COP290 Assignment 1 Report

Saransh Verma 2016CS10326
Tarun Kumar Yadav 2016CS10359

January 23, 2018

1 Introduction

Engineering drawings are used widely and play an essential role in traditional engineering. Three-view engineering drawings provide a means of easily describing the exact shape of any material object. Most existing products are presented in the form of engineering drawings in blueprint or in 2D CAD system. To manufacture these products the corresponding 3D objects have to be reconstructed from 2D engineering drawings.

This part of the project aims to produce mathematical model for obtaining three orthographic projections of 3D object and reconstructing 3D object from its orthographic projections.

2 Representation of 3D Object

We will represent a 3D solid by polygon tables representation. It is the specification of polygon surfaces using vertex coordinates and other attributes. Shown below is a sample representation of a 3D solid with polygon table.

VERTEX TABLE
V1 : x_1, y_1, z_1
V2 : x_2, y_2, z_2
V3 : x_3, y_3, z_3
V4 : x_4, y_4, z_4
V5 : x_5, y_5, z_5

EDGE TABLE
E1 : V_1, V_2
E2 : V_2, V_3
E3 : V_3, V_1
E4 : V_3, V_4
E5 : V_4, V_5
E6 : V_5, V_1

SURFACE TABLE
S1 : E_1, E_2, E_3
S2 : E_3, E_4, E_5, E_6

3 Assumptions

1. All the objects are assumed to be Polyhedron.
2. For reconstruction the orthographic projection should be on 3-perpendicular planes. If the time permits we may remove it.

3. There can only be one possible 3D solid satisfying the orthographic projections.

4 Transforming Points in Space

Since taking projections of lines and planes is equivalent to joining projections of its end points (3 non-collinear points) in this section we consider just a point and show how it can be rotated, projected and transformed.

4.1 Shear and Reflection of points

Any transformation of points in 2-D space can be thought of matrix multiplication i.e. given a set of points described in matrix X any transformation (rotation, translation, reflection, etc.) can be captured in a matrix T such that resulting matrix X' = X× T.
In general:

1. For a 2D point:

$$\begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \times \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} ax + cy & cx + dy \end{bmatrix}$$

2. and for a 3D object:

$$\begin{bmatrix} x^* & y^* & z^* \end{bmatrix} = \begin{bmatrix} x & y & z \end{bmatrix} \times \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = \begin{bmatrix} ax + dy + gz & bx + ey + hz & cx + fy + iz \end{bmatrix}$$

Common Cases of shear and reflection

1. 2D: When b=c=0 then

$$\begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \times \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix} = \begin{bmatrix} ax & dy \end{bmatrix}$$

2. 2D:When a=1 and d=-1 then points are reflected about x-axis

$$\begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} x & -y \end{bmatrix}$$

3. 2D:When a=-1 and d=1 then points are reflected about y-axis

$$\begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \times \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} -x & y \end{bmatrix}$$

4. 2D: Similarly when a=d=-1 then points are reflected about origin

$$\begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \times \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} -x & -y \end{bmatrix}$$

5. 3D: The multiplied matrix scales objects a, e and j times in x, y and z axis respectively

$$\begin{bmatrix} x^* & y^* & z^* \end{bmatrix} = \begin{bmatrix} x & y & z \end{bmatrix} \times \begin{bmatrix} a & 0 & 0 \\ 0 & e & 0 \\ 0 & 0 & j \end{bmatrix} = \begin{bmatrix} ax & ey & jz \end{bmatrix}$$

6. 3D: The multiplied matrix shears object in all directions

$$\begin{bmatrix} x^* & y^* & z^* \end{bmatrix} = \begin{bmatrix} x & y & z \end{bmatrix} \times \begin{bmatrix} 1 & b & c \\ d & 1 & f \\ g & h & 1 \end{bmatrix} = \begin{bmatrix} x + dy + gz & bx + y + hz & cx + fy + z \end{bmatrix}$$

7. 3D: Reflection about x-y plane corresponds to negating the z-coordinate

$$\begin{bmatrix} x^* & y^* & z^* \end{bmatrix} = \begin{bmatrix} x & y & z \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} = \begin{bmatrix} x & y & -z \end{bmatrix}$$

8. 3D: Similarly reflection matrix about y-z and x-z are

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4.2 Rotation of Points

1. Consider a point (x ,y) making an angle ϕ now assume that after rotation it makes an angle $\phi + \theta$. Writing (x, y) in polar form as (rcosϕ, rsinϕ) then after rotation (x', y') = (rcos(ϕ+θ), rsin(ϕ+θ)).

$$\begin{gathered} \text{P} = [\ \text{x, y}\ ] = [\ \text{rcos}\phi, \text{rsin}\phi\ ] \\ \text{P'} = [\text{x', y'}\ ] = [\ \text{rcos}(\phi + \theta), \text{rsin}(\phi + \theta)\ ] \\ \text{P'} = [\ \text{r}(\text{cos}\phi\text{cos}\theta - \text{sin}\phi\text{sin}\theta), \text{r}(\text{sin}\phi\text{cos}\theta + \text{sin}\theta\text{cos}\phi)] \\ \text{P'} = [\text{x', y'}] = [\text{xcos}\theta - \text{ysin}\theta\ , \text{xsin}\theta + \text{ycos}\theta\ ] \end{gathered}$$

$$\begin{bmatrix} X \end{bmatrix} \begin{bmatrix} T \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

Some common cases of Rotation

(a) Rotation of 90° counter-clockwise about the origin

$$\begin{bmatrix} T \end{bmatrix} = \begin{bmatrix} \cos 90^\circ & \sin 90^\circ \\ -\sin 90^\circ & \cos 90^\circ \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

(b) Rotation of 180° counter-clockwise about the origin

$$\begin{bmatrix} T \end{bmatrix} = \begin{bmatrix} \cos 180^\circ & \sin 180^\circ \\ -\sin 180^\circ & \cos 180^\circ \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

(c) Rotation of 270° counter-clockwise about the origin

$$[T] = \begin{bmatrix} \cos 270^\circ & \sin 270^\circ \\ -\sin 270^\circ & \cos 270^\circ \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

2. By extending the rotation matrix of 2D plane rotation matrix about z-axis by angle θ is given by

$$[T_{\mathrm{RZ}}] = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Similarly, rotation matrix about x-axis and y-axis is

$$[T_{\mathrm{RX}}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[T_{\mathrm{RY}}] = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4.3 Translation of Points

A 4*4 matrix needs to be introduced to translate points and scale 3D objects overall i.e.

$$\begin{bmatrix} a & b & c & j \\ d & e & f & k \\ g & h & i & l \\ m & n & o & p \end{bmatrix}$$

in which, m, n and o correspond to translation along x, y and z axis while p gives the overall scaling of object.

For Example, if we need to shift point (x_0, y_0, z_0) to origin then:

$$\begin{bmatrix} x^* & y^* & z^* & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_0 & -y_0 & -z_0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$$

4.4 Combining Multiple Transformations

A combined matrix capturing the effect of multiple transformation can be obtained by multiplying individual matrices in order.

For Example: To rotate an object about an axis parallel to z-axis

1. Translate the axis to coincide with z-axis
2. Rotate the object by required angle
3. Translate the axis back to original position

$$[T_1] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_0 & -y_0 & -z_0 & 1 \end{bmatrix} \quad [T_2] = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [T_3] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x_0 & y_0 & z_0 & 1 \end{bmatrix}$$

Thus a object described by [T] transforms to [T][T_0][T_1][T_2]

5 Orthographic Projections

Orthographic projection is a means to represent 3D object in 2D, it is a form of parallel projection in which all light rays are perpendicular to projection plane.

5.1 Orthographic Projection Matrix

Orthographic projection can be represented as the product of Coordinate matrix and Transformation Matrix.
Transformation Matrices for some common projections are

1. For projection on xy plane i.e z = 0 transformation matrix T_Z is

$$[T_Z] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2. For projection on xz plane i.e x = 0 transformation matrix T_Y is

$$[T_Y] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3. For projection on yz plane i.e x = 0 transformation matrix T_X is

$$[T_X] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5.2 Projection on a General Plane

Now let say we want to find projection of some point $\vec{v}$ on the plane Y = ax + by +cz = 0
Preposition: Let a point in space be represented by a matrix V and a plane ax + by + cz = 0 has basis vectors $\vec{b_1}$ = [b_{1x}, b_{1y}, b_{1z}] and $\vec{b_2}$ = [b_{2x}, b_{2y}, b_{2z}] then the projection P of V on plane ax + by + cz = 0 is given by

$$P = A(A^T A)^{-1} A^T V \tag{1}$$

$$where A = \begin{bmatrix} b_{1x} & b_{2x} \\ b_{1y} & b_{2y} \\ b_{1z} & b_{2z} \end{bmatrix}$$

Proof. Any point on plane can be represented by matrix product Ay where $\vec{y}$ = [y_1 , y_2] and y_1 ,$y_2 \in \mathbb{R}$. Now we can represent $\vec{v}$ as the sum of its component parallel and perpendicular to the plane

$$\vec{v} = \vec{v_{\parallel}} + \vec{v_{\perp}}$$

$$\vec{v} - \vec{v_{\parallel}} = \vec{v_{\perp}}$$

let C(A) denote the column space of A and N(A) denote the null space of A then C(A) = Y
then $Y^{\perp}$ = N(A^T) therefore

$$A^T(\vec{v} - \vec{v_{\parallel}}) = 0$$

$$A^T\vec{v} - A^T\vec{v_{\parallel}} = 0$$

now we have $\vec{v_{\parallel}}$ = Ay

$$A^T\vec{v} - A^TAy = 0$$

$$y = (A^TA)^{-1}A^T\vec{v}$$

but as we previously stated $\vec{v_{\parallel}}$ = Ay therefore

$$\vec{v_{\parallel}} = A(A^TA)^{-1}A^T\vec{v}$$

hence

$$P = A(A^TA)^{-1}A^TV \tag{2}$$

hence the transformation matrix for a plane as described previously is

$$T = (A(A^TA)^{-1}A^T)^T \tag{3}$$

6 Reconstuction of 3D object from Orthographic Views

The process of reconstruction is done in the following steps:

1. Convert 2D edges and vertices into 3D edges and vertices.
2. Generate planes from 3D vertex and edge list.
3. Handle dashed line information to remove redundant edges.
4. Construct planar graphs for each plane; search for basic loops in each planar graph and generate face loops.
5. Construct body loops from face loops.
6. Combine body loops to form candidate objects and find the exact solutions.

6.1 Wireframe Construction

Let $v_list(e)$ be a 2D vertex list, in which each vertex consists of a coordinate value (x,y), where (e) can be f, t or s,

Algorithm:

1. Generate 3D vertices by choosing a vertex from each projection's $v - list(e)$, and compare the coordinates of the three vertices on their common coordinate axes. The comparison stops if no more 3D vertices are created.

2. Generate 3D edges by selecting a pair of vertices (assume that there is an edge between the two vertices). If the three projections of this hypothetical edge are all in their corresponding $e - lists(e)$, a $p - edge$ is created; otherwise, select another pair of vertices, and repeat this hypothesis-and-verification process until no more 3D edges can be created.

The above algoritm with generate overlapping edges as in the following object, which originally as shown:

Figure 1: Generated wireframe

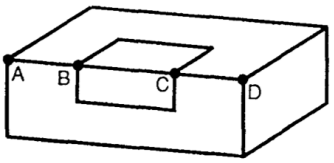
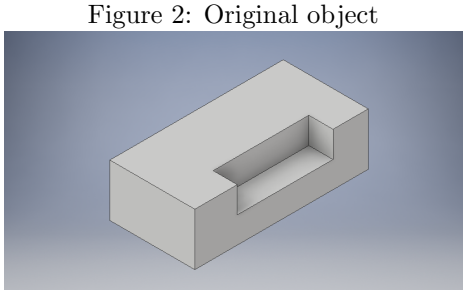


Figure 2: Original object



In order to remove such overlapping edges such as:

we follow the following algorithm:

Algorithm:

1. Initially, mark all the edges in $p - elist$ 'unexamined'.
2. Select an 'unexamined' edge e_i from $p - elist$, and set E $<-$ (e_i). If all the edges have been examined, procedure stops.
3. For every edge $e_j \in p - elist$, $e_j \neq e_i$, if e_j and e_i are collinear and e_j overlaps with one edge in E, then set E $<-$ E $\cup$ (e_j).
4. If there is only one edge in E, then no other edges overlap with the edge in E. Mark this edge 'examined', and go to 2; otherwise, continue.
5. If there is more than one edge in E, remove all the edges in E from $p-elist$, and sort all the vertices of the edges in E according to their coordinate values. Let $v_1 v_2 ... v_k$ be the sorted vertex sequence. Add edges (v_j, v_{j+1}), j= 1.... ,k-1, into $p - elist$, and mark them 'examined'. Go to Step 2.

Still some pathological edges such as BC remain and need to be removed for which we follow the following procedure:

Procedure: Let $\rho(p_vertex)$ represent the number of $p - edges$ shared at $p - vertex$.

1. Delete an isolated p-vertex if ρ(p-vertex) = 0.
2. If ρ(p-vertex)=1, remove the dangling edge from p-elist at the p-vertex, and remove the p-vertex from p-vlist.
3. If ρ(p-vertex)=2, and two edges are not collinear, delete both p-vertex and two adjacent edges at this p-vertex.
4. If ρ(p-vertex)=3, and two of three edges are collinear, delete the p-vertex and the third edge. Merge these two collinear edges.
5. If ρ(p-vertex)=3, and three edges are coplanar, but any two edges are not collinear, delete p-vertex and these three edges.
6. If ρ(p-vertex) >=4, the four edges are coplanar, and only two edges are collinear, then merge two collinear edges, and delete p-vertex and two noncollinear edges.
7. If two pairs of edges are collinear or all the edges are non- collinear, then delete p-vertex and all the 4 edges.

6.2 Generating Planes

To generate the surfaces of the object we need to know the planes of the surfaces and hence we use the following algorithm to generate planar graphs.

Algorithm:

1. Record the adjacent edges for each vertex in p-vlist.
2. (Construct Planes) A plane can be determined by any two non-collinear adjacent edges at this vertex. For each v_i in p-vlist, construct a plane determined by e_1 and e_2 ,where e_1=(v_i,v_j),and e2 = (v_i, v_k), i $\neq$ k.
3. (Hypothesize the outer normal vector of a plane) Since we do not know the outer side of the plane that was constructed above, the outer normal vector of each plane cannot be determined. The hypothetical normal vector of a plane is a vector starting from the origin, and perpendicularly penetrating the plane. The side from which the hypothetical normal vector leaves the plane is the positive side of the plane, and the other side is the negative side.
4. (Eliminate a duplicated plane) Delete plane j if there exists a plane i, i$\neq$j, such that the distance $D_j <= \xi$ (ξ is a tolerance number, say $10^(-5)$), where D_j is approximated by (((a$_i$ - a$_j$)^2 +(($b_i - b_j$)^2 + (($c_i - c_j$)^2 + (($d_i - d_j$)^2)$^(1/2)$, (a$_i$,b$_i$,c$_i$,d$_i$) and (a$_j$,b$_j$,c$_j$,d$_j$) are the coefficients of the corresponding plane equations.
5. (Search for all the edges on the plane) For every p-edge in the p-elist, compute the distances of its two endpoints D_1 and D_2 to the plane. If $D_1 <= \xi$ and $D_2 <= \xi$, p-edge is on the plane. The p-edge on the plane is used to generate face loops.

6.3 Removing invalid edges

We use the dashed-line information to remove invalid 3D edges generated from 2D projections.

Procedure:

1. From the input data, we know the type of projection line of each 2D edge. If its frontal/top/side projection is a broken line, there should be at least one planar graph that blocks this p-edge when viewing from the direction of + Oy/ +Oz/ +Ox. If no such planar graph exists, this p-edge must be removed.
2. If every p-edge has been examined, then go to Step 3; otherwise, select another p-edge, and go to Step 1.
3. The removal of invalid edges may create new pathological edges and vertices. Therefore, the pathological edge removal procedure is applied to purge further pathological edges and vertices.

6.4 Generating Face Loops

Until now we have found all the planes of the object, now we find loops on faces, first basic and then combine them to get full face loops. This algoritm generates all the face loop on a planar graph.

Algorithm:

1. (Construct vertex-edge adjacent table for the input planar graph) For each vertex in the planar graph, construct a vertex-edge list (v, ne, SE), where v∈p-vlist, ne is the number of adjacent edges at v, and SE= e_1, e_2.....,e_{ne} is a list of all the adjacent edges at v on the planar graph. The vertex-edge table adj-tab is a collection of all the vertex-edge lists.

2. Find the ordered adjacent edge of a vertex.

   (a) Select an entry (v, ne, SE) from adj-tab.

   (b) Let OE(v) be a set of ordered adjacent edges at v. The edges in OE(v) are arranged in ascending order on the basis of the clockwise angle with respect to a selected edge e_k. The clockwise angle is calculated around the hypothetical outer normal vector of the plane.

   (c) If all the entries have been processed, go to Step 3; otherwise, go to Step 2.b.

3. (Find all the basic loops within a planar graph) Each edge (v_i,v_j) has two alternative directions in which to travel: from v_i to v_j, and vice versa. Let E_c denote a selected current edge, and let V_{start} and V_{end}, denote its starting vertex and ending vertex of a basic loop, respectively. Initially, let the basic-loop set ψ <- ϕ, and a basic-loop edge set L<-ϕ.

   (a) Select an edge e=(v_i, v_j) in the planar graph, and let E_c<-e. If all the edges have been selected in this planar graph, then go to Step 4; otherwise, L<-L∪E_c, and continue.

   (b) Select the adjacent edge of E_c at V_{start} from OE(V_{start}).Suppose that OE(V_{start})= (e_1..... E_c, e_c',,...,e_{ne}, e_1). Then, e_c'= (v_k,v_m) is the first adjacent edge of E_c at V_{start}. Set L = L∪ e_c'.

   (c) If E_c has visited in both directions in the current basic loop L, then E_c is a bridge in the planar graph, and is removed. Let L=(e_1..... e_j,E_c,e_k,...,e_l, E_c)

   (d) If E_c has visited in both directions in the current basic loop L, then E_c is a bridge in the planar graph, and is removed. Let L=(e_1..... e_j,E_c,e_k,...,e_{cl}, E_c). A basic loop is formed by the edges from e_k to e_l in L. Set ψ <-$\psi \cup (L)$

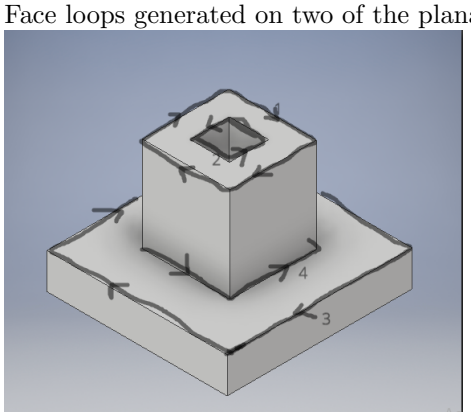
   (e) If V_{start}=V_{end},a basic loop is discovered; set $\psi = \psi \cup (L)$,and let L=ϕ, and go to Step 3.a; otherwise, go to Step F3.b.

4. (Identify inclusion relationship between the basic loops on a planar graph) For every pair of loops L_i and L_j, if L_i includes L_j, then include(i,j) = 1, else include (i,j)= 0.

5. (Form face loops) Given the inclusion relationship between basic loops, If L_i does not include any loops, then L_i forms a face loop;
   Else if L_i includes L_j, then L_i is an outer loop, L_j is an inner loop, and L_i includes L_i directly;
   Else if loop L_i includes more than one loop, i.e. L_{j_1},, L_{j_2}. . . . . L_{j_n}, check whether the L_js are included by other basic loops.
   A face loop is formed from L_i and those basic loops included by L_i directly. In this face loop, L_i is the outer basic loop, and the others are inner basic loops.

For each planar graph we use the above algorithm to generate the face loops.

Figure 3: Face loops generated on two of the planar graphs



6.5 Generation of Body Loops

After generating the face loops of all the faces of solid, we use the following algorithm to generate the body loop of the object. Body loops can be viewed as a group of subobjects which do not have common interior points. They can share some vertices, edges or face loops.

1. (Initialization) A face loop $\Im_i$ has two sides. The side from which the hypothetical normal vector comes is called the positive side, denoted as $+\Im_i$, while the other side is called the negative side, denoted as $-\Im_i$.Set each side of $\Im_i$ as 'unused' by setting χ(i,+)*<-0, χ(i,-)*<-0, 1≤i≤m. Let the set of face loops S($\Im$)<-ϕ.

2. (Select a starting face loop and its side selection.) Select an $\Im_j$ as a starting face loop to initiate a body loop search. If χ(j,+)=0, then set S($\Im$)<-S($\Im$)$\cup+\Im_j$ and χ(j, +)<-1, and mark $+\Im_j$ in S($\Im_j$) as 'unexpanded'; go to Step 3. If χ(j,-)=0, then set S($\Im$)<-S($\Im$)$\cup-\Im_j$ and χ(j,-)<-1, and mark $-\Im_j$ in S($\Im_j$) as 'unexpanded'.

3. (Select an 'unexpanded" face loop $\oplus\Im_k$ ($\oplus$ could be + or -) from S($\Im$), and compute the successive face loops of each edge on $\oplus\Im_k$'s boundary) Let the adjacent face loops at edge e on $\Im_k$ be $\Im_1$, $\Im_2$,..., $\Im_n$, n$\geq$2. The successive face loop of $\oplus\Im_j$ at edge e is $\oplus\Im_s$, 1$\leq$s$\leq$n and s$\neq$k, and $\Im_s$, needs the smallest rotating angle to coincide with $\oplus\Im_j$. Let α be the rotating angle between $\oplus\Im_k$ and $\oplus\Im_s$. Let $\mathbf{n_k}$ and $\mathbf{n_s}$, be the hypothetical normal vectors of $\Im_k$ and $\Im_s$, respectively. Let θ be the angle between $\mathbf{n_k}$ and $\mathbf{n_s}$.
Let a unit vector **e** be a vector along edge **e**, and satisfy the right-hand rule with $\mathbf{n_k}$. If $\oplus\Im_j$ is $+\Im_j$, its successive face loop at edge **e** can be computed with the following criteria. We find out the values of α and the side selection $\oplus$ for each adjacent face loop at edge **e** so that we choose the face loop with the smallest α value.

   (a) If $\mathbf{n_s} \times \mathbf{n_k}$ is in the same direction as **e**, and $+\Im_s$ is to the right of e, then $\alpha = \pi - \theta$; assign $+\Im_s$ to $\oplus\Im_s$

   (b) If $\mathbf{n_s} \times \mathbf{n_k}$ is in the opposite direction from **e**, and $+\Im_s$ is to the left of e, then $\alpha = \theta$; assign $-\Im_s$ to $\oplus\Im_s$

   (c) If $\mathbf{n_s} \times \mathbf{n_k}$ is in the same direction as **e**, and $+\Im_s$ is to the left of e, then $\alpha = 2\pi - \theta$; assign $-\Im_s$ to $\oplus\Im_s$

   (d) If $\mathbf{n_s} \times \mathbf{n_k}$ is in the opposite direction to **e**, and $+\Im_s$ is to the right of e, then $\alpha = \pi + \theta$; assign $+\Im_s$ to $\oplus\Im_s$

   If $+\Im_s \notin$S($\Im$), then S($\Im$)<-S($\Im$)$\cup+\Im_s$ and χ(s,+)<-1. Mark $+\Im_s$,'unexpanded' in S($\Im$). This side-selection procedure guarantees that either all the face loops in S($\Im$) are facing to the interior of the body loop being formed, or they are facing to the exterior of the body loop being formed. When the successive face loop of $\Im_k$ at each edge on $\Im_k$ has been determined using the above criteria, mark $+\Im_s$ 'expanded'. Go to Step 4. Similarly, if $\oplus\Im_s$ is $-\Im_s$, its successive face loop at edge **e** can be determined accordingly; go to Step 4.

4. Repeat Step 3 until there are no new face loops to be expanded in S($\Im$). The face loops in S($\Im$) may form a body loop. Go to Step 5.

5. (Check body-loop legality) A body loop must be closed by face loops, without dangling face loops. As for the current body loop in S($\Im$), if every edge involved in this body loop is shared by two and only two face loops, then the face loops in the current S($\Im$) form a valid body loop; store this closed body loop. Otherwise, face loops in S($\Im$) cannot form a closed body loop; go to Step B6 to initiate another search.

6. Set S ($\Im$) <- ϕ , and repeat Steps 1-5 until every face loop $\Im_i$, 1$\leq$i$\leq$m, had been visited on both sides.

The above algorithm generates all the body loops from the face loops. Some of the body loops generated above are unbounded, and are called outer body loops. The others are inner body loops, because they are bounded. The outer body loops are useless for constructing objects, and thus they are discarded. To classify body loops into inner or outer loops, we use the following steps:

1. For a body loop, select any two adjacent face loops $\oplus\Im_i$ and $\oplus\Im_j$ at their shared edge $\mathbf{e}$, and construct a ray l starting at the midpoint of e along the direction of
   $l = \frac{\oplus\mathbf{n_1}}{\mathbf{n_1}} + \frac{\oplus\mathbf{n_2}}{\mathbf{n_2}}$
   where $\oplus\mathbf{n_i}$ and $\oplus\mathbf{n_j}$ are the hypothetical normal vectors of $\oplus\Im_i$ and $\oplus\Im_j$, respectively, and $\oplus$ corresponds to their side selection.

2. Analyse the situations in which the line l intersects with the face loops of the body loop. If the number of intersecting points made by l and all the face loops on the body loop (excluding $\Im_i$ and $\Im_j$) is one or more, then the body loop is inner; otherwise, it is outer.

6.6 Checking consistency of object with three input views

First, a candidate object is projected onto the corresponding three projection planes (xOz, xOy, and yOz), and three new views are formed. Second, compare the three new orthographic views with the original input views. If they match each other completely, this object is a solution of the input views.
During the projection, a 3D edge may become a 2D vertex or a 2D line segment. We need to consider the following cases. If a 3D edge is not blocked by any planes, the projection of this 3D edge is a solid segment; otherwise, it is a broken-line segment. If a broken-line segment overlaps with a solid-line segment, the over-lapping sections should be solid. Therefore, 3D edges can be projected into a combination of several solid-line segments and several broken-line segments.
With the 2D line segments from the projection, we merge the adjacent line segments of the same type (solid or broken). Compare these 2D line segments with those in the input data files. If the corresponding line segments are fully overlapped and matched, the constructed object is a correct solution. Otherwise, it is not a solution of the input views.

References

1. Efficient algorithm for the reconstruction of 3D objects from orthographic projections Qing-Wen Yan, C L Philip Chen* and Zesheng Tangt

2. Images taken from Autodesk Inventor 2016 edition.

3. CS3162 Introduction to Computer Graphics Helena Wong, 2001 : Chapter 9 Notes.