Conversion of 2-Dimensional Drawings into 3-Dimensional Solid Model

A thesis presented

to

The Faculty of the College of Engineering and Technology

Ohio University

In

Partial Fulfillment

of the requirement for the degree

Master of Science

By

Amit Midha

June, 1991

OHIO UNIVERSITY
LIBRARY

ACKNOWLEDGEMENTS

The author thanks Dr. Jay S. Gunasekera for all the guidance and assistance during this research. The author also appreciates the assistance and advice of Mr. Bhavin Mehta, during the course of this research.

A special thanks to Universal Energy Systems Inc., and to the Wright Paterson Airforce Base for the financial and technical assistance provided for this research.

TABLES OF CONTENTS

ABSTRACT

This project attempts to develop an algorithm for automatically obtaining a 3-D solid model from 2-D orthographic projections. Starting from three views of an object, the algorithm determines a volumetric description in terms of solid material, empty space, and the topology of faces and edges. This algorithm has the capability to handle nonplanar 2-D sections of an object and obtaining a 3-D models from them.

The first step in the algorithm consists of inputting a set of data files having information about the coordinates and connectivities of different views. The vertices in each view are then back-projected to find all the vertices formed by the intersection of noncoplanar edges and some vertices formed by the intersection of the coplanar edges. The vertices found in this stage or further stages are called candidate vertices and may not be vertices or even points on the object.

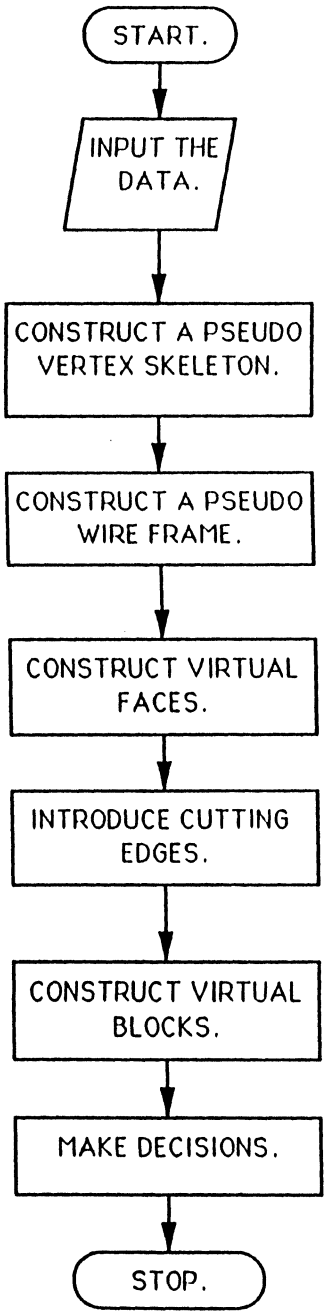
The structure formed after this stage is called the *pseudo skeleton* of the object.

In the next stage 'edges' are introduced based on the edges in the projections. Each vertex is taken and all the edges from it are drawn and checked in all the three views. Intersections are introduced as additional vertices and used

to partition the edges. The edge connectivity of all vertices is examined and the candidate edge and vertex lists are edited. This is called a *pseudo wire-frame* of the object.

All the graphs containing more than two noncollinear edges are processed. 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 called *virtual faces.* 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. A final decision process assigns solid or hole state to each virtual block, glues the solid blocks together, and finds all possible solid objects with the input orthographic views.

OUTLINE OF THE BASIC ALGORITHM



CHAPTER - 1

INTRODUCTION

1.1 Introduction

The application of computers to problems in mechanical design was first recognized over thirty years ago. Since that time much work has been done on the development of production systems both for the entry of a design into a computer data base and for the use of a mechanical design data base in design analysis and in manufacturing.

In the field of data base entry, computer drafting systems allow a designer to interact with a display or tablet to produce drawings of objects, generally in the classic manner of two-dimensional (2-D) projections of the edges of the three-dimensional (3-D) objects. Some systems also provide the capability of representing data in three-dimensions; for example depth coordinates may be added to the elements of a two-dimensional view, corresponding features in each of several views may be related, and isometric views may be constructed. These

computer drafting systems have been engineered to very high levels of performance and can greatly enhance the productivity of a designer. As well as producing drawings of the edges of objects, computer drafting systems exist which allow the description of the surfaces of objects as smooth curves or patches splined together at their boundaries; in general these surfaces are represented by means of a discrete mesh superimposed on the surface.

For decades, industries have documented large amounts of 3-D data in 2-D, non-computerized, graphic formats, such as engineering drawings and contour maps. The arrival of the reasonably priced, 3-D, computer-based graphics capability has revolutionized the capability to create, manipulate, and communicate three-dimensional information. A major problem is the labor-intensive steps presently required to transform existing 2-D information into 3-D form to allow this 3-D graphics capability to be further exploited for engineering applications.

The engineering applications of information in a 3-D format are numerous. Through the use of 3-D computer graphics, engineering information is more easily stored, accessed and modified. It is more clearly represented for, and is more conducive to use in engineering design(i.e., Computer Aided Design (CAD)), engineering analysis (Computer Aided Engineering (CAE)), and manufacturing (Computer Aided Manufacturing(CAM)).

Computer-based systems are also used in the analysis of designs and in the manufacture and assembly of objects. For example, finite element methods may be used for analysis of heat flow. The constraints between objects can be derived and mechanisms can be simulated; numerically controlled machine tool tapes can be generated to allow manufacture of a part, and robot actions to assemble parts can be generated. In general the full automation of these applications of the design data base requires 3-D volumetric information about an object, rather than just a description in terms of edges and surface mesh facets. At present, the volumetric form of the data base is considered to be rather difficult and expensive to acquire, and the analysis data are generated in a computer-assisted manner. For example, in the case of numerically controlled machine tools, the path of the cutter may be entered over a drawing at a graphics terminal.

One of the main requirements today in the industry is the need to model (in the computer) the geometry of a part in such a way that sufficient information is available for reuse later. Models are useful because often one can study certain characteristics of an object more easily based on the model rather than its physical counterpart.

Engineering drawings, widely used in design and engineering can also be viewed as models. They can be used to extract information for a number of tasks, including the creation of a mathematical and a physical model. But unlike

mathematical and physical models a designer can never get the feel of the object from engineering drawings. They sometimes tend to be fuzzy and unclear, whereas models can serve as important tools for designing or analyzing an object.

This project attempts to develop an algorithm for automatically obtaining a 3-D solid model from 2-D orthographic projections. Starting from three views of an object the algorithm determines a volumetric description in terms of solid material, empty space, and the topology of faces and edges. This algorithm has the capability to handle nonplanar 2-D sections of an object and obtaining a 3-D model from them.

1.2 Literature Survey

The idea of devising an algorithm for converting 2-dimensional (2-D) views into a 3-dimensional (3-D) object started about 15 years ago. A. G. Requicha and R. B. Tilove [1] published a technical memo describing the "Mathematical Foundations of Constructive solid Geometry: General Topology of closed Regular Sets." which forms the basis of this research. The book "Topology" by J.F. Hocking and G.S. Young [2] describe all the basic concepts about set topology used in this research. I.C. Braid [3] came up with a technique which allows a class of solid objects to be synthesized and stored using a computer. Synthesis begins with primitive solids like a cube, wedge, or cylinder which is used as a decision making process in this research.

Systematic work on 3-D conversion was started at International Business Machines by M. A. Wesley and G. Markawsky [4]. They came up with a solution to find the object if its wire-frame is given to you. Later M. Idesawa [6] did the same work in wire-frames, and automated surface generation and Solid design from the wire frame.

Stenstran and Connally [7] describe a process by which wire frames of polyhedra may be generated from multiple range views of a scene. Line segments were entracted from each view and transformed into a world coordinate system to produce the wire frame.

Rémi Lequette [8] presented an algorithm which finds all solids that fit with a set of two or three bi-dimensional drafting views. The views are orthoganal projections of the wire frame and hidden parts are not removed.

CHAPTER - 2

BASIC CONCEPTS

2.1 BASIC CONCEPTS

The purpose of defining these basic fudamentals here is to help understand the concepts used in the chapters ahead. In the forthcoming chapters all the applied definations are refered to in brackets.The basic concepts defined in this chapter are based on some fundamental topological ideas which are described in detail in [2].

Throughout the standard topology in $\Re^3$ and the induced topology on the subsets of $\Re^3$ are assumed. Vertices refer to points in $\Re^3$ and edges refer to line segments defined by the two points in $\Re^3$. The approach used in this chapter 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 $\Re^3$ whose boundary (denoted by ∂f) is the union of a finite number of line segments. $P_{\underline{f}}$ is used to denote the unique plane which contains *f*.

Definition 2

An *object, O*, is the closure of a nonempty, bounded, open subset of $\Re^3$ whose boundary (denoted by ∂O) is the union of a finite number of faces.

From the definitions above it is easy to see that the "cube"

$\{x,y,z \in \Re^3 \mid 0 \leq x \leq 1, 0 \leq y \leq 1, 0 \leq z \leq 1\}$

is an object and that

$\{(1,y,z) \in \Re^3 \mid 0 \leq y \leq 1, 0 \leq z \leq 1\}$

is one of its "square" faces. Starting of with open sets means that faces and objects have nontrivial interiors. It is not assumed that an object is the closure of a connected set. This allows objects that consists of disjoint "solids" or even objects which intersect only in edges, etc.

Definition 3

(a) Let *f* be a face. The *vertices* of *f* , V(*f*), are defined to be set of all points for which two noncollinear 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 points of e belongs to V(f).

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

$\{p\} = f_1 \cap f_2 \cap f_3 = P_{f_1} \cap P_{f_2} \cap P_{f_3}$ are both exactly the single point P.

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

1. The endpoints of e belong to V(O);

2. No interior point of e belongs to V(O);

3. For every point p of e, two noncoplanar faces can be found, $f_1, f_2 \subseteq \partial O$ such that $p \in f_1 \cap f_2$.

(e) Let O be an object. The *wire frame* of O, WF(O) is defined to be the ordered pair [V(O), E(O)].

It can be shown that V(f), E(f), V(O), and E(O) are all finite. V(f) and E(f) yield the intuitive representation of f. V(O) and E(O) do not quite represent O,

since the faces of O, which have not been defined so far, are needed.

Definition 4

A *1-cycle* is a collection of coplanar line segments $\{e_1,.....,e_k\}$ in $\Re^3$ having the following properties:

1. The intersection of two distinct elements e_i and e_j is either the empty set or a point which is an endpoint of both line segments;
2. Every point of $\Re^3$ is the end point of a *nonnegative even* number (in most cases 0) of the e_i.

In order to be able to give a complete definition of a face, it is necessary only to describe what is meant by the inside and outside of a 1-cycle. A 1-cycle has an inside (outside) if there is a bounded (unbounded), connected,open set whose boundary is the given 1-cycle. A 1-cycle may lack an inside or an outside, or may have them both. Whenever a point is said to be inside (outside) a 1-cycle C it is meant that C has an inside (outside) and that the point in question belongs to some bounded (unbounded) connected, open set whose boundary is C. Actually, if a 1-cycle, C, has an inside (outside), there is a component of the complement of C whose boundary is C and which contains all other bounded (unbounded), connected, open sets whose boundary is C.

Theorem 5

Let f be a face. Then 1-cycles, $C_0, C_1, \ldots, C_k$ ($k \geq 0$) contained in ∂f can be found such that

1. $\partial f = C_0 \cup C_1 \cup \ldots \cup C_k$;
2. Face f consists of all points inside C_0 and outside C_i ($i \geq 1$), and the points of $\bigcup_{i=0}^{k} C_i$;
3. The 1-cycle are all disjoint.

A typical face is pictured in figure 2.1. Note that the boundary can intersect itself at various points such as v_1, v_{16}, and v_{18}. The face in figure 2.1 can be defined in terms of the 1-cycles: C_0 (traced out by the following the sequence of vertices $v_1\ v_2 \ldots v_9\ v_{10}\ v_1\ v_{11}\ v_{12}\ v_{13}\ v_1$), C_1 (traced out by $v_{14}\ v_{15}\ v_{16}\ v_{17}\ v_{18}\ v_{19}\ v_{20}\ v_{18}\ v_{16}\ v_{14}$), and C_2 (traced out by $v_{21}\ v_{22} \ldots v_{24}\ v_{21}$). Thus the face in the figure 2.1 consists of all points in the inside of C_0 and in the outside of C_1 and C_2, plus the points actually belonging to C_0, C_1, and C_2. Note that the points v_2, v_1, and v_{11} are collinear, but by the definition of $E(O)$, v_2v_1 and v_1v_{11} actually belong to $E(O)$, but v_2v_{11} does not. Not every 1-cycle has an inside (C_1, for example, fails the connected set requirement).

Definition 6

Let O be an object. The *faces* of O, $F(O)$, are defined to be closures of the connected components of $\partial O - \cup E(O)$.

A number of results hold true for the faces of the object. Some of the important relationships are summarized in the following theorem.

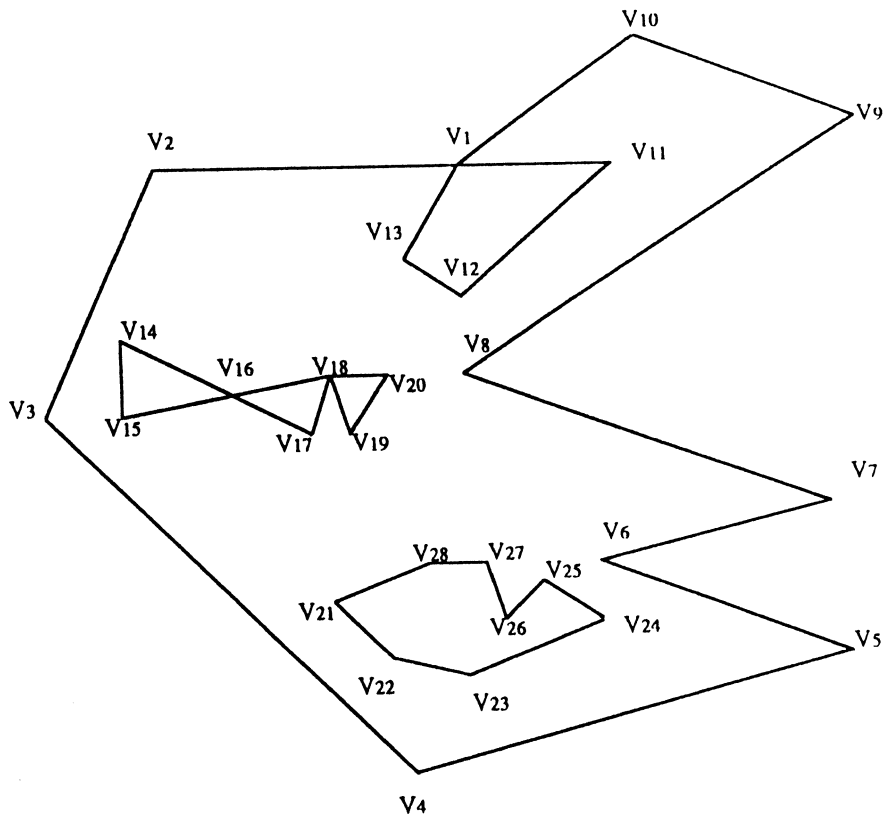


Figure 2.1 A typical face

Theorem 7

Let O be an object and $F(O) = \{f_1, \ldots\ldots, f_m\}$. Then

1. $\partial O = \bigcup_{i=1}^{m} f_i$;

2. $\bigcup_{i=1}^{m} E(f_i) \subseteq E(O) \cup \tau$, where τ is the set consisting of all line segments which are unions of elements of $E(O)$;

3. $\bigcup_{i=1}^{m} V(f_i) \subseteq V(O)$

4. Any face $f \subseteq \partial O$ for which $E(f) \subseteq E(O) \cup \tau$, where τ as is in (2) above, is the union of elements of $F(O)$;

5. The intersection of the two distinct elements of $F(O)$ is a union of elements of $E(O)$ and subsets of $V(O)$.

Vertices of an object need not be vertices of any face; e.g. in figure 2.2 point A is in $V(O)$ but is not a vertex of any face of O. Thus the corresponding edges must be broken up into smaller line segments when considered as edges of O, but this division does not occur if any face is considered separately.

Definition 8

Let O be an object, $\mathbf{P} \subset \Re^3$ a plane, and $\pi_p: \Re^3 \rightarrow \mathbf{P}$ the perpendicular projection. By the **P**-*projection* of O, denoted

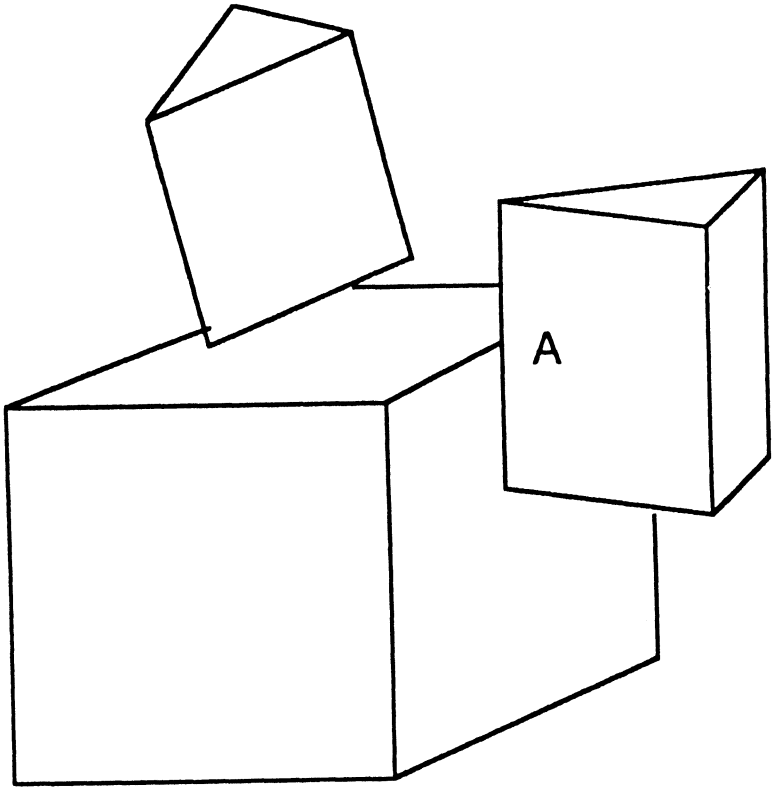


Figure 2.2 An object exhibiting various kinds of intersections

by $O \mid \mathbf{P}$, is meant the ordered pair, $(V(O \mid \mathbf{P}), E(O \mid \mathbf{P}))$, of **P**-*vertices* and **P**-*edges* of O defined by the following process. Let E^* be the set of images under π_p of all edges of O which are not perpendicular to **P**. Then the **P**-vertices of O are those points of **P** which lie on at least two noncollinear line segments in E^*.

The **P**-edges of O are those line segments of **P** which have elements of $V(O|\mathbf{P})$ as endpoints, have no points of $V(O|\mathbf{P})$ as interior points, and are subsets of unions of elements of E^*.

XY, **YZ**, and **ZX** are used to denote the planes $\mathbf{Z}=0$, $\mathbf{X}=0$, and $\mathbf{Y}=0$, respectively.

Figure 2.3 shows some of the things that can happen as a result of projection. The vertex A disappears in the first 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 projection of AB, AC, AD, and AE from distinct line segments.

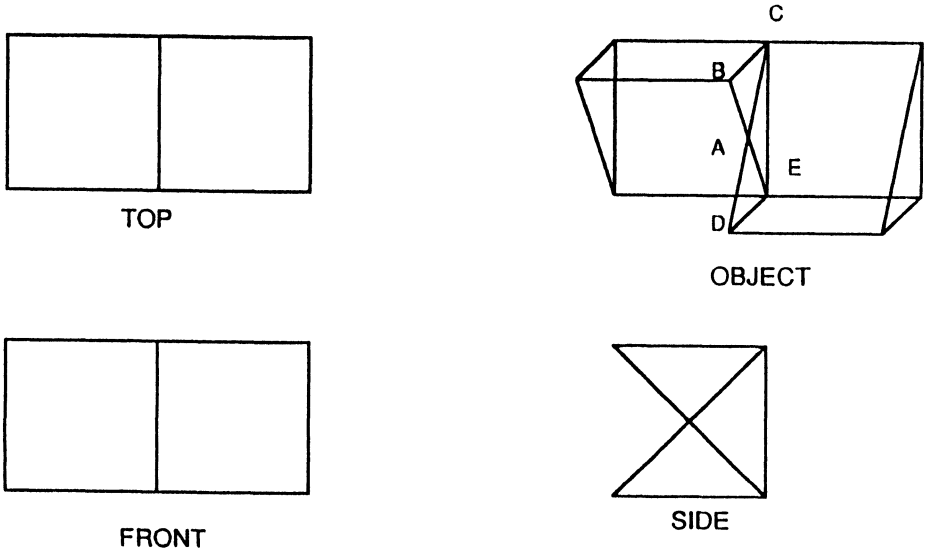


Figure 2.3 Example of projections

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 noncollinear 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 figure 2.3, vertex A is Class II; all other vertices are Class I. In general, one cannot expect to recover Class II vertices simply by erecting perpendicular and computing their intersections.

Definition 9

The *skeleton*, $S(O)$, of an object O is the ordered pair $(SV(O), SE(O))$ of *skeletal edges* where $SV(O)$ is the set of the Class I vertices of O and $SE(O)$ is a set of line segments joining the elements of $SV(O)$.

For $v_1, v_2 \in SV(O)$, there exists an edge or collinear sequence of edges of O joining v_1 and v_2 and not containing any other Class I vertex.

Theorem 10

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

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

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

The above theorem gives some insight into the working of the 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 portion of the edges of the skeleton and wire frame of the original object.

Theorem 11

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

From Theorem 10 it follows that v is the intersection of two noncollinear line segments, e_1 and e_2, which are unions of line segments of *O*, are contained in the boundary of *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 *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 *O* were to be on one side of **P**, there would be contradiction, since at least four noncoplanar faces would go through v, but all the edges containing v would be coplanar.

Definition 12

A *primitive object* is an object whose interior is connected.

The key point of this definition is to prevent problems caused by the peculiar types of intersections illustrated in Fig 2.2 In the figure, there are three primitive objects: a cube and two prisms. In general, an object can be decomposed into primitive objects which do not have any 2-dimensional intersections between any two of them.

Definition 13

A 2-cycle is a collection of faces $\{f_1,.....f_m\}$ in $\Re^3$ having the following properties:

1. The intersection of two distinct elements f_i and f_j is the union of the elements of $E(f_i) \cap E(f_j)$ and a finite number of points;

2. Every element of $\bigcup_{i=1}^{m} E(f_i)$ belongs to an even number of distinct faces.

With all of these concepts a description of primitive objects can be given in terms of 2-cycles. From this one can extrapolate to objects in general. This description is similar to the description of the faces.

Theorem 14

Let O be a primitive object. Then 2-cycles $C_0, C_1,....,C_k$ ($k \geq 0$) contained in ∂O can be found such that

1. $\partial O = C_0 \cup C_1 \cup ....\cup C_k$;

2. O consists of all points inside C_0 and outside C_i

$$(i \geq 1) \text{ and the points in } \bigcup_{i=0}^{k} C_i;$$

3. The 2-cycles are all disjoint.

CHAPTER - 3

CONSTRUCTION OF PSEUDO WIRE FRAME

The basic algorithm constructs all polyhedral solid objects whose wire frames have a given set of projections (or views). The input to the basic algorithm was a set of 2-D views of the object.

The vertices of each view were back-projected to find all Class I vertex (*i.e.*, vertices formed by the intersection of noncoplanar edges) and some Class II vertex (*i.e.*, vertices formed by the intersection of coplanar edges); at this point it was 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 stages ahead were called *candidate vertices.*

The vertices constructed in the previous stage formed a skeleton for the pseudo wire frame in the same sense that WF(*O*) derives from SV(*O*). Edges were introduced based on the edges in the projections. These edges were checked for mutual internal intersections. Intersections were introduced as additional vertices and used to partition the edges. The remaining of the Class II vertices were constructed in this manner. The vertices constructed here and in the

previous stage were the set of candidate vertices (denoted CV(*O*)), and the final set of edges constructed in this stage was the set of candidate edges (denoted CE(*O*)). Together the candidate edges and vertices form the *pseudo wire frame.*

To make the description of the algorithm more comprehensible, the example of Two Wedge problem (figure 3.1) is used to illustrate the various stages, *i.e.,* the problem is to recover the object from its three views.

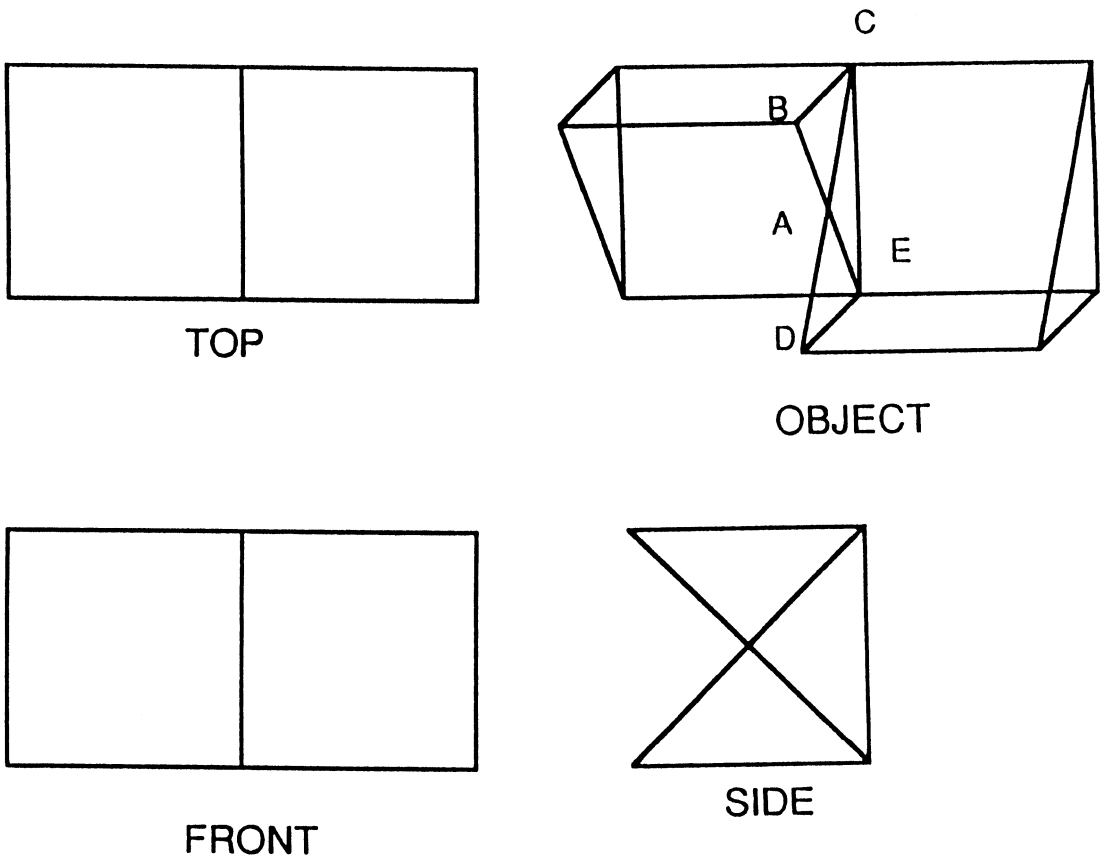


Figure 3.1: The Two Wedge problem.

3.1 Check input data

The input data to the basic algorithm was assumed to be a set of two dimensional (2-D) views of the whole wire frame of a polyhedral object. The views may be at arbitrary projection direction, but must meet a minimum requirement of at least two distinct projection. Each view is an ordered pair of vertices and edges (Definition 8) expressed relative to a local 2-D coordinate frame and accompanied by some extra information like data files containing the number of vertices and edges and naming the vertices and edges and giving the connectivities of the edges. The example of data files containing the front view coordinates and connectivities are shown in the following tables. The same is shown in the figure 3.2 and figure 3.3 respectively.

Similarly data files containing information about other views were also inputted to the algorithm (Appendix A).

Vertices	X-Coordinate	Y-Coordinate
1.	0.0	0.0
2.	1.0	0.0
3.	2.0	0.0
4.	2.0	1.0
5.	1.0	1.0
6.	0.0	1.0

Table:1 Coordinates of the Vertices of Front View

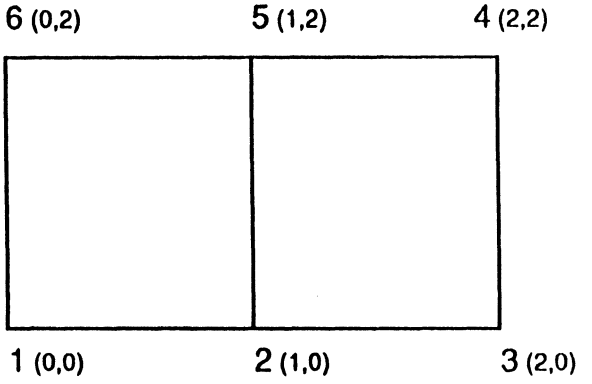


Figure 3.2: Coordinates of the Front View

Edges	Vertex 1	Vertex 2
1.	1	2
2.	2	3
3.	3	4
4.	4	5
5.	5	6
6.	6	1
7.	5	2

Table 2: Connectivities of the Edges of the Front View.

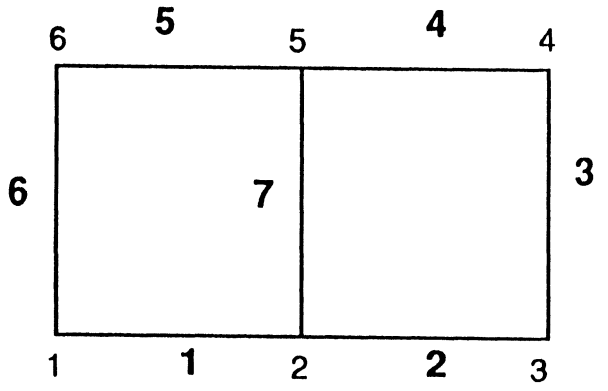


Figure 3.3: Connectivities of the Front View.

3.2 Construct pseudo vertex skeleton

In this stage, perpendiculars were erected at the vertices of each view as shown in the figure 3.4. Then, only those vertices lying on at least two noncollinear perpendiculars and which were consistent with all other projections, *i.e.*, their images were either vertices or interior points of edges, were selected. By doing so, all three coordinates of all the points in the pseudo skeleton were obtained. From Definition 8 it was noted that, all Class I vertices and possibly some Class II vertices were recovered here. In order that the projection be consistent, it was necessary that every **P**-vertex had at least one element of CV(*O*) (candidate vertices) in its inverse image. This check was performed as part of this stage. In addition if some **P**-vertex had a unique element of CV(*O*) in its inverse image, then that element of CV(*O*) must actually be an element of V(*O*). Such a vertex was assigned "type certain", and all other vertices were assigned "type uncertain".

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

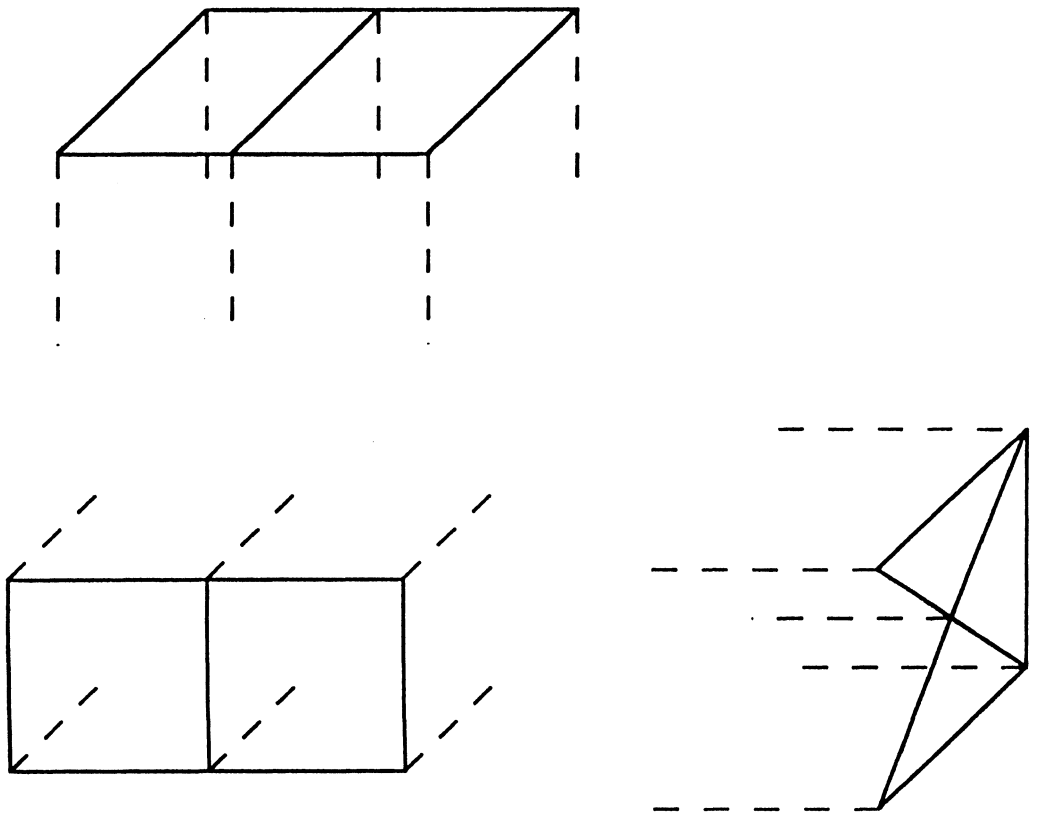


Figure 3.4: Perpendiculars erected at all vertices of each view

The pseudo vertex skeleton of the Two Wedge 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 [Figure 3.5]. This stage is explained as a flow chart in the figure 3.6.

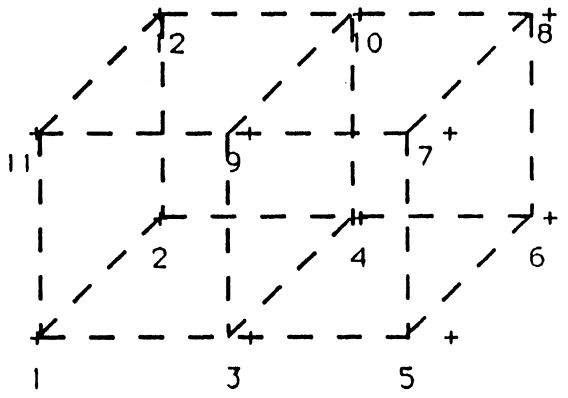


Figure 3.5: The vertex pseudo skeleton of the Two Wedge problem

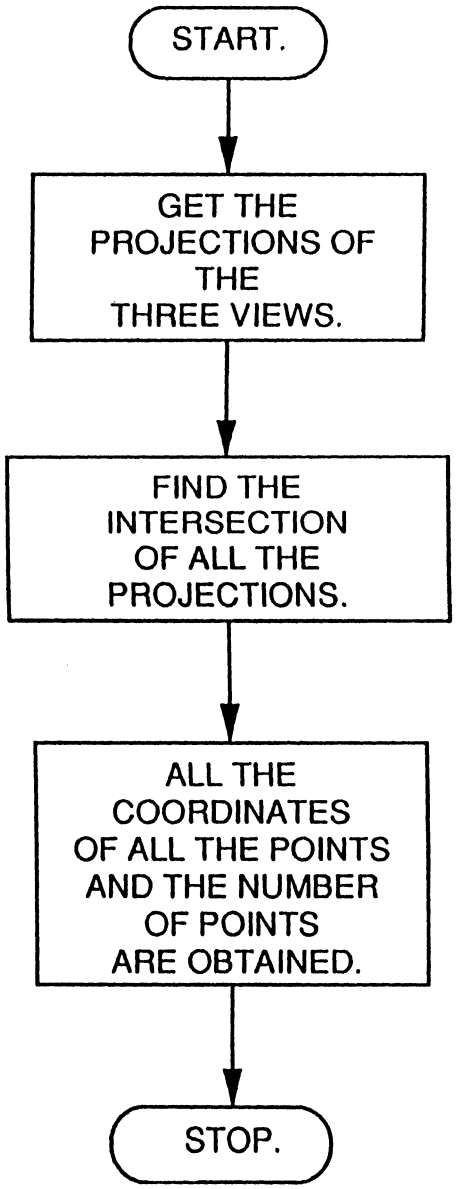


Figure 3.6: Flow Chart to get the skeleton of the object.

3.3 Construct pseudo wire frame

In this stage edges were constructed as a prelude to constructing the pseudo wire frame. To do this, two vertices were simply joined in the pseudo vertex skeleton by an edge if and only if, in every projection the images of these two vertices coincide or were joined by an edge or collinear set of edges and no other vertex of the pseudo vertex skeleton would be an interior point of the edge. This is as shown in the figure 3.7

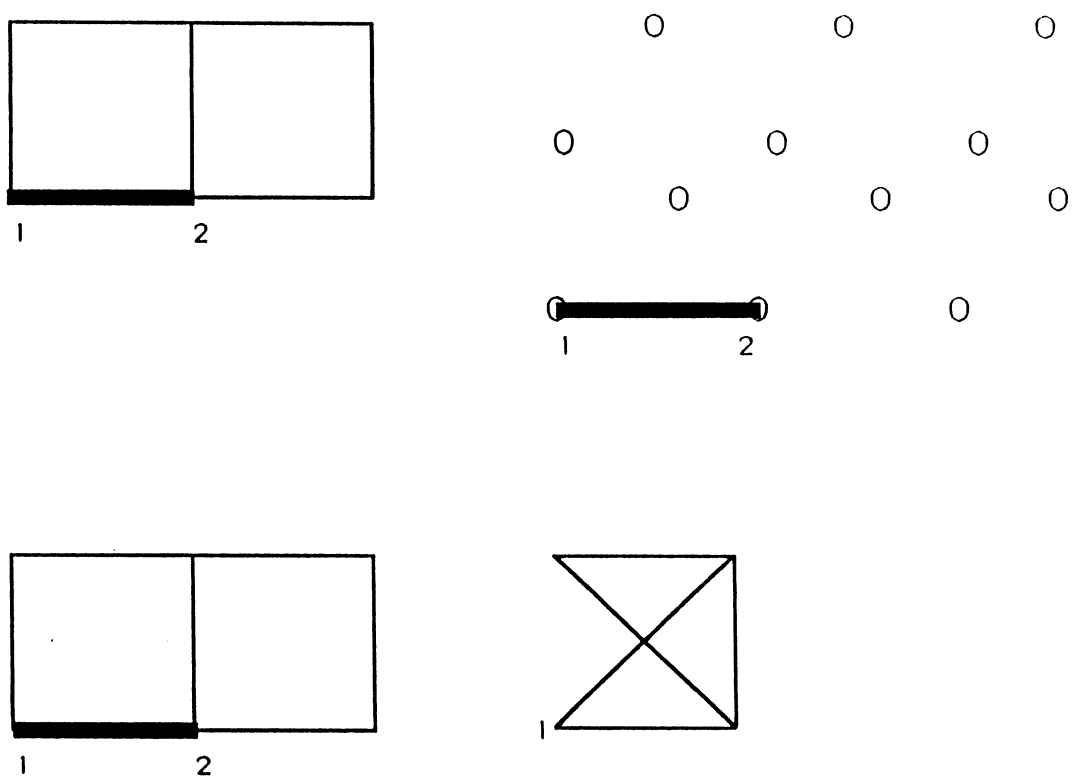


Figure 3.7 Construction of Pseudo Wire Frame

To obtain the entire wire frame, each vertex was connected to every other vertex. If the image of this edge formed by joining the two vertices was at least a line in two views and a point in the third view, it was taken as a valid edge. In some cases the image of the edge was a line in all the three views, then also it was taken as a valid edge. This is shown in the figure 3.8 as a flow chart.

These pseudo skeleton edges may intersect in mutually interior points. To obtain the pseudo wire frame from this skeleton, the techniques of Theorem 10 were duplicated, *i.e.*, to introduce edges in the obvious way so that all the edges had vertices as endpoints, that two edges intersected only in a vertex, and that no vertex would be an interior point of an edge.

So in addition to joining the vertices to form edges, it was also checked that if any of the two edges intersected with each other. If they did, then there intersection was considered a new vertex and was added to the list of vertices already existing.

It was verified that every **P**-edge (Definition- 8) had some element of CE(*O*) in its inverse image. In particular, if some **P**-edge had a unique inverse image, then that element of CE(*O*) must be real *i.e.*, it must actually be an element

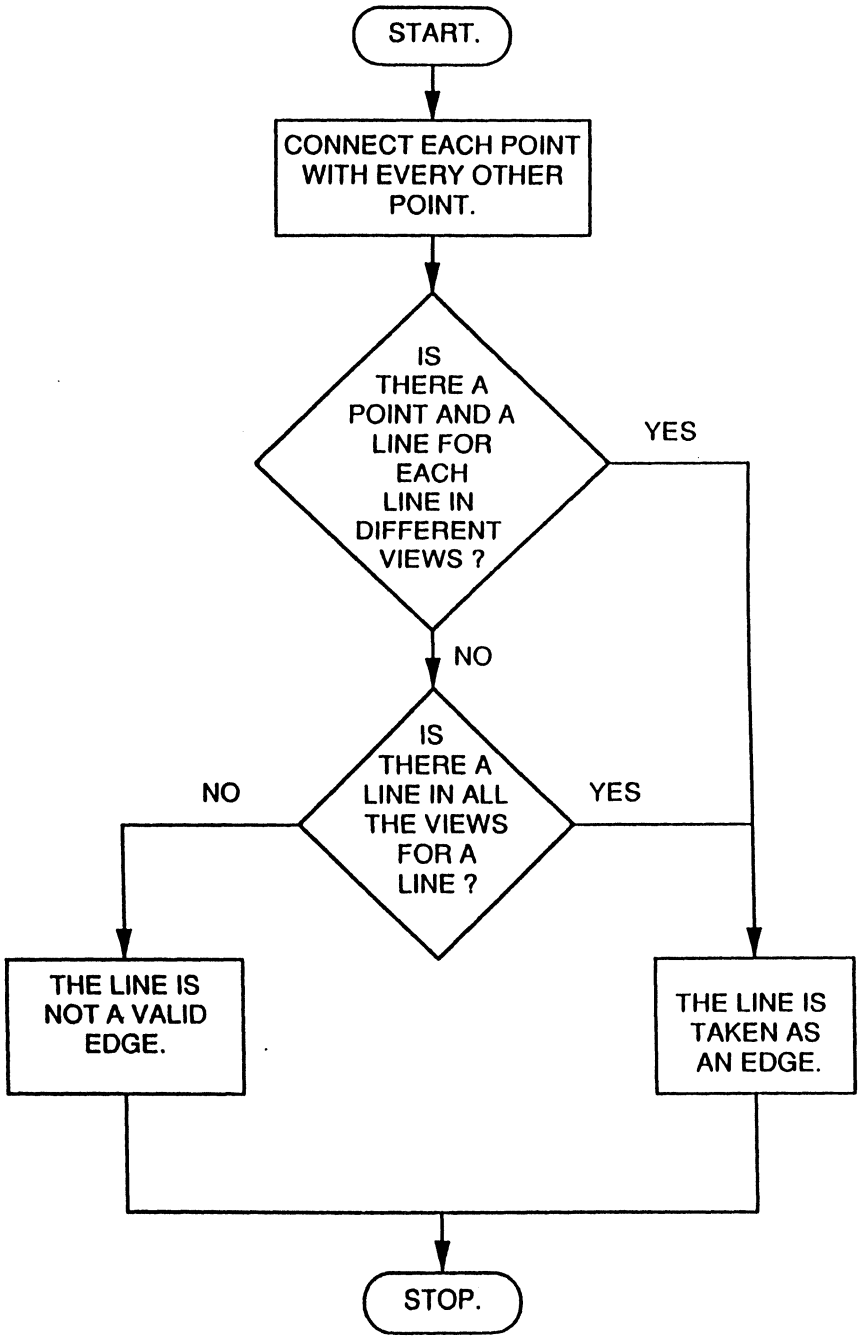
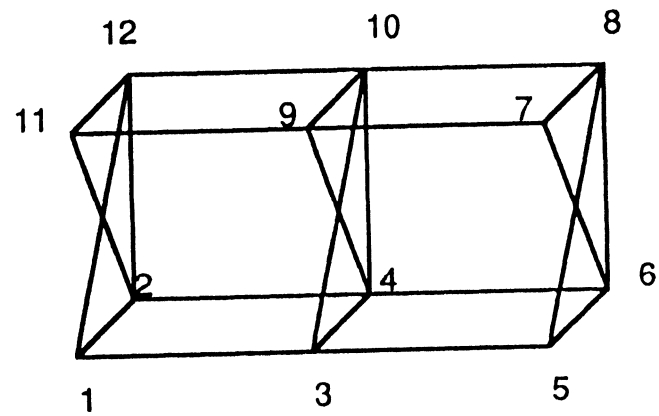


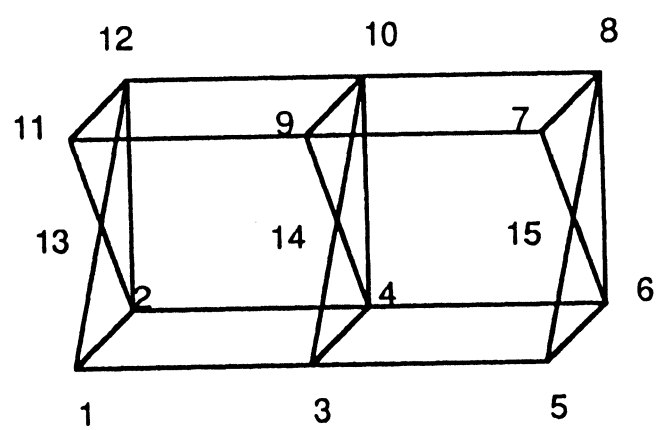
Figure 3.8: Flow Chart to find the connectivity of all the points

of E(*O*) and, like the rule for vertices, was classified as "certain".

New edges were introduced taking into consideration the fact that the end point of each edge was a vertex. Then the connectivities of the edges were updated. The new vertices and edges just found are as shown in the figure 3.9



(a)



(b)

Figure 3.9 (a) Pseudo Wire Frame without the new vertices

(b) Pseudo Wire Frame with the new vertices

So at the end of this stage a pseudo wire frame (CV(*O*), CE(*O*)) which looks like a wire frame was obtained. In many cases (CV(*O*), CE(*O*)) this was exactly the wire frame of *O*, but in some cases this pseudo wire frame contains uncertain edges and vertices, any of which may or may not exist in a solution.

CHAPTER - 4

CONSTRUCTION OF VIRTUALFACES

This chapter describes the method developed for obtaining virtual faces from the planes. Beginning with the pseudo wire frame generated in the last step, all virtual faces were found. All uncertain edges were checked for containment in at least two noncoplanar virtual faces. Any edge not meeting this criterion was deleted and the virtual faces updated. Any impossible virtual faces (*e.g., a certain edge piercing the interior of a virtual face*) were deleted. The consequences of deletions were propagated until a stable condition was reached.

In this chapter illegal intersections between two virtual faces, such that both faces cannot exist in an object were 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 would 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 which cannot co-exist in an object, these data structures were used in the final stages of the algorithm.

4.1 Finding the planar graphs

In this stage, all planes which contain at least two intersecting edges were found, and for each plane a graph was constructed of the edges and vertices in that plane. For each vertex in pseudo wire frame (CV(*O*), CE(*O*)), a list was formed of the edges for which the vertex was an end point. For each noncollinear pair of edges in the list, the plane containing the edge pair was computed and a list was formed of distinct planes at the vertex. Then each globally distinct plane was matched up at vertices to form graphs of the edges and vertices contained in the plane.

To do this, two edges passing through a vertex were taken. The three vertices from these two edges were obtained as both the edges had one common end point. With these three vertices an equation of a plane was obtained and all the other vertices lying on this plane were obtained. Some of the planes obtained in the Two Wedge problem are as shown in the figure 4.1

Thus, the output of this stage was a list of plane equations and, for each plane, graphs of vertices in the plane and the number of vertices in the plane.

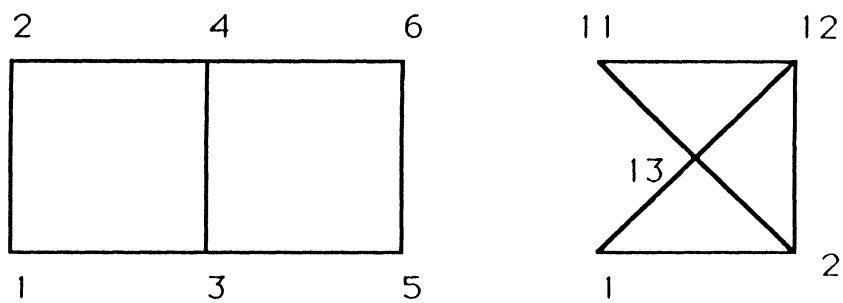


Figure 4.1: Some of the different Planes obtained in the Two Wedge problem

4.2 Calculation of 1-cycles and virtual faces

In this stage each planar edge and vertex graph was processed to find all subgraphs that could represent faces in accordance with Definitions 1 and 3. These subgraphs are candidates for faces of the object and are called virtual faces.

Virtual faces could be located by finding 1-cycles and determining the various nesting relationships among these 1-cycles. To make things easier, a plane P and a graph formed from the vertices of *O* which lie in P was assumed. In this plane one vertex was taken and checked for its connectivity. The edges which had this vertex as an end point and lie in this plane were sorted and then their other end points were obtained to get the vertices. In other words, the vertices connected to the first vertex were obtained. In the figure 4.2, vertex 1 was taken and connected to vertex 2 and 3. Then each subsequent vertex was taken and its connectivity was found out. This procedure was followed till one of the connecting vertices was either the very first vertex or any one of the vertices in between. In this case if it was the first vertex, it was taken as a virtual face and if it was any other vertex which had come before, then it was rejected as it was the wrong connection. All this is shown in the figure 4.2 and in figure 4.3 as a flow chart.

The 1-cycles obtained for the figure 4.2 for vertex 1 are

$C_1 = v_1\ v_2\ v_4\ v_3$

$C_2 = v_1\ v_2\ v_4\ v_6\ v_5\ v_3$

$C_3 = v_1\ v_3\ v_4\ v_2$

$C_4 = v_1\ v_3\ v_5\ v_6\ v_4\ v_2$

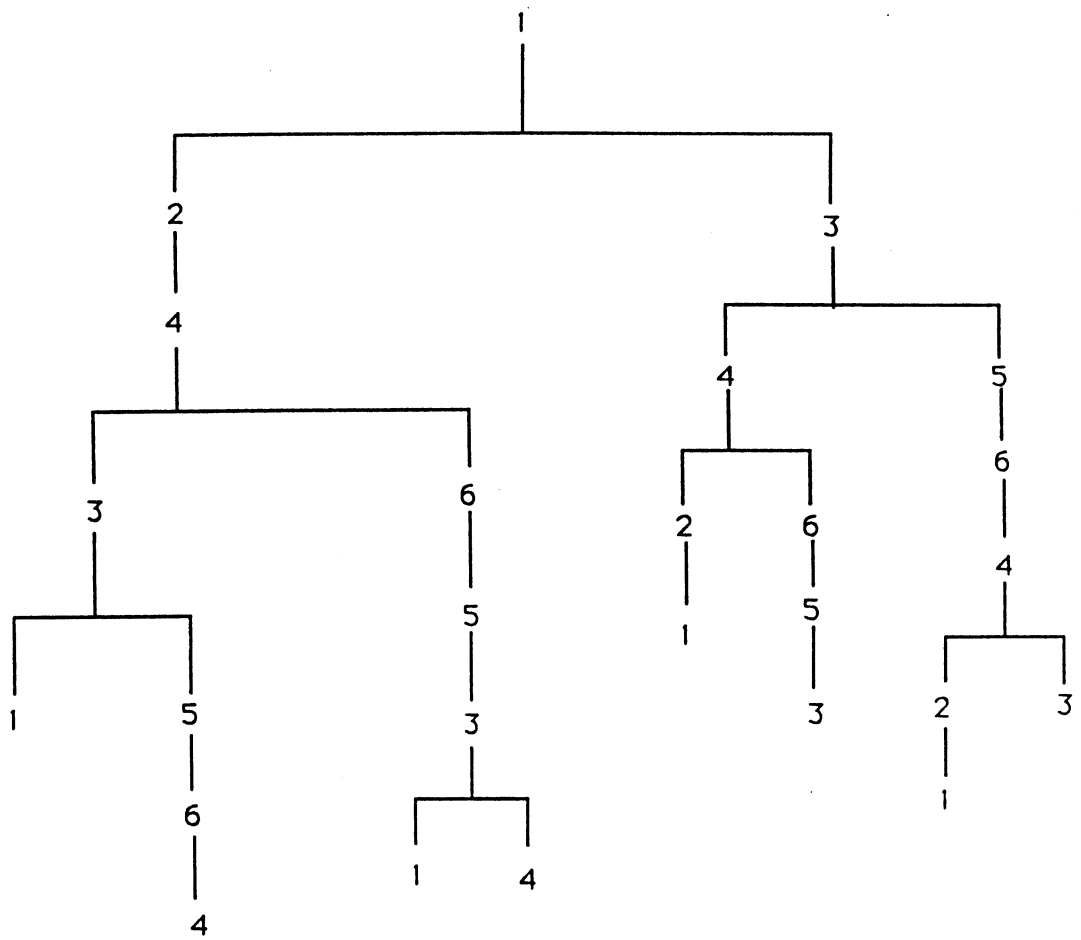
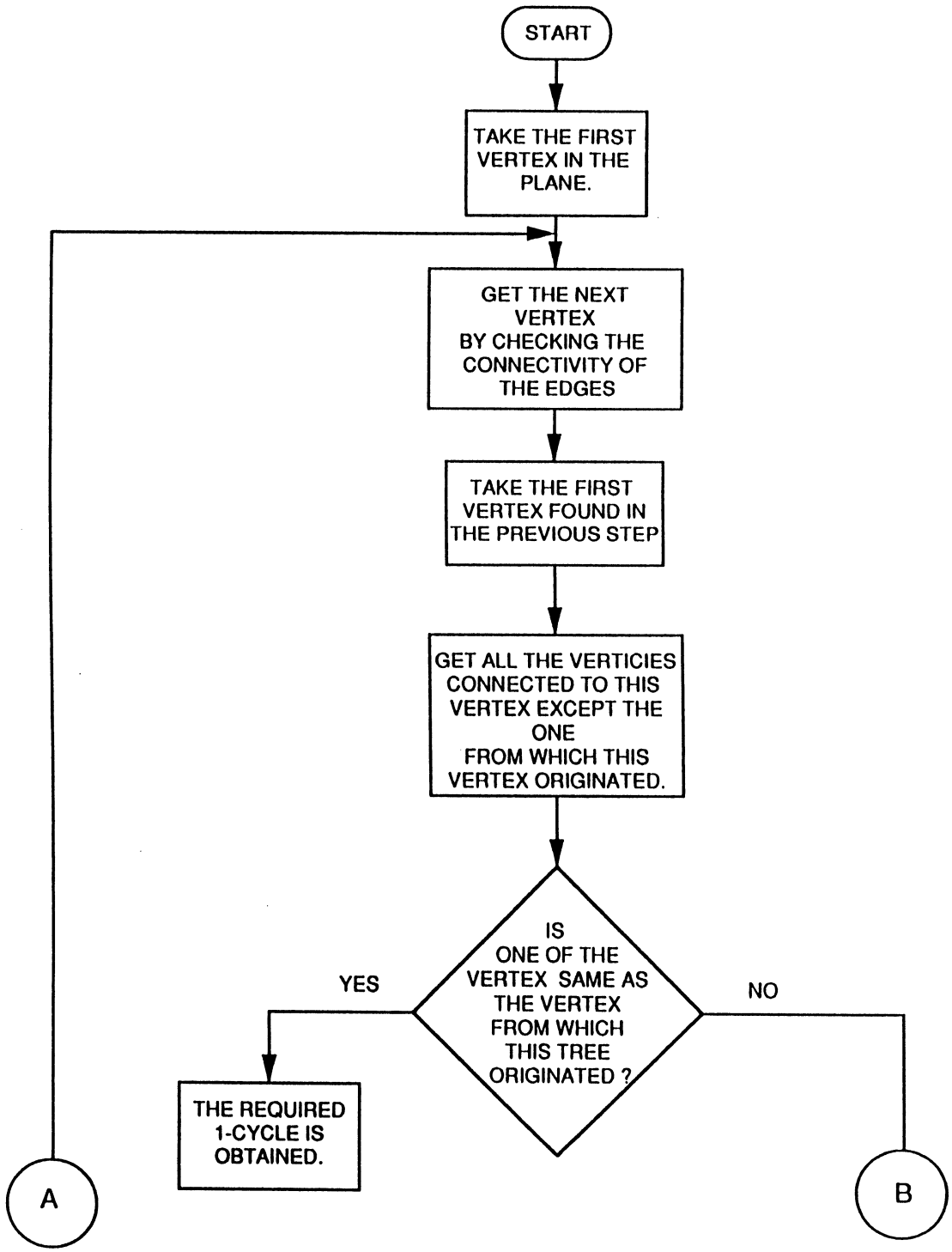


Figure 4.2: The elimination process for finding Faces.


START
TAKE THE FIRST VERTEX IN THE PLANE.
GET THE NEXT VERTEX BY CHECKING THE CONNECTIVITY OF THE EDGES
TAKE THE FIRST VERTEX FOUND IN THE PREVIOUS STEP
GET ALL THE VERTICIES CONNECTED TO THIS VERTEX EXCEPT THE ONE FROM WHICH THIS VERTEX ORIGINATED.
IS ONE OF THE VERTEX SAME AS THE VERTEX FROM WHICH THIS TREE ORIGINATED ?
YES
NO
THE REQUIRED 1-CYCLE IS OBTAINED.
A
B

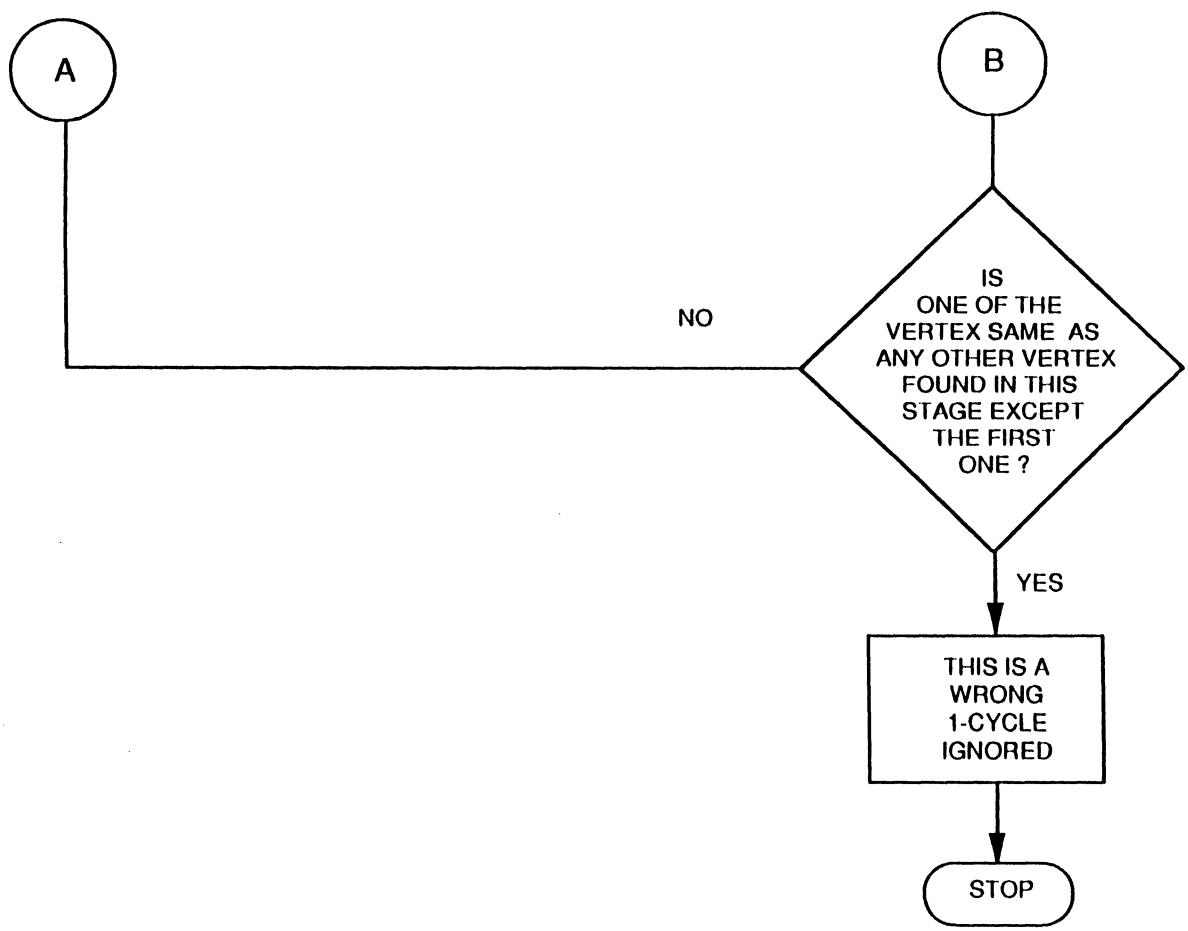


Figure 4.3 :Flow Chart to find all the 1-Cycles

Out of these four 1-cycles, cycle 1 and cycle 3, are the same and cycle 2 and cycle 4 are same.

All the 1-cycles were obtained like this, and then the cycle which had the minimum number of vertices was taken and stored in the list. In the present case, the virtual faces

were cycle 1 and cycle 3 as they had the smallest number of vertices among all the 1-cycles obtained for the vertex 1 of this plane. It was also noted that since cycle 1 and cycle 3 were same so only one virtual face was obtained. Similarly the other vertices were considered, and all the 1-cycles were obtained for those vertices. The 1-cycle which amounted to minimum area was chosen and was called the virtual face, and there might be more than one virtual face in the plane. After obtaining the vertices of the virtual faces, the edges that made the face were also obtained. In order to do that, any vertex of the virtual face was considered and checked for the edges passing through it, and then the other end points of edges were obtained and checked for the vertices of the virtual face in question. The number of faces from a vertex was also found out in this stage.

4.3 Checking for Illegal intersection between virtual faces

The description of objects in this chapter about Basic concepts, is based on 2-cycles (Definition 13), which have the property that the faces belonging to them intersect only at the boundary points of the faces. Two virtual faces intersect illegally when there exists a point in the intersection that is internal to both. In this case it is not possible for both to be real faces of the object. Illegal intersections can occur in either of two ways:

I. An interior point of an edge of one contains an interior point of the other;

II. The above type of intersection does not occur, yet a vertex of one is in the plane of the other, and there exists a point that is interior to both faces.

These illegal intersections, which are known as type I and type II intersections, respectively, are detected in this stage, and appropriate action taken as described in the following paragraphs.

A type I intersection occurred when any inside point of any virtual face was an inside point of any element of E(*O*). If such a condition was found, the virtual face was dropped from the list of virtual faces because it was impossible for it ever to be a face. To see this, it was noted that the edge of *O* belonged to actual

faces. If a virtual face intersects edges as described above, it would have to intersect the corresponding faces. Such an intersection would produce at least one edge emanating from an inside point of a face, which would be impossible.

The method to solve a problem of this type was based on the observation that a type II intersection consists of a finite number of line segments, the endpoints of which are elements of V(*O*). To see this, let f_1 and f_2 be faces that had type II intersection. Let $l = P_{\underline{f}_1} \cap P_{\underline{f}_2}$. Let $p \in f_1 \cap f_2$ be an interior point of both f_1 and f_2. Let p_1 and p_2 belong to $\partial f_1 \cap \partial f_2$. Since no boundary point of f_1 was an inside point of f_2 and *vice versa*, p_1, p_2 $\partial f_1 \cap \partial f_2$. If the edges of f_1 and f_2 which contained p_1 (p_2) were collinear, then f_1 and f_2 must be coplanar and must overlap in nontrivial ways. This was impossible in view of the tests performed in the previous stage. Thus p_1 and p_2 belonged to two noncollinear edges which could only intersect in an element of V(*O*). To help visualize this, the pseudo wire frame of the Two Wedge problem is referred as shown in fig 4.4 Here f_1 is given by v_1 v_3 v_{14} v_{10} v_{12} v_{13} and f_2 by v_2 v_4 v_{14} v_9 v_{11} v_{13}, p_1 is v_{13}, and p_2 is v_{14}. This gives a quick test for type II intersections: visit each vertex in turn and see if any of the virtual faces containing that vertex intersect.

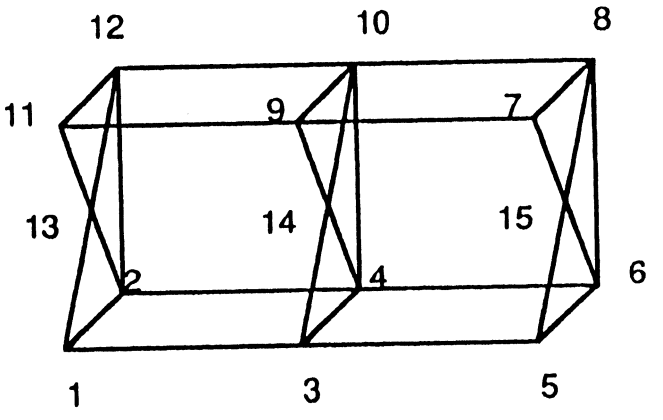


Figure 4.4: Pseudo Wire Frame of the Two Wedge problem

Suppose that f_1 and f_2 are two virtual faces having a type II intersection; the line segments of intersection as new edges, called *cutting edges*, was introduced. Also all the necessary points of intersection were introduced. The new vertices and edges were marked to distinguish them from the original vertices and edges. In general, these new vertices and edges would partition some of the virtual faces into smaller virtual faces.

So,in this stage,all the illegal faces were distinguished and using the algorithm described above all the new virtual faces out of these faces were obtained. In the Two Wedge problem a total of nineteen virtual faces were obtained and out of that, four were intersecting illegally. So cutting edges were introduced in these faces and these four virtual faces yielded eight virtual faces

making twenty three faces for this problem. After the introduction of these new cutting edges all the new connectivities were obtained.

In the Two Wedge problem, two cutting edges [v_{13} v_{14} and v_{14} v_{15} in the figure 4.5] are introduced. These two edges partition four virtual faces (v_1 v_3 v_{14} v_{10} v_{12} v_{13}, v_2 v_4 v_{14} v_9 v_{11} v_{13}, v_3 v_5 v_{15} v_8 v_{10} v_{14}, v_4 v_6 v_{15} v_7 v_9 v_{14}) into eight virtual faces (v_1 v_3 v_{14} v_{13}, v_{14} v_{13} v_{10} v_{12}, v_2 v_4 v_{14} v_{13}, v_{14} v_{13} v_9 v_{11}, v_3 v_5 v_{15} v_{14}, v_{15} v_{14} v_8 v_{10}, v_4 v_6 v_{15} v_{14}, v_{15} v_{14} v_9 v_7).

This is shown in the figure 4.5

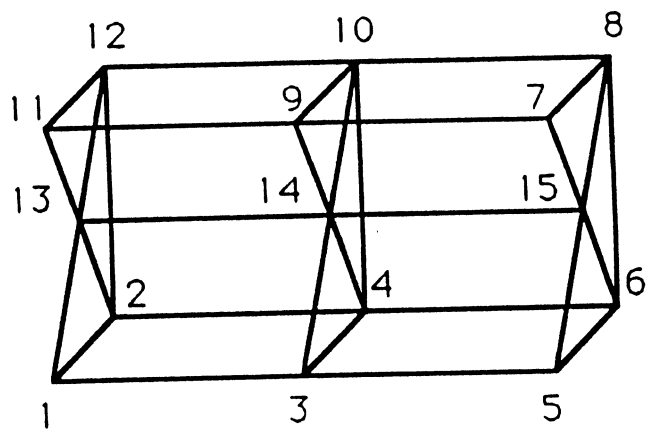


Figure 4.5: The Pseudo Wire Frame with the cutting edges introduced in it.

CHAPTER - 5

VIRTUALBLOCKS AND DECISION PROCESS

In this chapter, a description of the procedure to obtain virtual blocks is given. To obtain that, a circular list of the virtual faces which contain a particular edge was created. These lists were ordered radially around the edge. These lists were used to find all partitioning cycles of the virtual face graph; the nesting relationships among these cycles were found and used to uncover all candidates for solid regions. These candidates were called *virtual blocks*. Virtual blocks were bounded by virtual faces and partitioned $\Re^3$. Any virtual face which did not belong to two different virtual blocks was dropped.

Then a decision process was used to assign solid or hole state to the virtual blocks and all objects with the given projections were found. The process ensured that all cutting edges disappear in solution objects (*i.e.*, that they were either totally surrounded by space or by material, or they separate coplanar surfaces). Efficiency in the search process was 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.

5.1 Calculation of Virtual Blocks

In this stage virtual faces were fitted together to form candidate objects called virtual blocks. From Definition 13, it was clear that objects could be found by calculating all 2-cycles and finding the nesting relationships among them. This 3-D process was a close analog of the 2-D process of fitting edges together to form virtual faces. The definition of a 2-cycle was in terms of F(O), and at this stage of the algorithm, only the virtual faces VF(O) are available, where F(O) $\subseteq$ VF(O). Thus, VF(O) can obtain elements which are not faces of O and are known as *pseudo-faces*. Pseudo-faces arise through chance alignments of edges and may occur in two forms:

I. The interior of the virtual face is empty space;

II. The interior of the virtual face is interior to solid material.

Type I pseudo-faces were always rejected and type II were either rejected or were used to divide primitive objects into smaller subobjects.

An intersection of type I showed that the virtual face involved was really a pseudo-face. Similarly, an intersection of type II indicates that at least one of the virtual faces involved was a pseudo-face. Another kind of pseudo-face that was detected in this stage was a virtual face that does not belong to any 2-cycle. After detecting and handling all of these pseudo-faces, the remaining virtual faces break up into 2-cycles. These 2-cycles partition all of $\Re^3$ into connected components in

much the same way that the 1-cycles partition the planes.

The above concept was made more clear by using an example as shown in the figure 5.1

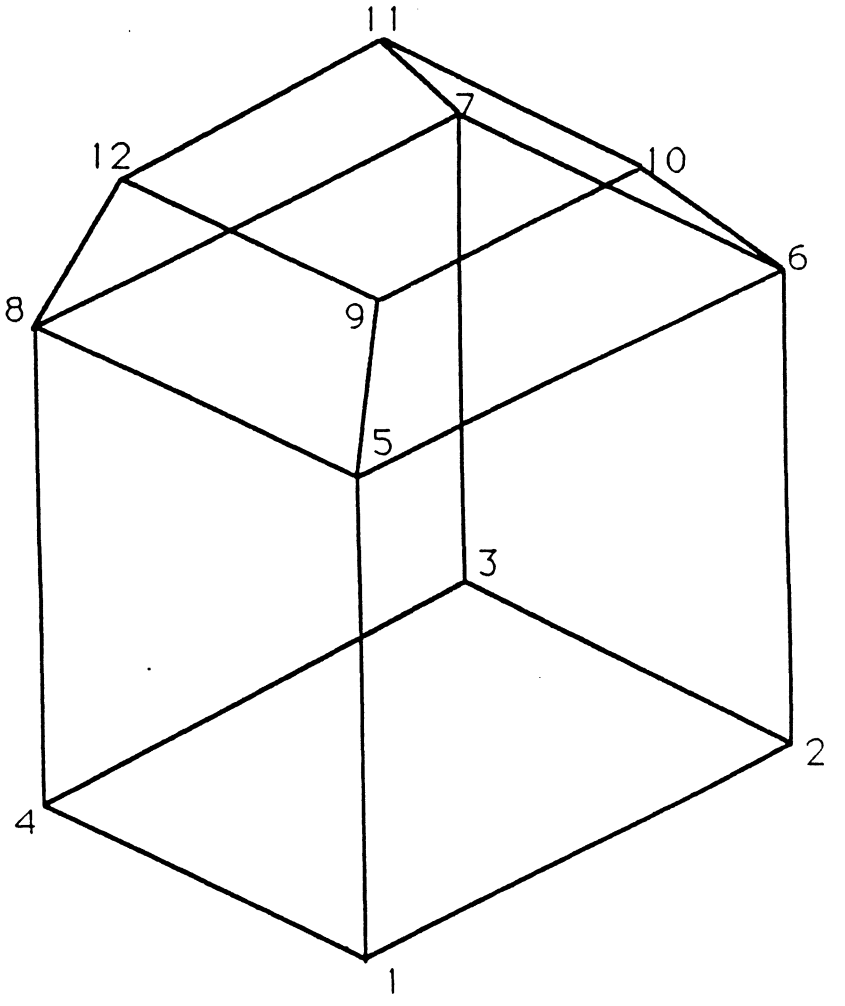


Figure 5.1: An example of pseudo face

An object could have a 1-cycle which results accidentally; as in figure 5.1, the virtual face, v_5 v_6 v_7 v_8 v_5, was a pseudo-face, because it was not an actual

boundary between empty space and solid material. This pseudo-face couldn't be detected until the object was considered globally, *i.e.*, when virtual faces were being found in the various planes, there was no way of distinguishing between faces and pseudo-faces. Only when the construction of the complete object in figure 5.1 was attempted was $v_5\ v_6\ v_7\ v_8\ v_5$ seen to be a pseudo-face.

To obtain the virtual blocks, first a virtual face (VF1) was taken and then one of its edge (VE1) was considered. Then all the virtual faces (VF's) containing that edge (VE1) were obtained. Next each virtual face (let VF2 be the first one), except the virtual face (VF1), which was taken initially from the edge (VE1), was considered and all the edges (VE's) from that particular virtual face (VF2), except the common edge (VE1) were obtained. Next all the edges (let VE2 be the first one) in that particular virtual face (VF2) were considered, and all the virtual faces (VF's) on that edge (VE2) were obtained and then a virtual face (VF3) was considered and all the edges it had were obtained and checked whether they contained any of the edges which were there in the first virtual face (VF1) other than the first common edge (VE1). Then all the faces on this edge were obtained and checked whether they had any of the faces which were same as the first virtual face (VF1). If any of the virtual face was same, then that was the next face for that virtual block, and if none of the faces were common then the next increment was given and the whole process was repeated. For final check, the containment of each edge in two blocks was checked. This part of the algorithm

is explained in the figure 5.3 as a flow chart. It is also explained in the following figure 5.2 taking the example of Two Wedge problem.

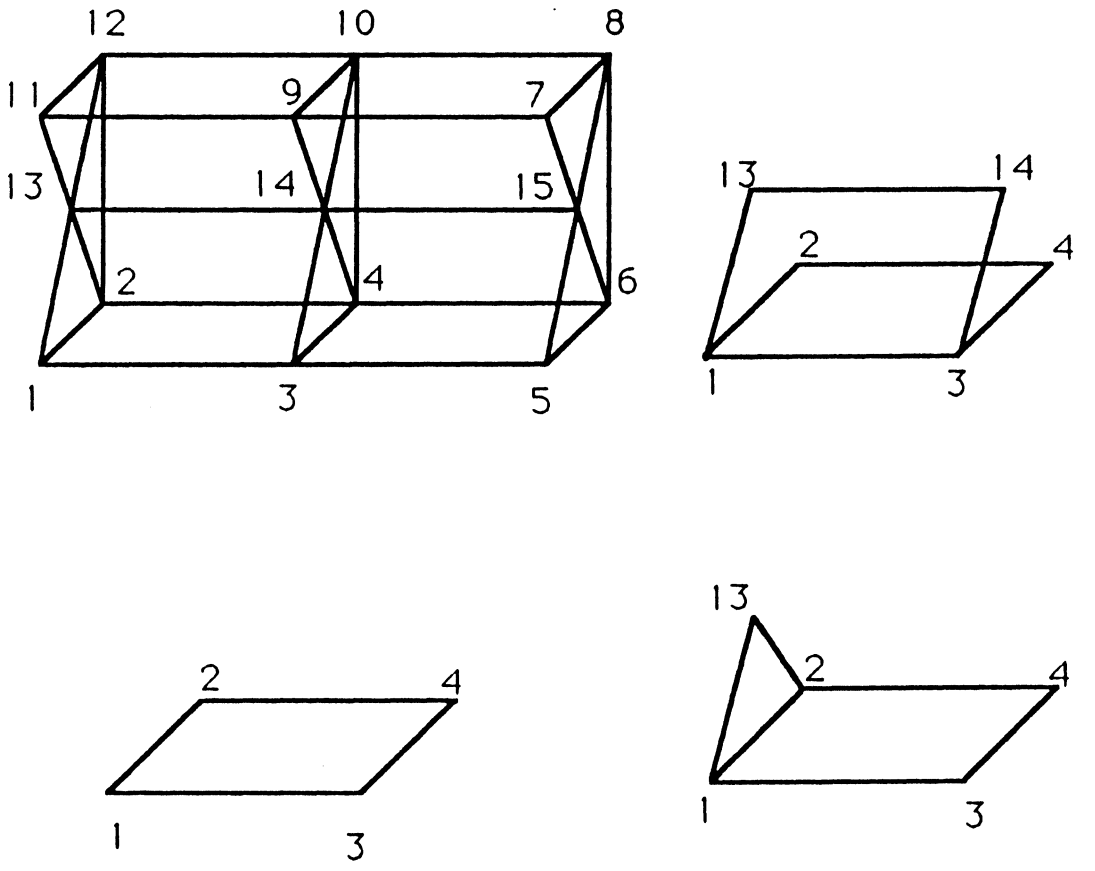
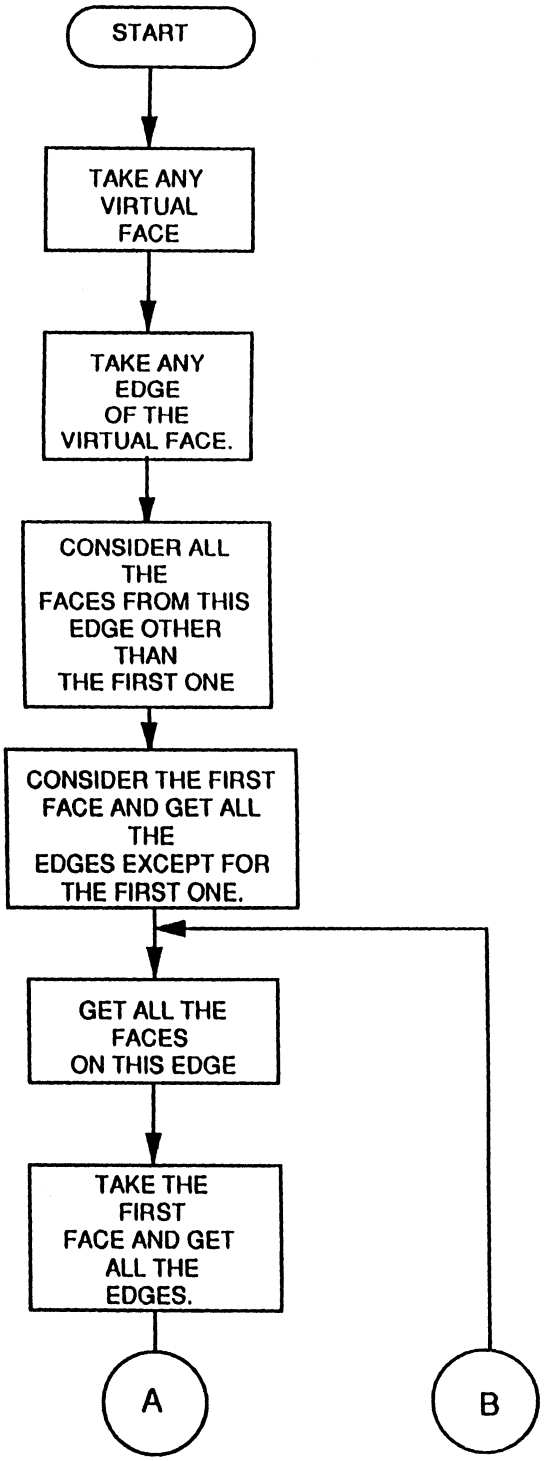


Figure 5.2: The procedure to make virtual blocks.


START
TAKE ANY VIRTUAL FACE
TAKE ANY EDGE OF THE VIRTUAL FACE.
CONSIDER ALL THE FACES FROM THIS EDGE OTHER THAN THE FIRST ONE
CONSIDER THE FIRST FACE AND GET ALL THE EDGES EXCEPT FOR THE FIRST ONE.
GET ALL THE FACES ON THIS EDGE
TAKE THE FIRST FACE AND GET ALL THE EDGES.
A
B

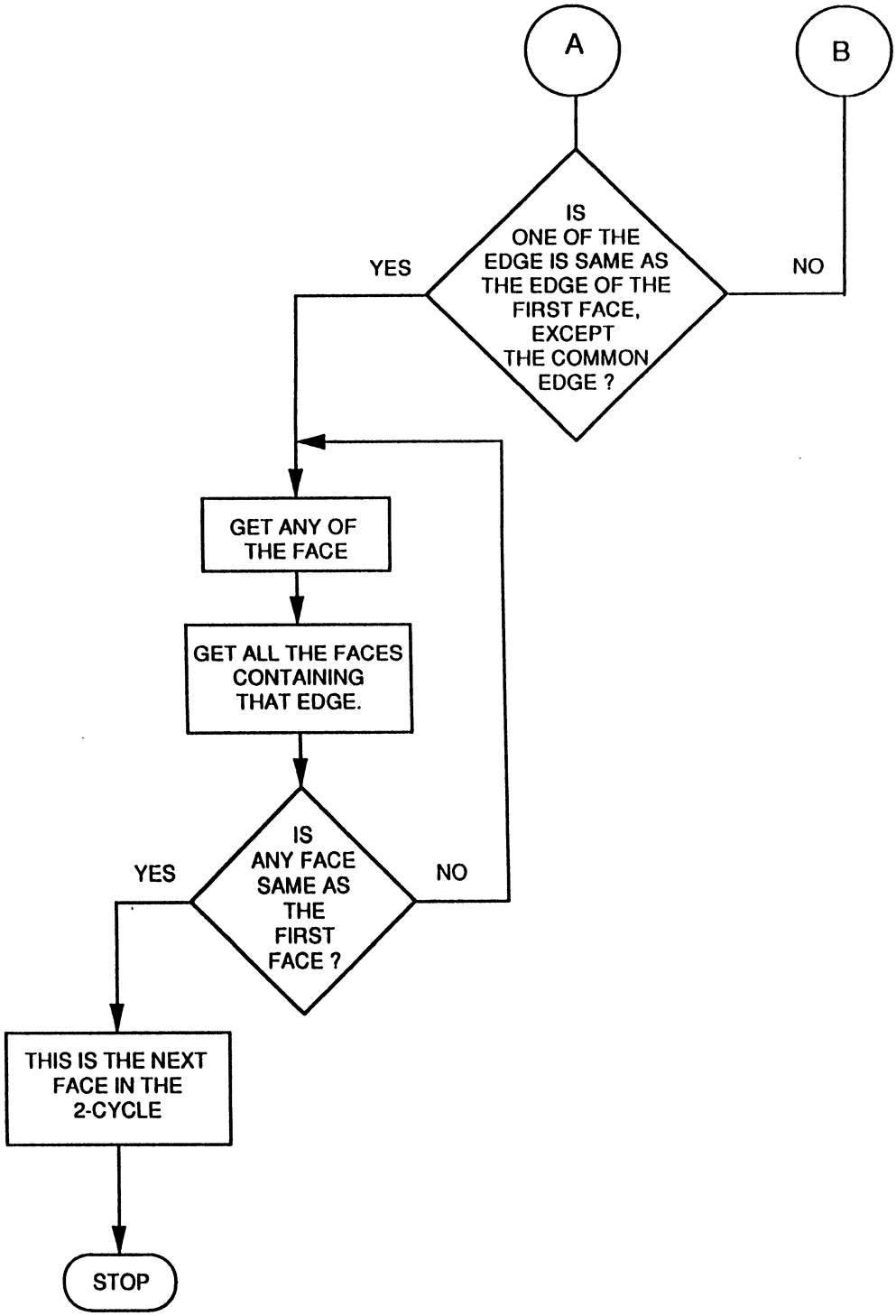


Figure 5.3: Flow Chart for the calculation of 2-Cycles

In the figure 5.2 the virtual face v_1 v_2 v_3 v_4 was considered as VF1 and the edge v_1 v_2 as VE1. The virtual faces containing this edge were v_1 v_2 v_3 v_4 and v_1 v_2 v_{13} . Then the first virtual face other than VF1 was considered, so in this case VF2 was v_1 v_2 v_{13}. Now the other edges of this face were v_1 v_2, v_1 v_{13} and v_2 v_{13} and VE2 was v_1 v_{13} and VE3 was v_2 v_{13}. Next all the virtual faces on VE2 were obtained and the face VF3 in this case was v_1 v_{13} v_{14} v_3 and then the edges of this virtual face were v_1 v_{13}, v_{13} v_{14}, v_{14} v_3 and v_3 v_1. Next it was checked that if one of these edges was same as that of the first face VF1 and that the edge v_3 v_1 was found common to both. Then all the virtual faces on this edge were obtained to check if one of the faces was VF1 or not and it was found that in this case one of the virtual face on the edge v_3 v_1 was VF1, so the face v_1 v_{13} v_{14} v_3 was the next face in the block. Now this face was taken as VF1 and the whole procedure was repeated for this virtual face to get the next virtual face in this block. In the Two Wedge problem, six finite virtual blocks are uncovered and are as shown in the figure 5.4:

B_1:(v_1 v_{13} v_{14} v_3 v_2 v_4),

B_2:(v_{12} v_{13} v_{14} v_9 v_2 v_4),

B_3:(v_3 v_{14} v_{15} v_5 v_6 v_4),

B_4:(v_9 v_{14} v_{15} v_2 v_8 v_{10}),

B_5:(v_9 v_{14} v_{15} v_6 v_8 v_{10}),

B_6:(v_{11} v_{13} v_{14} v_9 v_{10} v_{12}),

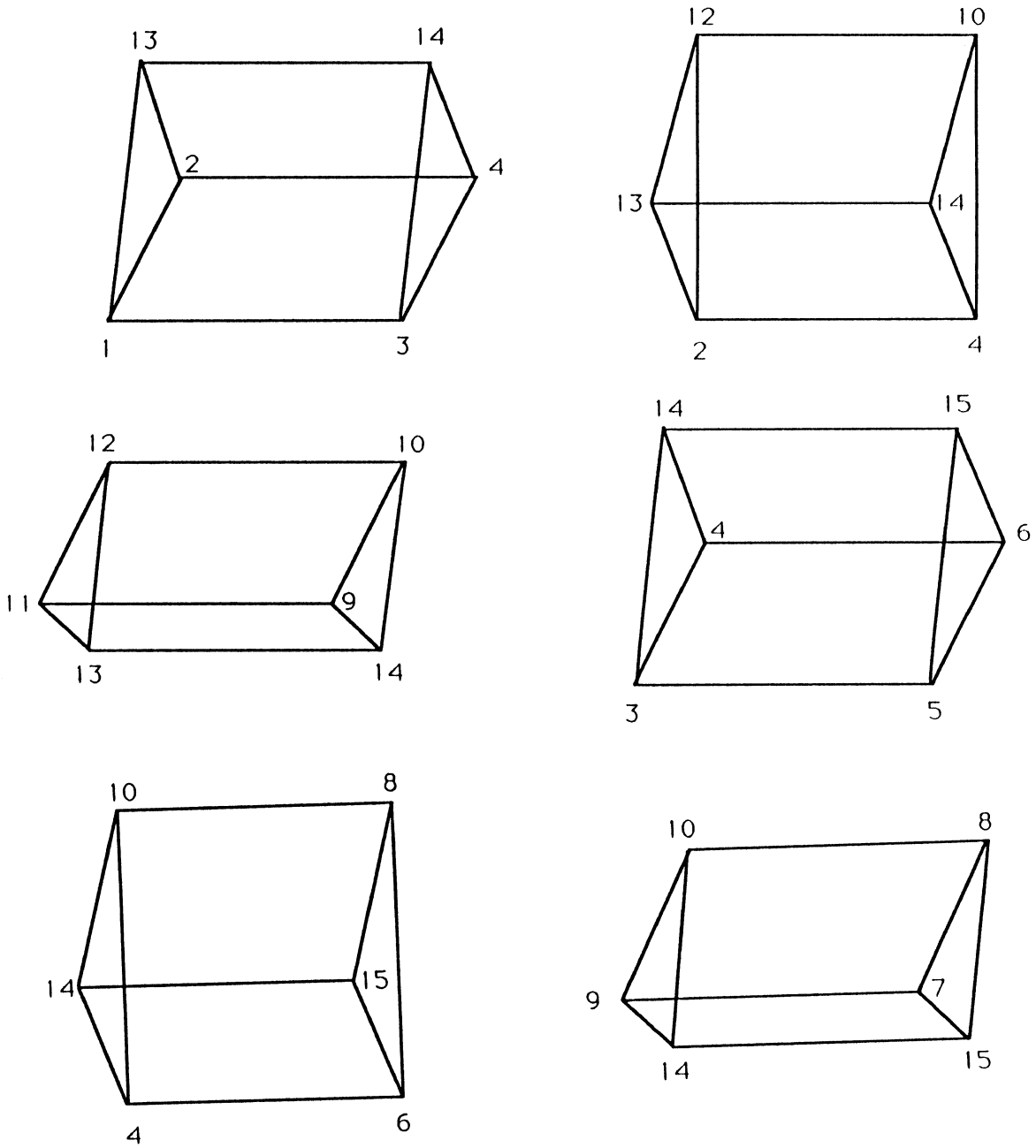


Figure 5.4: Different Virtual blocks in the Two Wedge problem.

5.2 Making Decisions

A decision tree, based on virtual blocks and using a few basic tests, assigns solid or hole state to all virtual blocks and thereby constructs all possible objects having a given set of projections. In this decision process, edges and cutting edges were subsequently removed.

In this stage, virtual blocks were fitted together to generate all objects with the given projections. 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 or hole to the virtual blocks yield an object with the correct wire frame. Assignments of solid or hole state yields correct wire frame if and only if

1. *Every* certain edge element $e \in E(O)$ 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 elements $e \in E(O)$ 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 was performed by assigning states in a virtual block state vector, whose elements were ordered *a priori*. The first element of the state vector was the unique infinite virtual block, which was assigned the empty state. For each edge, a list was formed of the faces containing the edge and the blocks they bound; this list was 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 was checked for consistency and consequential states were assigned. Thus, although the state vector might have dimension of many hundreds, the consistency check might be expected to prune large sections of the tree, while the propagation of consequential states might be expected to reduce substantially the number of decision to be made.

In some cases, particularly those where there were high degrees of symmetry and a limited number of views giving rise to many highly correlated uncertain edges, there might 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 allow a solution to be found efficiently, an exhaustive

search must ultimately be used. Efficient pruning of the decision tree was very important.

In the case of the Two Wedge problem, this stage produces the two solutions shown in figure 5.5, the decision procedure works as follows in this case. Suppose that the search in this case deals with the virtual blocks B_0,....,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.

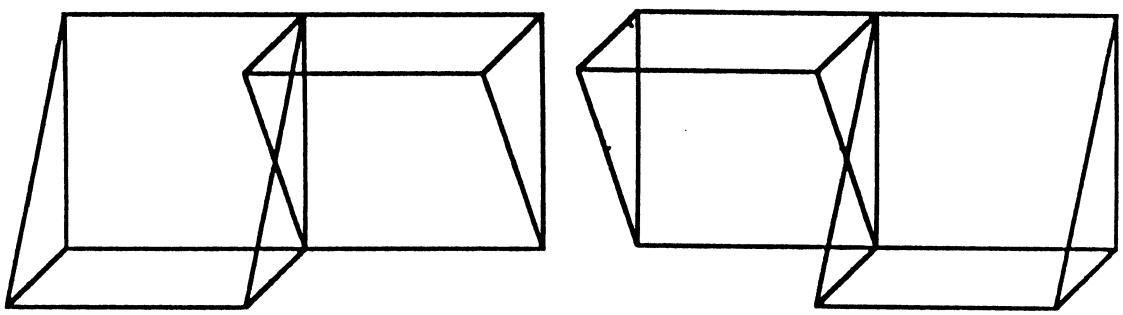


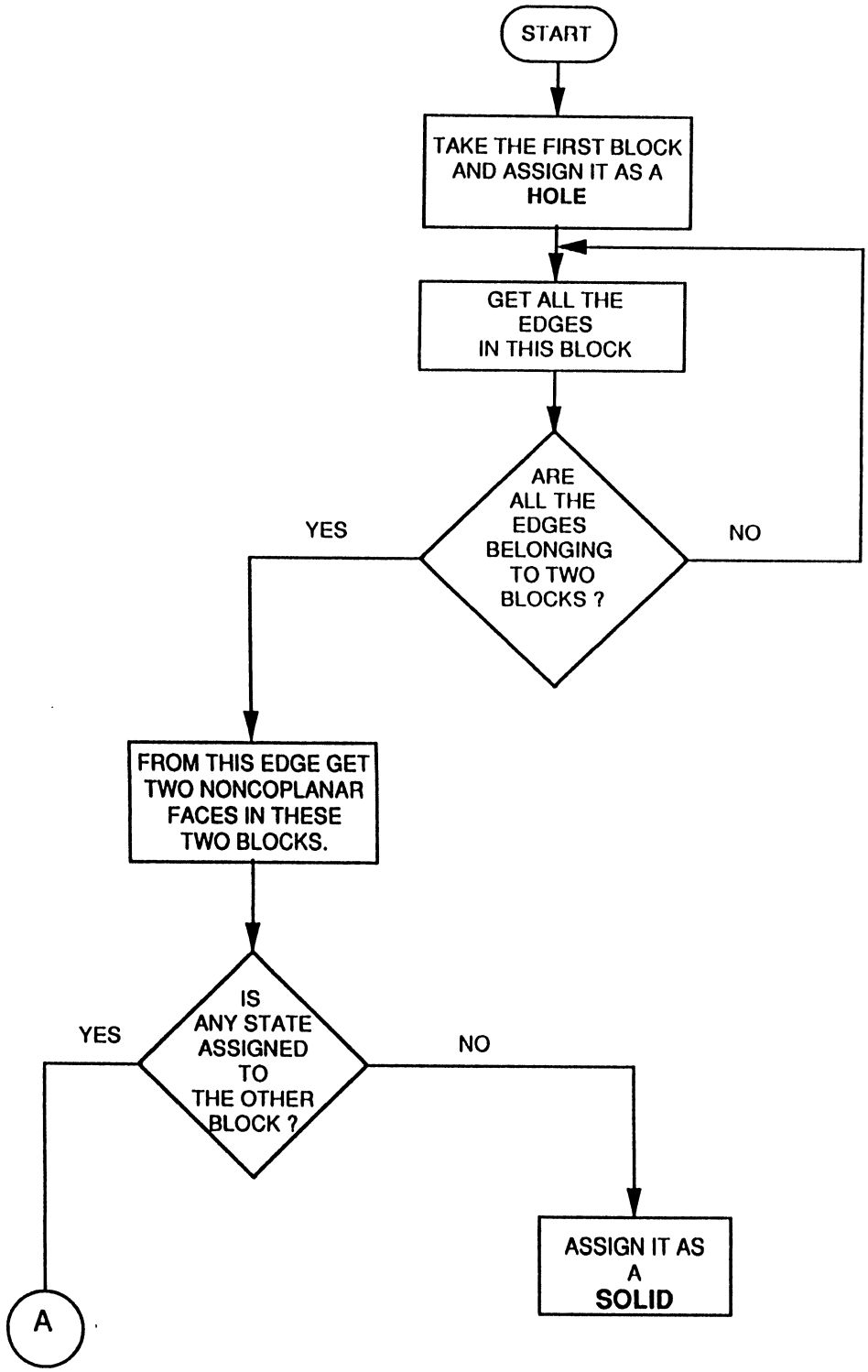
Figure 5.5: The two solutions of the Two Wedge problem.

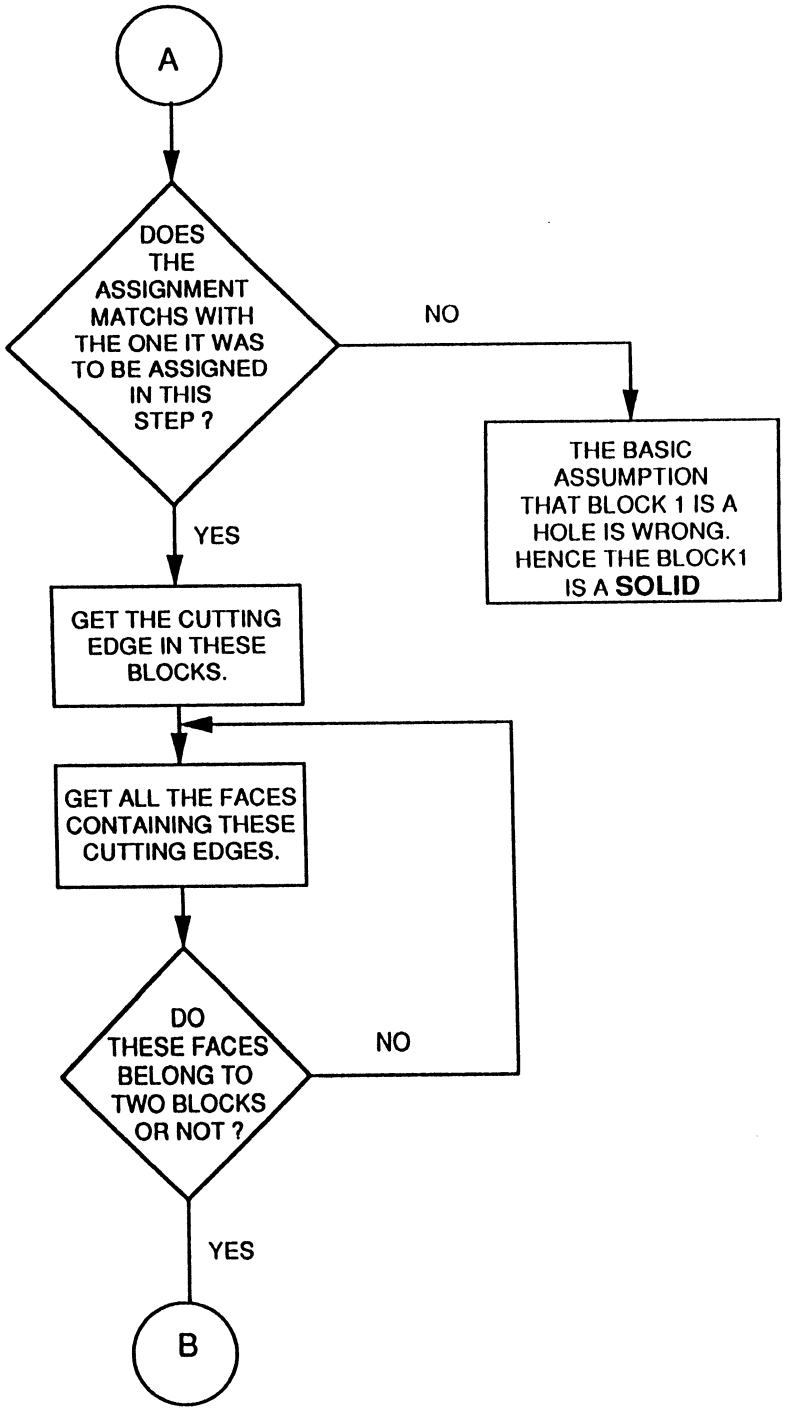
Let it be assumed that the first block be B_1 and also that it is a hole. Then, the first edge of this block is taken and all the faces which have this edge, were

obtained in all the blocks. Next all the faces in the first block B_1 were obtained. Each edge was checked to see if it was common to any two blocks. If it was found to be so, then from this edge, two noncoplanar faces in these two blocks were obtained. If this edge was not common to any two blocks, the algorithm proceeded to the next edge. Next a check was made to ascertain whether the block other than the original block had any state assigned to it. If no state had been assigned to the other block then it was assigned as a solid. If the block had some state assigned to it then a check was made to ascertain the state. If this state matched with the one it was to be assigned, if it didn't have any previous assignment, then proceed for the cutting edge requirement. If the assignment did not match with the one it was to be assigned, then the basic assumption that the block B_1 was a hole was wrong and hence was assigned solid.

To satisfy the second requirement of the cutting edges the cutting edges in the above mentioned two blocks were considered. Next all the faces in these blocks were obtained and checked again to see if these faces belonged to two blocks or not. If these faces belonged to two blocks then two noncoplanar faces in these two blocks were obtained and were checked to see if each one of them belonged to atleast one solid and one hole block then the basic assumption was wrong, and the block B_1 was a solid. If, on the other hand the faces did not belong to two blocks, then the basic assumption was correct and the block B_1 was a hole. All this is also shown in the figure 5.6 as a flow chart.

In the case of this Two Wedge problem, let block B_1 be a hole. Then the first edge v_1 v_2 was considered and checked to see if it belonged only to one block, *i.e.,* to block B_1 so, the next edge which was contained in two blocks *i.e.,* edge v_2 v_4 was considered and the two noncoplanar faces which contain this edge were obtained. The faces were v_1 v_2 v_4 v_3 and v_2 v_4 v_{10} v_{12} from the block B_1 and block B_3 respectively. At this stage the block B_3 did not have any state assigned to it so the block B_3 was assigned to be solid. Next it was checked whether the cutting edge requirement for these two blocks was satisfied or not. The cutting edge for these two blocks was v_{13} v_{14} and the two noncoplanar faces were v_2 v_{13} v_{14} v_4 and v_{13} v_{14} v_{12} v_{10} and they belonged to one solid and one hole block, so the assumption that the first block has a hole state is correct and is stored in the heuristic tree. Similarly the other edges in this block are considered and the assignment of the state to


START
TAKE THE FIRST BLOCK AND ASSIGN IT AS A HOLE
GET ALL THE EDGES IN THIS BLOCK
ARE ALL THE EDGES BELONGING TO TWO BLOCKS ?
YES
NO
FROM THIS EDGE GET TWO NONCOPLANAR FACES IN THESE TWO BLOCKS.
IS ANY STATE ASSIGNED TO THE OTHER BLOCK ?
YES
NO
ASSIGN IT AS A SOLID
A


A
DOES THE ASSIGNMENT MATCHS WITH THE ONE IT WAS TO BE ASSIGNED IN THIS STEP ?
NO
THE BASIC ASSUMPTION THAT BLOCK 1 IS A HOLE IS WRONG. HENCE THE BLOCK1 IS A SOLID
YES
GET THE CUTTING EDGE IN THESE BLOCKS.
GET ALL THE FACES CONTAINING THESE CUTTING EDGES.
DO THESE FACES BELONG TO TWO BLOCKS OR NOT ?
NO
YES
B

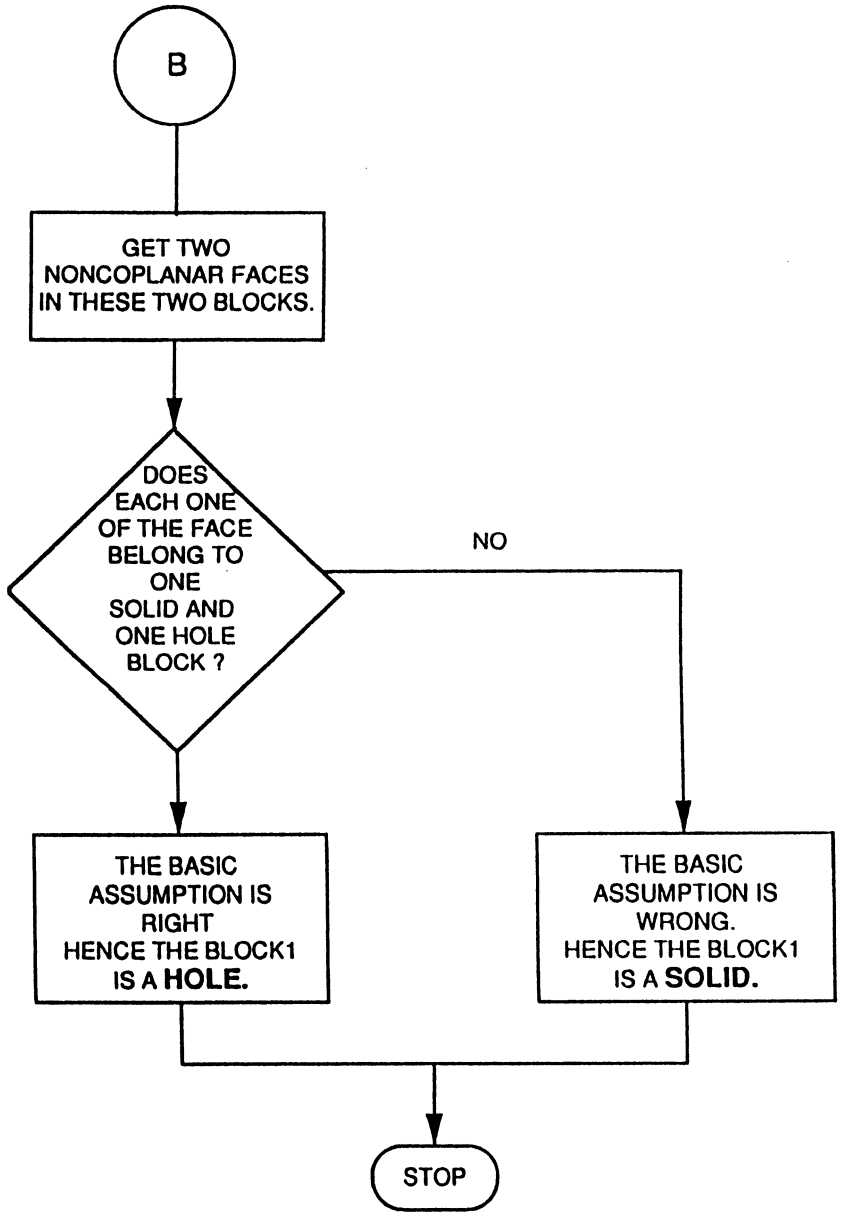


Figure 5.6: Flow Chart for the procedure of making decisions.

the other blocks is done and afterwards the next block is considered and same procedure is repeated.

The decision tree of the Two Wedge problem is as shown in the figure 5.6

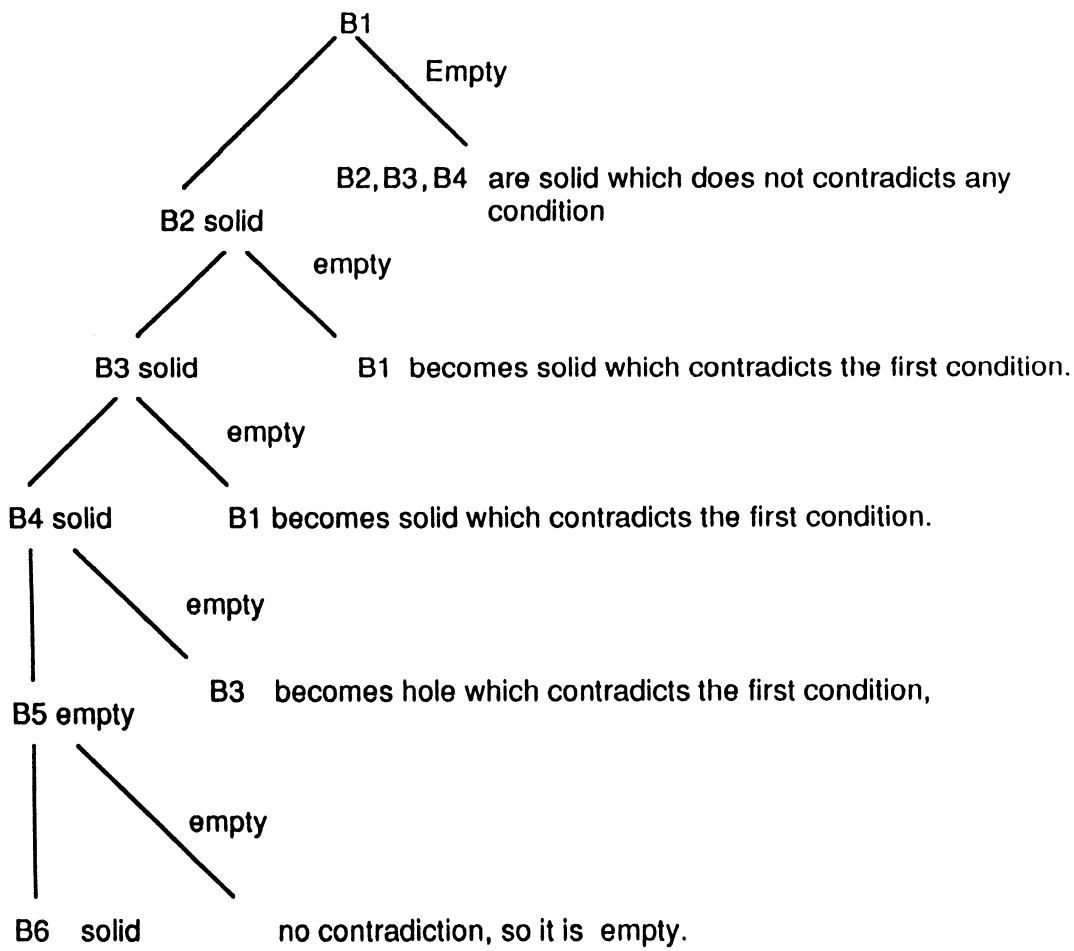


Figure 5.7: Decision Tree

CHAPTER - 6

CONCLUSION

6.1 CONCLUSION

In this thesis an attempt has been made to tackle the problem of conversion of 2-D orthographic views into 3-D models. From the literature survey and the study on the existing software available, it was found that not enough research had been done in this field.

In most softwares, each 2-D view is considered and manipulated separately to obtain a 3-D drawing. However, in the current research, a successful attempt has been made for the automatic conversion of 2-D orthographic views into 3-D drawing. In this algorithm coordinates and connectivities of each view are inputted. The vertices in each view are then back projected to find all the vertices formed by the intersection of noncoplanar edges. Edges are then introduced based on the connectivities in the 2-D views. A pseudo wire frame of the object was obtained from this knowledge of vertices and edges.

Planes are formed for more than two noncollinear edges. From these planes faces of the object are found out. These faces are nested together to form blocks. A final decision process assigns solid or hole state to each block.

Using the above approach a successful computer program has been written which converts 2-D orthographic views into 3-D solid model.

Limitations and Recommendations

This program is capable of handling only straight line edges. It is not capable of projecting any circular edges unless it is approximated by straight lines in the views. A similar algorithm can be developed to include the circular edges also. To resolve all the ambiguities the user might have to input some extra information like a detailed view of the particular section or the left side view. Another difficulty here is that all the inputs are fed into the program by the user in the form of data files. A digitizer can be used to digitize the different views and the coordinates and connectivites can be obtained from this digitized information. The program accepts a neutral file as input but can be modified to accept an IGES format file or any other graphics format as a standard input, so that program becomes compatible with other CAD softwares. The scanned information can also

be written in the form of a neutral file or in any standard format as IGES. Hidden line removal could be done using any of the existing hidden line programs.

CHAPTER - 7

REFERENCES

References

1. A. G. Requicha R. B. Tilove, "Mathematical Foundation of Constructive Solid Geometry: General Topology of Closed Sets," Technical memo. No. 27, Production Automation Project, University of Rochester, New York.
2. J. G. Hocking and G. S. Young, Topology, Addison-Wesley Publishing Co.
3. Braid, I. C., "The Synthesis of Solid Bounded by Many Faces", Comm. ACM.
4. George Markowsky and Michael A. Wesely, "Fleshing Out Wire Frames," IBM Journal of Research and Development.
5. George Markowsky and Michael A. Wesely, "Fleshing Out Projections," IBM Journal of Research and Development.
6. M. Idesawa, "A System to Generate a Solid Figure from Three View," Bulletin JSME

7. Stenstran and Connally, "Wireframe of polyhedra from multiple range views of a scene."

8. Lequette, R., "Automatic Construction of Curvilinear Solids from Wire Frame Views ", Computer Aided Design

9. I. Sutherland,"Three Dimensional Data Input by Tablet," Proceeding IEEE

10. G. Lafue, "Recognition of Three-Dimensional Objects from Orthographic Views," Proceeding of the International Joint Computer Symposium.

11. T. C. Woo and J. M. Hammer, "Reconstruction of Three-Dimensional Designs from Orthographic Projections," Proceedings of 9th CRIP Conference.

12. M. A. Wesley, T.Lozano-Perez, L. I. Lieberman, M. A. Lavin, and D. D. Grossman, "A Geometric Modeling System for Automated Mechanical Assembly," IBM Journal of Research and Development.

13. M. A. Wesley, "Construction and Use of Geometric Models," Chapter 2, Computer Aided Design, J. Encarnacao.

14. Aldefeld, B. And Richter, H., "Semiautomatic Three Dimensional Interpretation of Line Drawings", Computers and Graphics.

15. Preiss, K.,"Constructing the 3-D Representation of a Plane-Faced Object from a Digitized Engineering Drawing", Proceedings of International Artificial Intelligence Conference, Brighton, England.

16. Goldman, R. N., "Two Approaches to a Computer Model for Quadric Surfaces", IEEE Computer Graphics Application Volume 3 (September 1983).

17. T. C. Woo,"Computer Understanding of Design", Ph.D. Thesis, University of Illinois at Urbana-Champaign, 1975.

18. A. Baer, C. Eastman, M. Henrion, "Geometric Modeling: a Survey", Computer Aided Design.

19. H. B. Voelker , A. A. G. Requicha, "geometric Modeling of Mechanical Parts and Processes", Computers.

20. M. E. Mortenson, "Geometric Modeling".

21. M. S. Pickett, J. W. Boyse, "Solid Modeling by Computers from Theory to Application".

22. M. Mantyla, "An Introduction to Solid Modeling".

23. M. E. Mortenson ,"AN Introduction to the Mathematics and Geometry -Computer Graphics".

APPENDIX - A

This appendix shows a typical input to the computer program. It has two data files for each view one consisting of coordinates of the vertices of the view and the other connectivities of the vertices.

Front View

Number	Vertex	X-Coordinate	Y-Coordinate
1	1	0.0	0.0
2	2	1.0	0.0
3	3	2.0	0.0
4	4	2.0	2.0
5	5	1.0	2.0
6	6	1.0	2.0

Table 1 : Coordinates of the vertices of the front view.

Number	Vertex 1	Vertex 2
1	1	2
2	2	3
3	3	4
4	4	5
5	5	6
6	6	1
7	2	5

Table 2: Connectivities of front view

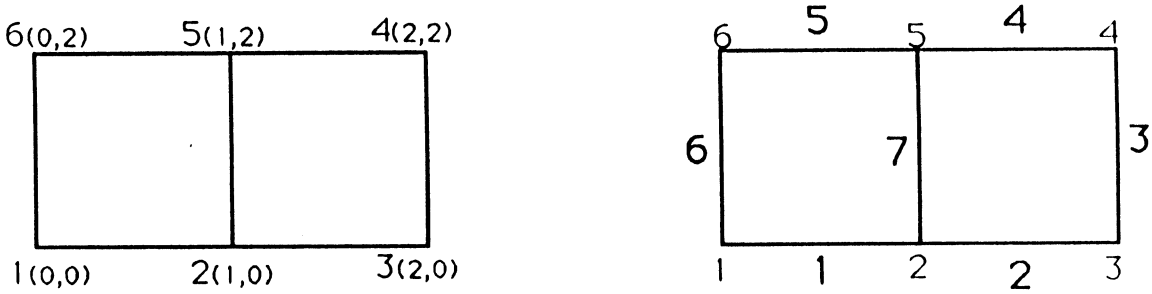


Figure A.1: Coordinates and connectivities of the Front View

Top View

Number	Vertices	X-Coordinate	Z-Coordinate
1	6	0.0	0.0
2	5	1.0	0.0
3	4	2.0	0.0
4	7	2.0	2.0
5	8	1.0	2.0
6	9	0.0	2.0

Table 3: Coordinates of the Top View.

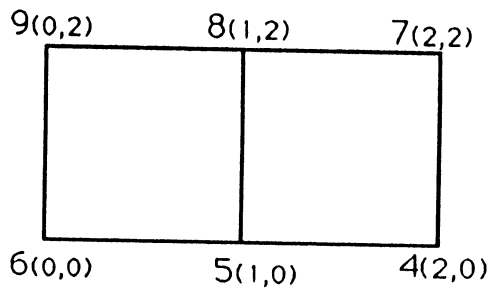


Figure A.2: Coordinates of the Top View.

Number	Vertex 1	Vertex 2
1	6	5
2	5	4
3	4	7
4	7	8
5	8	9
6	9	6
7	5	8

Table 4: Connectivities of the Top View

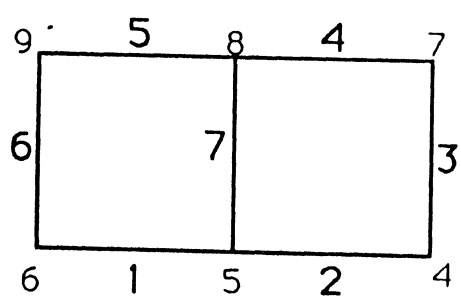


Figure A.3: Connectivities of the Top View

Number	Vertex	Y-Coordinate	Z-Coordinate
1	3	0.0	0.0
2	4	2.0	0.0
3	7	2.0	2.0
4	10	0.0	2.0

Table 5: Coordinates of the Side View

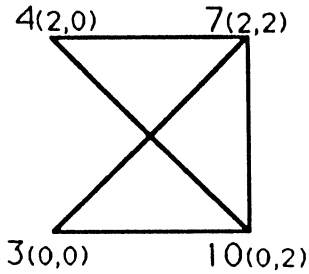


Figure A.4: Coordinates of Side View

Number	Vertex 1	Vertex 2
1	3	10
2	10	7
3	7	4
4	4	10
5	3	7

Table 6: Connectivities of the Side View

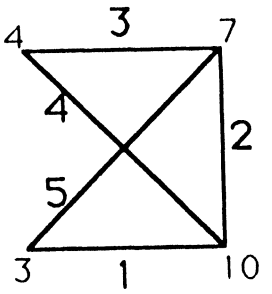


Figure A.5: Connectivities of the Side View.

APPENDIX - B

This appendix shows two sample solutions.

EXAMPLE 1

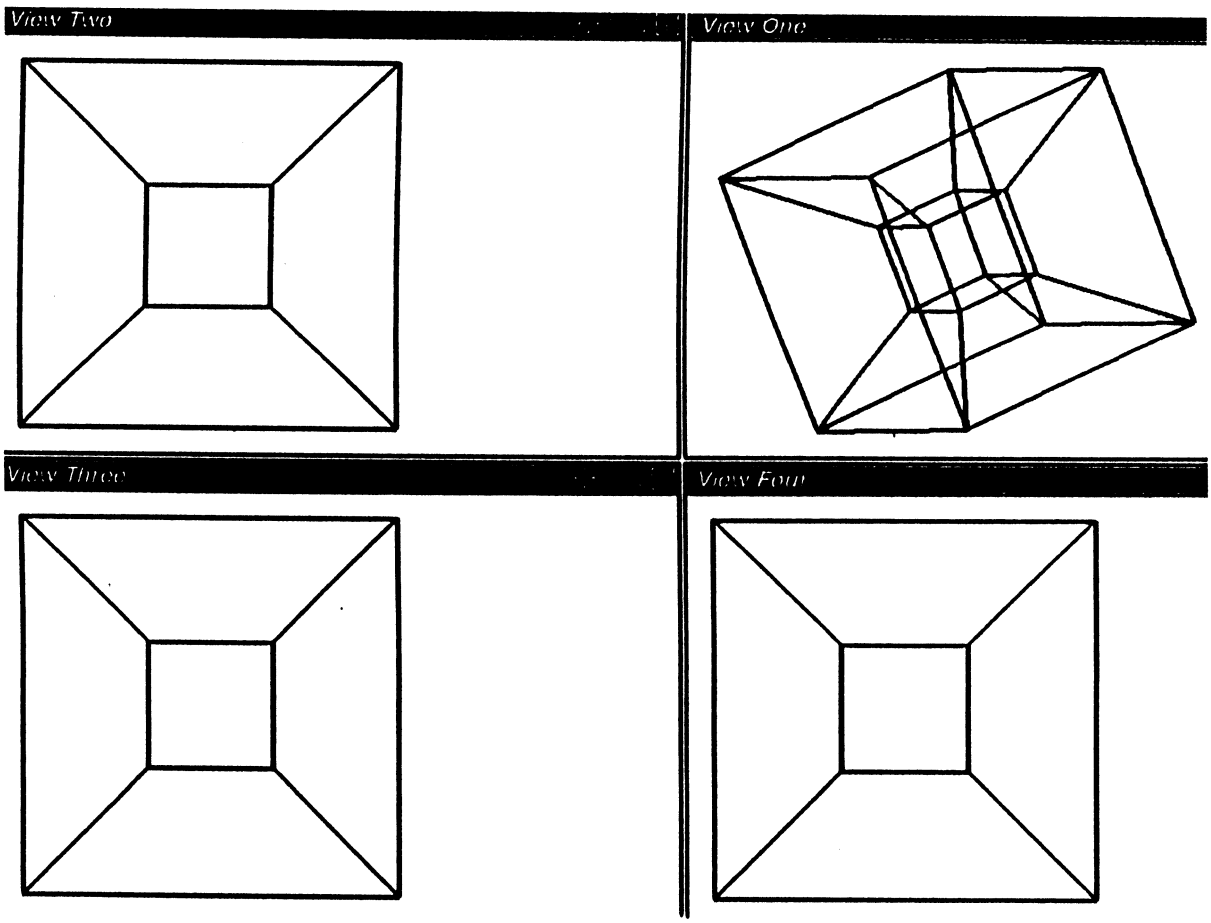


Figure A2.1 : The Three 2-D Views and the 3-D Model of the object in Example 1

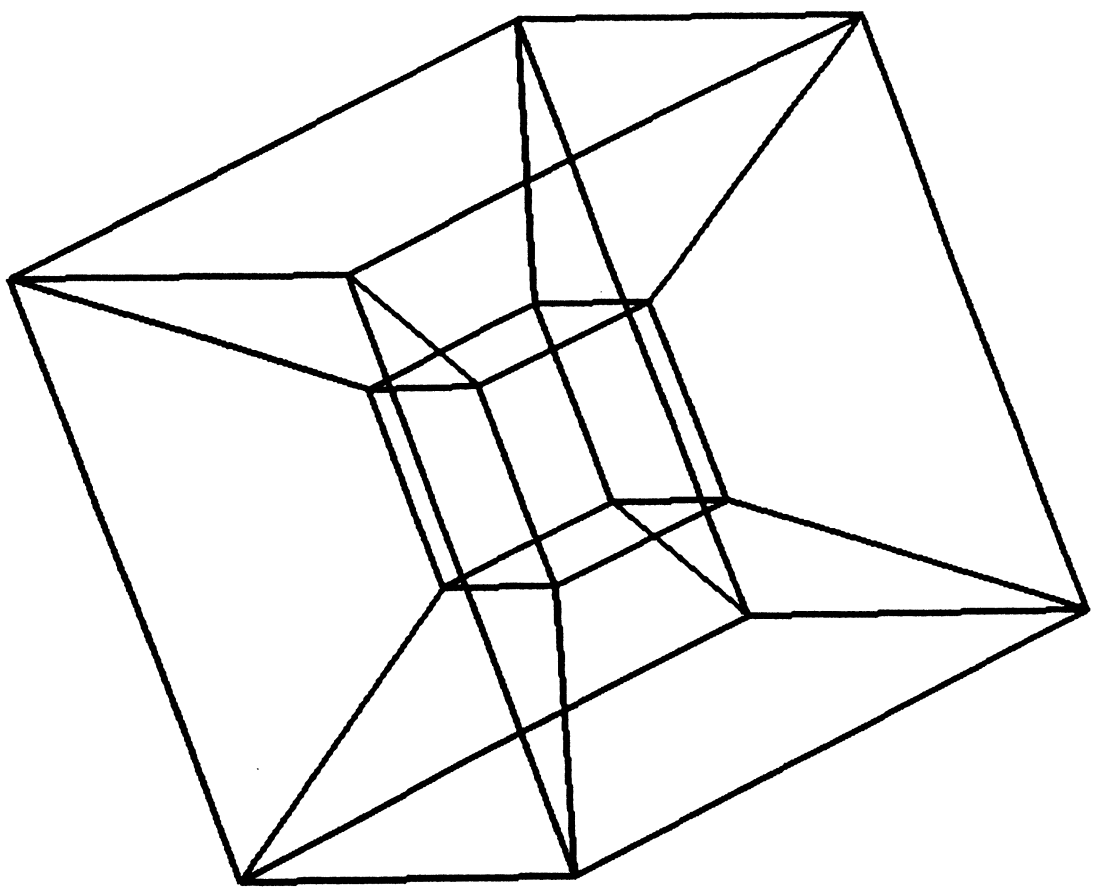


Figure A2.2 : 3-D Model of the object in Example 1.

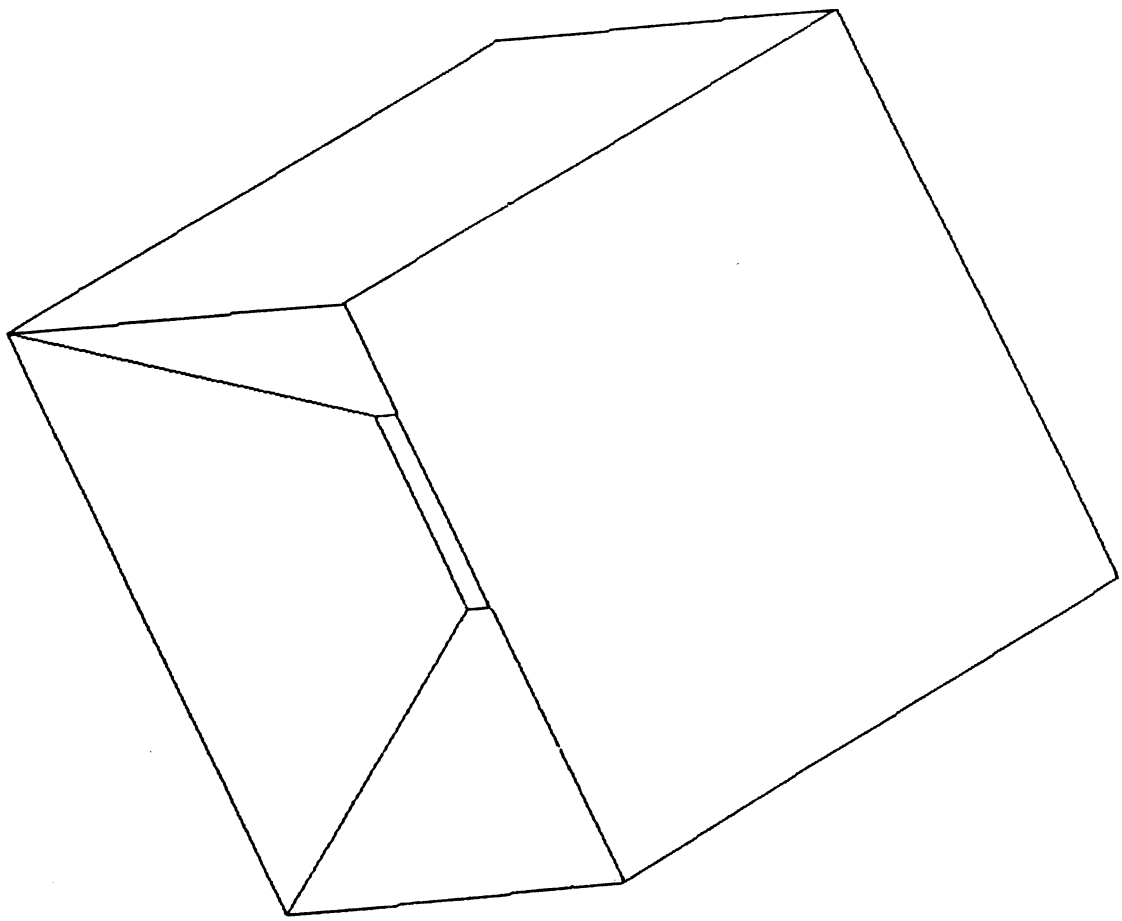


Figure A2.3 : 3-D Solid Model with Hidden Line Removal.

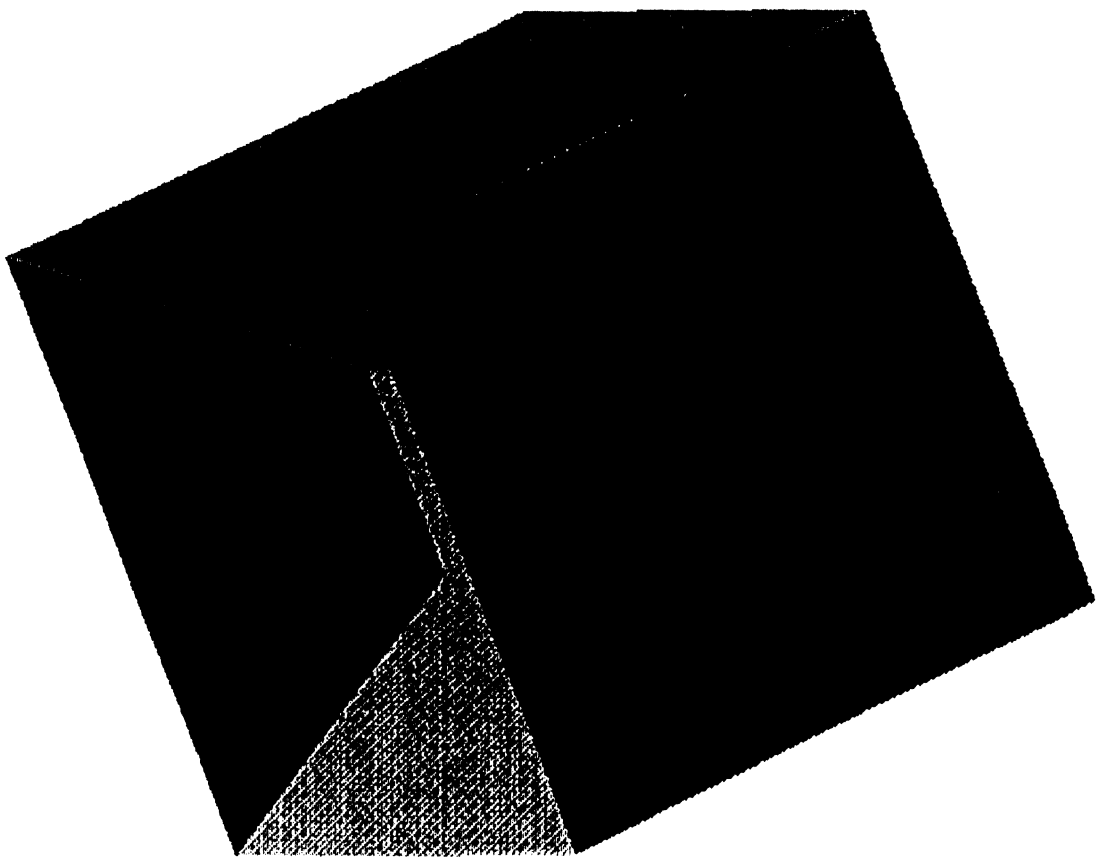


Figure A2.4 : 3-D Solid Model with Hidden line Removal and Shading.

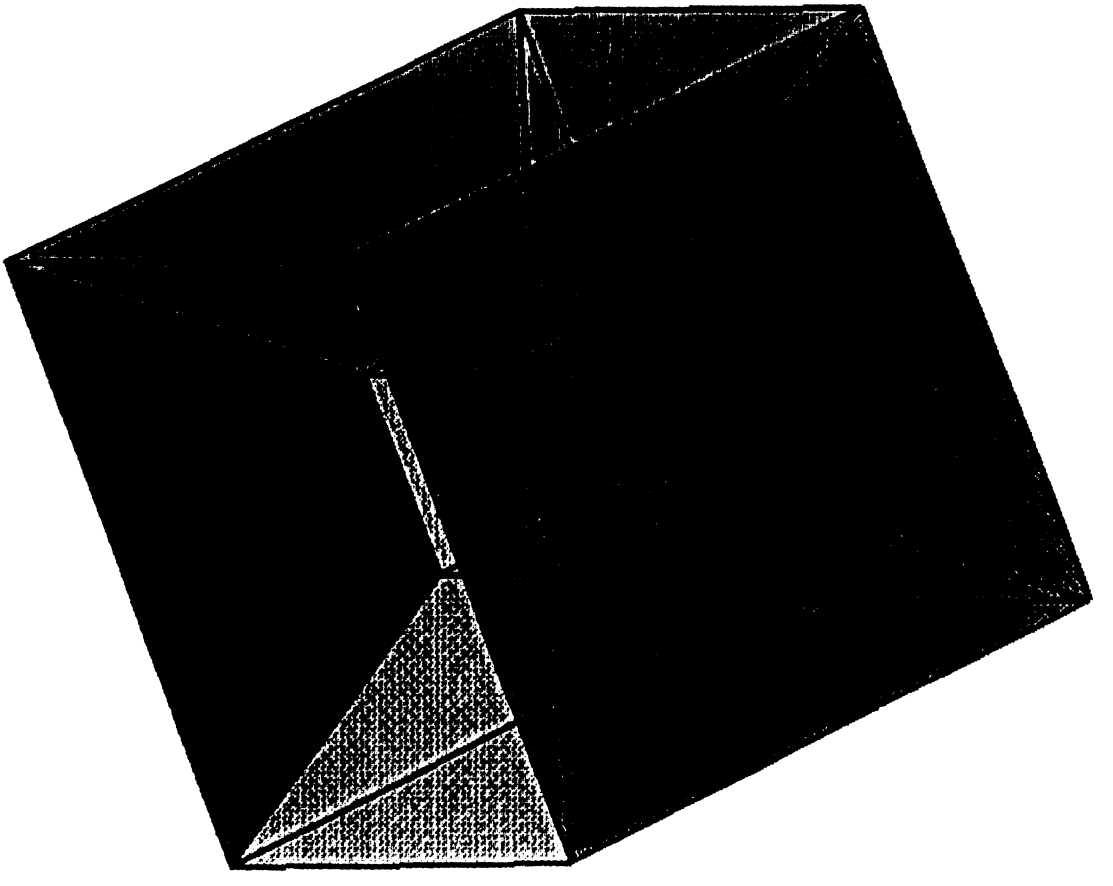


Figure A2.5 : 3-D Solid Model with Hidden Line Removal and Shading with Wireframe superimposed on it.

EXAMPLE 2

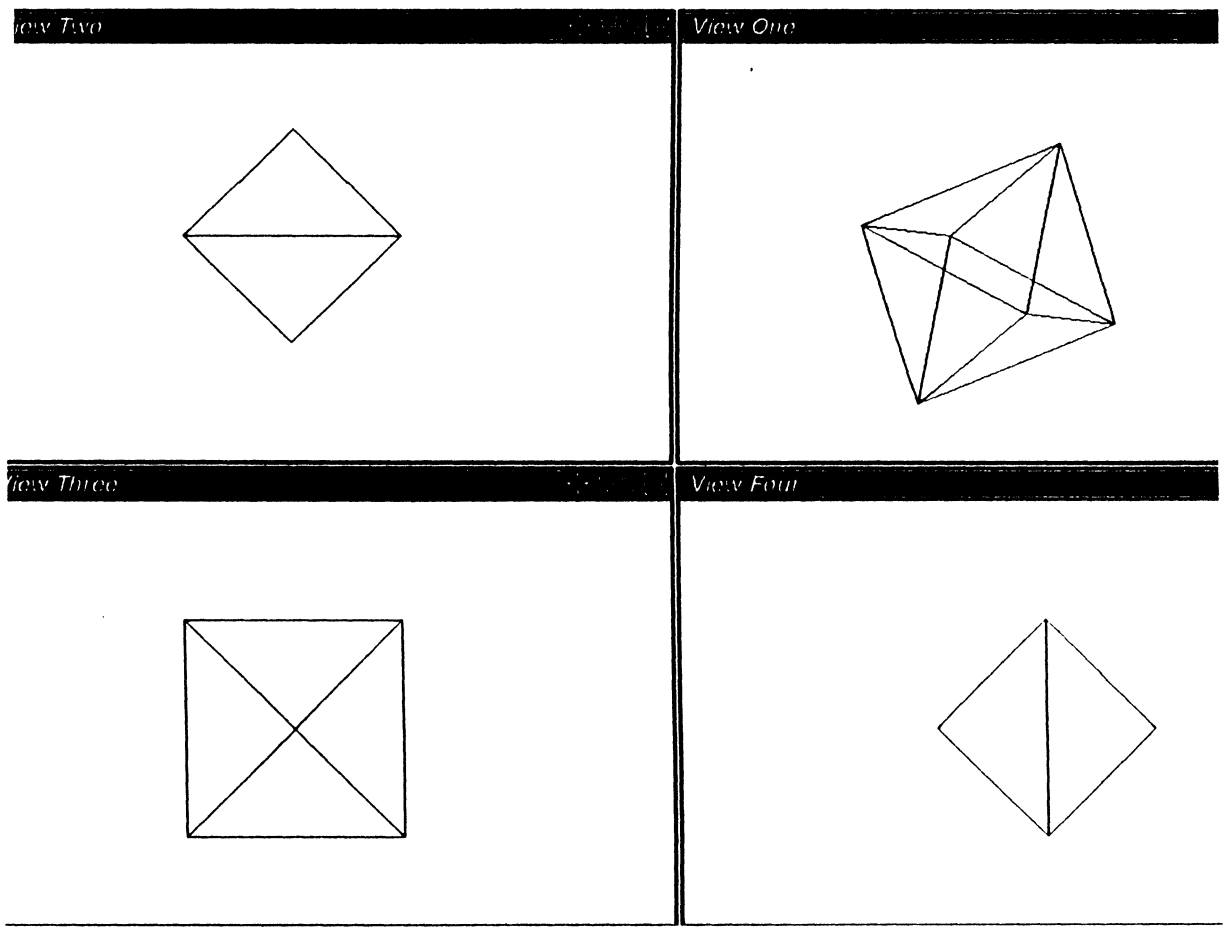


Figure A2.6 : The Three 2-D Views and the 3-D Model of the object in Example 2

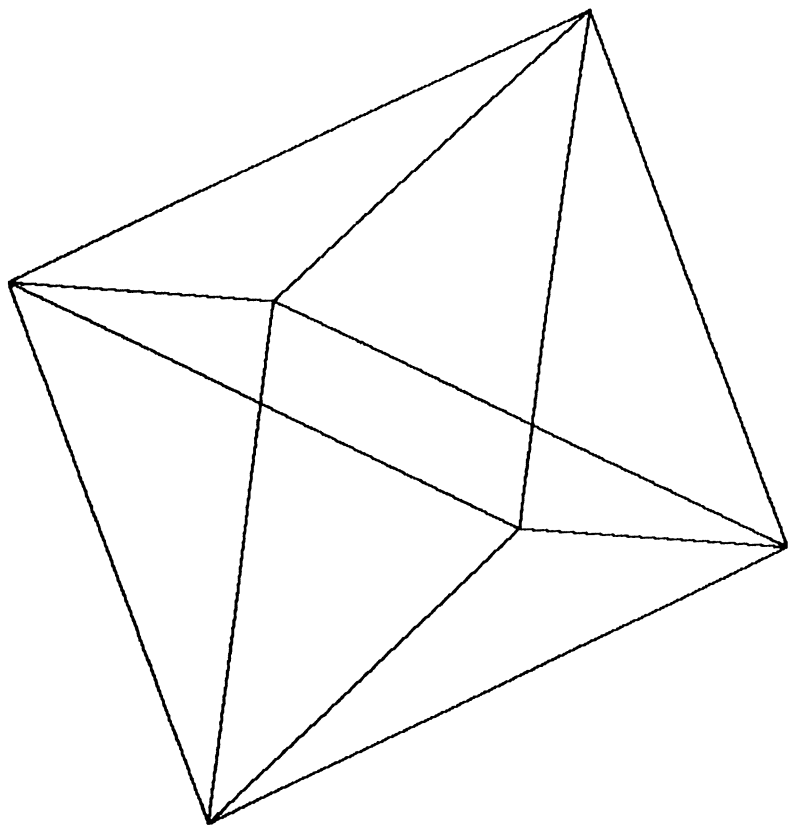


Figure A2.7 : 3-D Model of the object in Example 2.

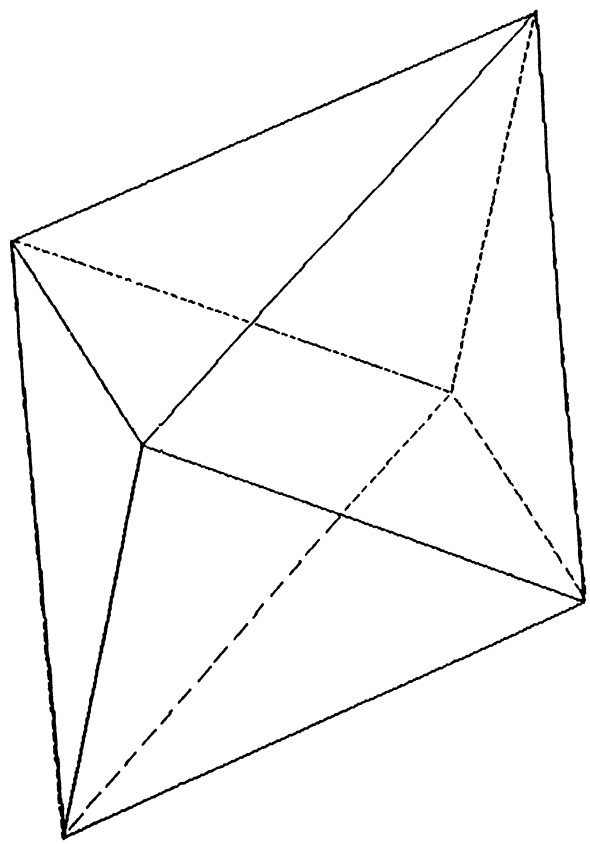


Figure A2.8 : 3-D Solid Model with Hidden Line Removal.

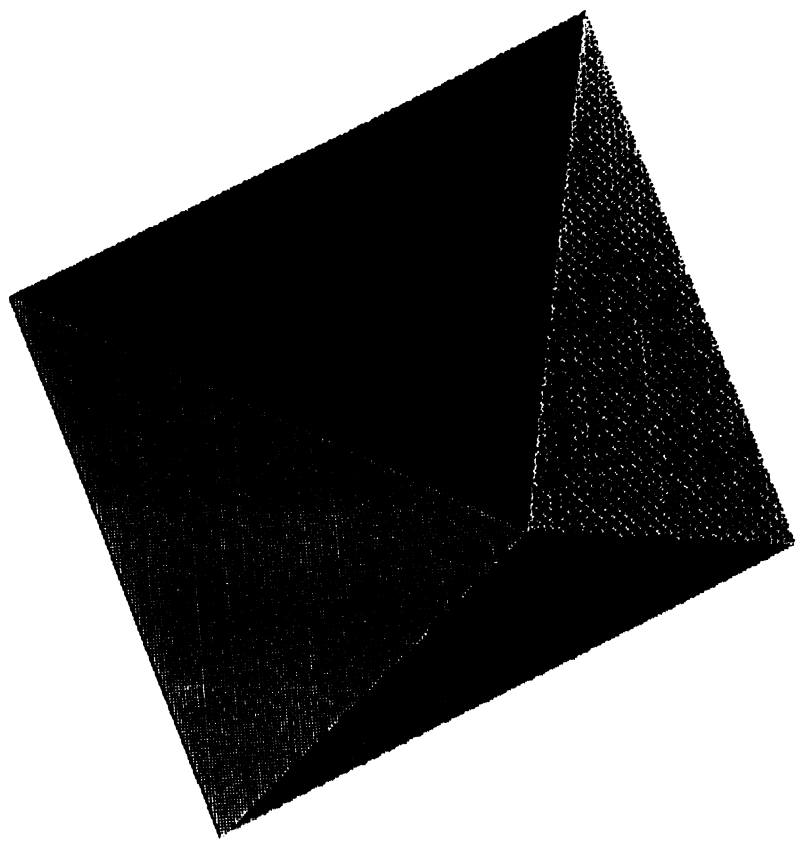


Figure A2.9 : 3-D Solid Model with Hidden Line Removal And Shading.

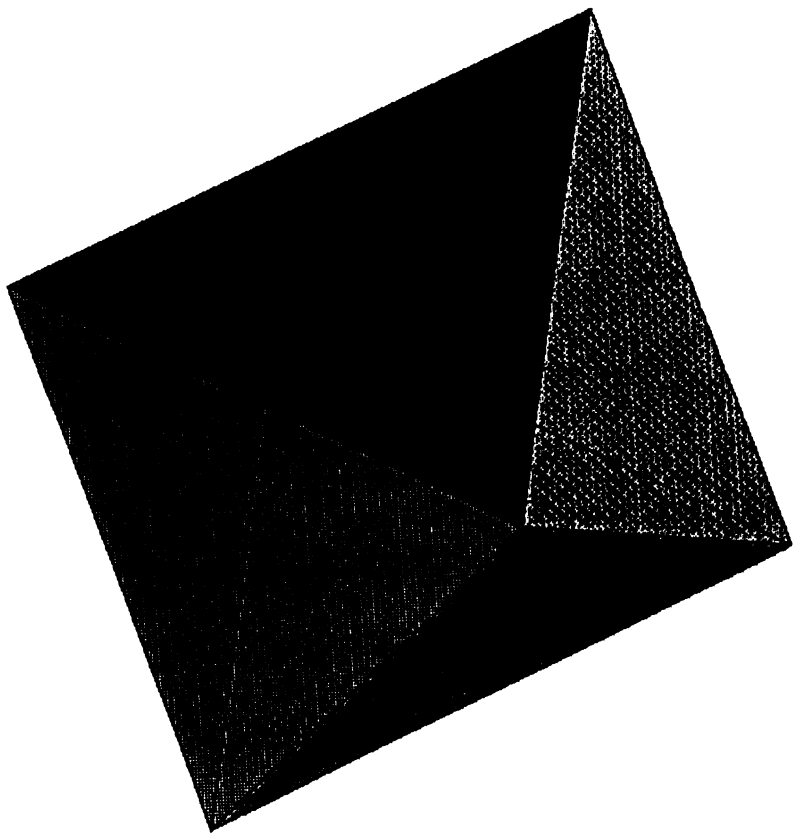


Figure A2.10 : 3-D Solid Model with Hidden Line Removal and Shading with wireframe superimposed on it.