M. A. Wesley
G. Markowsky*

Fleshing Out Projections

In an earlier paper, the authors presented an algorithm for finding all polyhedral solid objects with a given set of vertices and straight line edges (its wire frame*). This paper extends the Wire Frame algorithm to find all solid polyhedral objects with a given set of two dimensional projections. These projections may contain depth information in the form of dashed and solid lines, may represent cross sections, and may be overall or detail views. The choice of labeling conventions in the projections determines the difficulty of the problem. It is shown that with certain conventions and projections the problem of fleshing out projections essentially reduces to the problem of fleshing out wire frames. Even if no labeling is used, the Projections algorithm presented here finds all solutions even though it is possible to construct simple examples with a very large number of solutions. Such examples have a large amount of symmetry and various accidental coincidences which typically do not occur in objects of practical interest. Because of its generality, the algorithm can handle pathological cases if they arise. This Projections algorithm, which has applications in the conversion of engineering drawings in a Computer Aided Design, Computer Aided Manufacturing (CADCAM) system, has been implemented. The algorithm has successfully found solutions to problems that are rather complex in terms of either the number of possible solutions or the inherent complexity of projections of objects of engineering interest.*

1. Introduction

In an earlier paper [1] the authors presented an algorithm for finding all polyhedral solid objects with a given set of vertices and straight line edges (its wire frame). The Wire Frame algorithm was based on the concepts of algebraic topology and rigorous definitions of the geometric entities involved. It recognized that many solid objects may have the same wire frame and was able to find all possible solutions efficiently.

In this paper we extend the Wire Frame algorithm to polyhedral objects described by a set of two dimensional projections such as might be seen on an engineering drawing. The projection process may introduce another level of ambiguity into reconstruction problems and increases the possibility of there being many objects with the same set of projections. The Projections algorithm presented here can work with very little information, for example, only two projections, and find all possible objects matching the data. However, it is seen that the number of solutions may be very large and that it may be reasonable to provide more information in the form of three or more projections, by labeling corresponding features in divers views, and by providing depth information. The Projections algorithm is able to make use of this extra information and can also accept other forms of advice, such as whether given points are inside material.

Quite apart from its mathematical interest, the algorithm has practical applications in the automatic conversion of digitized engineering drawings into solid volumetric representations of the geometry of objects. These solid volumetric representations become the basis for the simulation and synthesis of large parts of the design validation, analysis, manufacture, inspection, and documentation process [2, 3].

The subject of reconstruction of solid polyhedral objects from their projections has been studied over a period

*Consistent use of alphabetical ordering of authors' names tends to slight people whose names begin with letters towards the end of the alphabet. Thus, the order of names on this paper is not meant to pass judgement on the relative contributions of the authors, but rather to illustrate the fact that names appearing in alphabetical order is not a "natural law."

of years. Early work [4–6] was largely based on labeling corresponding information in different views and requiring the user to conform to constraints on the manner of description of features such as faces. The historical trend has been to free the user of as many constraints as possible [7]. However, the relaxation of constraints has led to the possibility of multiple solutions to a given problem, and workers have tended to concentrate on heuristic approaches to find a probable solution. A recent paper [8] reports such a heuristic approach that allows complete freedom of input and has been implemented; another paper [9] outlines an approach that would allow certain views of cylindrical surfaces but does not include an implementation. None of this work appears to be based on formal geometric definitions and the concepts of algebraic topology. A closely related development path has been followed by workers in the fields of Computer Vision and Scene Analysis. This path has been based on vertex and edge configurations in a single view [10–12] and has generally been restricted to objects with trihedral vertices and views with no chance alignments; this approach has led to the Origami World [13] and a linear programming approach [14].

This paper presents a very general and complete approach based on the authors' previously published Wire Frame algorithm. In addressing the problem of constructing a solid object from a number of two dimensional views, it is shown that, on the one hand, complete labeling of edges and vertices leads to the previously published Wire Frame algorithm. On the other hand, the Projections algorithm described here is capable of working with no further information than the lines and points of the two dimensional projections and is able to enumerate all possible solutions to a given set of projections, with a cost commensurate with the number of solutions. The techniques presented are applicable when two or more projections are available. Of course, the one projection case has, in general, infinitely many solutions and is not discussed further in this paper. The chief advantage gained from providing more projections is quite naturally to reduce the number of possible ambiguities.

The Projections algorithm constructs polyhedral objects from projections containing only straight lines. The logical component of this algorithm is topological in nature and is, in principle, independent of whether the components are linear or nonlinear. While extension to objects with curved surfaces and projections with curved lines appears to be feasible, the ease of actually carrying out such an extension would depend greatly on the family of allowable curves and surfaces, as well as the projection conventions used.

The paper is organized as follows: Section 2 reviews the definitions of objects, faces, edges, and vertices used in the paper describing the Wire Frame algorithm [1] and then develops the basic results dealing with back projections and labeled projections. Section 3 outlines the original Wire Frame algorithm and describes the Basic Projections algorithm which handles the general case of unlabeled projections of wire frames of objects. Section 4 presents some extensions to the Basic Projections algorithm which enable it to make use of more general forms of input data. For example, various types of views (overall, detail, and cross section) and depth information distinguishing between visible and occulted lines are considered. In Section 5, some examples are given to clarify this discussion. These examples illustrate the execution of the algorithm in both the stylized world of geometric puzzles with multiple solutions and the practical world of engineering drawings. The engineering objects successfully constructed from their projections are sufficiently complicated that a human unfamiliar with the solid object generally has some difficulty envisioning it. Thus, the algorithm appears capable of handling real world problems.

2. Basic concepts and results

The basic concepts defined in this section are based on some fundamental topological ideas which are described in detail in [15]. Throughout the paper the standard topology in $\mathbb{R}^3$ and the induced topology on subsets of $\mathbb{R}^3$ are assumed. Vertices refer to points in $\mathbb{R}^3$ and edges refer to line segments defined by two points in $\mathbb{R}^3$. The approach used in this section is to define faces, objects, wire frames, and projections, and then describe the consequences of these definitions.

Definition 1

A *face*, f, is the closure of a nonempty, bounded, connected, coplanar, open (in the relative topology) subset of $\mathbb{R}^3$ whose boundary (denoted by ∂f) is the union of a finite number of line segments. P_f is used to denote the unique plane which contains f. □

Definition 2

An *object*, $\mathcal{O}$, is the closure of a nonempty, bounded, open subset of $\mathbb{R}^3$ whose boundary (denoted by $\partial\mathcal{O}$) is the union of a finite number of faces. □

From the definitions above it is easy to see that the "cube," $\{x, y, z \in \mathbb{R}^3 \mid 0 \le x \le 1, 0 \le y \le 1, 0 \le z \le 1\}$ is an object and that $\{(1, y, z) \in \mathbb{R}^3 \mid 0 \le y \le 1, 0 \le z \le 1\}$ is one of its "square" faces. Starting off with open sets means that faces and objects have nontrivial interiors. Notice that it is not assumed that an object is the closure of a connected set. This allows objects that consist of disjoint "solids" or even objects which intersect only in

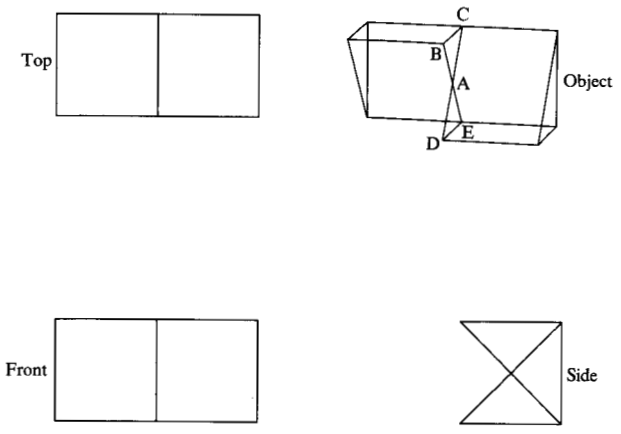

Figure 1 Examples of projections.

edges, etc. One can argue that this last case does not represent a "real" object, but in practice all sorts of strange objects can appear. Thus, we decided to handle the most general case possible. Furthermore, this generality does not exact any penalty other than creating a large number of solutions.

Another point worth noticing is that Definitions 1 and 2 allow many different representations of the boundaries of faces and objects by line segments and faces (respectively). However, there are canonical representations of the boundaries which correspond to one's intuitive notions about such things. To get to these representations it is necessary to introduce several additional concepts.

Definition 3

(a) Let f be a face. The *vertices* of f, V(f), are defined to be the set of all points for which two noncolinear line segments, contained in ∂f, can be found whose intersection is the given point.

(b) Let f be a face. The *edges* of f, E(f), are defined to be the set of all line segments e, contained in ∂f, satisfying the following conditions:

1. The endpoints of e belong to V(f);
2. No interior point of e belongs to V(f).

(c) Let $\mathbb{O}$ be an object. The *vertices* of $\mathbb{O}$, V($\mathbb{O}$), are defined to be the set of all points p for which faces $f_1, f_2, f_3 \subseteq \partial\mathbb{O}$ can be found such that $\{p\} = f_1 \cap f_2 \cap f_3 = P_{f_1} \cap P_{f_2} \cap P_{f_3}$.

(d) Let $\mathbb{O}$ be an object. The *edges* of $\mathbb{O}$, E($\mathbb{O}$), are defined to be the set of all line segments e, contained in $\partial\mathbb{O}$, satisfying the following conditions:

1. The endpoints of e belong to V($\mathbb{O}$);
2. No interior point of e belongs to V($\mathbb{O}$);
3. For every point p of e, two noncoplanar faces can be found, $f_1, f_2 \subseteq \partial\mathbb{O}$ such that $p \in f_1 \cap f_2$.

(e) Let $\mathbb{O}$ be an object. The *wire frame* of $\mathbb{O}$, WF($\mathbb{O}$), is defined to be the ordered pair (V($\mathbb{O}$), E($\mathbb{O}$)). □

It can be shown that the edges of an object can intersect only at vertices of the object, *i.e.*, at their endpoints.

The Wire Frame algorithm detailed in [1] allows one to construct all possible objects which have a given wire frame. It happens to be true, but not immediately obvious from the definitions, that V(f), E(f), V($\mathbb{O}$), and E($\mathbb{O}$) are all finite and well-defined. These facts and others are discussed in greater detail in [1].

The Wire Frame algorithm described in [1] runs on any collection of points and line segments in $\mathbb{R}^3$ and either returns all objects having the given collection as their wire frame or shows that the given collection could not be a valid wire frame. In presenting the Projections algorithm the first things to consider are the projections of the wire frame of a valid object. At this point it is necessary to make clear exactly what is meant by a projection.

Definition 4

Let $\mathbb{O}$ be an object, $\mathbf{P} \subset \mathbb{R}^3$ a plane, and $\pi_p: \mathbb{R}^3 \rightarrow \mathbf{P}$ the perpendicular projection. By the **P**-*projection* of $\mathbb{O}$, denoted by $\mathbb{O} \mid \mathbf{P}$, is meant the ordered pair, (V($\mathbb{O}$|**P**), E($\mathbb{O}$|**P**)), of **P**-*vertices* and **P**-*edges* of $\mathbb{O}$ defined by the following process. Let E* be the set of images under π_p of all edges of $\mathbb{O}$ which are not perpendicular to **P**. Then the **P**-vertices of $\mathbb{O}$ are those points of **P** which lie on at least two noncolinear line segments in E*. The **P**-edges of $\mathbb{O}$ are those line segments of **P** which have elements of V($\mathbb{O}$|**P**) as endpoints, have no points of V($\mathbb{O}$|**P**) as interior points, and are subsets of unions of elements of E*.

XY, YZ, and ZX are used to denote the planes Z = 0, X = 0, and Y = 0, respectively. □

Figure 1 shows some of the things that can happen as a result of projection. The vertex A disappears in the front and top views. Furthermore, the edges AB, AC, AD, and AE do not appear as such in these views. Rather a single edge appears which is the union of the projections of the four aforementioned line segments. However, in the side view the vertex A projects into a vertex, and the projections of AB, AC, AD, and AE form distinct line segments.

At this point it seems appropriate to discuss the situations in which vertices of an object project into vertices in

a given projection. Note that if a vertex of a polyhedral object is the intersection of at least three noncoplanar line segments, the image of that vertex under any projection is the intersection of at least two noncolinear line segments and is thus a vertex in that projection. For convenience, vertices which are the intersections of at least three noncoplanar line segments are called *Class I vertices*. Thus, if two different projections of an object are given, the Class I vertices are a subset of the set of all intersections of all the perpendiculars erected at the vertices in each projection.

All vertices of an object which are not Class I are to be called *Class II vertices*. In Fig. 1, vertex A is Class II; all other vertices are Class I. In general, one cannot expect to recover Class II vertices simply by erecting perpendiculars and computing their intersections.

There are a number of properties of Class I and Class II vertices which are useful in recovering an object from its projections. The key observation, which is formalized below, is that the wire frame of $\mathbb{O}$ can be recovered from the Class I vertices of $\mathbb{O}$ and certain line segments joining these vertices.

Definition 5
The *skeleton*, $S(\mathbb{O})$, of an object $\mathbb{O}$ is the ordered pair $(SV(\mathbb{O}), SE(\mathbb{O}))$ of *skeletal vertices* and *skeletal edges* where $SV(\mathbb{O})$ is the set of the Class I vertices of $\mathbb{O}$ and $SE(\mathbb{O})$ is a set of line segments joining the elements of $SV(\mathbb{O})$. For $v_1, v_2 \in SV(\mathbb{O})$, there exists $w \in SE(\mathbb{O})$, joining v_1, v_2 iff there exists an edge or colinear sequence of edges of $\mathbb{O}$ joining v_1 and v_2 and not containing any other Class I vertex. □

Theorem 6
Let $\mathbb{O}$ be an object. Then the wire frame of $\mathbb{O}$, $(V(\mathbb{O}), E(\mathbb{O}))$, can be recovered from the skeleton of $\mathbb{O}$, $(SV(\mathbb{O}), SE(\mathbb{O}))$, as follows. First, $V(\mathbb{O}) = V^*(\mathbb{O})$ where

$$V^*(\mathbb{O}) = SV(\mathbb{O}) \cup \{v | \{v\} = e_1 \cap e_2, e_1, e_2 \in SE(\mathbb{O})\}.$$

Thus, to get all vertices of $\mathbb{O}$ it is enough to add all intersection points of skeletal edges to the skeletal vertices. Second, $E(\mathbb{O})$ is simply the set of line segments which result from partitioning the skeletal edges using their points of intersection.

Proof Observe that from Definition 5 it follows that every skeletal edge is the union of edges of $\mathbb{O}$. Thus, the intersection of two skeletal edges is a point of intersection of two edges. However, edges of $\mathbb{O}$ intersect only in vertices of $\mathbb{O}$. Thus, $V^*(\mathbb{O}) \subseteq V(\mathbb{O})$.

It remains to show that $V(\mathbb{O}) \subseteq V^*(\mathbb{O})$. In particular it must only be demonstrated that every Class II vertex of $\mathbb{O}$ belongs to $V^*(\mathbb{O})$. To see this it is necessary to consider briefly the nature of the edges of $\mathbb{O}$. Let $e \in E(\mathbb{O})$, $p \in e$, and ℓ the infinite line through p containing e. Let X be the set of disjoint line segments formed by the intersection of ℓ and the boundary of $\mathbb{O}$. Now either p is in the interior of X (*i.e.*, there are points of X on both sides of p which are arbitrarily close to p) or p has arbitrarily close neighbors only to one side of it. Let f_1 and f_2 be the two noncoplanar faces whose intersection contains e. If p is not in the interior of X, then, since p is on the boundaries of f_1 and f_2 but is not in the interior of any of the edges, it must be a vertex of each of the faces, *i.e.*, there must be edges, $e_1 \in f_1$, $e_2 \in f_2$, not colinear with e such that $\{p\} = e \cap e_1 = e \cap e_2$. But in this case there are three noncoplanar edges through p, namely, e, e_1, and e_2. Thus, p is a Class I vertex.

The point of the preceding paragraph is to show that either a point, p, of an edge, e, is a Class I vertex or the line through p containing e has boundary points of $\mathbb{O}$ arbitrarily close to p, *i.e.*, there exists a line segment $s \supseteq e$ contained in $\partial\mathbb{O}$ for which p is an interior point. In particular, an edge, e, containing a Class II vertex p can be extended to a line segment s lying in $\partial\mathbb{O}$ containing e whose endpoints are Class I vertices, *i.e.*, every edge of $\mathbb{O}$ is contained in some skeletal edge. Since every vertex of $\mathbb{O}$ must lie on at least three edges, every Class II vertex of $\mathbb{O}$ must lie on at least two skeletal edges and hence $V(\mathbb{O}) = V^*(\mathbb{O})$.

Since every edge of $\mathbb{O}$ lies in some skeletal edge and $V(\mathbb{O}) = V^*(\mathbb{O})$, it follows that the edges of $\mathbb{O}$ are exactly the pieces into which the skeletal edges are partitioned by the vertices of $\mathbb{O}$. □

Theorem 6 gives some insight into the working of the Projections algorithm. Back projection yields a *pseudo skeleton* consisting of a set of vertices which includes the Class I vertices and a set of edges. This pseudo skeleton is processed to produce a *pseudo wire frame*. In general, the pseudo skeleton and pseudo wire frame contain vertices and edges not in the skeleton and wire frame of the original object. However, they do contain all the vertices and a partition of the edges of the skeleton and wire frame of the original object. In fact, the additional complexity of the Projections algorithm is based on the fact that back projection generally yields many vertices and edges not in the original object. The Projections algorithm thus proceeds along the lines laid down by the Wire Frame algorithm, but with suitable modifications made to deal with surplus information.

The discussion of Class II vertices in the proof of Theorem 6 shows that they have various properties, one

of which appears as Theorem 7. Theorem 7 is very useful in showing that certain points which arise from back projection cannot be vertices of $\mathbb{O}$. Example 4 later in the paper illustrates the power of this observation.

Theorem 7
Let $\mathbb{O}$ be an object and v a Class II vertex of $\mathbb{O}$. Any plane, **P**, through v separates $\mathbb{R}^3$ into two components each of which contains interior points of $\mathbb{O}$ which are arbitrarily close to v.

Proof From the proof of Theorem 6 it follows that v is the intersection of two noncolinear line segments, e_1 and e_2, which are unions of line segments of $\mathbb{O}$, are contained in the boundary of $\mathbb{O}$, and contain v as an interior point. Any plane through v not containing e_1 or e_2 is clearly going to contain interior points of $\mathbb{O}$ near v, and in this case this theorem is true. Also, there is only one plane, P, containing e_1 and e_2. If all of $\mathbb{O}$ were to one side of P, there would be a contradiction, since at least four noncoplanar faces would go through v, but all the edges containing v would be coplanar. □

The remainder of this section shows how much simpler things are when items are labeled or when special projections are used. The discussion of the unlabeled case is resumed in Section 3.

In mechanical drawing practice, one generally starts with $\mathbb{O}|XY$, $\mathbb{O}|YZ$, and $\mathbb{O}|ZX$, although it is always possible to use other planes. In fact, as will now be shown, for each object $\mathbb{O}$ it is always possible to find a plane **P** such that π_p distinguishes all the elements of WF($\mathbb{O}$).

Proposition 8
Let $\mathbb{O}$ be an object. Then there exists a plane **P** containing the origin for which π_p projects each element of V($\mathbb{O}$) into a distinct vertex of $\mathbb{O}|\mathbf{P}$, elements of E($\mathbb{O}$) project into distinct line segments which can intersect in at most one point, and no point in V($\mathbb{O}$) projects into a projection of an element of E($\mathbb{O}$) unless it is a member of it.

Proof The set of all planes in $\mathbb{R}^3$ containing the origin can be identified with the unit sphere, S^2, in $\mathbb{R}^3$, where each unit vector corresponds to the plane for which it is a unit normal. Clearly, in this manner exactly two points of S^2 correspond to each plane through the origin. In order for a projection π_p to map each vertex of $\mathbb{O}$ to a distinct member of V($\mathbb{O}|\mathbf{P}$), **P** cannot be perpendicular to any line which goes through at least two of the points of V($\mathbb{O}$) and cannot be perpendicular to any plane containing all edges incident with a Class II vertex. Each of these restrictions rules out exactly one plane, *i.e.*, two points on S^2. Thus, in order to get an injection on V($\mathbb{O}$), at most

$$2\left[\binom{|V(\mathbb{O})|}{2} + |V(\mathbb{O})|\right]$$

points on S^2 must be avoided.

Two elements of E($\mathbb{O}$) can have an intersection of more than one point in some projection if and only if they are coplanar. Furthermore, they can project with a nontrivial overlap only into planes which are perpendicular to the plane containing both of the elements of E($\mathbb{O}$). The set of all planes through the origin perpendicular to a given plane corresponds to a great circle of S^2. Thus, to get the desired behavior at most

$$\binom{|E(\mathbb{O})|}{2}$$

great circles on S^2 must be avoided.

To keep a point from projecting onto a line segment not containing it there are two cases to consider. First, the point and line segment might be colinear. In this case, one must avoid the plane perpendicular to the given line. Again this means avoiding two points. Thus, at most $2|V(\mathbb{O})|\,|E(\mathbb{O})|$ points must be avoided. Second, the point and line segment are not colinear. In this case it is enough to avoid all planes perpendicular to a given plane as before. Thus, at most $|V(\mathbb{O})|\,|E(\mathbb{O})|$ great circles on S^2 must be avoided.

Since points and great circles are nowhere dense in S^2 and the number of sets which must be avoided is finite, it follows from the Baire Category Theorem (see [15]) that there must be points of S^2 which do not lie in any of the forbidden sets. Using any such point yields a plane with the desired properties. □

Definition 9
Let $\mathbb{O}$ be an object, **P** a plane in 3-space, and π_p the projection of 3-space onto **P**. Projection π_p is said to be a *distinguishing projection* for $\mathbb{O}$ if it has all the properties of Proposition 8. □

Note that the proof of Proposition 8 shows that for a given object "most" projections are distinguishing projections since the nondistinguishing ones have a two dimensional measure of 0. The probability of picking a nondistinguishing projection at random is thus zero in an ideal model. However, in most practical situations there are only a finite number of choices for coordinates, and there is a nonzero probability of picking a nondistinguishing projection. Many objects of engineering interest have planar features aligned with the "natural" axes of the object, and the set of three standard views contains a maximum degree of concealment and self alignment.

At this point it is worthwhile to consider two cases. In the first case, the image of each vertex in each projection carries the labels of all the vertices of $\mathcal{O}$ that project into it, *i.e.*, the **P**-projections are labeled. In the second case, there are no labels on the vertices of the **P**-projection.

In the first case there is, quite naturally, significantly less ambiguity than in the second case. The following theorem shows exactly how much information can be recovered from labeled **P**-projections.

Theorem 10
Let $\mathbf{P}_1$ and $\mathbf{P}_2$ be two nonparallel planes in $\mathbb{R}^3$, and let $\mathcal{O}$ be an object. Assume that the $\mathbf{P}_1$ and $\mathbf{P}_2$ projections of $\mathcal{O}$ are labeled. Then there is a unique set of points in $\mathbb{R}^3$ which can be $V(\mathcal{O})$. Furthermore, if either of the projections is distinguishing or if all the edges in at least one **P**-projection are labeled with the pairs of vertices they connect, then $WF(\mathcal{O})$ can be reconstructed uniquely. In this case, reconstructing objects from projections reduces to the problem of reconstructing objects from wire frames.

Proof If $\mathbf{P}_1$-vertices and $\mathbf{P}_2$-vertices are labeled, to reconstruct a point $x \in SV(\mathcal{O})$, the images of x under the two projections are found and perpendiculars erected at those points. Since $\mathbf{P}_1$ and $\mathbf{P}_2$ are not parallel, these perpendiculars can meet in at most one point. Since they both go through x, x can be recovered as their unique intersection point. In this way $SV(\mathcal{O})$ can be reconstructed uniquely, which, by Theorem 6, means that $V(\mathcal{O})$ can also be reconstructed uniquely.

Clearly, if the edges of at least one **P**-projection are labeled as described above, $E(\mathcal{O})$ can be uniquely reconstructed. If one of the projections is distinguishing, $E(\mathcal{O})$ can be reconstructed by joining together two points of $V(\mathcal{O})$ if and only if they are joined together in the distinguishing projection (or in both projections). □

Thus, given a fairly small amount of information on projections, one can quickly and easily reconstruct a unique wire frame. In many practical situations, where the emphasis is on getting things done and not on creating puzzles, it seems quite likely that there will be ample information for constructing the correct wire frame easily. Unfortunately, there will also be many situations with inadequate information. The techniques developed for handling the unlabeled case are of great importance in such situations.

To complete the development of the labeled case, the situation in which there are no distinguishing projections must be discussed. Since this problem is a subset of the unlabeled case, the unlabeled case is considered next.

In the unlabeled case, there can be a number of distinguishing projections and it may not be possible to recover a wire frame uniquely. The following example illustrates this in the case of three distinguishing projections.

Example 11
Let $\mathcal{O}_1$ be the tetrahedron with vertices {(1, 1, 1), (1, 2, 2), (2, 1, 2), (2, 2, 1)} and $\mathcal{O}_2$ the tetrahedron with vertices {(1, 1, 2), (1, 2, 1), (2, 1, 1), (2, 2, 2)}. The projections of $\mathcal{O}_1$ and $\mathcal{O}_2$ into the XY, XZ, and YZ planes are all distinguishing and are identical in each plane, but do not allow construction of a unique wire frame. Actually, the projections into the various planes are all essentially the same, *i.e.*, by ignoring the coordinate which is fixed at 0 in each case, one gets the points {(1, 1), (1, 2), (2, 1), (2, 2)} and the six possible lines between them, *i.e.*, each projection looks like a square with both of its diagonals drawn in. In Section 5, the problem of reconstructing all objects for which all three standard projections look like a square with its diagonals is discussed in more detail. As shall be seen, there are surprisingly many solutions to this problem. □

The above discussion shows that labeling projections can be very useful in reducing the difficulty of reconstructing objects from projections. The truth of the preceding sentence becomes even more apparent after the discussion of the algorithm for reconstructing objects from unlabeled projections in Section 3 and the discussion of the examples in Section 5.

3. Fleshing out unlabeled projections

In order to aid in the comprehension of this rather complex algorithm, a basic form of the algorithm, which accepts only limited data, is presented here (Section 3). The basic algorithm constructs all polyhedral solid objects whose wire frames have a given set of projections (or *views*). The extension of the algorithm to a more general set of projection forms (*i.e.*, overall, detail, and cross section), and to the use of depth information to distinguish between visible and occulted edges, is deferred until Section 4. Since the Projections algorithm is an extension to the Wire Frame algorithm, the basic concepts of the Wire Frame algorithm and its terminology are reviewed first.

In the Wire Frame algorithm the input data [a wire frame, Fig. 2(a)] are processed to find all graphs containing more than two noncolinear edges. For each such graph, minimum enclosed areas are found and nested in a tree hierarchy. From this hierarchy candidate faces with an exterior boundary and possibly interior boundaries (*i.e.*, a face may have holes) are constructed—these are

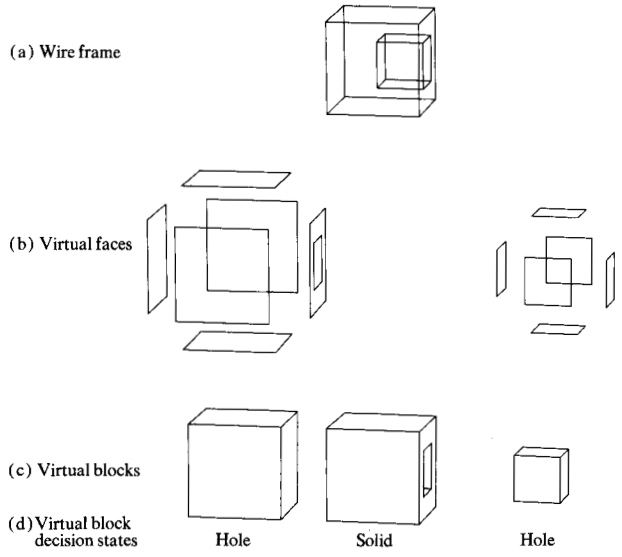

Figure 2 The Wire Frame algorithm in action.

called *virtual faces* [Fig. 2(b)]. For each edge, a list of virtual faces is formed and ordered radially around the edge. Minimum enclosed volumes are found and nested, again in a tree hierarchy. From this hierarchy, candidate volume regions called *virtual blocks* are found [Fig. 2(c)]. A final decision process assigns state solid or hole to each virtual block [Fig. 2(d)], glues the solid blocks together, and finds all possible solid objects with the input wire frame. Note that one virtual block is always an infinite envelope block (*i.e.*, it is inside out) and is always a hole.

The ability to handle all possible cases is embedded in the parts of the algorithm for finding enclosed regions (for example, bridges are ignored), for the handling of illegal intersections between virtual faces (Type I and Type II intersections, see below), and in the final decision process. The correctness of objects is derived from the use of directed edges and faces and from rules governing the number of times and directions with which edges and faces are used.

The several stages of the Projections algorithm are now described. Since many of these stages are quite similar to the corresponding stages of the Wire Frame algorithm, details are given about only those points which are different. The presentation is given in two parts: first, a brief outline of the stages, and second, a more detailed description of each stage.

The early stages (1, 2, and 3) of the Projections algorithm are concerned with converting, by means of a back projection process, a set of projections of an object to a pseudo skeleton and thence to a pseudo wire frame for the object. This pseudo wire frame contains supersets of the vertices of all objects with the given projections. Furthermore, the edges of this pseudo wire frame partition the edges of all objects with the given projections. The existence of various edges and vertices in objects may be known for certain or may be uncertain. All components of the pseudo wire frame are consistent with all the views.

The later stages (*i.e.*, 4–7) apply an extended form of the Wire Frame algorithm to a pseudo wire frame to find all polyhedral solid objects with the given projections.

- *Outline of the Basic Projections algorithm*

1. Check input data The input data to the basic algorithm are assumed to be a set of at least two distinct parallel projections of the wire frame of a polyhedral object. Extensions to handle more general forms of input data are presented in Section 4. The data are checked for validity and reduced to canonical form with edges and vertices distinct and with edges intersecting only in vertices.

2. Construct pseudo vertex skeleton The vertices in each view are back projected to find all Class I vertices (*i.e.*, vertices formed by the intersection of noncoplanar edges) and some Class II vertices (*i.e.*, vertices formed by the intersection of only coplanar edges); at this point it is not possible to distinguish between vertex classes. The vertices discovered here, and the remainder of any Class II vertices missed in this stage and found in Stage 3, are called *candidate vertices*. While not all vertices of $\mathbb{O}$ may be recovered at this stage, enough are recovered to enable the recovery of all vertices after passing through the next stage. Note also that candidate vertices may not be vertices or even points of $\mathbb{O}$.

3. Construct pseudo wire frame The vertices constructed in Stage 2 form a skeleton for the pseudo wire frame in the same sense that WF($\mathbb{O}$) derives from S($\mathbb{O}$). Edges are introduced based on the edges in the projections. These edges are checked for mutual internal intersections. Intersections are introduced as additional vertices and used to partition the edges. The remaining Class II vertices are constructed in this manner. The vertices constructed here and in Stage 2 are the set of candidate vertices (denoted CV($\mathbb{O}$)), and the final set of edges constructed in this stage is the set of candidate edges (denoted CE($\mathbb{O}$)). Together the candidate edges and vertices form the *pseudo wire frame*. The candidate vertices are a superset of V($\mathbb{O}$), and the candidate edges partition the elements of E($\mathbb{O}$). The edge connectivity of all vertices

is examined and the candidate edge and vertex lists edited. The editing process may remove impossible items, simplify colinear edges, and update the classification of vertices as Class I or II. Candidate edges and vertices which are the only possible candidates for some edges and vertices appearing in one of the projections are labeled as certain and must appear in a solution object; all others are labeled uncertain and may or may not appear in solution objects. For both candidate edges and vertices, cross reference lists are maintained between view edges and vertices and pseudo wire frame edges and vertices, and *vice versa*.

4. Construct virtual faces Beginning with the pseudo wire frame generated in Stage 3, all virtual faces are found in a manner analogous to that used in the Wire Frame algorithm. All uncertain edges are checked for containment in at least two noncoplanar virtual faces. Any edges not meeting this criterion are deleted and the virtual faces updated. Any impossible virtual faces (*e.g.*, a certain edge piercing the interior of a virtual face) are deleted. The consequences of deletions are propagated until a stable condition is reached.

5. Introduce cutting edges Illegal intersections between two virtual faces such that both faces cannot exist in an object are handled by the introduction of a temporary *cutting edge* along their line of intersection. The cutting edge partitions the virtual face into smaller independent virtual faces and will be removed in the final stages. All the partitioning processes in the algorithm, be they of edges or faces, generate lists of siblings with common parent edge or face, and also lists of correlations between edges or faces which cannot co-exist in an object; these data structures are used in the final stages of the algorithm.

6. Construct virtual blocks Virtual faces are pieced together to form *virtual blocks* in exactly the same manner as in the Wire Frame algorithm.

7. Make decisions A depth first decision process is used to assign solid or hole state to the virtual blocks and to find all objects with the given projections. The process ensures that all cutting edges disappear in solution objects (*i.e.*, that they are either totally surrounded by space or by material or they separate coplanar surfaces). Efficiency in the search process is obtained by careful pruning of the decision tree, for example, by recognizing that decisions involving partitioned edges and virtual faces may be propagated to the whole original edge or virtual face.

• *Detailed description of the Basic Projections algorithm*

To make the description of the algorithm more comprehensible, the example based on Fig. 1 is used to illustrate the various stages, *i.e.*, the problem is to recover the object in Fig. 1 from its three views. For brevity, this problem is referred to as the Two Wedges problem.

1. Check input data The input data to the basic algorithm are assumed to be a set of two dimensional views of the whole wire frame of a polyhedral object. The views may be at arbitrary projection directions, but must meet a minimum requirement of at least two distinct projections. Each view is an ordered pair of vertices and edges (Definition 4) expressed relative to a local two dimensional coordinate frame and accompanied by a transformation matrix between the coordinate frame of the three dimensional object and the two dimensional view.

In this and later stages, tests are performed on the data input to a stage of the algorithm, for detection of inconsistencies in the data, for reduction of the data to canonical form for the stage, and to obtain information to be used in later stages. The exact choice of which tests to include depends on the characteristics of the input data and performance trade-offs between the cost of performing a test first, the usefulness of information generated for later stages, and the desirability of reporting errors before incurring the cost of executing the algorithm. These issues are not considered further here. However, it will be seen that the combinatorial problems of the projections algorithm may be very severe, and there is therefore a need to minimize the quantity of surplus information generated in the early stages of the algorithm.

2. Construct pseudo vertex skeleton As stated earlier, in this stage perpendiculars are erected at each vertex of each view. Then, only those vertices lying on at least two noncolinear perpendiculars and which are consistent with all other projections, *i.e.*, their images are either vertices or interior points of edges, are selected. As noted after Definition 4, all Class I vertices and possibly some Class II vertices are recovered. In order for the projections to be consistent, it is necessary that every **P**-vertex have at least one element of $CV(\mathbb{O})$ in its inverse image. This check may be performed as part of this stage. In addition, if some **P**-vertex has a unique element of $CV(\mathbb{O})$ in its inverse image, then that element of $CV(\mathbb{O})$ must actually be an element of $V(\mathbb{O})$. Such a vertex is assigned type certain, and all other vertices are assigned type uncertain.

Each intersection is tested to see if it coincides with a previously found vertex and, if not, is introduced as a new vertex. Each vertex found is accompanied by a list of cross references to the view-vertex pairs from which it has been generated. Conversely, for each view vertex, a list is formed of the wire frame vertices into which it projects.

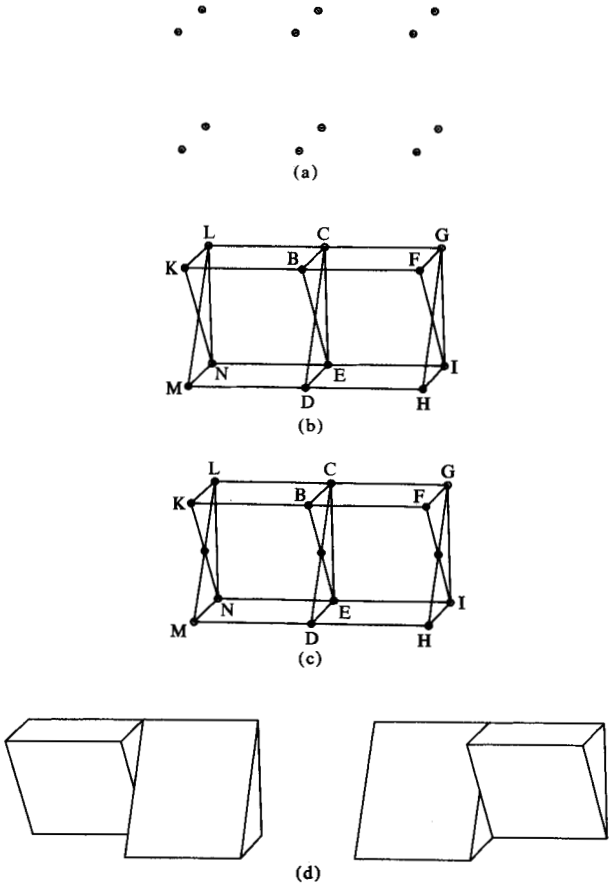

Figure 3 (a) The vertex pseudo skeleton of the Two Wedges problem. Edge recovery in the Two Wedges problem: (b) the pseudo skeleton and (c) the pseudo wire frame. (d) The two solutions to the Two Wedges problem.

The pseudo vertex skeleton of the Two Wedges problem consists of 12 points: the 8 points corresponding to the vertices of a cuboid and 4 points corresponding to the mid-points of the 4 horizontal edges [see Fig. 3(a)].

3. Construct pseudo wire frame In this stage all pseudo skeletal edges are constructed as a prelude to constructing the pseudo wire frame. To do this, simply join two vertices in the pseudo vertex skeleton by an edge iff in every projection the images of these two vertices coincide or are joined by an edge or colinear set of edges and no other vertex of the pseudo vertex skeleton would be an interior point of the edge.

In general, these pseudo skeletal edges may intersect in mutually interior points. To obtain the pseudo wire frame from this skeleton it is only necessary to duplicate the techniques of Theorem 6, *i.e.*, to introduce edges in the obvious way so that all edges have vertices as endpoints, that two edges intersect only in a vertex, and that no vertex be an interior point of an edge.

Note that the proof of Theorem 6 shows that $V(\mathbb{O}) \subseteq CV(\mathbb{O})$ and that every edge of $\mathbb{O}$ can be written as the union of candidate edges.

Many of the checks of Stage 2 are used on the vertices produced in this stage. With modification these checks are used on candidate edges. Thus, it should be verified that every **P**-edge has some element of $CE(\mathbb{O})$ in its inverse image. In particular, if some **P**-edge has a unique inverse image, then that element of $CE(\mathbb{O})$ must be real, *i.e.*, it must actually be an element of $E(\mathbb{O})$ and, like the rule for vertices above, is classified as type certain. At the end of this stage pruning operations are performed. All vertices with edge connectivity of degree ≤ 1 are removed, together with any incident edges. If the vertex has degree 2, the incident edges are checked for colinearity. If they are colinear, the vertex is removed and the two edges are merged into a single edge. If they are not colinear, they are removed together with the vertex. If a vertex of degree ≥ 3 has only coplanar edges, then any edges not having a colinear extension, and possibly also the vertex, are removed. Whenever edges are removed, the effects of the change are propagated until a stable configuration is achieved. In a similar manner to the vertices, cross reference lists are maintained from pseudo wire frame edges to view-edge pairs, and conversely, for each view edge, a cross reference list to the pseudo wire frame edges is formed.

Figures 3(b) and (c) show the results obtained during this stage in the case of the Two Wedges problem. Note that vertex A of the original figure appears in the pseudo wire frame exhibited in Fig. 3(c) but does not appear in the skeleton [Fig. 3(b)]. Note also that by Theorem 7 vertices J and O are clearly spurious since all solid material lies to one side of the planes KLN and FGI. However, these conditions cannot be derived until a later stage of the algorithm. □

The stages described above are fairly straightforward. Before describing the later stages of the Projections algorithm it will be helpful to understand exactly what has been produced so far. The pseudo wire frame $(CV\mathbb{O}, CE(\mathbb{O}))$ looks like a wire frame. Indeed, in many cases $(CV(\mathbb{O}), CE(\mathbb{O}))$ is exactly the wire frame of $\mathbb{O}$ and feeding $(CV(\mathbb{O}), CE(\mathbb{O}))$ to the Wire Frame algorithm will yield the correct solutions directly. The important thing is to understand the way in which simply applying the Wire Frame algorithm to $(CV(\mathbb{O}), CE(\mathbb{O}))$ can fail to find all solutions. The chief problem is that the original Wire Frame algorithm treats vertices and edges as real entities, whereas the pseudo wire frame contains uncertain edges and vertices, any of which may or may not exist in a solution. Any solid object having a *subset* of $(CV(\mathbb{O}),$

CE(𝒪)) as its wire frame and producing the correct projections is a solution of the projections problem. Thus, the Wire Frame algorithm approach may fail to find all solutions of the projections problem (it may in fact fail to find any). The assumption of reality of edges and vertices is crucial to two places in the Wire Frame algorithm:

- Dealing with illegal intersections between virtual faces, and
- Making decisions.

Whenever an edge pierces a virtual face (a Type I intersection) in a legitimate wire frame problem, it is safe to drop the virtual face since it is known that the edge is "real" and that "real" edges cannot pierce faces which separate solid material from space (these are the only important faces). In the present situation, it might very well be that the edge is not real and should itself be dropped instead. Of course, if it is known that a particular edge is real (*i.e.*, certain), the algorithm can proceed as before.

In the Wire Frame algorithm the decision process was concerned with finding those combinations of virtual blocks which made every edge (except the cutting edges) an edge of a real object. In the case of the Projections algorithm it is necessary only to find combinations of virtual blocks with projections agreeing with the given projections. In general, this means that not every uncertain element of (CV(𝒪), CE(𝒪)) is actually a member of (V(𝒪), E(𝒪)). Thus, the decision procedure must be modified to check that every edge in each projection comes from a candidate edge which becomes a real edge in the corresponding solution.

Cutting edges were introduced in [1] to handle illegal intersections between virtual faces when no internal point of an edge from one face was contained in the interior of another face, but there were points common to the interior of both faces (a Type II intersection). This situation was interpreted as one where the two faces could not co-exist in the solution, and temporary edges—cutting edges—were introduced along the line of intersection of the two faces. The cutting edges partitioned the faces into nonintersecting sub-faces, which could be used to build more, smaller, virtual blocks. The decision process ensured that cutting edges did not remain in the final solutions. Although introduced originally for Type II intersections, cutting edges are applicable also to Type I intersections, and are particularly relevant to the case of uncertain edges.

4. Construct virtual faces This stage is essentially identical with Stage 4 of the Wire Frame algorithm. As noted earlier, each candidate edge is checked to see whether it lies in at least two noncoplanar virtual faces. Thus, in the Two Wedges problem, 19 virtual faces [KLO, LON, MON, ABC, ACE, ADE, FGJ, GIJ, HIJ, KLCB, NLCE, MNED, BCGF, ECGI, DEIH, MOLCAD, KONEAB, DACGJH, BAEIJF in Fig. 3(c)] are discovered.

5. Introduce cutting edges This stage is very similar to its equivalent in the wire frame algorithm but has a minor modification to allow for uncertain edges. If an interior point of a certain edge is contained in the interior of a virtual face, then the virtual face cannot be a face of the object and is deleted. All other illegal intersections between virtual faces, *i.e.*, both faces cannot exist in the object, are handled by the introduction of temporary cutting edges. Cutting edges separate virtual faces into independent regions so far as the illegal intersection was concerned and are removed in the final decision process in Stage 7. When a virtual face is partitioned into subfaces, mapping tables and correlation lists are generated in a manner similar to that described for partitioned edges.

Note that if records are kept in the correct manner all reprocessing of virtual faces is done with reference to a particular virtual face, rather than starting with a general wire frame problem. Furthermore, if, when reprocessing a virtual face, f, to determine the smaller virtual faces into which it is partitioned by the cutting edges, a cutting edge, e, is found which is not on the boundary of one of the smaller virtual faces, then it can be dropped together with any virtual face, g, whose intersection with f is e. Face g can be dropped since it is impossible for g to be a member of a virtual block. As usual, dropping a virtual face will in general have other repercussions which are exploited until a stable situation results. For brevity, virtual faces found in Stage 4 will be called *original virtual faces*. Those arising because of cutting edges will be called *new virtual faces*.

In the Two Wedges problem, two cutting edges [OA and AJ in Fig. 3(c)] are introduced. These two edges partition four virtual faces (CADHJG, BAEIJF, MOLCAD, KONEAB) into eight virtual faces (CAJG, ADHJ, BAJF, AEIJ, KOAB, ONEA, MOAD, OLCA).

6. Construct virtual blocks This stage is identical with the corresponding stage in the Wire Frame algorithm. In the Two Wedges problem, six finite virtual blocks are uncovered:

B_1:(MONEAD),
B_2:(NOLCAE),
B_3:(LOKBAC),
B_4:(DAEIJH),
B_5:(EACGJI),
B_6:(BACGJF),

where the description of virtual blocks is in terms of the labeling of Fig. 3(c). The seventh virtual block, B_0, is the unique infinite empty block.

7. Make decisions The set of virtual blocks is fed to a decision procedure, which is an extension of the decision procedure used in the Wire Frame algorithm. The differences between the two procedures revolve around the fact that the Projections algorithm is aware that not every vertex and edge must be real.

The chief difference consists of the fact that whenever the nature of a new virtual face is determined (*i.e.*, whether or not it separates solid material and space), the same determination can be made for all other new virtual faces which are subdivisions of the same original virtual face. Furthermore, as soon as it is determined (or assumed) that an original virtual face, f, does separate solid material and space, all original virtual faces sharing a cutting edge with f are forced to be spurious. This means that any pair of virtual blocks using any part of any virtual face "cutting" f as a common boundary must both be assigned the same state. Similarly, if a virtual face is known to be spurious, all virtual blocks using any part of it as a boundary must have the same state.

These facts speed up the decision procedure considerably and offset the greater number of virtual blocks that have been introduced. Similar arguments apply to entire edges which have been partitioned in Stage 3. In the final solution, no cutting edge can be a real edge. Of course, all decisions respect the fact that the final outcome must be consistent with the original projections.

In this stage virtual blocks are fitted together to generate all objects with the given projections. Basically, each virtual block may have solid or hole state and, when a state assignment has been made to each virtual block, an object is obtained. However, not all assignments of solid and hole yield the desired projections. An assignment of solid or hole to the virtual blocks yields an object with the correct wire frame iff

1. *Every* certain edge element $e \in E(\mathbb{O})$ belongs to two noncoplanar virtual faces f_1 and f_2 each of which belongs to one virtual block assigned solid state and one assigned hole state;
2. *No* cutting edge belongs to two noncoplanar virtual faces f_1 and f_2 each of which belongs to one virtual block assigned solid state and one assigned hole state.
3. *Every* uncertain edge element $e \in E(\mathbb{O})$ may be assigned either to state certain and obeys the rule for certain edges (1) above or to state not-visible and obeys the rule for cutting edges (2) above, in a manner consistent with the input projections.

The decision process is performed by assigning states in a virtual block state vector, whose elements are ordered *a priori*. The first element of the state vector is the unique infinite virtual block, which is assigned the empty state. For each edge, a list is formed of the faces containing the edge and the blocks they bound; this list is sorted around the edge and allows the angular sequence of block state transitions to be discovered.

The decision process proceeds as a depth first search in the virtual block decision space tree. At any node in the tree, the current state vector is checked for consistency and consequential states are assigned. Thus, although the state vector may have dimension of many hundreds, the consistency check may be expected to prune large sections of the tree, while the propagation of consequential states may be expected to reduce substantially the number of decisions to be made.

The checks for consistency are essentially those listed above. The consequential state assignments are performed to meet the following criteria:

- A certain edge with all except one containing block assigned the same state forces the remaining block to be assigned the opposite state.
- An uncertain edge totally surrounded by either all material or by all space becomes nonvisible; an uncertain edge contained in blocks producing exactly two coplanar state transitions around the edge becomes nonvisible; an uncertain edge contained by blocks of both hole and solid states and with at least two noncoplanar state transitions around the edge becomes certain.
- An uncertain edge that is the only edge remaining to create a view edge becomes certain.
- A cutting edge whose surrounding blocks have the same state, *i.e.*, both solid or both hole, spanning regions 180 degrees apart, allows the same state to be assigned to *all* blocks around the edge.
- A cutting edge whose surrounding blocks have the same state <180 degrees apart around the edge allows any intermediate blocks to be assigned to the same state.
- A new virtual face which is a real face, *i.e.*, it separates blocks of different states, and which is a subdivision of an original virtual face formed by cutting edges allows the same solid-hole relationship to be given to all blocks containing sibling faces from the original virtual face. Similar rules apply when the face is not real.
- An uncertain edge which becomes a certain edge and which is a subdivision of an original wire frame edge allows its sibling edges to be upgraded to certain state.

In some cases, particularly those where there are high degrees of symmetry and a limited number of views, giving rise to many highly correlated uncertain edges, there may be a very large number of objects producing the given projections. Thus, although the depth first search and also heuristic search approaches to this problem [8] allow a solution to be found efficiently, an exhaustive search must ultimately be used, and efficient pruning of the decision tree is very important. It is evident that, in the case of problems with multiple solutions, the provision of rather small amounts of extra information by the user, for example, labeling of some uncertain edges, and assigning states to points in 3-space, can resolve the ambiguities completely. Thus, in a practical system, the user may be requested to assist with extra information when requested. The basis for the system requesting extra information in the early stages of the algorithm is the preponderance of uncertain edges, discovered in Stage 3, and self intersection of uncertain edges, dicovered in Stages 4 and 5.

At this point it can be appreciated that the use of cutting edges has allowed construction of a set of virtual blocks having the property that every solution of the projection problem can be built out of the virtual blocks in this set.

Stage 7 feeds into an output module which puts the output together in forms which can be understood by the user of the system. In our implementation of the algorithm, the output is in the form of a polyhedron for the Geometric Design Processor system [2].

In the case of the Two Wedges problem, this stage produces the two solutions shown in Fig. 3(d). The decision procedure works as follows in this case. Suppose that the search in this case deals with the virtual blocks $B_0, \cdots, B_6$ in that order. B_0 is known to be empty. Thus, the first branch of the decision tree corresponds to determining the state of B_1.

If B_1 is assumed to be solid, MOAD is seen to separate solid from space. This means that the entire virtual face MOLCAD must separate solid from space. In particular, B_2 must be solid and B_3 empty. Thus, the next step is to decide whether B_4 is solid or empty. Assuming that B_4 is solid forces B_5 to be solid and B_6 to be empty. However, the object resulting from making B_1, B_2, B_4, B_5 solid and B_0, B_3, B_6 empty clearly fails to have the right projections. Thus, the decision procedure backs up to the B_4 decisions and assigns hole to B_4. This means that the new virtual face DAJH is spurious and that the original virtual face DACGJH is spurious. Thus, B_5 and B_6 must have the same state. If they are both assumed to be empty, the object that results is just a simple wedge, which clearly has the wrong projections. Thus, B_5 and B_6 must both be assumed to be solid. The object that results is a left-right transform of the original object in Fig. 3 and clearly has the correct projections.

On the other hand, if B_1 is assigned hole, B_2 and B_3 must both be assigned the same state. Clearly, if B_2 and B_3 are also empty, it is impossible to obtain the correct front and top views. Thus, B_2 and B_3 must be solid in this case. Furthermore, assuming B_4 to be solid forces B_5 to be solid and B_6 to be empty. This yields the original object. Assuming B_4 to be empty forces B_5 and B_6 to have the same state. The objects that result are both wedges of differing width and are clearly not solutions.

It is clear that keeping track of the number of objects remaining in the inverse image of a projected artifact can be helpful in the decision procedure, *i.e.*, if assigning a particular state to a given virtual block removes the last vertex or edge in the back projection of some vertex or edge, then that assignment can be rejected and its consequences need not be explored further. □

The following section describes ways in which additional information can be extracted from various drawing conventions. The final section contains examples which should clarify the discussion in this and the next section.

4. Additional information from drawing conventions

Designers and draftsmen use a number of conventions and aids to clarify and help reduce ambiguity in engineering drawings. Extensions to the Basic Projections algorithm are presented in this section. These extensions cover two concepts: the generalization of the set of types of views to include overall, detail, and cross sectional, and the use of depth and detail information expressed by line types. The presentation is made within the context of the various stages of the algorithm presented previously.

- *Stages of the algorithm reconsidered*

1. Check input data

In extending the basic algorithm to handle several different types of view (*i.e.*, overall, detail, and cross sectional), the central problem is to be able to relate information from the different types of views. This is achieved here by classification of the edges of the object into two types: gross and detail. The gross edges describe the main structure of the object; the detail edges add more information in regions where there is fine structure in the object.

The edges of the views are labeled with edge types according to an agreed drawing standard. For example,

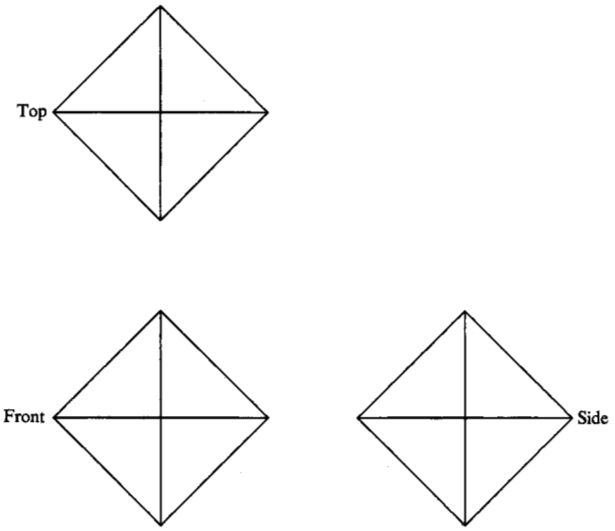

Figure 4 Three views of an object related to an octahedron.

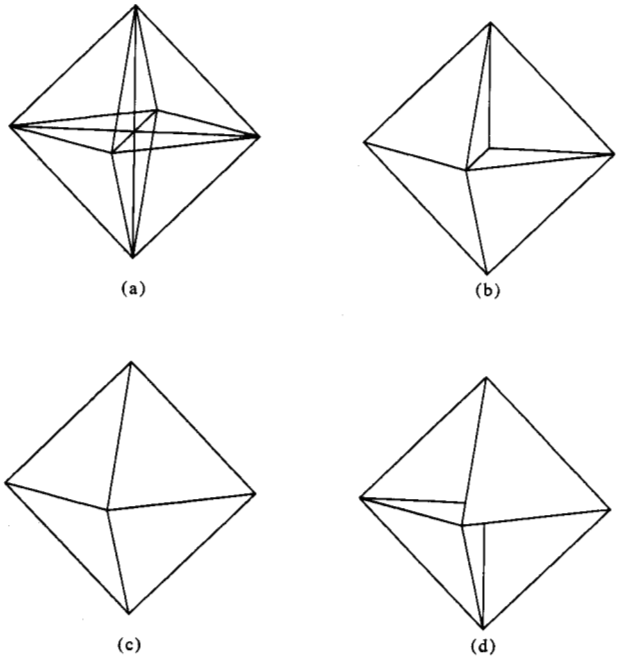

Figure 5 The solution to the problem of Figure 4: (a) the pseudo wire frame; all external edges are of type certain, internal edges are of type uncertain; (b, c, d) some of the 35 solid objects with the views of Fig. 4.

visible edges are generally drawn with line type *solid* and occulted edges with type *dashed*, which provides depth information. Another possibility, namely the omission of occulted edges, is not permitted; the Projections algorithm is based on geometric concepts and the premise that all edges are shown in all projections. An algorithm that attempts to fill in missing information would have to be based on heuristic ideas of what a most likely object would be as well as on the concepts of geometry.

An overall view is a projection of the major features of the whole object onto a plane outside the object. The set of overall views of the object contain projections of only the gross edges. Thus, every gross edge of the object is represented as an edge or a vertex in every overall view. Similarly, every object vertex that is the intersection of gross edges appears as a vertex or a point in an edge in every overall view.

A detail view is a projection of a portion of the object. The view has a defined polyhedral boundary and two extents along the projection direction. The boundary and extents define a right prismatic region in 3-space. The detail view is a projection of all edges and vertices of the object contained in the region. A detail view contains projections of both the gross and detail edges, without distinction, contained within its defined region.

A cross sectional view may be either overall or detail. The view is a planar cross section normal to the projection direction. In this case the view transformation contains the location of the section plane in the coordinate frame of the object. Note that edges are shown at the cross section plane that may not be present in the object (they lie in surfaces of the object), and may not be shown in other views of the object.

2. Construct pseudo vertex skeleton

This stage proceeds in a manner similar to before. However, somewhat greater care must be taken to treat the various projections consistently. Intersections between back projections of vertices from appropriate pairs of different views are considered candidate vertices. Appropriate means noncolinear projection directions and the same type of view, *i.e.*, both overall or both detail. In the case of pairs of detail views, the intersection point must lie within the intersection of their respective prismatic regions. In the case of a cross sectional view, the intersection point must lie in the halfspace defined by the section plane and projection direction. Also, a cross sectional view generates a set of vertices and edges in the plane of the view.

3. Construct pseudo wire frame

This stage is essentially unchanged from Stage 3 in Section 3. However, the following is a very useful observation: whenever a view shows two noncolinear solid (*i.e.*, visible) lines intersecting internally in a point, p, then there must be some vertex of $\mathcal{O}$ visible in the appropriate direction which projects onto p and which has only visible edges incident with it corresponding to

the solid lines incident with p. In particular, if in moving along the perpendicular from p one first encounters candidate vertices which are clearly not vertices of $\mathcal{O}$ (see discussion of Stage 3 in Section 3), then these vertices and all incident edges may be discarded. To appreciate the power of this observation see Example 4.

4. Construct virtual faces
This stage is essentially the same as Stage 4 in Section 3. However, it is possible at this point to use line type depth information to edit out some type II vertices and uncertain candidate edges, as well as to extract additional information for use at a later time.

The cross reference lists from view edges to edges in the wire frame are concatenated with the list of original (*i.e.*, before any partitioning) virtual faces and sorted by distance along the projection direction from the mid-point of the view edge. For any edge that is visible, *i.e.*, not dashed, the nearest pseudo wire frame edge is identified. Any interposing virtual faces cannot exist and are deleted. For an edge to be dashed, there must be at least one occulting virtual face in the projection direction. If there is only one such face, then it must be a real face separating solid material from space, and since the projection is from outside, the directedness of the face is known. This information is fed forward to the decision process as initial certain states of blocks and faces. As before, the consequences must be fully propagated.

5 and 6. Introduce cutting edges and form virtual blocks
These stages are the same as in Section 3.

7. Make decisions
This stage again is very similar to the corresponding stages described in Section 3. Clearly, however, the decision procedure must accommodate the drawing conventions in the correct manner. It is fairly apparent how this is to be done. Thus, for example, in the case that occulted edges are represented explicitly in views, each view edge must contain a visible edge in the view projection direction, and each nonvisible view edge must be occulted by an interposed face in the view projection direction. □

The examples in the next section illustrate the points made above. As shall be seen, pathological features do not appear to be common in objects of practical interest.

5. Examples

To clarify the discussion in Sections 3 and 4, several examples are presented in this section. The examples are chosen to illustrate particular features of the algorithm and some of the performance trade-offs involved in providing extra information.

- *Example 1—octahedron projections*

The octahedron illustrates a simple problem having many solutions, but for which the Projections algorithm does not need to introduce any cutting edges. Figure 4 shows three views of an octahedron. It is interesting to determine the set of all objects having the identical projections. The back projection process generates the 12 edges of the octahedron with type certain and the three intersecting diagonals with type uncertain. In a wire frame example of an octahedron [1] it was shown that the diagonal edges must be introduced as cutting edges for the Wire Frame algorithm to handle the mutually intersecting interior virtual faces. In the Projections case, the algorithm proceeds with no need to generate further edges and enters the decision process with eight virtual blocks, one for each octant around the intersection point of the diagonals. Since the interior edges are of type uncertain and the exterior are all of type certain, any selection of octants such that no two hole octants share a face is a solution. The decision process finds 35 solutions:

1 with all octants solid,
8 with one octant a hole,
16 with two octants holes,
8 with three octants holes, and
2 with four octants holes.

A sampling of these solutions is shown in Fig. 5. Note that in this case dashed lines do not reduce the amount of ambiguity. □

- *Example 2—cube projections*

The cube illustrates a simple use of cutting edges. Figure 6 shows two views, front and top, of a cube. Again the Projections algorithm determines the number of objects having the same two views. The back projection process finds the cube edges, albeit as type uncertain. However, in the direction perpendicular to the two given views, the cube face diagonals are found without intersection. A cutting edge is inserted between the intersection points of the face diagonals, and five virtual blocks are found (the envelope and four quadrant blocks). Five solutions are found as shown in Fig. 7. Note that if all three views of a cube were furnished, there would be a unique solution to this projections problem. □

- *Example 3—Two Y's problem*

Figure 8 shows a well known mechanical drawing puzzle: find all objects having the top and front views shown. Because of the way edges line up in the two views, the back projection process finds the pseudo skeleton with 29 edges and 12 vertices shown in Fig 9. Intersections of the edges yield three additional vertices where the diagonals intersect, and intersections of virtual faces yield eight cutting edges. The final pseudo wire frame is shown in

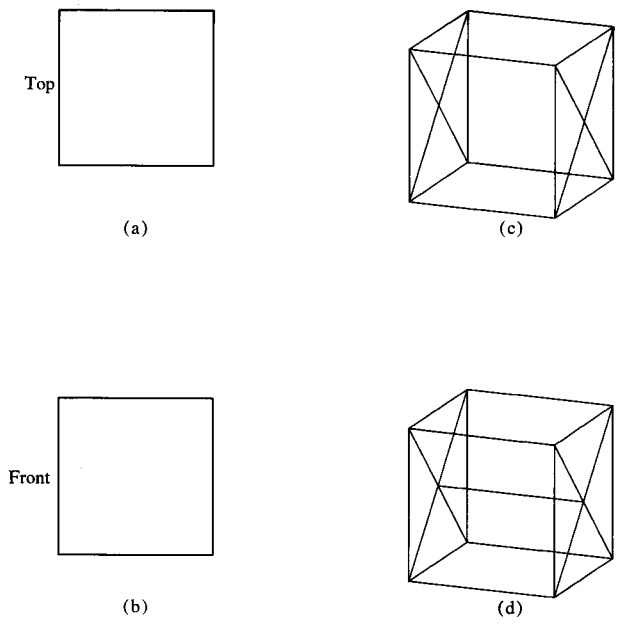

Figure 6 (a, b) Two views of an object related to a cube; (c) the pseudo wire frame; (d) the pseudo wire frame with a cutting edge inserted.

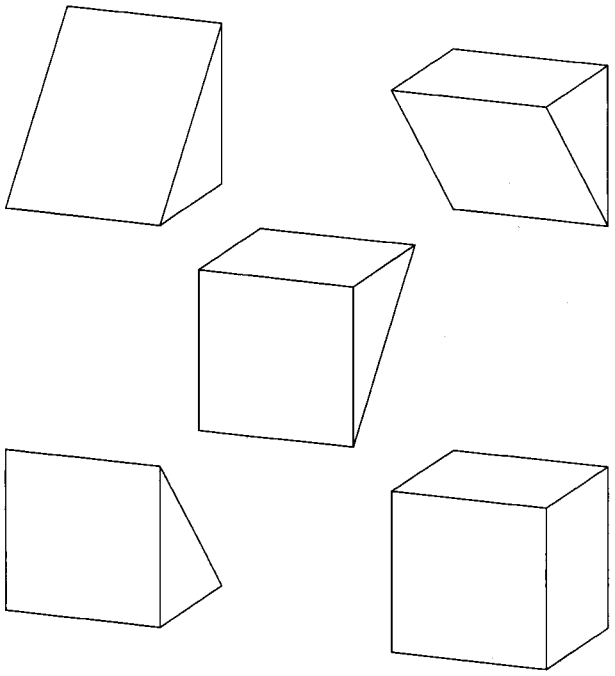

Figure 7 Objects with the views shown in Fig. 6 (a, b).

Fig. 10. The 16 internal virtual blocks found in Stage 7 are shown in Fig. 11. Under the assumptions of Section 3 there are 55 solutions to this problem. Under the assumptions of Section 4 (*i.e.*, all lines in the views are assumed to be solid, that is, visible) there are seven solutions. These are shown in Fig. 12.

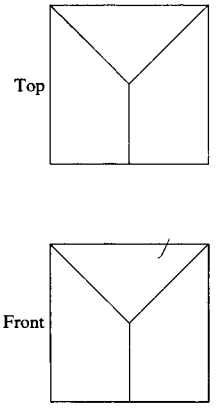

Figure 8 The Two Y's, a well known mechanical drawing puzzle: front and top views of an object.

The Two Y's problem is very sensitive to numerical considerations. If the branch point of one of the Y's is moved from the center of its view, there are no solutions to the corresponding projections problem. □

- *Example 4—Three X's problem*

The Three X's problem illustrates vividly the savings that can result from the use of depth information. Figure 13 shows an apparently minor modification to the problem of Fig. 4; the object is now clearly contained within a cube. However, further investigation shows that the solution process becomes surprisingly complex. The back projection process produces nine vertices—the cube vertices (uncertain) and its midpoint (certain) and thirty-two edges, all uncertain—the twelve cube edges, twelve face diagonal edges (initially with type II intersections, but later changed to mutually exclusive intersections), and eight cube diagonals from the midpoint. Note that, in contrast to the situation with Fig. 4, none of the edges found are of type certain and that ambiguities can be expected to stem from this lack of definite information. The pseudo skeleton that is obtained by back projection is shown in Fig. 14.

In the case without depth information, *i.e.*, all edges drawn regardless of occultation, the partitioning process, of intersecting edges to generate sub-edges and virtual faces to generate sub-virtual faces with cutting edges, divides space into many small regions. A total of 96 internal virtual blocks are found and the decision process uncovers 38 065 solutions. Clearly, searching a 96-level decision tree for 38 065 solutions is a complex process.

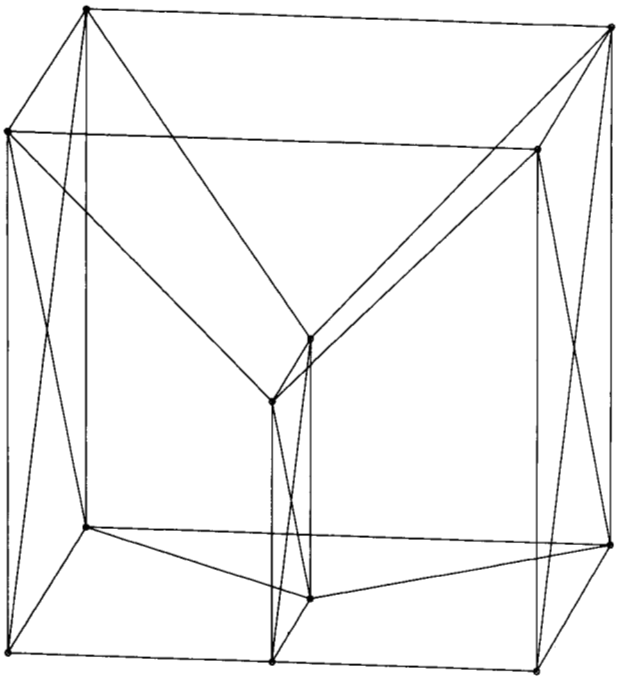

Figure 9 The pseudo skeleton of the Two Y's problem.

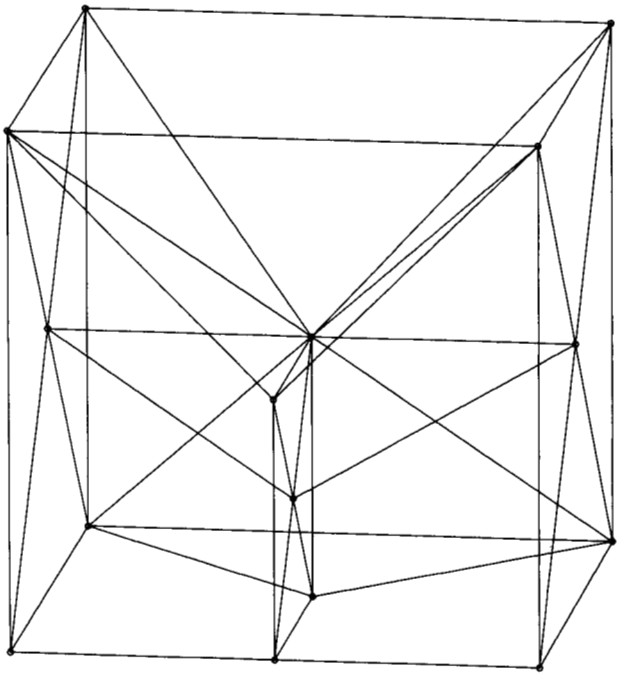

Figure 10 The pseudo wire frame with cutting edges added.

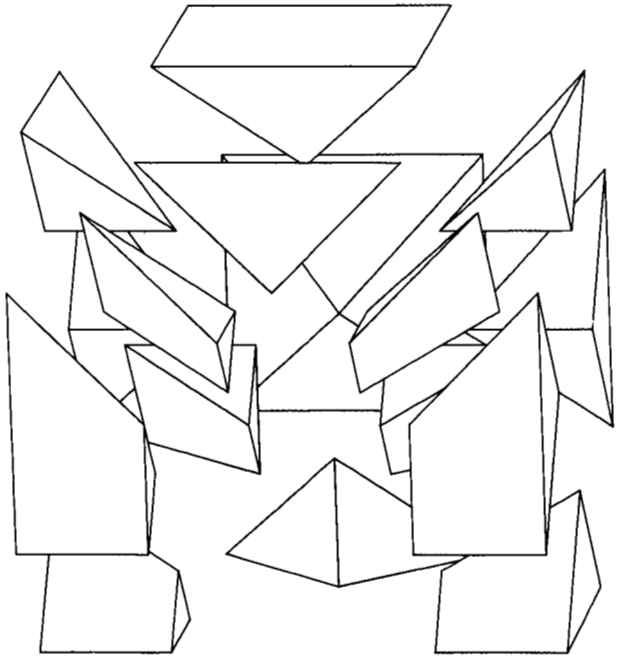

Figure 11 Sixteen virtual blocks found from the two views of Fig. 8.

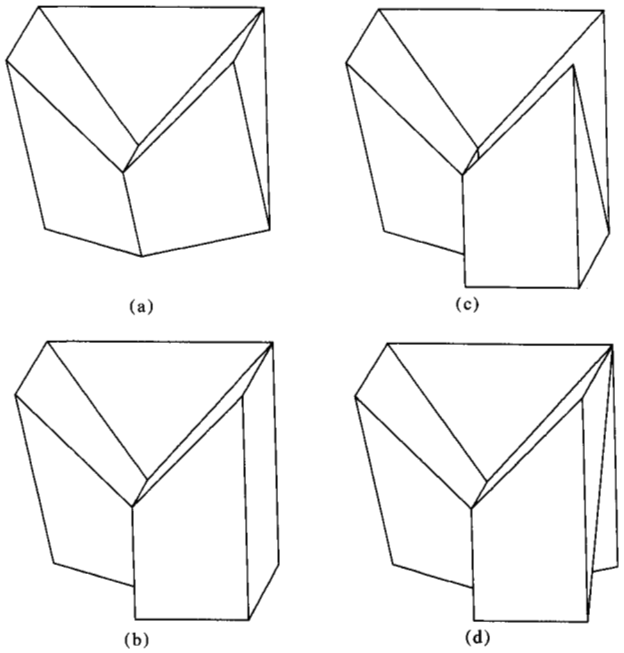

Figure 12 Objects with the two views of Fig. 8: (a) is symmetric; (b, c, d) are asymmetric and each is typical of a pair of objects.

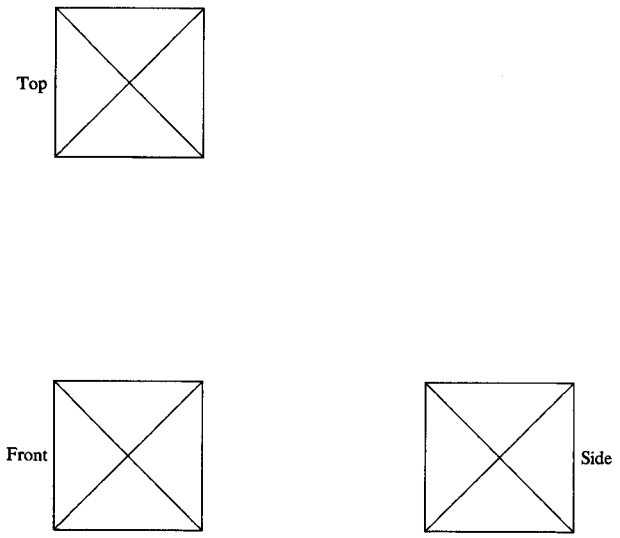

Figure 13 The Three X's problem: three views of an object whose extent is bounded by a cube.

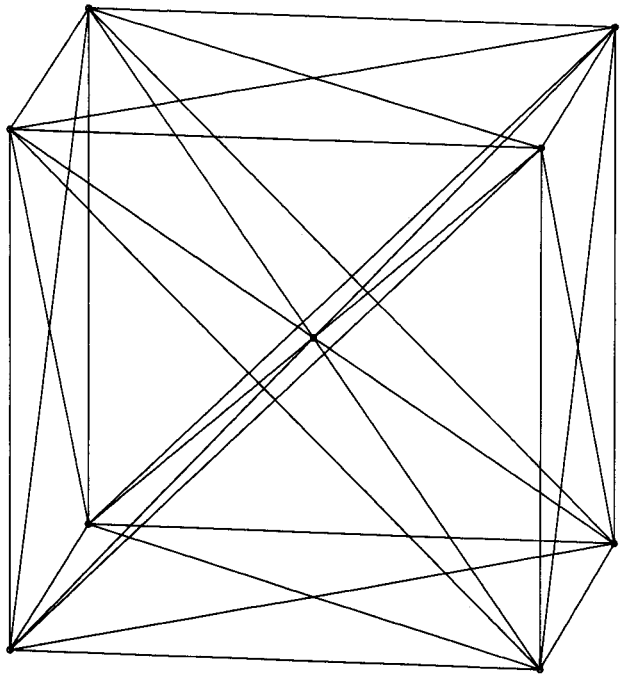

Figure 14 The pseudo skeleton for the Three X's problem.

The solution is made practicable by making heavy use of the mappings and correlations between original faces and edges and their partitioned forms. One solution, picked at random, is shown in Fig. 15. The object is hard to understand, even with a model in one's hand. It is a set of three tetrahedra, a pair with a common face and a third with edge contact only, *i.e.*, it is decomposable into two disjoint objects. The solutions found could be filtered to reject unstable objects of this form, but this test has not been executed.

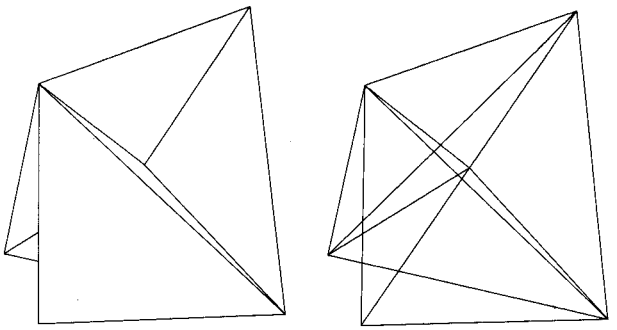

Figure 15 One of 38,065 objects found with the views of Fig. 13 and assuming that all edges are shown in each projection. The object is based on three tetrahedra.

The analysis of the case with depth information shows the power of Theorem 7 when used in Stage 3. Each of the three views shows solid lines intersecting in the center point. Following a perpendicular from any of the center points of any view leads first to a point in the center of a face of the cube containing the pseudo wire frame. This vertex cannot be a Class I vertex since all candidate edges incident with it are coplanar. It also cannot be a Class II vertex since all solid material lies to one side of the plane containing the face in question. Thus, the center points and all diagonal edges may be discarded from the front, top, and appropriate side faces of the cube. Furthermore, in Stage 4 corresponding faces of the cube are also discarded since they would obscure a vertex and lines in the interior. With these faces discarded, three of the leading edges of the cube must be discarded also since they no longer contain at least two noncoplanar virtual faces.

After these reductions, the algorithm goes on to find the ten solutions shown in Fig. 16. The solutions may be considered as being based on the union of three pyramids, as shown in Fig. 16(a). In all solutions, the view of the objects in the projection directions are the four triangular faces of the union of the three pyramids. The distinguishing features between the solutions are cavities in the "rear"; the viewpoint for the solutions in Figs. 16(b)–(g) is chosen to illustrate these cavities. The solutions are grouped as follows:

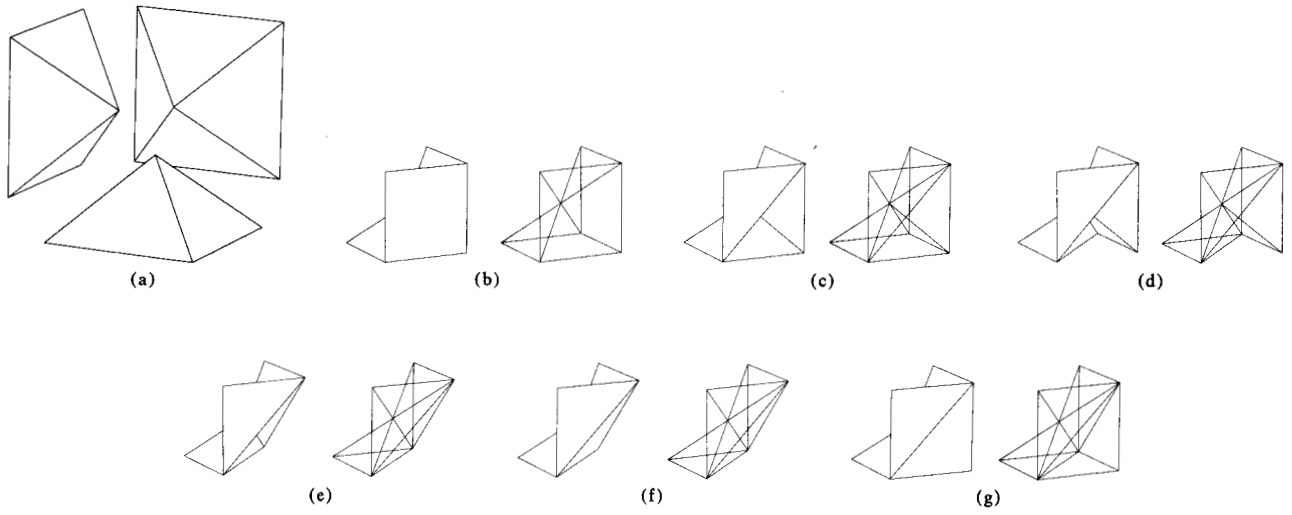

Figure 16 All ten solutions to the Three X's problem in dashed line mode: (a) three pyramids forming the basic solutions; (b) three pyramid solution; (c) one pyramid bisected; (d) two pyramids bisected; (e) all three pyramids bisected; (f) solution (b) cut by plane containing the diagonals of the square faces; (g) solution (b) with an internal tetrahedral cavity.

- Figure 16(b) shows all three pyramids complete,
- Figure16(c) shows one pyramid bisected and is one of a set of three solutions,
- Figure 16(d) shows two pyramids bisected and is one of a set of three solutions,
- Figure 16(e) shows all three pyramids bisected,
- Figure 16(f) shows the object (b) above cut by the plane containing the diagonals of the three square faces,
- Figure 16(g) shows the object (b) with an internal tetrahedral cavity just visible as diagonal edges of the square faces.

- *Example 5—Two Ramps problem*

The Two Ramps problem illustrates the effectiveness of the pruning operations of Stage 3. Figure 17 shows this well known two view puzzle problem reputed to have twelve solutions. The back projection process produces an array of three by four, *i.e.*, twelve, vertices on the left-hand face and an array of two by four for the right-hand face. Twelve edges are found linking the left and right sides. However, the number of possible edges in the end faces, *i.e.*, in the direction normal to the two given views, is large; see Fig. 18. Fortunately many of those in the left-hand face are rejected by the edge and virtual face connectivity test at the end of Stage 3 (Fig. 19). Some 108 internal virtual blocks are found and 107 distinct solutions. Only 12 of the solutions, however, pass the stable object criterion. Some of the solutions are shown in Fig. 20. □

- *Example 6—real engineering objects*

After developing the algorithm to be as general as possible, and proving it with problems chosen for their geometric difficulty and ambiguities, it is refreshing to look at some real engineering objects and consider their reconstruction from their three standard views. Figures 21 and 22, parts (a), (b), and (c), show two examples of engineering objects. Even without using depth information, only one solution is found to each object, and the reconstructed objects are shown in Figs. 21(d) and 22(d). It is apparent from the views that the polyhedral approximations of the cylindrical holes in the objects greatly increase the number of vertices and edges to be handled and that the projections of these polyhedral features can lead to many small edges in the view, indicating potential for numerical problems. However, our implementation of the Projections algorithm does not have problems in these areas with these examples. Further, it is clear that objects of this complexity raise real problems in ensuring the validity of the input data. The three views used as input in this example were obtained from an existing model and were therefore guaranteed to be correct. A human generating these views directly would have some difficulty ensuring their correctness and self consistency. The Projections algorithm in its present form does not attempt to handle incorrect (or incomplete) data. □

6. Summary

The Projections algorithm presented in this paper finds all polyhedral objects $\mathcal{O}$ with a given set of projections. It has

Figure 17 The Two Ramps problem: two views of an object.

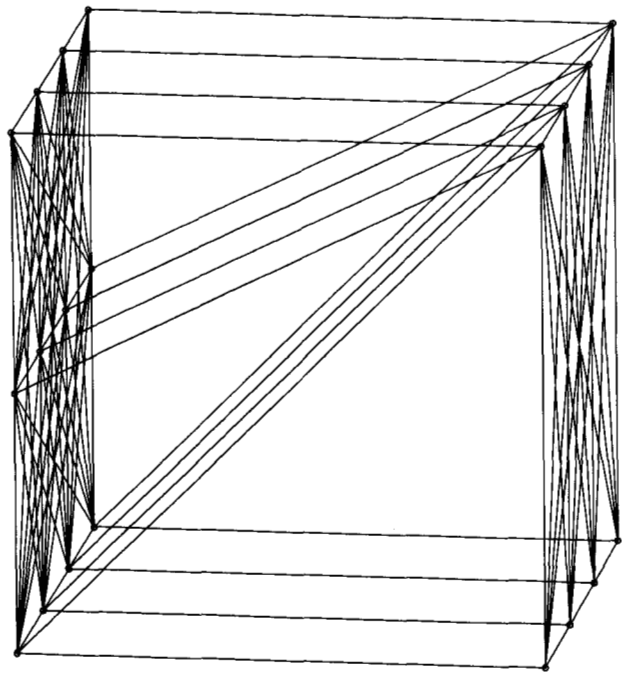

Figure 18 The pseudo skeleton of the Two Ramps problem.

been shown that, if the projections are labeled, the problem may be solved by the Wire Frame algorithm [1]; in the unlabeled case an extended form of the Wire Frame algorithm, the Projections algorithm, is needed.

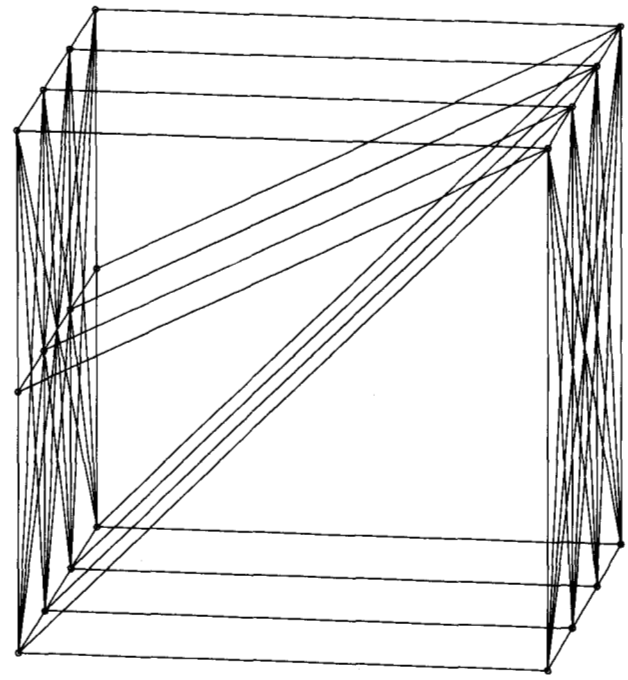

Figure 19 Pseudo wire frame after pruning in Stage 3; note the reduction of edges in the left-hand face.

It has been shown that an inverse projection process may be used to construct a superset of Class I vertices of $\mathbb{O}$ (vertices contained in at least three noncoplanar edges) together with a superset of unions of edges of $\mathbb{O}$. These edges and vertices constitute the skeleton of $\mathbb{O}$.

It has also been shown that a superset of the Class II vertices of $\mathbb{O}$ (vertices contained only in a set of coplanar edges) may be found as intersections of skeleton edges. These updated vertices and edges constitute a pseudo wire frame.

A pseudo wire frame differs from a wire frame in that it contains supersets of the edges and vertices of the wire frame. Some of these elements have been identified uniquely and have type certain; the rest are of type uncertain. Any object whose wire frame is composed of the certain elements of the pseudo wire frame and any subset of the uncertain elements and produces the correct projections is a solution.

The pseudo wire frame is processed to find candidate faces (virtual faces). Virtual faces are connected to enclose volume regions (virtual blocks). A depth first decision process with heavy pruning is used to find all state assignments of hole or solid to virtual blocks that produce solid objects with the correct projections.

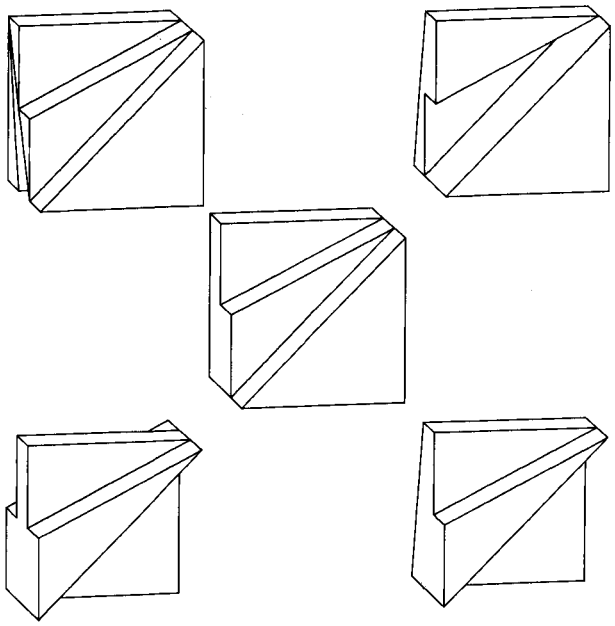

Figure 20 Some of the 107 solutions to the Two Ramps problem.

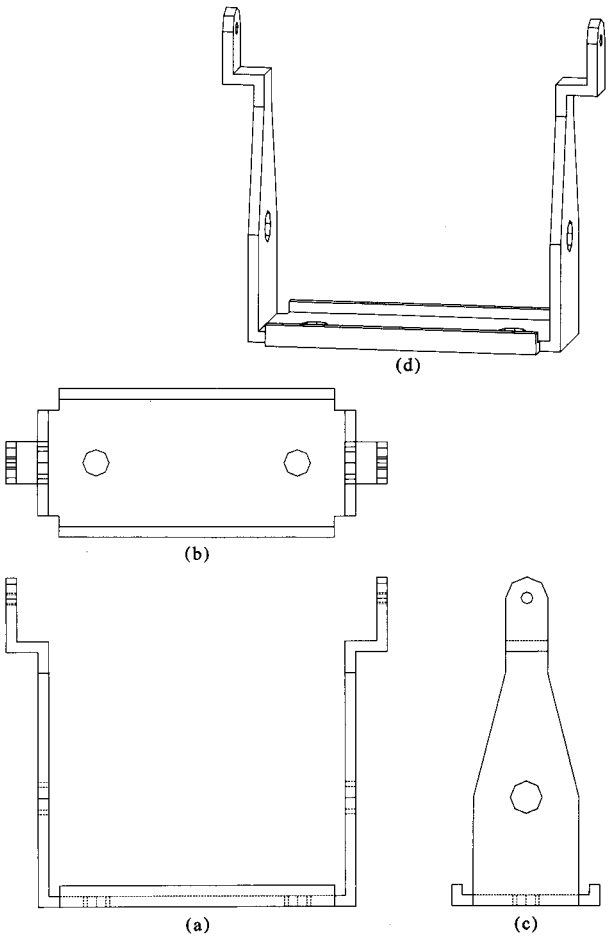

Figure 21 The Projections algorithm applied to an engineering object: (a, b, c) three views; (d) unique object constructed from the views.

The Basic Projections algorithm accepts projections of the wire frame of ℂ; extensions handle a more general set of projection types (detail, overall, and cross sectional) and projection conventions such as depth information obtained from occulted edges in a projection being shown as dashed.

The Projections algorithm has been implemented and its operation has been illustrated by a set of examples. These examples have shown that problems of a mechanical drawing puzzle nature, which typically have high degrees of symmetry leading to large numbers of uncertain elements in the pseudo wire frame, can have very large numbers of solutions. On the other hand, engineering objects, with projections sufficiently complex to require careful thought from a human, have been run and have produced unique solutions.

Acknowledgments

Our thanks are due to D. D. Grossman for encouraging us to tackle the projection reconstruction problem. A. F. Nightingale suggested the Two Ramps problem, and V. J. Milenkovic implemented the stable object test.

References

1. George Markowsky and Michael A. Wesley, "Fleshing Out Wire Frames," *IBM J. Res. Develop.* **24,** 582–597 (September 1980).
2. M. A. Wesley, T. Lozano-Pérez, L. I. Lieberman, M. A. Lavin, and D. D. Grossman, "A Geometric Modeling System for Automated Mechanical Assembly," *IBM J. Res. Develop.* **24,** 64–74 (January 1980).
3. M. A. Wesley, "Construction and Use of Geometric Models," Chapter 2, *Computer Aided Design*, J. Encarnacao, Ed., *Lecture Notes in Computer Science No. 89*, Springer-Verlag, New York, 1980.
4. I. Sutherland, "Three Dimensional Data Input by Tablet," *Proc. IEEE* **62** (April 1974).
5. M. Idesawa, "A System to Generate a Solid Figure from a Three View," *Bull. JSME* **16,** 216–225 (February 1973).
6. M. Idesawa, T. Soma, E. Goto, and S. Shibata, "Automatic Input of Line Drawing and Generation of Solid Figure from Three-View Data," *Proceedings of the International Joint Computer Symposium 1975*, pp. 304–311.
7. G. Lafue, "Recognition of Three-Dimensional Objects from Orthographic Views," *Proceedings 3rd Annual Conference on Computer Graphics, Interactive Techniques, and Image Processing*, ACM/SIGGRAPH, July 1976, pp. 103–108.
8. K. Preiss, "Constructing the 3-D Representation of a Plane-Faced Object from a Digitized Engineering Drawing," *Proceedings of International Artificial Intelligence Conference*, Brighton, England, April 1980.

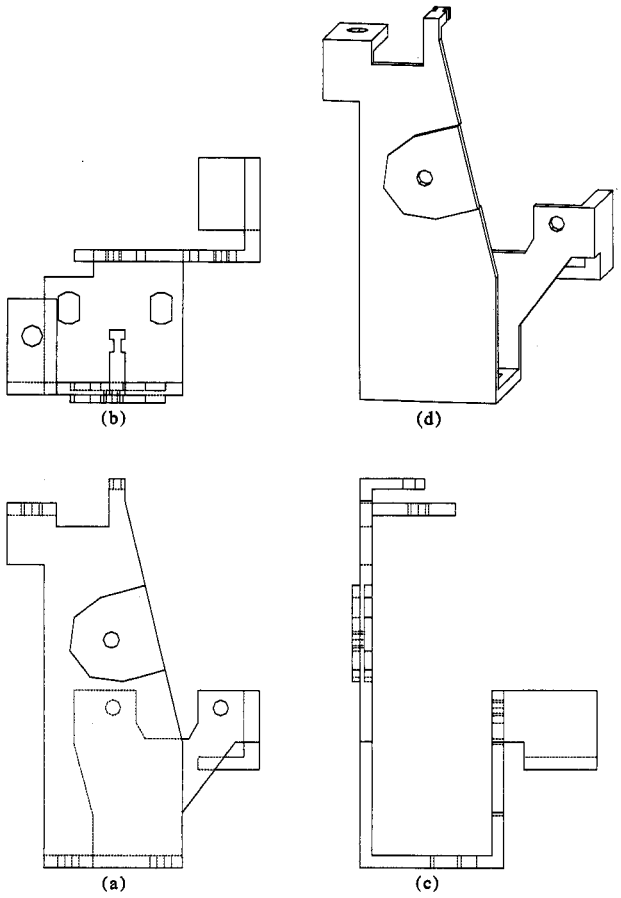

Figure 22 The Projections algorithm applied to an engineering object: (a, b, c) three views; (d) unique object constructed from the views.

9. T. C. Woo and J. M. Hammer, "Reconstruction of Three-Dimensional Designs from Orthographic Projections," *Proceedings of 9th CIRP Conference*, Cranfield Institute of Technology, Cranfield, England, 1977.
10. D. A. Huffman, "Impossible Objects as Nonsense Sentences," *Machine Intelligence* **6**, B. Meltzer and D. Michie, Eds., Edinburgh University Press, Edinburgh, Scotland, 1971, pp. 295–324.
11. M. B. Clowes, "On Seeing Things," *Artificial Intelligence* **2**, 79–116 (1971).
12. D. Waltz, "Understanding Line Drawings of Scenes with Shadows," *The Psychology of Computer Vision*, P. H. Winston, Ed., McGraw-Hill Book Co., Inc., New York, 1975, pp. 19–91.
13. T. Kanade, "A Theory of Origami World," *Report No. CMU-CS-78-144*, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, September 1978.
14. K. Sugihara, "Mathematical Structures of Line Drawings of Polyhedra—Towards Man-Machine Communication by Means of Line Drawings," Third Laboratory of the Department of Information Science, Faculty of Engineering, Nagoya University, Nagoya, Japan, May 1981.
15. J. G. Hocking and G. S. Young, *Topology*, Addison-Wesley Publishing Co., Reading, MA, 1961.

Received April 27, 1981; revised June 2, 1981

The authors are located at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598.