Quaternions and Motion Interpolation

Rosanna Heise
Bruce A. MacDonald
Department of Computer Science
The University of Calgary

Abstract

This paper explains straight-line interpolation of solid object motion, such as robot end effector translation and rotation. Smoothly changing orientation is accomplished using quaternions — a way of representing every orientation as four numbers (an angle and an axis of rotation). The first portion of the paper clarifies quaternions to provide an intuitive understanding of their role in rotation. Interpolation is then discussed, concluding with some problems in real manipulator implementations. The interpolation method has been tested on an Excalibur robot.

Introduction

This paper addresses the problem of planning a smooth trajectory for moving an object along a linear path. Such controlled motion is required, for example, in robotics, during obstacle avoidance and object approach. Often the position of an object is expressed with respect to world coordinates as

1. a location, (x, y, z), and
2. an orientation, (*roll*, *pitch*, *yaw*), indicating a rotation of *roll* about the x-axis, followed by a rotation of *pitch* around the y-axis, and finally a rotation of *yaw* about the z-axis.

As will be shown, it is trivial to interpolate location. The difficulty lies in interpolating the orientation. It is not enough to smoothly change each of the angles (*roll*, *pitch*, *yaw*), since this results in an uneven overall motion. The orientation change must be converted into rotation of a single angle, which is interpolated, about one axis. Quaternions aid this interpolation process by providing an explicit representation and efficient path control.

Recently several publications have appeared ([Brady 1986], [Canny 1988], [Pletincks 1988], [Shoemake 1985, 1987], and [Taylor 1986]) promoting quaternions for rotation in graphics and robotics. Many formulae are given indicating the ease with which this is done: how quaternions cause rotations and how to optimize the calculations to outperform standard matrix techniques. As yet it is difficult to gain a clear, intuitive understanding of quaternions, and, as one author comments " <one> may stumble a bit over quaternions" [Shoemake 1985]. This paper de-mystifies quaternions and their role in rotation. Section 1 provides motivation for quaternions and definitions of them and their operations. The next section describes how quaternion multiplication causes vector rotation. Straight-line interpolation, using quaternions for orientation change, is discussed in section 3. Finally, the implementation on an actual robot is discussed. This includes a section indicating the conversion process between robot joint angles and quaternions.

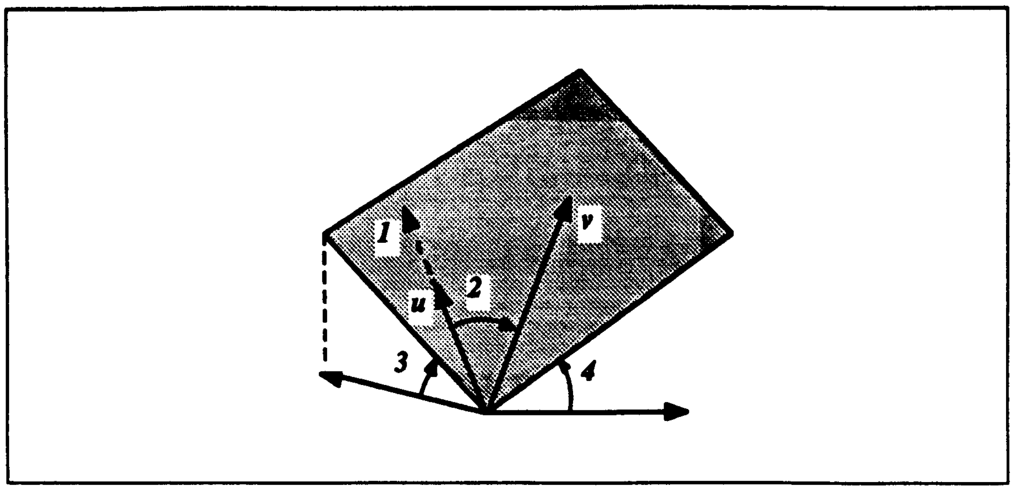

Figure 1: Changing u into v

1 What is a Quaternion?

Development of quaternions is due to Sir William R. Hamilton [Hamilton 1969, Kelland 1904], who sought an easy way to change one vector into another. Given any two vectors, u and v, there must be some simple quantity[1] q which transforms u into v:

$$qu = v.$$

In one dimension, determining q is easy: $q = \frac{v}{u}$. Generally, q expresses a relative length and direction between two vectors; it is the quotient of two vectors. In three dimensions, a vector has direction and magnitude, hence specifying the change in each of these is the minimum information required for q, as shown in figure 1:

a. Change the length of u to correspond to the length of v. This requires one number (1).

b. Rotate u through an angle in a plane until it is parallel to v. This requires three numbers — the angle of rotation (2) and the plane in which the rotation is to occur (the offset (3 and 4) from two known axes).

From this requirement of four numbers comes the name *quaternion*. Quaternions are an extension of the complex numbers to four-space, and are represented as algebraic quantities with three orthonormal "imaginary" axes ($\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$), as shown table 1. Since a quaternion is a four-vector, it inherits all vector properties and operations, including the dot product. This will be helpful during interpolation. Table 1 shows some common quaternion operations, from which it should be noted that quaternions form a non-abelian division ring [Herstein 1975].

[1] Matrices are one such quantity, but they are more complex than necessary.

A Quaternion $q = S + X\mathbf{i} + Y\mathbf{j} + Z\mathbf{k} = [S, \bar{w}]$ where $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$	
addition	$q_1 + q_2 = [S_1, \bar{w}_1] + [S_2, \bar{w}_2] = [S_1 + S_2, \bar{w}_1 + \bar{w}_2]$
additive identity	$0 = [0, \bar{0}]$
scalar multiplication	$\kappa q = [\kappa S, \kappa \bar{w}]$
multiplication	$q_1 q_2 = [S_1, \bar{w}_1][S_2, \bar{w}_2] = [S_1 S_2 - \bar{w}_1 \cdot \bar{w}_2, S_1 \bar{w}_1 + S_2 \bar{w}_2 + \bar{w}_1 \times \bar{w}_2]$
multiplicative identity	$1 = [1, \bar{0}]$
multiplicative inverse	$q^{-1} = [\frac{S}{\varsigma}, \frac{-\bar{w}}{\varsigma}]$ where $\varsigma = \parallel q \parallel$

Table 1: Quaternion operations

2 Quaternion Multiplication and Three-Space

An important use for quaternions is vector rotation. Any three-vector $\bar{v}$ can be mapped into four space as $v = [0, \bar{v}]$ and treated as a quaternion. When such vectors are multiplied by quaternions, rotation and scaling may occur. This section limits this to rotation by considering a special subset of quaternions, namely, the unit quaternions, which preserve the vector norm. A rotation of $\bar{v}$ by θ around the unit axis $\bar{u}$ is given by qvq^{-1} where

$$q = [\cos\frac{\theta}{2}, (\sin\frac{\theta}{2})\bar{u}].$$

An explanation justifying this expression proceeds below and gives intuitive insight into quaternion rotation. Any vector $\bar{v}$ can be decomposed into vectors perpendicular and parallel to any other vector $\bar{w}$. It is informative to determine the effect of quaternion multiplication on a three-vector by examining rotation about a perpendicular axis and about a parallel axis. Combining the two cases results in an

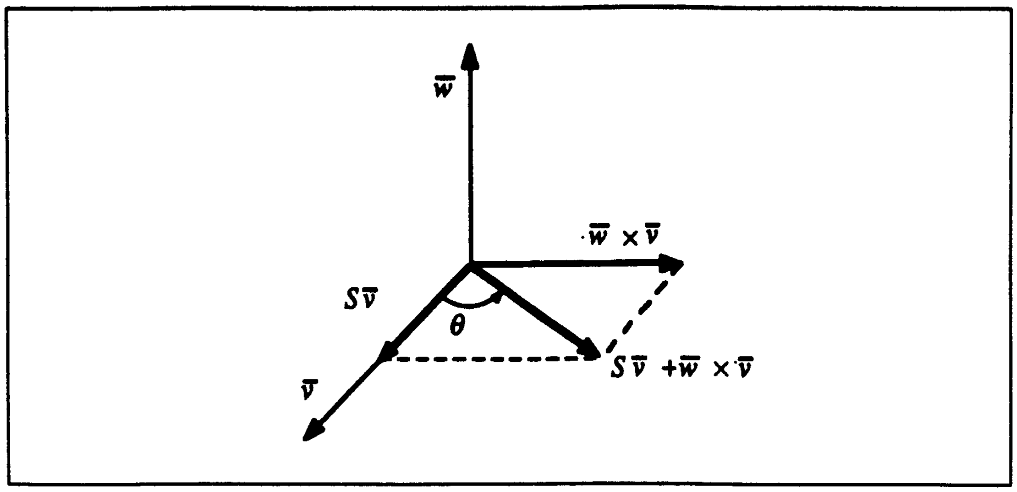

Figure 2: Effect of a quaternion on a perpendicular vector

understanding of general quaternion rotation.

2.1 Case I: Perpendicular

Suppose that the vector $\bar{v}$ is perpendicular to the vector portion of a unit quaternion $q = [S, \bar{w}]$, i.e. $\bar{v} \perp \bar{w}$. Multiplying on the left by q yields

$$\begin{aligned} qv &= [S, \bar{w}][0, \bar{v}] \\ &= [-\bar{w} \cdot \bar{v},\ S\bar{v} + \bar{w} \times \bar{v}] \\ &= [0,\ S\bar{v} + \bar{w} \times \bar{v}]. \end{aligned}$$

As the scalar portion is zero, the result represents another three-vector which is the original $\bar{v}$ rotated about $\bar{w}$, as illustrated in figure 2. To ascertain what this new vector is, it is necessary to determine its length and the angle through which it has rotated.

Using Pythagorus' Theorem, the squared length of the new vector is:

$$\begin{aligned} \| S\bar{v} + \bar{w} \times \bar{v} \|^2 &= \| S\bar{v} \|^2 + \| \bar{w} \times \bar{v} \|^2 \\ &= S^2 \| \bar{v} \|^2 + \| \bar{w} \|^2 \| \bar{v} \|^2 \\ &\qquad \text{from Lagrange's Identity, [Appendix A]} \\ &= (S^2 + \| \bar{w} \|^2) \| \bar{v} \|^2 \\ &= \| \bar{v} \|^2 \qquad \text{since } S^2 + \| \bar{w} \|^2 = \| q \| = 1. \end{aligned}$$

The length of the resulting vector is the same as the length of the original vector. It has, however, been rotated through an angle, θ, about the axis $\bar{w}$.

$$\cos\theta = \frac{\| S\bar{v} \|}{\| \bar{v} \|} = S.$$

The angle of rotation determines the first element or scalar portion of the quaternion, while the axis of rotation determines the vector portion. This vector portion must be chosen to ensure that the norm of the quaternion is one.

$$\begin{aligned} S^2 + \| \bar{w} \|^2 &= 1 \\ \cos^2\theta + \| \bar{w} \|^2 &= 1 \\ \| \bar{w} \| &= \sin\theta. \end{aligned}$$

Thus, $\bar{w} = (\sin\theta)\cdot$(unit axis of the rotation). Multiplying by a unit quaternion causes rotation of a vector that is perpendicular to the vector portion of the quaternion. The parallel case must now be considered, after which it shall be necessary to return to the perpendicular case.

2.2 Case II: Parallel

Now assume that the vector $\bar{v}$ is a scalar multiple of the vector portion of q, i.e. $A\bar{v} = \bar{w}$ for some scalar A. Rotation of a vector, which lies on the axis of rotation, should leave that vector unaltered. To determine if this happens with quaternions, multiply by q:

$$\begin{aligned} qv &= [S, \bar{w}][0, \bar{v}] \\ &= [-\bar{w}\cdot\bar{v},\ S\bar{v} + \bar{w}\times\bar{v}] \\ &= [-\bar{w}\cdot\bar{v},\ S\bar{v}]. \end{aligned}$$

As the scalar portion of this result is not zero, qv does not represent purely a rotation of v and the quantity has no intuitive meaning. To obtain a quaternion with a null scalar, one must examine qvq^{-1}.

$$\begin{aligned} qvq^{-1} &= [-\bar{w}\cdot\bar{v},\ S\bar{v}][S,\ -\bar{w}] \\ &= [-S\bar{w}\cdot\bar{v} + S\bar{v}\cdot\bar{w},\ (\bar{w}\cdot\bar{v})\bar{w} + S^2\bar{v} - S\bar{v}\times\bar{w}] \\ &= [0,\ S^2\bar{v} + (\bar{w}\cdot\bar{v})\bar{w}]. \end{aligned}$$

Further simplification of the vector portion confirms the expected identity.

$$\begin{aligned} S^2\bar{v} + (\bar{w}\cdot\bar{v})\bar{w} &= S^2\bar{v} + (A\bar{v}\cdot\bar{v})A\bar{v} \\ &= S^2\bar{v} + (A\bar{v}\cdot A\bar{v})\bar{v} \\ &= (S^2 + \| \bar{w} \|^2)\bar{v} \\ &= \bar{v}. \end{aligned}$$

It is essential to return to case I and determine how qvq^{-1} fares in the perpendicular case. Look at vq^{-1}:

$$\begin{aligned} vq^{-1} &= [0, \bar{v}][S, -\bar{w}] \\ &= [\bar{v}\cdot\bar{w}, S\bar{v} - \bar{v}\times\bar{w}] \\ &= [0, S\bar{v} + \bar{w}\times\bar{v}] \\ &= qv. \end{aligned}$$

In the perpendicular case, qvq^{-1} rotates the vector $\bar{v}$ twice as far about the same axis as qv. Thus qvq^{-1} is a general method for quaternion rotation. The next section summarizes this discussion, yielding a general formula for quaternion rotation.

Rotation of vector $\bar{v}$ by θ about the unit axis $\bar{u}$ is given by the vector portion of

$$qvq^{-1} = [0, \bar{v} + 2S(\bar{w} \times \bar{v}) + 2\bar{w} \times (\bar{w} \times \bar{v})]$$

where

$$q = [S, \bar{w}] = [\cos\frac{\theta}{2}, (\sin\frac{\theta}{2})\bar{u}]$$

Figure 3: Quaternion rotation (See [Funda 1988] for this simplified formula.)

2.3 General Quaternion Rotations

Suppose that $q = [S, \bar{w}]$ is a unit quaternion, where $S = \cos\alpha$ and $\bar{w} = (\sin\alpha)\cdot$(unit axis of rotation). Any vector $\bar{v}$ can be written as the sum of two vectors:

1. a part perpendicular to $\bar{w}$ and
2. a part parallel to $\bar{w}$.[2]

Combining cases I and II indicates that qvq^{-1} is a rotation which leaves the portion of $\bar{v}$ parallel to the axis alone and rotates the perpendicular part by 2α. This is summarized in figure 3.

It is possible to formulate quaternion rotation of a vector $\bar{v}$ as

$$\begin{aligned} qv_{\perp} + v_{\parallel} &= [0,\ S\bar{v} + (1-S)\frac{\bar{v}\cdot\bar{w}}{\|\bar{w}\|^2}\bar{w} + \bar{v} \times \bar{w}] \\ &= [0,\ \bar{v} - \bar{w} \times \bar{v} + \frac{1}{S+1}\bar{w} \times (\bar{w} \times \bar{v})] \end{aligned}$$

where $q = [S, \bar{w}] = [\cos\theta,\ (\sin\theta)\bar{u}]$ and $\bar{v}_{\perp}$ is the part of $\bar{v}$ which is perpendicular to the axis of rotation $\bar{u}$ while $\bar{v}_{\parallel}$ is parallel to $\bar{u}$. This method of representing rotations should not be used since it results in problems when θ is an odd multiple of π, i.e. $q = [-1, \bar{0}]$. In this case, the axis of rotation is lost in the representation, making it impossible to manipulate orientation using quaternions alone. For rotations of π the axis of rotation must be known in order to perform the transformation. When $q = [1, \bar{0}]$, θ is an even multiple of π and the absence of the axis has no effect since a vector rotated by zero degrees will always remain the same, independent of this axis. Another problem faced by this formulation arises during the composition of rotations. This is unmanagable as the quaternions cannot simply be multiplied. Each rotation must be kept separate and applied sequentially so that new perpendicular and parallel vectors can be calculated.

Using qvq^{-1} for rotations, as shown in figure 3, alleviates the aforementioned uncertainties. Here the axis of rotation is explicitly present, except when θ is an even multiple of π. No problem arises, since these rotations do not depend on the axis of the rotation — the orientation of the object remains the

[2]See any first year algebra textbook, e.g. [Anton 1981].

same. Composition of rotations is now well-defined as quaternion multiplication since

$$q_2(q_1vq_1^{-1})q_2^{-1} = (q_2q_1)v(q_2q_1)^{-1}.$$

Multiplying by the degenerate quaternions, $[1, \bar{0}]$ and $[-1, \bar{0}]$, creates no problems since they either have no effect or change the sign on the final result. But changing the sign on a quaternion preserves the rotation, as is observed from

$$(-q)v(-q^{-1}) = - - qvq^{-1} = qvq^{-1}.$$

Every orientation can be uniquely expressed as a quaternion lying on one hemisphere of the 4-D unit sphere. The biggest advantage of using quaternions, rather than matrices, is that the angle and axis of rotation are explicitly represented.

3 Interpolation

Often when using a manipulator it is necessary to move the end effector on a controlled path, the simplest of which is a linear path. Moving the "hot spot"[3] of a robot smoothly on a straight line is simple linear interpolation of Cartesian three-space, as shown first in this section. Thought must also be given to the change in the orientation of the gripper. In some cases, this may not be an issue since orientation can be changed once — at the end of the move. Other times, such drastic motions are intolerable anywhere along the path. It is essential that the orientation change evenly throughout. Any general linear interpolation would combine both location and orientation interpolation.

3.1 Location (Linear) Interpolation

Moving from a location p_1 to a new location p_2, as t goes from one to zero, is given by taking a fraction of the difference between the two points.

$$new\_location(t) = t(p_1) + (1-t)p_2 = p_2 - t(p_2 - p_1).$$

3.2 Orientation (Spherical) Interpolation

If orientation is specified as roll, pitch, and yaw, it may seem that smooth motion over time can be accomplished by interpolating each of these angles, changing $roll_1$ into $roll_2$, and so forth. When this is done, the orientation changes radically since the object is revolving about three different axes at the same time. It is essential to find a single angle which changes the first orientation into the second. This angle is interpolated for a smooth orientation change.

A natural way to handle orientation is through unit quaternions, since each orientation is represented as (the cosine of) an angle and an axis. Just as in three-space smooth motion occurs on a straight line — the shortest path between two vectors — the four space interpolation path must travel the shortest path between two quaternions. Since orientations lie on the unit 4-D sphere, interpolation involves traversing the "arc" joining two unit quaternions. Interpolating roll, pitch, and yaw did not work because the path was jumping all over the unit 4-D sphere, as suggested in the 3-D interpretation

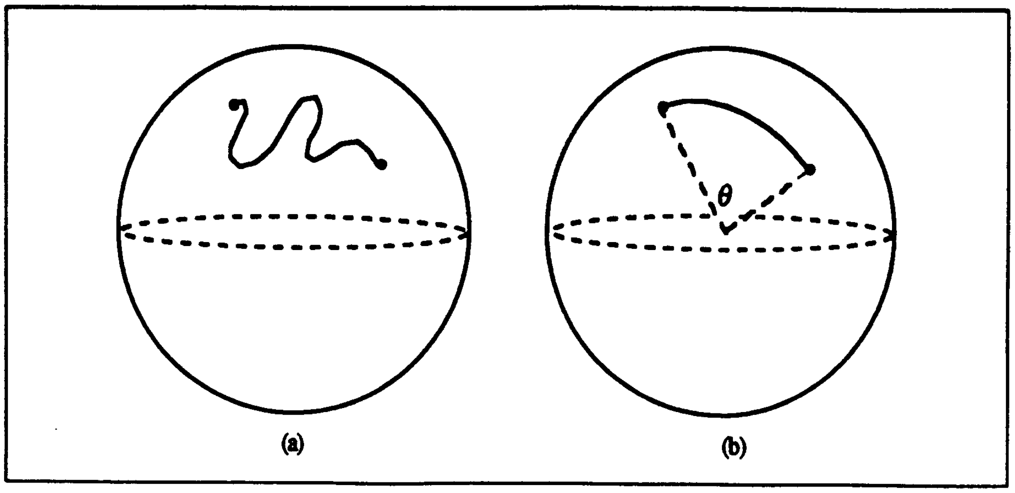

Figure 4: Interpolating on the sphere.

of figure 4(a). Shown in part (b) of the figure is the desired, shortest path. The equation for this arc, the interpolation path, for moving from q_1 to q_2 as t goes from one down to zero is:

$$new\_orientation(t) = \frac{\sin(t\theta)}{\sin\theta} q_1 + \frac{\sin[(1-t)\theta]}{\sin\theta} q_2 \tag{1}$$

where $\cos\theta = q_1 \cdot q_2$.

To remove any doubts about the validity of this formula, figure 5 shows that the orientation path always lies on the unit 4-D sphere. Figure 6 then shows that at any t the angle travelled is $(1-t)\theta$, indicating that the arc is being followed. The equation does what is required — except when $\sin\theta = 0$, i.e. $\theta = 0, \pi, -\pi$. When θ is close to zero, the arc between the two quaternions looks much like a straight line. Linear interpolation of four-space should be done, using the formula of the previous section. If $\theta = \pm\pi$ then $q_1 = -q_2$ and both quaternions represent the same orientation. Should it be necessary to move between two equivalent orientations, then the path must be divided into two rotations, each of $\frac{\pi}{2}$ radians: from q_1 to a midpoint quaternion q_{mid} and from q_{mid} to q_2. There are many choices for q_{mid}, provided that $q_1 \cdot q_{mid} = 0$. For example, if $q = [S, X, Y, Z]$, then a simple choice for q_{mid} is $[X, -S, Z, -Y]$. The same interpolation formula (somewhat reduced since $\sin\theta = 1$) is used on both parts of the path.

[Taylor 1986] provides a brief description of an alternate formulation for quaternion interpolation. He determines a quaternion q_{int} which transforms q_1 into q_2 through composition, i.e. $q_{int} = q_1^{-1}q_2$. From q_{int} the angle θ and axis of rotation $\bar{u}$ are determined, so that quaternion interpolation from q_1 to q_2 as t goes from one to zero is

$$q_1[\cos\frac{(1-t)\theta}{2}, \sin\frac{(1-t)\theta}{2}\bar{u}].$$

The computational requirements of this algorithm exceeds that of the great arc formula in equation 1.

[3]The middle point between the two fingers.

Check that the length of the quaternion *new_orientation*(t) is one for any $t \in [0, 1]$.

$$\begin{aligned}
\| \, new\_orientation \, \|^2 &= new\_orientation \cdot new\_orientation \\
&= \sin^2 t\theta \sin^2 \theta \, q_1 \cdot q_1 + \frac{2 \sin t\theta \sin(1-t)\theta}{\sin^2 \theta} q_1 \cdot q_2 \\
&\quad + \frac{\sin^2 (1-t)\theta}{\sin^2 \theta} q_2 \cdot q_2 \\
&= \frac{\sin^2 t\theta + 2 \sin t\theta \sin(\theta - t\theta) \cos \theta + \sin^2(\theta - t\theta)}{\sin^2 \theta} \\
&\qquad \text{Expand this using the identity} \\
&\qquad \sin(\alpha - \beta) = \sin \alpha \cos \beta - \sin \beta \cos \alpha \\
&= \frac{1}{\sin^2 \theta}(\sin^2 t\theta + 2 \sin t\theta \sin \theta \cos t\theta \cos \theta - 2 \sin^2 t\theta \cos^2 t\theta \\
&\qquad + \sin^2 \theta \cos^2 t\theta - 2 \sin \theta \cos t\theta \sin t\theta \cos \theta + \sin^2 t\theta \cos^2 \theta) \\
&= \frac{\sin^2 t\theta \, (1 - \cos^2 \theta) + \sin^2 \theta \cos^2 t\theta}{\sin^2 \theta} \\
&= \frac{\sin^2 \theta \, (\sin^2 t\theta + \cos^2 t\theta)}{\sin^2 \theta} \\
&= 1.
\end{aligned}$$

Figure 5: Ensuring that the orientation path lies on unit sphere.

Showing that the angle travelled at any t is $(1-t)\theta$, thus the orientation path is an arc. If α is this angle, then

$$\begin{aligned}
\cos\alpha &= q_1 \cdot (\frac{\sin(t\theta)}{\sin\theta} q_1 + \frac{\sin[(l-t)\theta]}{\sin\theta} q_2) \\
&= \frac{\sin t\theta}{\sin\theta} q_1 \cdot q_1 + \frac{\sin(\theta - t\theta)}{\sin\theta} q_1 \cdot q_2 \\
&= \frac{\sin t\theta}{\sin\theta} + \frac{\sin(\theta - t\theta)\cos\theta}{\sin\theta} \\
&= \frac{\sin t\theta + \sin\theta\cos t\theta\cos\theta - \sin t\theta\cos^2\theta}{\sin\theta} \\
&= \frac{\sin t\theta(1-\cos^2\theta) + \sin\theta\cos t\theta\cos\theta}{\sin\theta} \\
&= \frac{\sin\theta(\sin\theta\sin t\theta + \cos\theta\cos t\theta)}{\sin\theta} \\
&= \cos(\theta - t\theta).
\end{aligned}$$

As $\alpha = 0$ when $t = 1$ and $\alpha = \theta$ when $t = 0$, the above equation indicates that $\alpha = (1-t)\theta$ for arbitrary $t \in [0,1]$.

Figure 6: Interpolation path traces out arc.

Operation[a]	Quaternions	Matrices
Rotating a Vector	15 M, 12 A	9 M, 6 A
Composition of Rotations	16 M, 12 A	24 M, 15 A
Setting up the rotation from an angle and a unit axis	4 M, 1 A, 1 Sqrt, 1 Trig	22 M, 11 A, 1 Sqrt, 1 Trig
Extracting angle and axis from rotation	4 M, 1 A, 1 Trig, 1 Sqrt	10 M, 16 A, 2 Sqrt, 1 Trig
Interpolation — finding the next rotational knot point	8 M, 4 A, 2 Trig	30 M, 15 A

[a]References to the formulae used in calculating the operation counts appear in appendix B

Table 2: Operation counts for rotation tasks.

Comparisons between quaternions and matrices in rotational tasks are given in table 2. Notice that the complexity involved in using quaternions is lower than in using matrices for all tasks except vector rotation. Even here, quaternions are generally the preferred representation since setting up the matrix is more complex than determining the quaternion for rotation.

4 Implementation on a Robot Arm

Manipulator positions are measured in joint coordinates, that is, the position of each link is an angle relative to the previous link. The motion interpolation described in this paper assumes that position is a location and a quaternion, hence joint coordinates must be converted into this form. Software packages controlling a robot generally contain forward kinematics allowing transformation between joint coordinates and world coordinates, which are a Cartesian location and an orientation. This orientation is conventionally specified as:

1. three vectors ($\bar{n}$, $\bar{o}$, and $\bar{a}$) which form an orientation matrix, or
2. a sequence of angles such as *roll*, *pitch*, and *yaw*.

Details of the conversion between joint angles and these forms of orientation are well-known [Paul 1981], yet expressing orientation as a quaternion is rare. This section bridges the gap, showing the relation from matrices and sequences of angles to quaternions. Figure 7 diagrams the steps involved in robot straight-line motion — starting with joint angles, converting to a location and a quaternion, performing the interpolation, and finally, converting back to joint angles to move the robot. Ideally, the two extra steps necessary to convert between conventional orientation and quaternions would be alleviated, resulting in a direct path between joint angles and quaternions. More research in quaternion kinematics of general manipulators is necessary, but [Funda 1988] provides a good example in the application of quaternions to solve the inverse kinematics of a Puma robot arm. Following the conversion between the three equivalent forms of orientation, considerations for implementing straight-line motion on a robot are presented. A quaternion based interpolation method, following figure 7, has been implemented on the Excalibur robot.

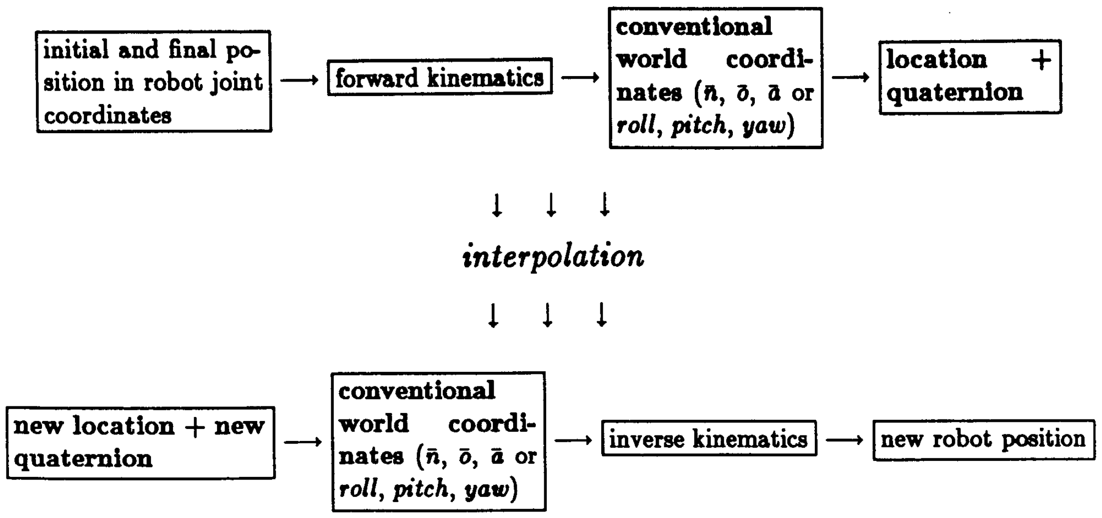
initial and final position in robot joint coordinates
forward kinematics
conventional world coordinates ($\bar{n}$, $\bar{o}$, $\bar{a}$ or *roll*, *pitch*, *yaw*)
location + quaternion
interpolation
new location + new quaternion
conventional world coordinates ($\bar{n}$, $\bar{o}$, $\bar{a}$ or *roll*, *pitch*, *yaw*)
inverse kinematics
new robot position

Figure 7: Robot motion interpolation

4.1 Quaternion to Matrix ($\bar{n}$, $\bar{o}$, $\bar{a}$)

The quickest way to determine the corresponding matrix, M, for any transformation is to investigate its effects on the standard basis.[4] Applying $q = [S, X, Y, Z]$ to element m in the standard basis yields the mth column of the matrix:

$$M = \begin{pmatrix} | & | & | \\ n & o & a \\ | & | & | \end{pmatrix} = \begin{pmatrix} 1-2Y^2-2Z^2 & 2XY-2SZ & 2XZ+2SY \\ 2XY+2SZ & 1-2X^2-2Z^2 & 2YZ-2SX \\ 2XZ-2SY & 2YZ+2SX & 1-2X^2-2Y^2 \end{pmatrix}. \qquad (2)$$

4.2 Matrix ($\bar{n}$, $\bar{o}$, $\bar{a}$) to Quaternion

To convert from an orthonormal matrix to a unit quaternion, assume that the matrix has the form of equation 2 and find $q = [S, X, Y, Z]$. This is accomplished by investigating linear combinations of the matrix components. First, examine the trace

$$\begin{aligned} trace + 1 &= 1-2Y^2-2Z^2+1-2X^2-2Z^2+1-2X^2-2Y^2+1 \\ &= 4-4(X^2+Y^2+Z^2) \\ &= 4-4(1-S^2) \quad \text{since } \| q \| = 1 \\ &= 4S^2. \end{aligned}$$

Thus, $S = \frac{1}{2}\sqrt{trace+1}$. Combining the M_{ij} element with the M_{ji} element yields the axis of rotation, which is easily normalized if necessary:

$$X = \frac{M_{32} - M_{23}}{4S}$$

[4]In R^3 this is $\{(1,0,0), (0,1,0), (0,0,1)\} = \{\mathbf{i}, \mathbf{j}, \mathbf{k}\}$. The order of the elements is important.

$$\begin{aligned} Y &= \frac{M_{13} - M_{31}}{4S} \\ X &= \frac{M_{21} - M_{12}}{4S}. \end{aligned}$$

When $S = 0$, these equations are undefined and other combinations of the simplified matrix components along with the identity $X^2 + Y^2 + Z^2 = 1$ are used to determine the axis of rotation. A full description appears in [Shoemake 1985].

4.3 Roll, Pitch, and Yaw to Quaternion

A series of rotations is converted into a quaternion by converting each individual rotation into a quaternion and multiplying them together in the proper order. Expressing each of *roll*, *pitch*, and *yaw* as a quaternion yields:

$$\begin{aligned} q_{roll} &= [\cos\frac{roll}{2},\ (0,0,\sin\frac{roll}{2})] \\ q_{pitch} &= [\cos\frac{pitch}{2},\ (0,\sin\frac{pitch}{2},0)] \\ q_{yaw} &= [\cos\frac{yaw}{2},\ (\sin\frac{yaw}{2},0,0)]. \end{aligned}$$

Multiplying these together as $q_{yaw}q_{pitch}q_{roll} = [S, X, Y, Z]$ gives the desired quaternion with

$$\begin{aligned} S &= \cos\frac{yaw}{2}\cos\frac{pitch}{2}\cos\frac{roll}{2} - \sin\frac{yaw}{2}\sin\frac{pitch}{2}\sin\frac{roll}{2} \\ X &= \cos\frac{yaw}{2}\sin\frac{pitch}{2}\sin\frac{roll}{2} + \sin\frac{yaw}{2}\cos\frac{pitch}{2}\cos\frac{roll}{2} \\ Y &= \cos\frac{yaw}{2}\sin\frac{pitch}{2}\cos\frac{roll}{2} - \sin\frac{yaw}{2}\cos\frac{pitch}{2}\sin\frac{roll}{2} \\ Z &= \cos\frac{yaw}{2}\cos\frac{pitch}{2}\sin\frac{roll}{2} + \sin\frac{yaw}{2}\sin\frac{pitch}{2}\cos\frac{roll}{2}. \end{aligned}$$

These formulae[5] specify the quaternion uniquely, though $-q$ induces the same rotation on a vector as q.

4.3.1 Quaternion to Roll, Pitch, and Yaw

Conversion in the other direction is much more difficult, since roll, pitch, and yaw angles are not unique. Inverting the previous equations to solve for roll, pitch, and yaw is practically impossible, so another method must be sought. If a transformation is represented in matrix form it is easy to determine the corresponding angles [Paul 1981].

To determine roll, pitch, and yaw, only seven of the matrix elements of equation 2 are required. If M_{ij} is the element occurring in the ith row and jth column of the matrix, then using the formulae given

[5] Different from [Shoemake 1985], since he does not account for the usual *sign* on the angles since he treats quaternion rotation as $q^{-1}vq$.

in [Paul 1981] yields:

$$\begin{aligned}
roll &= \begin{cases} 0 & \text{if both } M_{11} \text{ and } M_{21} \text{ are } 0 \\ atan(M_{21}, M_{11}) & \text{otherwise} \end{cases} \\
pitch &= atan(-M_{31}, M_{11}(\cos roll) + M_{21}(\sin roll))) \\
yaw &= atan(M_{13}(\sin roll) - M_{23}(\cos roll),\ M_{22}(\cos roll) - M_{12}(\sin roll)).
\end{aligned}$$

4.3.2 Choosing the Knot Points

The formulae given in the last two sections assumed t went continuously from 1 down to 0. When applying these to a machine discrete points from within this interval will have to be used as knot points. These knot points may need to be chosen in an application specific manner so that any error made on the path between the points is tolerable. [Brady 1986] suggests a control rate falling in the range of $20Hz$ to $200Hz$ so that the motion is smooth. He further suggests using joint interpolation, which is computationally less expensive, in between knot points so that the points are much closer in time than the natural period of the arm. [Taylor 1986] presents an approximate method for bounding the deviation from interpolated paths, by recursively halving the distance between knot points until a satisfactory deviation is obtained half-way between the points. The method reasonably assumes the mid-point deviation to be approximately the worst error over the segment.

4.3.3 Problems in Application to a Robot

In an implementation of straight line motion on a manipulator many problems arise. These difficulties are primarily due to the geometry of each robot arm. Three such issues, whose solutions are often manipulator and task dependent, or non-existent, are now described.

Although the manipulator can reach both endpoints, the straight line joining them may contain points which cannot be attained (because it would cause the robot to move "through itself" or it requires a joint position beyond the limits of the robot). It is difficult to predict such conditions without calculating the line and checking that no points are out of reach. This is computationally expensive. A better solution may be to start the linear path, stopping the manipulator when the unreachable point occurs. Verifying attainable positions must be done at each step of the path. This problem can be solved by pre-motion planning, so long as obstacles are known. See for example the recent and excellent work of [Canny 1988].

Degenerate manipulator configurations and redundant configurations reaching the same position are another source of problems in any controlled motion, and Cartesian interpolation breaks down under degeneracy [Paul 1981]. Is it possible to calculate all manipulator configurations which attain the required position? If so, it will be computationally expensive. Furthermore, how does one know which joint arrangement should be used?

Finally, small transitions in Cartesian position may cause huge changes in joint positions. This results in the manipulator moving irregularly. Even though the end effector travels linearly, delays in the motion may occur. The effects are difficult to predict [Paul 1981]. No solution to this problem exists. However, when there are additional degrees of freedom, above six, the extra joints might be used to smooth out these irregularities.

5 Conclusion

This paper describes straight line motion of objects, interpolating both location and orientation. Linear interpolation in Cartesian three-space is well-known, hence orientation was the focus of concentration. The recently revived method of quaternions is used. Rather than just presenting the formulae, an intuitive understanding of quaternions is encouraged by our careful explanation, showing how they relate to vectors, matrices, and roll, pitch, and yaw. Quaternion multiplication plays the main role in rotation, providing an alternative to the usual matrix multiplication. The greatest advantage quaternions have over matrices is the ease of (almost) explicit representation. A quaternion contains four components — the cosine of half the angle of rotation and the three-vector axis of rotation. It was shown that the quaternion method is superior to standard matrix techniques in general and in comparison of most standard tasks involved in vector rotation. Quaternion interpolation was successfully used in the implementation of straight-line motion for a robot. Uses for quaternions, however, extend far beyond robotics.

Acknowledgements

This work is supported by the Natural Sciences and Engineering Research Council of Canada and by a Province of Alberta Scholarship. Thanks to Jon Rokne for help at an important moment during the development of this paper.

References

[1] Anton, H., *Elementary Linear Algebra*, Toronto: John Wiley & Sons, 1981.

[2] Brady, M., "Trajectory Planning," in *Robot Motion: Planning and Control*, M. Brady, J.M. Hollerbach, T.L. Johnson, T. Lozano-Perez, and M.T. Mason (Eds.), Cambridge: The MIT Press, 1986.

[3] Canny, J.F., *The Complexity of Robot Motion Planning*, Cambridge: The MIT Press, 1988.

[4] Funda, J., *Quaternions and Homogeneous Transforms in Robotics*, Master's Thesis, Department of Computer and Information Science, The University of Pennsylvania, Philadelphia, 1988.

[5] Hamilton, Sir W.R., *Elements of Quaternions, Volume I*, Third Ed., New York: Chelsea Publishing Co., 1969.

[6] Herstein, I.N., *Topics in Algebra*, 2nd ed., Toronto: Wiley and Sons, 1975.

[7] Kelland, P., and Tait, P.G., *Introduction to Quaternions*, C.G. Knott (Prep.), New York: The MacMillan Company, 1904.

[8] Paul, R.P., *Robot Manipulators: Mathematics, Programming, and Control*, Cambridge: The MIT Press, 1981.

[9] Pletincks, D., "The Use of Quaternions for Animation, Modelling and Rendering", in *New Trends in Computer Graphics, Proceedings of CG International '88*, N. Magnenat and D. Thalmann (Eds.), New York: Springer-Verlag, 1988.

[10] Shoemake, K., "Animating Rotation with Quaternion Curves," in *Computer Graphics, 19(3)*, Siggraph, 1985.

[11] Shoemake, K., "Quaternion Calculus and Fast Animation," in *Siggraph 87 Course 10: "Computer Animation: 3D Motion Specification and Control,"* 1987.

[12] Taylor, R.H., "Planning and Execution of Straight-line Manipulator Trajectories," in *Robot Motion: Planning and Control*, M. Brady, J.M. Hollerbach, T.L. Johnson, T. Lozano-Perez, and M.T. Mason (Eds.), Cambridge: The MIT Press, 1986.

A Lagrange's Identity

Lagrange's identity states that

$$\| \bar{u} \times \bar{v} \|^2 = \| \bar{u} \|^2 \| \bar{v} \|^2 - (\bar{u} \cdot \bar{v})^2.$$

This can be shown by working out both sides of the equation from the definition of the parts. If θ is the angle between the two vectors, then substituting the definition for the dot product into the identity results in

$$\| \bar{u} \times \bar{v} \|^2 = \| \bar{u} \|^2 \| \bar{v} \|^2 - (\| \bar{u} \| \| \bar{v} \| \cos\theta) = \| \bar{u} \|^2 \| \bar{v} \|^2 (1 - \cos\theta)^2.$$

This leads to a formula for the length of a cross product

$$\| \bar{u} \times \bar{v} \| = \| \bar{u} \| \| \bar{v} \| \sin\theta.$$

B Notes on Formulae Used in Operations Counts

This appendix provides extra reference to the formulae used in calculating operation complexity for table 2.

Rotating a Vector

Quaternion: Using the formula given in figure 3.

Matrices: Pre-multiplying a three-vector by a 3×3 matrix.

Composition of Rotations

Quaternion: Using the formula from table 1.

Matrices: The first two columns obtained by matrix multiplication and the last column as the cross product of the first two.

Setting up the rotation from an angle and a unit axis

Quaternion: Specification of q as in figure 3.

Matrices: Formula on page 28 of [Paul 1981].

Extracting angle and axis from rotation

Quaternion: Inverting specification of q in figure 3.

Matrices: Method on page 19 of [Funda 1988].

Interpolation — finding the next rotational knot point

Quaternion: The calculation assumes that the endpoint quaternions have already been scaled by $\sin\theta$. An extra 6 M and 1 Trig are necessary for this once at the beginning of interpolation.

Matrices: It is unclear what the most efficient formulation of matrix-based orientation interpolation is. If the two endpoints are expressed as the matrices M_1 and M_2 then $M_{int} = M_1^T M_2$ is the matrix which changes M_1 into M_2. As t goes from zero to one the interpolated orientation is

$$M_1[f(t)M_{int}]$$

where $f(t)$ is a continuous function with $f(1) = 1$ and $f(0) = 0$. This formula is used for the operations counts, which may be higher depending on $f(t)$.