Rotation with Quaternions

Contents

Department of Computer Science
Concordia University
December 2001

1 Introduction

Rotation is weird. One example is the *plate trick*. Rest a flat object, such as a plate in the palm of your right hand. Start rotating it anticlockwise: your hand will pass between your body and your elbow; continue rotating anticlockwise, and you hand will go up above your head and will eventually return to the starting position. With care, you can avoid dropping the plate — which has turned through 720°! (Note that it is impossible to turn the plate through 360° and end up in the position that you started in.)

1.1 Translation

These notes are about rotation but, before investigating rotation, we consider a simpler problem: translation.

Translation simply means moving an object *without* rotating it. Finite translations in three-dimensional Euclidean space form a *group* because:

- the result of applying one translation then another is a translation (closure);
- for every translation, there is another translation that reverses its effect (inverse);
- there is a translation that has no effect (unit element).

Knowing that translations form a group does not help us to represent them. Fortunately, there is an easy way of representing a translation. We introduce a coordinate frame with three axes and represent a translation by its components in the direction of each axis. For example, the expression $(1, 2, 3)$ means "translate 1 unit along the X-axis, 2 units along the Y-axis, and 3 units along the Z-axis". In fact, we are representing translations as vectors.

Translations in this representation can be added simply by adding their components:

$$(1, 2, 3) + (2, -4, 1) \quad = \quad (3, -2, 4).$$

Addition of translations is commutative because addition of reals is commutative. We can even express the resolution of a translation into its components with this notation:

$$(1, 2, 3) \quad = \quad (1, 0, 0) + (0, 1, 0) + (0, 0, 1).$$

The unit element of the group is the translation $(0, 0, 0$, which has no effect.

Matrices provide an alternative representation of translations. Using four-dimensional homogeneous coordinates, the translation (x, y, z) is represented by the matrix

$$\begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To combine two translations, we *multiply* the corresponding matrices rather than *adding* the vectors. Although matrix multiplication is not commutative in general, translation matrices do commute. It is easy to verify that

$$\begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & x' \\ 0 & 1 & 0 & y' \\ 0 & 0 & 1 & z' \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x+x' \\ 0 & 1 & 0 & y+y' \\ 0 & 0 & 1 & z+z' \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x' \\ 0 & 1 & 0 & y' \\ 0 & 0 & 1 & z' \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In computer graphics, we animate objects by applying a sequence of translations to them. The properties of translations make animation simple to implement. Suppose $\mathbf{M_1}$ and $\mathbf{M_2}$ are two translation matrices. Then the matrix $(1-t)\mathbf{M_1}+t\mathbf{M_2}$, where $0 \le t \le 1$, represents an intermediate translation. By allowing t to vary from 0 to 1, we can make an object move in a straight line joining the positions corresponding to $\mathbf{M_1}$ and $\mathbf{M_2}$ respectively.

1.2 Rotation

To describe a rotation, we need an *axis* and an *angle*. The rotation consists of moving the object through the given angle while keeping the axis fixed. For example, during each hour, the earth rotates through 15° about an axis that passes through the North and South poles. Like translations, rotations form a group. The group is called the symmetric orthogonal group in three dimensions or, more briefly, $SO(3)$.

The important characteristic of the axis is its direction, which we can represent with a vector (usually, although not necessarily, a unit vector). Thus we can define a rotation as a pair, $(\theta, \mathbf{u})$, in which θ is an angle and $\mathbf{u}$ is a unit vector. In fact, we specify a rotation in OpenGL by writing

```
glRotatef(theta, x, y, z);
```

in which the vector is represented by its components (x, y, z). However, it is difficult to work with this representation. For example, it is not obvious how to convert two calls to `glRotatef` into a single call, even though this is always possible in principle.

We can also represent rotations using 4×4 matrices (actually, 3×3 matrices would do, but we use homogeneous coordinates so that we can combine rotations, translations, and other transformations in a simple and consistent way). The effect of the OpenGL function `glRotate` is in fact to construct a matrix and then to multiply the current matrix by it. The matrix constructed by the call above is

$$\begin{bmatrix} x^2+\cos\theta(1-x^2) & xy(1-\cos\theta)-z\sin\theta & zx(1-\cos\theta)+y\sin\theta & 0 \\ xy(1-\cos\theta)+z\sin\theta & y^2+\cos\theta(1-y^2) & yz(1-\cos\theta)-x\sin\theta & 0 \\ zx(1-\cos\theta)-y\sin\theta & yz(1-\cos\theta)+x\sin\theta & z^2+\cos\theta(1-z^2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Just as we resolve translations along the principal axes, we can consider arbitrary rotations as being built up from rotations about the principal axes. In Figure 1, the OpenGL call on the left corresponds to the matrix on the right.

To obtain a general rotation, we take three angles, θ, ϕ, and ψ, corresponding to rotations about the axes X, Y, and Z respectively, and multiply the matrices together. This gives [1, Exercise 5.15]

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

By choosing suitable values of θ, ϕ, and ψ, we can produce any rotation. The technique was discovered by Leonhard Euler (1707–1783) and the angles are called ***Euler angles***. It begins to look as if rotations behave like translations. Unfortunately, however, there are problems, due mainly to the fact the rotations do not commute.

`glRotatef(theta, 1, 0, 0);`	$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
`glRotatef(phi, 0, 1, 0);`	$\begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 0 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
`glRotatef(psi, 0, 0, 1);`	$\begin{bmatrix} \cos\psi & -\sin\psi & 0 & 0 \\ \sin\psi & \cos\psi & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Figure 1: Rotation about a principal axis

1. The first problem is that the order in which we apply the rotations about the principal axes affects the final result. This means that we have to establish a conventional sequence and then stick to it.

2. A more serious problem is that successive rotations can interfere with each other. Suppose we put $\phi = 90°$, so that $\sin\phi = 1$ and $\cos\phi = 0$. Then R becomes

$$\begin{bmatrix} 0 & 0 & -1 & 0 \\ \sin\theta\cos\psi - \cos\theta\sin\psi & \sin\theta\sin\psi + \cos\theta\cos\psi & 0 & 0 \\ \cos\theta\cos\psi + \sin\theta\sin\psi & \cos\theta\sin\psi - \sin\theta\cos\psi & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 & 0 \\ \sin\delta & \cos\delta & 0 & 0 \\ \cos\delta & -\sin\delta & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

   where $\delta = \theta - \psi$. Intuitively, what has happened is that, by rotating 90° about the Y axis, we have made the X and Z axes parallel. Consequently, rotating by ψ is now the same as rotating by $-\theta$ and we have lost the ability to construct arbitrary rotations. This phenomenon is often called ***gimbal lock*** because there is a corresponding problem in which a gyroscope, supported by three gimbals that allow it to rotate about the principal axes, gets into a position where it cannot rotate freely because two axes have become parallel.

3. There is no easy way to interpolate between two rotation matrices. If $\mathbf{M_1}$ and $\mathbf{M_2}$ are rotation matrices, the matrix $(1-t)\mathbf{M_1} + t\mathbf{M_2}$ (which we used above to interpolate translations) will not give a smooth movement from $\mathbf{M_1}$ to $\mathbf{M_2}$. In fact, the interpolated matrix is not in general a rotation matrix, which means that the rotating object will be distorted as it moves.

4. Combining rotations requires matrix multiplication. Combining a long sequence of matrix multiplications creates inaccuracies and sometimes even numerical instability. Numerical inaccuracies appear in the graphic images as distortions of the objects.

5. A rotation matrix has 16 components. Although seven of these are constant (0 or 1), we are still using 16 numbers when only four should be necessary (three to define the axis and one to define the amount of the rotation).

We need a better solution. Fortunately, there is one: we can use quaternions.

2 Quaternions

William Rowan Hamilton (1805–65) was an Irish mathematician. Although we do not know whether he had a graphics workstation — it seems unlikely — we do know that he was deeply interested in rotation. He knew, because it was well-known at the time, that complex numbers provide an elegant representation for rotation in two dimensions. The effect of multiplying a complex number $\mathbf{z}$ by $\cos\theta + \mathbf{i}\sin\theta$ is to rotate the point representing $\mathbf{z}$ in the Argand diagram through an angle θ about the origin.

Not surprisingly, Hamilton reasoned that if two numbers (that is, the two components of a complex number) were enough to describe 2D rotations, then three numbers should be enough for 3D rotations. After thinking about this for eight years, Hamilton realized that four numbers were needed. The revelation came to him on Monday, 16 December 1843, as he was walking; he paused in his walk to carve the fundamental equations of quaternion algebra on the Brougham Bridge in Dublin.

In this section, we will review quaternions informally, emphasizing the practical results that we can obtain from them. Appendix A contains a formal description of quaternion algebra.

As its name suggests, a quaternion is a tuple of four real numbers. There are various representations, but it is most convenient for our purposes to represent a quaternion as pair with a sclar component and a 3D vector component. Thus we will write $q = (s, \mathbf{v})$, where $\mathbf{v} = (x, y, z)$, to represent a typical quaternion.

One of the reasons that Hamilton took a long time to discover quaternions is that he wanted a complete number system. (In modern terminology, he was looking for a field, not just a group.) As with complex numbers, the sum, difference, product, and ratio of two quaternions is a quaternion. The most important operation for our purposes is multiplication, defined by

$$q_1\, q_2 \;=\; (s_1,\, \mathbf{v_1})\;(s_2,\, \mathbf{v_2}) \;=\; (s_1 s_2 - \mathbf{v}_1 \cdot \mathbf{v}_2,\; s_1\mathbf{v}_2 + s_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2)$$

in which $\mathbf{v}_1 \cdot \mathbf{v}_2$ and $\mathbf{v}_1 \times \mathbf{v}_2$ are the conventional vector inner and outer products, respectively. ("Conventional" is rather misleading in this context. The familiar vector operations were introduced by the American mathematician Josiah Willard Gibbs (1839–1903); he got the idea by studying quaternion operations.)

The ***conjugate*** of the quaternion $q = (s, \mathbf{v})$ is the quaternion $q^* = (s, -\mathbf{v})$. The ***norm*** of the quaternion $q = (s, \mathbf{v})$ is the real number $\| q \| = q\, q^* = q^*\, q = s^2 + \mathbf{v} \cdot \mathbf{v}$. We can use the definition of conjugate to define an inverse operation. Given a quaternion q, we seek a quaternion q^{-1} such that $q^{-1}\, q = 1$. Multiplying the equation

$$1 \;=\; \frac{q\, q^*}{\| q \|}$$

by q^{-1} gives

$$q^{-1} \;=\; \frac{q^{-1}\, q\, q^*}{\| q \|} \;=\; \frac{q^*}{\| q \|}.$$

We can use the inverse to define quaternion division:

$$q_1/q_2 \;=\; q_1\, q_2^{-1}.$$

A quaternion with zero scalar component is called a ***pure quaternion***, rather in the same way as multiples of $\sqrt{-1}$ are called "pure imaginary". We will write vectors in quaternion products, with the convention that the vector $\mathbf{v}$ is to be read as the quaternion $(0, \mathbf{v})$.

If $\| q \| = 1$, then q is a ***unit quaternion***. For a unit quaternion, $q^{-1} = q^*$. The unit quaternions form a group with quaternion multiplication as the group operation. In other words:

- If q_1 and q_2 are unit quaternions, then $q_1\,q_2$ and $q_2\,q_1$ are unit quaternions.
- The inverse q^{-1} of a unit quaternion q is a unit quaternion.
- There is a unit element $e = (1,\, \mathbf{0})$: if q is any unit quaternion, $q\,e = e\,q = q$.

If $(s,\, \mathbf{v})$ is a unit quaternion, $s^2 + \mathbf{v} \cdot \mathbf{v} = 1$. Consequently we can write any unit quaternion as $(\cos\theta,\, \mathbf{u}\sin\theta)$, in which $\mathbf{u}$ is a unit vector.

Section A.3 contains more information about unit quaternions.

3 Rotations Represented as Quaternions

The connection between rotations and quaternions is as follows: suppose that $\mathbf{v_1}$ is a vector and $q = (\cos\theta,\, \mathbf{u}\sin\theta)$ is a unit quaternion. Then the vector $\mathbf{v_2}$ defined by

$$\mathbf{v_2} \;=\; q\,\mathbf{v_1}\,q^{-1}$$

is the vector $\mathbf{v_1}$ rotated through an angle 2θ about the axis $\mathbf{u}$. (Recall that $\mathbf{v_1}$ in the equation stands for the quaternion $(0,\, \mathbf{v_1})$.)

Quaternions have a number of advantages over matrices as a means of representing rotations:

1. Only four numbers — the minimum possible — are needed to represent a rotation.
2. Combining rotations by multiplying quaternions is fast and accurate.
3. It is easy to renormalize quaternions after many calculations.
4. It is easy to interpolate between quaternions to smoothly animate rotating objects.

3.1 Dynamics

Suppose that a body is rotating about an axis $\mathbf{u}$ with angular velocity ω. If its angular position at time t is θ, its position a short time Δt later is given by

$$\begin{aligned} \theta + \Delta\theta &\approx \theta + \frac{d\theta}{dt}\,\Delta t \\ &= \theta + \omega\,\Delta t \end{aligned}$$

The rotations can be represented by the quaternions

$$\begin{aligned} q &= \left(\cos\frac{\theta}{2},\, \mathbf{u}\sin\frac{\theta}{\mathbf{2}}\right) \\ \Delta q &= \left(\cos\frac{\Delta\theta}{2},\, \mathbf{u}\sin\frac{\mathbf{\Delta}\theta}{\mathbf{2}}\right) \end{aligned}$$

and, using the addition law (Appendix A.3, equation (1)), we have

$$q\,\Delta q \;=\; \left(\cos\frac{\theta+\Delta\theta}{2},\, \mathbf{u}\sin\frac{\theta+\mathbf{\Delta}\theta}{\mathbf{2}}\right)$$

Consequently, we can integrate angular motion using

$$q(t+\Delta t) \;=\; q(t)\left(\cos\frac{\omega\Delta t}{2},\, \mathbf{u}\sin\frac{\omega\mathbf{\Delta t}}{\mathbf{2}}\right)$$

A A Formal Approach to Quaternions

There are three ways of writing quaternions.

1. In the most abstract formulation, a quaternion is a tuple of four real numbers: (s, x, y, z).
2. By analogy with complex numbers (e.g., $\mathbf{z} = x + \mathbf{i}\,y$), we can write a quaternion in the form $s + \mathbf{i}\,x + \mathbf{j}\,y + \mathbf{k}\,z$. We can then use the normal rules of algebra to manipulate quaternions, provided that we respect the multiplication table for $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ shown in Figure 2.
3. Since three of the components of a quaternion behave like a vector, we can write a quaternion as a pair with a scalar component and a vector component: $(s,\, \mathbf{v})$.

	$\mathbf{i}$	$\mathbf{j}$	$\mathbf{k}$
$\mathbf{i}$	-1	$\mathbf{k}$	$-\mathbf{j}$
$\mathbf{j}$	$-\mathbf{k}$	-1	$\mathbf{i}$
$\mathbf{k}$	$\mathbf{j}$	$-\mathbf{i}$	-1

Figure 2: Multiplication of quaternion components

We will use the third notation for the formal definitions of quaternion operations that follow. Vector operations ($s\,\mathbf{v}$, $\mathbf{v_1} \cdot \mathbf{v_2}$, and $\mathbf{v_1} \times \mathbf{v_2}$) have their usual significance.

A.1 Definitions

Unit:	$\mathbf{1}$	$=$	$(1,\, \mathbf{0})$
Norm:	$\| (s,\, \mathbf{v}) \|$	$=$	$s^2 + \mathbf{v} \cdot \mathbf{v}$
Conjugate:	$(s,\, \mathbf{v})^*$	$=$	$(s,\, -\mathbf{v})$
Inverse:	q^{-1}	$=$	$q^* / \| q \|$
Addition:	$(s_1,\, \mathbf{v_1}) + (s_2,\, \mathbf{v_2})$	$=$	$(s_1 + s_2,\, \mathbf{v_1} + \mathbf{v_2})$
Subtraction:	$(s_1,\, \mathbf{v_1}) - (s_2,\, \mathbf{v_2})$	$=$	$(s_1 - s_2,\, \mathbf{v_1} - \mathbf{v_2})$
Multiplication:	$(s_1,\, \mathbf{v_1})\ (s_2,\, \mathbf{v_2})$	$=$	$(s_1 s_2 - \mathbf{v_1} \cdot \mathbf{v_2},\ s_1\,\mathbf{v_2} + s_2\,\mathbf{v_1} + \mathbf{v_1} \times \mathbf{v_2})$
Division:	q_1 / q_2	$=$	$q_1\, q_2^{-1}$

A.2 Properties

Unit:	$\mathbf{1}\,q$	$=$	$q\,\mathbf{1}$	
		$=$	q	
Norm:	$\|\,q\,\|$	$=$	$\|\,q^*\,\|$	
		$=$	$q\,q^*$	
		$=$	$q^*\,q$	
		$=$	$s^2+x^2+y^2+z^2$	[assuming $q=(s,(x,y,z))$]
	$\|\,q_1\,q_2\,\|$	$=$	$\|\,q_1\,\|\;\|\,q_2\,\|$	
Conjugate:	$(q^*)^*$	$=$	q	
	$(p\,q)^*$	$=$	$q^*\,p^*$	
	$(p+q)*$	$=$	p^*+q^*	
Addition:	q_1+q_2	$=$	q_2+q_1	
Inverse:	$(q^{-1})^{-1}$	$=$	q	
	$(q_1\,q_2)^{-1}$	$=$	$q_2^{-1}\,q_1^{-1}$	
Multiplication:	$q_1\,q_2$	$\neq$	$q_2\,q_1$	(in general)
	$(q_1\,q_2)\,q_3$	$=$	$q_1\,(q_2\,q_3)$	
Division:	q_1/q_2	$=$	$\dfrac{q_1\,q_2^*}{\|\,q_2\,\|}$	
Bilinearity:	$q_1\,(s_2\,q_2+s_3\,q_3)$	$=$	$s_2\,q_1\,q_2+s_3\,q_1\,q_3$	
	$(s_2\,q_2+s_3\,q_3)\,q_1$	$=$	$s_2\,q_2\,q_1+s_3\,q_3\,q_1$	

A.3 Unit Quaternions

If $\|\,q\,\|=1$, then q is a ***unit quaternion***. In this section, all quaternions are assumed to be unit quaternions.

A unit quaternion can be written in the form $(\cos\theta,\,\mathbf{u}\sin\theta)$. Proof: if $\|\,q\,\|=1$ and $q=(s,(x,y,z))$, then $s^2+x^2+y^2+z^2=1$. Since $s^2<1$, we can assume that $s=\cos\theta$ for some θ. From the identity $\sin^2\theta+\cos^2\theta=1$ we infer $x^2+y^2+z^2=\sin^2\theta$. Then $\mathbf{u}$ must be the unit vector $(x/r,y/r,z/r)$ where $r=\sqrt{x^2+y^2+z^2}$.

From the definition of conjugate, the conjugate of $(\cos\theta,\,\mathbf{u}\sin\theta)$ is $(\cos\theta,\,-\mathbf{u}\sin\theta)$ which we can write as $(\cos(-\theta),\,\mathbf{u}\sin(-\theta))$. It is straightforward to confirm that the inverse of a unit quaternion is a unit quaternion:

$$\begin{aligned} & (\cos\theta,\,\mathbf{u}\sin\theta)\,(\cos(-\theta),\,\mathbf{u}\sin(-\theta)) \\ = \;& (\cos^2\theta-\mathbf{u}\sin\theta\cdot\mathbf{u}\sin(-\theta),\,\mathbf{u}\cos\theta\sin\theta+\mathbf{u}\cos\theta\sin(-\theta)+\mathbf{u}\sin\theta\times\mathbf{u}\sin(-\theta)) \\ = \;& (1,\,\mathbf{0}) \end{aligned}$$

The product of unit quaternions is a unit quaternion:

$$\|\,q_1\,q_2\,\| \;=\; \|\,q_1\,\|\;\|\,q_2\,\|$$

$$\begin{aligned} &= 1 \times 1 \\ &= 1 \end{aligned}$$

The unit quaternions form a multiplicative group with unit element $(1, \mathbf{0})$ (see page 5).

Unit quaternions with the same axis perform addition of angles:

$$\begin{aligned} & (\cos\theta,\, \mathbf{u}\sin\theta)\,(\cos\phi,\, \mathbf{u}\sin\phi) \qquad (1) \\ = \;& (\cos\theta - (\mathbf{u}\cdot\mathbf{u})\sin\theta\sin\phi,\, \mathbf{u}\cos\theta\sin\phi + \mathbf{u}\sin\theta\cos\phi + (\mathbf{u}\times\mathbf{u})\sin\theta\sin\phi)) \\ = \;& (\cos(\theta+\phi),\, \mathbf{u}\sin(\theta+\phi)) \end{aligned}$$

If $(s,(x,y,z))$ is a unit quaternion, the corresponding OpenGL rotation matrix is

$$\begin{bmatrix} 1-2(y^2+z^2) & 2(xy-sz) & 2(xz+sy) & 0 \\ 2(xy+sz) & 1-2(x^2+z^2) & 2(yz-sx) & 0 \\ 2(xz-sy) & 2(yz+sx) & 1-2(x^2+y^2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A.4 Unit Quaternions Represent Rotations

A rotation involves an axis and an angle. For example, during one hour, the earth rotates 15° about its North-South axis. In general, we can represent a rotation by a unit vector, $\mathbf{u}$, in the direction of the axis of rotation, and an angle, θ, giving the amount of rotation.

Define $R(\theta,\mathbf{u})$ as an operation on vectors: its effect is to rotate a vector through an angle θ about an axis defined by the unit vector $\mathbf{u}$. Following [8, page 359ff.], we compute the vector $R(\theta,\mathbf{u})\,\mathbf{v}$.

The first step is to resolve $\mathbf{v}$ into components parallel to and orthogonal to $\mathbf{u}$:

$$\begin{aligned} \mathbf{v}_p &= (\mathbf{u}\cdot\mathbf{v})\mathbf{u} \\ \mathbf{v}_o &= \mathbf{v} - (\mathbf{u}\cdot\mathbf{v})\mathbf{u} \end{aligned}$$

During the rotation, $\mathbf{v}_p$ remains unchanged and $\mathbf{v}_o$ moves to $R\,\mathbf{v}_o$. Let $\mathbf{v}$ be a vector perpendicular to $\mathbf{v}_p$ and lying in the plane passing through $\mathbf{v}$ and $\mathbf{R}\,\mathbf{v}$. Then

$$\begin{aligned} \mathbf{v} &= \mathbf{u}\times\mathbf{v}_p \\ &= \mathbf{u}\times\mathbf{v} \end{aligned}$$

Consequently,

$$R\,\mathbf{v}_o = (\cos\theta)\mathbf{v}_p + (\sin\theta)\mathbf{v}$$

and hence

$$\begin{aligned} R\,\mathbf{v} &= R\,\mathbf{v}_p + R\,\mathbf{v}_o \\ &= R\,\mathbf{v}_p + (\cos\theta)\mathbf{v}_p + (\sin\theta)\mathbf{v} \\ &= (\mathbf{u}\cdot\mathbf{v})\mathbf{u} + \cos\theta(\mathbf{v} - (\mathbf{u}\cdot\mathbf{v})\mathbf{u}) + (\sin\theta)(\mathbf{u}\times\mathbf{v}) \\ &= (\cos\theta)\mathbf{v} + (1-\cos\theta)\mathbf{u}(\mathbf{u}\cdot\mathbf{v}) + (\sin\theta)(\mathbf{u}\times\mathbf{v}) \qquad (2) \end{aligned}$$

Let $p = (0, \mathbf{v})$ be a pure quaternion and $q = (\cos\phi, \mathbf{u}\sin\phi)$ be a unit quaternion and consider the quaternion

$$\begin{aligned} q\,p\,q^* &= (\cos\phi,\ \mathbf{u}\sin\phi)\ (0, \mathbf{v})\ (\cos\phi,\ -\mathbf{u}\sin\phi) \\ &= (0, (\cos^2\phi - \sin^2\phi)\mathbf{v} + 2\sin^2\phi(\mathbf{u}\cdot\mathbf{v})\mathbf{u} + 2\cos\phi\sin\phi(\mathbf{u}\times\mathbf{v})) \\ &= (0, (\cos 2\phi)\mathbf{v} + (1-\cos 2\phi)(\mathbf{u}\cdot\mathbf{v})\mathbf{u} + \sin 2\phi(\mathbf{u}\times\mathbf{v})) \end{aligned} \tag{3}$$

Comparing (2) and (3), we see that they are the same if we substitute $\theta = 2\phi$.

References

[1] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice.* Addison-Wesley, 1996. Second Edition in C.

[2] Paul S. Heckbert, editor. *Graphics Gems IV.* Acedemic Press, 1994.

[3] Ken Shoemake. Quaternions, 1993. Available at various web sites, including (December 2001) http://www.fasterlight.com/hugg/links/gamelinks.html.

[4] Ken Shoemake. Arcball rotation control. In [2], pages 175–192. Academic Press, 1994.

[5] Ken Shoemake. Euler angle conversion. In [2], pages 222–229. Academic Press, 1994.

[6] Ken Shoemake. Fiber bundle twist reduction. In [2], pages 230–238. Academic Press, 1994.

[7] Ken Shoemake. Polar matrix decomposition. In [2], pages 207–221. Academic Press, 1994.

[8] Alan Watt and Mark Watt. *Advanced Animation and Rendering Techniques.* Addison-Wesley, 1992.