# Efficient Iterated Rotation of an Object

W. RANDOLPH FRANKLIN

***Abstract*—This paper presents a more efficient method for iterated rotation in three dimensions where multiple points are being rotated by multiple angles about the same axis, as would be done in robotic simulation or computer graphic animation. General axes that do not necessarily pass through the origin and multiple composed rotations are also handled. The algorithm is numerically well conditioned for all axis directions.**

***Index Terms*—Animation, Cayley-Klein parameters, computer-aided design, computer graphics, Euler angle, quaternion, robotics, rotation, transformation.**

Manuscript received October 30, 1980; revised April 18, 1983. This material is based upon work supported by the National Science Foundation under Grants ENG 79-08139 and ECS 80-21504.

The author is with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12181.

## INTRODUCTION

A frequent operation in computer-aided design (CAD), computer graphics, and computer-assisted animation is the rotation of an object or surface in three dimensions. The usual procedure is to rotate certain fixed points such as vertices or knots such that the remainder of the object follows automatically. We frequently wish to rotate the same object repeatedly about the same axis by successively larger angles. One important use of animation is the verification of programs to control robot manipulator arms. Foley and Van Dam [4, pp. 254–255] present some other efficiency considerations for real-time rotation. If the object is following a more complicated path, each movement can be decomposed into a rotation and a translation or alternatively into two rotations. Optimizing the calculation of the composition of several iterated rotations is important in robot manipulator languages, but at present, calculating this in real time "is beyond the capabilities of most computers" [9, p. 263].

After defining its assumptions and notation, this paper will survey the various general rotation formulas. Some authors present only special cases, such as Chasen [1] who describes how to transform a curve in an oblique plane into another coordinate system. The general methods include: 1) rotation about the three coordinate axes in turn where the axes are either fixed in space or move with the object, 2) rotation using Euler angles or Cayley-Klein parameters, 3) rotation of the coordinate system about the $X$ and $Y$ axes to make the axis of rotation coincident with the $Z$ axis, 4) a vector formulation, and 5) a quaternion formulation. Finally, new formulas optimized for the robot manipulator and animation cases will be presented.

## ASSUMPTIONS AND NOTATION

We rotate a point, represented as a horizontal three-tuple or four-tuple, relative to a fixed set of axes. If the transformation is represented as a matrix, then it postmultiplies the vector representing the point.

The measures of execution time will be $\tau_+$ and $\tau_\times$, respectively, the number of additions and subtractions, and the number of multiplications of floating-point numbers. The time for bookkeeping, integer arithmetic, and so on will be ignored in accordance with conventional procedures. This isolates the essential differences and suppresses variables that depend more on the compiler's efficiency than on the algorithm. We will sometimes count $\tau_t$, the number of trignometric evaluations needed in calculating coefficients of an equation, before it is applied to the points. Other aspects of this precalculation time will generally be ignored.

## ROTATION ABOUT THE $X$, $Y$, AND $Z$ AXES

This formulation appears in Giloi [5], Foley and van Dam [4, pp. 255–259], Paul [9, pp. 25–27], and Rogers and Adams [11]. They compute a rotation matrix as the composition of the following three matrices:

$$R_z(\alpha) = \begin{vmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

$$R_y(\beta) = \begin{vmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{vmatrix}$$

0018-9340/83/1100-1064$01.00 

and

$$R_x(\gamma) = \begin{vmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & \sin\gamma \\ 0 & -\sin\gamma & \cos\gamma \end{vmatrix}$$

to give $R(\alpha, \beta, \gamma) = R_z(\alpha) \times R_y(\beta) \times R_x(\gamma)$.

This method is a composition of rotations in the $XY$, $XZ$, and $YZ$ planes in that order. It can be difficult to determine $\alpha$, $\beta$, and $\gamma$ given the source and destination orientations of some object.

Applying $R_z(\alpha)$, $R_y(\beta)$, and $R_x(\gamma)$ successively to the point requires a total of $\tau_x = 12$ and $\tau_+ = 6$. Composing them first to give $R(\alpha, \beta, \gamma)$ and then applying that is cheaper since it would cost $\tau_x = 14$ and $\tau_+ = 4$ for the composition followed by $\tau_x = 9$ and $\tau_+ = 6$ per application using the formula for $R(\alpha, \beta, \gamma)$ given in [5, p. 96].

Paul gives a method for determining the single axis and angle of rotation to which a general rotation matrix is equivalent. He also presents important numerical accuracy considerations for this operation, and gives the formulas for the general matrix. The axis of the general rotation can also be determined as the eigenvector of $R(\alpha, \beta, \gamma)$ with a corresponding eigenvalue of unity. The angle can be determined from the other two eigenvalues, which are complex and have the form

$$\cos\theta \pm i \sin\theta.$$

## Rotation by Euler Angles

An alternative rotation uses the Euler angles [2, pp. 174-177], [7], [9, pp. 43-45]. $\phi$, $\theta$, and $\psi$ which represent a rotation by $\phi$ about the $Z$ axis, followed by a rotation by $\theta$ about the new $X$ axis, and concluded by a rotation by $\psi$ about the newer $Z$ axis. These three rotations may equivalently be applied in the reverse order about the fixed world axes. The combined rotation matrix is $R_z(\phi, \theta, \psi) = R_z(\psi) \times R_x(\theta) \times R_z(\phi)$.

The execution times are the same as before, that is, $\tau_x = 9$ and $\tau_+ = 6$ if $R_z(\phi)$, $R_x(\theta)$, and $R_z(\psi)$ are combined into one matrix before being applied. Paul gives the resulting matrix.

The three Euler angles can also be expressed as a $2 \times 2$ matrix of complex Cayley-Klein parameters [6]:

$$\begin{vmatrix} \alpha & \beta \\ \gamma & \partial \end{vmatrix} = \begin{vmatrix} \exp -i\phi/2 & 0 \\ 0 & \exp i\phi/2 \end{vmatrix} \begin{vmatrix} \cos\theta/2 & -i\sin\theta/2 \\ -i\sin\theta/2 & \cos\theta/2 \end{vmatrix} \begin{vmatrix} \exp -i\psi/2 & 0 \\ 0 & \exp i\psi/2 \end{vmatrix}.$$

If we are given two rotations that are to be combined, then the Cayley-Klein parameter matrix of the combination is the product of the parameter matrices of the original rotations. From this, we can determine the Euler angles of the combined rotation.

## Rotation About the $X$, $Y$, $Z$, $Y$, and $X$ Axes in Turn

Newman and Sproull [8] apply the following five matrices for a rotation about the origin:

$$R' = R_1 R_2 R_\theta R_2^{-1} R_1^{-1}$$

where

$$R_1 = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & c/v & b/v & 0 \\ 0 & -b/v & c/v & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad \text{where } v = \sqrt{b^2 + c^2}$$

$$R_2 = \begin{vmatrix} v & 0 & a & 0 \\ 0 & 1 & 0 & 0 \\ -a & 0 & v & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

$$R_\theta = \begin{vmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Here $(a, b, c)$ is a unit vector along the rotation axis and $\theta$ is the angle of rotation. This method essentially performs the rotation in a coordinate system where the axis of rotation is coincident with the $z$ axis, which is ill conditioned if $(a, b, c)$ is almost coincident with $(1, 0, 0)$. Multiplying by each of the five matrices requires $\tau_x = 4$ and $\tau_+ = 2$, for a total of $\tau_x = 20$ and $\tau_+ = 10$.

If the five matrices are first composed, costing $\tau_x = 40$ and $\tau_+ = 18$ (where multiplications by 0 or 1 or additions of 0 are not counted), then applying the resulting $4 \times 4$ matrix requires $\tau_x = 9$ and $\tau_+ = 6$ (since the fourth row and column will be changed). This is advantageous if the same rotation matrix is being used more than about three times (depending on the relative costs of the additions, multiplications, and bookkeeping).

## A Vector Formulation

Faux and Pratt [3] have a vector representation:

$$\boldsymbol{p}' = (\boldsymbol{p} \cdot \boldsymbol{u})\boldsymbol{u} + \cos\theta(\boldsymbol{p} - (\boldsymbol{p} \cdot \boldsymbol{u})\boldsymbol{u}) + \sin\theta(\boldsymbol{u} \times \boldsymbol{p}). \quad (1)$$

See Fig. 1. Here point $\boldsymbol{p}$ is being rotated about axis $\boldsymbol{u}$ through the origin by angle $\theta$. "$\cdot$" and "$\times$" are the scalar (dot) and vector (cross) products, respectively. Since a dot product costs $\tau_x = 3$ and $\tau_+ = 2$ and a cross product costs $\tau_x = 6$ and $\tau_+ = 3$, (1) requires $\tau_x = 24$ and $\tau_+ = 16$. While this appears slower than the previous formulations, the input, i.e., the axis direction and angle of rotation, is in a form that is easy to visualize. Also, the setup calculations required are just one sine and one cosine, which compares quite favorably to the matrix initializations required in the previous methods. Thus, for small objects where the setup time for the matrices is longer than the time to apply them, this method is faster.

Pavlidis [10] derives a rotation formula about an axis specified by its direction cosines using vector and tensor notation, and he then determines the rotation matrix from it.

## Quaternions

The quaternion representation [2, pp. 168-172] is both very compact and allows us to determine the axis and angle of the single rotation resulting from the composition of several rotations. (A quaternion $q = a + bi + cj + dk$ is a noncommutative extension of a complex number where $i^2 = j^2 = k^2 = -1$ and $ij = -ji = k$, $jk = -kj = i$, $ki = -ik = j$). A 3-D point $(p_x, p_y, p_z)$ can be represented as a pure quaternion:

$$\boldsymbol{p} = p_x \boldsymbol{i} + p_y \boldsymbol{j} + p_z \boldsymbol{k}.$$

A rotation of $\boldsymbol{p}$ about axis $\boldsymbol{u}$, where $\boldsymbol{u}$ is a unit vector, by angle $\theta$ can be expressed as

$$\boldsymbol{p}' = \boldsymbol{q}\boldsymbol{p}\boldsymbol{q}^* \quad (2)$$

where

$$\boldsymbol{q} = \cos\theta/2 + (u_x \boldsymbol{i} + u_y \boldsymbol{j} + u_z \boldsymbol{k}) \sin\theta/2$$

and $q^*$ is the quaternion conjugate of $q$. Its execution times are $\tau_x = 24$ and $\tau_+ = 16$. Equation (2) is quite compact. The quaternion and vector formulations use only four numbers to represent a general rotation, unlike a $3 \times 3$ matrix representation which requires nine numbers.

If we rotate $\boldsymbol{p}$ by $\boldsymbol{q}_1$ to get $\boldsymbol{p}'$ and then by $\boldsymbol{q}_2$ to get $\boldsymbol{p}''$, we have

$$\boldsymbol{p}' = \boldsymbol{q}_1 \boldsymbol{p} \boldsymbol{q}_1^*$$

$$\boldsymbol{p}'' = \boldsymbol{q}_2 \boldsymbol{p}' \boldsymbol{q}_2^*$$

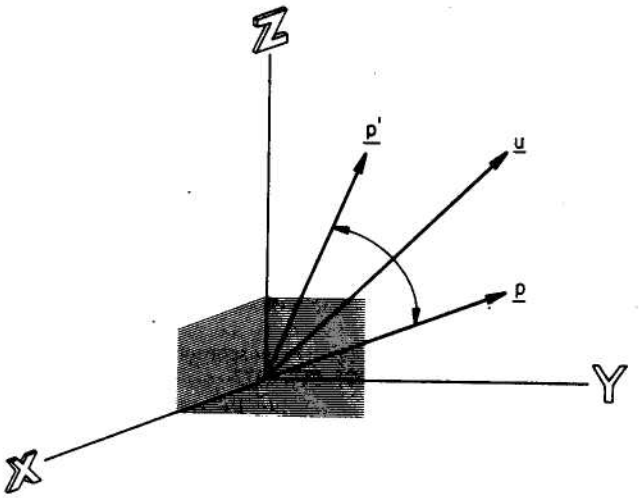


Fig. 1. Rotation about an axis by an angle. (Note: characters with underbars appear boldface in text.)

so that

$$p'' = q_0 p q_0^*$$

where

$$q_0 = q_2 q_1$$

using quaternion multiplication. $q_0$ can be easily separated into the axis and angle of the combined rotation.

## Rotation About an Axis Not Passing Through the Origin

To rotate a point about an axis not passing through the origin, there are two methods that can be combined with any rotation formula.

1) We can translate the point before and after, at an extra cost of $\tau_+ = 6$. Thus, to rotate point $\boldsymbol{p}$ with rotation matrix $R$ about the center of rotation $\boldsymbol{d}$ to give point $\boldsymbol{p}'$, we have

$$\boldsymbol{p}' = R(\boldsymbol{p} - \boldsymbol{d}) + \boldsymbol{d}. \tag{3}$$

2) If we are rotating several points with the same $R$ and $\boldsymbol{d}$, then it is cheaper to calculate

$$\boldsymbol{f} = \boldsymbol{d} - R\boldsymbol{d}$$

so that

$$\boldsymbol{p}' = R\boldsymbol{p} + \boldsymbol{f}. \tag{4}$$

The time to calculate (4) is three additions more than the time to calculate $R\boldsymbol{p}$, which varies for the different methods. Calculating $\boldsymbol{f}$ is the same as applying (4) once. Equation (4) is faster than (3) when many points are being rotated about the same axis through the same angle; the breakeven point depends on the particular method used. For example, for rotation by a general $3 \times 3$ matrix, and assuming that additions and multiplications cost the same, then (4) is faster than (3) if more than six points are being rotated.

## A More Efficient Formula

If we desire to multiply the same point or set of points by many angles, then the matrices in the above formulations must be recalculated anew for each angle. We will transform (1) above into a form more suited to iterated rotations by separating (1) into a combination of components depending on $\theta$ alone and components depending only on $\boldsymbol{p}$ and $\boldsymbol{u}$:

$$\boldsymbol{p1} = (\boldsymbol{p} \cdot \boldsymbol{u})\boldsymbol{u}$$

$$\boldsymbol{p2} = \boldsymbol{p} - \boldsymbol{p1}$$

$$\boldsymbol{p3} = \boldsymbol{u} \times \boldsymbol{p}$$

so that

$$\boldsymbol{p}' = \boldsymbol{p1} + \boldsymbol{p2} \cos\theta + \boldsymbol{p3} \sin\theta. \tag{5}$$

Now we need to compute $\boldsymbol{p1}$, $\boldsymbol{p2}$, and $\boldsymbol{p3}$ only once per point at a cost of $\tau_x = 12$ and $\tau_+ = 8$ if we are willing to store nine numbers per point. For simplicity, we assume that each number requires one word of storage. For each angle, we calculate and store sin $(\theta)$ and cos $(\theta)$. Then, for each (point, angle) pair in an iterated rotation, we need to spend only $\tau_x = 6$ and $\tau_+ = 6$. In (5), we can make space/time tradeoffs such as calculating $\boldsymbol{p2}$ instead of storing it to save three words per point memory, but requiring an extra $\tau_+ = 3$ to calculate each (point, angle) pair.

In contrast to the other methods, (5) is just as fast when the axis of rotation is not through the origin. Let

$$\boldsymbol{p1} = (\boldsymbol{p} - \boldsymbol{d}) \cdot \boldsymbol{u}\,\boldsymbol{u} + \boldsymbol{d}$$

$$\boldsymbol{p2} = \boldsymbol{p} - \boldsymbol{p1}$$

$$\boldsymbol{p3} = \boldsymbol{u} \times (\boldsymbol{p} - \boldsymbol{d}).$$

Then we still obtain

$$\boldsymbol{p}' = \boldsymbol{p1} + \boldsymbol{p2} \cos\theta + \boldsymbol{p3} \sin\theta. \tag{5}$$

If $\Delta\theta$ is fixed as we vary $\theta$, then there are two cases. If the points remain the same, then we can reapply (5) to the transformed points. Alternatively, if the points are different, as would happen if this rotation is only one of a sequence of transformations that are all varying then we can use either a lookup table for sin $(\theta)$ and cos $(\theta)$ or alternatively just calculate sin $(\theta + \Delta\theta)$ and cos $(\theta + \Delta\theta)$ with difference equations. If we use a fifth-order approximate equation for each component of $\boldsymbol{p}'$, then each iteration of each equation costs only five additions, so the total cost, including the trig functions, of evaluating (5) is $\tau_x = 0$ and $\tau_+ = 15$. Such an iteration would be more difficult with the rotation formats that combine three rotations about the coordinate axes. It is also possible to use the trignometric angle addition formulas at a cost of $\tau_x = 4$ and $\tau_+ = 2$:

$$\sin(\theta + \Delta\theta) = \sin\theta \cos\Delta\theta + \sin\Delta\theta \cos\theta$$

$$\cos(\theta + \Delta\theta) = \cos\theta \cos\Delta\theta - \sin\theta \sin\Delta\theta. \tag{6}$$

Assuming that sin $(\theta)$, cos $(\theta)$, sin $(\Delta\theta)$, and cos $(\Delta\theta)$ are already known, evaluating (5) at a new angle costs only $\tau_x = 10$ and $\tau_+ = 8$ and requires no more trig evaluations. Equations (6) also can be used for the other rotation formulas as long as they are defined in terms of only one angle.

Equation (5) also handles the case of several composed rotations, such as in a robot manipulator arm where we are iterating one of the angles. In matrix notation, let

$$\boldsymbol{p}' = \boldsymbol{p} R_1(\theta_1) R_2(\theta_2) \cdots R_k(\theta_k).$$

Each $R_i(\theta_i)$ represents a rotation about some fixed axis by the angle $\theta_i$. The rotations can be applied in order from $R_1$ to $R_k$, with each $R_i$ operating in the fixed world coordinate system or, equivalently, they can be applied in the reverse order from $R_k$ to $R_1$ in a reference frame rotating with the object. If we are varying only one of the $\theta_i$ while keeping the others fixed, then (5) still applies, with $\theta = \theta_i$. However, now the formulas for $\boldsymbol{p1}$, $\boldsymbol{p2}$, and $\boldsymbol{p3}$ are more complicated.

If we are varying all $k$ of the $\theta_i$ simultaneously, then there are two choices. Since $\boldsymbol{p}'$ is a multilinear function in all the sin $(\theta_i)$ and cos $(\theta_i)$, we could calculate the coefficients of the single resulting equation and evaluate it. However, it is more efficient to apply (5) separately for each $R_i$.

## Summary

Table I lists the number of additions and multiplications to rotate a point about an axis for the various methods, assuming that we are rotating many points through many angles about the same axis. Both the special case of the axis passing through the origin and the general case are given. Bookkeeping and setup costs are ignored, except for