International Journal of Computer Integrated Manufacturing
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tcim20

Automatic construction of CSG solids from a single isometric drawing

M. C. Wu & M.S. Lin

Published online: 08 Nov 2010.

To cite this article: M. C. Wu & M.S. Lin (1996) Automatic construction of CSG solids from a single isometric drawing, International Journal of Computer Integrated Manufacturing, 9:1, 1-21, DOI: 10.1080/095119296131779

To link to this article: http://dx.doi.org/10.1080/095119296131779

INT. J. COMPUTER INTEGRATED MANUFACTURING, 1996, VOL. 9, NO. 1, 1–21

Automatic construction of CSG solids from a single isometric drawing

M. C. WU and M. S. LIN

Abstract. This paper presents an algorithm for constructing a 3D workpiece in CSG (constructive solid geometry) representation from a single 2D isometric drawing. Currently, the concerned workpieces are limited to simple rectilinear polyhedra where any two faces or edges are either parallel or perpendicular to each other, however, the shape of constructed workpiece can further be modified by the application of wireframe-based approach of variational geometry. Compared to previous relevant work, the proposed algorithm is distinguished by its capability to deal with degenerate 2D line drawings; that is, problems where two distinct edges are projected on to one line can be solved.

1. Introduction

The application of CAD/CAM systems have been recognized as effective tools for enhancing the productivity of factories. Solid modellers, designed to represent a workpiece in computational form, are essentially the core of CAD/CAM systems. CSG (constructive solid geometry) and BREP (boundary representation) are two major representation schemes in solid modellers, which have been widely used in developing CAD/CAM software. However, the manual description of a workpiece in BREP is quite time consuming and error-prone (Requicha 1980), and about 50% of users feel uncomfortable with the use of CSG as a scheme for describing solids (Yoshiura *et al.* 1984). Therefore, developing a user-friendly input scheme for solid modellers has been a significant research topic in CAD/CAM areas.

Traditionally, designers have adopted engineering drawings as the standard communication medium in describing a workpiece, in which the three-view drawings (front, top and side views) are most popularly used. Many researchers therefore aimed to develop algorithms for converting three-view drawings into a solid model, either in CSG or BREP (Wang and Grinstein, 1993). However, adopting the three-view drawings to construct a solid is inherently deficient in three aspects.

First, the information contained in the three-view drawings may be incomplete in describing a 3D solid. That is, a set of three-view drawings may be used to infer more than one 3D solid (Guja and Nagendra 1989). In the discipline of engineering graphics, the three orthographic projection views, though most widely used, only constitute part of the engineering drawings. Other drawings, such as pictorial views, auxiliary views, and sectional views are generally used to aid the description of a workpiece (French *et al.* 1986). However, these drawings have seldom been considered as a part of the input scheme for solid modellers in previous research.

Second, the three orthographic views do not readily enable designers to directly describe their design intention. In the conceptual design stage, most designers tend to use pictorial views such as isometric, oblique or perspective projection views (Tu 1992). When the conceptual design work has been completed, the pictorial views are then expressed in detail by giving their three orthographic projection views with dimensioning.

Third, the procedure of manually converting pictorial views to the three orthographic views is time-consuming and error-prone because the three orthographic views are not 'intuitive' in showing the 3D shape of the workpiece. Manual conversion errors, such as missing out lines or erroneously adding extra lines on certain views, would result in a set of three-views which cannot represent a valid solid.

From the above discussion, it is quite natural to consider pictorial views as an essential part of the input scheme for solid modellers. That is, algorithms for converting the pictorial views of a workpiece into its CSG or BREP are demanded. However, pictorial views involve three classes: axonometric projection, oblique projection and perspective projection views; axonometric projection is further divided into three types: isometric projection, dimetric projection and trimetric projection (French *et al.* 1986). Among these pictorial drawings, isometric views are quite popular.

Authors: M. C. Wu and M. S. Lin, Department of Industrial Engineering and Management, National Chiao Tung University, Hsin Chu, Taiwan.

0951-192X/96 $12·00

This paper presents an algorithm for converting the isometric view of a workpiece into its CSG. Currently, the workpieces concerned are limited to simple rectilinear polyhedra; that is, any two faces or edges on the solid are either parallel or perpendicular to each other. Like the three-view drawings, the information contained in a single isometric view may not be sufficient to describe a unique solid. That is, one isometric view may map to more than one or even an infinite number of solids. In this algorithm, we generate only one solid which tends to have the least number of surface intersections on the hidden part of the workpiece. Note that the shape of a constructed 3D rectilinear solid can further be embellished by the application of a variational geometry approach (Han 1993) so that inclined faces and hole features can be created.

2. Relevant research

Research relevant to the reconstruction of a 3D solid from its 2D line drawing(s) or projection(s) has been reported in several fields such as CAD, artificial intelligence and a computer vision. A thorough survey was given by Wang and Grinstein (1993), in which previous studies are essentially characterized by two criteria: the number of 2D line drawing(s) available (single-view or multi-view), and the internal representation (CSG or BREP) of the reconstructed solid.

The reconstruction of a 3D solid from the three-view (multi-view) drawings in general has been claimed to be a research issue in the CAD area. The main purpose of solving such a problem is to facilitate the input process of a solid modeller. Idesawa (1973) was the pioneer to investigate this topic, and his algorithm was developed to generate polyhedral solids in BREP. Along the BREP track, much research work was subsequently proposed to either enhance the robustness of the reconstruction process or extend the applicable workpiece domain (Lafue 1976, 1978, Markowsky and Wesley 1980, Wesley and Markowsky 1981, Preiss 1981, 1984, Haralick *et al.* 1982, Sakurai and Gossard 1983, Yoshiura *et al.* 1984, Richards and Onwubola 1986, Iwata *et al.* 1988, Gujar *et al.* 1989, Muller and Richter 1990). Several studies proposed algorithms for constructing CSG solids (Aldefeld 1983, Aldefeld and Richter 1984, Ho 1986, Chen and Perng 1988, Chen *et al.* 1992).

In the field of artificial intelligence and computer vision, the work of reconstructing a 3D polyhedral solid from a single 2D line drawing can be traced back to Guzman's work (1968). Later, Huffman (1971) and Clowes (1971) independently developed a famous labelling scheme for junctions which provides the necessary conditions for physically realizing a 2D line drawing. Sugihara (1986) developed a linear algebraic approach which first provides the necessary and sufficient conditions for the physical realization of a 2D line drawing. In his work, a polyhedra object in BREP can be efficiently constructed with the aid of the Huffman–Clowes labelling scheme. With the same labelling scheme, Wang and Grinstein (1989) and Wang (1992) developed methods for the construction of polyhedra in CSG from a single 2D projection.

Note that in the single-view approach, most work assumes that the 3D object to be reconstructed is viewed from a *general* position; that is, any two distinct edges should not overlap on their projections. The assumption of a general position is to eliminate degenerate cases. Further, it implies that the solid to be reconstructed is an existing one. However, in designing a new workpiece using the isometric view, engineers would find that two distinct edges may quite easily have an overlap on their projections, if a particular dimension value is desirable (Figure 1). Surely, a small rotation on the workpiece would eliminate the edge overlapping problem. However, in the design stage, where the 2D line drawing is created by designers instead of being obtained from the projection of an existing 3D solid, such a rotation is impossible. Therefore, the problem of overlapping edges on a 2D line drawing should not be ignored in the sense of providing a user-friendly input scheme for solid modellers.

This paper presents a new algorithm for constructing a CSG rectilinear polyhedron from a single-view

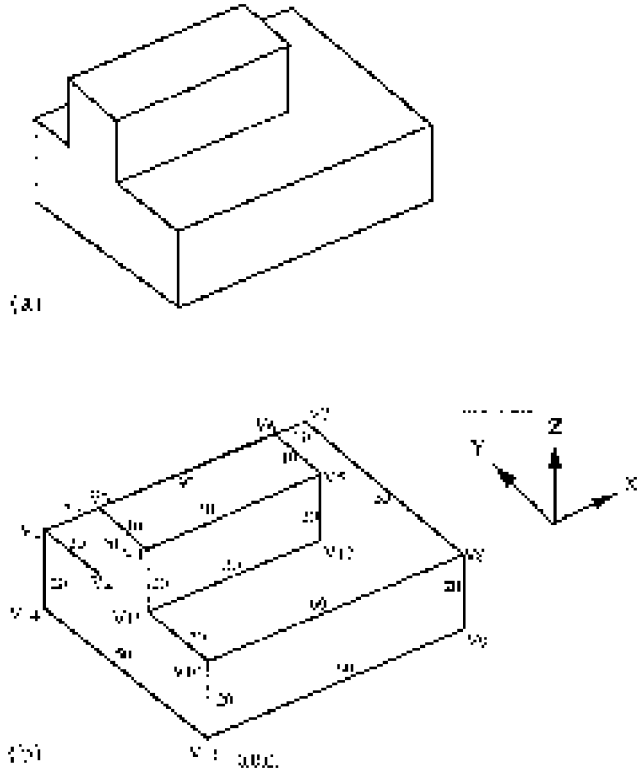


Figure 1. Degenerate cases which show that the projections of two distinct edges may overlap.

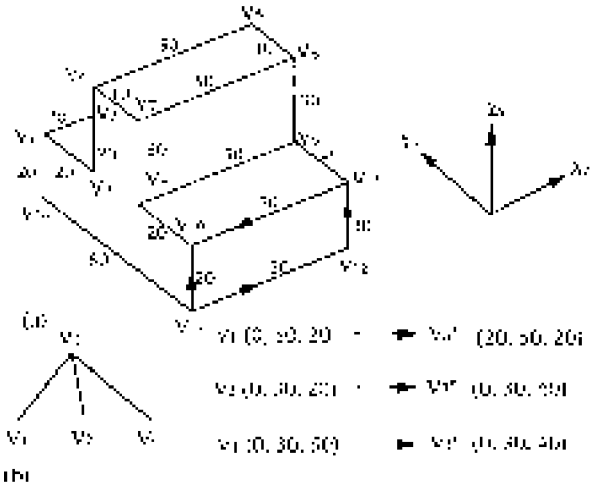


Figure 2. Relationship between 2D and 3D coordinates of a vertex.

isometric drawing. Compared to previous relevant work, this algorithm can deal with the degenerate cases which have been ignored in the single-view approach, and the constructing primitives for representing a rectilinear CSG polyhedron is not limited to rectangular blocks (Wang and Grinstein 1989).

3. Notation and characteristics of isometric drawings

Isometric projection is a type of orthographic projection in which only one projection plane is used, and the projected object has been turned so that its three faces, mutually perpendicular in the 3D world, can be shown on the 2D projection to obtain a pictorial effect (French *et al.* 1986). The isometric projections of the X, Y, and Z coordinate axes are called *isometric axes*, with their angles being 30°, 90°, and 150°, respectively, with respect to the horizontal line. Each edge on an isometric projection is foreshortened equally to about 82% of its original length. Therefore, the term *isometric drawing* has been defined to be one which keeps 100% true length of edges by scaling up the isometric projection (French *et al.* 1986). In this research, the input is an isometric drawing represented in IGES format (Zeid 1991).

3.1. *Notation*

Definition: A *directed edge* denoted by $\overrightarrow{V_iV_j}$ is a finite length vector on an isometric drawing, where V_i is the starting vertex and V_j is the ending one. Alternatively, $\overrightarrow{V_iV_j}$ denotes a ray or a semi-infinite line.

Definition: A *closed loop* comprises a set of sequentially connected edges, with the last edge connecting to the first one so as to form an enclosed region, which is represented by $L(V_1, V_2, \ldots, V_n)$ where V_i is a vertex on the loop. Alternatively, if the last edge does not connect to the first one, the set of edges is known as an *open loop*, represented by $\hat{L}(V_1, V_2, \ldots, V_n)$.

Definition: A closed loop is called a *simple loop* if its enclosed region contains no edge, such as loop $L(V_{13}, V_{12}, V_{11}, V_{10})$ in Figure 2(a); otherwise it is known as a *non-simple loop*, e.g. loop $L(V_{13}, V_{12}, V_{11}, V_9, V_8, V_{10})$ in the same figure.

Definition: An edge is known as an *exterior edge* if it serves as a part of the boundary separating the free space and an isometric drawing such as edge $V_{13}V_{12}$ in Figure 2(a); otherwise it is known as an *interior edge*, e.g. $V_{11}V_{10}$ in the same figure.

3.2. *Characteristics of simple loops*

For a simple loop on an isometric drawing, there are two noteworthy characteristics. First, if a simple loop contains edges with three types of slopes, its bounding region cannot be a complete face in the 3D world. As stated, the workpieces are rectilinear ones, each edge therefore should be parallel to one axis of the 3D coordinate system. This implies that there exist only three types of edges on a workpiece and only two types of edges on a face. The isometric projection of a face, if not hidden at all, would contain only two types of edges. Should it contain three types of edges, such as loop $L(V_1, V_2, V_3)$ in Figure 2(a), its bounding region would not be a complete face in the 3D world.

Second, on an isometric drawing, an exterior edge belongs to only one simple loop and an interior edge belongs to two simple loops. As shown in Figure 2(a), exterior edge $V_{13}V_{12}$ belongs only to loop $L(V_{13}, V_{12}, V_{11}, V_{10})$, and interior edge $V_{10}V_{11}$ belongs to two simple loops, $L(V_{13}, V_{12}, V_{11}, V_{10})$ and $L(V_{10}, V_{11}, V_9, V_8)$.

The rationale can be explained below. An edge on a 3D workpiece has only two neighbouring faces; its isometric projection should have two neighbouring faces or have only one if the other is completely hidden. An exterior edge can have only one neighbouring face visible; if its two neighbouring faces are both visible, this edge cannot be a boundary of the isometric drawing. Therefore, an exterior edge belongs to only one simple loop. Alternatively, on an isometric drawing, the whole region bounded by exterior edges is separated into several subregions. An interior edge is the boundary of two subregions and therefore belongs to two simple loops.

3.3. *Hidden effects on isometric drawings*

The isometric projection of a 3D workpiece may result in some hidden phenomenon; that is, the projection of a face or an edge is partially or wholly hidden by that

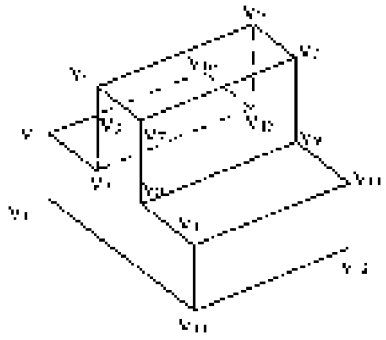


Figure 3. Original form of a partially visible face.

of some other geometric entities. Some terms relevant to the hidden effects are defined below: *overlaid edges*, *partially visible edges*, *partially visible faces*, *completely visible faces* and *almost invisible faces*.

An *overlaid edge* is a 2D line segment which denotes a portion commonly shared by the projections of two distinct edges. As shown in Figure 1(b), line segment V_3V_6 is an overlaid edge which is commonly shared by two distinct projections (V_3V_6 and V_1V_7).

A *partially visible edge* is the portion of an edge projection which has been partially hidden on the isometric drawing. For example, edge V_1V_3 (Figure 2(a)) is a partially visible edge if we consider V_1V_{16} (Figure 3) as its original form in the 3D world.

A *partially visible face* is a bounded region on an isometric drawing, whose inverse projection is not a complete face in the 3D world (Figure 2(a)) the region bounded by loop $L(V_3, V_1, V_2)$ is a partially visible face; its original form in 3D world may be a face bounded by a loop $L(V_1, V_2, V_{15}, V_{16})$ as shown in Figure 3. In the 3D world edge V_2V_3 is not a part of this face; the other two edges V_1V_3 and V_1V_2 are really on the face and form an open loop $\hat{L}(V_1, V_2, V_3)$ which is known as the corresponding *partially visible loop* of the face.

A *completely visible face* is a bounded region on an isometric drawing, whose inverse projection is a complete face in the 3D world; that is, the face projection is not hidden at all. An example is the face bounded by loop $L(V_{13}, V_{12}, V_{11}, V_{10})$ in Figure 2(a). The corresponding loop of such a face is known as a *completely visible loop*.

An *almost invisible face* is one which is wholly hidden except some of its bounding edges when the isometric projection is applied. As shown in Figure 2(a) the connection of edges V_2V_4 and V_4V_5 denotes that a face containing these two edges, like the one bounded by loop $L(V_2, V_4, V_5, V_{15})$ in Figure 3, should exist in the 3D world, even though it is almost invisible in the 2D world. To characterize this type of face, its corresponding open loop, such as $\hat{L}(V_2, V_4, V_5)$ is known as an *almost invisible loop*.

The visibility of faces can be determined by a simple rule. That is, a workpiece face should have its outward normal pointing in the $-X$, $-Y$, and $+Z$ axis directions if its isometric projection is to be completely or partially visible (Figure 1(b)). Conversely, a workpiece face with its outward normal pointing in the opposite directions (i.e. $+X$, $+Y$, $-Z$ directions) will be completely invisible or almost invisible. That is, the outward normal of a face bounded by either a partially visible loop, a completely visible loop, or an almost invisible loop can be determined from the two types of its comprising edges.

In the following discussion, completely visible and partially visible faces are both termed as *visible faces*, because at least part of their isometric projections are visible. For ease of presentation, faces and their corresponding loops are sometimes taken as synonyms.

4. System framework of the algorithm

The system framework of the proposed algorithm is shown in Figure 4. The input to the system is the isometric drawing of a workpiece, a set of 2D line segments or edges (Figure 5(a)) represented in IGES format where the 2D coordinate of each vertex and the neighbouring relationships between vertices and edges can be explicitly determined from the input file.

4.1. *Two major modules*

The algorithm is composed of two major modules. The first one is designed to find all the *visible faces* and to determine the 3D coordinates of vertices on the

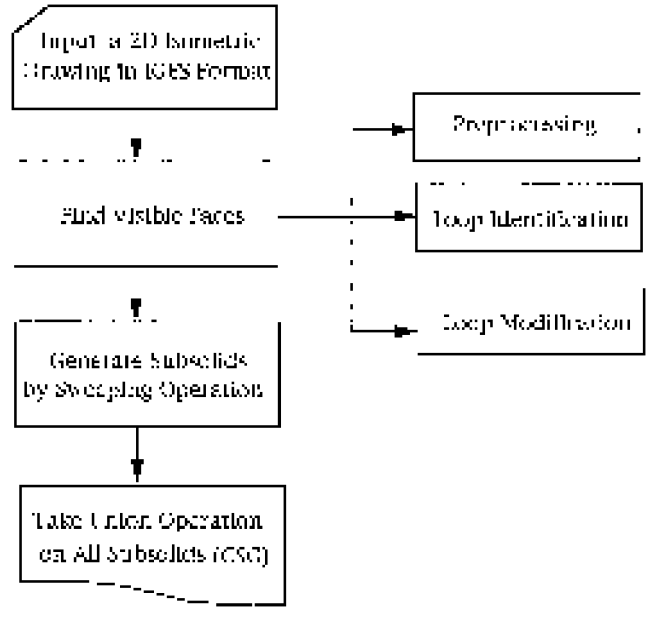


Figure 4. System framework of the proposed algorithm.

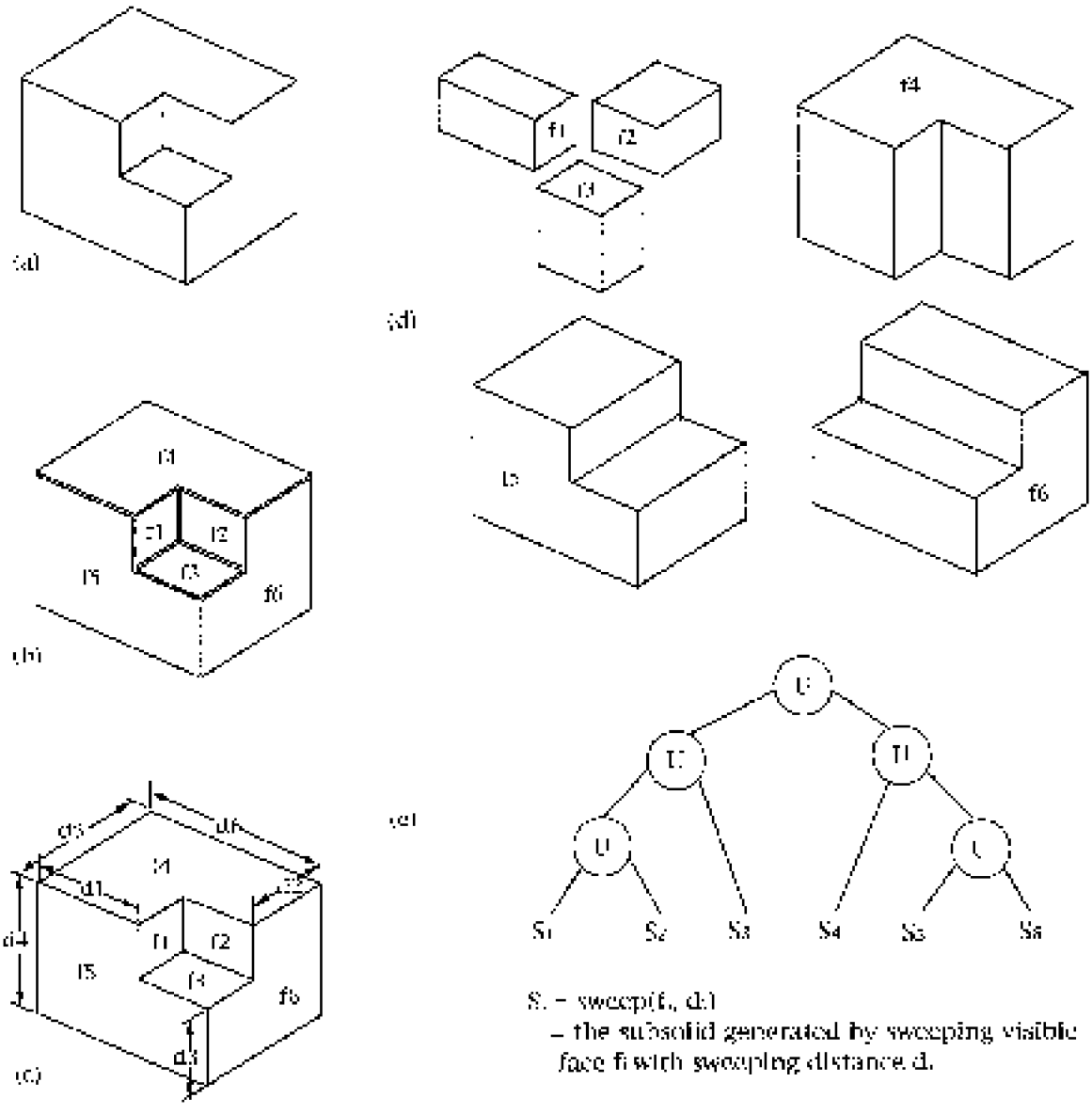


Figure 5. Examples for illustrating procedures of the proposed algorithm.

isometric drawing. In the illustrated example, visible faces involve faces $f1$, $f2$, $f3$, $f4$, $f5$ and $f6$ (Figure 5(b)).

With the 3D coordinates of vertices having been determined, the second module aims to find a 3D *sweeping distance* for each visible face, and generates a solid by moving the visible face along its inward normal with the sweeping distance. The generated solid is known as a *subsolid* because it is a subset of the original workpiece (Figure 5(c)). For visible faces $f1, f2, \ldots$ and $f6$, their corresponding sweeping distances are $d1$, $d2, \ldots$ and $d6$, respectively, and the generated subsolids are as shown in Figure 5(d). Finally, the Boolean union operation on all the subsolids found is the resulting workpiece solid model in CSG (Figure 5(e)).

4.2. *Three submodules for finding visible faces*

The module for finding visible faces is relatively complicated and requires further explanation of its three submodules (Figure 4).

The first one is known as the *preprocessing submodule* which is designed for three purposes: (1) determining the 3D coordinate of each vertex on the isometric drawing; (2) identifying partially visible and overlaid edges; and (3) modifying the data structure of the isometric drawing whenever overlaid edges are detected. That is, the final data structure of the isometric drawing should be able to represent an overlaid edge as two distinct edges to reflect its 3D topological relationship.

The second submodule is known as the *loop identification submodule* which is designed for identifying three types of loops: (1) completely visible loops; (2) partially visible loops; and (3) almost invisible loops. The first two types are identified for the generation of their corresponding subsolids. Almost invisible loops are recognized to aid the reconstruction of partially visible loops.

The third submodule is known as the *loop modification submodule* which aims to modify all partially visible and some almost invisible loops into their expected original forms without being hidden, which are closed loops and will be used to generate subsolids.

5. Preprocessing submodule

In the preprocessing submodule, the determination of 3D coordinates of vertices will be presented as below. We first discuss the basic relationship between 2D and 3D coordinates of a vertex, then depict the characteristics of those vertices which have multiple inverse projections, and illustrate the inference method for determining their 3D coordinates. The derived 3D coordinates are finally used in the identification of overlaid and partially visible edges.

5.1. *Relationship between 2D and 3D coordinates of a vertex*

As stated, each edge on the isometric drawing must be parallel to one isometric axis, and the 2D length of an edge should be exactly equal to its 3D length. These two properties are used in determining the 3D coordinates of vertices on an isometric drawing.

In Figure 2, the directed edge $\overrightarrow{v_{13}v_{12}}$ on the isometric drawing is parallel to the X isometric axis (X_s), with its 2D length equal to $+50$ units along the $+X_s$ axis (alternatively, the 2D length of directed edge $\overrightarrow{v_{12}v_{13}}$ is -50 units along the $+X_s$ axis). Suppose vertex V_{13} is the origin of the 3D workpiece coordinate system, we can infer that the 3D coordinate of vertex V_{12} is $(50, 0, 0)$. Likewise, the length of directed edge $\overrightarrow{v_{12}v_{11}}$ is $+20$ units along the $+Z_s$ axis; the length of edge $\overrightarrow{v_{11}v_{10}}$ is -50 units along the $+X_s$ axis. The 3D coordinates of vertex V_{11} and V_{10} can be determined to be $(50, 0, 20)$, and $(0, 0, 20)$, respectively.

5.2. *Multiple inverse projections and collinear vertices*

Notice that the 3D coordinates of some vertices may be inferred to be identical even by travelling through different paths. For example, vertex V_{10} in Figure 2(a) can be directly inferred from vertex V_{13}, or indirectly by traveling through the path ($V_{13} \rightarrow V_{12} \rightarrow V_{11} \rightarrow V_{10}$). By either path, the two inferred coordinates for vertex V_{10} are the same, both being $(0, 0, 20)$.

However, the above situation may not be valid for some vertices like V_3 in this figure. The 3D coordinate of vertex V_3 can be inferred by its three neighbouring vertices V_1, V_2 and V_4. Note that the 3D coordinate of vertex V_3 can be determined to be $(20, 50, 20)$ when inferred from vertex V_1; and can be $(0, 30, 40)$ when either inferred from vertices V_2 or V_4 (Figure 2(b)).

Two different 3D coordinates being inferred for vertex V_3 indicates that vertex V_3 on the isometric view is the projection of two different points in the 3D world. Its real situation may be as shown in Figure 3, where edges V_1V_{16} and V_2V_4 have no intersection in the 3D world while their projections intersect at vertex V_3 on the isometric drawing. With respect to edge V_2V_4 in the 3D world, vertex V_3 in fact is an in-between point rather than an end point of the edge.

To characterize the type of vertices like V_3, a vertex on the isometric drawing is known as a *collinear* vertex if it is collinear with two other vertices and stays in-between on their connecting edge. Conversely, a vertex other than a collinear one is known as a *non-collinear vertex*. Considering the isometric drawing in Figure 2(a), only vertex V_3 is a collinear one, the other vertices are all non-collinear.

The distinction between collinear and non-collinear vertices is very important. As shall be proved in Appendix 1, a collinear vertex may have one or multiple inverse projections; while a non-collinear vertex has only one. This property implies that, for a non-collinear vertex, one of its neighbouring vertices should have a single inverse projection to ensure that a unique 3D coordinate for the non-collinear vertex can be obtained. This further means that a travelling path consisting of vertices with single inverse projection can always be found between the origin and a non-collinear vertex. Therefore, 3D coordinates of all non-collinear vertices can be uniquely determined by applying the inference method stated above.

5.3. *Inferring 3D coordinates for collinear vertices*

For a collinear vertex, the inference of all its possible 3D coordinates requires the construction of a *coordinate reference tree* which is created according to the following rules. First, create a tree with the concerned collinear vertex being assigned to the root node. Second, for a vertex already assigned to the tree, each of its neighbouring vertices, which has not been assigned, should be given as its children nodes on the tree. The second rule has to be repeatedly performed until each terminal node of the tree has been assigned a non-collinear vertex.

Taking the workpiece in Figure 1(b) as an example, the coordinate reference tree for vertex V_3, a collinear vertex, can be created as shown in Figure 6(a). Vertex V_3 is the root node at level 1 and its four neighbouring vertices V_1, V_2, V_4, and V_6 are assigned as its children nodes, at level 2. Since vertex V_6 is also a collinear vertex,

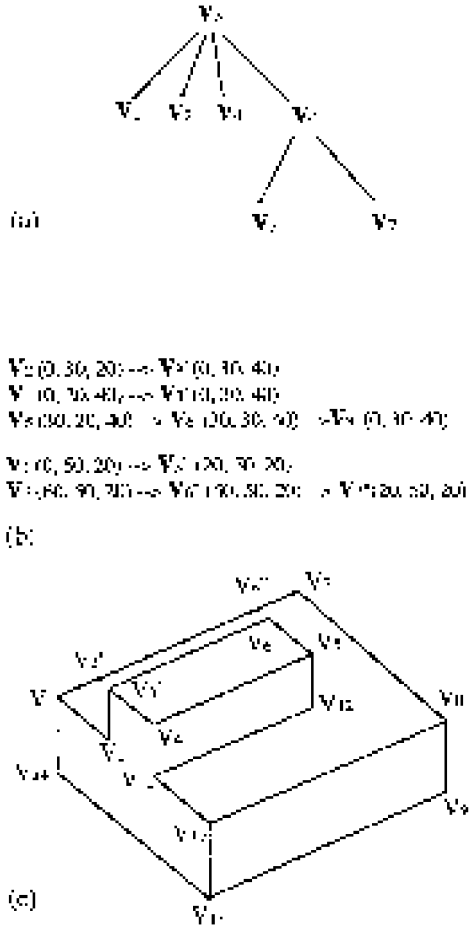


Figure 6. Coordinate reference tree and the modification of input data structure for detecting overlaid edges.

its two neighbouring vertices V_5 and V_7 should be given as its children nodes, at level 3. The coordinate reference tree of vertex V_3 now has been created because each terminal node has been assigned a non-collinear vertex. From Figure 6(b) we can see that there are two solutions obtained for the 3D coordinates of vertex V_3. Starting from non-collinear vertices V_2, V_4 and V_5 to infer the 3D coordinates of vertex V_3, we can get the first solution, $V_3' = (0, 30, 40)$. The second solution (V_3'') can be derived to be $(20, 50, 20)$ whenever either vertex V_1 or V_7 is chosen as the starting point of the inference path. Notice that vertex V_6, a collinear one, can also be determined to have two solutions, $V_6 = (30, 30, 40)$ and $V_6'' = (50, 50, 20)$.

Two 3D coordinates being inferred indicates that vertex V_3 has two inverse projections, V_3' and V_3''. Referring to Figure 6(a) and 6(b), the vertices at level 2 of the coordinate reference tree can be classified into two groups, $G1 = \{V_2, V_4, V_6\}$ and $G2 = \{V_1, V_6''\}$, where in the 3D world each one in group $G1$ is a neighbour of vertex V_3' and each one in group $G2$ is a neighbour of vertex V_3'. An edge connecting vertex V_3' and any vertex in group $G1$ is called a *neighbouring edge* of vertex V_3', likewise for any member in group $G2$ and vertex V_3''.

5.4. *Detecting partially visible and overlaid edges*

For a collinear vertex, the number of its 3D neighbouring vertices in a particular group helps determine the visibility of its neighbouring edges. As shall be proved in Appendix 2, if a collinear vertex has only one neighbouring vertex in a group, then the unique neighbouring edge in this group is a partially visible edge; if it has two or more neighbouring vertices, then the neighbouring edges in this group are all completely visible edges. These two characteristics can also be used to detect overlaid edges.

An example of detecting a partially visible edge is shown in Figure 2(b) where vertex V_3 has two inverse projections, V_3' and V_3''. Vertex V_3' has only one neighbouring vertex, therefore edge V_3V_1 on the isometric drawing is a partially visible edge. Vertex V_3'' has two neighbouring vertices, therefore edges V_3V_4 and V_3V_2 are completely visible edges; these two edges are collinear and can be combined into one, edge V_2V_4.

An example of detecting the overlaid edge in Figure 1(b) is shown in Figure 6. For collinear vertex V_3, its neighbouring vertices can be classified into two groups, $G1 = \{V_2, V_4, V_6'\}$ and $G2 = \{V_1, V_6''\}$, the number of neighbouring vertices in each group is more than one and therefore its neighbouring edges are all completely visible. Again for collinear vertex V_6, its neighbouring vertices can also be classified into two groups, $F1 = \{V_5, V_3'\}$ for vertex V_6' and $F2 = \{V_7, V_3''\}$ for vertex V_6; the neighbouring edges in each group likewise can all be inferred to be completely visible. That is, edges $V_3''V_6''$ in group $G2$ and edge $V_3'V_6'$ in group $G1$ are both completely visible. Since these two edges do not have intersection in the 3D world, but have the same isometric projection, edge V_3V_6 on the isometric drawing can therefore be recognized to be an overlaid edge.

5.5. *Modification of data structure for overlaid edges*

An overlaid edge denotes two distinct edges in the 3D world; its representation on the isometric drawing should be modified by splitting this edge into two in order to facilitate the finding of visible faces.

For example, the overlaid edge V_3V_6 in Figure 1(b) has two inverse projections ($V_3'V_6'$ and $V_3''V_6''$) in 3D world (Figure 6(b)); its neighbouring edges can be classified into two groups, $M1 = \{V_2V_3', V_4V_3', V_5V_6'\}$ and $M2 = \{V_1V_3'', V_7V_6''\}$ where each member of $M1$ is a neighbouring edge of $V_3'V_6'$ and each member of $M2$ is a neighbouring edge of $V_3''V_6''$. The original data structure should be modified so that vertices V_3 and V_4 are split into four vertices V_3', V_3'', V_4', and V_4'' while preserving

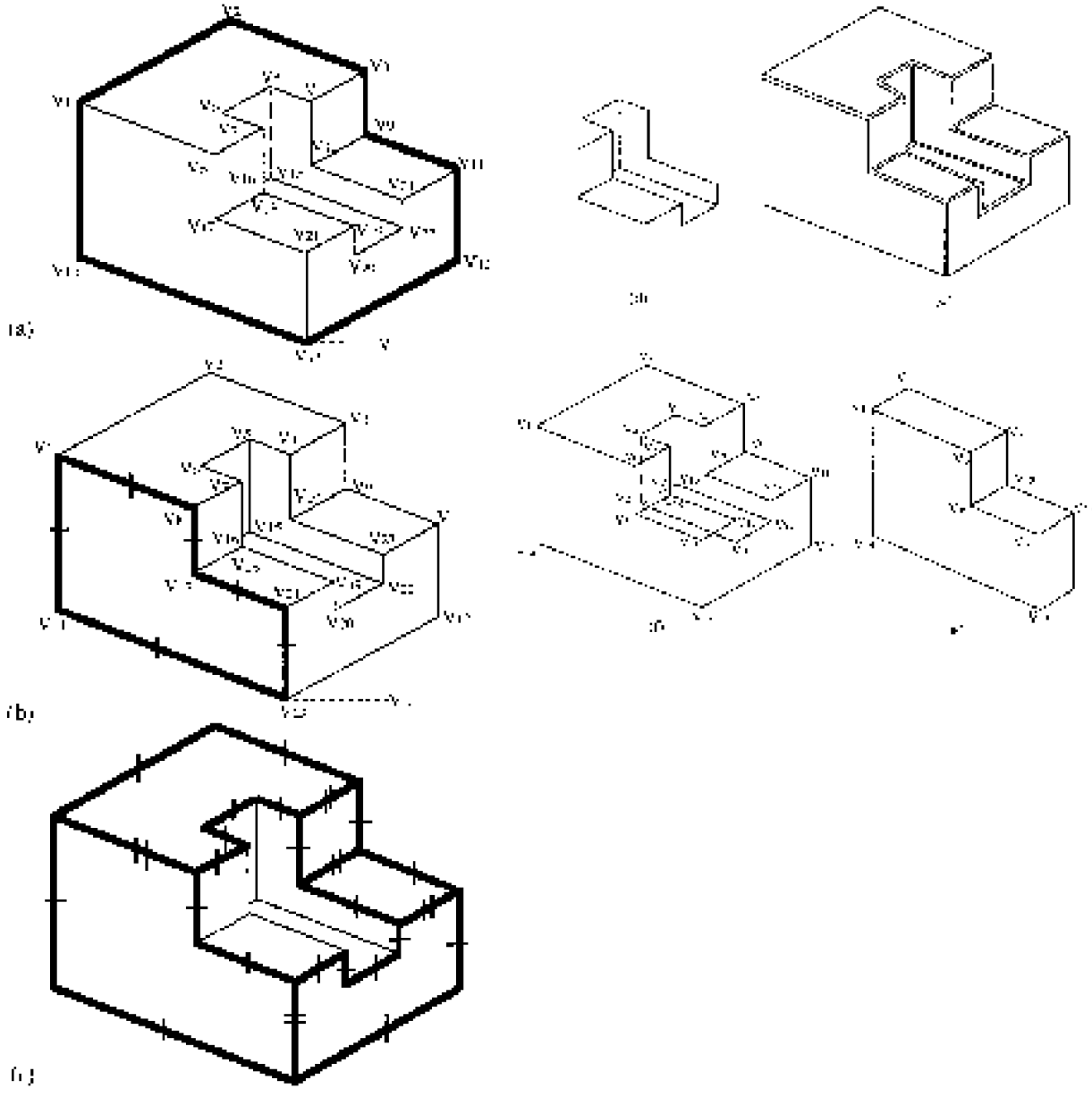


Figure 7. Examples of finding simple loops and the generation of subsolids.

their 2D coordinates and clarifying their new connecting relationships. The data structure after modification would denote an isometric drawing as shown in Figure 6(c) where the overlaid edge has been split into two.

6. Loop identification module

The loop identification module is designed to find three types of loops: completely visible, partially visible, and almost invisible ones. A completely visible and a partially visible face are both enclosed by a simple loop and therefore an algorithm for identifying simple loops is given first.

6.1. *Identifying simple loops*

As stated, an exterior edge belongs to only one simple loop and an interior edge belongs to two simple loops. With these two characteristics, simple loops can be detected by creating some travelling paths on the isometric drawing. These travelling paths are established according to two principles—'turn-to-the-leftmost' and 'turn-to-the-rightmost'.

The principle of 'turn-to-the-leftmost' is used to identify all the exterior edges on an isometric drawing. As shown in Figure 7(a), vertex V_{13} which has the lowest y coordinate value is chosen as the starting point for travelling the isometric drawing, with a virtual directed

edge ($\overline{V_L V_{13}}$) given as the starting travelling edge where V_L is a virtual vertex to the right of V_{13}. With respect to edge $V_L V_{13}$, there are three neighbouring directed edges ($V_{13}V_{12}$, $V_{13}V_{21}$, $V_{13}V_{14}$) from which to choose the next to be visited. Among these three, $V_{13}V_{14}$ is the leftmost one (with the smallest intersection angle with respect to edge $V_L V_{13}$) and is selected. Then with respect to directed edge $V_{13}V_{14}$, there exists only one neighbour, $V_{14}V_1$, which is chosen as the next to be visited. Repeatedly performing the principle of 'turn-to-the-leftmost', a closed loop consisting of all exterior edges, $L(V_{13}, V_{14}, V_1, V_2, V_3, V_9, V_{11}, V_{12})$, can be identified.

The principle of 'turn-to-the-rightmost' is used to identify simple loops on an isometric drawing. By starting from an exterior edge and creating a travelling path in the way of turn-to-the-rightmost, a simple loop can be detected. Taking directed edge $V_{13}V_{14}$ in Figure 7(b) as the starting one, the travelling path created in the way of 'turn-to-the-rightmost' is a simple loop, $L(V_{13}, V_{14}, V_1, V_8, V_{17}, V_{21})$. Notice that directed edges, $V_{14}V_1$, V_1V_8, V_8V_{17}, $V_{17}V_{21}$, and $V_{21}V_{13}$ are chosen because each of them is either a unique one or the rightmost one (with the largest intersection angle with respect to its preceding directed edge).

When a simple loop is detected, each edge of the loop should be marked once to denote that it already belongs to a particular sample loop (Figure 7(b)). The 'turn-to-the-rightmost' principle should be repeatedly performed until each exterior edge has been marked once. As shown in Figure 7(c), we can see that five simple loops have been identified and each one is composed of at least one exterior edge. Notice that some interior edges in this figure are only marked once or not at all; this indicates that their simple loops have not been exhaustively identified (an interior edge belongs to two simple loops).

To characterize these unidentified simple loops, a procedure for removing edges and resetting edge markings on the isometric drawing should be taken. That is, all exterior edges as well as those interior edges which have been marked twice should be removed from the isometric drawing (Figure 7(c)), because their belonging to simple loops has been exhaustively recognized. After that, the marking of the residual edges are reset to zero (Figure 7(d)).

On the residual drawing, repeating the procedures of 'turn-to-the-leftmost' and 'turn-to-the-rightmost', we can identify four more simple loops from Figure 7(d), and the final result can further be identified as a simple loop. The isometric drawing in Figure 7(a) can be recognized to have ten simple loops (Figure 7(e)).

6.2. *Deleting covering edges from partially visible loops*

Some edges on a partially visible face are known as *covering edges* because they do not exist on the face in the 3D world and would cover the face when isometric projection is applied. Examples of covering edges are like edges V_6V_7, V_7V_8, V_8V_9, and V_9V_{10} with respect to loop $L(V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8, V_9, V_{10})$ in Figure 8. Covering edges on a partially visible face should be identified and deleted in order to reconstruct the original form of the face in the 3D world.

A covering edge on a partially visible face can be identified by using the following two characteristics. First, a covering edge intersects a partially visible edge at a collinear vertex, such as edges V_7V_{16} and $V_{18}V_{16}$ in Figure 7 where edge $V_{15}V_{16}$ is a partially visible one. The rationale of this characteristic will be explained in Appendix 2. Second, a covering edge in the 3D world cannot be on the plane passing through the partially visible face.

The applications of the second characteristic are explained below. Considering loop $L(V_5, V_6, V_7, V_{16}, V_{15})$ in Figure 7(a), a set of three mutually connecting edges with edge V_6V_7 as a member, would involve three sets: $(V_{15}V_5, V_5V_6, V_6V_7)$, $(V_5V_6, V_6V_7, V_7V_{16})$, $(V_6V_7, V_7V_{16}, V_{16}V_{15})$. Each of these sets has three types of slopes and cannot uniquely determine a planar face. Therefore, V_6V_7 cannot be an edge on the partially visible face in the 3D world, and is recognized as a covering edge. Similarly, edge V_7V_{16} can also be identified as a covering edge. The closed loop $L(V_{16}, V_{15}, V_5, V_6, V_7)$ in Figure 7(a) can finally be modified into an open loop $\hat{L}(V_{16}, V_{15}, V_5, V_6)$ which can uniquely determine a plane.

Intuitively, a covering edge in case of being an interior edge should belong to two simple loops in the 2D world. However, one of the neighbouring face (the partially visible one) does not contain the covering edge in the 3D world. Therefore, a covering edge should be seen as belonging to only one simple loop instead of two. For example, edge V_6V_7 belongs to simple loop $L(V_6, V_7, V_8, V_1, V_2, V_3, V_4, V_5)$ only and does not belong to simple loop $L(V_{16}, V_{15}, V_5, V_6, V_7)$. This characteristic is very important for identifying almost invisible loops.

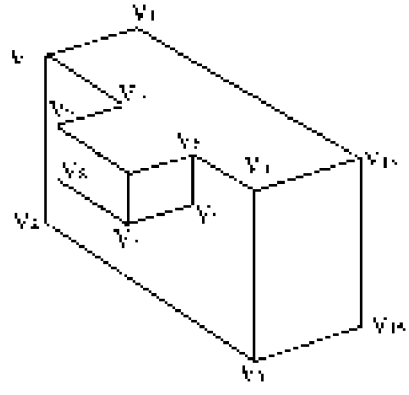


Figure 8. An example of a partially visible face containing no collinear vertex.

6.3. *Identifying almost invisible loops*

The identification of almost invisible faces consists of two steps. First, for each external and covering edge, determine the normal for the particular neighbouring face which has been hidden. For example, in Figure 2(a), edge V_4V_5 belongs to only one simple loop $L(V_4, V_5, V_6, V_7)$. Since edge V_4V_5 is parallel to the X_s axis, the normals of its two neighbouring faces should be parallel to either the Y_s or the Z_s axis. With the face normal of loop $L(V_4, V_5, V_6, V_7)$ being parallel to the Z_s axis, we can infer that the other neighbouring face, though invisible on the isometric drawing, should have its normal parallel to the Y_s axis.

Second, classify all external and covering edges into groups by the following criterion. In a group, the comprising edges should be mutually connected, and the normals of their hidden neighbouring faces should be parallel to each other. That is, these edges should be on a plane and their connection would form an open loop. As shown in Figure 2(a), edges V_5V_6, V_6V_9, V_9V_{11}, and $V_{11}V_{12}$ would be classified into one group and their consecutive connection would form an open loop $\hat{L}(V_5, V_6, V_9, V_{11}, V_{12})$. The two ending edges of an open loop are called *open edges* (i.e. V_5V_6 and $V_{11}V_{12}$) and the two ending vertices are known as *open vertices* (i.e. vertices V_5 and V_{12}).

Notice that an almost invisible loop can be composed of only one edge. An example is the almost invisible loop $\hat{L}(V_5, V_{11}, V_6)$ shown in Figure 9(a), which can be seen as an edge V_5V_6. This edge comprises two line segments, one (V_6V_{11}) can be identified as an external edge and the other (V_5V_{11}) as a covering edge. This indicates that the edge should belong to an almost invisible loop. However, neither of its two neighbouring edges can reside on its almost invisible face by observing their normals; therefore the edge itself is an almost invisible loop.

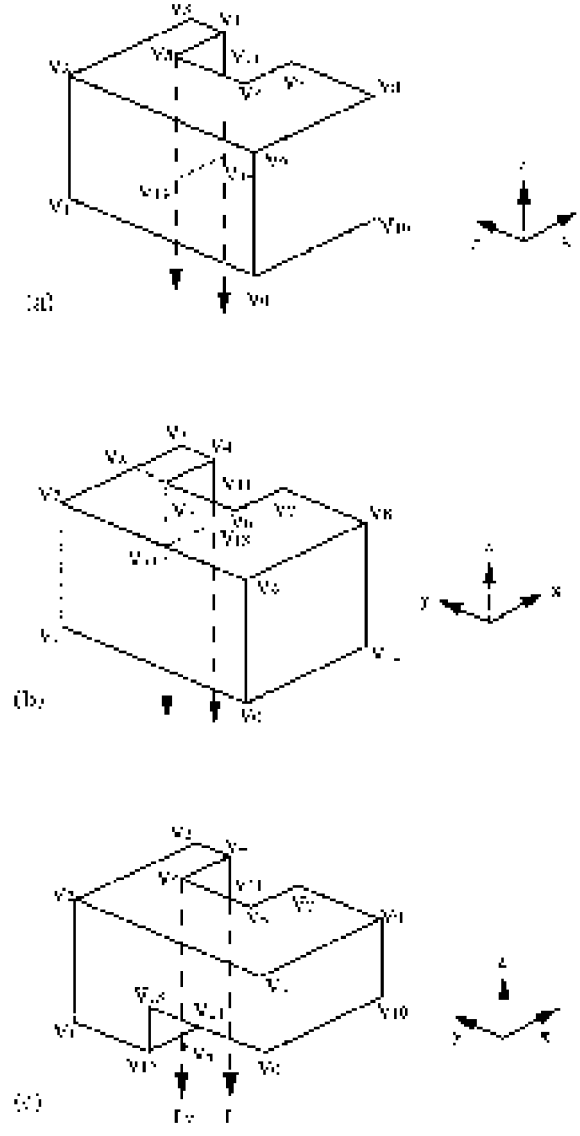


Figure 9. (a) constraint plane is defined for determining new edges which are used to reconstruct the original form of a partially visible face; (b) one-to-many mapping; (c) the closest constraint plane should be chosen.

7. Loop modification module

Partially visible loops detected from an isometric drawing are all open loops which require to be modified into their original form, closed loops, in order to generate their corresponding subsolids. We first discuss the locating constraints of new edges to be added on to a partially visible loop. Then, three methods for solving the loop modification problem are given.

7.1. *Locating constraints of newly added edges*

A partially visible loop has two open edges, to which new edges should be added in order to form a closed loop. For the purpose of obtaining a valid solid, new edges should be created so that the partially visible loop has its planar and topological characteristics maintained. That is, a new edge should be connected to an open vertex and stay on the plane where the partially visible loop resides. This implies that the slope of a new edge should be either one of the two types which are allowed to exist on the partially visible loop. To describe the locating constraints of new edges, we define the semi-infinite line which starts at an open vertex and passes through a new edge to be their *extended ray*, which is intended for depicting the starting vertex and the slope of a new edge. For example (Figure 3) $\overrightarrow{V_2V_{15}}$ is an extended ray if V_2V_{15} is considered as a newly added edge for open loop $\hat{L}(V_2, V_1, V_3)$.

Since an open vertex may either be a collinear or a non-collinear one, there are two ways in determining

the extended ray of a new edge. First, if an open vertex is a collinear one, the extended ray should be collinear with the open edge. The reason is that the open edge connecting a collinear vertex has to be a partially visible one of which some parts have been hidden. The new edge therefore has to be collinear with the open edge in order to reconstruct its original form back. As shown in Figure 7(f), the open vertex V_{16}, a collinear one on the partially visible loop $\hat{L}(V_{16}, V_{15}, V_{22}, V_{20})$, should have its extended ray $\overrightarrow{V_{16}V_{24}}$ being collinear with the open edge $V_{16}V_{15}$.

Second, if an open vertex is a non-collinear one, it should have three neighbouring edges with distinct slopes. And at least two of the three edges would have been displayed on the isometric drawing; otherwise the open vertex cannot be visible. The new edge then can be easily determined to have the particular slope which has not been displayed. As shown in Figure 7(f), the open vertex V_{20} is a non-collinear one with two neighbouring edges ($V_{20}V_{22}$ and $V_{20}V_{19}$) already being displayed on the isometric drawing. The extended ray $\overrightarrow{V_{20}V_{24}}$ then has to be perpendicular to both edges, $V_{20}V_{22}$ and $V_{20}V_{19}$.

7.2. *Introduction to three methods for loop modification*

According to the type of information referred to in the loop modification process, there are three methods provided to modify a partially visible loop. The first one, with the highest priority, considers only the information of the *partially visible loop*. If the first method cannot modify the partially visible loop into a closed one, then the second one, which augments the information of its *neighbouring open loops*, is tried. The third method, which refers to the information of some *non-neighbouring open loops*, is applied only when the loop modification problem cannot be solved by the first two methods.

7.3. *First method for loop modification*

For a partially visible loop, the intersection of its two extended rays may be a point, an edge, or null. When there is an intersection, either a point or an edge, the partially visible loop can be modified into a closed one by considering only the information of this partially visible loop.

When the two extended rays intersect at a point, the loop is modified by adding two new edges, either one is on an extended ray and ends at the intersection point. As shown in Figure 7(f), the two extended rays of partially visible loop $\hat{L}_1 = \hat{L}(V_{16}, V_{15}, V_{22}, V_{20})$ has an intersection point, vertex V_{24}, which can be used to determine that the two new edges to be added are $V_{20}V_{24}$ and $V_{16}V_{24}$. Loop $\hat{L}_1$ then can be modified into a closed one, $L(V_{15}, V_{22}, V_{20}, V_{24})$. Notice that, with a new edge ($V_{16}V_{24}$) added, one of its neighbours, partially visible loop $\hat{L}(V_{16}, V_{15}, V_5, V_6)$, now has been converted into a new form, $\hat{L}(V_{24}, V_{15}, V_5, V_6)$.

When the two extended rays are collinear and therefore intersect at an edge, the loop is modified by adding in the edge which has the two open vertices as its ending points. As shown in Figure 7(f), partially visible loop $\hat{L}_2 = \hat{L}(V_{24}, V_{15}, V_5, V_6)$, with the new edge $V_{16}V_{24}$ having been included, has its two extended rays ($\overrightarrow{V_6V_{24}}$ and $\overrightarrow{V_{24}V_6}$) collinear with each other. Partially visible loop $\hat{L}_2$ can be modified into a closed one $L(V_{24}, V_{15}, V_5, V_6)$ by the addition of edge V_6V_{24}, which has open vertices V_6 and V_{24} as its two ending points.

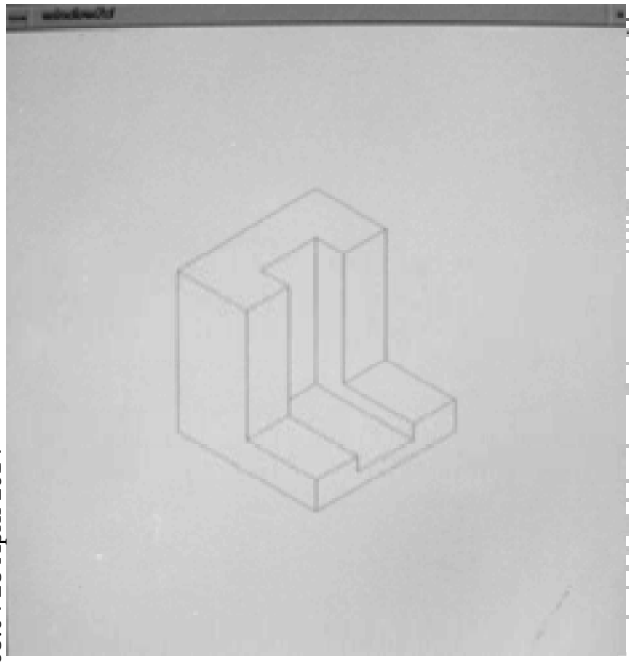
7.4. *Second method for loop modification*

The second method for loop modification is developed for the case when the two extended rays have no intersection. Its main idea is to derive some new information from its neighbouring open loops before modifying the partially visible loop. That is, we first consider the possibility of modifying the neighbouring open loops into closed ones, which are originally either partially visible or almost invisible ones. Then, the newly added edges, being created through processing the neighbouring loops, would appear as a part of the concerned partially visible loop. This in turn may help modify the concerned partially visible loop into a closed one.

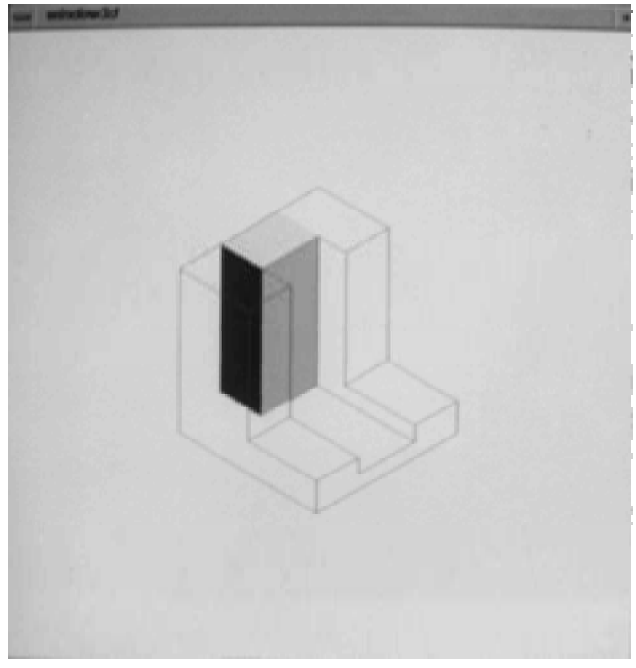
As shown in Figure 3, for partially visible loop $\hat{L}_1 = \hat{L}(V_3, V_1, V_2)$, its two extended rays ($\overrightarrow{V_3V_{16}}$ and $\overrightarrow{V_2V_{15}}$) are parallel and have no intersection; the loop cannot be modified into a closed one if we consider only the information of its two extended rays. Referring to one of its neighbouring open loops, $\hat{L}_2 = \hat{L}(V_2, V_3, V_4, V_5)$, an almost invisible one, the two extended rays of loop $\hat{L}_2$ intersects at vertex V_{15}. Loop $\hat{L}_2$ then can be modified into a closed one if two new edges (V_2V_{15} and V_5V_{15}) are added. With the new edge V_2V_{15} added, the concerned visible loop $\hat{L}_1$ now has the form of $\hat{L}(V_3, V_1, V_2, V_{15})$, which can be modified into a closed one by applying the first method, because its two extended rays now have an intersection at vertex V_{16}.

7.5. *Third method for loop modification*

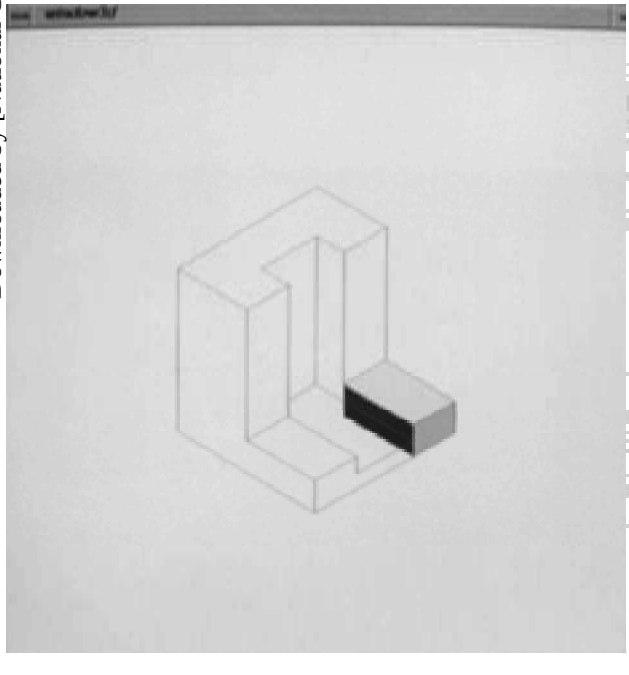
The third method for loop modification is designed for the case where the two extended lines have no intersection and using the second method cannot solve the loop modification problem. As shown in Figure 9(a), partially visible loop $\hat{L}(V_5, V_4, V_{11})$ has two extended rays which have no intersection at all. Of its three neighbouring loops, only loops $\hat{L}(V_{11}, V_4, V_3)$ and $\hat{L}(V_5, V_{11}, V_6)$ are open ones. As we can see in this figure, the two open neighbouring loops cannot be modified into closed ones by considering only their extended rays;



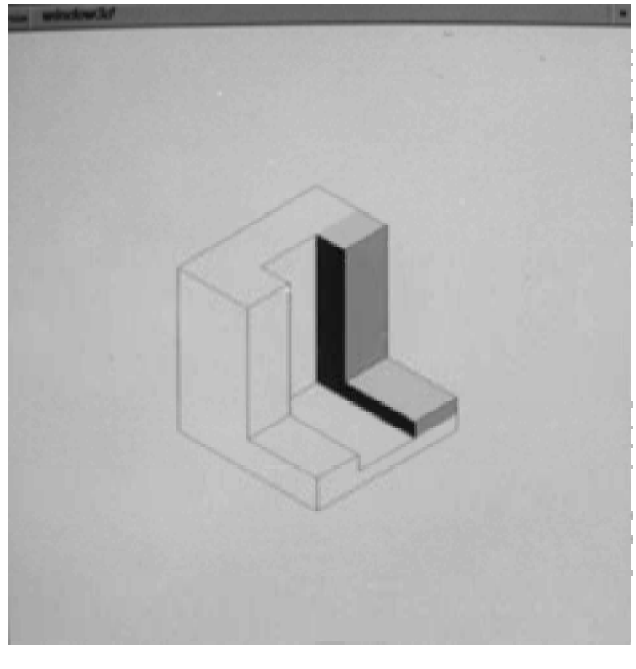
(a)



(b)

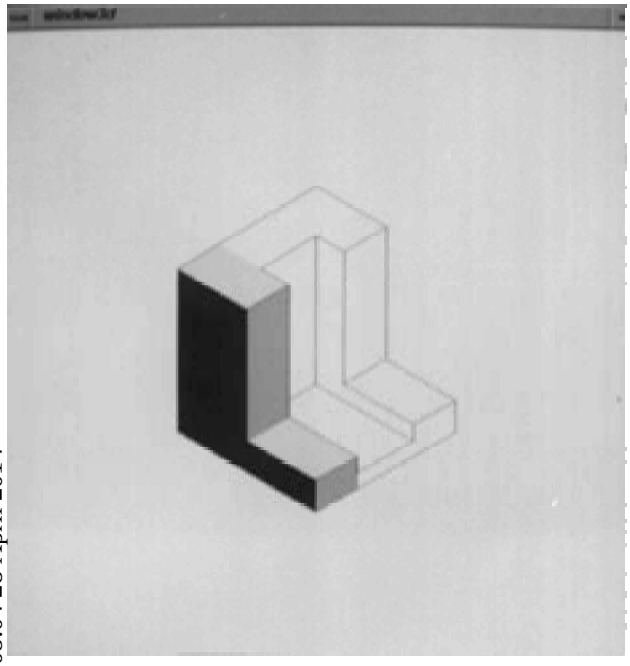


(c)

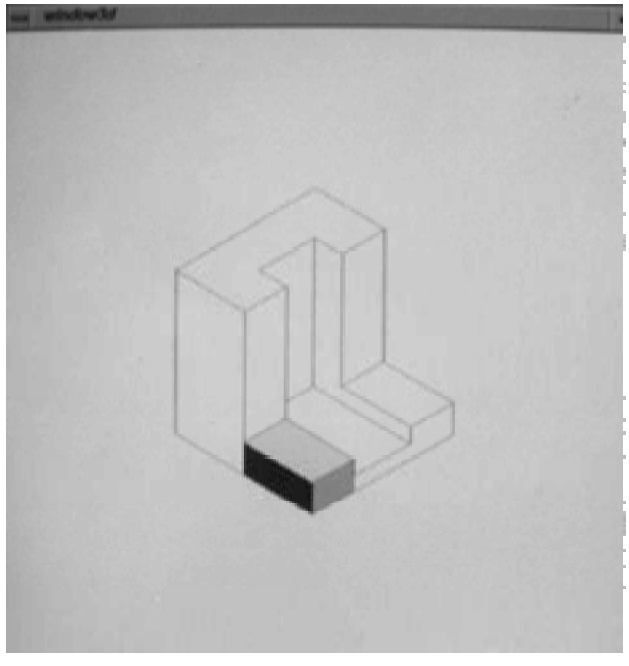


(d)

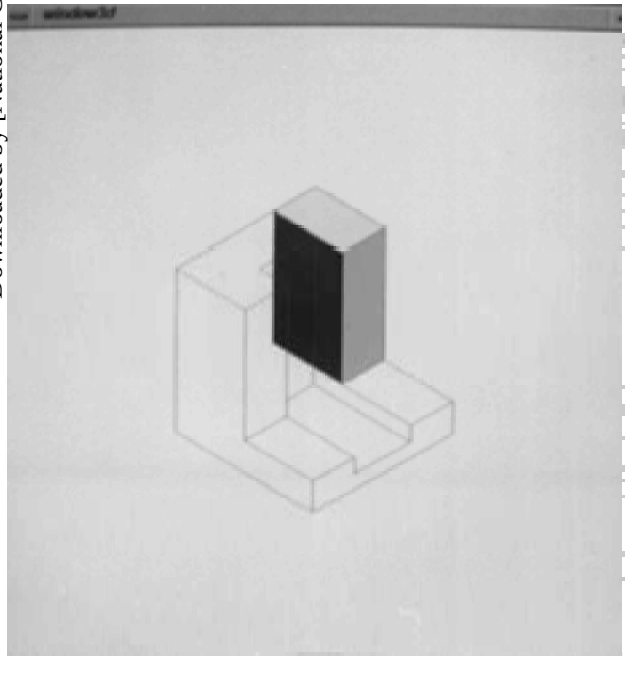
Figure 10. First testing example of implementation.



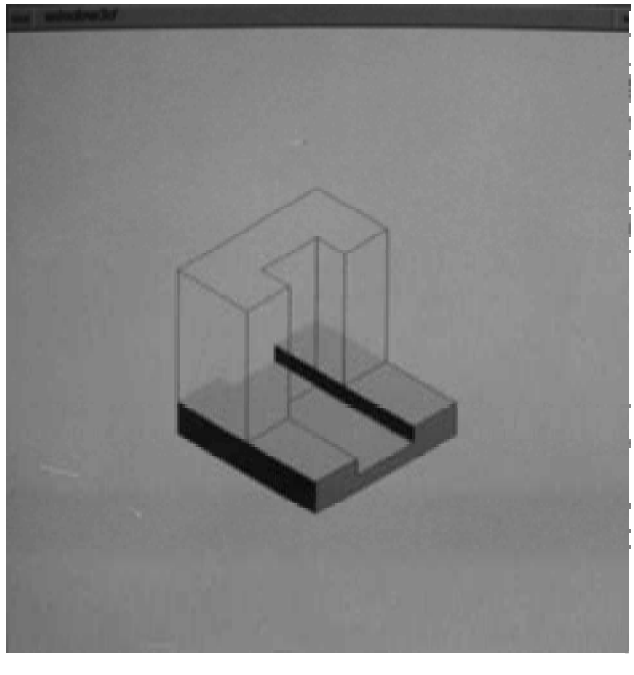
(e)



(f)

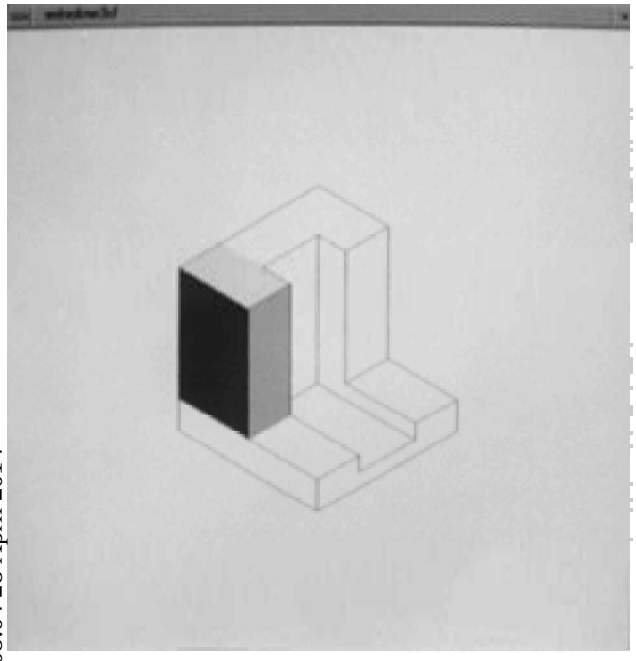


(g)

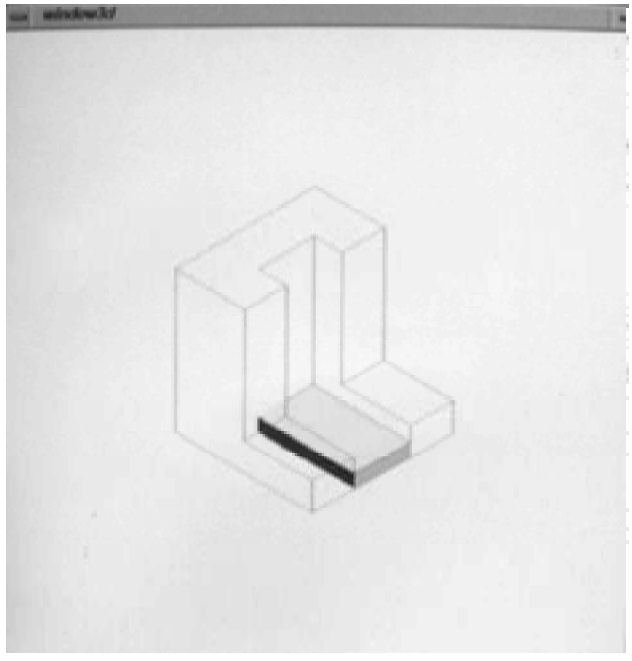


(h)

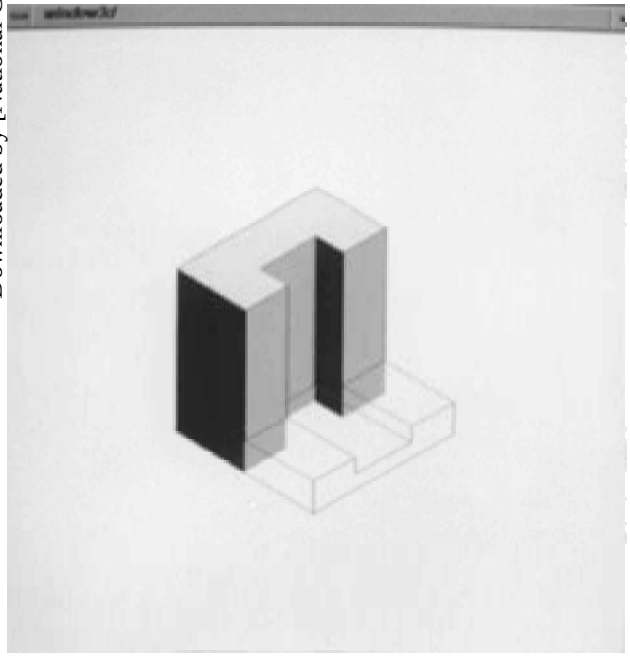
Figure 10.— *continued.*



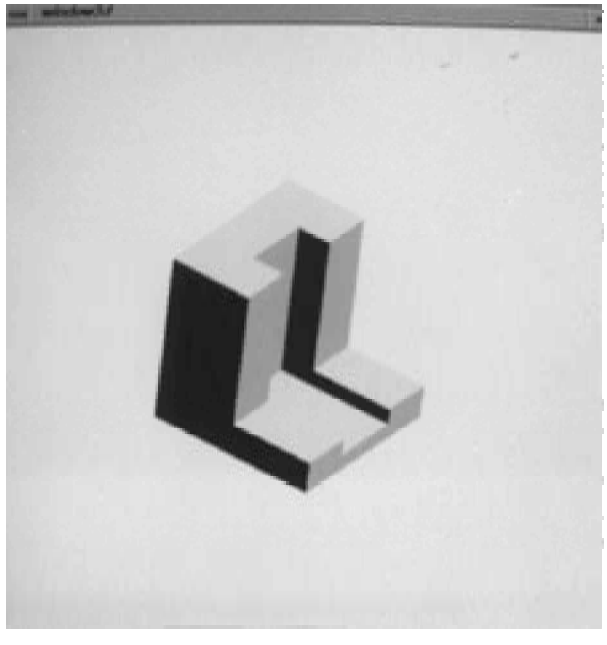
(i)



(j)



(k)



(l)

Figure 10.— *continued.*

therefore the first and the second methods stated above cannot be applied.

For such a partially visible loop, we have to determine two new vertices, one on each extended ray, in order to create new edges and modify this loop into a closed one. Because each new vertex is invisible on the isometric drawing, this indicates that in the 3D world it has to be contained in the original form of either a partially visible face or an almost invisible one. That is, a new vertex should be the intersection of an extended ray and a plane, passing through a partially visible loop or an almost invisible one, which is here known as a *constraint plane* because it is used to constrain the length of a new edge. A new vertex is then the projection of an open vertex on to a constraint plane.

Consider the case where only one constraint plane exists. As shown in Figure 8(a), partially visible loop $\hat{L}(V_5, V_4, V_{11})$ has two extended rays, with no intersection and both pointing in the $-Z$-axis direction. The two extended rays intersect the plane passing through the almost invisible loop $\hat{L}(V_1, V_0, V_{10})$ because their normals are parallel to the Z axis. As the 3D coordinates of all vertices on the isometric drawing are known, we can easily compute the projections of vertices V_5 and V_4 on to the constraint plane, which in turn are taken as the new vertices. The two projections or new vertices (V_{13} and V_{14}) then can be used to create three new edges (V_5V_{14}, $V_{14}V_{13}$, and $V_{13}V_{11}$) and modify the partially visible loop into a closed one.

Notice that if new edges V_5V_{14} and $V_{11}V_{13}$ are equally shortened by a certain length, the partially visible loop can still be modified into a closed one (Figure 8(b)). Moreover, one valid 3D solid would also exist for such an isometric drawing. That is, a valid solid for the isometric drawing can be seen as the union of two subsolids, one is the translation of loop $L_1 = L(V_x, V_5, V_6, V_7, V_8, V_9, V_2)$ and the other is that of loop $L_2 = L(V_x, V_5, V_4, V_3)$. The translation distance for loop L_1 is fixed (the length of V_1V_2); however that of loop L_2 can be varied (any length in between that of V_4V_{11} and V_5V_{14}). The union of these two subsolids can generate an infinite number of valid solids by varying the translation distance of loop L_2.

In this research, we prefer to construct a solid which tends to have a minimum number of faces and edges. Therefore, it is proposed that the open vertices projections on a constraint plane are chosen as the new vertices. That is, new edges V_5V_{14} and $V_{13}V_{11}$ in Figure 8(a) are not shortened, even though valid solids can also be generated when they are shortened.

For cases where two or more constraint planes exist, it is suggested that the one which is nearest to the open vertices be chosen for determining the new vertices. The other constraint planes, more distant to open vertices, are not chosen because the new edges created may pass through the workpiece and stay partly in free space, which is prohibited. As shown in Figure 8(c), for partially visible loop $\hat{L}(V_5, V_4, V_{11})$, two constraint planes exist, passing through the almost invisible loops $\hat{L}(V_1, V_{12}, V_{14})$ and $\hat{L}(V_{13}, V_0, V_{10})$. The constraint plane passing through $\hat{L}(V_{13}, V_0, V_{10})$, closer to open vertex V_5, is chosen for determining new vertices. Should the other constraint plane be chosen, a portion of the new edge (V_5V_x) would exist in the free space (Figure 8(c)).

As previously mentioned, a plane passing through a partially visible loop may also be used to create new vertices and edges. However, such a plane is not used as a constraint plane for the following two reasons. First, if the partially visible loop can be modified into a closed one, the projection of an open vertex on to the constraint plane should also be a vertex of the partially visible loop; yet this may not always be true. Second, an extended ray would cross the boundary of the isometric drawing and can always intersect at least one plane containing an almost invisible loop; however, it may not always intersect a plane containing a partially visible loop.

8. Construction of subsolids and final workpiece

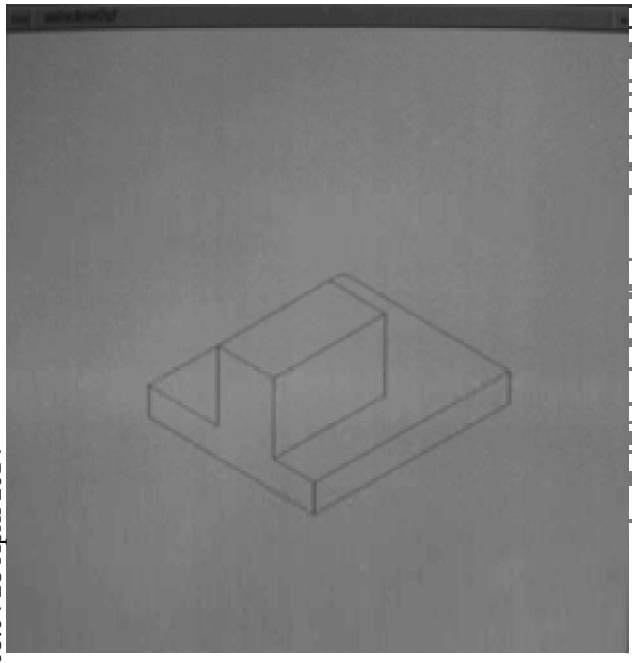
After modifying all partially visible loops into closed ones, each face bounded by a closed loop can be used to generate a subsolid in the 3D world. The union of all subsolids is the final workpiece proposed in this research.

8.1. *Creating subsolids by sweeping operation*

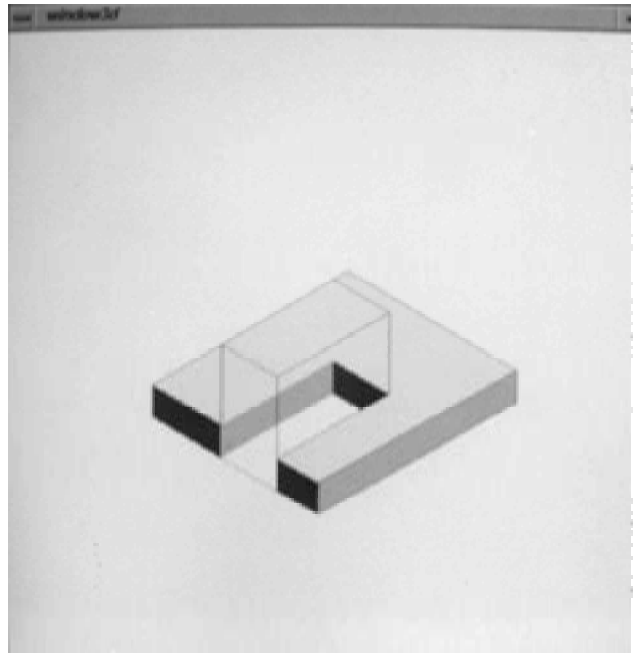
A subsolid is a 3D object created by a sweeping operation which moves a face on the workpiece, along its inward normal, by a certain distance. The face to be moved on the isometric drawing is one bounded by a closed loop which was either a completely visible one or a partially visible one that has been modified into the closed form. As shown in Figure 9(f), the face bounded by completely visible loop $L(V_1, V_{14}, V_{13}, V_{21}, V_{17}, V_8)$ and that bounded by closed loop $L(V_{15}, V_{16}, V_{24}, V_{20}, V_{22})$, originally a partially visible one, can both be swept by a certain distance to generate a subsolid in the 3D world. To facilitate discussion, the face to be moved in a sweeping operation is called the *primitive face*, its moving direction and distance are respectively known as the *sweeping direction* and *sweeping distance*.

8.2. *Computing sweeping distance*

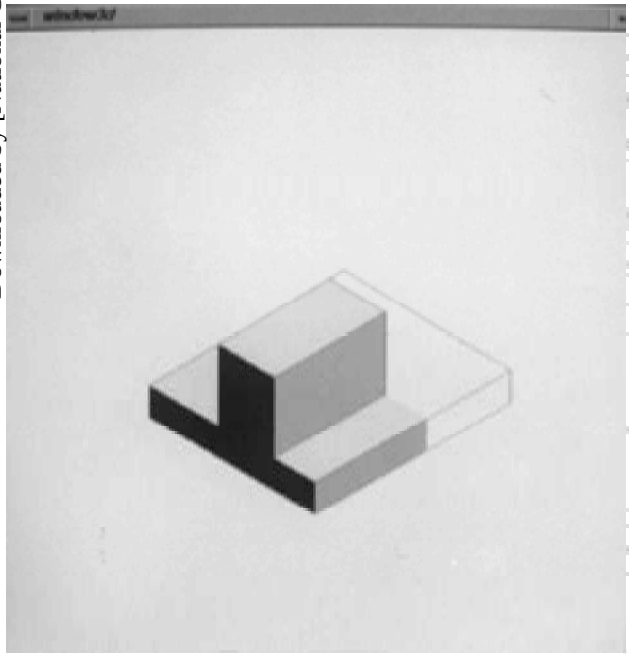
The sweeping distance of a primitive face is constrained by prohibiting the face from moving into the free space. That is, the primitive face can be continuously moved until it intersects an almost invisible face whose outward normal points in the sweeping direction. Crossing such an almost invisible face will



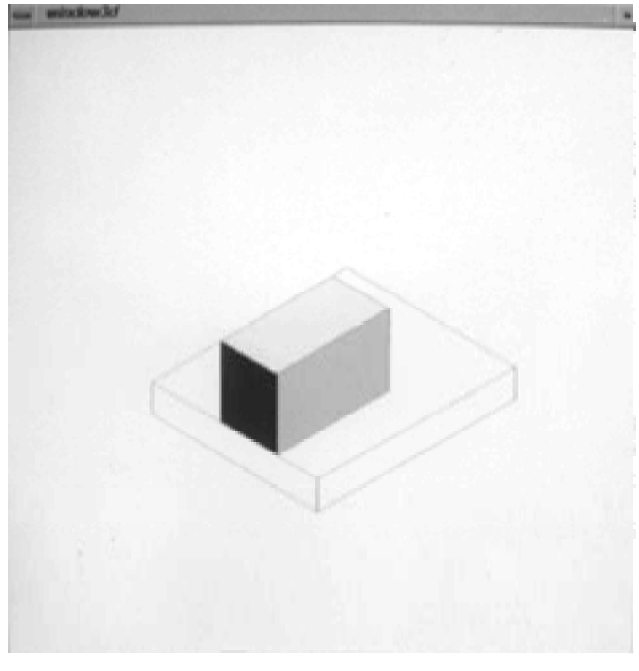
(a)



(b)

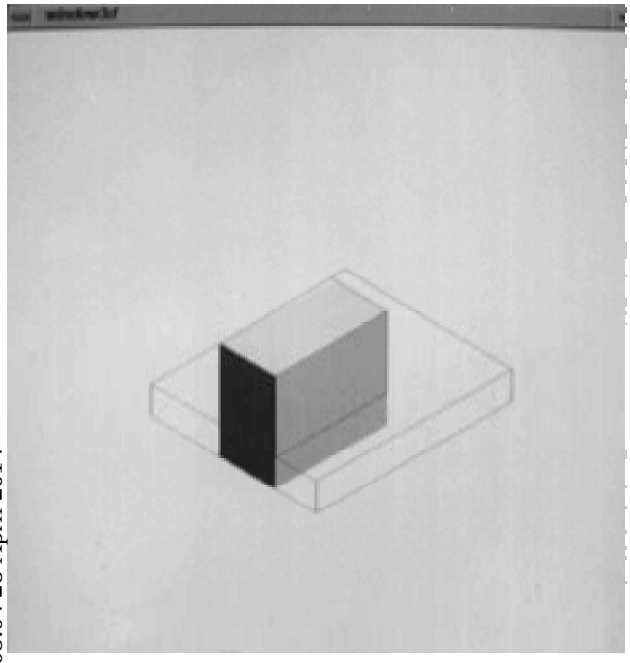


(c)

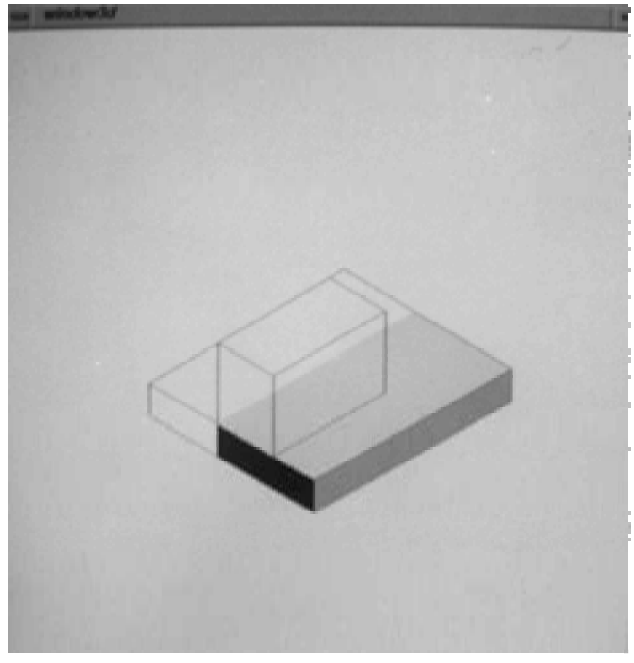


(d)

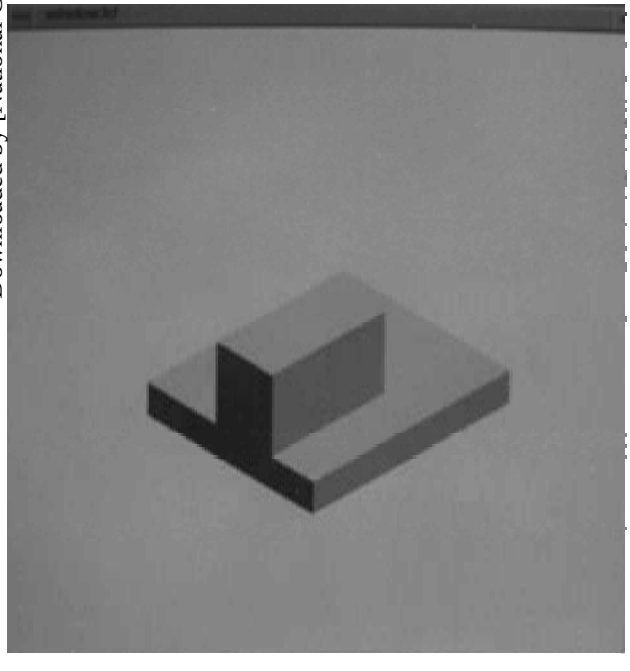
Figure 11. Second testing example of implementation.



(e)



(f)



(g)

Figure 11.— *continued.*

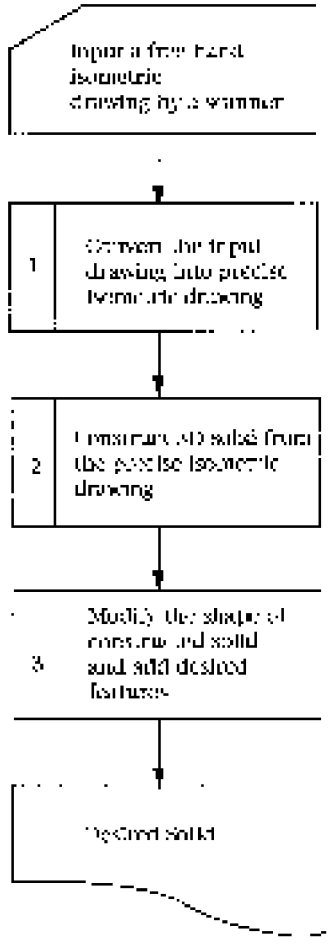


Figure 12. Framework of a user-friendly solid modelling input system.

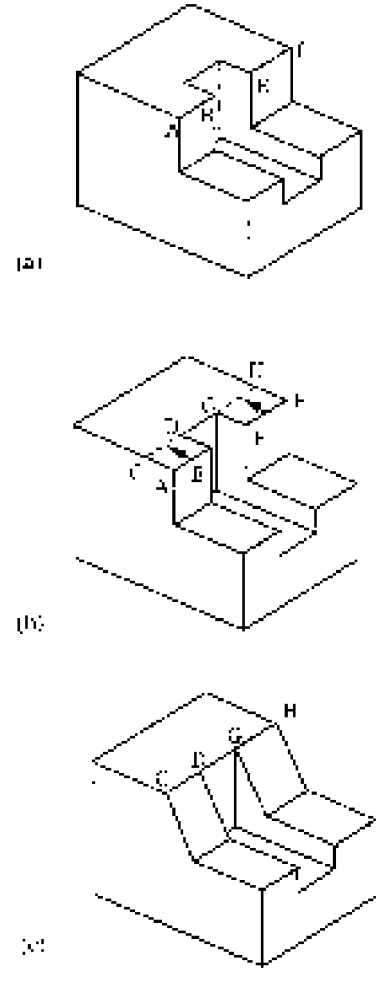


Figure 13. A wireframe based approach of variational geometry.

invade free space. As shown in Figure 9(f), the primitive face bounded by loop $L(V_1, V_{14}, V_{13}, V_{21}, V_{17}, V_8)$ can be continuously moved until it intersects the face bounded by the almost invisible loop $\hat{L}(V_{20}, V_{19}, V_{18}, V_{16}, V_7, V_6)$. The sweeping distance then can be determined to be equal to the length of edge $V_{17}V_{18}$. Should the sweeping distance be a little longer, the primitive face would cross the workpiece boundary and generate an undesired subsolid, which is partly contained in the free space.

It may be questioned why the face stopping the movement of a primitive face should be an almost invisible one and with its outward normal pointing in the sweeping direction. By definition, the outward normal of a workpiece face always points towards free space. Crossing a workpiece face while advancing along its outward normal, the primitive face would move into the free space, which is prohibited. Therefore, the stopping face should have its outward normal pointing in the sweeping direction. This also indicates that the stopping face is an almost invisible one, because its outward normal is opposite to that of the primitive face—a visible one.

In summary, the sweeping distance of a primitive face can be determined by computing the distance between two parallel planes, one passing through the primitive face and the other (known as the *stopping plane*) passing through an almost invisible loop whose projection intersects with the primitive face. If the number of such planes is more than one, the one which is closest to the primitive face should be chosen. Should the other ones be chosen, the generated subsolids might invade free space.

9. Validity of final workpiece

In the process of generating subsolids, each visible face on the isometric drawing has to individually generate one subsolid. The union of all subsolids is the final workpiece proposed in this research. It is essential to explain or prove that the proposed workpiece is valid; that is, it has the same isometric drawing as the input one.

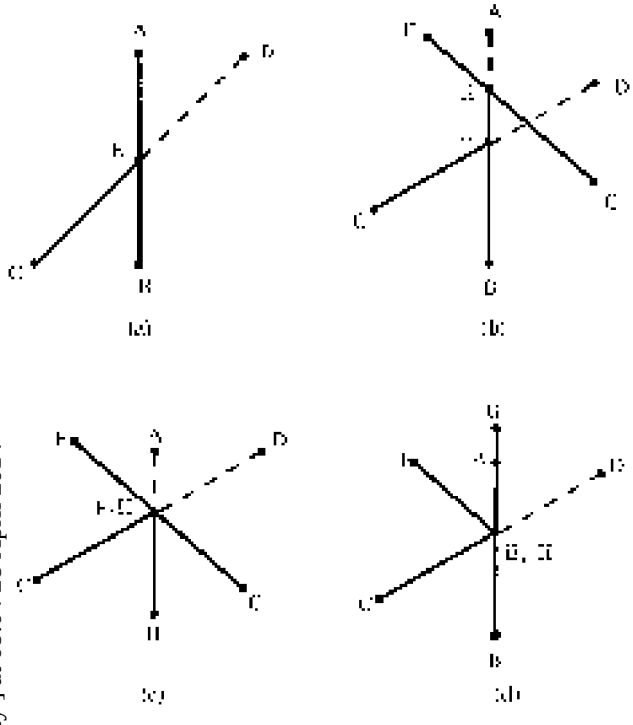


Figure 14. Characteristics of collinear vertices.

9.1. *Isometric drawing of an individual subsolid*

We first investigate the visibility of an individual subsolid. Faces on the boundary of a subsolid involve three types: the primitive face, the face on the stopping plane, and those generated by the movement of edges on the primitive face. The primitive face is visible, and the face on the stopping plane is invisible. Of faces generated by the movement of edges, some would be visible and some not.

To facilitate discussion, an edge on a primitive face is known as a *sweeping edge*, and the face generated by sweeping the edge is called a *sweeping region*. A sweeping edge is the intersection of two faces, one is the primitive face and the other one is known as the *alien face*. For example, considering the primitive face $L(V_1, V_{14}, V_{13}, V_{21}, V_{17}, V_8)$ in Figure 7(f), edge V_1V_8 is a sweeping edge, face $L(V_1, V_8, V_7, V_6, V_5, V_4, V_3, V_2)$ is its alien face and face $L(V_1, V_x, V_7, V_8)$ in Figure 7(g) is its sweeping region.

If the alien face of a sweeping edge is bounded by an almost invisible loop, then the sweeping region of this edge should be on a plane passing through the almost invisible loop, which is invisible on an isometric drawing; therefore the sweeping region also has to be invisible. As shown in Figure 7(g), for primitive face $L(V_1, V_{14}, V_{13}, V_{21}, V_{17}, V_8)$, the sweeping regions created by edges V_1V_{14} and $V_{14}V_{13}$ are both invisible with respect to the isometric drawing of the subsolid.

If the alien face of a sweeping edge is completely visible or partially visible, the sweeping region of this edge should be on a plane which is visible on an isometric drawing. This indicates that the sweeping region should also be visible. As shown in Figure 7(g), for primitive face $L(V_1, V_{14}, V_{13}, V_{21}, V_{17}, V_8)$, the sweeping regions created by edges V_1V_8, V_8V_{17}, $V_{17}V_{21}$, and $V_{21}V_{13}$ are all visible with respect to the isometric drawing of the subsolid.

9.2. *Ignore the information of visible sweeping regions*

Notice that a sweeping region generated by an edge on a visible plane is a subset of its alien face. As shown in Figure 7(g), sweeping region $L(V_1, V_8, V_7, V_x)$ is a subset of the visible face $L(V_1, V_8, V_7, V_6, V_5, V_4, V_3, V_2)$. This characteristic can be briefly proved as below.

If the sweeping region is not a subset of the alien face, then some portion of the sweeping region should be outside the boundary of the alien face. This implies that the sweeping region already invades free space. Therefore, a visible sweeping region is a subset of its alien face which is a primitive face of another subsolid and has to be visible. That is, the visibility information of a visible sweeping region is always embedded in the primitive face of some other subsolid. Therefore, a subsolid essentially gives the visibility information of its primitive face.

9.3. *Hidden effects among primitive faces*

With the generation of all subsolids, each one of the primitive faces is completely visible if we consider only the isometric drawing of an individual subsolid. However, some part of a primitive face might be hidden when the other subsolids are also displayed. The hidden results among primitive faces should be exactly the same as the input isometric drawing, because the input one can be seen as being solely composed of primitive faces. Furthermore, the hidden effects among primitive faces should be also the same as that among subsolids, because for a subsolid only the visibility information of primitive faces is meaningful. Therefore, we can conclude that the isometric drawing of the final workpiece should be the same as the input one.

10. Implementation and application

The proposed algorithm has been implemented in the C language on an IRIS workstation equipped with the ACIS solid modeller (Spatial Technology 1986). The final work-piece together with the generated subsolids can be graphically displayed on the IRIS workstation. Two of the various testing examples are illustrated below.

The first one addresses the isometric drawing shown in Figure 10(a). As discussed previously, this isometric drawing has ten simple loops or visible faces; therefore ten subsolids should be generated (Figure 10(b–k)).

The isometric projection of the final workpiece is given in Figure 10.1. From this figure, we can see that the final workpiece has exactly the same isometric drawing as the input one.

The second testing example is given for the demonstration of processing overlaid edges. The input isometric drawing appears to have five simple loops (Figure 11(a)); however, it has only four when the overlaid edge is taken into two. That is, two simple loops would be combined into one and generate a subsolid (Figure 11(b)). The other four subsolids are as shown in Figure 11(c–f). As we can see, the final workpiece also has the same isometric drawing as the input one (Figure 11(g)).

The present algorithm can be integrated with some other algorithms to constitute a user-friendly solid modelling input system. The framework of such a solid modelling input system is as shown in Figure 12. A free-hand isometric drawing is input to the system by a scanner and is represented by a set of pixels. The first module applies some computer vision technique to convert the free-hand drawing into a precise one that is represented by a set of line segments (Chang 1993). The second module utilizes the algorithm presented in this paper to construct a 3D solid from the precise isometric drawing. A wireframe-based approach of variational geometry is developed in the third module which can modify the shape of the constructed solid and add features on it (Haan 1993).

In the third module, a constructed workpiece (Figure 13(a)) can be modified into one (Figure 13(c)) which involves inclined faces. The modification is by interactively manipulating the wireframe of the constructed solid. As shown in Figure 13(b), by moving edges *AB* and *EF* to new positions (*CD* and *GH*), the wireframe is modified and its bounded planar faces subsequently will be changed. Some other operations such as smoothing the sharp junction of two planar faces and the addition of hole features are also included in this module. Note that the constructed solid can be interactively rotated so that desired features or modification can be made on a portion of the solid, which is originally hidden on the input isometric drawing.

11. Concluding remarks

An algorithm for constructing a CSG solid from a single isometric drawing is proposed. Currently, the work-pieces are limited to rectilinear simple polyhedra. This algorithm is distinguished by a capability to solve the degenerate cases (edge overlapping problems) which have been ignored in previous relevant research, and the construction of primitives to represent the CSG polyhedron are not limited to blocks.

One possible extension of this work is to enlarge the workpiece domain and vary the projection methods of input drawings. That is, the constraints on the shape and the orientation of workpiece faces can be relaxed, and the input drawing can be a dimetric projection, trimetric projection, oblique projection, or perspective projection. The other possible extension is the development of an algorithm for constructing a 3D solid from its engineering drawings which may be so comprehensive that all the relevant views are included.

Acknowledgement

This study was supported by National Science Council, Republic of China, under Contract number NSC 83-0415-E-009-001.

References

ALDEFELD, B., 1983, On automatic recognition of 3D structures from 2D representations. *Computer-Aided Design*, **15**, 59–64.

ALDEFELD, B., and RICHTER, H., Semiautomatic three dimensional interpretation of line drawing. *Computer & Graphics*, **8**, 371–380.

CHANG, K. H., 1993, *An Image Processing System for Freehand Drawings of Pictorial Views*, Master thesis, National Chiao Tung University, Taiwan.

CHEN, Z., and PERNG, D. B., 1988, Automatic reconstruction of 3D solid objects from 2D orthographic views. *Pattern Recognition*, **21**, 439–449.

CHEN, Z., PERNG, D. B., and WU, C. S., 1992, Fast reconstruction of 3D mechanical parts from 2D orthographic views with rules. *International Journal Computer Integrated Manufacturing*, **5**, 2–9.

CLOWES, M. B., 1971, On seeing things. *Artificial Intelligence*, **2**, 79–112.

FRENCH, T. E., VIERCK, C. J., and FOSTER, R. J., 1986, *Engineering Drawing and Graphic Technology* (McGraw-Hill, New York).

GUJAR, U. C., and NAGENDRA, I. V., 1989, Construction of 3D solid objects from orthographic views. *Computers & Graphics*, **13**, 505–521.

GUZMAN, A., 1968, Decomposition of a visual scene into three-dimensional bodies. *AFIPS Proceedings Fall Joint Computer Conference*, **33**, 291–304.

HAN, T. F., 1993, *A Wireframe Variational Geometry Method*, M.S. thesis, National Chiao Tung University, Taiwan.

HARALICK, R. M., and QUEENEY, D., 1982, Understanding engineering drawings. *Computer Graphics and Image Processing*, **20**, 244–258.

HO, B., 1986, Inputting constructive solid geometry representations directly from 2D orthographic engineering drawings. *Computer-Aided Design*, **18**, 147–155.

HUFFMAN, D. A., 1971, Impossible object as nonsense sentences. In B. Heltzer and D. Michie (eds) *Machine Intelligence* (Edinburgh University Press) pp. 295–323.

IDESAWA, M., 1973, A system to generate a solid figure from three-view. *Bulletin of the JSME: Japan Society of Mechanical Engineers*, **16**, 216–225.

IWATA, K., YAMAMOTO, M., and MICHIKO, I., 1988, Recognition system for three-view mechanical drawings. *Pattern*

Recognition 4th International Conference Proceedings, pp. 240–249.

LAFUE, G., 1976, Recognition of three-dimensional objects from orthographic views. *ACM Computer Graphics*, **10**, 103–108.

LAFUE, G., 1978, A theorem prover for recognizing 2-D representations of 3-D objects. *Proceedings of the IFIP TC-5 Working Conference on Artificial Intelligence and Pattern Recognition in Computer Aided Design*, pp. 391–401.

MARKOWSKY, G., and WESLEY, M. A., 1980, Fleshing out wireframes. *IBM Journal of Research and Development*, **24**, 582–597.

MULLER, A., and RICHTER, D., 1990, Reconstruction of a boundary representation model from three orthographic projections: a geometrical approach. *Eurographics '90*, pp. 237–250.

PREISS, K., 1981, Algorithms for automatic conversion of a 3-view drawing of a plane-faced part to the 3D representation. *Computers in Industry*, **2**, 133–139.

PREISS, K., 1984, Constructing the solid representation from engineering projections. *Computers & Graphics*, **2**, 381–389.

REQUICHA, A. A. G., 1980, Representations for rigid solids: theory, methods and systems. *Computing Surveys*, **12**, 437–464.

RICHARDS, T. H., and ONWUBOLU, G. C., 1986, Automatic interpretation of engineering drawings for 3D surface representation in CAD. *Computer Aided Design*, **18**, 156–160.

SAKURAI, H., and GOSSARD, D. C., 1983, Solid model input through orthographic views. *ACM Computer Graphics*, **17**, 243–252.

SPATIAL TECHNOLOGY INC., 1986, *ACIS Geometric Modeler Interface Guide.*

SUGIHARA, K., 1986, *Machine Interpretation of Line Drawings* (MIT Press).

TU, Y. H., 1992, *An Investigation on the Demand and Supply of CAD Systems in Product Design*, Master thesis, National Chiao Tung University, Taiwan.

WANG, W., and GRINSTEIN, G., 1989, A polyhedral object's CSG-Rep reconstruction from a single 2D line drawing. *Proceedings of 1989 SPIE Intelligent Robots and Computer Vision III: Algorithms and Techniques*, **1192**, 230–238 (Nov 1989).

WANG, W., 1992, *On the automatic reconstructon problem of a 3D object's constructive solid geometry representation from its 2D projection.* Doctor of Science dissertation, University of Massachusetts at Lowell.

WANG, W., and GRINSTEIN, G., 1993, A survey of 3D solid reconstruction from 2D projection line drawings. *Computer Graphics Forum*, **12**, 137–158.

WESLEY, M. A., and MARKOWSKY, G., 1981, Fleshing out projection. *IBM Journal of Research and Development*, **25**, 934–954.

YOSHIURA, H., FUJIMURA, K., and KUNII, T. L., 1984, Top-down construction of 3-D mechanical object shapes from engineering drawings. *IEEE Computer*, **12**, 32–40.

ZEID, I., 1991, *CAD/CAM Theory and Practice* (McGraw-Hill, New York).

Appendix 1

This appendix explains that a collinear vertex may have multiple inverse projections, while a non-collinear vertex has only one.

As shown in Figure 14(a), two edges ($\overline{AB}$ and $\overline{CD}$) which do not intersect in 3D world may have an intersection on their isometric projections; some portion of an edge (ED) is then hidden and becomes invisible. The intersection (point E) on the isometric drawing is a collinear point, and has multiple inverse projections. One may wonder whether the collinear characteristics may disappear by the presence of some other edges.

A new edge (FG) which may interfere with the collinear characteristic would be present in three patterns. First, as shown in Figure 14(b), the new edge tends to hide an ending point of an existing edge (e.g. point A) such that the collinear characteristic would disappear. However, this intention would require an intersection (point H) to occur between edges FG and AB. The existence of intersection point H, in turn, replace the role of point A and keep the collinear characteristic of point E.

Second, the new edge (FG) tends to hide the collinear point (point E), and the collinear point becomes an in-between point of the new edge (Figure 14(c)). As we can see, edge AE becomes wholly hidden by the presence of the new edge; even so, point E now preserves its collinear characteristic on the new edge FG.

Third, see Figure 14(d), the new edge (FH) intersects the collinear point (point E) at one of its vertex (point H). Since point H is a vertex, it would have three neighbouring edges; one is edge FH and the other two would be collinear with two of the four edges (AE, CE, BE, and DE), because there are only three types of edges. Wherever the other two edges are placed, we can see that point E always keep the collinear characteristic.

Appendix 2

This appendix explains three characteristics of a collinear vertex. First, if a collinear vertex has only one neighbouring vertex in a group, then the unique neighbouring edge in this group is a partially visible edge. Second, if it has two or more neighbouring vertices in a group, then the neighbouring edges in this group are all completely visible.

For the first characteristic, see Figure 14(a), a partially visible edge is one where one of its ending points (point D) has been hidden and the collinear vertex (point E) appears as the new ending point. The 3D coordinate of the collinear vertex (point D) can only be inferred from the ending point which is not hidden (point C). Consider the cases where another neighbouring point (e.g. either point A or point B) can be used to infer the same 3D coordinate. In such cases, point E would be concluded as a vertex rather than an in-between point, this implies that edge CE is not a partially visible one-conflict situation. Conversely, we may validate the second characteristic; that is, when a collinear vertex has two or more neighbouring vertices in a group, all the neighbouring edges would be completely visible. Should there exist a partially visible one, it would be impossible to infer only one 3D coordinate for the concerned collinear vertex.