Handling Sectional Views in Volume-based Approach to Automatically Construct 3D Solid from 2D Views

Jitendra Dimri and B. Gurumoorthy
Dept. of Mechanical Engineering
Indian Institute of Science
Bangalore 560012, INDIA
{jeetu, bgm} @mecheng.iisc.ernet.in

Abstract

This paper presents an algorithm for solid model reconstruction from 2D sectional views based on volume-based approach. None of the existing work in automatic reconstruction from 2D orthographic views have addressed sectional views in detail. It is believed that the volume-based approach is better suited to handle different types of sectional views. The volume-based approach constructs the 3D solid by a boolean combination of elementary solids. The elementary solids are formed by sweep operation on loops identified in the input views. The only adjustment to be made for the presence of sectional views is in the identification of loops that would form the elemental solids. In the algorithm, the conventions of engineering drawing for sectional views, are used to identify the loops correctly. The algorithm is simple and intuitive in nature. Results have been obtained for full sections, offset sections and half sections. Future work will address other types of sectional views such as removed and revolved sections and broken-out sections.

Keywords : Orthographic projections, Sectional views, Solid models, Reconstruction

1 Introduction

An engineering drawing has been the principal means for the communication of shape and other product information. Engineering drawing allows technical data to be recorded on paper so that a person can translate the information precisely into the specified product. Although a drawing contains complete information about an object, realising it for other task domains requires human interpretation. A solid model is more useful in the design process as it is easier to visualise and modify. It is also useful in other design tasks such as CFD/FEM, rapid prototyping, interference detection.

Figure 1. (a) Drawing with many hidden lines[12] (b) Sectional view

The problem of constructing 3D solid model from 2D views has been well researched (refer [4, 10] for a review of current literature). The current art addresses only regular orthographic views. A considerable portion of the existing drawings contain sectional views. Under the conventions of orthographic projections the interior part of an object that are hidden from the view direction can only be represented by dashed lines. When these lines become numerous the drawing is difficult to interpret. A sectional view is used to overcome this difficulty. They are easy to draw and read. Figure 1 shows a drawing with many hidden lines and its corresponding sectional view¹. Clearly a sectional view is easier to understand.

For some objects whose shape varies rapidly the principal views are not enough to describe them completely. Sections are helpful in such situations. Figure 2 shows such an object. Therefore, the sectional views are used to present hidden details without cluttering the drawing.

¹figures in this paper are drawn using first angle projection

Figure 2. Removed section[11]

Figure 3. Revolved section[12]

1.1 Sectional Views

A section is any view seen when a portion of the object nearest to the observer has been imaginarily removed by a cutting plane or planes, thus revealing interior construction. There are different types of sectional views. The following definitions from [11, 12] are repeated here for completeness.

1. Full sections

This section is produced by one cutting plane and shows not only what appears in this plane, but also what can be seen behind this plane. Figure 1(b) is an example of full section.

2. Removed sections

This indicates the cross sectional shape of the object at a particular location. Figure 2 is an example of removed section.

3. Revolved sections

Revolved section is produced by cutting through the object at a chosen location by an imaginary plane and then revolving the view obtained by 90° so the cross-sectional shape at this location can be seen. This section view is drawn on top of the existing view. Figure 3 shows a revolved section.

4. Half sections

For some objects, an external view or a full section alone is incapable of completely revealing the shape and internal features. For example in Figure 4, if we made a full section of the front view, the top view alone would fail to show the height of the upper eye as this element would be absent in the section. In such cases both the external view and the sectional view are combined in one view in an appropriate proportion. The boundary between the portion can be a freehand break line or it can be line of symmetry.

Figure 4. Half section[11]

5. Broken-out sections

Broken-out sections reveal the shape of internal features such as small blind hole in large objects. Figure 5 shows an example of a broken-out section.

6. Offset sections

Here several cutting planes, offset or bent, are used to cut through the portions of interest. Figure 6 shows an offset section.

1.2 Literature Review

Although lot of work has been done on the reconstruction of a three dimensional (3D) solid model from two dimensional (2D) orthographic projections, the problem of reconstruction from sectional views has not been addressed in any detail. The approaches reported thus far can be classified into two broad categories [10] - the bottom-up or wireframe based approach and the top-down or volume-based approach. In the wireframe approach [5, 6, 13, 14], 3D vertices are first generated from the input 2D views. These

Figure 5. Broken-out section[11]

Figure 6. Offset section

3D vertices are connected by 3D edges in accordance to the input views, resulting in a wireframe. This wireframe is then covered with all possible faces to obtain the solution solids. Obtaining solids from a wireframe is a time consuming process. The correct solution is identified by comparing the solution solids with input views. Only Markowsky and Wesley [13] have discussed handling sectional views using wireframe approach. They take sectional views (full section only) into consideration along with regular orthographic views. If there is a sectional view in the input then in the step of generating 3D vertices, an additional condition has to be satisfied in the location of a candidate vertex. The candidate vertex must lie in the half-space defined by the section plane and view direction. A sectional view will provide some extra vertices and edges to be processed, thereby increasing the combinatorial problem associated with wireframe approach. They consider only full sectional views and no results of implementation have been provided for drawings with sectional views. Geng *et al.* [5] mimic human engineer's approach to interpret the solid model from 2D views. The object space is divided into layers for its fast examination. Then a *baseface* is selected, sometimes

with the help of user. Every vertex of the *baseface* is then swept along its sweep trajectory. When selection of a *baseface* is difficult from regular views they use a sectional view. Therefore, they use the sectional view to only assist in the reconstruction from regular orthographic views. They do not handle the sectional views themselves explicitly.

The volume-based approach [1, 9, 10], works by identifying structures like loops in the input views which can be swept to create elementary solids. The combination of elementary solids gives the solution solid. This approach has a limited domain as the solids with faces inclined to all projection planes can not be obtained. The volume-based, automatic reconstruction techniques by Aldefeld [1] or by Shum *et al.* [9] can not handle sectional views because they do not account for the changes in terms of edges and loops in sectional view as compared to a regular view. There are also semiautomatic, volume based methods by Aldefeld & Richter [2] and by Bin [3] which also take sectional views into consideration but these methods require user intervention.

This paper describes a volume based approach (based on that described in [10]) to handle sectional views. In the following, we first present some terminology and definitions followed by an overview of our volume based approach. Issues in handling sectional views in this framework are then described. Details of the algorithm are then presented followed by results of implementation. The paper concludes with a discussion of the current limitations of the proposed approach and scope for future work.

2 Definitions and Terminology

This section presents definitions and terminology used in the description of the algorithm. Most of the definitions and terms here are from Soni & Gurumoorthy [10].

1. **Common coordinate axis :** It is the coordinate axis that is common between a pair of views. In Figure 7, x-axis is the common coordinate axis between view 1 and view 2.
2. **Loop :** A simple closed cycle of edges in a view is defined as a loop.
3. **Tangency vertex :** A vertex in a 2D view that separates a pair of edges that are first order continuous, is said to be a tangency vertex. Since the notion of tangency vertex is associated with a pair of edges, it is used for vertices in a loop. In Figure 7, the vertex 5 and 6 in the loop{3,4,5,6,7,8,3 } in view 1, are tangency vertices. A tangency vertex can also be in between two curved edges or one curved edge and one straight edge.

Figure 7. Matching vertices and matching loops

4. **Silhouette vertex :** A vertex ${}^{G_1}V$ on a curved edge in a 2D view G_1 is said to be silhouette with respect to an adjacent view G_2 if the tangent to the edges at that vertex is perpendicular to the common coordinate axis between G_1 and G_2 . Silhouette vertex is therefore defined for a pair of views. For example in Figure 7, vertices 1 and 2 on the circular loop in view 1 are silhouette vertices with respect to view 2.
5. **Extreme vertices of a loop :** Vertices of a loop, with minimum and maximum coordinate values along *Common coordinate axis* are called *Extreme vertices of a loop*.
6. **Classification of loops :** A loop ${}^G L_2$ can be classified with respect to loop ${}^G L_1$ as follows:
 - loop ${}^G L_1$ is said to have IN loop ${}^G L_2$, if all edges ${}^G E \in {}^G L_2$ lie inside region **P** defined by loop ${}^G L_1$.
 - if some edges ${}^G E \in {}^G L_2$ lie inside and others on the boundary of region **P**, then ${}^G L_1$ is said to have ON-IN loop ${}^G L_2$.
 - ${}^G L_1$ is said to have ON-OUT loop ${}^G L_2$, if some ${}^G E \in {}^G L_2$ lies outside and others on the boundary of region **P**.
7. **Solid-loop & dotted loop :** A loop in a 2D view is said to be solid-loop, if all of its edges are represented by solid line type in the view. Otherwise, it is called as dotted-loop.
8. **Fundamental loop & non-fundamental loop :** Fundamental loop is a loop that does not have any ON-IN loop in the view but it may have an IN loop. All other loops are called non-fundamental loops. Loops $\{1,2,1\}$, $\{5,6,9,10,5\}$ and $\{3,4,5,10,9,6,7,8,3\}$

in view 1 of Figure 7, are fundamental loops. Loop $\{1,2,3,11,12,6,7,8,9,10,1\}$ in view 2 is a non-fundamental loop. The fundamental loops in a view can be determined by the leftmost search described in Yan *et al.* [14].

9. **Disjoint loop :** A disjoint loop is a fundamental loop that does not have any ON-OUT loop in the view but it may have an OUT loop. Loop $\{1,2,1\}$ is a disjoint loop in Figure 7, view 1.
10. **Matching vertices :** A vertex ${}^{G_1}V$ in a view G_1 is said to be matching with a vertex ${}^{G_2}V$ in another view G_2 , if they have the same coordinate value along the common coordinate axis *i.e.*, ${}^{G_1}_i V = {}^{G_2}_i V$ where i denotes the coordinate of vertex along common coordinate axis, ${}^{G_1}V \in G_1(V, E)$, ${}^{G_2}V \in G_2(V, E)$ and $G_1 \neq G_2$. There may be one or more vertices in a view matching a vertex in the other view. For instance, in Figure 7, vertex 3 in view 1 is matching with vertices 1 and 2 in view 2.
11. **Matching edge of a vertex :** A straight edge ${}^{G_1}E \in G_1$ is called matching edge of a vertex ${}^{G_2}V \in G_2$ ($G_1 \neq G_2$), if both end vertices of the edge are matching vertices with vertex ${}^{G_2}V$.
12. **Matching loop :** A loop ${}^{G_1}L \in G_1$ is said to have matching loop ${}^{G_2}L \in G_2$ ($G_1 \neq G_2$), if every vertex in ${}^{G_2}V' \subset {}^{G_2}L$ matches with at least one vertex ${}^{G_1}V \in {}^{G_1}L$. ${}^{G_2}V'$ is a subset of all vertices in the loop ${}^{G_2}L$ obtained by removing all tangency vertices that are not silhouette vertices for this pair of views. This matching is denoted by ${}^{G_1}L \rightarrow {}^{G_2}L$. Note that it is not necessary that ${}^{G_2}L$ also have ${}^{G_1}L$ as a matching loop. If ${}^{G_1}L \rightarrow {}^{G_2}L$ and ${}^{G_2}L \rightarrow {}^{G_1}L$ then these loops are referred to as **onto-matching loops** denoted by ${}^{G_1}L \leftrightarrow {}^{G_2}L$. In Figure 7, loop $\{3,4,5,6,3\}$ and loop $\{9,10,11,12,9\}$ in view 2 has loop $\{5,6,9,10,5\}$ and loop $\{1,2,1\}$ respectively as onto-matching loop in view 1. On the other hand loop $\{6,5,10,9,6\}$ in view 1 in the figure 7 has a matching loop $\{1,2,3,11,12,6,7,8,9,10,1\}$ in view 2. However these two loops do not form an onto-matching loop as the loop $\{1,2,3,11,12,6,7,8,9,10,1\}$ in view 2 does not have loop $\{6,5,10,9,6\}$ as its matching loop in view 1.
13. **Onto-formation :** An onto-formation $\mathcal{L}$ is a set of loops, satisfying following properties :
 - a) for $i, j = 1, \dots, n$ ($i \neq j$), ${}^{G_i}L_i \leftrightarrow {}^{G_j}L_j$, where ${}^{G_i}L_i, {}^{G_j}L_j \in \mathcal{L}$
 - b) $|\mathcal{L}| = n$, where n = number of input 2D views.
14. **Desirable edge :** It is an edge in the input orthographic views that is not present in the 2D projection of current solid.

15. **Elemental solid :** All the member loops of an *Onto-formation* are swept perpendicular to their plane. The intersection of these swept solids gives an *Elemental solid*.
16. **Bound on solid :** The outer loop of every view is swept perpendicular to its plane. The intersection of all these obtained solids is called *Bound on solid*

3 Overview of our volume based reconstruction

Pre-processing of input

The drawings are imported in the .DXF format. After reading in the edges, vertices and information about sectional loops, view segmentation and view-plane folding as described by Lysak *et al.* [7]) is carried out.

View segmentation and View-plane folding

An engineering drawing may be considered as a planar graph. That means, it is a graph of vertices and edges embedded in the plane of the drawing such that two edges must meet only at a vertex. Once all vertices and edges are identified with respect to drawing-based coordinate system, these must be organised into views, and the view adjacency relationships must be determined. This is called *view segmentation*.

Now each view in the drawing is considered to be adjacent to at least one other view. A view tree is developed on the basis of relative position of the views and is used to determine the direction of each view. Next, the corresponding points in views are identified. Finally, each view is transformed from view-based coordinate system to object-based coordinates. This act of transforming views from plane of drawing sheet to respective view-planes is known as *view-plane folding*.

Apart from straight edges, circular edges are also allowed in the input views. It is assumed that the circular edges are segmented at silhouette vertices. Information about broken/dotted edges are also available in views. The fundamental loops and outer loop in each view can be determined by leftmost search and rightmost search respectively, as described in Yan *et al.* [14]. After processing of input drawing as described above, the segmented views in their folded locations, along with the additional information (fundamental loops etc.) are input to the main algorithm for conversion to 3D solid.

Before explaining the algorithm, we first describe two important operations used in the algorithm.

Validate Solid : Procedure *validate solid* checks for the consistency of a solid with respect to the input views.

By consistency we imply that all the projected edges of the solid is an edge or vertex in the input views.

The 3D edges in the solid are projected back to orthographic plane of each view. If all of these projections match with 2D edges in input views, then the solid is said to be valid. The 2D edges in input views that match the projection of 3D edges in elementary solid are flagged as *done* and remaining edges in views are classified as desirable edges. It should be noted that projection of 3D edges between tangency vertices that are not silhouette vertices are not considered during the validation. This is because these edges are artifacts of the sweep operator only and they will not be present in input views.

Boolean operations : Elementary solids, constructed by sweep are combined to get final solid. Boolean operations are performed to update current-solid. The current-solid is the solid obtained in intermediate stages of *boolean operations*. Initially it is the first elemental solid. Union or difference operations are used to add or remove the material corresponding to the elementary solid from the current-solid. The elementary solids are classified with respect to current-solid to decide the appropriate Boolean operation. Elementary solid (*ES*) and current-solid (*CS*) are classified as follows. If solid *ES* is ON-IN or IN with respect to solid *CS*, it implies that the solid *ES* is contained within the solid *CS*. So, difference operation is chosen for this case, since union will not alter the solid. In case when solid *ES* is ON-OUT with respect to solid *CS*, the two solids have overlapping boundaries(faces) . The union operation is possible in this case. First priority is given to the difference operation following which, the union operations are performed. If the elementary solid *ES* is either IN-OUT or OUT it is passed over for combining with the current-solid and this elementary solid is handled after combining other elementary solids.

The main steps involved in the algorithm are as follows.

1. Form *onto-formations* by matching of loops in different views.
2. Generate *elemental solids* from each *onto-formation*.
3. Validate *elemental solids*.
If validation is successful then go to step 5.
4. If validation is unsuccessful then obtain *bound on solid* and then go to step 5.

If validation is unsuccessful it implies that (a) there are entities in the interior of the views that can not be obtained by a sweep operation. (b) there are entities corresponding to the outer loop.

The latter case is handled by obtaining *bound on solid*.

Figure 8. (a) A sectional drawing[11] (b) Side view with hidden lines

Some of the former can be handled by allowing other types of sweep operators such as solids of revolution [10].

5. Apply *boolean operations* on all elementary solids obtained to get the solution solid.

The main task here is the identification of loops corresponding to sweep entities. *Onto-formation* is a tool to do so. Signature of these loops (*onto-formation*) gets disturbed due to interaction of entities in space, incomplete drawing or presence of sectional views.

4 Handling sectional views in volume based reconstruction

The key task is to use our knowledge of sectional views to identify the *onto-formations* in the presence of incomplete data caused by sectional views.

4.1 Some important points about sectional views

1. Sectional views normally do not affect other views, but sometimes when one view is sectional then its matching hidden lines in other view can be left out, if other views give complete information about the object. For example in Figure 8(a) the front view is sectioned. It can be observed that in the side view there should have been two dotted lines as shown in the Figure 8(b) with arrows. The important thing here is even without these lines the drawing is easy to understand and the view becomes more concise. Therefore, there may be missing matching elements (e.g. loops, edges) in a sectional drawing. If a set of matching loops do not form

Figure 9. A sectional drawing with hidden lines in hatched region

an *onto-formation* only because of missing entities in a sectional drawing then this set can be considered as an *onto-formation*.

2. Hidden lines behind the cutting plane are usually omitted for clarity. Sometimes hidden lines are added when they are essential for clarification or for dimensioning purposes. In Figure 9, there are dotted lines in the hatched region indicated by the arrow. These dotted lines are important as in their absence the drawing does not contain complete information about the wedge and holes in the part. Therefore, a loop in a sectioned view may represent many fundamental loops in a regular view. This implies that merging of loops is required before checking for *onto-formation*. Due to the same reason, there may be only matching edges in the sectional view for a loop in the regular view.
3. In a half sectional view, hidden lines can be left out in the unsectioned half of the view, because they would only repeat the inside features already revealed in the sectioned half. Therefore, in the sectional view there may not be matching loops for loops in a regular view. Only matching edges of vertices in the loop may be available.

As observed above there may be many dotted lines missing in sectional views and in half sections there may be many edges unconnected at one end. All this leads to less number of loops in a view. The main step of the volume-based algorithm described above is the construction of *onto-formations*. This is possible when there is a matching loop for a loop, in every view of the drawing. As seen from the discussion about sectional views, the number of loops formed are less in a sectional view, resulting in less number of *onto-formations*. Therefore obtaining all the ele-

Figure 10. Merging of loops[11]

mentary solids is not possible with the existing. The proposed algorithm uses the observations made regarding sectional views above to identify the additional onto-formations as follows.

1. Form *onto-formations* and generate elemental solids corresponding to them.
2. In some drawings *onto-formation* can be made by merging some loops in a view. The resulting loop is also candidate for sweep as observed in item 2 in section 4.1. In the front view of Figure 10, loop{1,2,3,4,1} does not have a matching loop in the side view, but if three loops in side view i.e. loop{5,6,7,8,5}, loop{7,8,9,10,7}, and loop{9,10,11,12,9} are merged to form the loop{5,6,11,12,5}, then this merged loop with loop{1,2,3,4,1} in front view (sectional view) and loop{13,14,15,16,13} in top view becomes an *onto-formation*.
3. If a loop is not disjoint because it is connected by dotted edges only, then consider it as a disjoint loop. In Figure 11, the loop marked with arrow in the front view is a solid loop (no edge in loop is dotted) and the only edges touching this loop are two dotted edges. Clearly this loop and these two dotted edges are not at same level (depth wise) as loop is solid and edges are dotted, so such loops are considered as *disjoint* loops for the rest of algorithm.
4. In drawings with more than two views, if in one view there is a disjoint loop that has no *onto-formation* then check onto-matching loops for this disjoint loop in one more view only and sweep it. A disjoint loop in a view clearly indicates a hole or protrusion structure in the object. For a disjoint loop, an onto-matching loop in only one other view is sufficient to form an

Figure 11. Extra disjoint loop[11]

onto-formation and the pair of loops are candidates for elemental sweep solids. In Figure 8(a) the loop marked with arrow in the top view matches with loop {1,2,3,4,1} in the front view. But there is no matching loop in side view corresponding to them. In such cases the elemental solid is obtained by sweeping of matching loops from two views only.

It may be noted that steps 3 & 4 are also true when addressing regular views.

5. If for a disjoint loop there is no onto-matching loop in the sectional view, then find matching edges or edge (in case of half sectional view) and sweep it along the edges. This will result in an onto-formation by the items 2 and 3 in section 4.1. In Figure 6 the loop marked with arrow in the top view has no matching loop in the other view. In such cases first the two extreme vertices (with respect to common co-ordinate axis) in the disjoint loop are found. Here vertices 5 and 6 are extreme vertices for the loop. For each of these vertices matching edges are found. Edge 1-2 and edge 3-4 in the front view, matches with vertex 5 and vertex 6 respectively. If matching edges are of equal span then disjoint loop is swept along these edges to obtain an elemental solid. In case of half section there may not be a matching edge for one of the extreme vertices. For example in Figure 4, the loop{3,4,5,6,3} is a disjoint loop and vertices 3 (or 6), and 4 (or 5) respectively can be considered as extreme vertices. But there is no matching edge for vertex 3 (or 6) in the front view. The loop is swept along the edge 1-2, which is a matching edge of the other extreme vertex 4 (or 5).

6. Validate solids obtained with respect to input views. If validation is successful then go to step 8.
7. If validation is unsuccessful then obtain *bound on solid* and then go to step 8.

Onto-formations allow identification of swept volumes corresponding to fundamental loops. These would usually be in the interior as outer loop would normally not be a fundamental loop. Therefore we need to find the elemental solid corresponding to entities in the outer loop. Straight forward way is to find the *bound on solid*. However some of the entities on the outer loop would have been identified as *onto-formations* and these have to be accounted for. If the entities on the outer loop that are part of the onto-formations correspond to protrusions on the object, the outer loop has to be modified to identify the bound on solid correctly.

The procedure to do this is as follows :

Modify outer loop

As mentioned above, an elemental solid corresponds to a protrusion or a depression with respect to the part. In order to determine if the elemental solid is a protrusion or a depression the following procedure is used. The elemental solid is projected back on the view planes to obtain projected loops. For example if a cylinder is projected then its projection is rectangular loop in one view and a circular loop in the other. These loops are compared with the outer loop of the corresponding input view to decide whether it is a protrusion or depression. The loop obtained by projection is considered as DOTTED if any edge in input view, which matches with the edge of the projection loop, is DOTTED otherwise it is taken as SOLID. The loop obtained by projection is referred to as *target loop*.

Figure 12. Modify outer loop

In Figure 12(a), the loop {1,2,3,4,1} is the *target loop* of an elemental solid and loop {1,2,3,4,5,6,7,8,1} is the outer loop in the view. So to find whether the *target loop* is a protrusion or depression, the outer loop is

traversed starting with a vertex which is not common with *target loop*. Only vertices with degree² greater than two need to be considered. Let 'V' be the first such vertex while traversing the outer loop. In this figure if the outer loop is traversed clockwise the first such vertex is vertex 1. Let the edges incident on 'V' be denoted by vectors 'a', 'b', and 'c'. The first edge which is not in the *target loop* is taken as vector 'a'. Here the edge 8-1 is vector 'a'. The next edge in the outer loop which is common to *target loop* is taken as vector 'b'. Here the edge 1-2 is vector 'b'. The other edge from the selected vertex 'V', which is member of *target loop* but not of outer loop, is taken as vector 'c'. Here the edge 1-4 is vector 'c'. Procedure is_Protrusion_Depression() takes as arguments three vectors 'a', 'b', 'c' and two loops *target loop* and outer loop, and returns whether the *target loop* corresponds to a protrusion or depression.

The algorithm is as follows.

```

Procedure is_Protrusion_Depression() {
  if all the vertices of target loop are on the outer loop
  then then
    return Protrusion
  else if 'a' and 'c' are collinear && target loop is
  SOLID then
    return Protrusion
  else if 'a' and 'b' are collinear then
    return Depression
  else if target loop is DOTTED then
    return Depression
  else if target loop is SOLID then
    return Protrusion
  end if
}

```

A similar but more restricted algorithm has been used by Meeran & Pratt [8] to identify features.

If the function is_Protrusion_Depression() returns protrusion the outer loop is modified by merging it with *target loop*. The procedure of merging of two loops is as follows.

Merging of two loops : Two adjacent loops in a view can be merged to form a new loop. Merge operation is a symmetric difference operation, based on the edges present in the loops. If loop ${}^G L_1$ and loop ${}^G L_2$ are adjacent loops then merging these two loops will result in a loop ${}^G L_n$, that will have the following edges

$${}^G L_n E = ({}^G L_1 E \cup {}^G L_2 E) - ({}^G L_1 E \cap {}^G L_2 E)$$

where ${}^G L_1 E \subset {}^G L_1$, ${}^G L_2 E \subset {}^G L_2$, and ${}^G L_n E \subset {}^G L_n$

²degree of a vertex is defined as the number of edges incident on the vertex

If a loop is considered as a face then merging of two loops can result in either union or difference of the two. For example in view 2 in Figure 7, merging of loop {3,4,5,6,3} with loop {1,2,3,6,7,8,1} will result in loop {1,2,3,4,5,6,7,8,1} which is union of the two. While in view 1, the merging of loop {3,4,5,6,7,8,3} with loop {5,6,9,10} will give the difference of the two i.e., loop {3,4,5,10,9,6,7,8,3}.

The overall Procedure `Modify_outer_loop` works as follows.

```

Procedure Modify_outer_loop() {
  if is_Protrusion_Depression() returns Protrusion
  then
    Merge outer loop and target loop and return
    Merged loop
  end if
  if Merged loop != NULL then
    outer loop = Merged loop
  end if
}

```

If there is a through hole in an object for example a cylindrical hole in a cube, then its projection will be as shown in Figure 12(b). For this case (through hole) the function `is_Protrusion_Depression()` will return protrusion, because here loop{2,3,6,7,2} is the *target loop* and loop{1,2,3,4,5,,6,7,8,1} is the *outer loop*. As all the vertices of *target loop* fall on the *outer loop*, the first 'if' condition will return protrusion although it is a depression. But this does not affect the algorithm adversely. In this case when *target loop* and *outer loop* are merged the result will be two loops as shown in Figure 12(c). The Merging of two loops function returns NULL when the resultant merged loop is not a single loop. Therefore in function `Modify_outer_loop` the *outer loop* remains unchanged.

8. Apply *boolean operations* on all obtained solids to get the solution solid.

5 Results and Discussion

The algorithm has been implemented on both Linux(Red Hat) and Windows platforms running on a Pentium IV machine. For geometric operations, ACIS 8.0 (www.spatial.com) geometric kernel has been used.

Example 1

Consider the full sectional drawing of Figure 10. There are two *onto-formations* possible as shown in Figure 13(a). The loops labelled 1 (in each front, side and top view) form an *onto-formation* and the loops labelled 2 in each view form another *onto-formation*. From these *onto-formations* two

Figure 13. (a) Onto-formations of Figure 10. (b) Onto-formation by merging of loops

Figure 14. Solution solid of Figure 10

cylindrical elemental solids are obtained. By the second step of the algorithm another *onto-formation* is obtained by merging three loops in side view of Figure 13(b) into a single loop, as explained in section 4. This *onto-formation* gives a rectangular elemental solid. All these elemental solids are shown in Figure 14(a). No other elemental solids are possible here. After passing other steps of algorithm the *bound on solid* is obtained as shown in Figure 14(b). *Boolean operations* on all obtained solids gives the solution solid of Figure 14(c). The '+' and '-' signs with arrows indicate the 'union' and 'difference' boolean operation respectively.

Example 2

Figure 15. Matching loops of Figure 8(a)

Consider the full sectional drawing of Figure 8(a). There is one *onto-formation* formed by loops labelled 1 in each view of Figure 15. For the disjoint loop of top view marked with number 2, there is no *onto-formation*. By the fourth step of algorithm (section 4) the two loops labelled 2 in the figure become candidates for obtaining elemental solid. Figure 16(a) shows the two elemental solids obtained and Figure 16(b) shows the *bound on solid*. The solution solid is obtained after boolean operations as shown in Figure 16(c).

Example 3

Figure 17(a) shows a half sectional drawing. There is one *onto-formation* possible here between loops marked with arrow 1, in Figure 17(b). For the three disjoint loops (one rectangular and two circular) in top view there is no matching loop in the other view. By the fifth step of the algorithm (section 4) matching edges are found for these loops. For the rectangular dotted loop the two edges marked with arrow 2 are found. For the inner circular loop the only matching edge is marked with arrow 3 and for the

Figure 16. Solution solid of Figure 8(a)

other circular loop the matching edge is marked with arrow 4. Elemental solids are obtained by sweeping these three loops along their matching edges. The elemental solids are shown in Figure 19(a). After validation, the elemental solids are projected back on view planes, to determine whether they are protrusions or depressions in the function *Modify_outer_loop*. Figure 18(a) shows projections of two cylindrical objects and the outer loop in both views. Clearly both the projections form depressions, so the outer loop remains unchanged.

Figure 17. (a) A half section drawing[11] (b) Matchings for the drawing

Figure 18(b) shows projections of two rectangular solids and outer loops in both view. Here both the projections in front view are identified as protrusions, so the outer loop gets modified. The modified outer loops are shown in Figure 18(c). The *bound on solid* obtained by these modified outer loops is shown in Figure 19(b). Figure 19(c)

Figure 18. Modification of outer loop corresponding to Figure 17(a).

shows the solution solid.

Figure 19. Corresponding to Figure 17, (a) Elemental solids (b) Bound on solid (c) Solution solid

Figure 20 shows results corresponding to Figure 6 and Figure 11.

5.1 Discussion

For all the results shown the running time of the construction algorithm is in the order of seconds. For the algorithm it is necessary to know the type of sectional view that

Figure 20. (a) Solution solid of offset section of Figure 6 (b) Solution solid of Figure 11

has been input. At present we are manually incorporating this information in the input file to the program. The information about hatched loops is available in the .DXF file. Sometimes there are free hand curves/polylines in sections (Figure 4). Such lines may be drawn as spline curves in the drawing file (.DXF file), but the processing of a spline curve is not yet incorporated in the algorithm.

There are examples for which correct elemental solid can not be obtained by sweep operations we are using. For example in Figure 21, the two small loops marked with arrows in the front view have no matching loop in the top view. The elemental solids corresponding to these loops are not obtainable here. An approach based on decomposing existing loops by introducing proper edges in the views is being explored to form loops that would allow the construction of the correct elemental solids. The edges introduced would correspond to those edges that are destroyed during the boolean union or subtraction of the elementary solids.

At present we obtain elemental solids by sweeping a planar loop, perpendicular to its plane. The range of algorithm can be increased by using variational sweep.

6 Conclusion

A volume-based approach to construct 3D solid models from 2D views has been described to handle sectional views. The approach builds upon an existing volume-based approach. The presence of sectional views prevents identification of some loops corresponding to elemental solids used by the volume-based approach. Algorithm has been developed to identify the additional elementary solids in the presence of sectional views. Drawing conventions pertaining to sectional views have been used to identify the loops and subsequently the elementary solids. The algorithm has been implemented and tested on full sections, half sections and

Figure 21. A half sectional drawing

offset sections. While its extension to broken out sections is straightforward, handling other sections like removed and revolved sections requires further research.

References

- [1] B. Aldefeld. On automatic recognition of 3D structures from 2D representations. *Computer-Aided Design*, 15(2):59–64, 1983.
- [2] B. Aldefeld and H. Richter. Semiautomatic three-dimensional interpretation of line drawings. *Computer-Aided Design*, 8(4):371–380, 1984.
- [3] H. Bin. Inputting constructive solid geometry representations directly from 2-D orthographic engineering drawings. *Computer-Aided Design*, 18(3):147–155, 1986.
- [4] D. Dori and K. Tombre. From engineering drawings to 3D cad models : are we ready now ? *Computer-Aided Design*, 27(4):243–254, 1995.
- [5] W. Geng, J. Wang, and Y. Zhang. Embedding visual cognition in 3D reconstruction from multi-view engineering drawings. *Computer-Aided Design*, 34(4):321–336, 2002.
- [6] S. X. Liu, S. M. Hu, Y. J. Chen, and J. G. Sun. Reconstruction of curved solids from engineering drawings. *Computer-Aided Design*, 33(14):1059–1072, December 2001.
- [7] D. B. Lysak, P. M. Devaux, and R. Kasturi. View labeling for automated interpretation of engineering drawings. *Pattern Recognition*, 28(3):393–407, 1995.
- [8] S. Meeran and M. J. Pratt. Automated feature recognition from 2D drawings. *Computer-Aided Design*, 25(1):7–17, 1993.
- [9] S. S. P. Shum, W. S. Lau, M. M. F. Yuen, and K. M. Yu. Solid reconstruction from orthographic views using 2-stage extrusion. *Computer-Aided Design*, 33(1):91–102, 2001.
- [10] S. Soni and B. Gurumoorthy. Handling solids of revolution in volume-based construction of solid models from orthographic views. *Journal of Computing and Information Science in Engineering*, 3:250–259, September 2003.
- [11] I. S. Vyshnepolsky. *Engineering Drawing*. Mir Publishers, Moscow, 1985.
- [12] P. Wallach. *Metric Drafting*. Glencoe Publishing Co., California, 1979.
- [13] M. A. Wesley and G. Markowsky. Fleshing out projections. *IBM Journal of Research & Development*, 25(6):934–954, 1981.
- [14] Q. W. Yan, C. L. P. Chen, and Z. Tang. Efficient algorithm for the reconstruction of 3D objects from orthographic projections. *Computer-Aided Design*, 26(9):699–717, 1994.