WEDGE

Figure 9



OBJECT

Figure 10



PYRAMID

Figure 11

Bibliography

1. D. Waltz, "Understanding Line Drawings of Scenes with Shadows," in The Psychology of Computer Vision, P. H. Winston, ed., New York: McGraw-Hill Book Company, 1975.

2. M. Herman, "Towards Learning Descriptions of Three-Dimensional Objects," Papers of the Workshop on Current Developments in Machine Learning, Carnegie-Mellon University, July 16-18, 1980.

3. M. Karima, K. S. Sadhal, and T. O. McNeil, "From Paper Drawings to Computer-Aided Design," IEEE Transactions on Computer Graphics and Applications, February 1985, pp. 27-39.

4. P. H. Winston, Artificial Intelligence, Reading, MA: Addison-Wesley Publishing Co., 1984.

5. A. K. Mackworth, "Interpreting Pictures of Polyhedral Scenes," Artificial Intelligence, Vol. 4, 1973, pp. 121-137.

6. K. Sugihara, "Mathematical Structures of Line Drawings of Polyhedrons--Toward Man-Machine Communications by Means of Line Drawings," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-4, No. 5, September 1982, pp. 458-469.

7. K. Sugihara, "A Necessary and Sufficient Condition for a Picture to Represent a Polyhedral Scene," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-6, No. 5, September 1984, pp. 578-586.

8. T. E. French, A Manual of Engineering Drawing for Students and Draftsmen, 7th Edition, New York: McGraw-Hill Book Company, Inc., 1947.

9. H. C. Spencer and J. T. Dygon, Basic Technical Drawing, New York: Macmillan Publishing Co., Inc., 1974.

10. C. L. Forgy, OPS5 User's Manual, Department of Computer Science, Carnegie-Mellon University, July 1981.

11. F. Hayes-Roth, D. A. Waterman, and D. B. Lenat, eds., Building Expert Systems, Reading, MA: Addison-Wesley Publishing Company, Inc., 1983.

12. E. T. Whitaker, Rule-Based Geometric Reasoning Using Orthogonal Views of Polyhedra, M. S. Thesis, Department of Electrical and Computer Engineering, University of South Carolina, May 1985.