Interpreting Pictures of Polyhedral Scenes*

A. K. Mackworth

Laboratory of Experimental Psychology, University of Sussex, Falmer, Sussex, England

Recommended by M. B. Clowes

ABSTRACT

A program that achieves the interpretation of line drawings as polyhedral scenes is described. The method is based on general coherence rules that the surfaces and edges must satisfy, thereby avoiding the use of predetermined interpretations of particular categories of picture junctions and corners.

1. Introduction

One way to capture the meaning of pictures is to investigate the relationship between two domains: the picture and whatever it is that is depicted—the scene (1). This paper closely examines that relationship for pictures consisting of straight line segments and scenes made up of opaque polyhedra. A program, POLY, in the same tradition as Guzman's SEE [3] and Clowes' OBSCENE [1] is presented. POLY exploits the relationship between the domains and also coherence rules that entities in the scene domain must satisfy. Following a description of POLY, some feasible extensions to this scheme are described. Finally, the relevance of this program to other scene analysis programs is discussed.

The work reported here stemmed from consideration of several unsatisfactory aspects of OBSCENE. The "predicate table" embodied in that program appears to be a rigid and opaque theory of three-surface corners and the picture-taking process. Secondly, OBSCENE has a very weak grip on the consistency of the viewing direction. Finally, it interprets many pictures as polyhedra which cannot, in fact, exist. The conceptual framework for POLY was inspired by Huffman's "dual-graph" [6], which was presented as a device for checking an interpretation provided by the Huffman–Clowes labelling process.

* Accepted for presentation at the Third International Joint Conference on Artificial Intelligence, 1973.

Artificial Intelligence 4 (1973), 121–137

2. Scene Coherence

Let us first establish a representation for the geometry of polyhedra and the picture taking process.

2.1. Dual Space

In conventional Cartesian space we describe a point by giving its coordinates (x, y, z) and a plane by a constraint upon the coordinates of a point: $a_x x + a_y y + a_z z + 1 = 0$. The representation is as it were point-oriented. Since planes are of more interest to us than points in the context of planar-faced polyhedra, it is desirable to use a representation that is plane-oriented. Such a representation is the dual space [5] in which a plane is represented as a point—specifically by the coefficients a_x, a_y, a_z of the variables in the equation of the "real" plane. It follows that the dual of a point (x, y, z) is a plane such that (a_x, a_y, a_z) is on the plane if

$$x a_x + y a_y + z a_z + 1 = 0.$$

If a line in real space is construed as the intersection of two real planes then its dual is the line passing through the points in dual space which represent those real planes.

2.2. Viewpoint

A two-dimensional image of a three-dimensional body is a projection whose form can be specified in terms of a viewing position and a picture plane.

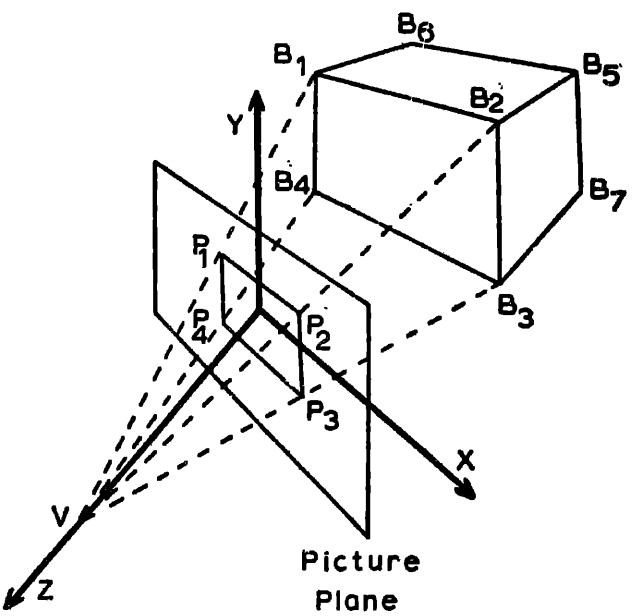

FIG. 1. The picture-taking process.

Figure 1 illustrates such a situation where the picture plane is the x–y plane and the viewpoint V is on the z-axis. If we consider a particular line such as P_1P_2, then P_1P_2, B_1B_2 (the corresponding edge) and V all lie in a plane. This plane we call the *plane of interpretation* (I) of P_1P_2 since given P_1P_2 in a picture we have only to hypothesize the position of V relative to that picture to achieve a powerful constraint upon the possible interpretations of P_1P_2 as an edge, namely that the edge lies in the plane I beyond P_1P_2. Such a hypothesis about V has global implications for it determines the planes of interpretation for all other picture lines simultaneously because *all planes of interpretation must pass through* V. This fact is expressed elegantly in the dual space as the assertion that the duals of all the interpretation planes must lie on the dual of V namely a plane in the dual space D.

If V is at infinity relative to the picture (an ideal point) the projection is orthographic, otherwise it is perspective.

2.3. Bodies

The interpretation of some of picture lines as edges bounding a plane surface of a body is expressed in dual space as the requirement that the duals of these edges all pass through the dual point representing that surface. Hidden edges of a partially visible surface would of course also be subjected to this requirement as would the dual of *any* line presumed to be upon the surface.

The interpretation of a picture junction as the corner of a polyhedron can also be usefully characterised in dual space. A point in real space can be construed as the intersection of a set of planes so that we can identify the planes with the surfaces of the corner and the point with the corner itself. Each edge of the corner is then the line of intersection of a pair of planes, and has as its dual a line which passes through the dual (point) of each of the pair of planes. This set of dual lines forms a polygon lying in a plane in D, that plane which is the dual of the point in real space that we identified with the corner. Thus the dual of an n-surface corner of a polyhedron is a plane n-gon.

Assumptions that the objects in the portrayed three-dimensional situation are polyhedra interface with a model of viewpoint in a particularly simple way. Both the picture line and the edge it depicts lie in the plane of interpretation, I, for that line. Thus the dual of the edge (a line in D) must pass through the dual of I. This can be combined with the requirement that the dual of an edge pass through the duals of the surfaces it belongs to, to obtain the requirement that the duals of I and the two surfaces intersecting in I lie on the dual of the edge which is that intersection. Thus in Fig. 1 the duals of the surfaces $B_1B_2B_3B_4$, $B_1B_2B_5B_6$ and the plane of interpretation of P_1P_2 all lie on the dual line of B_1B_2.

2.4. The Gradient Space

A particularly interesting 2-D subspace of the dual space D is the gradient space G. A point (a_x, a_y, a_z) in D corresponds to the point

$$\left(\frac{a_x}{a_z}, \frac{a_y}{a_z}\right)$$

in G. Geometrically, this corresponds to projecting (a_x, a_y, a_z) into the $a_z = 1$ plane with centre of projection at O and using (0, 0, 1) as the origin O_G of G. In Fig. 2, I in D is projected into I_G in G.

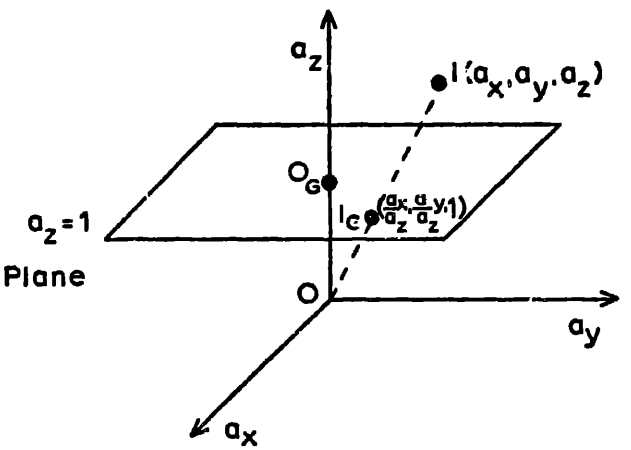

FIG. 2. Projection of a dual point, I, into gradient space.

Several interesting remarks can be made about G. If the equation of the plane is rewritten as

$$-z = \frac{a_x}{a_z}x + \frac{a_y}{a_z}y + \frac{1}{a_z},$$

then we can see why $(G_x, x_y) = (a_x/a_z, a_y/a_z)$ is called the gradient of the plane. In Fig. 1, $-z$ is the distance from the point (x, y, z) on a surface of the object as $B_1B_2B_3B_4$ to the picture plane, $z = 0$. The gradient represents the vector rate of change of this distance with respect to movement in the picture plane, that is,

$$(G_x, G_y) = \left(\frac{\partial(-z)}{\partial x}, \frac{\partial(-z)}{\partial y}\right).$$

The length of the vector from O_G to a point W in G is the tangent of the angle between the picture plane and the plane corresponding to W; the direction of that vector is the direction of the dip of the plane corresponding to W relative to the picture plane. Since the dual of the picture plane is the ideal point on the a_z-axis, O_G the zero gradient, corresponds to it. The projection into the gradient space of the dual line representing an edge may

be called the gradient line of that edge. A perpendicular dropped from O_G to that line is the gradient of that edge in that its direction and magnitude are the direction and tangent of the angle of dip of that edge relative to the picture plane. A family of mutually parallel planes represented by the coordinates (ka_x, ka_y, ka_z) in D will have coincident representations $(a_x/a_z, a_y/a_z)$ in G. Planes which are steeply inclined to the picture plane will be relatively remote from O_G in G. Most of the relationships that were shown to hold in D must necessarily hold in G. In particular, the gradients of the interpretation plane and the two object surfaces that intersect in an edge must be on the gradient line of that edge.

The orientation of a picture line determines the direction of the gradient of its interpretation plane in G. Thus a picture line which is parallel to the y-axis, say, will have an interpretation plane whose gradient I_G lies on the a_x axis. If we align the x–y (picture) axes with the a_x–a_y axes of G the direction of I_G relative to O_G will be perpendicular to the picture line.

The above remarks are all true regardless of the viewing position V and so are true for both orthographic and perspective pictures. We shall now concern ourselves with the gradient space for orthographic pictures. The duals of all the interpretation planes (which must lie on the dual of V) will then be on the $a_z = 0$ plane of D. Projecting them into G will therefore put all their gradients at infinity (they become ideal points in the gradient space). Another way of looking at it is to realize that as V goes further from the picture plane the angles between the picture and the interpretation planes all approach 90° and so the lengths of the gradients approach tan 90° ($\rightarrow \infty$).

The distance of I_G from O_G increases with the (presumed) distance in real space of the viewing point V from the picture plane. The projection onto G of the dual of the *edge* depicted by the picture line, will be a line passing through I_G. For any picture line there is an infinite family of such lines in G being the projection onto G of duals of the possible edges depicted by the line. As V tends to infinity, the picture tends to an orthographic projection of the scene and I_G tends to an ideal point. The family of edge gradient lines in G simultaneously tends toward a set of parallel lines whose orientation is that of the direction of I_G, that is, perpendicular to the picture line.

Consider an orthographic picture of a scene with a visible edge joining two visible surfaces A and B. (We call such an edge a "connect" edge). The gradient space configuration corresponding to that consists of the two gradients (G_A and G_B) joined by a line which is the projection of the dual of the edge. That line is perpendicular to the picture line if the gradient space is superimposed on the picture space as described above. Moreover, it can easily be shown that if the gradients are ordered on the dual line in the same direction as the corresponding surfaces appear at the edge then that edge is

convex but if they are ordered in the reverse direction then it is concave. (Intuitively, imagine a convex edge, then rotate one of the surfaces until it is concave. When the edge is flat, the gradients must coincide.) This crucial fact allows the exploitation of the gradient space for convex/concave interpretations.

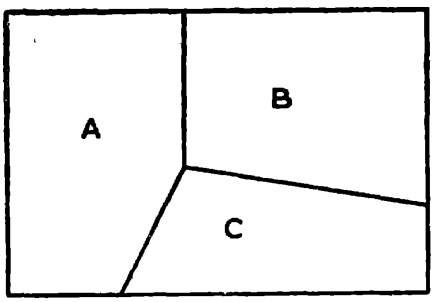

FIG. 3. A FORK junction.

As a simple example of the use of this consider a FORK junction (Fig. 3) where it is known that all the edges are connect. The configuration of the gradients of surfaces A, B, and C (G_A, G_B and G_C) can only take on one of the two forms of Fig. 4 if they are to satisfy the requirement that the mutual vector difference be perpendicular to the line depicting the edge that connects the two surfaces. These configurations can, of course, be translated and expanded in the gradient space and still satisfy the requirement. Comparing the relative positions of the gradients in Fig. 4(a) with the ordering of the regions in the picture shows that all the edges must be convex for that interpretation while for the interpretation given by Fig. 4(b) all the edges must be concave. That switch of interpretations which can be achieved by mapping every gradient G into its negation $-G$ is known in the literature of psychology as the Necker reversal.

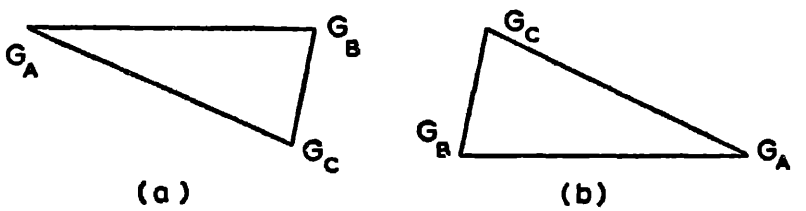

FIG. 4. Two gradient space configurations derived from Fig. 3.

3. Description of the Program

The task for POLY can be specified as follows: using these constraints on the coherent interpretation of polyhedra subjected to this picture-taking process, what information can be derived from the picture? In particular, the program must provide easily accessible answers to questions, such as,

Which edges are connect edges?
„ „ „ convex „ ?
„ „ „ concave „ ?
„ „ „ occluding „ ?
If an edge is occluding, which surface is in front?
How much of the hidden structure of the scene can be recovered?
What is the orientation of each surface and each edge?
and so on.

A program POLY will now be described which recovers these attributes and relationships of the scene. POLY is an existence proof that such questions can be answered. It does not purport to be a stand alone scene analysis program but it can be thought of as a useful embodiment of most of the knowledge specific to these picture and scene domains and their inter-relationship that a scene-based problem solver would need to have available.

FIG. 5. The organization of the program.

The overall structure of POLY is shown in Fig. 5. The program is written in ALGOL 60 extended to allow for the representation and manipulation of data objects, attributes and binary relationships. The input is obtained by drawing a picture on the graphical display; the input phase passes to the parsing phase the end points of the lines. The parsing phase recovers the picture structure by examining the lines for join relationships, and establishing the junctions and closures and the regions made up of closures. The picture is that given in Fig. 6(a). Then the scene correspondents of this data structure are created following the relation of representation [1] as shown in Fig. 6(b).

The CONNECT part of the program uses the rules of coherence sketched earlier to establish which edges are connect and which are not. This part of the program searches over a binary tree with each level representing a different edge in the scene, the left branches being connect (edge) = true and the right branches connect (edge) = false. This tree is not searched in either of the conventional depth-first or breadth-first ways. To achieve the most connected interpretation first, the top level goal requires all edges to be connected and then, when that fails, all edges but one and so on. The tree search is affected by the usual backtracking method with state saving which in this case is achieved by a recursive procedure. The edges are not searched in random order; starting from the background region, each region is interpreted in turn: the next region chosen is that uninterpreted region with the most lines adjacent to the interpreted regions. In this context, to interpret

a region means to fix the position of the corresponding surface in gradient space. Because the region selected by that criterion will correspond to the most constrained surface, this strategy results in the most efficient search. So, in fact, the order of the search is given in advance by the parser but there is no reason why the program could not modify the order of search dynamically if it were embedded in a larger system that could supply advice or hypotheses about the orientation of particular surfaces or the status of various edges.

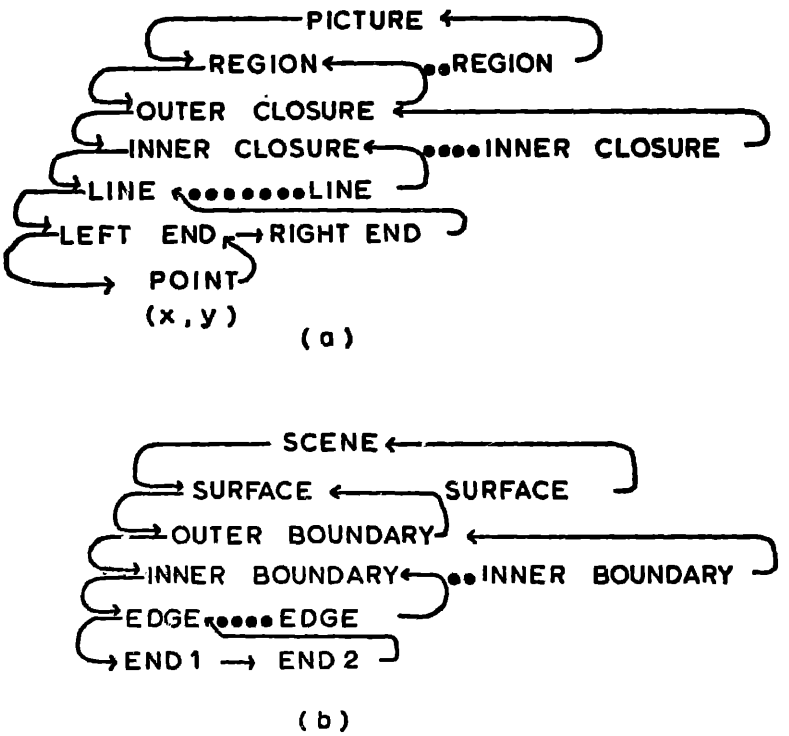

FIG. 6. The picture and scene data structures.

A simple example will make the workings of CONNECT clearer. Consider the picture in Fig. 7. CONNECT fails to find any interpretation with five or with four connect edges for reasons that will become obvious. So with the goal of establishing three connect edges, CONNECT starts with the background A, and for convenience sets G_A at the origin in gradient space. It then examines the lines on the inner closure of A (1, 2, 4 and 5) and finds that none of the regions on the other sides of those lines have been interpreted, so it can say nothing yet about these lines. It then chooses the uninterpreted region that has the most adjacencies to the interpreted regions as the next region to interpret. The ordering of the edges as a tree is determined by this strategy of addressing the picture. Both B and C have two adjacencies, so the choice is arbitrary, say B. Now it examines lines on the

outer closure of B in sequence trying to establish connect edges. Say it looks at 1 first. It establishes it as a connect edge which means that G_B must lie on a line perpendicular to 1 through $G_A = (0, 0)$.

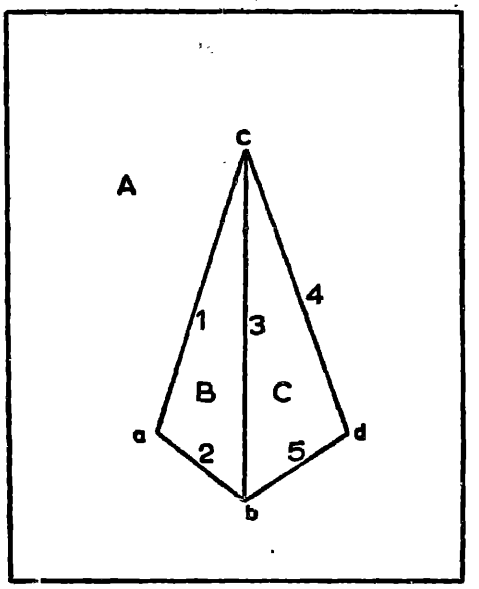

FIG. 7. A simple example.

In general, the position of a gradient is defined by the intersection of two or more dual lines arising from edges assigned connect status; however, for the first two gradients (G_A and G_B, here) there is no loss of generality if we do not use that requirement to locate them since the origin and scale of the gradient space can be subsequently altered. So we put G_B at unit distance from G_A on that line. The next picture line to be considered is 2, which CONNECT also tries to establish as a connect edge but this would require G_B to lie on a line perpendicular to 2 through G_A which is incompatible with the current interpretation of G_B. Thus the interpretation in which both 1 and 2 are connect edges is said to be *incoherent*. This makes it clear why CONNECT failed in its original goal of establishing all the edges as connect edges. 2 is established as an occluding edge and CONNECT looks next at 3. Since the region on the other side, C, is not yet interpreted, it says nothing about 3. The remaining region C is then interpreted. 3 is established as a connect edge requiring G_C to lie on a line perpendicular to 3 passing through G_B. The actual position of G_C is established by defining its relationship to G_A by making 4 or 5 (but *not* both) connect edges. The interpretation in which 1, 3 and 5 are connect and 2 and 4 are occluding edges is rejected by the single rule that three non-collinear points in space (the corners a, b and c) cannot simultaneously lie on two planes (A and B). So one legal connect interpretation is that 1, 3 and 4 are connect edges, while 2 and 5 are not.

Continued search of the tree will only yield one more interpretation with 3 connect edges, viz. 2, 3 and 5 connect, 1 and 4 occluding. For 1, 3 and 4 connect, the final gradient space configuration will be as shown in Fig. 8, in which the gradient line of connect edge 1 is labelled as 1′ etc.

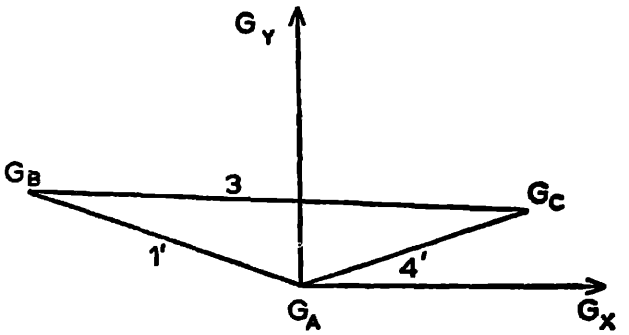

FIG. 8. One possible gradient configuration derived from Fig. 7 by CONNECT.

Then VEXCAVE takes over and decides which of the connect edges are convex and which concave. VEXCAVE starts by partitioning the gradient space graph into 2-connected subgraphs using the gradient lines of connect edges as arcs. For each subgraph VEXCAVE then determines its two possible interpretations using the ordering rule for gradients. In the example, VEXCAVE will decide in the interpretation for which 1, 3 and 4 are connect edges that the whole graph is 2-connected and that either 1 and 4 are concave edges while 3 is convex or 1 and 4 are convex while 3 is concave. Note that, for the latter interpretation, junction b is assigned an "accidental" status.

Finally, OCCLUDE looks at the non-connect edges and uses two inference rules to achieve a complete interpretation. The first rule expresses the fact that if two surfaces intersect in a connect edge that is known to be, say, convex, then at any position in the picture it will be apparent which surface is in front. Using this rule it becomes clear, for many occluding edges, which surface (of the two that it apparently bounds) the edge actually belongs to.

The rule also adds a hidden surface attached to that edge. The fact that such a surface is both turned away from the viewing direction and obscured by the visible surface means that it obeys the same constraint as it would if the edge were concave and connect. This rule is used in the example to decide for the case where 1 and 4 are concave and 3 is convex that occluding edges 2 and 5 belong to surfaces B and C, respectively. The second rule for occlusion completes the polygon of gradients corresponding to the visible and hidden surfaces meeting at each corner. It does this by allowing for the hidden surfaces created by the first rule and introducing the minimum number of extra hidden surfaces required. The minimum number is achieved by allowing two occluding edges to share the same hidden surface wherever

possible. In the example, the second rule confirms that the polygon of gradients is complete for corner c since the surfaces at that corner, A, B and C, are all visible and there are no occluding edges. For corner d it completes the polygon by introducing a hidden edge between the hidden surface at edge 5 and the background. Similarly for corner a and the hidden surface at edge 2. Then at corner b it decides that those two hidden surfaces could be the same surface, D, and still obey the constraints. So the final gradient space configuration is Fig. 9 which looks like a picture of a wire-frame tetrahedron because the tetrahedron is the only self-dual polyhedron [5].

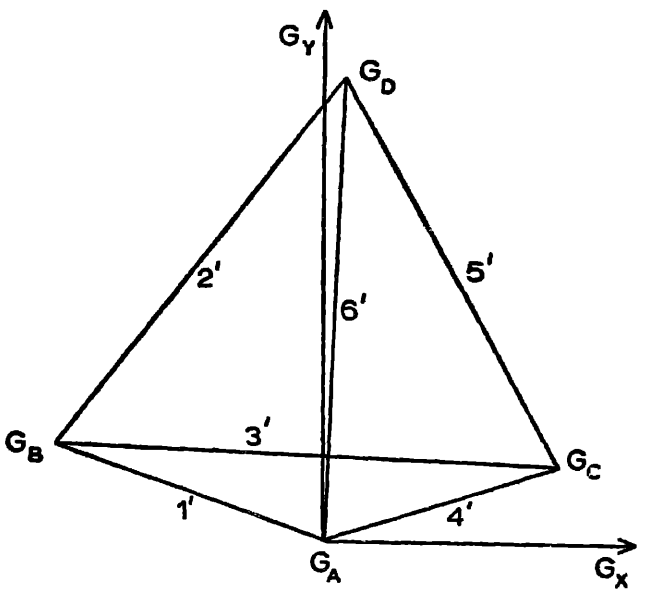

FIG. 9. The gradient configuration derived from Fig. 8 by OCCLUDE.

The interpretation pursued in the example above is one of the first produced by POLY but the program will continue to generate less connected interpretations. For example, the tetrahedron separate from the background surface has only one connect edge, 3', but its gradient space configuration has the same structure as Fig. 9 with the exception that G_A in that figure is replaced by the gradient of a second hidden surface, G_E and G_A is now an isolated point in gradient space. Interpretations such as this with complete bodies separate from the background can be easily generated first by giving CONNECT the advice that all the lines on the inner closure of the frame represent non-connect edges.

When OCCLUDE has finished, then the interpretation process is complete. Each edge node in the scene data structure is related to other scene entities such as the surface it bounds and the corners which bound it. An edge node also contains attributes such as connect, convex, concave or occluding and its slope relative to the picture plane. Nodes for the original visible surfaces

and the hidden surfaces introduced by OCCLUDE contain the gradient vectors which are relative to the gradient of some other surface, usually the background. These gradients may be uniformly scaled by a positive number before being added to the gradient of that other surface to obtain the true gradient. The scale factor must be positive because the work done by OCCLUDE on the hidden surfaces will not survive the Necker transformation that a negative scale factor would involve. This transformation was allowed for earlier, in VEXCAVE, when two versions of the configuration were generated.

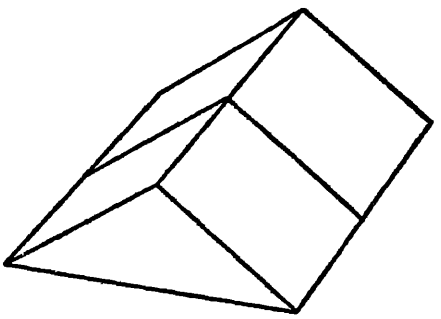

FIG. 10. Two wedges.

Two further points about the program should be made. First, POLY has no difficulty in making sense of cracks as in Fig. 10. Cracks are simply connect edges where the two adjacent surfaces have identical gradients. Finally, the processing time required to produce the first interpretation is proportional to the number of picture lines if that interpretation is completely connected but that tends towards an exponential relationship if the first interpretation is less connected.

4. Possible Extensions of POLY

There are many possible elaborations of this scheme. Since surfaces are represented by their gradients, we learn only the orientation of each surface and not its position in space. It is clear, however, that one could take the results of POLY and by fixing the actual position of one surface propagate the positions of the other surfaces through the connect edges. Alternatively one could use the dual space itself as a representation and build a program that directly exploited the constraints outlined above. Such a program would not have the conceptual simplicity of the implemented scheme.

In theory, POLY only considers orthographic projections but this is not a practical limitation on a scene analysis program. However, one could reformulate the program to deal with perspective projection. In outline, this means that the interpretation plane is now represented as a real point in gradient space since it is not, in general, perpendicular to the picture plane.

The vector from the origin of the gradient space to that point will still be perpendicular to the picture line but the gradient line of the edge is only required to pass through that point and also contain the gradients of the two object planes. Since the gradients of the planes of interpretation of all the picture lines are determined by the geometry of the picture and the position of the viewpoint relative to the picture plane these constraints can be systematically exploited to construct a perspective interpretation of the line drawing.

If the scene is lit by one or more discrete light sources, the boundaries of the shadows cast are depicted as straight lines. Consider a shadow plane formed by the light source, a shadow-casting edge and the corresponding shadow boundary. The gradient of such a plane must lie on the gradient line of the edge (which is perpendicular to the picture line and contains the gradients of the two object surfaces meeting at the edge). The shadow boundary will also have a gradient line which contains the gradients of the shadow-receiving surface and the shadow plane. Moreover, for a source producing a parallel beam, the gradients of all the shadow planes must lie on a straight line in gradient space that is perpendicular to the direction of illumination in the picture. Such constraints allow the extension of the scheme to include shadow interpretation. Such an extension could also use the assumption of uniformly reflecting surfaces of constant albedo that would enable the program to infer constraints on the gradient of a surface relative to the light source from the apparent luminance of that surface.

5. POLY and Related Programs

POLY has particularly interesting relationships with three other vision programs, namely, Guzman's SEE [3], Clowes' OBSCENE [1] and Falk's INTERPRET [2].

SEE accepts input in the same form as POLY does and produces groupings of the picture regions on the basis of the putative body membership of the surfaces depicted. The program starts by classifying each junction into one of a small number of junction categories. It then uses this classification to place links between regions if certain local junction configurations exist. The resultant graph (with surfaces as nodes and links as arcs) is then examined for well connected subgraphs which are declared to represent bodies. In the language of POLY, Guzman's links can be thought of as good guesses at connect edges and, indeed SEE's graph structure is a weaker version of POLY's gradient space representation. But SEE fails to exploit fully our knowledge of three-dimensional situations; for example, there is no representation in the program for the fact that if two edges between two surfaces are not collinear, they cannot both be connect edges. SEE's tendency to see holes in objects as separate objects [8] is only one consequence of the fact

that the program ignores inherent ambiguities in the interpretation process that are exposed by the next program to be considered here, OBSCENE.

OBSCENE (and Huffman's labelling algorithm [6]) gives each edge in the scene one of four interpretations, namely, convex, concave and the two occluding possibilities. OBSCENE works by mapping junctions into corners using some of the junction categories described by Guzman. Each junction type (ELL, FORK, ARROW and TEE) has a small number of pre-determined corner interpretations. The program then pursues the legal combinations of these using the coherence rule that an edge must have the same interpretation at each end. OBSCENE can be seen as a theory of why SEE works, in that it makes explicit what is implicit in SEE. POLY, in turn, is a theory of why OBSCENE works, in that it shows how to derive the junction categories of OBSCENE.

POLY can be seen as a descendant of OBSCENE in several ways. The coherence rules for OBSCENE are at the edge level whereas POLY requires each surface to have a unique orientation in space. This higher level of coherence means, for example, that POLY rejects as ill-formed, skewed objects, such as Fig. 10, that OBSCENE will accept.

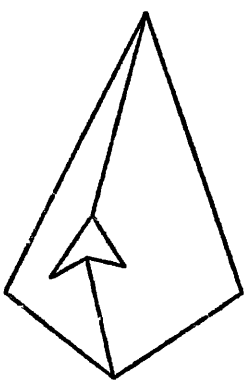

FIG. 11. Skewed tetrahedron with a notch cut in it.

The mapping rules for OBSCENE are at the corner level while those for POLY are at the level of edges. For OBSCENE, this results in the rather unsatisfactory predicate table in which are listed all the various 3-surface corners which could be depicted by each junction type. All the entries are worked out in advance by the programmer whereas drawing each junction type as input to POLY would result in interpretations that are the OBSCENE predicate table entries.

OBSCENE's edge mapping is

$$\text{line} \rightarrow \left\{\begin{array}{l}\text{convex}\\ \text{concave}\\ \text{hind 1}\\ \text{hind 2}\end{array}\right\} \text{edge},$$

but POLY uses the fact that those four categories are really hiding two boolean predicates. POLY's mapping is:

$$\text{line} \rightarrow \left\{\begin{matrix}\text{connect} \\ \text{non-connect}\end{matrix}\right\} \text{edge},$$

and the combinatorial search is done on this predicate alone with the other predicate determined by non-search procedures.

Waltz [7] considerably extended the Huffman–Clowes labelling procedure by sub-dividing the four categories of edge types and adding cracks and shadow boundaries; furthermore, he ingeniously modified the search mechanism to avoid the combinatorial explosion of a straightforward breadth-first procedure. Nevertheless, most of the remarks made here about the Huffman–Clowes procedure seen in terms of POLY apply equally to Waltz's extension. Waltz also assigns to each surface an illumination status: illuminated, turned away from the light or shaded by another surface. Such hypotheses would be better justified if the surface orientations were explicitly represented as in the gradient space. Similarly his treatment of shadows does not include the global consistencies outlined in Section 4 above. Finally, Waltz suggested a scheme to check a labelled picture by using quantized versions of line, edge and surface orientations related through tabulations of possible values. POLY exploits directly a more concise and transparent representation of those relationships to construct rich scene interpretations thereby dispensing with possible corner lists.

Falk's INTERPRET [2] is a well-documented account of a complete scene analysis system that interprets line drawings, and so it is instructive to see how POLY could relate to that program. But first, consider Falk's "face adjacency graph". This concept, although not used in the program, is outlined in his paper because it "would be valuable for a scene analysis system which operated in a universe of planar faced solids more complicated" [2, p. 112] than the nine simple objects his program recognizes. Falk suggests doing a Huffman–Clowes analysis of the picture to determine which edges are connect and then constructing a graph with surfaces as nodes and connect edges as arcs. It is then shown that a property of this graph, "mergeability", gives the number of independent points that need to be located in three dimensions in order to specify the scene completely. In particular, if the graph is 1-mergeable (2-connected in graph-theoretic terms), then 4 points must be located. These 4 scalars correspond to setting the origin and scale of the gradient space and the distance of the object from the picture plane. Falk's result applies directly to the gradient space configuration produced by POLY, which is not confined to isolated, degree 3 polyhedra.

With respect to INTERPRET itself, Falk mentions several times that a

Huffman–Clowes labelling would have helped the program. Those remarks apply, a fortiori, to POLY. In addition, the surface orientations available in POLY would help in support determination and in recognition; also, the inclination of edges relative to the picture plane give the foreshortening factor necessary to calculate true edge length. Finally, consider the seven somewhat opaque heuristics that Falk uses to determine possible base edges. He is forced to use these because at that stage the program is functioning entirely in the picture domain. The analysis offered by POLY, which constructs a scene interpretation without "recognizing" the objects, provides a structure in which one would find the lowest hidden surface and simply ask which visible edges are attached to it.

The caveat should be entered that, as they stand, both OBSCENE and POLY require complete line drawings while Falk interprets pictures in which lines can be missing. However, examination of the manner of failure of POLY on a particular picture will suggest where lines may be missing or extraneous by showing, at the very least, which subpictures can be sensibly interpreted. Furthermore, hypotheses concerning lines to be added or removed can be confirmed by successful analysis by POLY.

6. Conclusion

Although POLY is restricted to the interpretation of compete line drawings showing an orthographic view of a shadow-free polyhedral scene, the extensions in Section 4 and the discussion in Section 5 (particularly the relationship between POLY and Falk's INTERPRET) suggest that all of these restrictions except the overriding commitment to polyhedra can be overcome. Be that as it may, the program does demonstrate just how much structural information can be inferred from the picture using knowledge of the picture-taking process and the general nature of polyhedra but without using specific polyhedral prototypes.

ACKNOWLEDGEMENTS

Dr. Max Clowes provided much welcome advice and criticism. The author also acknowledges facilities provided by Professor N. S. Sutherland, the financial support of the National Research Council of Canada and computing made available by the Science Research Council of Great Britain.

REFERENCES

1. Clowes, M. B. On seeing things. *Artificial Intelligence* 2 (1) (1971), 79–116.
2. Falk, G. Interpretation of imperfect line data as a three dimensional scene. *Artificial Intelligence* 3 (2) (1972), 101–144.
3. Guzman, A. Decomposition of a visual scene into three-dimensional bodies. *AFIPS Proc. Fall Joint Comp. Conf.*, 33 (1968), 291–304.
4. Harary, F. *Graph Theory*, Addison Wesley, Reading, Mass., 1969.

5. Hilbert, D. and Cohn-Vossen, S. *Geometry and the Imagination*, Chelsea, New York, 1952.
6. Huffman, D. A. Impossible objects as nonsense sentences. *Machine Intelligence* **6**, B. Meltzer and D. Michie (eds.), Edinburgh University Press, Edinburgh, 1971, 295–323.
7. Waltz, D. L. Generating semantic descriptions from drawing of scenes with shadows. Thesis, AI TR–271, Massachusetts Inst. of Technol., Cambridge, Mass. (1972).
8. Winston, P. H. Holes. A.I. Memo No. 163. Massachusetts Inst. of Technol. (April 1970).

Received 14 June 1973