International Journal of Trend in Research and Development, Volume 4(5), ISSN: 2394-9333
www.ijtrd.com

# An Elegant Intuitive Algorithm for Rotation about a Mobile Axis

Chaman Lal Sabharwal
Computer Science Department, Missouri University of Science and Technology, Rolla, MO-65409, USA

***Abstract:*** Rotation Transformation is the backbone of visualization of 3D objects. Rotation transformations allow viewing objects from different angles. Rotations about the principal axes are straightforward whereas the rotation about an arbitrary axis is complex. In mobile environment, rotations is computed continuously as the objects move and is thus a costly operation. We give intuitive, constructive, and simpler than the existing methods for creating rotation transformation about mobile axis. It builds upon the intrinsic property of rotation about an arbitrary axis, not the angles the axis makes with the principal axes. We compare it with the existing proofs. Finally, we give an example by interactively creating an axis, and 3D object created by rotation of a mouse drawn polygonal curve. We hope that the application developers and practitioners will find it useful.

***Keywords****: Transformations, Rotations, Arbitrary Axis, Coordinate Frames*

## I. INTRODUCTION

Rotation Transformation is the backbone of visualization of 3D objects. Rotation transformations allow viewing objects from different angles. Rotations about the principal axes are straightforward whereas the rotation about an arbitrary axis is complex. In mobile environment, rotation axis is continuously update as the objects move and is thus a costly operation. We give an intuitive, constructive, and algorithm simpler than the existing methods for creating rotation transformation about mobile axis. It builds upon the intrinsic property of rotation about a principal axis, but it does not compute the angles the axis makes with the principal axes. We compare it with the existing proofs. We hope that the application developers and practitioners will find it useful.

The objects rotate or revolve about an axis. Normally, the rotation is about an axis through the center of the object, whereas revolution is about an axis depending on another object. For example, the earth rotates about the axis through its centroid, whereas a spatial satellite revolves around the earth, and the earth revolves around the sun. Rotation in 3D about principle axes is simple but rotation about an arbitrary axis is harder [1, 2, 3, 4]. The counterclockwise direction is determined by placing the righthand heel at the start of axis, right hand thumb along the axis direction and fingers curl towards the palm to measure positive angle of rotation, see Figure 1.

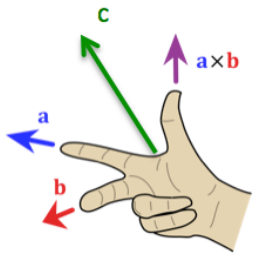
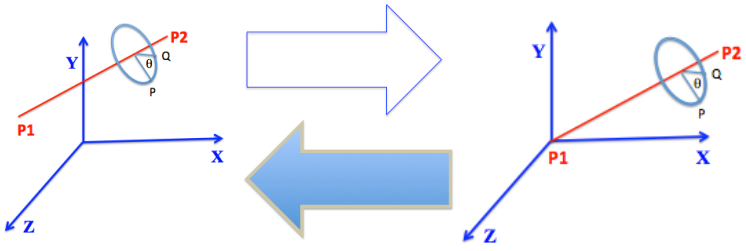



Figure 1: Right hand system of three vectors **a, b, c**.

The rotation about an arbitrary axis requires calculation of angles and seven matrix transformation matrices [3]. We give intuitive, constructive, and simpler than the existing methods for creating transformation for rotation about arbitrary axis. It builds on rotation about *one* of the principal axes (see Figure2). It does not matter which method is used for rotation of an object about an arbitrary axis, first step is to translate the axis to pass though the origin, Figure2.




(a) (b)

Figure 2: Translate-Rotate-Translate (a) $P_1P_2$ defines axis through two arbitrary points $P_1$ and $P_2$.(b) shows transition of axis of rotation, (c). Translate $P_1$ to the origin so that line $P_1P_2$ passes through the origin.

The open GL [5] algorithm creates orthonormal transformation and suffers from several shortcomings: (1) it does not work for cases particularly when $u_x = 0$, (2) angles are computed which can be avoided no matter what the axis is, (3) no mathematical proof is available showing the construction and accuracy of the final matrix as compared with other methods of matrix derivation.

## II. BACKGROUND *LINEAR ALGEBRA.*

The row vector, column vector, position vector, free vector are standard terms defined in any linear algebra book. We use the following notation for vectors: simple variables are in lowercase *italic*, the vectors are in lowercase *bold* and the matrices are in uppercase. For example, the vector $\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$ is 3×1 column vector, $\mathbf{v} = [v_1, v_2, v_3]$ is a 1×3 row vector, and 3×3 matrix A is denoted by $A = [a_{ij}]$. If A is expressed in terms of columns, then

if $A = [\mathbf{c_1}, \mathbf{c_2}, \mathbf{c_3}]$, (three columns) then

$A\times\mathbf{u} = [\mathbf{c_1}\times\mathbf{u},\ \mathbf{c_2}\times\mathbf{u},\ \mathbf{c_3}\times\mathbf{u}]$ is a matrix.

For example, $\mathbf{u}\times\mathbf{v} = \mathbf{u}\times\mathbf{I}\ \mathbf{v}$, where identity matrix is $I = [\mathbf{i}\ \mathbf{j}\ \mathbf{k}]$ with $\mathbf{i} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$, $\mathbf{j} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$, $\mathbf{k} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$; and

$$\mathbf{u}\times I = [\mathbf{u}\times\mathbf{i}, \mathbf{u}\times\mathbf{j}, \mathbf{u}\times\mathbf{k}] = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix}$$

Also the 3×3 matrix A can be expressed in terms of rows, e.g. $A = \begin{bmatrix} \mathbf{r_1} \\ \mathbf{r_2} \\ \mathbf{r_3} \end{bmatrix}$ which has 3 rows of row vectors $\mathbf{r_i}$. If $\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$ is 3×1 column vector, then $\mathbf{r_i u} = [r_{i1}, r_{i2}, r_{i3}] \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = r_{i1}*u_1 + r_{i2}*u_2 + r_{i3}*u_3$ is a number. In vector algebra notation,

If $A = \begin{bmatrix} \mathbf{r_1} \\ \mathbf{r_2} \\ \mathbf{r_3} \end{bmatrix}$ (column of three row vectors), then $A\mathbf{u} = \begin{bmatrix} \mathbf{r_1 u} \\ \mathbf{r_2 u} \\ \mathbf{r_3 u} \end{bmatrix}$ is a column.

### *A. Some Useful Identities*

If **u, v,** and **w** form a righthanded system of orthonormal unit vectors, then

$$\mathbf{u}\bullet\mathbf{v} = \mathbf{0},\ \mathbf{v}\bullet\mathbf{w} = \mathbf{0},\ \mathbf{w}\bullet\mathbf{u} = \mathbf{0}$$

$$\mathbf{u} = \mathbf{v}\times\mathbf{w},\ \mathbf{v} = \mathbf{w}\times\mathbf{u},\ \mathbf{w} = \mathbf{u}\times\mathbf{v}$$

$$\mathbf{u^t u} + \mathbf{v^t v} + \mathbf{w w^t} = \mathbf{I}$$

$$\mathbf{u^t u} = \mathbf{1},\ \mathbf{v^t v} = \mathbf{1},\ \mathbf{w^t w} = \mathbf{1}$$

### *B. Rotation About the Principal Axes*

Let P be a point = $(x, y, z)$ or a position row vector $[x, y, z]$ or column vector $\begin{bmatrix} x \\ y \\ z \end{bmatrix} = [x, y, z]^T$, let a point P be rotated to the point $P' = (x', y', z')$ around one of the principal axis [1,2,3,4].

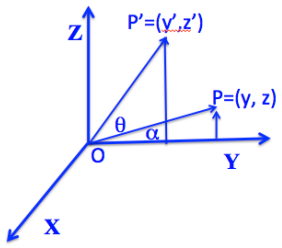



Figure 3. Rotation about x-axis.

Figure 3, For rotation about x-axis, the3D rotation matrix R becomes

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

Most of the time, particularly in Robotics [6], we use short hand notation $c_1 = \cos(\theta_x)$ and $s_1 = \sin(\theta_x)$, and $R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_1 & -s_1 \\ 0 & s_1 & c_1 \end{bmatrix}$.

Similarly $R_y = \begin{bmatrix} c_2 & 0 & s_2 \\ 0 & 1 & 0 \\ -s_2 & 0 & c_2 \end{bmatrix}$ and $R_z = \begin{bmatrix} c_3 & -s_3 & 0 \\ s_3 & c_3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

*Heuristic Shortcuts For Rotations:* The *three matrices* for rotation around the x-, y-, and z-axes can be *replaced* with *single matrix*

*as follows*:

Let $M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{bmatrix}$,

Then $M\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$, $M\begin{bmatrix} y \\ z \\ x \end{bmatrix} = \begin{bmatrix} y' \\ z' \\ x' \end{bmatrix}$, $M\begin{bmatrix} z \\ x \\ y \end{bmatrix} = \begin{bmatrix} z' \\ x' \\ y' \end{bmatrix}$ rotate $[x,y,z]^T$ about *x-axis, y-axis or z-axis* respectively.

### *C. Change Of Basis*

If **u,v,w** are pairwise orthonormal vectors at the origin, the matrix M = [**u, v, w**] maps **i, j, k** to u, v, w system. The matrix M = [**u v w**]also represents an orthonormal system, thus M is invertible and $M^{-1} = M^T = \begin{bmatrix} \mathbf{u}^T \\ \mathbf{v}^T \\ \mathbf{w}^T \end{bmatrix}$. For example, if $P_{ijk}$ and $P_{uvw}$ represent coordinates of a point relative to two coordinate frames, then

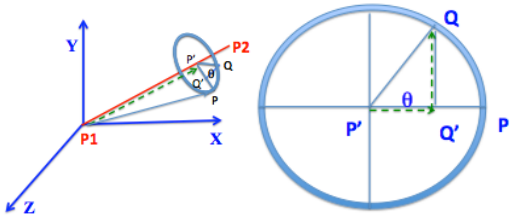
$P_{ijk} = MP_{uvw}$ or $P_{uvw} = M^T P_{ijk}$

## III. TRADITIONAL ROTATION ABOUT AN ARBITRARY AXIS

There is no unique way to derive the rotation matrix. [7,8,9,10,11,12,13]one way (*geometric/projective approach*) is to directly arrive at by traversing the vector path, an alternate way (*algebraic/transformation approach*) is to create and combine matrices to determine orthonormal matrix for rotation about the arbitrary axis [5]. To design the transformation for rotation about an arbitrary axis$P_1P_2$ by an angle $\theta$from P to Q, let the direction vector be $\mathbf{u} = \frac{P_1P_2}{|P_1P_2|}$. If the line does not pass through the origin, (see Figure 2a), first translate the line so that it passes through the origin, see Figure 2b. After we calculate the orthonormal transformation, we can translate the line back to original place. In each of the two methods, translation of the axis of rotation to pass through the origin is factored out and is not repeated. Only the core rotation transformation about the axis through the origin is described below. The easiest way to specify axis is by defining one point $P_2$ and assuming the other point $P_1$at the origin.

### *A. Geometric Approach*

The *geometric* approach [7,8,9] uses polygonal path of vectors from O to Q, see Figure 4. It is a little challenging to define vectors along the path from O to Q from the knowledge of P. For counter clockwise rotation from P about line $P_1P_2$ by an angle $\theta$ to Q, first define the direction vector $\mathbf{u} = \frac{P_1P_2}{|P_1P_2|}$. (see Figure3) Now we project P on line $P_1P_2$to get P', rotate P'P about P' in the counter clockwise direction by angle $\theta$ to get P'Q, then project Q on P'P to get Q'. After calculating all the projections, directions and lengths, to get from O to Q, algorithm proceeds from O to P', from P' to Q' , and finally from Q' to Q.




(a) (b)

Figure 4. Description vectors by dotted linesfrom the origin to Q [3,7].

3.1.1Traditional View and Heuristics [3,4,7,8,9]for calculating Matrix to rotate about the axis direction **u,** itis in two parts: the part along **u** remains unchanged. Only one part is rotated, because the rotation is in a plane orthonormal to the axis, see Figure 3b. The position vector **P** can be decomposed into two parts: **P||** along the **u** axis direction, that remains unchanged and **P⊥**orthogonalto the axis, that rotates. The rotation preserves the length of |**P⊥**|.As P is rotated to Q about P', the distance of points P and Q from the vector **u** (or from the point P' with position vector **P•u u**) does not change. Now the vector P'Q is composed to two parts: one P'Q' along **P⊥**, and other Q'Q orthogonal to **P⊥**with lengths |**P⊥**| cos($\theta$) and |**P⊥**| sin($\theta$) respectively.That is, the rotated part is decomposed into two parts along P⊥ and orthogonal to P⊥. The vector component along P⊥ has length |P⊥| cos($\theta$) and along the orthogonal to P⊥ has length |P⊥| sin($\theta$). Now

$$\mathbf{Q} = \mathbf{P}\bullet\mathbf{u}\ \mathbf{u} + |P\perp| \cos(\theta) \frac{\mathbf{P}-\mathbf{P}\bullet\mathbf{uu}}{|P\perp|} + |P\perp| \sin(\theta) \frac{\mathbf{u}\times(\mathbf{P}-\mathbf{P}\bullet\mathbf{uu})}{|P\perp|}$$

$$\mathbf{Q} = \mathbf{P}\bullet\mathbf{u}\ \mathbf{u} + \cos(\theta)\ (\mathbf{P}-\mathbf{P}\bullet\mathbf{uu}) + \sin(\theta)\mathbf{u}\times\mathbf{P}$$

$$\mathbf{Q} = (1 - \cos(\theta))\mathbf{P}\bullet\mathbf{uu} + \mathbf{P}\cos(\theta) + \sin(\theta)\mathbf{u}\times\mathbf{P}$$

$$\mathbf{Q} = (1 - \cos(\theta))\ \mathbf{uu}^T\ P + \cos(\theta)\ I\ P + \sin(\theta)\mathbf{u}\times\mathbf{I}\ \mathbf{P}$$

$$\mathbf{Q} = (\ (1 - \cos(\theta))\ \mathbf{uu}^T + \cos(\theta)\ I + \sin(\theta)\mathbf{u}\times\mathbf{I}\ )\ \mathbf{P}$$

The matrix of transformation is

$$M = (1 - \cos(\theta))\ \mathbf{uu}^T + \cos(\theta)\ I + \sin(\theta)\mathbf{u}\times\mathbf{I}$$

The full length expansion of M becomes Rodrigue's formula, see Section 4.

### *B. Algebraic Approach*

Interestingly, in this approach the **u** vector is rotated to create rotation about the **u** vector. One approach is again more conceptual and computation intensive, the other approach is simpler and creates orthonormal transformation[10, 11, 12].

This is one of the original ways of conceptual thinking about the rotation transformation, in terms of Euler's angles. Euler's angles appear in many applications, e.g. Robotics. Intuitively, it is taken for guaranteed without mathematical proof of the equivalent formula derived geometrically accurate. Again, assuming that the axis passes through the origin, in this case, five transformations are:

Step 1: [see Figure5] Calculate the angle $\alpha$ of rotation, and rotate axis vector about X-axis by an angle $\alpha$ to get into the x - z plane
Step 2: [see Figure5] Calculate the angle $\beta$ of rotation, and rotate the resulting vector about the Y-axis to get it in the X-axis direction
Step 3: Rotate about x-axis by angle $\theta$
Step 4: Reverse the rotation about the Y axis
Step 5: Reverse the rotation about the X axis
Calculate the total transformation that is the product of 5 matrices and is orthonormal matrix.

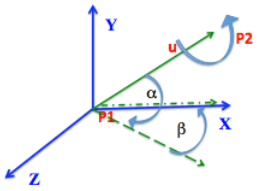



Figure 5. *Conceptual Angles.*First rotate **u** about x-axis by angle α to bring it to z-x plane. Then rotate it about y-axis by angle β to align with x- axis.

## IV. NEW PROOF OF ROTATION ABOUT AN ARBITRARY AXIS CHANGE OF BASIS REPRESENTATION

In this section, we dwell upon the intrinsic property of rotation axis of rotation, not the angles it makes with the principal axes/planes. There was an attempt to use change of basis [13], their argument is inaccurate and inefficient. There are several sources of inaccuracy and inefficiency: (1) it computes **v** as $\mathbf{v} = \mathbf{u}\times(u_1, 0, 0)$, it fails to work if $u_1 = 0$, (2) it performs cross product to determine **v** which is not necessary and can be eliminated altogether as shown below [see Figure 6], (3) there is no mathematical proof of single matrix representation equivalent to other methods to authenticate the formula in Section 3.1. We address these questions here: (1) we use *three* matrices to compose transformation instead of *five* matrices, (2) we do not calculate angles thereby replacing *two* angles computations, (3) we provide mathematical proof.

First [10], consider the case in which the axis of rotation is along one of the principal axes. Then, consider the general case. Recall that if **u, v, w** are mutually orthonormal unit vectors forming a right handed system, then the matrix R = [**u v w**] is orthonormal and its inverse is $R^{-1} = R^T = \begin{bmatrix}\mathbf{u}^T\\ \mathbf{v}^T\\ \mathbf{w}^T\end{bmatrix}$ because

$$R^T R = \begin{bmatrix}\mathbf{u}^T\\ \mathbf{v}^T\\ \mathbf{w}^T\end{bmatrix}[\mathbf{u}\ \mathbf{v}\ \mathbf{w}] = \begin{bmatrix}\mathbf{u}^T\mathbf{u} & \mathbf{u}^T\mathbf{v} & \mathbf{u}^T\mathbf{w}\\ \mathbf{v}^T\mathbf{u} & \mathbf{v}^T\mathbf{v} & \mathbf{v}^T\mathbf{w}\\ \mathbf{w}^T\mathbf{u} & \mathbf{w}^T\mathbf{v} & \mathbf{w}^T\mathbf{w}\end{bmatrix}$$

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

Therefore $R^T R = RR^T = I$ and $[\mathbf{u}\ \mathbf{v}\ \mathbf{w}]\begin{bmatrix}\mathbf{u}^T\\ \mathbf{v}^T\\ \mathbf{w}^T\end{bmatrix} = I$ implies $\mathbf{uu}^T + \mathbf{vv}^T + \mathbf{ww}^T = \boldsymbol{I}$

**Lemma.** If R = [**u v w**] is an orthonormal matrix, $RR^T = R^TR = I$, then

$$\mathbf{wv}^T - \mathbf{v}\,\mathbf{w}^T = \mathbf{u}\times\mathbf{I}$$

Proof. $\mathbf{wv}^T - \mathbf{v}\,\mathbf{w}^T$

$= \mathbf{u}\times\mathbf{v}\,\mathbf{v}^T - \mathbf{w}\times\mathbf{u}\,\mathbf{w}^T$

$= \mathbf{u}\times\mathbf{v}\,\mathbf{v}^T + \mathbf{u}\times\mathbf{w}\,\mathbf{w}^T$

$= \mathbf{u}\times(\mathbf{v}\,\mathbf{v}^T + \mathbf{w}\,\mathbf{w}^T)$

$= \mathbf{u}\times(\mathbf{I}-\mathbf{uu}^T)$

$= \mathbf{u}\times\mathbf{I} - \mathbf{u}\times\mathbf{u}\,\mathbf{u}^T$ because $\mathbf{u}\times\mathbf{u} = \mathbf{0}$

$= \mathbf{u}\times\mathbf{I}$

We know that computation of rotation around principal axes x-, y-, z-axis is straightforward. For simplicity let the direction **u** pass through to the origin. Otherwise we use translation to make it pass through the origin, that is O becomes the origin of the coordinate system.

Since it is a rotation about an axis through the origin, we do not need to use homogeneous coordinates.

The advantages of our algorithm are that we use

(1)*three*3×3matrices to compose instead of *five*4×4 matrices

(2) *zero* calculations of angles instead of *two* angles.

(3) *we do not rotate the axis of rotation*.

If we can align **u** along x- axis, then we can use rotation about x-axis, then realign the result back along **u** direction. Since we know **u**, unlike previous methods, we can find arbitrary mutually orthonormal unit vectors, **v** and **w** so that **u, v, w** form a right handed system. However this is simple **v**: if $u_1 = 0$, then set $\mathbf{v} = \begin{bmatrix}1\\0\\0\end{bmatrix}$, $\mathbf{w} = \begin{bmatrix}0\\u_3\\-u_2\end{bmatrix}$, else set $\mathbf{v} = \begin{bmatrix}-u_2\\u_1\\0\end{bmatrix}$, $\mathbf{v} = \frac{\mathbf{v}}{|\mathbf{v}|}$; $\mathbf{w} = \mathbf{u}\times\mathbf{v}$;


(a) (b)

Figure 6. New frame(green) and old frame(blue). Rotation after change of basis vectors

Let R = [**u v w**], clearly R**i** = **u**, R**j** = **v**, R**k** = **w**, thus the matrix R = [**u v w**] maps **i, j, k**, to **u, v, w** directions. Then the inverse $R^T = \begin{bmatrix}\mathbf{u}^T\\\mathbf{v}^T\\\mathbf{w}^T\end{bmatrix}$ maps the **u, v, w** to **i, j, k.** Similarly we can align **u** with any of x-, y- or z-axis principal directions, if desired.

Now we can align **u** along x- axis, then we rotate about x-axis, then realign the result back along **u**direction. The transformation becomes

$$[\mathbf{u}\ \mathbf{v}\ \mathbf{w}]\begin{bmatrix}1 & 0 & 0\\0 & \cos(\theta) & -\sin(\theta)\\0 & \sin(\theta) & \cos(\theta)\end{bmatrix}\begin{bmatrix}\mathbf{u}^T\\\mathbf{v}^T\\\mathbf{w}^T\end{bmatrix}$$

$$= \mathbf{u}\,\mathbf{u}^T + (\cos(\theta)\,(\mathbf{v}\,\mathbf{v}^T + \mathbf{w}\mathbf{w}^T) + \sin(\theta)\,(-\mathbf{v}\,\mathbf{w}^T + \mathbf{w}\mathbf{v}^T)$$

Since $\mathbf{u}\mathbf{u}^T+\mathbf{v}\mathbf{v}^T+\mathbf{w}\mathbf{w}^T = I$,and $\mathbf{w}\mathbf{v}^T - \mathbf{v}\mathbf{w}^T = \mathbf{u}\times I$

$$= \mathbf{u}\,\mathbf{u}^T + \cos(\theta)\,(\mathbf{I} - \mathbf{u}\mathbf{u}^T) + \sin(\theta)\,\mathbf{u}\times I$$

$$= (1 - \cos(\theta))\mathbf{u}\mathbf{u}^T + \cos(\theta)\,I + \sin(\theta)\,\mathbf{u}\times I$$

$$= (1 - \cos(\theta))\,\mathbf{u}\mathbf{u}^T + \cos(\theta)\,I + \sin(\theta)\,\mathbf{u}\times I$$

These two expressions obtained by geometric and algebraic approach are identical.

The matrix expansion becomes Rodrigue's formula, named after Olinde Rodrigues, as below.

$$R = \begin{bmatrix}(1-cos(\theta))\,u_1u_1 + cos(\theta) & (1-cos(\theta))\,u_1u_2 - sin(\theta)\,u_3 & (1-cos(\theta))\,u_1u_3 + sin(\theta)\,u_2\\(1-cos(\theta))\,u_2u_1 + sin(\theta)\,u_3 & (1-cos(\theta))\,u_2u_2 + cos(\theta) & (1-cos(\theta))\,u_2u_3 - sin(\theta)\,u_1\\(1-cos(\theta))\,u_3u_1 - sin(\theta)\,u_2 & (1-cos(\theta))\,u_3u_2 + sin(\theta)\,u_1 & (1-cos(\theta))\,u_3u_3 + cos(\theta)\end{bmatrix}$$

The *algorithm* for rotation about an arbitrary axis $P_1P_2$ by an angle in the counter clockwise direction becomes

1. Find $\mathbf{u} = \frac{P_1P_2}{|P_1P_2|}$.

2. Define **v** (without calculating angles) and **w**:

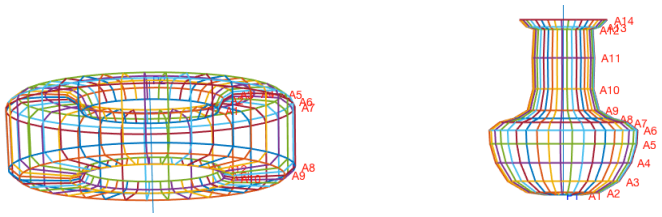
if $u_1 = 0$, then set $\mathbf{v} = \begin{bmatrix}1\\0\\0\end{bmatrix}$, $\mathbf{w} = \begin{bmatrix}0\\u_3\\-u_2\end{bmatrix}$ else set $\mathbf{v} = \begin{bmatrix}-u_2\\u_1\\0\end{bmatrix}$,$\mathbf{v} = \frac{\mathbf{v}}{|\mathbf{v}|}$, $\mathbf{w} = \mathbf{u}\times\mathbf{v}$

3. Define rotation matrix $\mathbf{R} = [\mathbf{u}\ \mathbf{v}\ \mathbf{w}]\begin{bmatrix}1 & 0 & 0\\0 & \cos(\theta) & -\sin(\theta)\\0 & \sin(\theta) & \cos(\theta)\end{bmatrix}\begin{bmatrix}\mathbf{u}^T\\\mathbf{v}^T\\\mathbf{w}^T\end{bmatrix}$

The transformation becomes translate $P_1$ to origin, apply R, then retranslate origin to original position of $P_1$. Thus the transformation becomes: $T_r(P_1)RT_r(-P_1)$

## V. VISUALIZATION DISPLAYS OF EXPERIMENTS

Interaction and Visualization is an important statistical activity. The code is implemented using MATLAB_R2016a MAC OS Sierra8 GB 1600 MHz DDR3. Mouse is used for interactive input for construction of arbitrary axis and base polygonal curve. This curve is rotated to create 3D object for visualization. Sample displays are shown in Figure7.




(a) (b)

Figure 7. Use mouse to draw tire(Figure 6a) or bottle(Figure 6b). First $P_1P_2$ axis is drawn, then base curve points $A_1$, $A_2$, …, $A_n$ etc. are drawn.T he polygonal curve $A_1$, $A_2$, …, $A_n$ is rotated about the axis.

## CONCLUSION

We have presented a new look at the proof of the rotation transformation about an arbitrary axis. It is intuitive and is simpler for the practitioner and application developers. We have derived the matrix of rotation about arbitrary axis in 3D. Besides 3D Graphics, rotation matrices are used in Principle Component Analysis (PCA) also. This idea may be extended to PCA SVD, and multidimensional scaling.

### *References*

[1] David J. Eck, Introduction to Computer Graphics, 2016, Free book, http://math.hws.edu/eck/cs424/downloads/graphicsbook-linked.pdf
[2] Rogers, D. F. (2000), Mathematical Elements for Computer Graphics. New York: McGraw Hill.
[3] John F Hughes, Andries Van Dam, Morgan McGuire, David F. Sklar, James D. Foley, Steven K. Feiner, Kurt Akeley, Computer Graphics; Principles and Practice, 3rdEdition Addison Wesley, 2014.
[4] Sumanta Guha, Computer Graphics through Open GL from theory to Experiments, CRC press, 2011
[5] Philippe B. Laval Rotation About an Arbitrary Axis, ksuweb.kennesaw.edu/~plaval/math4490/rotgen.pdf
[6] Saeed B. Niku, Introduction to Robotics, Analysis, Control, Applications, John_Wiley, 2011
[7] Hearn, Baker and Carithers, Computer Graphics with Open GL, Prentice 2011
[8] Angel, projections] Edward Angel, Dave Shreiner, Interactive Computer graphics with OpenGL,Addison Wesley, 2012
[9] Hill, F.S. JR., Computer Graphics Using Open GL, Prentice Hall, 2001.
[10] Paul Bourke, Rotate A Point About An Arbitrary Axis, 1992, Updated August 2002 paulbourke.net/geometry/rotate/
[11] Jona Gomes, Luiz Velho, Mario Costa Sousa Computer Graphics, Theory and Practice, CRC publisher, 2012
[12] Steven Gortler, Foundaitons of 3d Computer Graphics Open GL, MIT Press, 2012
[13] Information Coding /Computer Graphics, ISY, LiTH, Rotation about arbitrary axis: http://www.computer-graphics.se/TSBK07-files/pdf13/7b.pdf