3-Dimensional Rotation and Quaternions

David Mount

September 26, 2000 (for CMSC 828C)

Sources: This is derived from a lecture given in my computer graphics class. Much of it is based on the presentation given in "Advanced Animation and Rendering Techniques" by A. Watt and M. Watt, Addison-Wesley, 1992, pp. 356–368.

Rotation in 3-Space: Parameterizing rotations in 3-space is a tricky, but interesting problem. Describing and managing rotations in 3-space is a difficult task (at least conceptually), compared with the relative simplicity of rotations in the plane.

Why do we care about rotations? Suppose that you are an animation programmer for a computer graphics studio. The object that you are animating is to be moved smoothly from one location to another. We can think of the position of an object as being described by two quantities, its *location* and its *orientation* (that is, a rotation relative to some fixed reference frame). If the object is in the same orientation before and after, we can just translate from one location to the other. If not, we need to find a way of smoothly interpolating between its two orientations. (In animation this is called "tweening".) This usually involves rotations in 3-space. But how should these rotations be performed so that the animation looks natural?

Given two orthonormal frames, G and H, we would like a way to interpolate smoothly between these two (say if we want to produce a smooth animation from one to the other). Since smoothly interpolating positions by translation is pretty easy to understand, let us ignore the issue of position, and just focus on rotations about the origin. A naive solution would be to represent G and H by their orthonormal basis vectors, and interpolate from each vector to the corresponding vector, but the intermediate vectors would not necessarily be orthonormal. We will explore two methods for dealing with rotation, *Euler angles* and *quaternions*.

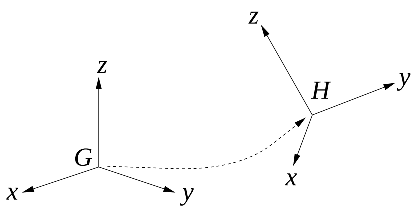

Figure 1: Smooth Interpolation of Frames.

Euler Angles: Euler proved that the composition any number of rotations in three-space is just a single rotation in 3-space (about an appropriately chosen vector). Euler also showed that any rotation in 3-space could be broken down into exactly three rotations, one about each of the coordinate axes, called *rolls*.

For example, consider the orthonormal frame G, described earlier. Suppose that we want to rotate the standard frame F so that it coincides with G. Let us consider the process in reverse. We will see how to rotate G so that it coincides with the standard frame. Perhaps the easiest way to see this is to consider the three rotations that bring G into alignment with F, and then reverse these rotations.

This process is easier to describe than it is to visualize. Suppose that we label G's basis vectors $\vec{u}$, $\vec{v}$ and $\vec{w}$. Let us assume that these vectors are in *general position*, that is particular none is parallel to any of the coordinate axes. Let $\vec{w}'$ denote the projection of w onto the yz-coordinate plane. First rotate about the x-axis by some angle θ_x, until the vector $\vec{w}'$ coincides with the z-axis. (See part (a) of the figure below.) The original vector $\vec{w}$ will now lie on the xz-coordinate plane. Next, rotate about the y-axis (thus keeping the xz-coordinate plane fixed) until $\vec{w}$ coincides with the z-axis. (See part (b) of the figure.) Call this angle θ_y. When this happens, the two vectors $\vec{u}$ and $\vec{v}$ (being orthogonal to $\vec{w}$) must lie on the xy-plane. Finally, we rotate about the z-axis until $\vec{u}$ coincides with the x-axis. (See part (c) of the figure.) Call this angle θ_z. Assuming that G is orthonormal and right-handed, it follows that $\vec{v}$ will coincide with the y-axis. Thus, by these three rotations, one about each of the axes, we can bring G into alignment with F.

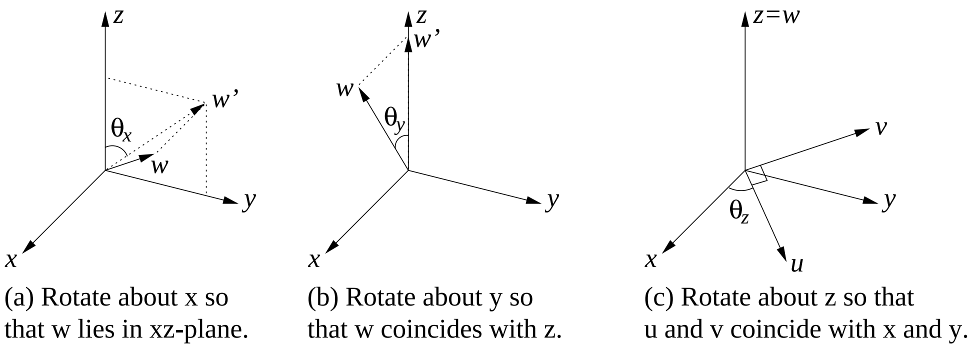

Figure 2: Rotating a frame to coincide with the standard frame.

It follows that it is possible to perform any change of orientation with three rotations, one about each of the coordinate axes, by a triple of three angles, $(\theta_x, \theta_y, \theta_z)$. These define a general rotation matrix, by composing the three basic rotations:

$$R(\theta_x, \theta_y, \theta_z) = R_z(\theta_z) R_y(\theta_y) R_x(\theta_x).$$

These three angles are called the *Euler angles* for the rotation. Thus, we can parameterize *any* rotation in 3-space as triple of numbers, each in the range $[0, 2\pi]$.

Now, given two orientations in space, say given by the Euler angles $\Theta = (\theta_x, \theta_y, \theta_z)$ and $\Phi = (\phi_x, \phi_y, \phi_z)$, we can interpolate between then, say by taking convex combinations. Given any $\alpha \in [0, 1]$, we can define

$$R(\alpha) = R((1 - \alpha)\Theta + \alpha\Phi),$$

for example. As α varies from 0 to 1, this will smoothly rotate from one orientation to the other.

Rotation Matrices: Recall from basic affine geometry, that we may represent each of the basic axis rotations using matrix form. We will assume that points in three space are represented using homogeneous coordinates.

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

If we compose these three matrices, for Euler angles θ_x, θ_y and θ_z, it is easy to verify that the final result is

$$R(\theta_x, \theta_y, \theta_z) = \begin{pmatrix} c_y c_z & c_y s_z & -s_y & 0 \\ s_x s_y c_z - c_x s_z & s_x s_y s_z + c_x c_z & s_x c_y & 0 \\ c_x s_y c_z + s_x s_z & c_x s_y s_z - s_x c_z & c_x c_y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

where $c_x = \cos\theta_x$, $s_x = \sin\theta_x$, and similarly for the other two axes.

Problems with Euler Angles and Gimbal Lock: There are a number of problems with the use of Euler angles. Most of these originate from the fact that the this representation is not intrinsic, and depends critically on the choice of coordinate system. In the plane, a 30 degree rotation is the same, no matter what direction the axes are pointing (as long as they are orthonormal and right-handed). However, the result of an Euler angle orientation depends very much on where the frame is, and even the order in which you name the axes. It would be nice to have a method of describing rotations that is independent of the choice of coordinate systems. Another problem, is that of *Gimbal lock*, which we will discuss next.

The term Gimbal lock derives from a mechanical problem in the gimbal mechanism that is used to support a compass or gyroscope. This mechanism consists of three concentric rings. Under certain conditions a degree of freedom is lost, resulting in *gimbal lock*.

Suppose that we use the Euler angle mechanism to describe a rotation, and we select a y-roll of $\pi/2$. (Consider for a moment how such a situation would arise using the Euler angle construction above.) It turns out that the x-rolls and z-rolls have exactly the same effect, and hence we will have suffered a loss of one degree of freedom.

To see this, consider first a roll by angle θ_x about x, followed by a roll of $\pi/2$ about y. (Note that rolls move points in space, not the axes, so position of the y is the same irrespective of the x-roll. However, in the composition of the two rotations, observe that the original x-axis has been rotated so that it is aligned with the $(-z)$-axis, and hence a rotation by θ_x about x is *effectively* equivalent to a rotation by $-\theta_x$ about z. The final rotation about z will only add to the original rotation.

Another way to see this is to consider the matrix $R(\theta_x, \pi/2, \theta_z)$. Observe that $c_y = 0$ and $s_y = 1$, implying that the result is the following (nongeneral) transformation.

$$\begin{pmatrix} 0 & 0 & -1 & 0 \\ s_x c_z - c_x s_z & s_x s_z + c_x c_z & 0 & 0 \\ c_x c_z + s_x s_z & c_x s_z - s_x c_z & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & -1 & 0 \\ \sin(\theta_x - \theta_z) & \cos(\theta_x - \theta_z) & 0 & 0 \\ \cos(\theta_x - \theta_z) & -\sin(\theta_x - \theta_z) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Angular Displacement: A more intrinsic way to express rotations (about the origin) in 3-space is in terms of two quantities, $(\theta, \vec{u})$, consisting of an angle θ, and an axis of rotation $\vec{u}$. Let's consider how we might do this. First consider a vector $\vec{v}$ to be rotated. Let us assume that $\vec{u}$ is of unit length.

Our goal is to describe the image of $\vec{v}$ under this rotation as a function of θ and $\vec{u}$. Let $R(\vec{v})$ denote this image. In order to derive this, we begin by decomposing $\vec{v}$ as the sum of its components that are parallel to and orthogonal to $\vec{u}$, respectively.

$$\vec{v}_{||} = (\vec{u} \cdot \vec{v})\vec{u} \qquad \vec{v}_{\perp} = \vec{v} - \vec{v}_{||} = \vec{v} - (\vec{u} \cdot \vec{v})\vec{u}.$$

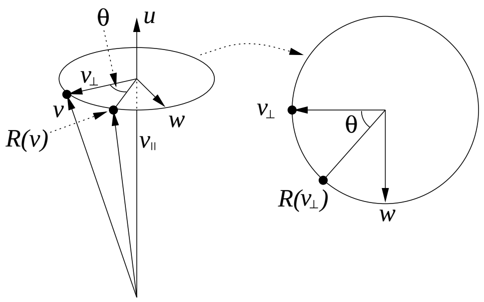

Figure 3: Angular displacement.

Note that $\vec{v}_{||}$ is unaffected by the rotation, but $\vec{v}_{\perp}$ is rotated to a new position $R(\vec{v}_{\perp})$. To determine this rotated position, we will first construct a vector that is orthogonal to $\vec{v}_{\perp}$ lying in the plane of rotation.

$$\vec{w} = \vec{u} \times \vec{v}_{\perp} = \vec{u} \times (\vec{v} - \vec{v}_{||}) = (\vec{u} \times \vec{v}) - (\vec{u} \times \vec{v}_{||}) = \vec{u} \times \vec{v}.$$

The last step follows from the fact that $\vec{u}$ and $\vec{v}_{||}$ are parallel, and so the cross product is zero. Clearly $\vec{w}$ is orthogonal to both $\vec{v}_{\perp}$ and $\vec{u}$. Furthermore, because $\vec{v}_{\perp}$ is orthogonal to the unit vector $\vec{u}$, it follows that $\vec{w}$ is of the same length as $\vec{v}_{\perp}$.

Now, consider the plane spanned by $\vec{v}_{\perp}$ and $\vec{w}$. We have

$$R(\vec{v}_{\perp}) = (\cos\theta)\vec{v}_{\perp} + (\sin\theta)\vec{w}.$$

From this we have

$$\begin{aligned} R(\vec{v}) &= R(\vec{v}_{||}) + R(\vec{v}_{\perp}) \\ &= R(\vec{v}_{||}) + (\cos\theta)\vec{v}_{\perp} + (\sin\theta)\vec{w} \\ &= (\vec{u} \cdot \vec{v})\vec{u} + (\cos\theta)(\vec{v} - (\vec{u} \cdot \vec{v})\vec{u}) + (\sin\theta)\vec{w} \\ &= (\cos\theta)\vec{v} + (1 - \cos\theta)\vec{u}(\vec{u} \cdot \vec{v}) + (\sin\theta)(\vec{u} \times \vec{v}). \end{aligned}$$

This last expression is the image of $\vec{v}$ under the rotation. Notice that, unlike Euler angles, this is expressed entirely in terms of geometric quantities, which do not depend on the choice of coordinates. This is an advantage over Euler angles. But it is rather hard to handle in this form.

Quaternions: We will now delve into a subject, which at first may seem quite unrelated. But keep the above expression in mind, since it will reappear in most surprising way.

This story begins in the early 19th century, when the great mathematician Hamilton was searching for a generalization of the complex number system. Imaginary numbers can be thought of as linear combinations of two basis elements, 1 and i, which satisfy the multiplication rules $1^2 = 1$, $i^2 = -1$ and $1 \cdot i = i \cdot 1 = i$. (The interpretation of $i = \sqrt{-1}$ arises from the second rule.) Hamilton searched for a generalization involving two imaginary basis values, i and j, but couldn't make it work. After many years he hit upon the trick, which was to consider three imaginary values i, j, and k, which behave as follows:

$$i^2 = j^2 = k^2 = -1 \qquad ij = k,\ jk = i,\ ki = j.$$

Combining these, it follows that $ji = -k$, $kj = -i$ and $ik = -j$. A *quaternion* is defined to be a "generalized complex number" of the form

$$\mathbf{q} = q_0 + q_1 i + q_2 j + q_3 k.$$

As a mathematical aside, the elements ± 1, $\pm i$, $\pm j$, $\pm k$ are called the *quaternion units.* It is easy to verify that they form a nonabelian (noncommutative) group of order 8. This group is called the *unit quaternion group.* Quaternions form a noncommutative ring, and have a number of interesting algebraic properties. In particular, quaternions with integer coefficients are instrumental in proving a classic theorem of Lagrange, that every positive integer can be expressed as the sum of four perfect squares.

We will see that quaternions bear a striking resemblance to our notation for angular displacement. In particular, we can rewrite the quaternion notation in terms of a scalar and vector as

$$\mathbf{q} = (s, \vec{u}) = s + u_x i + u_y j + u_z k.$$

Given the rules above for multiplication, it is easy to derive the rules for multiplying quaternions. Consider two quaternions $\mathbf{q}_1 = (s, \vec{u})$ and $\mathbf{q}_2 = (t, \vec{v})$. If we multiply them together we have

$$\begin{aligned}
\mathbf{q}_1\mathbf{q}_2 &= (s + u_x i + u_y j + u_z k)(t + v_x i + v_y j + v_z k) \\
&= (st + sv_x i + sv_y j + sv_z k) \ + \ (u_x ti + u_x v_x i^2 + u_x v_y ij + u_x v_z ik) \\
&\quad + \ (u_y tj + u_y v_x ji + u_y v_y j^2 + u_y v_z jk) \ + \ (u_z tk + u_z v_x ki + u_z v_y kj + u_z v_z k^2) \\
&= (st + sv_x i + sv_y j + sv_z k) \ + \ (u_x ti - u_x v_x + u_x v_y k - u_x v_z j) \\
&\quad + \ (u_y tj - u_y v_x k - u_y v_y + u_y v_z i) \ + \ (u_z tk + u_z v_x j - u_z v_y i - u_z v_z).
\end{aligned}$$

Now we collect common terms in i, j, and k.

$$\begin{aligned}
\mathbf{q}_1\mathbf{q}_2 &= (st - u_x v_x - u_y v_y - u_z v_z) \ + \ (sv_x + tu_x + u_y v_z - u_z v_y)i \\
&\quad + \ (sv_y + tu_y - u_x v_z + u_z v_x)j \ + \ (sv_z + tu_z + u_x v_y - u_y v_x)k.
\end{aligned}$$

Observe that the scalar term is $st - (\vec{u} \cdot \vec{v})$. If we consider the first and second terms of the other parts, we see that they are equal to $s\vec{v} + t\vec{u}$. Finally, if we consider the third and fourth terms of each part together, observe that they are equivalent to the determinant

$$\begin{vmatrix} i & j & k \\ u_x & u_y & u_z \\ v_x & v_y & v_z \end{vmatrix},$$

which is just $\vec{u} \times \vec{v}$. Combining this, we have the following formula for *quaternion multiplication.*

$$\mathbf{q}_1\mathbf{q}_2 = (st - (\vec{u} \cdot \vec{v}), s\vec{v} + t\vec{u} + (\vec{u} \times \vec{v})).$$

If you ignore the cross-product term, this bears a striking superficial resemblance to the rule for complex multiplication. Note that quaternion multiplication is associative but not commutative.

Define the *conjugate* of a quaternion $\mathbf{q} = (s, \vec{u})$ to be

$$\overline{\mathbf{q}} = (s, -\vec{u}).$$

It is easy to show that the product of a quaternion and its conjugate has a zero vector component, and hence may be thought of as a scalar. Define the *magnitude* of a quaternion to be the square root of this product

$$|\mathbf{q}|^2 = \mathbf{q}\overline{\mathbf{q}} = s^2 + |\vec{u}|^2.$$

(Notice that we are abusing notation a bit here.) A *unit quaternion* is one of unit magnitude, $|\mathbf{q}| = 1$. A *pure quaternion* is one with a 0 scalar component

$$\mathbf{p} = (0, \vec{v}).$$

Any quaternion of nonzero magnitude has a multiplicative *inverse*, which is

$$\mathbf{q}^{-1} = \frac{1}{|\mathbf{q}|^2}\overline{\mathbf{q}}.$$

(Try multiplying $\mathbf{q}\mathbf{q}^{-1}$ to see why this is so.) Observe that if $\mathbf{q}$ is a unit quaternion, then $\mathbf{q}^{-1} = \overline{\mathbf{q}}$.

Quaternions and Rotation: What do quaternions have to do with rotation in 3-space? We will represent a rotation by a unit quaternion $\mathbf{q}$ (we will see exactly how later). Given any point $P = (v_x, v_y, v_z)$ in 3-space, we will represent it by mapping it to a pure quaternion, $\mathbf{p} = (0, \vec{v})$. The image of $\mathbf{p}$ under the rotation $\mathbf{q}$ will again be a pure quaternion (and so is easy to map back to a point in 3-space). Define the *rotation operator*

$$R_{\mathbf{q}}(\mathbf{p}) = \mathbf{q}\mathbf{p}\mathbf{q}^{-1}.$$

By applying the multiplication rule, and using the fact that $\mathbf{q}^{-1} = \overline{\mathbf{q}}$ for unit quaternions, it is easy to derive that

$$R_{\mathbf{q}}(\mathbf{p}) = (0,\ (s^2 - (\vec{u} \cdot \vec{u}))\vec{v} + 2\vec{u}(\vec{u} \cdot \vec{v}) + 2s(\vec{u} \times \vec{v})).$$

So what does this have to do with rotation? Let us see if we can express this in a more suggestive form. Since $\mathbf{q}$ is of unit magnitude, we can express it as

$$\mathbf{q} = (\cos\theta, (\sin\theta)\vec{u}), \qquad \text{where } |\vec{u}| = 1.$$

Plugging this into the above expression, we have

$$\begin{aligned} R_{\mathbf{q}}(\mathbf{p}) &= (0,\ (\cos^2\theta - \sin^2\theta)\vec{v} + 2(\sin^2\theta)\vec{u}(\vec{u} \cdot \vec{v}) + 2\cos\theta\sin\theta(\vec{u} \times \vec{v})) \\ &= (0,\ (\cos 2\theta)\vec{v} + (1 - \cos 2\theta)\vec{u}(\vec{u} \cdot \vec{v}) + \sin 2\theta(\vec{u} \times \vec{v})). \end{aligned}$$

Now, recall the rotation displacement equation presented earlier. The vector part of this quaternion is identical, except that we have 2θ in place of θ.

Thus, in summary, we encode points in 3-space as pure quaternions

$$\mathbf{p} = (0, \vec{v}),$$

and we encode a rotation by angle θ about a unit vector $\vec{u}$ as a unit quaternion

$$\mathbf{q} = (\cos(\theta/2), \sin(\theta/2)\vec{u}),$$

then the image of the point under this rotation is given by the vector part of the result of the quaternion rotation operator $R_{\mathbf{q}}(\mathbf{p})$.

Example: Consider the 3-d rotation shown in the figure below. This rotation can be achieved by performing a rotation about the y-axis by -90 degrees. Thus $\theta = -\pi/2$, and $\vec{u} = (0, 1, 0)$. Thus the quaternion that encodes this rotation is

$$\mathbf{q} = (\cos(-\pi/4), \sin(-\pi/4)(0,1,0)) = \left(\frac{1}{\sqrt{2}}, \left(0, -\frac{1}{\sqrt{2}}, 0\right)\right).$$

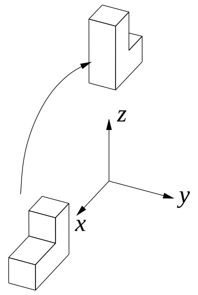

Figure 4: Rotation example.

The image of the point P on the tip of the x-unit vector with homogeneous coordinates $(1, 0, 0, 1)$, comes by representing P by a vector $\vec{v} = (1, 0, 0)$, and then encoding $\vec{v}$ as a pure quaternion $\mathbf{p} = (0, (1, 0, 0))$. Then we apply the rotation operator

$$\begin{aligned}
R_{\mathbf{q}}(\mathbf{p}) &= (0,\ (1/2 - 1/2)(1,0,0) + 2(0,1,0)0 + (2/\sqrt{2})((0, -1/\sqrt{2}, 0) \times (1,0,0))) \\
&= (0, (0,0,0) + (0,0,0) + (-1)(0,0,-1)) \\
&= (0, (0,0,1)).
\end{aligned}$$

Thus P is mapped to a point on the z-axis, as expected.

Composing Rotations: We have shown that each unit quaternion corresponds to a rotation in 3-space. This is an elegant representation, but can we manipulate rotations through quaternion operations? The answer is yes. In particular, the action of multiplying two unit quaternions results in another unit quaternion. Furthermore, the resulting product quaternion corresponds to the composition of the two rotations. In particular, given two unit quaternions $\mathbf{q}$ and $\mathbf{q}'$, a rotation by $\mathbf{q}$ followed by a rotation by $\mathbf{q}'$ is equivalent to a single rotation by the product $\mathbf{q}'' = \mathbf{q}'\mathbf{q}$. That is,

$$R_{\mathbf{q}'} R_{\mathbf{q}} = R_{\mathbf{q}''} \qquad \text{where } \mathbf{q}'' = \mathbf{q}'\mathbf{q}.$$

This follows from the associativity of quaternion multiplication, and hence the fact that $(\mathbf{q}\mathbf{q}')^{-1} = \mathbf{q}^{-1}\mathbf{q}'^{-1}$, as shown below.

$$\begin{aligned}
R_{\mathbf{q}'}(R_{\mathbf{q}}(\mathbf{p})) &= \mathbf{q}'(\mathbf{q}\mathbf{p}\mathbf{q}^{-1})\mathbf{q}'^{-1} \\
&= (\mathbf{q}'\mathbf{q})\mathbf{p}(\mathbf{q}^{-1}\mathbf{q}'^{-1}) \\
&= (\mathbf{q}'\mathbf{q})\mathbf{p}(\mathbf{q}'\mathbf{q})^{-1} \\
&= \mathbf{q}''\mathbf{p}\mathbf{q}''^{-1} \\
&= R_{\mathbf{q}''}(\mathbf{p}).
\end{aligned}$$

Matrices and Quaternions: Quaternions provide a very elegant way of representing rotations in 3-space. Returning to the problem of interpolating smoothly between two orientations, we can see that we can describe the before and after orientations of any object by two quaternions, $\mathbf{q}$ and $\mathbf{p}$. Then, to interpolate smoothly between these two orientations, we just interpolate between $\mathbf{q}$ and $\mathbf{p}$ in quaternion space. This is not really a linear interpolation, because the quaternions must be of unit length. It is more like interpolating between two points on the surface of a sphere.

Because many graphics systems have hardware support for matrix multiplication, it is desirable to express quaternion operations in this form. Given a unit quaternion

$$\mathbf{q} \;=\; (\cos(\theta/2), \sin(\theta/2)\vec{u}) \;=\; (w, (x, y, z)),$$

what is the corresponding affine transformation (expressed as a rotation matrix). By simply expanding the definition of of $R_{\mathbf{q}}(\mathbf{p})$, it is not hard to show that the following (homogeneous) matrix is equivalent

$$\begin{pmatrix} 1-2y^2-2z^2 & 2xy-2wz & 2xz+2wy & 0 \\ 2xy+2wz & 1-2x^2-2z^2 & 2yz-2wx & 0 \\ 2xz-2wy & 2yz+2wx & 1-2x^2-2y^2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Note that unlike the Euler angle transformation matrix, this one is symmetric with respect to x, y, and z.

The inverse problem, mapping an arbitrary rotation matrix to a quaternion, is also a relatively straightforward process. Watt and Watt show how to solve this problem.

Interpolation Using Quaternions: Let us return to the initial problem of smoothly interpolating between two given orientations. Using our quaternion representation, let $\mathbf{q}_0$ and $\mathbf{q}_1$ be the unit quaternions corresponding to our initial and final orientations, respectively. We want to find a quaternion $\mathbf{q}(t)$, for $0 \le t \le 1$, such that $\mathbf{q}(0) = \mathbf{q}_0$ and $\mathbf{q}(1) = \mathbf{q}_1$. Notice that we cannot simply interpolate linearly between these two (as $(1-t)\mathbf{q}_0 + t\mathbf{q}_1$), since the intermediate quaternions would not be units, and hence would not correspond to a rotation. Rather, we may think of unit quaternions as being points on a 4-dimensional sphere, and our goal is to interpolate between two along a great arc of this sphere.

We will use something called a *spherical linear interpolation* (or *slerp*). To see how to do this, let us consider the simpler problem of interpolating between two unit points along an arc of the unit circle. Let $\vec{u}_0$ and $\vec{u}_1$ be these vectors. Let ϕ denote the angle between these vectors, and let $\theta = \phi t$ be the intermediate angle of the interpolated unit vector. We seek an interpolation of the following form:

$$\vec{u}(t) = \alpha(t)\vec{u}_0 + \beta(t)\vec{u}_1,$$

for scalars $\alpha(t)$ and $\beta(t)$ (functions of t) subject to the constraints

$$\begin{aligned} (\vec{u}(t) \cdot \vec{u}(t)) &= 1, \\ (\vec{u}_0 \cdot \vec{u}_1) &= \cos\phi, \\ (\vec{u}_0 \cdot \vec{u}(t)) &= \cos\theta. \end{aligned}$$

Solving for $\alpha(t)$ and $\beta(t)$ we have

$$\vec{u}(t) \;=\; \frac{\sin(\phi-\theta)}{\sin\phi}\vec{u}_0 + \frac{\sin\theta}{\sin\phi}\vec{u}_1$$

$$= \frac{\sin((1-t)\phi)}{\sin\phi}\vec{u}_0 + \frac{\sin t\phi}{\sin\phi}\vec{u}_1.$$

We can generalize this immediately to quaternions, yielding

$$\mathbf{q}(t) = \frac{\sin((1-t)\phi)}{\sin\phi}\mathbf{q}_0 + \frac{\sin t\phi}{\sin\phi}\mathbf{q}_1.$$

One of the problems with interpolation on the sphere is that there are two possible ways of moving from one point to another (one through angle ϕ and the other through angle $2\pi - \phi$). It would seem logical to take the closer of the two. Unfortunately, things are not so simple on the quaternion sphere. The reason is that this sphere has the strange topological feature that the rotation generated by $\mathbf{q}$ and $-\mathbf{q}$ are identical. (Do you see why?)

Since we want to interpolate along the shortest path, one way to do this is to choose to interpolate from $\mathbf{q}_0$ to either $\mathbf{q}_1$ or $-\mathbf{q}_1$, whichever is closer. To decide which to use, we compute the magnitude of their differences $|\mathbf{q}_0 - \mathbf{q}_1|$ or $|\mathbf{q}_0 + \mathbf{q}_1|$ and choose the one ($\mathbf{q}_1$ or $-\mathbf{q}_1$) which results in the smaller difference.

Conclusions: We have presented two methods for representing rotations and orientations in 3-space. Euler angles provide a simple method based on three rotations about the coordinate axes and quaternions. Quaternions provide a number of nice features over Euler angles, in the sense that the iterpolation between two orientations is independent of the choice of the coordinate axes. We also discussed how to map the quaternion representation to the more standard matrix notation and how to perform spherical linear interpolation between two quaternions.