Chapter 29

Revised: The Relationship of 4D Rotations to Quaternions

29.1 What Happened in Three Dimensions

In three dimensions, there were many ways to deduce the quadratic mapping from quaternions to the 3×3 rotation matrix belonging to the group $\mathbf{SO}(3)$ and implementing a rotation on ordinary 3D frames. The one most directly derived from the quaternion algebra conjugates "pure" quaternion three-vectors $v_i = (0, \mathbf{V}_i)$ and pulls out the elements of the rotation matrix in the following way:

$$\sum_{j=1}^{3} R(q)_{ij} v_j = q \star v_i \star q^{-1}.$$

We easily find that the quadratic relationship between $\mathbf{R}_3(q)$ and $q = (q_0, q_1, q_2, q_3)$ is

$$\mathbf{R}_3 = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}. \tag{29.1}$$

29.2 Quaternions and Four Dimensions

In the 4D case, which we should really regard as the more fundamental one because it includes the 3D transformation as a special case, we can find the induced $\mathbf{SO}(4)$ matrix by extending quaternion multiplication to act on full four-vector quaternions $v_\mu = (v_0, \mathbf{V})_\mu$ and not just three-vectors ("pure" quaternions) $v = (0, \mathbf{V})$ in the following way:

$$\sum_{\nu=0}^{3} R_{\mu\nu} v_\nu = p \star v_\mu \star q^{-1}.$$

```
QQTo4DRot[p_List,q_List] :=
  Module[{p0 = p[[1]], p1 = p[[2]], p2 = p[[3]], p3 = p[[4]],
          q0 = q[[1]], q1 = q[[2]], q2 = q[[3]], q3 = q[[4]]},
{{p0*q0 + p1*q1 + p2*q2 + p3*q3,
   p0*q1 - p1*q0 - p2*q3 + p3*q2,
   p0*q2 - p2*q0 + p1*q3 - p3*q1,
   p0*q3 - p3*q0 - p1*q2 + p2*q1},
 {-p0*q1 + p1*q0 - p2*q3 + p3*q2,
   p0*q0 + p1*q1 - p2*q2 - p3*q3,
  -p0*q3 - p3*q0 + p1*q2 + p2*q1,
   p0*q2 + p2*q0 + p1*q3 + p3*q1},
 {-p0*q2 + p2*q0 + p1*q3 - p3*q1,
   p0*q3 + p3*q0 + p1*q2 + p2*q1,
   p0*q0 - p1*q1 + p2*q2 - p3*q3,
  -p0*q1 - p1*q0 + p2*q3 + p3*q2},
 {-p0*q3 + p3*q0 - p1*q2 + p2*q1,
  -p0*q2 - p2*q0 + p1*q3 + p3*q1,
   p0*q1 + p1*q0 + p2*q3 + p3*q2,
   p0*q0 - p1*q1 - p2*q2 + p3*q3}} ]
```

Table 29.1: Mathematica code for the 4×4 orthogonal rotation matrix in terms of a double quaternion.

Working out the algebra, we find that the 3D rotation matrix $\mathbf{R}_3$ is just the degenerate $p = q$ case of the following 4D rotation matrix:

$$\mathbf{R}_4 = \begin{bmatrix} p_0q_0 + p_1q_1 + p_2q_2 + p_3q_3 & p_0q_1 - p_1q_0 - p_2q_3 + p_3q_2 & p_0q_2 - p_2q_0 + p_1q_3 - p_3q_1 & p_0q_3 - p_3q_0 - p_1q_2 + p_2q_1 \\ -p_0q_1 + p_1q_0 - p_2q_3 + p_3q_2 & p_0q_0 + p_1q_1 - p_2q_2 - p_3q_3 & -p_0q_3 - p_3q_0 + p_1q_2 + p_2q_1 & p_0q_2 + p_2q_0 + p_1q_3 + p_3q_1 \\ -p_0q_2 + p_2q_0 + p_1q_3 - p_3q_1 & p_0q_3 + p_3q_0 + p_1q_2 + p_2q_1 & p_0q_0 - p_1q_1 + p_2q_2 - p_3q_3 & -p_0q_1 - p_1q_0 + p_2q_3 + p_3q_2 \\ -p_0q_3 + p_3q_0 - p_1q_2 + p_2q_1 & -p_0q_2 - p_2q_0 + p_1q_3 + p_3q_1 & p_0q_1 + p_1q_0 + p_2q_3 + p_3q_2 & p_0q_0 - p_1q_1 - p_2q_2 + p_3q_3 \end{bmatrix} . \quad (29.2)$$

One may check that Equation 29.1 is just the lower right-hand corner of the degenerate $p = q$ case of Equation 29.2. An implementation of Equation 29.2 is presented in Table 29.1.

We may take this form and plug in

$$p_0 = \cos(\phi/2), \qquad \mathbf{p} = \hat{\mathbf{m}} \sin(\phi/2)$$

to get a new form of the 4D orthogonal rotation matrix *parameterized in terms of two separate three-sphere coordinates*:

$$\mathbf{R}_4 \;=\; [A_0 \quad A_1 \quad A_2 \quad A_3], \tag{29.3}$$

where

$$A_0 \;=\; \frac{1}{2}\begin{bmatrix} C_+ + C_- + \hat{\mathbf{m}}\cdot\hat{\mathbf{n}}(C_- - C_+) \\ -m_{23}^- C_- + m_{23}^- C_+ + m_1^+ S_- + m_1^- S_+ \\ -m_{31}^- C_- + m_{31}^- C_+ + m_2^+ S_- + m_2^- S_+ \\ -m_{12}^- C_- + m_{12}^- C_+ + m_3^+ S_- + m_3^- S_+ \end{bmatrix},$$

$$A_1 \;=\; \frac{1}{2}\begin{bmatrix} -m_{23}^- C_- + m_{23}^- C_+ - m_1^+ S_- - m_1^- S_+ \\ C_+ + C_- + (m_1 n_1 - m_2 n_2 - m_3 n_3)(C_- - C_+) \\ m_{12}^+ C_- - m_{12}^+ C_+ + m_3^- S_- + m_3^+ S_+ \\ m_{31}^+ C_- - m_{31}^+ C_+ - m_2^- S_- - m_2^+ S_+ \end{bmatrix},$$

$$A_2 \;=\; \frac{1}{2}\begin{bmatrix} -m_{31}^- C_- + m_{31}^- C_+ - m_2^+ S_- - m_2^- S_+ \\ m_{12}^+ C_- - m_{12}^+ C_+ - m_3^- S_- - m_3^+ S_+ \\ C_+ + C_- + (-m_1 n_1 + m_2 n_2 - m_3 n_3)(C_- - C_+) \\ m_{23}^+ C_- - m_{23}^+ C_+ + m_1^- S_- + m_1^+ S_+ \end{bmatrix},$$

$$A_3 \;=\; \frac{1}{2}\begin{bmatrix} -m_{12}^- C_- + m_{12}^- C_+ - m_3^+ S_- - m_3^- S_+ \\ m_{31}^+ C_- - m_{31}^+ C_+ + m_2^- S_- + m_2^+ S_+ \\ m_{23}^+ C_- - m_{23}^+ C_+ - m_1^- S_- - m_1^+ S_+ \\ C_+ + C_- + (-m_1 n_1 - m_2 n_2 + m_3 n_3)(C_- - C_+) \end{bmatrix}.$$

Here, $C_\pm = \cos\frac{1}{2}(\phi \pm \theta)$, $S_\pm = \sin\frac{1}{2}(\phi \pm \theta)$, $m_i^\pm = (m_i \pm n_i)$, and $m_{ij}^\pm = (m_i n_j \pm m_j n_i)$.

Shoemake-style interpolation between two distinct 4D frames is now achieved by applying the desired SLERP-based interpolation method independently to a set of coordinates $p(t)$ on one three-sphere, and to a separate set of coordinates $q(t)$ on another. The resulting matrix $\mathbf{R}_4(t)$ gives geodesic interpolations for simple SLERPs, and smooth interpolations based on infinitesimal geodesic components when the spline methods of Chapter 25 are used in tandem on both quaternions of the pair at the same time.

Controls*:* A three-degree-of-freedom controller can in fact be used to generalize the two-degree-of-freedom rolling-ball controller [66] from 3D to 4D orientation control [34,72] . This 4D orientation control technique can be used with a 3D tracker or 3D haptic probe to carry out interactive view control or to specify keyframes for 4D double-quaternion interpolations. As pointed out by Shoemake [151] , the Arcball controller can also be adapted with complete faithfulness of spirit to the 4D case, in that one can pick *two* points in a three-sphere to specify an initial 4D frame and then pick *two more* points in the three-sphere to define the current 4D frame. Note that Equation 29.2 gives the complete 4D rotation formula. Alternatively, one can replace the 4D rolling ball or virtual sphere controls described at the beginning by a pair (or more) of 3D controllers as noted by Hanson [66].