Mathematical Modelling of Software package for engineering drawing

Done By: Abhishek Maderana(2016CS50399), Aniket Kumar(2016CS50398)

January 23, 2018

Contents

1 Modeling and analysis

Software package we are going to make require graphical calculations in 3D. We should be able to move the 3D object, rotate the object about any axis and project it on any plane. It also requires feature to rebuild 3D object from given projections on 2D plane. All these processes could be done on every point of the object individually. Playing with a point in 3D is very effectively done with the help of matrix. In the given subsections, we have given in detail how these transformations could be done using matrices.

1.1 Translation

Translation is moving every point of a figure by same distance in a given direction. A translation is done with no point of the body being fixed and in which case we uses Matrix multiplication which always have the origin as a fixed point.

There is a common workaround using homogeneous coordinates to represent a translation of a vector space with matrix multiplication: Write the 3-dimensional vector $w = (w_x, w_y, w_y)$ using 4 homogeneous coordinates as $w = (w_x, w_y, w_y, 1)$.

To translate an object by a vector v, each homogeneous vector p(written in homogeneous coordinates) can be multiplied by this translation matrix:

$$T_v = \begin{bmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The multiplication will give the following result:

$$T_v P = \begin{bmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} p_x + v_x \\ p_y + v_y \\ p_z + v_z \\ 1 \end{bmatrix}$$

The inverse of translation matrix can be obtained by reversing the direction of the vector:

$$T_v^{-1} = T_{-v}$$

Similarly, the product of translation matrices is given by adding the vectors:

$$T_u T_v = T_{u+v}$$

Because addition of vectors is commutative, multiplication of translation matrices is therefor also commutative(unlike multiplication of arbitrary matrices).

1.2 Rotation

In linear algebra, a rotation matrix is a matrix that is used to perform a rotation in Euclidean space. For example, using the convention below, the matrix

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

rotates points in the xy-plane counterclockwise through an angle θ about the origin of the Cartesian coordinate system. To perform the rotation using a rotation matrix T, the position of each point must be represented by a column vector v, containing the coordinates of the point. A rotated vector is obtained by using the matrix multiplication R_v

1.2.1 Rotation in 2D

Rotation about center

The rotation matrix around the center can be derived from the Euler Formula, by considering how a complex number is transformed under multiplication by another complex number of unit modulus and at angle θ more than my complex number.

$$z' = ze^{\iota\theta}$$

From the above equation, coordinates (x', y') of the point (x, y) after rotation are

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

We can change these equations in matrix form and write above equations as follows.

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

Thus in 2D, rotation matrix could be written as follows:

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation about any point

We can do this in three steps:

- Shifting the origin to the point of rotation.
- Rotating about the new origin.
- Shifting the axis back to the old origin.

1.2.2 Rotation in 3D

Rotation about a coordinate axis

When we are rotating about an axis, suppose about z-axis, the z-coordinate of the point remain same. Thus, it is similar to 2-D rotation about the center of x-y plane. So, we can write the rotation matrix for rotation about z-axis as:

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

Similarly we can write R_x and R_y as:

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

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

General rotation about any axis passing through the origin

Let the axis of rotation be n. The rotation about n-axis could be done in three steps:

1. Rotate cartesian system to point n-axis in direction of z-axis.
2. Do the rotation about z-axis.
3. Do inverse of first step.

First step could be done in 2 rotations:

1. Rotation about x-axis to bring n-axis on x-z plane.
2. Rotation about y-axis to bring n-axis along z-axis.

Let we rotate angle α around x-axis and then angle β around y-axis. Then the Rotation matrix could be written as:

$$R_n = R_x(-\alpha)R_y(-\beta)R_z(\theta)R_y(\beta)R_x(\alpha)$$

Now, we have to find α and β. Let the axis is pointing from origin to (a, b, c). Value of α and β are easy to find using geometry. On solving, we find α and β as:

$$tan(\alpha) = \frac{b}{c}$$
$$tan(\beta) = \frac{a}{\sqrt{b^2 + c^2}}$$

General rotation about any axis

This could be done by first shifting the origin somewhere on the axis. Then rotating and shifting origin back.

$$R = T_{-v}R'T_v$$

1.3 Projection 3D to 2D

To project a 3D object on a plane, we could project each vertex on the plane, and then join the lines correspoding to the lines in object.

1.3.1 Projection of a point on cartesian plane

A simple orthographic projection onto the plane z=0 can be defined by the following matrix:

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The above matrix can be showed to satisfy the need, because for projecting a point (x, y, z) in x-y plane we just have to keep (x', y')=(x, y) and remove z.

Often, it is more useful to use homogeneous coordinates. The transformation above can be represented for homogeneous coordinates as

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For each homogeneous vector $v = (v_x, v_y, v_z, 1)$, the transformed vector would

be

$$P_v = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ 1 \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ 0 \\ 1 \end{bmatrix}$$

Projection matrix for projection on $z = d$ could be written as:

$$P_{z=d} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

+

1.3.2 Projection of a point on any plane

Projection on any plane is similar to rotation about any axis. Let the normal vector of plane passing from plane away from the origin be $\vec{v}$ and the distance of plane from origin be d. The projection again could be done in three steps:

1. Rotate the cartesian system to point $\vec{v}$ toward z-axis.
2. Take projection on $z = d$.
3. Do inverse of step 1.

Suppose for step 1, we rotate the system by angle α around x-axis and then by angle β around y-axis. And $\vec{v}$ pass from (a, b, c) Then, Similar to rotation, projection matrix could be written as:

$$P = R_x(-\alpha)R_y(-\beta)P_{z=d}R_y(\beta)R_x(\alpha)$$

$$tan(\alpha) = \frac{b}{c}$$

$$tan(\beta) = \frac{a}{\sqrt{b^2 + c^2}}$$

1.4 3D conversion from orthographic projections

When all the lines are given along with 2D image

In this case only front and top views are enough for 3D reconstruction. we will be given (x, y) coordinate of the object in top view and (x, z) coordinate in front view. Thus, we will have (x, y, z) coordinate of every vertex of the object. And all the connected lines are given in both the views. Thus, we get all the vertices

and edges of the 3D model. If, we suppose that the wireframe is correct, it will have only one 3D formation possible. Thus, we get our 3D model.

If the lines connecting the vertex is not given

In this case we have to figure out ourselves all the edges possible for 3D construction by just looking at the line segments in 2D projections. Given the views, after plotting all the points in 3D, for each point suppose p, a set of points to which it could be connected is found corresponding to each projections. Taking intersection of all the sets, a set of points to which p could form an edge is found. Make an edge for all the points in the set. Do the same for all the points.

Now, we have a wireframe. The program could figure out all the possible planes in this wireframe. But the wireframe contains some extra edges and faces. Deletion of pseudo elements(could be edges or planes) can be carried out following given rules to get a original 3-dimensional object:-

1. When an edge is adjacent to more than two faces, at most two faces can be true, the rest must be pseudo.
2. When two faces intersect, only one of them can be true.(If they were both true, the intersection edge would have four adjacent faces which is a contradiction).
3. When an edge is adjacent to only one face, both the edge and face are false.
4. When an edge is adjacent to exactly two co-planar faces, the edge is false and the co-planar faces can be merged.
5. If a true edge is adjacent to exactly two non-co-planar faces, then these faces are both true.(If either of the adjacent faces were false, it would contradict rule 3)
6. A face that is co-planar and adjacent to a true face, when their shared edge projects onto a solid line and is not occluded in a view, is a false face. (The shared edge is needed to project onto the solid line because any other edge would be occluded by the true face. If the candidate face was true, rules 3 and 4 would apply and the shared edge would be deleted).

First and second rules are check for correctness of the model. If both the rules satisfy, check next four rules and then we could come to the conclusion that the model is ready. If any of them not satisfy, then move forward to check next four rules. If any of these rules change anything then restart from rule 1. If there is no change after one loop, then a fixed 3D model could not be formed using these projections.

These method could also be implemented in the case when only two views of the model is given.