From: SOLID MODELING BY COMPUTERS
Edited by Mary S. Pickett and John W. Boyse
(Plenum Publishing Corporation, 1984)

GENERATION OF SOLID MODELS FROM TWO-DIMENSIONAL AND THREE-DIMENSIONAL DATA

MICHAEL A. WESLEY and GEORGE MARKOWSKY*

IBM Thomas J. Watson Research Center
Yorktown Heights, New York

**Now at Department of Computer Science, University of Maine, Orono, Maine*

ABSTRACT

Many important CAD data bases exist only in wire frame (three-dimensional edge and vertex) or projection (two-dimensional planar view) form. In order to exploit the many advantages of computer-based solid modeling, the data descriptors of the objects in these data bases must be converted to solid form. This paper surveys methods for performing the transformation automatically, describes one polyhedral algorithm in some detail, and explores the degree of automation that is both possible and practicable.

INTRODUCTION

The early evolution of the mechanical design process has been characterized by a transition from *makers-of-things* (craftsmen) to *makers-of-drawings* (designers) [Jones 1980]. This transition allowed a change from the evolutionary build-it-and-see approach of the craftsman to the separation of trial-and-error from production provided by the mechanical drawing. The drawing made it possible to divide up production work, thereby increasing productivity, and made possible the design of things that are too big for a single craftsman.

The two-dimensional drawing as the medium for expressing a designer's concept of a three-dimensional mechanical design has evolved into the stylized art form of today's mechanical drawings with:

- Standardized multiple views of an object showing projections of the object's boundary,

References pp. 46-48

- Auxiliary views showing details and allowing nonstandard viewing directions,
- Coded line types (*e.g.*, heavy, light, solid, dashed) to express concepts such as occultation,
- Symbolic representations of features (*e.g.*, a tapped hole, or the radius of a circle),
- Symbolic representation of machining requirements (*e.g.*, surface finish),
- Dimensions,
- Tolerances, and
- Textual annotations, including "not to scale" and "do not measure."

These stylized representations of objects could, in turn, be interpreted by skilled humans who could envisage, from the two-dimensional drawings, the three-dimensional nature of the object. This visualization, coupled with given dimensions, allowed some manual verification of the correctness of the design and its ability to meet its functional requirements. However, the production of drawings was expensive and itself error prone, particularly as the complexity of the design increased.

The first descriptions of computer aided design (CAD) systems appreciated the ability of a computer to interpret detailed functional semantics of a design. For example, SKETCHPAD [Sutherland 1963] allowed representation of relationships between objects (*e.g.*, a kinematic linkage) or geometric constraints (points to be evenly spaced on a circle). However, the first generation of CAD systems in production use did not exploit this potential and were essentially electronic draughting systems working still with only two-dimensional projections. These first generation systems greatly increased the productivity of designers and, through some additional functions, such as projection of lines between views and the ability to overlay drawings, helped to reduce errors.

A major extension to these electronic two-dimensional draughting systems introduced the third-dimension. Designs could now be described by their three-dimensional edge outlines known as their *wire frames*. Graphics systems offered real time rotation and scaling of images on the screen, and designer productivity was again increased.

The CAD system technologies described above are based on the representation of an object by its natural edges or, in the case of curved surfaces such as spherical or cylindrical, by their silhouette or other artificial edges and attendant vertices. These systems can give high performance visual feedback to the designer for his interpretation of the object. However, the object description data do not contain sufficient semantic information for immediate generation of volume properties of the object. The edge and vertex data do not indicate explicitly the location of surfaces and hence it is not known explicitly where a surface lies and where there is solid material. Thus we are dealing with com-

puter assisted *draughting* systems which constitute the major fraction of installed CAD systems today.

A major increase in the semantic content of the design data base came from the explicit representation of objects as solids [Braid 1974, Baumgart 1974]. Surfaces are now defined explicitly and have associated inward and outward directions; edges have an associated direction to the interior of a face. Data now allow derivation of properties of objects as volumes. For a single object, properties such as volume moments can readily be found. For multiple objects, questions of interference, both static and dynamic, can be answered, and kinematics and dynamics of spatial mechanisms can be analyzed. In principle, the data can be used to generate tool paths for machining and fabrication operations and to generate programs for assembly robots. The combination of representation of objects as solids, and relationships between objects, can allow automatic investigation of all the properties of a product that depend on its volume geometry; this provides the basis for integrating computer-aided design with manufacturing.

A parallel extension to the CAD process is the concept of structured design. A complex object design may be expressed recursively in terms of subcomponents, forming a directed acyclic graph (often a tree) structure with the most primitive objects occurring at the terminal nodes. The combination of hierarchical design approach with primitive volume elements was proposed by Braid [Braid 1974]. Grossman showed the power of writing programs to express the tree structure [Grossman 1976], and Requicha has developed the theory of structured representation of objects [Requicha 1980].

The recognition of the advantages of solid modeling of objects, and the ability to represent them, leads to the problem of the user interface to a solid modeling system. Braid proposed a set of primitive volumes from which more complex objects could be synthesized using operations of volume addition and subtraction [Braid 1974]. Baumgart also proposed objects of translation or rotation of a planar shape [Baumgart 1974]. More recent work has led to solid modeling systems that have extended the class of representable objects by introducing surface patches. These surfaces are of particular interest to the aerospace and automobile industries and allow representation of complex curved surfaces that are at least C^2 continuous.

The discussion above has concentrated on the historical sequence of computer-based design from the viewpoint of mechanical engineering. In other domains different design symbolisms may be used. For example, in architecture, a design may be expressed in terms of elevations and floor plans, with symbolic descriptions of standard components such as doors and windows. Although similar advantages occur from the use of volume representations (*e.g.*, weight, load capacity, bill of materials), the same use of volume primitives may not be directly applicable as the means of design specification. For example, an architectural system may use an outline of the floor plans with symbolic descriptions of wall thicknesses, doors, windows, and so on, as input, and use extrusion op-

References pp. 46-48

erators to go from symbolic design to volume representation [Borkin *et al.* 1978]. Also, an architect may be more interested in the spaces than the solids.

In the domain of mechanical design, analysis, and manufacturing, the importance of solid modeling is firmly established [Wesley 1980]. New design systems are coming into production use that are based on design with, and representation of, solids. However, there remain many existing nonsolid design systems and also extensive data bases that could benefit from conversion to solid form. The existing nonsolid data bases may be in the form of:

- Paper based two-dimensional engineering drawings,
- Computer based two-dimensional engineering drawings, and
- Computer based three-dimensional wire frames.

Hybrid systems that mix nonsolid and solid forms, either as extensions to existing nonsolid systems or *vice versa*, also need to provide tools for conversion to solid form.

In this paper we will explore the problems of conversion of these nonsolid data to solid form. It should be recognized that, quite apart from any inherent algorithmic complexity in the conversion process, the original loss of semantic information incurred by the designer in mapping his solid design concept into a nonsolid data representation leads to problems in the reconstruction process. In particular, a two-dimensional or a wire frame representation does not necessarily correspond to a unique solid object, and reconstruction algorithms must be able to handle the case of multiple solutions.

In the sections that follow, we will first discuss the definition of solid objects, their boundaries and their projections. Then we will compare the existing approaches to solutions, discuss the problems remaining, and, finally, propose an approach to a complete solution.

DEFINITION AND REPRESENTATION OF SOLIDS

Although humans have a good intuitive grasp of the basic concepts of geometry, the algorithms used in computer-based geometry become surprisingly complex when full generality is sought. Formal definitions are needed more to provide a language for precise discussion than to allow detailed mathematical proofs.

The basic definitions of the point sets spanned by a solid polyhedral object and its faces may be taken from the wire frame algorithm given in [Markowsky and Wesley, 1980].

- ***Definition 1.*** A *face* is the closure of a nonempty, bounded, connected, coplanar, open (in the relative topology) subset of E^3 whose boundary is the union of a finite number of line segments.
- ***Definition 2.*** An *object* is the closure of a nonempty, bounded, open subset of E^3 whose boundary is the union of a finite number of faces.

These definitions may be extended to cover objects with "smooth" curved surfaces. These latter are the general class of objects allowed for the discussion below, but it will be seen that most of the work has been restricted to planar surfaced objects.

The aim of these definitions is, on the one hand, to avoid empty sets and dangling edges and faces, and on the other hand, to allow generality in the topological class of objects that can be represented. 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 at edges, and so forth. One can argue that this last case, illustrated in polyhedral form in Figure 1, does not represent a "real" object, but in practice all sorts of strange objects can appear. In particular, in the process of synthesizing a manufacturable design from geometric primitives, the designer may wish to go through intermediate stages that are not manufacturable. Thus, rather than place constraints on the designer, these definitions are intended to handle the most general cases possible.

CSG Trees, Volume, and Boundary Representations—Definitions of objects in terms of point sets are clearly not suitable for direct computation, so other forms of representation are required. Requicha has formalized two approaches: the constructive solid geometry (CSG) tree and the boundary representation

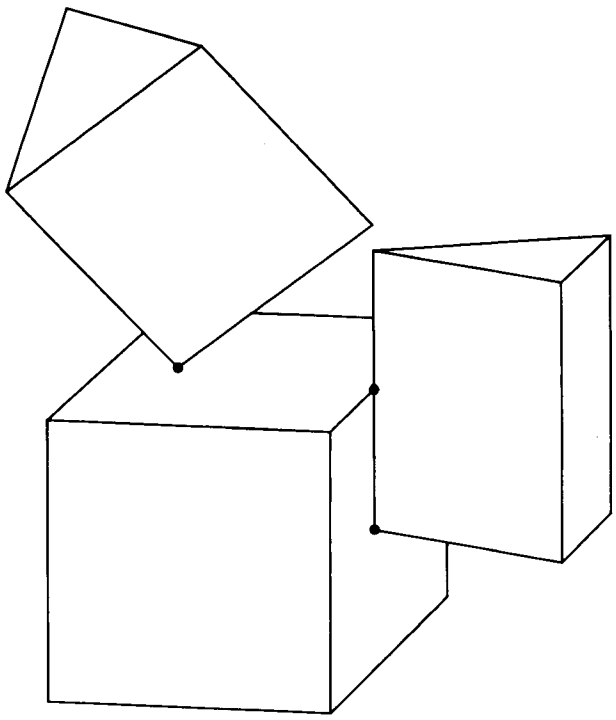


Figure 1. *A typical polyhedral object.*

References pp. 46-48

[Requicha 1980]. The two approaches will be used here for the classification of model representation. A *CSG tree* describes the recursive synthesis of composite objects from subobjects in terms of their relative positions (expressed as rigid body coordinate frame transformations), and primitive point set operations of union, intersection, and difference. A terminal subobject is an instance of a member of a small set of parameterized primitive volume objects. The set of primitive objects may include, for example, cuboid, cylinder, cone, and torus, and may be represented by the intersection of half spaces. The operations of union, intersection, and difference are defined by *regularized set operators* that guarantee that composite objects satisfy Definition 1 and Definition 2 above.

A special case of the CSG tree is volume filling, is generally recursive, and uses a single type of oriented volume primitive. One example, using a cube and binary subdivision, is the octree [Doctor and Torborg 1981]. These representations are approximations to objects with arbitrary surface orientations and are therefore orientation dependent. None of the work on volume reconstruction discussed below is based on volume filling.

The *boundary representation* of a solid object is an object description in terms of its boundary, that is, its faces, edges, and vertices and their connectivity. Clearly the boundary representation must satisfy certain rules in order to be able to represent the object shown in Figure 1 and to reject boundaries that do not enclose volumes, for example, the surface of a Klein bottle.

The basic ideas of boundary representation are that the boundary should be closed (*i.e.*, the boundary cannot have a boundary), orientable (*i.e.*, the object must have a consistent inside and outside), and should not self-intersect. Different authors have taken different approaches to the use of these ideas. For the wire frame algorithm, the concepts of closedness and orientability are embedded in cycles that have insides and outsides. A face is defined in terms of oriented cycles of edges (1-cycles) which, when nested in a tree hierarchy, express the concept that a face has an exterior and may have interior holes. An object is defined similarly in terms of oriented cycles of faces (2-cycles). From a computational point of view, there are local and global conditions to be satisfied. A local condition may be the connectivity in the edge sequence at a vertex in a face, or the face sequence in a solid at an edge. A global condition may be the closure and orientability of a cycle. The condition of no internal self-intersections of the boundary is expressed by tests for illegal (*i.e.*, internal) intersections between faces.

The quantities involved in the representation of geometric objects must be mapped into the data representations of the computer being used. Some quantities may be represented directly and exactly as integers, for example, the relationship that edge-*a* is bounded by face-*a* and face-*b*. These quantities have come to be known as the topological relationships of the object. Other, geometric, quantities that exist mathematically as real numbers cannot be expressed exactly by any finite length numeric data type. Thus we are faced with a choice between either inexact internal representation of the desired mathematical con-

cept or exact (*e.g.*, integer) internal representation of a mathematical approximation. The problem is well known among workers in the field, but has received little attention so far in the literature [Sutherland *et al.* 1974]. The following discussion assumes that floating point approximations to real numbers are being used.

Designers of boundary representation based algorithms for generating solid objects have to deal with the need for global and local consistency and also with the consistency between the topology and geometry of objects. The *winged edge* representation [Baumgart 1974] has become widely used as a data structure for both the geometric and topological components of the boundary representation of solid objects.

Although both the CSG tree and boundary representation allow representation of any object that can be synthesized from the intersections of members of an allowable set of half spaces, there are fundamental differences between the approaches. An object described by a CSG tree is guaranteed to be a valid solid object, independent of the number representation of the machine used. The tree can be mapped into a character string and transmitted to another machine without loss of validity. On a given machine, the object may not be the required object, but at least it is guaranteed to be valid. Computation of the properties of a parent object must ultimately be performed in terms of the terminal nodes and is subject to numerical errors stemming from finite precision representation of any real numbers. The boundary representation of a parent object must be derived recursively from those of the subobjects. A boundary representation is a composite of real numbers (geometry) and integers (topology). Its recursive computation is subject to numerical errors. No system based on boundary representations known to the authors is able to guarantee the validity of an arbitrary object, let alone whether it is the correct object. The basic difficulty arises from the need, when computing the boundary of a composite object in terms of its subcomponents, to make local numerical decisions (*e.g.*, whether a vertex lies above, in, or below a surface) without being able to check the global consequences of the decision. Hence, although an object may satisfy local tests, it may not satisfy a global test. Numerical tests are generally performed with respect to a tolerance. Thus the validity of an object is also dependent on the tolerance used. These numerical issues play an important part in the practicability of converting two-dimensional and three-dimensional information to solid form.

CANDIDATE APPROACHES

Although there have been many partial solutions to the problem of automatic generation of solid object representations from two-dimensional or three-dimensional nonsolid inputs, there is no general and complete automatic conversion process known to the authors. In this section we discuss the various at-

References pp. 46–48

tempted solutions in terms of the approaches used. These approaches taken may be classified in terms of a number of parameters:

- Nature of the input data: single view, multiple views, or wire frame,
- Accuracy of solution: approximate or exact,
- Target volume object representation: CSG tree or boundary representation,
- Provision of clues to volume interpretation: use of labels or constraints on the manner of sequence of input,
- Constraints on class of surfaces handled: planar only or smooth curved, and
- Ability to handle all geometric configurations.

Fortunately, most of the approaches taken by the various authors can be typified by a dominant parameter and will be discussed in the parameter sequence given above. A sampling of the authors in the field and the parameter values associated with their work is shown in Figure 2.

Single View—Early workers in machine vision developed algorithms for deriving the three-dimensional visible surface structure of scenes containing polyhedral objects from camera images. The most successful methods [Huffman 1971, Clowes 1971] involved labelling the edges in the view as convex or concave. In the case of views of scenes with up to trihedral vertices and no cracks or shadows, the number of distinct vertex labelling classes is quite small (*e.g.*, eighteen) [Winston 1977]; relaxing constraints on the scene can expand the number of classes to many thousands. Waltz derived an effective algorithm for propagating local changes in an imperfectly labelled image to achieve global consistency [Waltz 1975].

This work in scene analysis has been used as the basis for construction of solids from single images. The Origami World of Kanade interprets a view in terms of folded paper objects, that is, objects constructed from planar material of zero thickness [Kanade 1978]. Sugihara has used a labelling approach in the formation of solid polyhedral objects from a hand-drawn sketch [Sugihara 1978, 1981]. The sketch is made up of straight lines, with hidden lines marked, and is input during an interative sketching session with a tablet and display. A notable feature of this system is the ability to detect whether a given sketch can represent a valid polyhedral object and, for a sketch that meets this criterion, the ability to reposition vertices to make the sketch correct.

The Sugihara solid generation algorithm uses a linear programming technique to satisfy constraints on components of the boundary representation of the object (*e.g.*, that the vertices for a face be coplanar). The complexity of the largest sketch that can be handled is not stated, but application to highly complex scenes does not appear to be practicable. The author proposes this system for initial, rough input of the overall shape of a design.

Approximate Solutions—Many solid modeling systems have the basis for generating an approximate solution for the two-dimensional projection to vol-

	Sugihara	Kanade	Baumgart, Boyse	Aldefeld	Sutherland	Lafue	Idesawa	Preiss	Markowsky and Wesley	Sakurai and Gossard
Nature of the input data s — single view t — multiple perspective views l — multiple parallel views w — wire frame	s	s	l	l	t & l	l	l	l	l & w	l
Approximate or true solution a — approximate t — true	t	t	a	t	t	t	t	t	t	t
Output object representation b — boundary representation c — CSG tree w — wire frame	b	b	b	c	w	b	b	b	b	b
Clues to three-dimensional structures v — vertex e — edge f — face n — none	n	n	n	n	v & e	f	n	n	n	n
Surfaces p — planar c — curved n — none	p	p	p	c	n	p	p	p	p	c
All geometric configurations y — yes n — no	n	n	–	n	–	n	n	n	y	n

Figure 2. *Sample of approaches to volume reconstruction.*

ume problem, by using functions for forming a constant thickness lamina or extrusion from a planar shape. One of the earliest workers in solid modeling, Baumgart, used the intersection of polyhedral extrusions of each of a number of views to produce a volume object containing the true object [Baumgart 1974]. He showed a rearing horse figure reconstructed by this method from several views. A three orthogonal view example is shown in Figure 3. This approach has also been suggested by Boyse as a first step in an interactive two-dimensional to volume conversion [Boyse 1983].

Generation of CSG from Two-Dimensional Projections—Aldefeld has described ongoing work to construct a solid representation from two-dimensional projections based on identification of members of a set of primitive volumes

References pp. 46–48

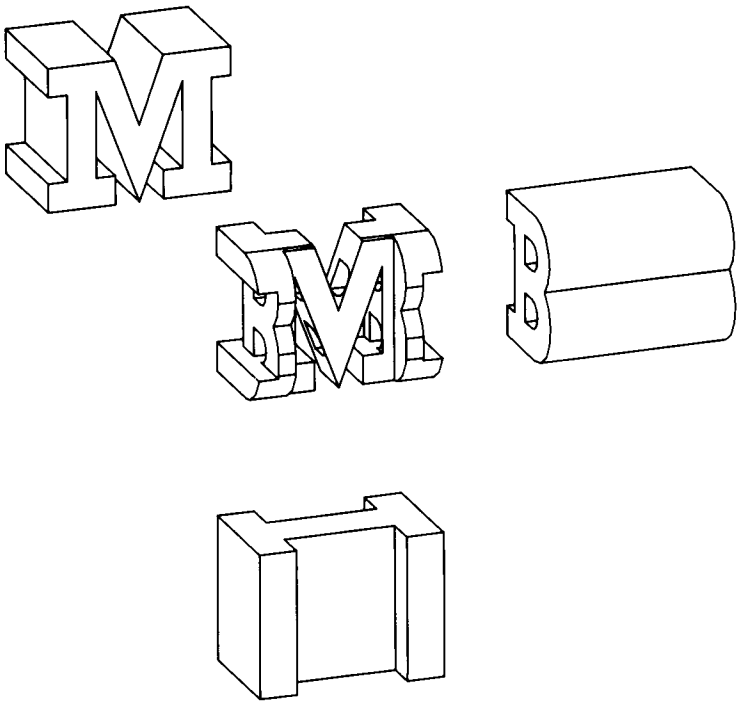


Figure 3. *A volume object as the intersection of extrusions of its views.*

whose combination (at present, only union is allowed) constitutes the object [Aldefeld 1983]. The motivation for the work is that it be incorporated into an existing two-dimensional draughting system.

In the description given, the system is limited to laminate primitive objects that are generated by planar shapes (composed of circular arcs and straight lines) that are parallel to one of the three orthogonal viewing planes. Thus, the class of primitive objects includes cuboids, cylinders, and so on. Note that both cylindrical and planar surfaces are handled but that the constraint on surface orientation is very restrictive. The system proceeds by segmenting the three views and then matches the segmented shapes to find instances of laminae. The relationship between volume object and the laminae may be described as a CSG tree.

The implementation is much simplified by the single class of oriented objects and exploits explicitly the semantics of extrusion to guide heuristic search techniques. Extension to cover negative objects (*i.e.*, difference operations) appears straightforward; extension to general object classes and orientations appears to be difficult, if not impracticable.

Provisions of Clues to Volume Interpretation—A problem arising in reconstructing a solid representation from two-dimensional and three-dimensional

nonsolid data is the correlation of features between multiple views of the object. This process can lead to numerical errors, ambiguity, and computational complexity. Some workers have finessed the problem by requiring the user to give clues to the three-dimensional structures arising from a set of views. Other workers have addressed the full correlation problem and have not required the provision of clues.

Labels—Labelling corresponding edges and vertices in different views avoids multiview cross-correlation operations and leads directly to the construction of the three-dimensional wire frame of an object [Wesley and Markowsky 1981]. Appel and Stein show an early system for labelled input of polyhedra [Appel and Stein 1972]. The complete and automatic conversion of nondigital drawings is seen by the authors to be an impossible task. However, Sutherland described a multi-pen, large tablet based system for interactive digitization of existing engineering drawings; this system could also handle perspective projections as seen in photographs [Sutherland 1974]. Using two or more (up to seven!) pens, the user could indicate corresponding features (points and edges) in several views, both the basic orthogonal views and also auxiliary or detail views. Holders for pens simplified entry of constant values for a view, for example, all the features in a plane. Although not intended for generating solid representations, a three-dimensional straight line wire frame produced by this process could be converted to solid form by the wire frame algorithm [Markowsky and Wesley 1980].

Constraints on sequence of input—Lafue freed the user from some of the rigid and low-level data input requirements of having to identify corresponding points and edges in different views by adopting a *face-wise* approach. The user was required to input the nonnull area projection of each solid face in each view by a complete but unordered sequence of projected edges defining the face; a hole in a solid face was entered by defining a temporary bridging edge to link the exterior to the interior boundary. Thus the low level, multiview edge and vertex correlation problem has been constrained to the simpler, and less ambiguous, problem of cross correlation of faces, though the user now has the responsibility of entering the temporary edges consistently in all views. The three-dimensional wire frame generated in this manner was converted to solid form using a set of axioms expressing the local and global boundary representation rules and a theorem prover. Pathological cases were not covered [Lafue 1976].

Constraints on Classes of Surfaces Handled—In this section we describe the area that has received the most attention and has seen the most detailed implementations.

Planar surfaces—Idesawa elegantly described the general reconstruction problem in terms of projections and inverse projections, leading to an algorithm for the construction of a wire frame and thence a solid [Idesawa 1973, Idesawa *et al.* 1975]. He went on to present a polyhedron implementation with a partially successful heuristic approach to ghosts, that is, vertex and edge elements generated by the inverse projection process that are not part of the boundary repre-

References pp. 46-48

sentation of the target object. A rather complex example is shown. Preiss described a heuristic search approach based on satisfaction of geometric and topological rules, but gave few details and only simple examples [Preiss 1980].

Markowsky and Wesley have designed, implemented, and used two polyhedral algorithms. The first, the wire frame algorithm [Markowsky and Wesley 1980], finds all polyhedral solids with a given wire frame. The second, the projections algorithm [Wesley and Markowsky 1981], finds all polyhedral solids with a given set of orthogonal projections. The wire frame algorithm starts out with a given wire frame, all the components of which must exist in the final object. The projections algorithm uses the given views to generate a superset pseudo wire frame which contains ghost (that is, uncertain) components. An extended form of the basic wire frame algorithm must then handle both certain and uncertain components. Since the projections algorithm is an extension of the wire frame algorithm, the concepts of the wire frame algorithm and its terminology are reviewed first.

In the basic wire frame algorithm, the input data (a wire frame, Figure 4(a)) are processed to find all planar graphs containing more than two noncolinear edges. For each such graph, minimum enclosed areas (1-cycles of edges), using each edge twice with opposite sense, are found; these areas are 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 called *virtual faces* (Figure 4(b)). For each edge, a list of virtual faces is formed and ordered radially around the edge. Minimum enclosed volumes (2-cycles of faces) are found using each virtual face twice with opposite sense. These volumes are nested, again in a tree hierarchy. From this hierarchy, candidate volume regions called *virtual blocks* are found (Figure 4(c)). A final decision process assigns state *solid* or *hole* to each virtual block (Figure 4(d)), glues the solid blocks together, and finds all possible solid objects with the input wire frame. Note that there is always at least one virtual block that is an unbounded envelope block (*i.e.*, it is inside out) and that is always a hole.

The ability to handle all possible cases is embedded in the parts of the algorithm for finding enclosed regions (*e.g.*, bridges are ignored), for the handling of illegal intersections between virtual faces and in the final decision process. The correctness of objects is derived from the use of directed edges and faces, and from the 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 only for those points which are different.

The early stages (1 through 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 pseudoskeleton 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

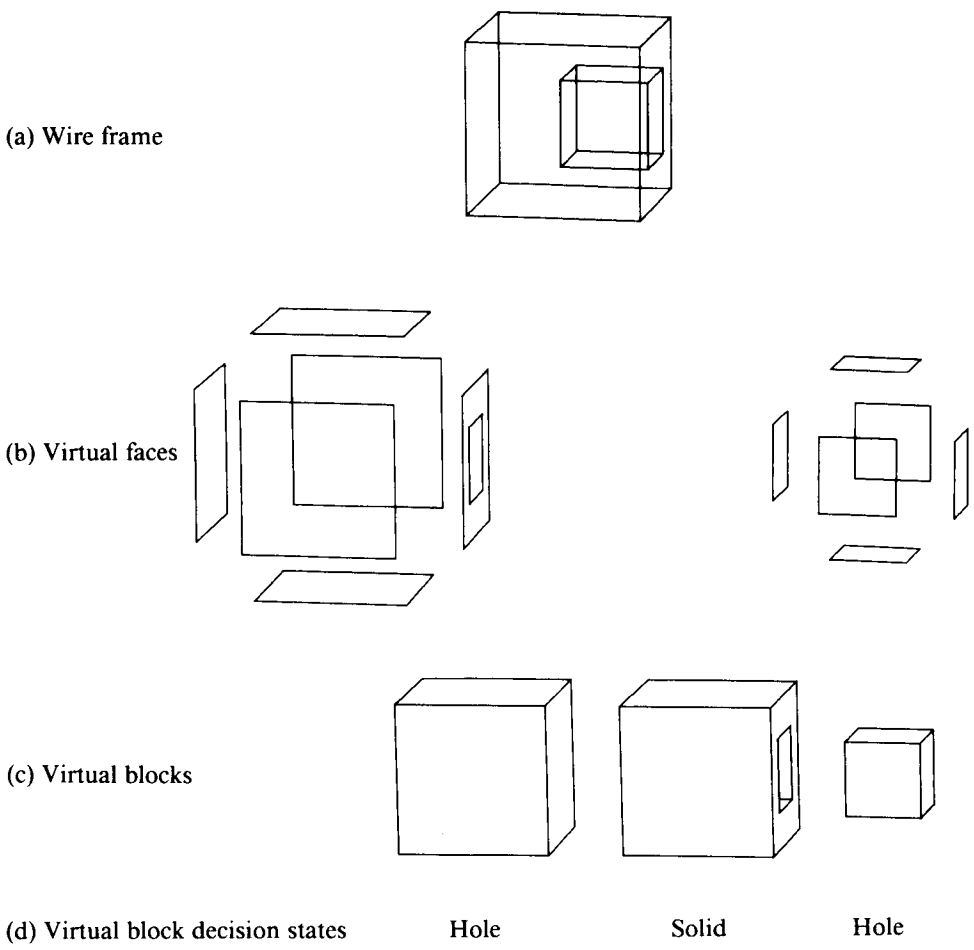


Figure 4. *The wire frame algorithm in action.*

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 through 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.

The stages are now described briefly. They are illustrated by the reconstruction of the two wedges object shown with its projections in Figure 5.

Stage 1: *Check Input Data*—The input data to the basic algorithm are assumed to be a set of at least two distinct perpendicular projections of the wire frame of a polyhedral object. Extensions to handle more general forms of input data such as details, cross sections, and occultations are presented in the origi-

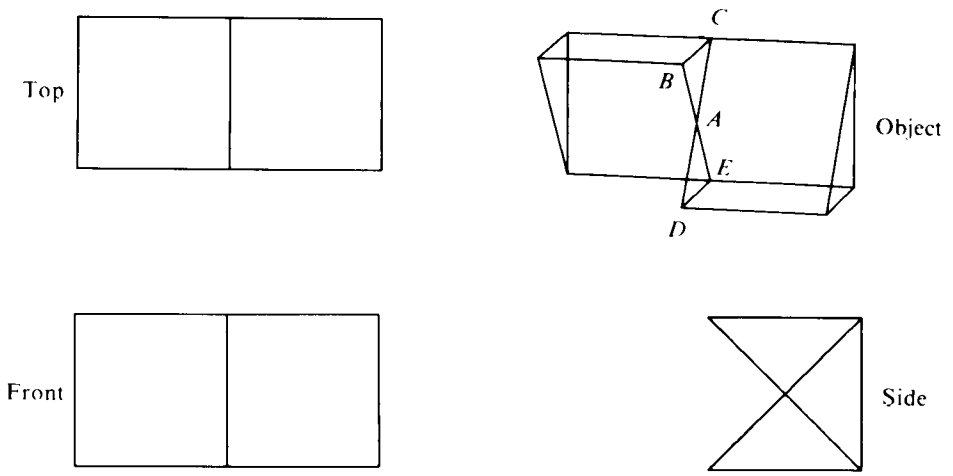


Figure 5. *The two wedges object and its projections.*

nal paper [Wesley and Markowsky 1981]. The data are checked for validity and reduced to canonical form with edges and vertices distinct and with edges intersecting only in vertices.

Stage 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. While not all vertices may be recovered at this stage, enough are recovered to enable the recovery of all vertices after passing through the next stage. The Class I vertices of the two wedges problem are shown in Figure 6(a).

Stage 3: *Construct Pseudo Wire Frame*—The vertices constructed in Stage 2 form a skeleton for the pseudo wire frame. Edges are introduced based on the edges in the projections, as shown in Figure 6(b). These edges are checked for mutual internal intersections. Intersections may introduce additional vertices that are used to partition the edges. The remaining Class II vertices are constructed in this manner, as shown in Figure 6(c). The final set of vertices constructed here and in Stage 2 is the set of candidate vertices, and the final set of edges constructed in this stage is the set of candidate edges. Together the candidate edges and vertices form the pseudo wire frame. Note that candidate vertices (edges) might not be vertices (edges) or even points of the object. The candidate vertices are a superset of $V(Obj)$, that is, the set of all vertices of the object, and the candidate edges partition the elements of $E(Obj)$, that is, the set of all edges of the object. 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

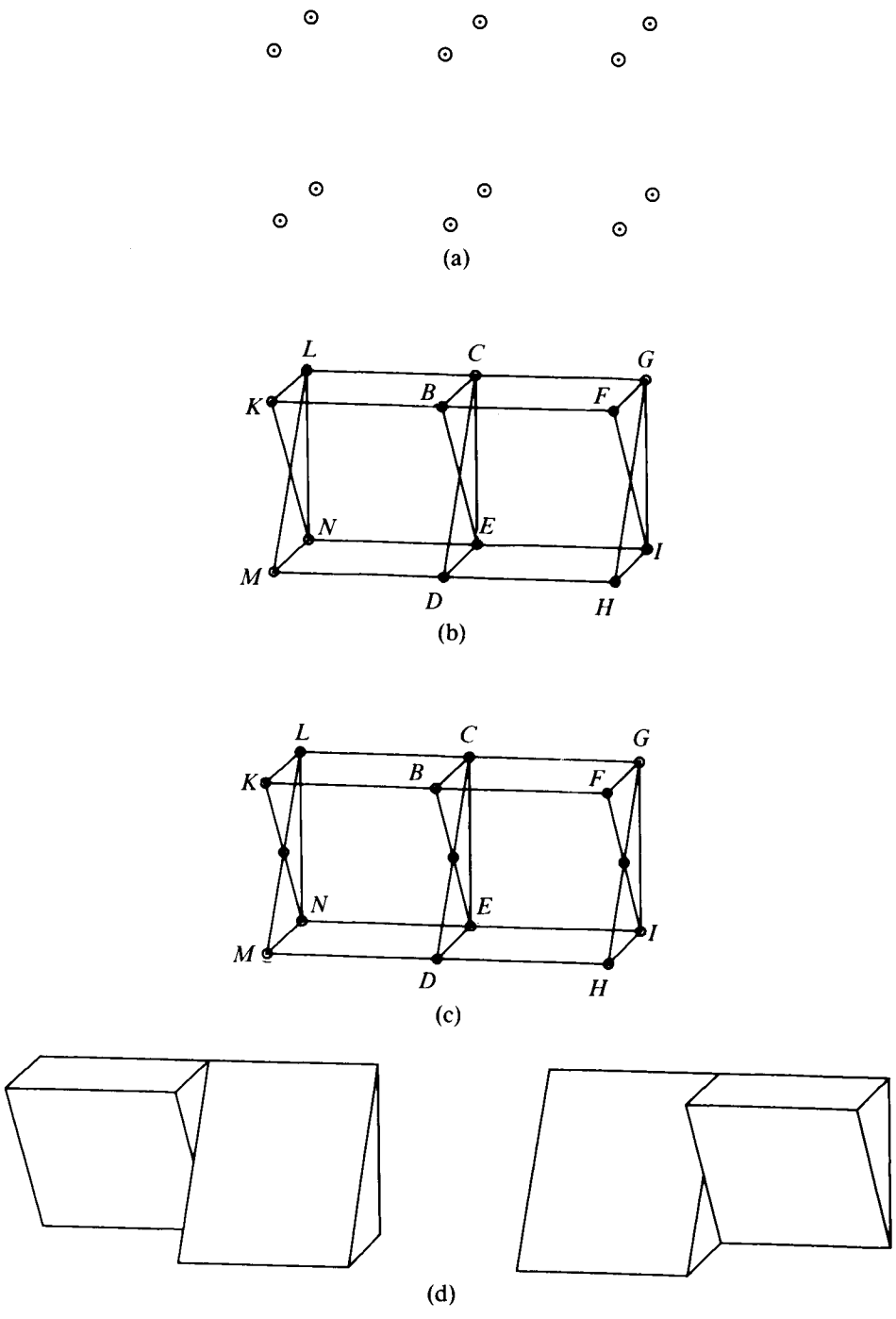


Figure 6. *Stages of the two wedges reconstruction.*

References pp. 46-48

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. In the two wedges problem, all the vertices and edges of the pseudo wire frame are uncertain.

Stage 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.*, any certain edges piercing the interior of a virtual face) are deleted. The consequences of deletions are propagated until a stable condition is reached. Note that coplanar loops such as *BEIF* and *CDHG* in Figure 6(c) will be found as virtual faces. The fact that these particular virtual faces intersect illegally will be recognized in the next stage.

Stage 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* and attendant vertices 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. In order to reduce later computational complexity, 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 coexist in an object; these data structures are used in the final stages of the algorithm. Thus, cutting edges are introduced between the three Class II vertices in Figure 6(c).

Stage 6: *Construct Virtual Blocks*—Virtual faces are pieced together to form *virtual blocks* in exactly the same manner as in the wire frame algorithm. In the two wedges problem, six right-triangular prisms and an enveloping (*i.e.*, inside-out) cuboid are found.

Stage 7: *Make Decisions*—A depth-first decision process is used to assign the state *solid* or *hole* to each virtual block and hence to find all objects with the given projections. This process ensures that all cutting edges disappear in solution objects, that is, either they are 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. The two possible polyhedral solutions found for the two wedges problem are shown in Figure 6(d). For each of the solutions, four prisms are assigned solid state and two are assigned hole state. The enveloping cuboid is always a hole.

A well known mechanical drawing problem is shown in Figure 7, and the stages of its reconstruction are given in Figure 8. The projections and reconstruction of an engineering object are shown in Figure 9.

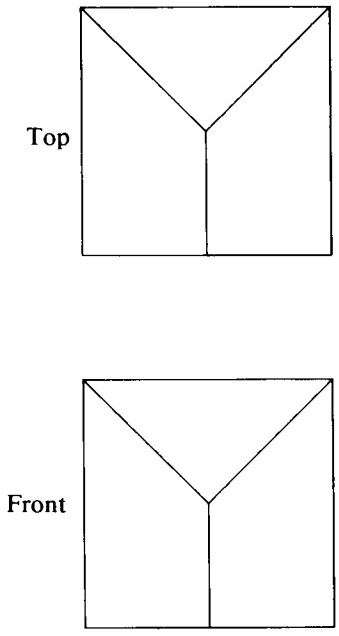


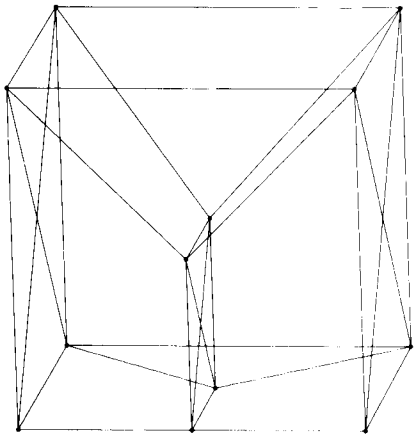
Figure 7. *The two Y's problem.*

Curved and oriented surfaces—Sakurai and Gossard have extended the general polyhedron solution given above to handle certain cases of input data consisting of three orthogonal views made up from straight lines and circular arcs [Sakurai and Gossard 1983.] Thus, the solid may have certain planar, cylindrical, conical, spheroidal, and toroidal surfaces. The data representation used allows an object to have additional implicit nonvisible edges and vertices, such as *tangential* edges (*e.g.*, where a cylinder is tangent to a plane) and *silhouette* edges (*e.g.*, the projected outline of a sphere as seen in a view). The algorithm proceeds by inverse projection of the views and uses edge and vertex type rules to hypothesize the identity, in three-dimensions, of vertex and edge types. This is followed by identification of faces and the assembly of faces into solids to form all possible solids with the views. However, the requirement that the arcs in all views be circular constrains the surface orientations; a further constraint requires curved surfaces to intersect only with planar surfaces.

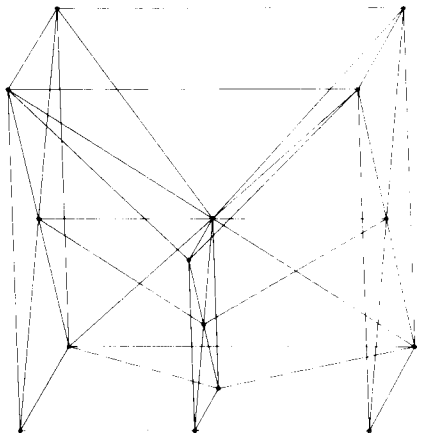
PROBLEMS

In the presence of so many partial solutions, the question of what remains to be done to achieve a full solution may be addressed in terms of several areas of unsolved problems.

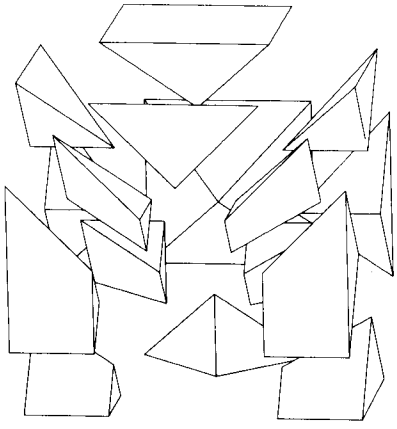
Input Data Problems—When the input data are generated by humans, there is the probability of error arising from several sources. When the data are gener-



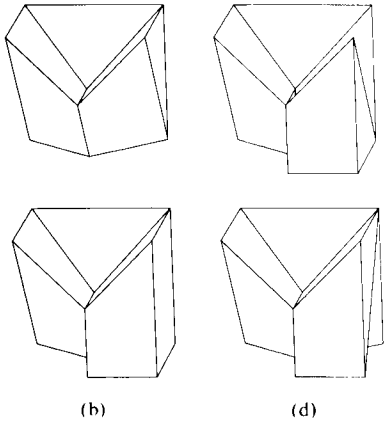
(a) The pseudoskeleton of the two Y's problem.



(b) The pseudo wire frame with cutting edges added.



(c) Sixteen virtual blocks found from the two views.



(d) Objects with the two views.

Figure 8. *Stages of the two Y's reconstruction.*

ated from manually produced engineering drawings, the caveats "do not measure" and "not to scale," frequently found in such drawings, become a major concern. It has been our experience in building solid models using dimensions taken from such drawings, and then generating drawings from the models and comparing with the originals, that positional errors are common. The drawings have the general appearance of the object but rely on the dimensions for the location of features. Further, the feature errors in several views may be inconsistent, so that direct generation of a solid is impossible. A preprocessing phase, using methods such as those of Fitzgerald, may be used to bring the given di-

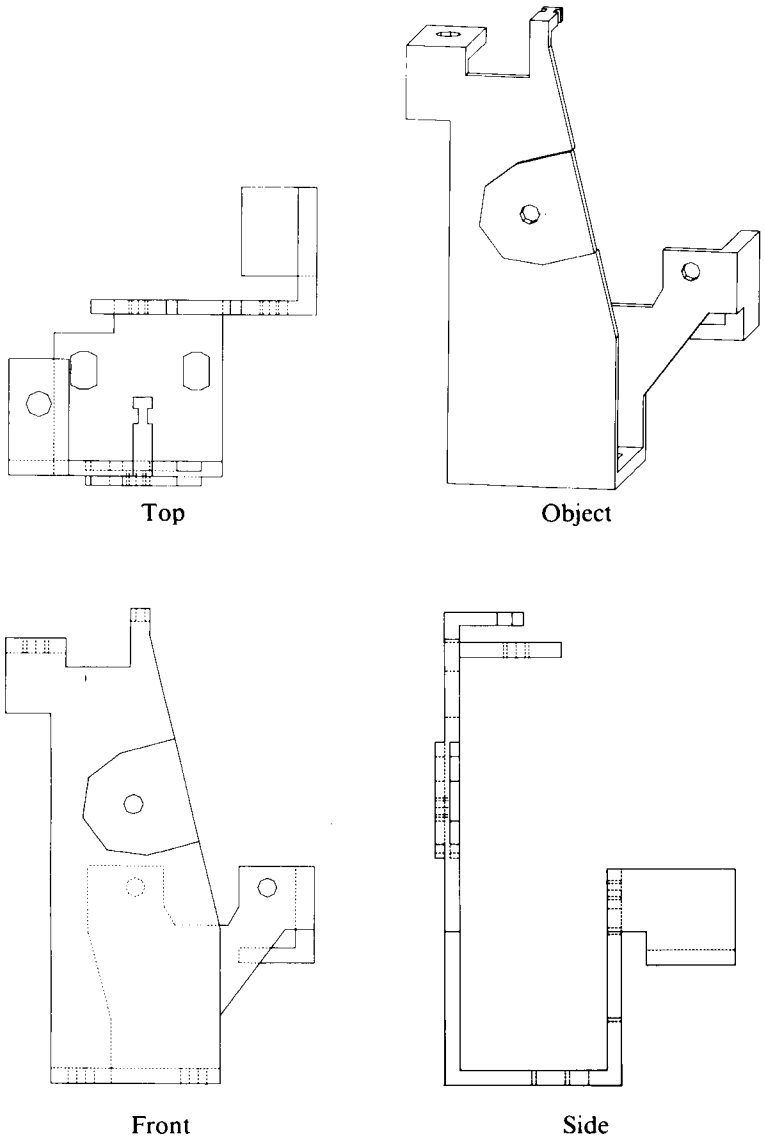


Figure 9. *Projections and reconstruction of an engineering object.*

mensions into accordance with the line and vertex positions [Fitzgerald 1981], at least in a single view. Human intervention will be needed to resolve topological discrepancies.

Electronic drafting systems modify the concept of measurement errors; the drawings *are* to scale to the resolution of the representation used. The human still has to decide where to put lines and where to omit them. In a complex ob-

References pp. 46-48

ject the probability of the human's being able to guarantee that the views are correct, complete, and consistent is still very low.

Numerical Problems—Numerical errors cause severe problems in any boundary representation modeling system when complex objects are involved. In the reconstruction problem, algorithms try to establish global properties from a sequence of local properties. For example, in the case when no three-dimensional clues are provided, the existence of a planar surface may be derived from the close numerical agreement of cross products (*i.e.*, normals) at adjacent nodes of an edge connectivity graph. In the presence of numerical error, this could lead to the entire upper surface of a finely divided geodesic dome being considered planar. Modeling systems generally have considerable difficulty with the numerical aspects of boundary representations even when there is higher level semantic information available (*i.e.*, parametric definition of a surface type with parameters to the precision of the system number representation). Numerical problems in reconstruction are not likely to be solved before the general numerical representation problems are contained. One CSG approach [Aldefeld 1983] may be more successful than the boundary representation approach for numerical reasons.

Computational Complexity—None of the published algorithms are accompanied by treatment of the computational complexity involved. In order to give a feel for some of the computational complexity issues, we give here a brief discussion of the stage-by-stage time complexity order of the projections algorithm for polyhedral objects. These measures are summarized in Figure 10.

Stages of the algorithm	Time complexity
1. Check input data V— number of vertices E — number of edges	maximum ($V\log V$, $E\log E$)
2. Construct pseudo vertex skeleton V — number of vertices N — number of views	less than V^N
3. Construct pseudo wire frame E — number of edges	less than E^2
4. Construct virtual faces D — vertex edge degree E — number of edges	maximum (D^2, E)
5. Introduce cutting edges F — number of virtual faces	less than F^2
6. Construct virtual blocks B — number of virtual blocks	B
7. Make decisions B — number of virtual blocks	less than 2^B

Figure 10. *Complexity of stages of projections algorithm.*

Stage 1: *Check Input Data*—The complexity of most of the tests in this stage is fairly low. The overall complexity of the stage depends on which tests are implemented. The following apply to input data with V vertices and E edges.

- Checking for duplicate vertices and edges in each view can be done in time V^2 and E^2 if a straightforward comparison is used, in time $V \log V$ and $E \log E$ if techniques based on sorting are used, or in time V and E if a hashing technique is used.
- Checking edges for nonvertex intersections requires time E^2 if done with brute force by looking at every pair of edges. The time complexity can be reduced significantly by partitioning space into zones and checking pairs of edges that are both in the same zone.

Stage 2: *Construct Pseudo Vertex Skeleton*—The time required for back projecting all the vertices depends on the number of vertices in each view. If done simply, this operation would involve taking all N-tuples of points, where N is the number of views, and seeing whether a point exists in 3-space that projects all the points in the N-tuple. A much better approach involves considering only points from each view that have a chance of intersecting in 3-space. For example, no point in 3-space could possibly project into (3,4,0) and (0,5,5) if projections are parallel to the Z and X axes. For projections parallel to the axes, it is easy to partition the vertices in each view into groups that have a chance of intersecting with vertices selected earlier in the N-tuple.

Stage 3: *Construct Pseudo Wire Frame*—It is hard to estimate the overall complexity of this stage because it is hard to say anything about how the edges in the different views will cooperate to create edges in the pseudo wire frame. For most objects this stage proceeds rapidly, because there are few sets of lines in the projections that are the images of the same line.

The intersection of pairs of edges can take E^2 time if all pairs are checked for intersection. If the pseudo wire frame is decomposed into geometrical zones this will proceed much faster.

Stage 4: *Construct Virtual Faces*—This stage proceeds rapidly. At each vertex of the pseudo wire frame, all pairs of edges are examined, and a face containing them is sought. The complexity of this step at each node is of order D^2 where D is the degree of the vertex. This is quite a bit better than working with all possible pairs of edges.

Once a pair of edges, E_1 and E_2, is selected, it is used to define a plane, P. The algorithm searches the edges of the pseudo wire frame that are contained in P for an oriented cycle beginning with E_2 and ending with E_2. If there are several edges in P at a vertex, the algorithm chooses the next left-most edge (relative to the orientation chosen at the starting vertex) and proceeds. Since the choices are forced, no trees are found and the algorithm quickly completes its search.

References pp. 46-48

Stage 5: *Introduce Cutting Edges*—The simplest way to introduce cutting edges is to check all pairs of virtual faces for intersection. This approach is time consuming because of the large number of faces involved. Once the edges are found, all the points of intersection among the cutting edges in each virtual face must be found. Clearly, the amount of time required depends on the number of edges and faces involved. Some refinements based on geometrical zones can be made to reduce the computation required. Cutting edges are a problem primarily with highly symmetrical objects, such as the ones that arise from posed problems. They occur less frequently in realistic objects.

Stage 6: *Construct Virtual Blocks*—The complexity of finding virtual blocks is linear in the number of virtual faces. If there are no contradictions in the objects, every virtual face will be in two virtual blocks. The algorithm finds the blocks by starting with a face and adding to it neighboring faces, trying to construct a two-dimensional polygonal surface such that crossing an edge always leads to another surface. Every virtual face belongs to at most two virtual blocks, and thus there is little searching to do since, at each edge, the choice of virtual face is forced. Virtual faces that belong to only one virtual block can be dropped, since they cannot be real faces of the object.

Stage 7: *Make Decisions*—Finding all solutions to the original problem is an exponential problem, since there can be exponentially many solutions even with three views of the object. With most real objects, the search proceeds very rapidly, especially in the cases where there is only one solution.

Our experience with the execution of this algorithm has been that the introduction of cutting edges in Stage 5 is itself computationally expensive and can also lead to a large number of artificially small potential edges and virtual faces. These in turn lead to a large number of virtual blocks. An exhaustive, depth-first search to assign the state *solid* or *hole* to virtual blocks is exponential in the number of blocks; the implementation relies on heavy pruning (*e.g.*, propagation of the decision properties of a partitioned edge to the whole parent edge) to reduce the computational complexity to manageable level. Observation of the block state assignment process in action has shown that once a state assignment decision is reached for a single block, the consequences generally propagate extremely rapidly. The provision of volume clues by the user (*e.g.*, the assertion that a spatial point is within material) should speed up the initial block decision process considerably.

Although the multipliers have not been discussed, the time (and, in fact, space) complexity of these algorithms is not a serious limitation on practicality. Even the exponential complexity of the final decision process has yielded to the pruning operation, and highly symmetric views generating about one hundred virtual blocks have been solved quickly.

Auxiliary and Symbolic Data—Apart from the Sutherland digitization procedure [Sutherland 1974] and the projection algorithm, none of the workers ad-

dresses the inclusion of data from auxiliary and detail views on two-dimensional projection inputs. This is not intrinsically difficult, but it is required for a full solution.

Treatment of symbolic data, such as description of a tapped hole or radius of a circle, is a much more difficult problem. It requires a knowledge-based approach with domains that include geometric relationships such as tangency [Light 1982], machining processes such as tapping, and mechanical drawing techniques and standards.

APPROACHES TO A USABLE SOLUTION

In view of all these problems, together with the need to resolve ambiguities, a near-term usable solution will have to be interactive. In this section we discuss a possible overall approach based on integration of the individual candidate approaches presented earlier in this paper.

Rather than risk absorbing all the data and either dying under the computational load or, after extensive computation, reporting that no solution is possible, a successive refinement approach is favoured with the system doing the detail work and expecting high-level guidance from the user. The first step will be to obtain an approximate volume shape into which details may be introduced later. The approximate solution approach of intersecting extrusions of the projections will bound the object space and give the user a feel for the overall shape. Alternatively, Sugihara's sketch input approach [Sugihara 1978] would also provide a volume starting point.

In order to reduce the computational load, the user will be able to guide the detail process. Onto the initial volume, the system will overlay additional three-dimensionally derived edge and vertex data. These data will already have been pruned to satisfy consistency tests (*e.g.*, a valid edge must separate an even, nonzero number of faces). Any further interactive pruning by the user can be expected to propagate and reduce the uncertain vertex, edge, and face sets. Assertion that points are in material will have a similar effect. Numerical errors can be reduced by user assertion of surface membership and even high level properties.

In principle, input data errors may be observed by the user and corrected. The localization of errors is a hard problem. An error may cause a global consistency test to fail at a point far from the originating error. In this case, the whole global net will have to be shown to the user for study and data error diagnosis.

The interactive approach also allows progressive handling of symbolic data. Initially, the conversion system will require the user to interpret the symbolic data; later, if the system acquires expert capabilities, the symbolic data can be processed more automatically. Invention is needed to handle a wider and less constrained class of surfaces. As a first step the work of Sakurai and Gossard

References pp. 46-48

must be extended to remove orientation and intersection constraints on curved surfaces [Sakurai and Gossard 1983].

SUMMARY AND CONCLUSIONS

In this paper we have examined the many partial solutions to the problem of constructing volume models from two-dimensional and three-dimensional non-solid data. Although none of these are in themselves complete, an evolutionary and interactive approach to a useable solution is proposed.

REFERENCES

[Aldefeld 1983]
B. Aldefeld, "On Automatic Recognition of 3D Structures from 2D Representations," *Computer-Aided Design*, Vol. 15, No. 2, March 1983, pp. 59–64.

[Appel and Stein 1972]
A. Appel and A. Stein, "A System for the Interactive Design of Polyhedra," *Proceedings of Online 72, International Conference on Online Interactive Computing*, Uxbridge, Middlesex, England, September 4-7, 1972, pp. 363–402.

[Baer *et al.* 1979]
A. Baer, C. Eastman, and M. Henrion, "Geometric Modeling: A Survey," *Computer Aided Design*, Vol. 11, No. 5, September 1979, pp. 253–272.

[Baumgart 1974]
B. G. Baumgart, "Geometric Modeling for Computer Vision," Ph.D Dissertation, Artificial Intelligence Laboratory Memo AIM-249, Stanford University, Stanford, California, October 1974.

[Borkin *et al.* 1978]
H. J. Borkin, J. F. McIntosh, and J. A. Turner, "Development of Three Dimensional Spatial Modeling Techniques for the Construction Planning of Nuclear Power Plants," *ACM Computer Graphics*, Vol. 12, No. 3 (SIGGRAPH '78 Conference Proceedings, Atlanta, Georgia, August 23-25, 1978) pp. 341–347.

[Boyse 1983]
J. Boyse, Private communication.

[Braid 1974]
I. C. Braid, *Designing with Volumes*, Cantab Press, Cambridge, England, 1974.

[Clowes 1971]
M. B. Clowes, "On Seeing Things," *Artificial Intelligence*, Vol. 2, 1971, pp. 79–116.

[Doctor and Torborg 1981]
L. J. Doctor and J. G. Torborg, "Display Techniques for Octree Encoded Objects," *IEEE Computer Graphics and Applications*, Vol. 1, No. 3, July 1981, pp. 29–38.

[Fitzgerald 1981]
W. J. Fitzgerald, "Using Axial Dimensions to Determine the Proportions of Line Drawings in Computer Graphics," *Computer Aided Design*, Vol. 13, No. 6, November 1981, pp. 377–382.

[Grossman 1976]
D. D. Grossman, "Procedural Representation of Three Dimensional Objects," *IBM Journal of Research and Development*, Vol. 20, No. 6, November 1976, pp. 582–589.

[Huffman 1971]
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.

[Idesawa 1973]
M. Idesawa, "A System to Generate a Solid Figure from a Three View," *Bulletin of the Japan Society of Mechanical Engineering*, Vol. 16, No. 92, February 1973, pp. 216–225.

[Idesawa *et al.* 1975]
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.

[Jones 1980]
J. C. Jones, *Design Methods: Seeds of Human Futures*, 2nd ed., John Wiley and Sons, Inc., New York, New York, 1980.

[Kanade 1978]
T. Kanade, "A Theory of Origami World," Report No. CMU-CS-78-144, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania, September 1978.

[Lafue 1976]
G. Lafue, "Recognition of Three-Dimensional Objects from Orthographic Views," *ACM Computer Graphics*, Vol. 10, No. 2 (SIGGRAPH '76 Conference Proceedings, Philadelphia, Pennsylvania, July 14–16, 1976) pp. 103–108.

[Light 1982]
R. A. Light, "Variational Geometry: Modification of Part Geometry by Changing Dimensional Values," *Proceedings of the Conference on CAD/CAM Technology in Mechanical Engineering*, Massachusetts Institute of Technology, Cambridge, Massachusetts, March 24-26, 1982, pp. 64–75.

[Markowsky and Wesley 1980]
G. Markowsky and M. A. Wesley, "Fleshing Out Wire Frames," *IBM Journal of Research and Development*, Vol. 24, No. 5, September 1980, pp. 582–597.

[Preiss 1980]
K. Preiss, "Constructing the 3-D Representation of a Plane-Faced Object from a Digitized Engineering Drawing," *Proceedings of the Fourth International Conference and Exhibition on Computers in Design Engineering*, Brighton, England, March 1980, pp. 257–265.

[Requicha 1980]
A. A. G. Requicha, "Representations for Rigid Solids: Theory, Methods, and Systems," *ACM Computing Surveys*, Vol. 12, No. 4, December 1980, pp. 437–464.

[Sakurai and Gossard 1983]
H. Sakurai and D. C. Gossard, "Solid Model Input through Orthographic Views," *ACM Computer Graphics*, Vol. 17, No. 3 (SIGGRAPH '83 Proceedings, Detroit, Michigan, July 25–29, 1983) pp. 243–247.

[Sugihara 1978]
K. Sugihara, "A Step Toward Man-Machine Communication by Means of Line Drawings of Polyhedra," *Bulletin of the Electrotechnical Laboratory*, Vol. 42, Nos. 11 & 12, 1978, pp. 848–871.

[Sugihara 1981]
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.

[Sutherland 1963]
I. E. Sutherland, "SKETCHPAD: A Man-Machine Graphical Communication System," *Proceedings SJCC*, Vol. 23, 1963, p. 329.

[Sutherland 1974]
I. E. Sutherland, "Three Dimensional Data Input by Tablet," *Proceedings of the IEEE*, Vol. 62, No. 4, April 1974, pp. 453–461.

[Sutherland *et al.* 1974]
Ivan E. Sutherland, Robert F. Sproull, and Robert A. Schumacker, "A Characterization of Ten Hidden-Surface Algorithms," *Computing Surveys*, Vol. 6, No. 1, March 1974, pp. 1–55.

[Waltz 1975]
D. Waltz, "Understanding Line Drawings of Scenes with Shadows," *The Psychology of Computer Vision*, P. H. Winston, ed., McGraw-Hill, New York, New York, 1975, pp. 19–91.

[Wesley 1980]
M. A. Wesley, "Construction and Use of Geometric Models," Chapter 2 in *Computer Aided Design*, J. Encarnação, ed., Lecture Notes in Computer Science No. 89, Springer-Verlag, New York, New York, 1980, pp. 79–136.

[Wesley and Markowsky 1981]
M. A. Wesley and G. Markowsky, "Fleshing Out Projections," *IBM Journal of Research and Development*, Vol. 25, No. 6, November 1981, pp. 934–954.

[Winston 1977]
P. H. Winston, *Artificial Intelligence*, Addison-Wesley Publishing Co., Inc., Reading, Massachusetts, 1977.

DISCUSSION

Wesley

I must say I feel nervous speaking to an audience that has so many people in it whose work I'm talking about. I see Chuck Eastman there and I see David Gossard is about to put up his hand, so I know I've got trouble coming.

Robert Johnson *(R.H. Johnson & Associates)*

As far as user iteraction is concerned, do any of the systems have the ability to make use of the coherence of the object as developed? In other words, if information is input in a number of steps, a different model would result at every

step. It might be possible to use different algorithms at each subsequent stage rather than make a total appraisal of all of the environment. It could be a way of constructing a solid object.

Wesley

Forcing the user to work in a step-by-step consistent mode is a big constraint on how the user works and he may not want to do that. I think you are thinking very much of building a CSG representation. But to answer your question, no, I do not think that any of these systems do that.

M. A. Wesley

John Hinds *(General Electric)*

In one of the examples, you successfully reconstructed the objects from 1250 edges. Was that from three projections or four projections?

Wesley

No, that was just a wire frame. So the input data contained the three-dimensional vertices and the linking edges.

Hinds

Does that take a long time?

Wesley

Well, the code was written very much in the mode of "let's get it written," and so it's mostly n^2 code with occasional ventures into something a little more

efficient if it seems necessary. That particular algorithm was run in a System IBM 370/168 and I think took rather less than a minute of CPU time. But that was a nonoptimized implementation. The potential customers who looked at the algorithm to convert a wire frame of that complexity to a solid model felt that it was a very acceptable price to pay.

David Gossard *(Massachusetts Institute of Technology)*

I was raising my hand to cheer you on. Your statement that most approaches are unable to handle wrong or incomplete data is absolutely right. I was curious to know of any work which has dealt with problems of either inconsistencies or incompleteness in various views of a geometric entity.

Wesley

I don't know of any such work. One can think of a lot of things one could do, but I haven't seen any explicit work to address that.

Michel Melkanoff *(University of California–Los Angeles)*

We are working on that area. We are trying to develop an algorithm for a simple, smart CAD system which can discover certain problems; in particular, one of the hardest problems is the inconsistency of multiple views.

Kalman Brauner *(Boeing Commercial Airplane Company)*

It is my understanding that you have done some work on being able to characterize conditions under which solutions are unique, *i.e.*, exactly one object is reconstructed given certain projections. Could you comment on that?

Wesley

One would very much like to be able to look at the input data and come up with a statement about the number of solutions. The only way we have of doing it is to find them all and count them. We haven't found any cheaper way of doing that. Every time you try to think about measures for what might be clues towards ambiguities, it really seems that you might as well do the whole job and see what comes out.

Brauner

So, for example convexity or a lack of holes is not adequate?

Wesley

Right.

Fumihiko Kimura *(University of Tokyo)*

I feel that your comment on a graphic approach to the user interface is very interesting. Have you done any work in this area?

Wesley

I think that you, Professor Kimura, are well qualified to make that comment because you in fact have worked extensively on the interactive design of objects. I didn't mention your work because I felt it came outside this little classification I was using, but I think your design with tablet input describing geometric components of an object is very interesting. We have not done any work on that.