Fast 3D Modelling Using Orthographic Views

Kuldip Deelip Wagh
Department of Computer Engineering
MCT's Rajiv Gandhi Institute of Technology
Mumbai (400053), India
kanhatop@hotmail.com

Abstract—**3D modelling is very essential step in the process of product design especially in mechanical field, when no other visual model is available before production. Three orthographic views are sufficient for proper understanding of the design of most of the machine objects. Conversion of 2D orthographic view to the 3D object model requires large number of complex calculations like 3D vector arithmetic, complex equation solving, large matrix multiplications etc. Taking into account, the complexity and the resources requirement in this process, the proposed algorithm makes this 2D to 3D conversion possible without a single mathematical calculation but only using conditional statements for comparison; resulting into efficient and faster conversion, which can be more faster in the current processors which are optimized for conditional branching. Also the method is simple to understand due to absence of mathematical calculations.**

Keywords—orthographic, projections, modelling, non curvilinear, reconstruction, 3D, engineering drawing

I. INTRODUCTION

In mechanical field, before manufacturing the object, designers model the object using one of the modelling software like CAD. Orthographic views are the perpendicular projections of an object on rectangular coordinate planes, hence retaining the original dimensions of an object. For most of the objects with no curved surface, at least three orthographic views i.e. front view, side view and top view are required for reconstruction. Only outline edges are necessary to represent any such object. The process of conversion of orthographic views to 3D model consists of four basic steps (figure 1),

- Taking orthographic input in terms of interconnected segments formed by connecting 2D points in any of the view.
- Extracting 3D points from orthographic 2D points.
- Finding the connectivity of obtained 3D points in 3D space i.e. finding the 3D edges of an object.
- Projection of 3D edges to 2D plane for displaying purpose.

The algorithms used in this paper for processing these steps are been derived by observing and analyzing the human thought process while drawing isometric views from orthographic views in engineering drawing.

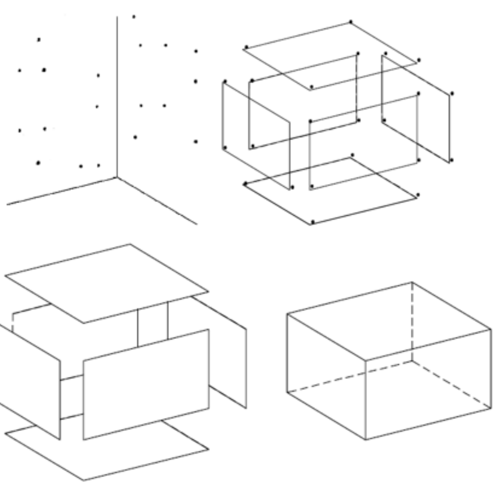
Fig. 1. Steps in modelling

One of the earlier method for this conversion used coordinate geometry in which each of the line is represented by matrix form. Hence the matrix multiplications is complex task in that approach.

The other method [2] consists of very old approach which are not accurate and involves vector algebra e.g. finding normal finding various products of vectors etc. and algorithms like Minimum internal angle algorithm, Decision chaining algorithm etc.

II. METHOD

A. Conventions

In this paper, front view is drawn in X-Y plane, side view in Y-Z plane and top view in X-Z plane (figure 2). That means, each segment drawn in front view will have end vertices with Z coordinate zero. Similarly, side view having X coordinate zero and top view having Y coordinate zero. These coordinates are completely logical and one can assign different coordinate axis to different view.

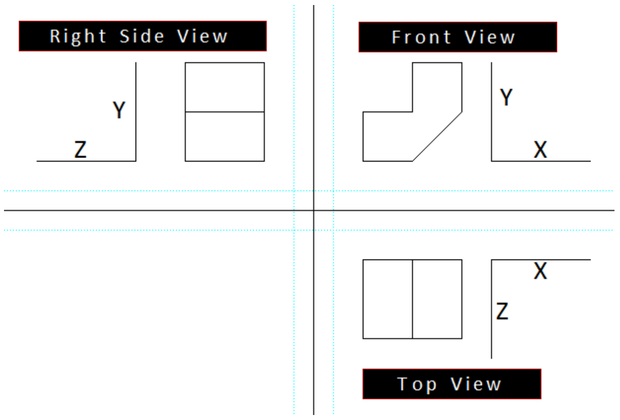

Fig. 2. Convention of 3D coordinates for each view

B. Conceptual Explanation

Orthographic views are projections of any 3D object on X, Y and Z planes. If we draw normal from the vertices of all the views then they all intersect into various points in 3D space (figure 3). These new points are actual 3D vertices of the object in space. The same way, one can find all 3D points for object only using comparison and equality checking.

C. Logical Working

By considering the conventions, first select any one point from front view. Suppose p1 (figure 4). Then find the associated points for p1(x, y, 0) in side view with same Y coordinate as p1(x, y, 0), since Y axis is common in both the views as per convention. The associated points in side view for p1(x, y, 0) are s1(0, y, z1) and s2(0, y, z2). Now, similarly find the associated points for p1 in top view with matching X coordinate. The obtained points are t1(x, 0. Z3) and t2(x, 0, z4).Now among the obtained points find those points in side view matching their Z coordinate with Z coordinates of points in the top view. In the example, s1 matches t2 and s2 matches with t1. I.e. z1 and z4 are equal and z2 and z3 are equal. Hence for point p1 we got two associate points in 3D having coordinates as (x, y, z1) and (x, y, z2). This way process all the points in front view and find all the 3D points of object. The reason to choose the front view vertex points is that, mostly the front view contains more no. of points than the other views hence extracting all the 3D points easily. If one select side view then the result will still be same but the number of comparisons will increase.

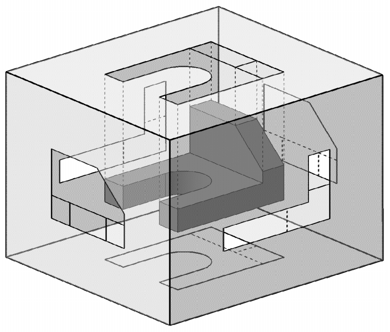

Fig. 3. Intersection of normals from vertices of all views

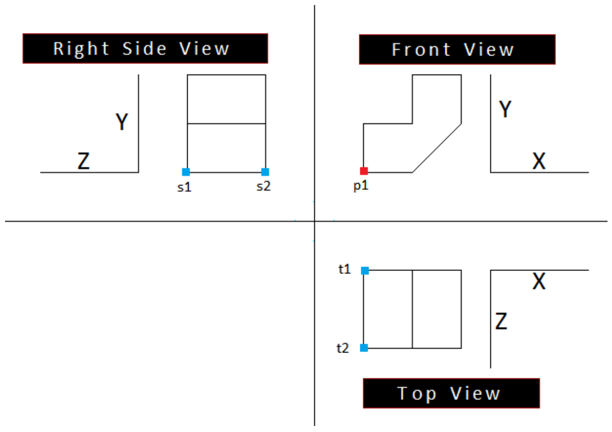

Fig. 4. Example

D. Finding Edges

Next task is to find the connectivity of points in 3D space. It is simple observation that most of the vertices in 3D are intersection point of exactly three edges in 3D. Sometimes it is intersection of more than three but mostly three. That means through each vertex in 3D, three edges pass.

- Each edge which is parallel to any of the three axis in 3D is represented by the segment of same length in any two orthographic views.
- Each edge with length L, which is not parallel to any of the axis and is perpendicular to any one of the axis in 3D i.e. lying in one of the coordinate planes, is represented in one of the orthographic view as a segment with the end points having the same coordinate as the coordinates of end points of the line in 3D except the coordinate for axis to which it is perpendicular.
- For the line which is present in neither of planes, is displayed by the segment with length same as projection of 3D line on any of the coordinate planes.

These observations are sufficient to determine all the edges and also to verify them as written in algorithms.

E. After Processing and Display

The points if contributing exactly three edges verifies the correctness of the points. If any of the point is contributing less than three edges represents an error. But there can also be some points contributing to more than three edges. There are several ways to display an model to screen depending on the type of view chosen.

III. ALGORITHMS

Let F is one of the points in front view, T from top view, S from side view. X coordinate of a point P in orthographic view can be called by xcoordinate[P] can be called as ‘X coordinate of point P’. Similarly Y coordinate by ycoordiante[P]. and Z coordinate by zcoordiante[P]. After converting into 3D, the point has given the index no and its 3D coordinates are stored into x[index], y[index] and z[index]. Each of the point is connected to 3 other points to form an edge. Suppose point p1 is connected to p2 along X axis, then it is stored into data structure by storing index of p2 into xconnection[index of p1] and index of p1 into xconnection[index of p2]. Similarly point along Y axis is stored into yconnection[index] and Z into zconnection[index]. One can use linked list rather than arrays for data structure.

A. *Finding the Points*

```
Initialize index
For each point F(x, y) in front view
 For each point T(x', z) in top view
  If(x' equals x)
   For each point S(y', z) in side view
    If(y' equals y)
     3D coordinates of point are (x, y, z) and are stored into x[index], y[index], z[index].
     Increment Index
    End if
   End for
  End if
 End for
End for
```

B. *Finding Edges Parallel to Coordinate Axes*

```
For each obtained 3D point p1
 For each obtained 3D point p2
  If(any two coordinates of p1 matches with respective coordinates of p2 and third coordinate is different)
   Store the index of p2 into either xconnection[point] or yconnection[point] or zconnection[point] where appropriate connection is chosen by the axis along which coordinates of two points differ.
  End if
 End for
End for
```

C. *Finding Edges non Parallel to Every Coordinate Axes and Perpendicular to One of the Axis*

Now check if every obtained 3D point has initialized xconnection, yconnection and zconnection. If every connection is initialized then stop else perform the following algorithm.

```
For each 3D point p1 whose connection is not initialized
 For each 3D point p2 whose connection is not initialized
  If(both points have exactly one coordinate same and there is an segment in any of the view having end points' coordinate matching with remaining two coordinates of p1 and p2 )
   Store the index of p2 into either of xconnection[p1] or yconnection[p1] or zconnection[p1] where appropriate connection is the one which is not initialized yet. Similarly fill the uninitialized connection of p2 by p1.
  End if
 End for
End for
```

D. *Finding remaining Edges*

Now, all the edges are found and connected for almost all objects with linear surfaces. There are some cases where few points are having more than edges intersecting. But since probability of existence of such object is very little one can directly connect such two points which are still having uninitialized xconnection or yconnection or zconnection.

E. *Output of Example*

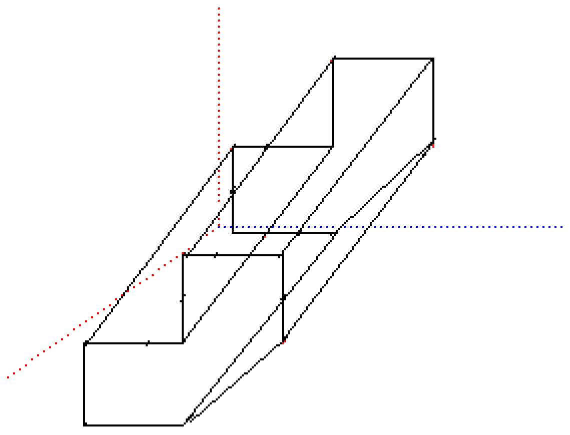

Fig. 5. Output 3D model for above example

F. *Limitations*

- One basic limitation is that this algorithm works only for objects with no curved surfaces.
- The objects with large number of points intersecting more than four edges through them may lead to the wrong edge detection.
- Flushing due to branching can become one negligible issue when executed in older processor.

IV. CONCLUSION

Though there are advancement in techniques for 2D to 3D conversion, still it require lot of computational power. High performance GPUs are used for numeric calculations but not everyone can afford it. The presented algorithm is solution to such problem and can also be modified for curvilinear surfaces in future. This approach of comparison, in the modelling process can lower lot of resource requirement in the field of video games and animation; resulting into high performance on low specification machines.

ACKNOWLEDGMENT

I would like to thank Mrs. Ekata Upadhye and Mr. Kale sir for providing with all the required resources needed for this paper.

REFERENCES

[1] R. E. Marston and M. H. Kuo, "Reconstruction of 3D Objects from Three Orthographic Projections Using Decision-Chaining Method," IAPR Workshop on Machine Vision Applications, Kawasaki, pp. 13-15, December 1994.

[2] Fahiem, M. A. Haq, and Saleemi, "A Review of 3D Reconstruction Techniques from 2D Orthographic Line Drawings," Geometric Modeling and Imaging, July 2007, pp.60-66.

[3] R. Furferi, L. Governi, M. Palai and Y. Volpe, "From 2D Orthographic views to 3D Pseudo-Wireframe: An Automatic Procedure," in IJCA(0975-8887), vol. 5,No. 6, August 2010, pp. 18-24.

[4] W. Wang and Grinstein, "A Survey of 3D Solid Reconstruction from 2D projection line drawings," Computer Graphics Forum, May 1998.

[5] N. Ahuja and J Veenstra, "Generating Octrees from Silhouettes in Orthographic Views," IEEE Trans. On PAMI, Vol 11, pp. 137-149, 1989.

[6] J. Y. Zheng, "Acquiring 3D Model from Sequence of Contours," IEEE Trans. on PAMI, Vol. 16, no. 2, pp. 163-178, 1994.