Mathematical Model for an Engineering Drawing Software Package

Pratyush Maini and Pranav Baurasia
(2016CS10412) , (2016CS10369)

January 23, 2018

Abstract

We are designing a mathematical model for converting 3-D images of object into orthographic projections and vice versa, in order to be used for Computer Aided Design (CAD) implementations in the future. This includes allowing rotation and translation of 3D objects around the X-axis, Y-axis and Z-axis through matrix representation of vectors of the corners of the solid. First, we get the projection of corner points and connect them using straight lines wherever required to get the top view, front view and side view.

In the reverse engineering, we transform 2D orthographic projections into 3D models. The approach is a two step method where-in we first lift the 2D points into the 3D space to construct a wire-frame model, and then convert this 3D wire-frame into a solid object. The complexity, uniqueness and solveability of the problem is determined by the type of input data, such as labeled and unlabeled projections and many other aspects that we see through the course of this document.

1 Introduction

The given document focuses on the conversion of polyhedral objects from 3D to 2D projections and vice versa.

Definition 1:
In geometry, a **polyhedron** is a solid in three dimensions, with flat polygonal faces, straight edges and sharp corners or vertices.

Definition 2:
An **Orthographic projection** is a means of representing a three-dimensional object in two dimensions through parallel projections, in which all the projection lines are orthogonal to the projection plane, resulting in every plane of the scene appearing in affine transformation on the viewing surface.

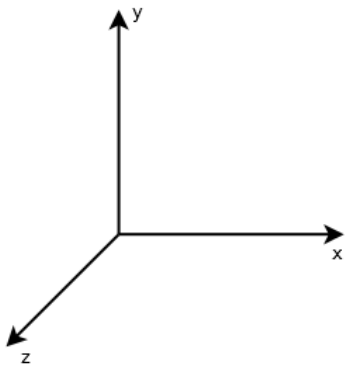
Over the past few decades, with accelerating Computer Aided Design technologies, the focus on shifting sophisticated projection and reconstruction algorithms into computationally inexpensive techniques has been growing.

We will be working on the problem of polyhedral objects, in order to accelerate the process of digitization of Machine Drawings or conversely provide the user an environment to explore various features of a 3D interface, that will also be extended to the process of computationally carrying out orthographic projections, by reducing the human intensive intervention.

We have closely followed the paper A Matrix-Based Approach to Reconstruction of 3D Objects from Three Orthographic Views, by Shi-Xia Lid, Shi-Min Hua, Chiew-Lan Taib and Jia-Guang Suna for any extended analysis on reconstruction of non polyhedral objects. For the discussion of polyhedral objects we either use generic algorithms or take cues from Fleshing Out Wire Frames by Markowsky and Wesley.

2 Understanding 3D objects

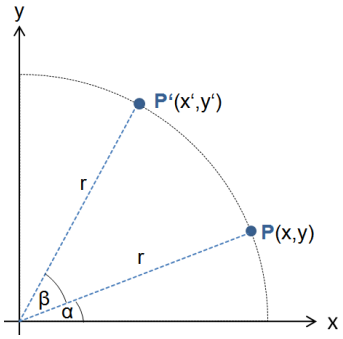
We will be working in a 3D space, with a right-handed base. This means that if you take your right hand and point your thumb in the direction of the X-axis, your ring- finger in the direction of the Y-axis, and the Z-axis will point straight up your hand-palm, as shown in the figure below:



In the given problem we assume that we have, available with us the co-ordinates of each vertex in the 3D plane, along with an adjacency list of all edges associated with a point available to us. Further, for each edge we assume an adjacency list of planes that are connected to an edge. Given this information, we will calculate the rotation or transformational effects to a 3D body, along with the projection on a 2D plane.

3 Derivation of rotation about X, Y or Z axis

3.1 2D rotation of an arbitrary point around the origin



P is a point on X-Y plane. Taking projections of point on x and y axes, we get -

$$x = r * \cos\alpha$$
$$y = r * \sin\alpha$$

and

$$x' = r * \cos\ (\alpha+\beta)$$
$$y' = r * \sin\ (\alpha+\beta)$$

Using trigonometric identities,

$$\cos(\alpha+\beta) = \cos\alpha * \cos\beta - \sin\alpha * \sin\beta$$
$$\sin(\alpha+\beta) = \sin\alpha * \cos\beta + \cos\alpha * \sin\beta$$

we get,

$$x' = r * \cos\ (\alpha+\beta)$$
$$x' = r * (\cos\ \alpha\ *\ \cos\ \beta\ -\ \sin\ \alpha\ *\ \sin\ \beta)$$
$$x' = r * \cos\ \alpha\ *\ \cos\ \beta\ -\ r^*\ \sin\ \alpha\ *\ \sin\ \beta$$
$$x' = r\cos\ \alpha\ *\ (\cos\ \beta)\ -\ r\ \sin\ \alpha\ *\ (\sin\ \beta)$$
$$x' = x * \cos\ \beta\ -\ y\ *\ \sin\ \beta$$

Similarly, the y will be transformed to y'

$$y' = r * \sin\ (\alpha+\beta)$$
$$y' = r * (\sin\ \alpha\ *\ \cos\ \beta\ +\ \cos\ \alpha\ *\ \sin\ \beta)$$
$$y' = r * \sin\ \alpha\ *\ \cos\ \beta\ +\ r^*\ \cos\ \alpha\ *\ \sin\ \beta$$
$$y' = r\sin\ \alpha\ *\ (\cos\ \beta)\ +\ r\ \cos\ \alpha\ *\ (\sin\ \beta)$$
$$y' = y * \cos\ \beta\ +\ x\ *\ \sin\ \beta$$

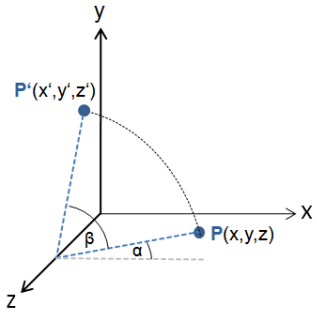
(x', y') will be the new coordinates of P, i.e, P' P' described in terms of P and acute angle β in matrix notation is given by -

$$\begin{bmatrix} x' & y' \end{bmatrix} = \begin{bmatrix} \cos\beta & \sin\beta \\ -\sin\beta & \cos\beta \end{bmatrix} \begin{bmatrix} x & y \end{bmatrix}$$

This is the rotation of point about Origin but if we want to make rotation about any point then we will first shift origin to the point of rotation and then rotate it by the equation and then shift the origin to its original place.

3.2 3D rotation about axis

In 3D, rotation will be defined by rotation axis and angle of rotation unlike in 2D, where only angle of rotation and point about which rotation takes place is necessary.If rotation is about a major axis then one component always remain same.



In the above figure, the z-component of the point remains same, so actually it's the same as rotating in the x-y plane which is same as the 2D case.

Rotation around the z-axis in matrix notation:

$$\begin{bmatrix} x' & y' & z' \end{bmatrix} = \begin{bmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x & y & z \end{bmatrix}$$

Similarly, the rotation about X by angle γ and Y by angle α axes are respectively given by-

$$\begin{bmatrix} x' & y' & z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & \sin\gamma \\ 0 & -\sin\gamma & \cos\gamma \end{bmatrix} \begin{bmatrix} x & y & z \end{bmatrix}$$

$$\begin{bmatrix} x' & y' & z' \end{bmatrix} = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix} \begin{bmatrix} x & y & z \end{bmatrix}$$

3.3 Rotation about X, Y and Z-axes

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{1}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

3.4 General rotations

The general rotation matrix R can be obtained by multiplying all the three matrices.

$$R = R_z(\gamma)\, R_y(\beta)\, R_x(\alpha)$$

represents an rotation whose angles are α, β and γ about axes X, Y and Z

The general rotation matrix depends on the order of rotation,i.e. , first the rotation is about x-axis then y-axis and finally z-axis.

Given a 3×3 rotation matrix R, a point on the rotation axis (given by the vector $\mathbf{u}$) must satisfy

$$R\mathbf{u} = \mathbf{u}$$

3.5 Determinant is 1 for rotation matrix

$$\text{Since } |R_x| = |R_y| = |R_z| = 1$$
$$\text{Therefore } |R| = |R_x||R_y||R_z|$$
$$|R| = 1$$

$$[\text{As } |R_xR_yR_z| = |R_x||R_y||R_z|]$$

4 Creation of orthographic projection from 3D view

The input will be isometric projection or 3D object model with all the corners labeled such that all the coordinates of them are known.
All the points being represented by the set $\{X_i\}_{i=1,2...N}$ where N is the number of corners in the solid(assuming polyhedral solid) and

$$X_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$$

Therefore, when projection of the solid is taken on a plane, each X_i will be mapped to only one point in the plane and that can be get by the following equations :-

For the front view, the plane is X-Y plane and the (x_i, y_i) coordinates will be given by

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix}$$

For the side view, the plane is Y-Z plane and the (y_i, z_i) coordinates will be given by

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} = \begin{pmatrix} y_i \\ z_i \end{pmatrix}$$

For the top view, the plane is X-Z plane, then the (x_i, z_i) coordinates will be given by

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} = \begin{pmatrix} x_i \\ z_i \end{pmatrix}$$

This can be done on all the points and we can get all points on any views.

Now, if there is an edge between two points then a line will be created in the view and if there is an edge behind a plane then a dotted line will be created for that edge.

4.1 Special Consideration: Dashed Lines

Orthographic projections contain dashed lines for edges that are not visible to the viewer from the direction of projection. We devise a simple algorithm for the same. The projection assumptions stated above give us an adjacency list of planes and lines. The equation of these planes is available to us in the form

$$P: ax + by + cz = d,$$

in the 3D space. Further, the equation of a line with direction vector *d= (l,m,n)* and passes through the vertex (x_1, y_1, z_1) is given by the formula:

$$\frac{x - x_1}{l} = \frac{y - y_1}{m} = \frac{z - z_1}{n}$$

If a line in 3D space intersects a point on the plane, then we will get another vertex point in the 3D space. Therefore, we can conclude that we have all distinct line segments available in the adjacency lists of the corresponding vertices in the 3D planes. Now, we simply need to compare the distance of any point on this line segment (in our case vertex points) with that of any plane. If the equation is of the same sign as that of the positive infinity for the projection direction, then it implies that this vertex lies in front of the plane P. Hence, the corresponding line segment in the 2D projection should form a solid line.
For example, consider the task of creating the front view orthographic projection, given the set of 3D co-ordinates. Therefore, the projection direction is from positive x-axis. So, now we compare the sign of a in the equation of plane P. Now, we compare the sign of a with the sign of

$$exp : ax_1 + by_1 + cz_1 - d$$

If the sign is opposite, then we will draw a dashed line in the front view for the lines leading from this point.

Corner Case : Now we consider the case in which a line is not completely overlapped by a plane, and therefore we need to take separate measures for this. After projecting the line on the 2D plane, we will obtain some extra intersection points that are not present in the original object. We retrace these points back into the 3D space and then re-apply the same algorithm for these points.

5 Reconstruction of 3-D objects given 2-D orthographic projections

5.1 Approach

We follow a matrix based approach to solve the reconstruction problem. Given computing resources, we restrict our domain to polyhedral surfaces.
The following methodological approach for the solution to the problem:

1. Derive a relationship between planar surfaces and its orthographic projection using matrix theory.

2. Find the minimum number of views that are necessary for reconstructing an object with quadric/polyhedral surfaces separately.

3. Reconstruct the conic edges by finding their matrix representations in space to effectively construct a wire-frame model corresponding to the two/ three orthographic views.

4. Search for volume information within the wire-frame model to form the final solids.

5.2 Review of Pre-existing Literature

The problem of converting 2D orthographic projections into 3D objects, has been a well-studied one in the scientific community for many years. Starting from Wesley's algorithm way back in the 1980s to reconstruct polyhedral solids from orthographic projection, till today after immense work from pioneers in this field like Markowsky, B-Rep approaches by Idesawa et. al. have contributed to the growth of this study. The B-rep oriented method is a bottom up approach. It consists of the following steps:

- Generate 3D candidate vertices from 2D vertices in each view
- Generate 3D candidate edges from 3D candidate vertices.
- Construct 3D candidate faces from 3D candidate edges on the same surface.
- Construct 3D objects from candidate faces.

In the first part of our work, we analyzed pre-existing literature and then tailor it to our needs, considering limitation in terms of computational resources and time.

5.3 Simplifying the Problem by Vectorisation

In this section we provide a view to simplify objects that are not polyhedral, in order to be able to compute effectively a solution of reconstruction.

For simple polyhedral reconstruction problem, we can also use the following discussion, but will only make the work computationally expensive. However, this provides source of extending the present scope of work from simply polyhedral objects to many other quadric objects.

Consider any quadric object projected in the x-y plane. The equation of this conic can be given by:

$$a_{11}x^2 + 2a_{12}xy + a_{22}y^2 + 2a_{13}x + 2a_{23}y + a_{33} = 0$$

Vectorizing the same we have:
$f(m) = m^T Am = 0$
where m, A are the matrices given by:

$$m = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \text{ and } A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

The matrix A is a symmetric matrix, and must have a non-zero determinant to be the representation of a non-degenerate conic (a degenerate conic can have its equation *f(m)* further factorised into linear equations). Consider any transformation of the projection such that: $m = Rm_r$

Then, $A_r = R^T AR$, represents the transformation caused in the co-efficient matrix.

5.4 Minimum Number of Views for Reconstructing 3D Objects

We will use the concept of non-degenerate parallel projections to solve this problem.

Under a parallel projection, if the plane containing the space conic is not perpendicular to the projection plane, then the parallel projection is non-degenerate.

This implies that under non-degenerate parallel projections, ellipses, parabolas, and hyperbolas in line drawings are projections of ellipses, parabolas, and hyperbolas, respectively, in space.

So we can assert that if we get any projection of a curve in space in the form of a 2D projection of quadric surface, then the same mapping can be sought backwards as well.

This suggests that, we can identify the type of quadric object that existed in 3D space, simply by analyzing the projection views of the same.

Consider, a conic in space that is lying in plane p. We can assign a co-ordinate system c_p, such that the x-y axes lie in this plane.
Let there exist a global coordinate system c, in the 3D space.

We can perform a Rotation followed by a Translation to either of the co-ordinate systems to align them with the other. Consider the following equation, $\theta = R\ \theta_p + T$, where R and T are the rotation and translation matrices respectively,

$$\text{and } \theta = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \theta_p = \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix}$$

Hence, we have:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} + \begin{bmatrix} t_0 \\ t_1 \\ t_2 \end{bmatrix}$$

For the given matrix θ_p, we know that the x-y plane lies in the plane p of the conic. Hence, for all points z_p must have a zero value. The given equation can be re-written as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ 0 \end{bmatrix} + \begin{bmatrix} t_0 \\ t_1 \\ t_2 \end{bmatrix}$$

We first create the outer boundary framework call *wire-frame* from the given orthographic projections of the top, front and side views of the object. Then we will use the *B-rep* method to create the solid faces of the object to visualize it.

5.5 Unlabeled Projections

If the input data is unlabeled then three distinct views of the solid is required.

Claim: Three distinct orthographic projections are sufficient to uniquely recover a space conic.

For recovering a space sonic, we need three distinct orthographic projections of the conic.

Proof:

There are two cases for the space conic:-

1. Non-degenerate:- In all the three views, conic is not visible as a straight line or point.
2. Degenerate:- Where in at least one view, the conic is a straight line or point.

Case 1:

Let, A be a space conic which lies on a plane p

$$m_p^T A m_p = 0$$

and the curves obtained in different views of A_i are given by

$$m_i^T A_i m_i = 0$$

$$i = 1, 2, ...$$

Putting the transformation of the coordinate matrix, $m_i = R_i m_p$ in the above equation, we get

$$m_p^T R_i^T A_i R_i m_p = 0$$

Combining the above equations, we get
$R_i^T A_i R_i = P^T C_i^T A_i C_i P = A$ for i = 1,2,..

Here, A and P are unknown and definitely non-zero.

Therefore, we get 18 equations and 15 unknowns. Since the no. of equations is greater than unknowns, we get unique solution. Hence, the claim is true for non-degenerate case.

Case 2:

If the conic is degenerate, then in at least one of the projection, we get a straight line in one projection plane. Thus, we can find the plane in which the conic lies by the definition of orthographic projections. The normal of the will be perpendicular to the straight line. Since, the conic will be conic in some projections. Thus, we can find the center and its axes in that by using projections in that plane. Hence, the claim is true for degenerate case.

Thus , the claim is true.

5.6 Labeled Projections

In this part, we first discuss the minimum number of views required to construct a wire-frame from the given orthographic projections. As discussed in the next section, we can uniquely determine a wire-frame, from labeled correspondences in orthographic view. **Claim:** Therefore, we should only require 2 views to draw a unique wire-frame model.

Proof of Claim: Consider, we have any two views of the object, such that they are distinct.

From simply viewing one projection, say top view, we get to know the number of vertices on the object (since all the points are labeled), and the x-y co-ordinates of each of these points.

Now, consider another view, say front view, from this view we can use the correspondences to get the z-coordinate of each vertex.

Now, we have placed all the vertices in the 3D space. The only task remaining is to find out all possible edges.

We can assign the edge pairs directly from the adjacency list of edges obtained to us, by viewing the object from two different views. Note that for any two edges to be distinct, they must contain at least one point that is one but not in the other. This proves that we can find all edges, but we still need to prove that this method does not yield any extra edges.

Let us call any extra edges as candidate edges. The pre-requisite for any candidate edge to be detected by the algorithm is that there must exist lines between vertices A, B in the top view and a correspondence between vertices A' and B', in the front view. If there are any other edges C, D in these views which overlap with these points to join the line in both the views, then by the matching of co-ordinates in all three axes, the point A and B must coincide with C and D itself, thus yielding the same points. But then these can not be called candidate vertices as then this edge actually exists between CD. Hence, by contradiction no candidate edges are obtained.

6 Projections to Wire-frame

Consider the matrices P_T, P_F, P_S, as the collection of points as seen in the orthographic projections of the respective views (top, front and side).

$$P_T = \begin{bmatrix} P_{T_1}(x) & P_{T_1}(y)) \\ P_{T_2}(x) & P_{T_2}(y)) \\ .... & .... \end{bmatrix}, P_F = \begin{bmatrix} P_{F_1}(z) & P_{F_1}(x)) \\ P_{F_2}(z) & P_{F_2}(x)) \\ .... & .... \end{bmatrix}, P_S = \begin{bmatrix} P_{S_1}(y) & P_{S_1}(z)) \\ P_{S_2}(y) & P_{S_2}(z)) \\ .... & .... \end{bmatrix}$$

We consider the case of unlabeled projection here, where we need to identify correspondences between different views.

Let ε be the tolerance allowed in the coordinate measurement. Then we can identify any point in the 3D space if:

$$\begin{gathered} \text{abs}(P_F(x) - P_T(x)) < \varepsilon, \\ \text{abs}(P_S(y) - P_T(y)) < \varepsilon, \text{ and} \\ \text{abs}(P_S(z) - P_F(z)) < \varepsilon \end{gathered}$$

then we know that $(P_T(x), P_T(y))$, $(P_F(z), P_F(x))$ and $(P_S(y), P_S(z))$, are the corresponding projections in each view of a 3D vertex. Where ϵ is a tolerance that allows for inexact match. Thus, all the vertices will be generated by this method.

For generating the 3D edges, a symmetric matrix A will be determined by the equation

$$G_i^T A_i G_i = P^T C_i^T A_i C_i P = A$$

$$i = 1, 2, 3$$

where C_i's are projection matrices :-

$$C_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad C_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad C_3 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Thus, from $G_i = C_i P$, we obtain

$$G_1 = \begin{bmatrix} r_{00} & r_{01} & t_0 \\ r_{10} & r_{11} & t_1 \\ 0 & 0 & 1 \end{bmatrix} \quad G_2 = \begin{bmatrix} r_{00} & r_{01} & t_0 \\ r_{20} & r_{21} & t_2 \\ 0 & 0 & 1 \end{bmatrix} \quad G_3 = \begin{bmatrix} r_{10} & r_{11} & t_1 \\ r_{20} & r_{21} & t_2 \\ 0 & 0 & 1 \end{bmatrix}$$

From this matrix representation, we can obtain results for 3D polyhedral objects

7 From Wire-frame to 3D solid

We will make use of the Markowsky-Wesley approach to solving the problem of 3D reconstruction, because this helps generalize the problem for even those cases, where the wire-frame was constructed from an unlabeled projection. The goal of this algorithm is to reconstruct all possible 3D objects that share the given wire-frame, and then cancel out those reconstructed objects that are not physically possible given the constraints of 3D models. The following methodological approach shall be used for the same.

7.1 Take input data

We assume an adjacency list of edges corresponding to every vertex in the 2D projection. The assumptions made here are that the set of vertices and edges provided to us are that of a valid wire-frame. This means that a 3D wire-frame model should be physically constructible with the given parameters. No assumption is made regarding the existence of a solid or the number of possible
solids from this wire-frame. Validity tests at this stage include:
i. Every vertex, edge is distinct in the set.
ii. Each vertex is connected to at least 3 edges.
We avoid the problem of multiple unknown points of intersection considering that the orthographic projections provided to us are complete.

7.2 Finding out all possible planes:

Given any two intersecting edges, we define the outward normal for that plane through the cross-product of these edges. For each vertex in the 3D wire-frame, we check its adjacency list for the connected edges to that vertex. Computing the non-collinearity check for vertices at this stage, we move to assume a plane for every two non-collinear edges from a vertex. The creation of every new plane of the form

$$ax + by + cz = d$$

, is complemented by a check on the matching of any pre-existing plane with similar characteristics in terms of direction cosines a, b, c and distance along that from the origin, d.

7.3 Finding loops and distinguishing virtual faces:

This is the most important step of the algorithm, because it helps us in solving the question of - *Where to fill?*, in a 3D object. First, we take the set of all valid vertices with their adjacency list of edges obtained from step 2 of the algorithm. We check for all those vertex, edge pairs that could lead to the formation of a possible face.
Definition 3: Consider a face F, with vertex set V(F) and edge set, E(F) associated with it. It must satisfy the following conditions:
i. All points in V(F) are an intersection of two non-collinear line segments as fond in the set E(F).

ii. The end points of e, in E(F) belong to completely and exhaustively in the set V(F).

Consider the graph based approach of finding all cycles in the graph of edges and vertices, and figuring out the connected components in this graph. Any edge on the graph may lie inside a cycle or outside any cycle. For a valid solid model, all the edges should be part of a cycle forming closed loops,and we must remove different bridges.

The basic algorithm to be used for the same is to identify a vertex, take an edge from its adjacency list and perform depth first search on the same to obtain any cycle sin the graph.

After removing the bridges, all the cycles that we obtain, account for virtual faces in the graph.

7.4 Checking for illegal intersection between virtual faces

Consider any two virtual faces, such that they intersect at a point that lies in the interior region of both them. This intuition is used to develop the rules for illegal intersections between virtual or virtual faces. Illegal intersections can occur in the following scenarios:
1. **For an edge, e of face, F1, we have an interior point of face, F2 corresponding to it**

In such a scenario, we remove the virtual face, F2, from the list of faces that was created originally, as it is not possible for a face to have a point such that, it lies in

its interior but is also a part of edge, e, of the object.

2. **There exists a common point, v, such that it is interior to faces F1 and F2**
For this part, we begin with the set of all virtual faces, and construct a recursively enumerable subset A of the set of all faces, such that for every further recursively enumerable subset B of these faces, either B is co-finite or B is a finite variant of A or B is not a superset of A. This may give us a unique set B, or more than one set, for which in either case we proceed with the next step of the algorithm, wherein we use another filtering mechanism while checking for virtual blocks.

7.5 Checking for Virtual Blocks:

The process is analogous to that of section 6.3 wherein we find virtual faces. In fact, the same algorithm could be used for this process as well with minor changes. Note that in this case the graph that we construct is a graph of edges as nodes, and associated planes as edges. First, we take the set of all valid edges with their adjacency list of planes obtained from step 4 of the algorithm. We check for all those edge, plane pairs that could lead to the formation of a possible face.

Again performing a depth first like search to uncover all cycles that exist in this graph. This gives us all those planes that join up together to give a closed surface. Similar to the previous understanding of bridges, we once again remove these sections as they are not physically realizable.

Note: In a labeled case of projection with a valid orthographic view, we will already obtain a graph with no such bridges.

7.6 Yielding all possible solutions to the wire-frame model:

At this stage we combine the Virtual Blocks created in the previous step of the algorithm. The state of each virtual block may be of a hole, or that of a solid. These states yield a valid object in the scenario that:

i. For every edge, e ϵ E(O), where E(O) is the edge set of the object, belongs to two non-coplanar faces F_1 and F_2 , each of which is associated to different virtual faces, such that one of them has a solid state, and the other has hole state.

ii. No cutting edge belongs to two non-coplanar virtual faces in such a way that they are associated with a virtual block in solid and hole state respectively. This implies that all cutting edges are present in regions with solid space.

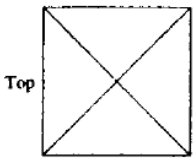
The algorithm for the same is in such a way that we need to systematically go inward into the object with the pre-acquired knowledge that the outermost boundary of the object must contain a solid object. We first assign the outermost virtual block that lies into the infinities outside the outline of the object with a hole space. Having gotten the base case right, the further application is straight forward, where in we apply the two conditions above in a decision tree based approach, to yield the final result.

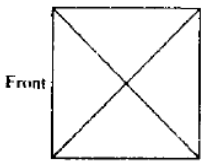
8 Uniqueness of Solution

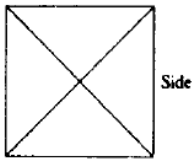
A separate analysis is provided in this section for both labeled and unlabeled objects, since the result varies with the type of input data provided to us.

8.1 Unlabeled Projections

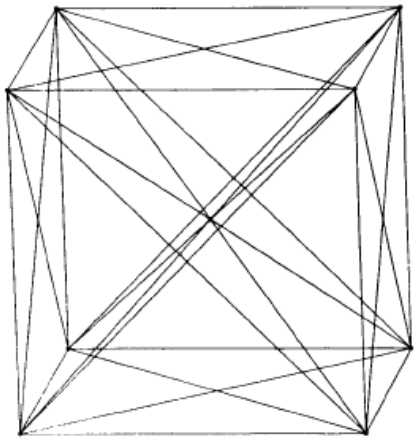
The example of a three cross projection is a suitable counter to the proponents of the uniqueness of unlabeled projections. The Orthographic projections have been shown below:



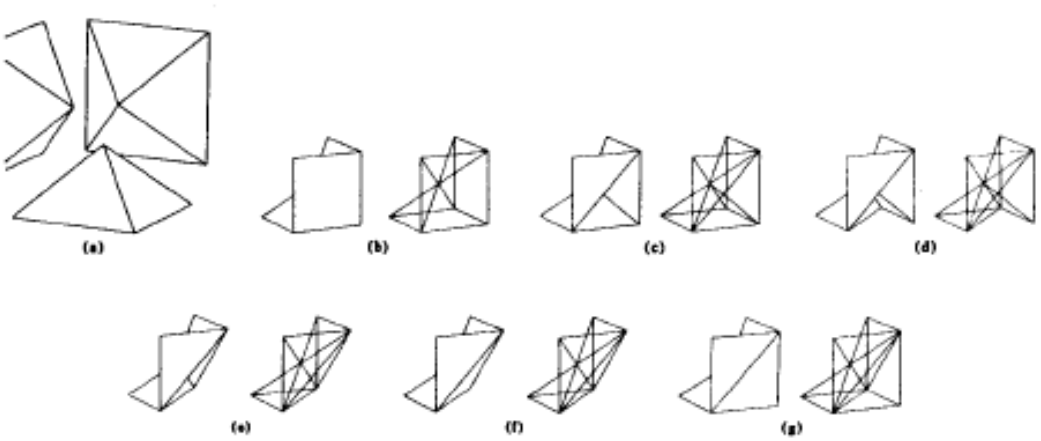




If the diagram above does not have labeled points then the wire-frame will have the following skeleton. This means that we can not uniquely determine the correspondence of various points and even the number of points that may be possible. Hence, we need further analysis to restrict the number of possibilities of solutions



The following 12 solid figures are possible for the given wire-frame:



Hence, through this counter example we can show that a unique solution may not exist for the problem of unlabeled points.

8.2 Labeled Projections

This case is just an extension of the previous case. Note that, in the labeled case each vertex on the projection space is known to the user, along with its correspondence in other views. This implies that we can uniquely determine all edge, vertex sets in the 3D wire-frame model. The problem of 3D reconstruction yields multiple solutions for the given wire-frame because we present to you a generalised algorithm that considers all possible edges that may be present given a projection.

However, this problem does not occur in such a case and we can uniquely draw all virtual blocks without the problem of removing illegal intersections and others specified in Section 6.

The last step, section 6.6, of the reconstruction process is to assign labels to these blocks of either being solid or empty. Since, this is simply a decision tree based process of determining the various possibilities given the base case of outermost virtual block being empty, all these uniquely determined blocks can only have a single parity, unlike the case of the unlabeled frames in which multiple possibilities exist w.r.t. the construction of the virtual blocks.

9 Conclusion

The mathematical analysis in this paper provides a systematic approach to creating a Software Package for Engineering Drawing. We have tackled the important questions of how many views are necessary? how many are sufficient? How can one compute projections given the 3D description? How can one compute the 3D description given one or more projections? We focus on labeled polyhedral objects, while drawing extensions to either unlabeled or quadric objects wherever possible. This enables us to start-off the software package with a strong understanding of polyhedral objects, at the same time allowing us to enhance the scope if possible to more generic objects.

10 References

- A Matrix-Based Approach to Reconstruction of 3D Objects from Three Orthographic Views by *Shi-Xia Lid, Shi-Min Hua, Chiew-Lan Taib and Jia-Guang Suna*
- G Markowsky, MA, Wesley,' Fleshing Out Wire Frames, IBM J. RES.DEVELOP, 1980
- MA Wesley, G Markowsky, Fleshing Out Projection. IBM J. RES.DEVELOP, 1981
- Two Accelerating Techniques for 3D Reconstruction by *LIU Shixia, HU Shimin and SUN Jiaguang*
- 3D Model Generation From The Engineering Drawing by *Jozef VASKÝ, Michal ELIÁŠ, Pavol BEZÁK, Zuzana CERVENANSKÁ, Ladislav IZAKOVIC*
- Reconstruction of a 3D Solid Model from Orthographic Projections by *Zhe Wang, Mohammed Latif*