1 Using Quaternions to Represent Rotation

1.1 Complex Numbers (a brief review)

As you know, the system of complex numbers is defined in terms of $i$, a square root of $-1$.

$$i * i = -1$$

Although $i$ is not a real number, we can write any complex number in terms of real numbers, using a real and a complex part. We can also define a conjugate and a magnitude (or absolute value) for these numbers.

$$z = a + bi$$

$$z' = a - bi$$

$$|z| = \sqrt{z * z'} = \sqrt{a^2 + b^2}$$

Using the properties above, we can describe multiplication for complex numbers

$$z_1 = a_1 + b_1 i$$

$$z_2 = a_2 + b_2 i$$

$$z_1 * z_2 = (a_1 a_2 - b_1 b_2) + (a_1 b_2 + b_1 a_2) i$$

1.2 Quaternions

Quaternions are an extension of complex numbers. Instead of just $i$, we have three different numbers that are all square roots of $-1$ labelled $i$, $j$, and $k$.

$$i * i = -1$$

$$j * j = -1$$

$$k * k = -1$$

When you multiply two of these numbers together, they behave similarly to cross products of the unit basis vectors.

$$i * j = -j * i = k$$

$$j * k = -k * j = i$$

$$k * i = -i * k = j$$

The conjugate and magnitude of a quaternion are found in much the same way as complex conjugate and magnitude. If a quaternion $q$ has length 1, we say that $q$ is a unit quaternion.

$$q = w + xi + yj + zk$$

$$q' = w - xi - yj - zk$$

$$|q| = \sqrt{q * q'} = \sqrt{w^2 + x^2 + y^2 + z^2}$$

Unit quaternions $|q| = 1 \Rightarrow q^{-1} = q'$

Quaternions are associative $(q_1 * q_2) * q_3 = q_1 * (q_2 * q_3)$

Quaternions are not commutative $q_1 * q_2 \neq q_2 * q_1$
We can represent a quaternion in several ways,

- as a linear combination of $1$, $i$, $j$, and $k$,
- as a vector of the four coefficients in this linear combination,
- or as a scalar for the coefficient of $1$ and a vector for the coefficients of the imaginary terms.

$$q = w + xi + yj + zk = [x \ y \ z \ w] = (s, v)$$

We can write the product of two quaternions in terms of the $(s, v)$ representation using standard vector products in the following way:

$$q_1 = (s_1, v_1)$$
$$q_2 = (s_2, v_2)$$
$$q_1 * q_2 = (s_1s_2 - v_1 \cdot v_2, s_1v_2 + s_2v_1 + v_1 \times v_2)$$

### 1.3 Representing Rotations with Quaternions

We will compute a rotation about the unit vector, $u$ by an angle $\theta$. The quaternion that computes this rotation is

$$q = (s, v) = \left(\cos \frac{\theta}{2}, u \sin \frac{\theta}{2}\right)$$

We will represent a point $p$ in space by the quaternion $P = (0, p)$. We compute the desired rotation about that point by

$$P = (0, p)$$
$$P_{\text{rotated}} = q * P * q^{-1}$$

Now, the quaternion $P_{\text{rotated}}$ should be $(0, p_{\text{rotated}})$. Actually, we could put any value into the scalar part of $P$, i.e. $P = (c, p)$ and after performing the quaternion multiplication, we should get back $P_{\text{rotated}} = (c, p_{\text{rotated}})$.

You may want to confirm that $q$ is a unit quaternion, since that will allow us to use the fact that the inverse of $q$ is $q'$ if $q$ is a unit quaternion.

### 1.4 Concatenating Rotations

Suppose we want to perform two rotations on an object. This may come up in a manipulation interface where each movement of the mouse adds another rotation to the current object pose. This is very easy and numerically stable with a quaternion representation.

Suppose $q_1$ and $q_2$ are unit quaternions representing two rotations. We want to perform $q_1$ first and then $q_2$. To do this, we apply $q_2$ to the result of $q_1$, regroup the product using associativity, and find that the composite rotation is represented by the quaternion $q_2 * q_1$.

$$q_2 * (q_1 * P * q_1^{-1}) * q_2^{-1} = (q_2 * q_1) * P * (q_1^{-1} * q_2^{-1}) = (q_2 * q_1) * P * (q_2 * q_1)^{-1}$$

Therefore, the only time we need to compute the matrix is when we want to transform the object. For other operations we need only look at the quaternions. A matrix product requires many more operations than a quaternion product so we can save a lot of time and preserve more numerical accuracy with quaternions than with matrices.
2 A Matrix Representation for Quaternion Multiplication

We can use the rules above to compute the product of two quaternions.

\[ q_1 = w_1 + x_1 i + y_1 j + z_1 k \]
\[ q_2 = w_2 + x_2 i + y_2 j + z_2 k \]
\[ q_1 \cdot q_2 = (w_1 w_2 - x_1 x_2 - y_1 y_2 - z_1 z_2) + (w_1 x_2 + x_1 w_2 + y_1 z_2 - z_1 y_2)i + (w_1 y_2 - x_1 z_2 + y_1 w_2 + z_1 x_2)j + (w_1 z_2 + x_1 y_2 - y_1 x_2 + z_1 w_2)k \]

If you examine each term in this product, you can see that each term depends linearly on the coefficients for \( q_1 \). Also each term depends linearly on the coefficients for \( q_2 \).

So, we can write the product of two quaternions in terms of a matrix multiplication.

When the matrix \( L_{\text{ROW}}(q_1) \) multiplies a row vector \( q_2 \), the result is a row vector representation for \( q_1 \cdot q_2 \).

When the matrix \( R_{\text{ROW}}(q_2) \) multiplies a row vector \( q_1 \), the result is also a row vector representation for \( q_1 \cdot q_2 \).

\[ q_1 \cdot q_2 = q_2 L_{\text{ROW}}(q_1) = \begin{bmatrix} x_2 & y_2 & z_2 & w_2 \end{bmatrix} \begin{bmatrix} w_1 & z_1 & -y_1 & -x_1 \\ -z_1 & w_1 & x_1 & -y_1 \\ y_1 & -x_1 & w_1 & -z_1 \\ x_1 & y_1 & z_1 & w_1 \end{bmatrix} \]
\[ q_1 R_{\text{ROW}}(q_2) = \begin{bmatrix} x_1 & y_1 & z_1 & w_1 \end{bmatrix} \begin{bmatrix} w_2 & -z_2 & y_2 & -x_2 \\ z_2 & w_2 & -x_2 & -y_2 \\ -y_2 & x_2 & w_2 & -z_2 \\ x_2 & y_2 & z_2 & w_2 \end{bmatrix} \]

2.1 Computing Rotation Matrices from Quaternions

Now we have all the tools we need to use quaternions to generate a rotation matrix for the given rotation. We have a matrix form for left-multiplication by \( q \)

\[ P \cdot L_{\text{ROW}}(q) = \begin{bmatrix} x_p & y_p & z_p & 0 \end{bmatrix} \begin{bmatrix} w_q & z_q & -y_q & -x_q \\ -z_q & w_q & x_q & -y_q \\ y_q & -x_q & w_q & -z_q \\ x_q & y_q & z_q & w_q \end{bmatrix} \]

and a matrix form for right-multiplication by \( q^{-1} \).

\[ q^{-1} = q' = \begin{bmatrix} -x_q & -y_q & z_q & w_q \end{bmatrix} \]
\[ P \cdot R_{\text{ROW}}(q^{-1}) = \begin{bmatrix} x_p & y_p & z_p & 0 \end{bmatrix} \begin{bmatrix} w_q & z_q & -y_q & -x_q \\ -z_q & w_q & x_q & y_q \\ y_q & -x_q & w_q & z_q \\ -x_q & -y_q & z_q & w_q \end{bmatrix} \]
The resulting rotation matrix is the product of these two matrices.

\[
Q_{\text{row}} = R_{\text{row}}(q^{-1}) \cdot L_{\text{row}}(q)
\]

\[
= \begin{bmatrix}
    w_q & z_q & -y_q & x_q \\
    -z_q & w_q & x_q & y_q \\
    y_q & -x_q & w_q & z_q \\
    -x_q & y_q & -z_q & w_q
\end{bmatrix}
\begin{bmatrix}
    w_q & z_q & -y_q & -x_q \\
    -z_q & w_q & x_q & -y_q \\
    y_q & -x_q & w_q & -z_q \\
    -x_q & y_q & -z_q & w_q
\end{bmatrix}
\]

\[
= \begin{bmatrix}
    w^2 + x^2 - y^2 - z^2 & 2wz + 2xy & 2xz - 2wy & 0 \\
    2wz - 2xy & w^2 - x^2 + y^2 - z^2 & 2yz + 2wx & 0 \\
    2xz - 2wy & 2yz - 2wx & w^2 - x^2 - y^2 + z^2 & 0 \\
    0 & 0 & 0 & w^2 + x^2 + y^2 + z^2
\end{bmatrix}
\]

Although matrices do not generally commute (in general \(AB \neq BA\)), because these matrices represent left and right multiplication and quaternion multiplication is associative, these particular matrices do commute. So I could have written \(Q_{\text{row}} = L_{\text{row}}(q) \cdot R_{\text{row}}(q^{-1})\) instead of \(Q_{\text{row}} = R_{\text{row}}(q^{-1}) \cdot L_{\text{row}}(q)\) and I would have gotten the same result. You can check that on these matrices if you like.

So using this matrix, we could compute \(P_{\text{rotated}}\) another way:

\[
P_{\text{rotated}} = P \cdot Q_{\text{row}}
\]

3 Euler Angles

In many fields Euler angles are used to represent rotations. Any rotation can be broken down into a series of three rotations about the major axes. We can simulate any arbitrary rotation with one rotation about the x-axis, one about the y-axis, and then one about the z-axis. For example, consider an airplane pointing along the x-axis with the z-axis pointing up. We can represent any pose by

- the "roll" about the x-axis along the plane,
- the "pitch" about the y-axis which extends along the wings of the plane,
- and the "yaw" or "heading" about the z-axis

as a vector \((\text{roll}, \text{pitch}, \text{yaw})\). This representation is useful and intuitive in some cases, such as this one, but Euler angles have some drawbacks.

- There is no universal standard for Euler rotations. Different fields use different sequences of Euler angles, for example some physicists use \(z-y-z\) as opposed to the \(x-y-z\) system described above.
- Since any rotation can be represented by either a set of Euler angles or a matrix, we should be able to convert between them. However, computing the required angles is expensive and can introduce errors.
- Interpolation between two poses represented in this system does not follow the great arc between the rotations, but involves wild swings about the canonical axes.

4 References