# Rotation Representations and Performance Issues

David Eberly
Geometric Tools, LLC
http://www.geometrictools.com/


Created: January 21, 2002
Last Modified: March 1, 2008

## Contents

This document is a summary of representations of rotations by matrices, quaternions, or axis-angle pairs. Conversions between the representations is provided. The document also looks at performance issues by comparing the memory usage for each of the representations and by comparing the computation time needed to perform various operations such as rotating a vector, composing rotations, and interpolation of rotations.

# 1 Matrix Representation

A 2D rotation is a tranformation of the form

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

where $\theta$ is the angle of rotation. A 3D rotation is a 2D rotation that is applied within a specified plane that contains the origin. Such a rotation can be represented by a $3 \times 3$ *rotation matrix* $R = [\mathbf{R}_0\ \mathbf{R}_1\ \mathbf{R}_2]$ whose columns $\mathbf{R}_0$, $\mathbf{R}_1$, and $\mathbf{R}_2$ form a right-handed orthonormal set. That is, $|\mathbf{R}_0| = |\mathbf{R}_1| = |\mathbf{R}_2| = 1$, $\mathbf{R}_0 \cdot \mathbf{R}_1 = \mathbf{R}_0 \cdot \mathbf{R}_2 = \mathbf{R}_1 \cdot \mathbf{R}_2 = 0$, and $\mathbf{R}_0 \cdot \mathbf{R}_1 \times \mathbf{R}_2 = 1$. The columns of the matrix correspond to the final rotated values of the standard basis vectors $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$, in that order. Given a $3 \times 1$ vector $\mathbf{X} = [x_j]$ and $3 \times 3$ rotation matrix $R = [r_{ij}]$, the rotated vector is

$$R\mathbf{X} = \left[ \sum_{j=0}^{2} r_{ij} x_j \right]. \tag{1}$$

# 2 Axis-Angle Representation

If the plane of rotation has unit length normal $\mathbf{W}$, then the *axis-angle representation* of the rotation is the pair $\langle \mathbf{W}, \theta \rangle$. The direction of rotation is chosen so that as you look down on the plane from the side to which $\mathbf{W}$ points, the rotation is counterclockwise about the origin for $\theta > 0$. This is the same convention used for a 2D rotation.

## 2.1 Axis-Angle to Matrix

If $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$ form a right-handed orthonormal set, then any point can be represented as $\mathbf{X} = u_0\mathbf{U} + v_0\mathbf{V} + w_0\mathbf{W}$. Rotation of $\mathbf{X}$ about the axis $\mathbf{W}$ by the angle $\theta$ produces $R\mathbf{X} = u_1\mathbf{U} + v_1\mathbf{V} + w_1\mathbf{W}$. Clearly from the geometry, $w_1 = w_0 = \mathbf{W} \cdot \mathbf{X}$. The other two components are changed as if a 2D rotation has been applied to them, so $u_1 = \cos(\theta)u_0 - \sin(\theta)v_0$ and $v_1 = \sin(\theta)u_0 + \cos(\theta)v_0$. Using the right-handedness of the orthonormal set, it is easily shown that

$$\mathbf{W} \times \mathbf{X} = u_0\mathbf{W} \times \mathbf{U} + v_0\mathbf{W} \times \mathbf{V} + w_0\mathbf{W} \times \mathbf{W} = -v_0\mathbf{U} + u_0\mathbf{V}$$

and

$$\mathbf{W} \times (\mathbf{W} \times \mathbf{X}) = -v_0\mathbf{W} \times \mathbf{U} + u_0\mathbf{W} \times \mathbf{V} = -u_0\mathbf{U} - v_0\mathbf{V}.$$

Combining these in the form shown and using the relationship between $u_0$, $v_0$, $u_1$, and $v_1$ produces

$$\begin{aligned}(\sin\theta)\mathbf{W}\times\mathbf{X}+(1-\cos\theta)\mathbf{W}\times(\mathbf{W}\times\mathbf{X}) &= (-v_0\sin\theta-u_0(1-\cos\theta))\mathbf{U}+(u_0\sin\theta-v_0(1-\cos\theta))\mathbf{V}\\ &= (u_1-u_0)\mathbf{U}+(v_1-v_0)\mathbf{V}\\ &= R\mathbf{X}-\mathbf{X}.\end{aligned}$$

Therefore, the rotation of $\mathbf{X}$ given the axis $\mathbf{W}$ and angle $\theta$ is

$$R\mathbf{X}=\mathbf{X}+(\sin\theta)\mathbf{W}\times\mathbf{X}+(1-\cos\theta)\mathbf{W}\times(\mathbf{W}\times\mathbf{X}). \tag{2}$$

This can also be written in matrix form by defining the following where $\mathbf{W}=(a,b,c)$,

$$S=\begin{bmatrix}0 & -c & b\\ c & 0 & -a\\ -b & a & 0\end{bmatrix}$$

in which case

$$R=I+(\sin\theta)S+(1-\cos\theta)S^2$$

and consequently $R\mathbf{X}=\mathbf{X}+(\sin\theta)S\mathbf{X}+(1-\cos\theta)S^2\mathbf{X}$.

## 2.2 Matrix to Axis-Angle

The inverse problem is to start with the rotation matrix and extract an angle and unit-length axis. There are multiple solutions since $-\mathbf{W}$ is a valid axis whenever $\mathbf{W}$ is and $\theta+2\pi k$ is a valid solution whenever $\theta$ is. First, the *trace* of a matrix is defined to be the sum of the diagonal terms. Some algebra will show that $\cos\theta=(\text{trace}(R)-1)/2$, in which case

$$\theta=\cos^{-1}((\text{trace}(R)-1)/2)\in[0,\pi]. \tag{3}$$

Also, it is easily shown that

$$R-R^{\mathsf{T}}=(2\sin\theta)S \tag{4}$$

where $S$ is a skew-symmetric matrix. The constructions below are based on the cases $\theta=0$, $\theta\in(0,\pi)$, and $\theta=\pi$.

If $\theta=0$, then any unit-length diretion vector for the axis is valid since there is no rotation.

If $\theta\in(0,\pi)$, equation (4) allows direct extraction of the axis, $\mathbf{D}=(r_{21}-r_{12},r_{02}-r_{20},r_{10}-r_{01})$ and $\mathbf{W}=\mathbf{D}/|\mathbf{D}|$.

If $\theta=\pi$, equation (4) does not help with constructing the axis since $R-R^{\mathsf{T}}=0$. In this case note that

$$R=I+2S^2=\begin{bmatrix}1-2(w_1^2+w_2^2) & 2w_0w_1 & 2w_0w_2\\ 2w_0w_1 & 1-2(w_0^2+w_2^2) & 2w_1w_2\\ 2w_0w_2 & 2w_1w_2 & 1-2(w_0^2+w_1^2)\end{bmatrix}$$

where $\mathbf{W} = (w_0, w_1, w_2)$. The idea is to extract the maximum component of the axis from the diagonal entries of the rotation matrix. If $r_{00}$ is maximum, then $w_0$ must be the largest component in magnitude. Compute $4w_0^2 = r_{00} - r_{11} - r_{22} + 1$ and select $w_0 = \sqrt{r_{00} - r_{11} - r_{22} + 1}/2$. Consequently, $w_1 = r_{01}/(2w_0)$ and $w_2 = r_{02}/(2w_0)$. If $r_{11}$ is maximum, then compute $4w_1^2 = r_{11} - r_{00} - r_{22} + 1$ and select $w_1 = \sqrt{r_{11} - r_{00} - r_{22} + 1}/2$. Consequently, $w_0 = r_{01}/(2w_1)$ and $w_2 = r_{12}/(2w_1)$. Finallly, if $r_{22}$ is maximum, then compute $4w_2^2 = r_{22} - r_{00} - r_{11} + 1$ and select $w_2 = \sqrt{r_{22} - r_{00} - r_{11} + 1}/2$. Consequently, $w_0 = r_{02}/(2w_2)$ and $w_1 = r_{12}/(2w_2)$.

# 3 Quaternion Representation

A third representation involves *unit quaternions*. Only a summary is provided here. A unit quaternion is denoted by $q = w + xi + yj + zk$ where $w$, $x$, $y$, and $z$ are real numbers and where the 4-tuple $(w, x, y, z)$ is unit length. The set of unit quaternions is just the unit hypersphere in $\mathbb{R}^4$. The products of $i$, $j$, and $k$ are defined by $i^2 = j^2 = k^2 = -1$, $ij = -ji = k$, $jk = -kj = i$, and $ki = -ik = j$. Observe that the products are *not commutative*. The product of two unit quaternions $q_n = w_n + x_n i + y_n j + z_n k$ for $n = 0, 1$ is defined by distributing the product over the sums, keeping in mind that the order of operands is important:

$$\begin{aligned} q_0 q_1 \quad = \quad & (w_0 w_1 - x_0 x_1 - y_0 y_1 - z_0 z_1) + \\ & (w_0 x_1 + x_0 w_1 + y_0 z_1 - z_0 y_1) i + \\ & (w_0 y_1 - x_0 z_1 + y_0 w_1 + z_0 x_1) j + \\ & (w_0 z_1 + x_0 y_1 - y_0 x_1 + z_0 w_1) k. \end{aligned}$$

The conjugate of $q$ is defined by

$$q^* = w - xi - yj - zk.$$

Observe that $qq^* = q^*q = 1$ where the right-hand side 1 is the $w$-term of the quaternion, the $x$-, $y$-, and $z$-terms being all 0.

## 3.1 Axis-Angle to Quaternion

If $\mathbf{A} = (x_0, y_0, z_0)$ is the unit length axis of rotation and if $\theta$ is the angle of rotation, a quaternion $q = w + xi + yj + zk$ that represents the rotation satisfies $w = \cos(\theta/2)$, $x = x_0 \sin(\theta/2)$, $y = y_0 \sin(\theta/2)$, and $z = z_0 \sin(\theta/2)$.

If a vector $\mathbf{V} = (v_0, v_1, v_2)$ is represented as the quaternion $\hat{v} = v_0 i + v_1 j + v_2 k$, and if $q$ represents a rotation, then the rotated vector $\mathbf{U}$ is represented by quaternion $\hat{u} = u_0 i + u_1 j + u_2 k$ where

$$\hat{u} = q\hat{v}q^*. \tag{5}$$

It can be shown that the $w$-term of $\hat{u}$ must really be 0.

## 3.2 Quaternion to Axis-Angle

Let $q = w + xi + yj + zk$ be a unit quaternion. If $|w| = 1$, then the angle is $\theta = 0$ and any unit-length direction vector for the axis will do since there is no rotation. If $|w| < 1$, the angle is obtained as $\theta = 2\cos^{-1}(w)$ and the axis is computed as $\mathbf{A} = (x, y, z)/\sqrt{1 - w^2}$.

### 3.3 Quaternion to Matrix

Using the identities $2\sin^2(\theta/2) = 1 - \cos(\theta)$ and $\sin(\theta) = 2\sin(\theta/2)\cos(\theta/2)$, it is easily shown that $2wx = (\sin\theta)w_0$, $2wy = (\sin\theta)w_1$, $2wz = (\sin\theta)w_2$, $2x^2 = (1-\cos\theta)w_0^2$, $2xy = (1-\cos\theta)w_0w_1$, $2xz = (1-\cos\theta)w_0w_2$, $2y^2 = (1-\cos\theta)w_1^2$, $2yz = (1-\cos\theta)w_1w_2$, and $2z^2 = (1-\cos\theta)w_2^2$. The right-hand sides of all these equations are terms in the expression $R = I + (\sin\theta)S + (1-\cos\theta)S^2$. Replacing them yields

$$R = \begin{bmatrix} 1-2y^2-2z^2 & 2xy-2wz & 2xz+2wy \\ 2xy+2wz & 1-2x^2-2z^2 & 2yz-2wx \\ 2xz-2wy & 2yz+2wx & 1-2x^2-2y^2 \end{bmatrix} \tag{6}$$

### 3.4 Matrix to Quaternion

Earlier it was mentioned that $\cos\theta = (\text{trace}(R)-1)/2$. Using the identity $2\cos^2(\theta/2) = 1+\cos\theta$ yields $w^2 = \cos^2(\theta/2) = (\text{trace}(R)+1)/4$ or $|w| = \sqrt{\text{trace}(R)+1}/2$. If $\text{trace}(R) > 0$, then $|w| > 1/2$, so without loss of generality choose $w$ to be the positive square root, $w = \sqrt{\text{trace}(R)+1}/2$. The identity $R - R^{\mathsf{T}} = (2\sin\theta)S$ also yielded $(r_{12}-r_{21}, r_{20}-r_{02}, r_{01}-r_{10}) = 2\sin\theta(w_0, w_1, w_2)$. Finally, identities derived earlier were $2xw = w_0\sin\theta$, $2yw = w_1\sin\theta$, and $2zw = w_2\sin\theta$. Combining these leads to $x = (r_{12}-r_{21})/(4w)$, $y = (r_{20}-r_{02})/(4w)$, and $z = (r_{01}-r_{10})/(4w)$.

If $\text{trace}(R) \le 0$, then $|w| \le 1/2$. The idea is to first extract the largest one of $x$, $y$, or $z$ from the diagonal terms of the rotation $R$ in equation 6. If $r_{00}$ is the maximum diagonal term, then $x$ is larger in magnitude than $y$ or $z$. Some algebra shows that $4x^2 = r_{00} - r_{11} - r_{22} + 1$ from which is chosen $x = \sqrt{r_{00}-r_{11}-r_{22}+1}/2$. Consequently, $w = (r_{12}-r_{21})/(4x)$, $y = (r_{01}+r_{10})/(4x)$, and $z = (r_{02}+r_{20})/(4x)$. If $r_{11}$ is the maximum diagonal term, then compute $4y^2 = r_{11} - r_{00} - r_{22} + 1$ and choose $y = \sqrt{r_{11}-r_{00}-r_{22}+1}/2$. Consequently, $w = (r_{20}-r_{02})/(4y)$, $x = (r_{01}+r_{10})/(4y)$, and $z = (r_{12}+r_{21})/(4y)$. Finally, if $r_{22}$ is the maximum diagonal term, then compute $4z^2 = r_{22} - r_{00} - r_{11} + 1$ and choose $z = \sqrt{r_{22}-r_{00}-r_{11}+1}/2$. Consequently, $w = (r_{01}-r_{10})/(4z)$, $x = (r_{02}+r_{20})/(4z)$, and $y = (r_{12}+r_{21})/(4z)$.

## 4 Performance Issues

A question asked quite often is "What is the best representation to use for rotations"? As with most computer science topics, there is no answer to this question, only trade offs to consider. In the discussion, the rotation matrix is $R$, the quaternion is $q$, and the axis-angle pair is $(\mathbf{A}, \theta)$. Various high level operations are compared by a count of low level operations including multiplication (M), addition or subtraction (A), division (D), and expensive math library function evaluations (F). In an actual implementation, comparisons (C) should also be counted because they can be even more expensive than multiplications and/or additions. Summary tables are provided to allow you to quickly compare the performance.

### 4.1 Memory Usage

A rotation matrix requires 9 floats, a quaternion requires 4 floats, and an axis-angle pair requires 4 floats, so clearly the rotation matrix will use more memory. Storing only the angle in the axis-angle formulation is

clearly not helpful when transforming is required since you need to know the values of $\sin\theta$ and $1-\cos\theta$. Evaluating the trigonometric functions is quite expensive. It is better to precompute both quantities and store them, so in fact an axis-angle pair will require 6 floats, making the quaternion representation the cheapest in memory usage. Table 1 is a summary of the memory usage. The axis-angle count includes 3

| representation | floats | comments |
|---|---|---|
| rotation matrix | 9 | |
| axis-angle | 4 | no precompute of $\sin\theta$ or $1-\cos\theta$ |
| axis-angle | 6 | precompute of $\sin\theta$ and $1-\cos\theta$ |
| quaternion | 4 | |

Table 1: Comparison of memory usage.

floats for the axis, 1 float for the angle $\theta$, and 2 floats for $\sin\theta$ and $1-\cos\theta$. Without the precomputation of the trigonometric functions, any operation requiring the function values will be quite expensive.

## 4.2 Conversion Time

Applications using rotations invariably have to convert from one representation to another, so it is useful to have measurements of costs for the conversions. The entities involved are a rotation matrix $R$, an axis-angle pair $(\mathbf{A},\theta)$, and a quaternion $q$. It is assumed that the angle of rotation is in $(0,\pi)$.

### 4.2.1 Axis-Angle to Matrix

Evaluation of $\sigma=\sin(\theta)$ and $\gamma=\cos(\theta)$ requires two function calls. The term $1-\gamma$ requires 1 addition. The skew-symmetric matrix $S$ obtained from $\mathbf{A}$ requires no computation. The matrix $S^2$ requires 6 unique multiplications and 3 additions; sign changes are not counted. The term $(1-\gamma)S^2$ requires 6 unique multiplications. The term $\sigma S$ requires 3 unique multiplications. Finally, the combination $R=I+\sigma S+(1-\gamma)S^2$ uses 9 additions. The total cost is $13A+15M+2F$.

### 4.2.2 Matrix to Axis-Angle

The extraction $\theta=\cos^{-1}((\text{trace}(R)-1)/2)$ requires 3 additions, 1 multiplication, and 1 function call. The vector $\mathbf{D}=(r_{21}-r_{12},r_{02}-r_{20},r_{10}-r_{01})$ requires 3 additions. The normalized vector $\mathbf{A}=\mathbf{D}/|\mathbf{D}|$ requires 6 multiplications, 2 additions, 1 division, and 1 function call. The total cost is $8A+7M+1D+2F$.

### 4.2.3 Axis-Angle to Quaternion

Extracting $\theta=2\cos^{-1}(w)$ requires 1 function call and 1 multiplication. Constructing $\mathbf{A}=(x,y,z)/\sqrt{1-w^2}$ requires 4 multiplications, 1 addition, 1 division, and 1 function call. The total cost is $1A+5M+1D+2F$.

#### 4.2.4 Quaternion to Axis-Angle

Evaluation of $\theta/2$ uses 1 multiplication. Evalution of $\sigma = \sin(\theta/2)$ and $w = \cos(\theta/2)$ requires 2 function calls. The products $(x, y, z) = \sigma\mathbf{A}$ require 3 multiplications. The total cost is $4M + 2F$.

#### 4.2.5 Quaternion to Matrix

The conversion requires 12 multiplications. The terms $t_x = 2x$, $t_y = 2y$, and $t_z = 2z$ are computed. From these the following terms are computed: $t_{wx} = wt_x$, $t_{wy} = wt_y$, $t_{wz} = wt_z$, $t_{xx} = t_x x$, $t_{xy} = xt_y$, $t_{xz} = xt_z$, $t_{yy} = t_y y$, $t_{yz} = yt_z$, and $t_{zz} = t_z z$. The rotation matrix entries require 12 additions: $r_{00} = 1 - t_{yy} - t_{zz}$, $r_{01} = t_{xy} - t_{wz}$, $r_{02} = t_{xz} + t_{wy}$, $r_{10} = t_{xy} + t_{wz}$, $r_{11} = 1 - t_{xx} - t_{zz}$, $r_{12} = t_{yz} - t_{wx}$, $r_{20} = t_{xz} - t_{wy}$, $r_{21} = t_{yz} + t_{wx}$, and $r_{22} = 1 - t_{xx} - t_{yy}$. The total cost is $12A + 12M$.

#### 4.2.6 Matrix to Quaternion

The conversion depends on the sign of the trace of $R$. Computing the trace $\tau = \text{trace}(R)$ requires 2 additions. Suppose that $\tau > 0$ (this comparison is $1C$ in cost). The calculation $w = \sqrt{\tau + 1}/2$ requires 1 addition, 1 multiplication, and 1 function call. The expression $\lambda = 1/(4w)$ requires 1 multiplication and 1 division. The terms $x = \lambda(r_{12} - r_{21})$, $y = \lambda(r_{20} - r_{02})$, and $z = \lambda(r_{01} - r_{10})$ require 3 additions and 3 multiplications. The total cost is $6A + 5M + 1D + 1F + 1C$.

If $\tau \leq 0$, the maximum of the diagonal entries of the rotation matrix must be found. This requires two comparisons, call this cost $2C$. For the sake of argument, suppose that $r_{00}$ is the maximum. The calculation $x = \sqrt{r_{00} - r_{11} - r_{22} + 1}/2$ requires 3 additions, 1 multiplication, and 1 function call. The expression $\lambda = 1/(4x)$ requires 1 multiplicaion and 1 division. The terms $w = \lambda(r_{12} - r_{21})$, $y = \lambda(r_{01} + r_{10})$, and $z = \lambda(r_{02} + r_{20})$ require 3 additions and 3 multiplications. The total cost is $6A + 5M + 1D + 1F + 3C$.

Table 2 is a summary of the costs of converting among the various rotation representations.

| conversion | A | M | D | F | C |
|---|---|---|---|---|---|
| axis-angle to matrix | 13 | 15 | | 2 | |
| matrix to axis-angle | 8 | 7 | 1 | 2 | |
| axis-angle to quaternion | 1 | 5 | 1 | 2 | |
| quaternion to axis-angle | | 4 | | 2 | |
| quaternion to matrix | 12 | 12 | | | |
| matrix to quaternion ($\tau > 0$) | 6 | 5 | 1 | 1 | 1 |
| matrix to quaternion ($\tau \leq 0$) | 6 | 5 | 1 | 1 | 3 |

Table 2: Comparison of operation counts for converting between representations of rotations.

### 4.3 Transformation Time

The transformation of $\mathbf{V}$ by a rotation matrix is the product $\mathbf{U} = R\mathbf{V}$ and requires 9 multiplications and 6 additions for a total of 15 operations.

If $\mathbf{V} = (v_0, v_1, v_2)$ and if $\hat{V} = v_0 i + v_1 j + v_2 k$ is the corresponding quaternion with zero $w$ component, then the rotate vector $\mathbf{U} = (u_0, u_1, u_2)$ is computed as $\hat{U} = u_0 i + u_1 j + u_2 k = q\hat{V}q^*$. Applying the general formula for quaternion multiplication directly, the product $p = q\hat{V}$ requires 16 multiplications and 12 additions. The product $pq^*$ also uses the same number of operations. The total operation count is 56. However, since $\hat{V}$ has no $w$ term, $p$ only requires 12 multiplications and 8 additions–one term is theoretically zero, so no need to compute it. We also know that $\hat{U}$ has no $w$ term, so the product $pq^*$ only requires 12 multiplications and 9 additions. Using these optimizations, the total operation count is 41. Observe that conversion from quaternion $q$ to rotation matrix $R$ requires 12 multiplications and 12 additions. Transforming $\mathbf{V}$ by $R$ takes 15 operations. Therefore, the process of converting to rotation and multiplying uses 39 operations, two less than calculating $q\hat{V}q^*$. Purists who implement quaternion libraries and only use quaternions will sadly lose a lot of cycles when transforming large sets of vertices.

The formula for transforming $\mathbf{V}$ using an axis-angle pair is

$$R\mathbf{V} = \mathbf{V} + (\sin\theta)\mathbf{A} \times \mathbf{V} + (1 - \cos\theta)\mathbf{A} \times (\mathbf{A} \times \mathbf{V}).$$

As indicated earlier, $\sin\theta$ and $1 - \cos\theta$ should be precomputed and stored in adition to the axis and angle, a total of 6 floats. The cross product $\mathbf{A} \times \mathbf{V}$ uses 6 multiplications and 3 additions. So does $\mathbf{A} \times (\mathbf{A} \times \mathbf{V})$, assuming the cross product in the parentheses was computed first and stored in temporary memory. Multiplying the cross products by a scalar requires 6 multiplications. Adding three vectors requires 6 additions. Therefore, we need to use 18 multiplications and 12 additions for a total of 30 operations.

Therefore, the rotational formulation yields the fastest transforming. The quaternion formulation yields the slowest transforming for a single vector. But keep in mind that a batch transform of $n$ vectors requires converting the quaternion to a rotation matrix only once at a cost of 24 operations. The total operations for transforming by quaternion is $24 + 15n$. The axis-angle formulation uses $30n$, so the quaternion transformation is faster for two or more vectors. Table 3 is a summary of the operation counts for transforming a single vector.

| representation | A | M | comments |
|---|---|---|---|
| rotation matrix | 6 | 9 | |
| axis-angle | 12 | 18 | |
| quaternion | 24 | 32 | using generic quaternion multiplies |
| quaternion | 17 | 24 | using specialized quaternion multiplies |
| quaternion | 18 | 21 | convert to matrix, then multiply |

Table 3: Comparison of operation counts for transforming 1 vector.

Table 4 is a summary of the operation counts for transforming $n$ vectors.

| representation | A | M | comments |
|---|---|---|---|
| rotation matrix | 6n | 9n | |
| axis-angle | 12n | 18n | |
| quaternion | 24n | 32n | using generic quaternion multiplies |
| quaternion | 17n | 24n | using specialized quaternion multiplies |
| quaternion | 12+6n | 12+9n | convert to matrix, then multiply |

Table 4: Comparison of operation counts for transforming $n$ vector.

## 4.4 Composition

The product of two rotation matrices requires 27 multiplications and 18 additions for a total cost of $18A + 27M$.

The product of two quaternions requires 16 multiplications and 12 additions for a total cost of $12A + 16M$, clearly outperforming matrix multiplication. Moreover, renormalizing a quaternion to adjust for floating point errors is cheaper than renormalizing a rotation matrix using Gram-Schmidt orthonormalization.

Composition of two axis-angle pairs is unthinkable in an application that requires computational efficiency. One way to do the composition is to convert to matrices, multiply the matrices, then extract the axis-angle pair. The two conversions from axis-angle to matrix cost $26A + 30M + 4F$, the matrix product costs $18A + 27M$, and the conversion from matrix to axis-angle costs $8A + 7M + 1D + 2F$. The total cost is $52A + 64M + 1D + 6F$.

Another way to do the composition of two axis-angle pairs is to convert to quaternions, multiply the quaternions, then extract the axis-angle pair. The two conversions from axis-angle to quaternion cost $2A + 10M + 2D + 4F$, the quaternion product costs $12A + 16M$, and the conversion from quaternion to axis-angle costs $4M + 2F$. The total cost is $14A + 30M + 2D + 6F$.

Table 5 is a summary of the operation counts for composing two rotations.

| representation | A | M | D | F |
|---|---|---|---|---|
| rotation matrix | 18 | 27 | | |
| quaternion | 12 | 16 | | |
| axis-angle (convert to matrix) | 52 | 64 | 1 | 6 |
| axis-angle (convert to quaternion) | 14 | 30 | 2 | 6 |

Table 5: Comparison of operation counts for composition.

## 4.5 Interpolation

### 4.5.1 Quaternion Interpolation

Quaternions are quite amenable to interpolation. The standard operation that is used is *spherical linear interpolation*, affectionately known as *slerp*. Given quaternions $p$ and $q$ with acute angle $\theta$ between them, slerp is defined as $s(t;p,q) = p(p^*q)^t$ for $t \in [0,1]$. Note that $s(0;p,q) = p$ and $s(1;p,q) = q$. An equivalent definition of slerp that is more amenable to calculation is

$$s(t;p,q) = \frac{\sin((1-t)\theta)p + \sin(t\theta)q}{\sin(\theta)}.$$

If $p$ and $q$ are thought of as points on a unit circle, the formula above is a parameterization of the shortest arc between them. If a particle travels on that curve according to the parameterization, it does so with constant speed. Thus, any uniform sampling of $t$ in $[0,1]$ produces equally spaced points on the arc.

We assume that only $p$, $q$, and $t$ are specified. Moreover, since $q$ and $-q$ represent the same rotation, you can replace $q$ by $-q$ if necessary to guarantee that the angle between $p$ and $q$ treated as 4-tuples is acute. That is, $p \cdot q \geq 0$. As 4-tuples, $p$ and $q$ are unit length. The dot product is therefore $p \cdot q = \cos(\theta)$. Table 6 shows the operation counts. Any term shown on the left that includes an already computed term has only its *additional* operations counted to avoid double counting operations.

| term | A | M | D | F |
|---|---|---|---|---|
| $a_0 = p \cdot q$ | 3 | 4 | | |
| $\theta = \cos^{-1}(a_0)$ | | | | 1 |
| $1 - t$ | 1 | | | |
| $(1-t)\theta$ | | 1 | | |
| $t\theta$ | | 1 | | |
| $\sin(\theta)$ | | | | 1 |
| $\sin((1-t)\theta)$ | | | | 1 |
| $\sin(t\theta)$ | | | | 1 |
| $a_1 = 1/\sin(\theta)$ | | | 1 | |
| $a_2 = a_1 \sin((1-t)\theta)$ | | 1 | | |
| $a_3 = a_1 \sin(t\theta)$ | | 1 | | |
| $a_2 p + a_3 q$ | 4 | 8 | | |
| total | 8 | 16 | 1 | 4 |

Table 6: Operation counts for quaternion interpolation.

### 4.5.2 Rotation Matrix Interpolation

The absence of a meaningful interpolation formula that directly applies to rotation matrices is used as an argument for the superiority of quaternions over rotation matrices. However, rotations can be interpolated directly in a way equivalent to what slerp produces. If $P$ and $Q$ are rotations corresponding to quaternions $p$ and $q$, the slerp of the matrices is

$$S(t; P, Q) = P(P^{\mathsf{T}}Q)^t,$$

the same formula that defines slerp for quaternions. The technical problem is to define what is meant by $R^t$ for a rotation $R$ and real-valued $t$. If the rotation has axis $\mathbf{A}$ and angle $\theta$, then $R^t$ has the same rotation axis, but the angle of rotation is $t\theta$. The procedure for computing slerp of the rotation matrices is

1. Compute $R = P^{\mathsf{T}}Q$.
2. Extract an axis $\mathbf{A}$ and an angle $\theta$ from $R$.
3. Compute $R^t$ by converting the axis-angle pair $\mathbf{A}$, $t\theta$.
4. Compute the $S(t; P, Q) = PR^t$.

This algorithm requires an axis-angle extraction that involves an inverse trigonometric function call and a square root operation, a couple of trigonometric evaluations (for $t\theta$), and a conversion back to a rotation matrix. This is quite a bit more expensive than computing the slerp for quaternions which requires three trigonometric function calls. The quaternion interpolation is therefore more efficient, but a purist wishing to avoid quaternions in an application has, indeed, a method for interpolating rotation matrices.

Table 7 shows the operation counts and uses the same format and rules as the table for quaternion interpolation.

Both the quaternion and rotation matrix interpolation use 1 division and 4 function evaluations. However, the number of additions and multiplications in the rotation matrix interpolation is excessive compared to that of quaternion interpolation.

### 4.5.3 Axis-Angle Interpolation

There is no obvious and natural way to produce the same interpolation that occurs with quaternions and rotation matrices. The only choice is to convert to one of the other representations, interpolate in that form, then convert the interpolated result back to axis-angle form. A very expensive proposition, just as in composition of rotations.

| term | A | M | D | F |
|---|---|---|---|---|
| $R = P^*Q$ | 18 | 27 | | |
| $a_0 = 0.5(\text{trace}(R) - 1)$ | 4 | 1 | | |
| $\theta = \cos^{-1}(a_0)$ | | | | 1 |
| $\mathbf{D} = (r_{21} - r_{12}, r_{02} - r_{20}, r_{10} - r_{01})$ | 3 | | | |
| $a_1 = 1/\lvert\mathbf{D}\rvert$ | 2 | 3 | 1 | 1 |
| $\mathbf{A} = a_1\mathbf{D}$ | | 3 | | |
| $t\theta$ | | 1 | | |
| $a_2 = \sin(t\theta)$ | | | | 1 |
| $a_3 = 1 - \cos(t\theta)$ | 1 | | | 1 |
| matrix $S$, no cost | | | | |
| $S^2$ | 3 | 6 | | |
| $R^t = I + a_2S + a_3S^2$ | 9 | 9 | | |
| $PR^t$ | 18 | 27 | | |
| total | 58 | 77 | 1 | 4 |

Table 7: Operation counts for rotation matrix interpolation.