NEW ALGORITHMS FOR IMPLEMENTING AND INTERPOLATING ROTATIONS

Bob Palais and Richard Palais

Preliminary Version March 21, 2008

Abstract. We have not found in the computer graphics literature what we believe may be the simplest algorithm for implementing a $3D$ rotation with a mouse. We present this algorithm, and observe that it leads to new mathematical realizations of the $3D$ rotation group and its double-cover, the unit quaternions. We use this to obtain an algorithm for interpolating rotations of a 3D scene that is more efficient than the standard quaternion based methods, and also to derive the quaternion composition formula from the geometry of 3D rotations.

1. Implementing prescribed rotations by reflecting pairs.

There are many methods well-known in computer graphics for implementing a three-dimensional rotation $\mathbf{R} \in \mathbf{SO}(3)$ specified by two-dimensional user input from various controllers, e.g., a joystick, a trackball, or a mouse (e.g., [5–7, 22, 40–41]). These algorithms are typically based on a dynamic mapping of the position of the controller to a unit vector in $\mathbb{R}^3$ (we describe below the procedure for the case of a mouse used as a virtual trackball). Such a mapping converts the initial position of the controller to a unit vector $\mathbf{u}_I \in \mathbb{R}^3$, and when that position is moved, the final position is converted to a second unit vector $\mathbf{u}_F \in \mathbb{R}^3$. For the object on the screen to follow the mouse naturally, as if it were being dragged, we must perform a rotation ρ on it satisfying $\mathbf{u}_F := \rho\mathbf{u}_I$. But there are infinitely many such rotations, since we could also perform any rotation about $\mathbf{u}_I$, followed by ρ, followed by any rotation about $\mathbf{u}_F$, and still have $\mathbf{u}_I$ map to $\mathbf{u}_F$. However, if $\mathbf{u}_F \neq -\mathbf{u}_I$, there is a unique rotation that takes $\mathbf{u}_I$ to $\mathbf{u}_F$ and that acts as the identity on the orthogonal complement of $V := \text{span}\{\mathbf{u}_I, \mathbf{u}_F\}$. It is this rotation that is conventionally specified by giving a unit vector and its image, and we call it the *transvection* ([4, 21, 49]) taking $\mathbf{u}_I$ to $\mathbf{u}_F$, denoted $\mathbf{T}_{(\mathbf{u}_I,\mathbf{u}_F)}$. Note that, by orthogonality, $\mathbf{T}_{(\mathbf{u}_I,\mathbf{u}_F)}$ preserves V.

In the computer graphics literature, a number of different methods have been recommended for implementing $\mathbf{T}_{(\mathbf{u}_I,\mathbf{u}_F)}$, including: construction of the rotation matrix (Chen, [5–7, 28)), using axis-angle (Euler-Rodrigues) formulas ([14, 26, 31, 32, 35]), construction of a unit quaternion, Q, and then conjugating by Q or else converting Q to a matrix (Shoemake, [40–41]), and even using Euler angles. Similar needs for specifying, implementing, and composing rotations, and proposed solutions in terms of quaternions and matrices also arise in other fields, including robotics ([11, 27, 37, 38]), photogrammetry ([25]), and rocket motion control ([45]).

Key words and phrases. rotation, reflection, transvection, quaternion.

The first author was supported in part by the NSF under the award DUE-0231459.

Here we present a simpler approach that we have not seen discussed elsewhere in the computer graphics literature and that we feel has numerous advantages. Let $\mathbf{s} = \mathbf{u}_I + \mathbf{u}_F$ and $c = \frac{2}{\mathbf{s}\cdot\mathbf{s}}$. Then for any $\mathbf{v} \in \mathbb{R}^n$, if we let $\mathbf{w} = 2(\mathbf{v}\cdot\mathbf{u}_I)\mathbf{u}_I - \mathbf{v}$, we find $\mathbf{T}_{(\mathbf{u}_I,\mathbf{u}_F)}\mathbf{v} = c(\mathbf{w}\cdot\mathbf{s})\mathbf{s} - \mathbf{w}$. This requires for each $\mathbf{v}$ two dot products and scalar multiplications and essentially no overhead, while by contrast, if $\mathbf{R}$ is the matrix of $\mathbf{T}_{(\mathbf{u}_I,\mathbf{u}_F)}$, then computing $\mathbf{R}\mathbf{v}$ not only involves three dot products for each $\mathbf{v}$, but also has the extra overhead of first requiring that we construct $\mathbf{R}$ from $\mathbf{u}_I$ and $\mathbf{u}_F$. The advantage is most pronounced when rotating a single object or scene, since then (see the appendix for details) we only need to perform the inverse rotation on the few vectors that define our perspective viewing frame, rather than having to rotate the many points defining the polygons of the object itself, so the overhead of computing $\mathbf{R}$ becomes significant. (Note that the inverse rotation is obtained just by reversing the roles of $\mathbf{u}_I$ and $\mathbf{u}_F$ above.)

To understand this algorithm better, we recall the the more familiar representation of a rotation as the composition of reflection in two planes. The two steps above are reflections in lines, a less familiar concept. In general, reflection in a subspace is the linear operator that acts as the identity on that subspace and minus the identity on its orthogonal complement. By Pythagoras, any reflection is distance preserving, and orientation reversing or preserving depending on whether the orthogonal complement is odd or even dimensional. But in either case, the composition of two reflections is distance and orientation preserving, i.e., a rotation. Note that any reflection ρ is its own inverse, and is therefore symmetric, since it is also orthogonal, i.e., $\rho^T\rho = \mathbf{I}$.

If U is the subspace spanned by a unit vector $\mathbf{u} \in \mathbb{R}^3$, the operator ρ_U performing reflection in U is given explicitly by the formula $\rho_U = \rho_\mathbf{u} = 2P_\mathbf{u} - \mathbf{I}$, where $P_\mathbf{u}$ is the orthogonal projection on U (or $\mathbf{u}$), $P_\mathbf{u}\mathbf{v} = (\mathbf{v}\cdot\mathbf{u})\mathbf{u}$ and so $\rho_\mathbf{u}\mathbf{v} = 2(\mathbf{v}\cdot\mathbf{u})\mathbf{u} - \mathbf{v}$. We check that if $\mathbf{v} = \mathbf{u}$, $\rho_\mathbf{u}\mathbf{v} = 2(\mathbf{u}\cdot\mathbf{u})\mathbf{u} - \mathbf{u} = \mathbf{u}$, so $\mathbf{v}$ is fixed, and if $\mathbf{v}$ is orthogonal to $\mathbf{u}$, $\rho_\mathbf{u}\mathbf{v} = 2(\mathbf{v}\cdot\mathbf{u})\mathbf{u} - \mathbf{v} = -\mathbf{v}$, so $\mathbf{v}$ is reversed, which verifies the defining properties of the reflection in U.

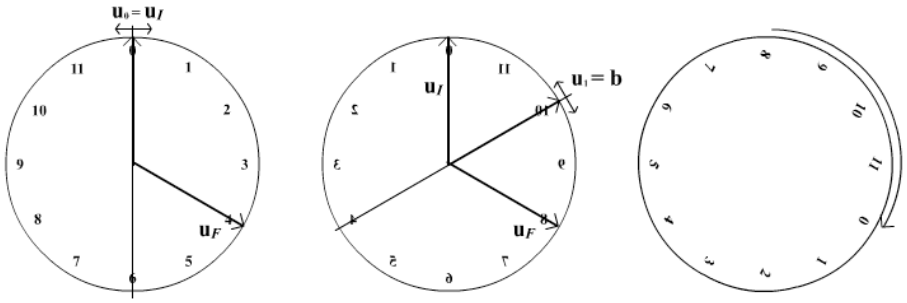

Figure 1. Implementing a rotation using a reflecting pair

The algorithm presented above is equivalent to $\mathbf{T}_{(\mathbf{u}_I,\mathbf{u}_F)} = \rho_\mathbf{b}\rho_{\mathbf{u}_I}$ where $\mathbf{b} = \frac{\mathbf{s}}{||\mathbf{s}||}$ is the unit bisector of $\mathbf{u}_I$ and $\mathbf{u}_F$, since the factor c implicitly normalizes $\mathbf{s} = \mathbf{u}_I + \mathbf{u}_F$, without taking a square root. To see this we use the fact that $\mathbf{d} = \mathbf{u}_I - \mathbf{u}_F$ is orthogonal to $\mathbf{s}$, and that $\mathbf{u}_I = \frac{1}{2}(\mathbf{s} + \mathbf{d})$, $\mathbf{u}_F = \frac{1}{2}(\mathbf{s} - \mathbf{d})$, and the properties of reflections, which confirms that $\rho_\mathbf{b}\rho_{\mathbf{u}_I}\mathbf{u}_I = \rho_\mathbf{b}\mathbf{u}_I = \mathbf{u}_F$ (see Figure 1). Also, since $\rho_{\mathbf{u}_I}$ reverses the orthogonal complement of $\mathbf{u}_I$, and $\rho_\mathbf{b}$ reverses the orthogonal complement of $\mathbf{b}$, their composition acts as the identity on the intersection of their complements, which is the same as the orthogonal complement of the span of $\mathbf{u}_I$ and $\mathbf{b}$, or $\mathbf{u}_F$.

We call an ordered pair of unit vectors $(\mathbf{u}_0, \mathbf{u}_1)$ whose reflection operators implement a transvection a *reflecting pair* for that transvection. We make the preliminary observation that $(\mathbf{u}_I, \mathbf{b})$ and $(\mathbf{b}, \mathbf{u}_F)$ are both reflecting pairs for $\mathbf{T}_{(\mathbf{u}_I, \mathbf{u}_F)}$, since by arguments parallel to those above, $\rho_{\mathbf{u}_F}\rho_{\mathbf{b}}\mathbf{u}_I = \rho_{\mathbf{u}_F}\mathbf{u}_F = \mathbf{u}_F$, and $\rho_{\mathbf{u}_F}\rho_{\mathbf{b}}$ acts as the identity on the orthogonal complement of the span of $\mathbf{u}_I$ and $\mathbf{u}_F$.

Perhaps more geometrically, we can interpret this algorithm in terms of its restriction to a sphere, and a globe of the Earth in particular. In this setting, reflection in an axis through some city N, ('reflection in N') corresponds to swapping all pairs of cities that are equidistant from N on any great circle through N. Then if we imagine N = New York to be the midpoint of B = Boston and W = Washington, DC, we accomplish the rotation of the globe taking B to W and preserving the great circle joining them by reflecting in B, then reflecting in N. The argument is that the first reflection fixes B, the second swaps B and W since they are equidistant from N, and both reflections preserve all distances and reverse all great circles through B and W, respectively. Therefore, their composition preserves the great circle containing both, and its orientation, i.e., this is the transvection taking B to W. Again, we could also reflect first in N then in W with the same result, or in any pair of cities on the same great circle whose oriented distance apart is the same. In the next section we will consider the question of the complete set of reflecting pairs corresponding to a particular transvection, and its consequences.

This algorithm could certainly also be specified in terms of commonly used Householder reflections in planes, $\mathbf{I} - 2\mathbf{u}\mathbf{u}^T = -\rho_{\mathbf{u}}$, since the two minus signs cancel, but perhaps the fact that the normal to the fixed plane, $\mathbf{u}$, is reversed instead of fixed is why this approach has been ignored thus far. It may also be that ever since Euler's Theorem on the axis of a three-dimensional rotation ([13, 29–30]), the axis-angle description of a rotation has supplanted all others. Here we observe that not only is it possible to omit the axis entirely from the computation, but also that it seems as physically natural to specify a rotation of a ball by pulling it along an equator as it is to twirl it about some axis.

Another advantage, beyond simplicity, is that nothing in the above depended upon the dimension of $\mathbb{R}^3$, and the same construction is valid in any $\mathbb{R}^n$. Indeed, the concept of transvection and the inspiration for the algorithm comes from Élie Cartan's beautiful theory of symmetric spaces ([4, 21]), and much of the above generalizes to that much broader context.

In the next section we will describe other consequences of this approach, including an algorithm for interpolating rotations, and a simple geometric realization of quaternions and derivation of their composition law.

2. Using Equivalent Reflecting Pairs for Composing and Interpolating Rotations

As we have noted above, $\rho_{\mathbf{b}}\rho_{\mathbf{u}_I}$ and $\rho_{\mathbf{u}_F}\rho_{\mathbf{b}}$ are equivalent to the unique rotation $\mathbf{T}_{(\mathbf{u}_I, \mathbf{u}_F)}$ that takes $\mathbf{u}_I$ to $\mathbf{u}_F$ and acts as the identity on the orthogonal complement of the span of $\mathbf{u}_I$ and $\mathbf{u}_F$. It is natural to expect if there are two such reflecting pairs, then there are more, and to ask what are all reflecting pairs correspond to the same rotation. Geometrically, it is intuitively clear that if $\mathbf{u}_0'$ and $\mathbf{u}_1'$ are any two unit vectors spanning the same two-dimensional subspace as $\mathbf{u}_0$ and $\mathbf{u}_1$, and the oriented angle between $\mathbf{u}_0'$ and $\mathbf{u}_1'$, and between $\mathbf{u}_0$ and $\mathbf{u}_1$ are the same, then $\rho_{\mathbf{u}_1}\rho_{\mathbf{u}_0} = \rho_{\mathbf{u}_1'}\rho_{\mathbf{u}_0'}$ (see Figure 2).

To prove this rigorously, we can use the conjugate relation between a rotation $\mathbf{R}$ and a reflection ρ: $\rho_{\mathbf{Ru}}\mathbf{v} = \mathbf{R}\rho_{\mathbf{u}}\mathbf{R}^{-1}\mathbf{v}$. Algebraically, if we define the map $\mathbf{Q}$ from reflecting pairs to $\mathbb{R}^1 \times \mathbb{R}^3$ by

$$\mathbf{Q} : (\mathbf{u}_0, \mathbf{u}_1) \to \mathbf{Q}(\mathbf{u}_0, \mathbf{u}_1) = [q, \mathbf{q}] = [\mathbf{u}_0 \cdot \mathbf{u}_1, \mathbf{u}_0 \times \mathbf{u}_1],$$

this condition is equivalent to $\mathbf{Q}(\mathbf{u}_0, \mathbf{u}_1) = \mathbf{Q}(\mathbf{u}'_0, \mathbf{u}'_1)$. We call q the scalar part of $\mathbf{Q}$ and $\mathbf{q}$ the vector part. It is an elementary vector identity that for any vectors $\mathbf{u}_0, \mathbf{u}_1$ in $\mathbb{R}^3$, $(\mathbf{u}_0 \cdot \mathbf{u}_1)^2 + ||\mathbf{u}_0 \times \mathbf{u}_1||^2 = ||\mathbf{u}_0||^2||\mathbf{u}_1||^2$, so we see that if we identify $\mathbb{R}^1 \times \mathbb{R}^3$ with $\mathbb{R}^4$, $\mathbf{Q}$ maps reflecting pairs of unit vectors to the unit sphere, $\mathbf{S}^3 \subseteq \mathbb{R}^4$.

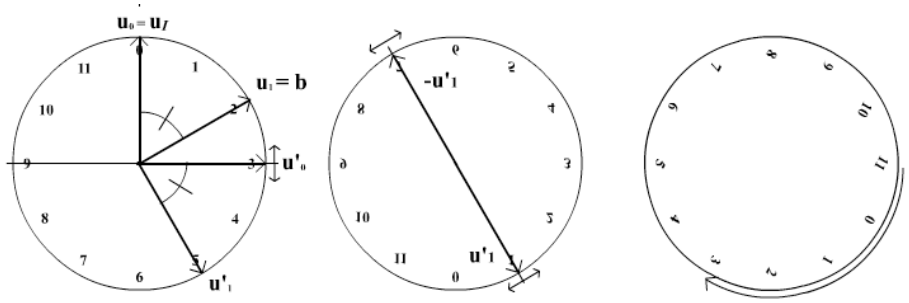

Figure 2. Implementing the same rotation using an equivalent reflecting pair

We have also noted that reflecting in $\mathbf{u}$ is the same as reflecting in $-\mathbf{u}$. Therefore, if we reverse either $\mathbf{u}_0$ or $\mathbf{u}_1$, we do not affect the resulting rotation, although both the scalar and vector parts of the resulting $\mathbf{Q}$ are reversed. If we reverse both $\mathbf{u}_0$ and $\mathbf{u}_1$, we have just performed a half-turn rotation in the $\mathbf{u}_0$–$\mathbf{u}_1$ plane, and geometry and algebra both tell us that the scalar and vector parts of the resulting $\mathbf{Q}$ are unchanged. Symbolically, we can write this as $\mathbf{Q}(-\mathbf{u}_0, \mathbf{u}_1) = \mathbf{Q}(\mathbf{u}_0, -\mathbf{u}_1) = -\mathbf{Q}(\mathbf{u}_0, \mathbf{u}_1) = -\mathbf{Q}(-\mathbf{u}_0, -\mathbf{u}_1)$, where negation is meant component-wise in $\mathbb{R}^1 \times \mathbb{R}^3 \equiv \mathbb{R}^4$, and for the rotations, $\mathbf{T}_{(-\mathbf{u}_0,\mathbf{u}_1)} = \mathbf{T}_{(\mathbf{u}_0,-\mathbf{u}_1)} = \mathbf{T}_{(\mathbf{u}_0,\mathbf{u}_1)} = \mathbf{T}_{(-\mathbf{u}_0,-\mathbf{u}_1)}$. It is not hard to show that no other reflecting pairs correspond to the same rotation, in other words, $\mathbf{T}_{(\mathbf{u}_0,\mathbf{u}_1)} = \mathbf{T}_{(\mathbf{u}'_0,\mathbf{u}'_1)}$ if and only if $\mathbf{Q}(\mathbf{u}_0, \mathbf{u}_1) = \mathbf{Q}(\mathbf{u}'_0, \mathbf{u}'_1)$ or $\mathbf{Q}(\mathbf{u}_0, \mathbf{u}_1) = -\mathbf{Q}(\mathbf{u}'_0, \mathbf{u}'_1)$. (This includes the special cases when $\mathbf{u}_0 = \pm\mathbf{u}_1$, for which $\mathbf{T}_{(\mathbf{u}_0,\mathbf{u}_1)} = \mathbf{I}$, the identity rotation, and $\mathbf{Q}(\mathbf{u}_0, \mathbf{u}_1) = \pm[1, \langle 0, 0, 0\rangle]$.)

This brings us to the second main topic of the paper. The equivalence classes of reflecting pairs under the equivalence relation $(\mathbf{u}_0, \mathbf{u}_1) = (\mathbf{u}'_0, \mathbf{u}'_1)$ if and only if $\mathbf{Q}(\mathbf{u}_0, \mathbf{u}_1) = \mathbf{Q}(\mathbf{u}'_0, \mathbf{u}'_1)$ is isomorphic to (and gives a useful geometric realization of) the unit quaternion group, $\mathbf{SU}(2)$. (See [29] for an excellent modern introduction to the relationship between the Lie groups $\mathbf{SO}(3)$ and $\mathbf{SU}(2)$, Cayley-Klein parameters, Pauli matrices and spinors.) The equivalence classes of reflecting pairs under the equivalence relation $(\mathbf{u}_0, \mathbf{u}_1) = (\mathbf{u}'_0, \mathbf{u}'_1)$ if and only if $\mathbf{Q}(\mathbf{u}_0, \mathbf{u}_1) = \pm\mathbf{Q}(\mathbf{u}'_0, \mathbf{u}'_1)$ is isomorphic to (and gives a useful geometric realization of) the three-dimensional rotation group, $\mathbf{SO}(3)$

The key to this fact is the following: just as we can define a composition law for equivalence classes of ordered pairs of points in Euclidean space (i.e., the parallelogram law) to realize geometrically composition of translations of $\mathbb{R}^n$, and and just as we can define a composition for equivalence classes of ordered pairs of unit vectors in the plane (i.e., angle addition) to realize geometrically the composition law for the plane rotation group, $\mathbf{SO}(2) = \mathbf{U}(1)$, ([3]) in a completely analogous way we can compose equivalence classes of ordered pairs of reflections to help us

understand the composition of unit quaternions and rotations in $\mathbf{SO}(3)$. If we are given any two ordered pairs of unit vectors, then since any two planes through the origin in $\mathbb{R}^3$ have non-empty intersection, we can always find two equivalent pairs with the first unit vector of the second pair, $\mathbf{u}$, equal to the second unit vector of the first pair. In Figure 3, the original second pair $(\mathbf{u}', \mathbf{u}_1')$ is equivalent to $(\mathbf{u}, \mathbf{u}_1)$, and the original first pair, $(\mathbf{u}_0', \mathbf{u}'')$, not shown, is equivalent to $(\mathbf{u}_0, \mathbf{u})$. Since the composition of the original rotations, $\rho_{\mathbf{u}_1'}\rho_{\mathbf{u}'}\rho_{\mathbf{u}''}\rho_{\mathbf{u}_0'}$ is equivalent to $\rho_{\mathbf{u}_1}\rho_{\mathbf{u}}\rho_{\mathbf{u}}\rho_{\mathbf{u}_0}$ and because the composition of any reflection with itself is the identity, $\rho_{\mathbf{u}}^2 = \mathbf{I}$, the composition of the original rotations is equivalent to $\rho_{\mathbf{u}_1}\rho_{\mathbf{u}_0}$, i.e., the corresponding rotation is just the composition of reflection in the first vector of the first pair followed by reflection in the second vector of the second pair. So we may truly think of this as 'vector addition' or 'angle addition' for three-dimensional rotations, as long as we are careful to remember that the ordered pairs of vectors give the invariant subspace and *half-angle* of the rotation, rather than being in an 'initial-final' relation as in the other settings. Given the invariant plane and half-angle of two rotations, this gives a coordinate-free description of the invariant plane and half-angle of their composition that we can perform with our fingers just as we do with displacements or plane rotations. The construction is in some sense dual to the Euler–Schwarz spherical triangle construction ([1, 9, 34, 36]).

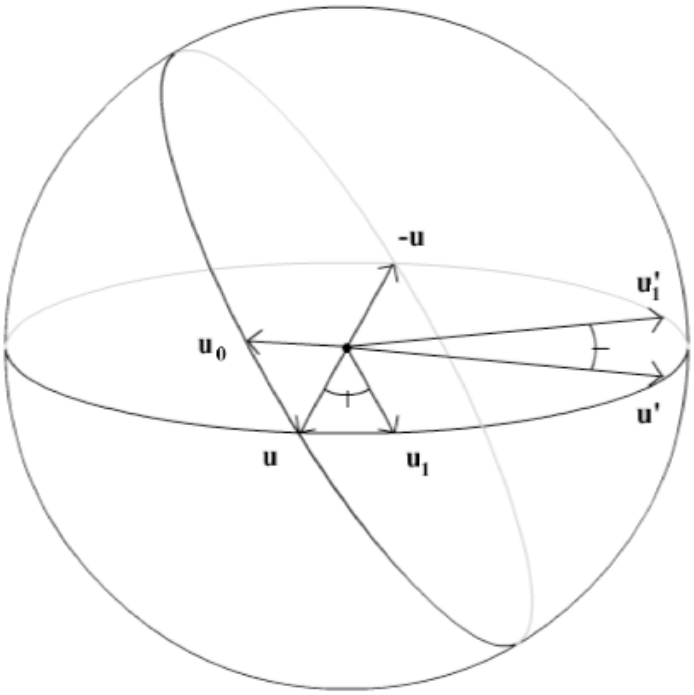

Figure 3. Composition of reflecting pairs

From an algebraic perspective, this construction shows that equivalence classes of reflecting pairs are closed under composition, that the class $(\mathbf{u}, \mathbf{u})$ for any unit vector $\mathbf{u}$ is its identity, and the class of $(\mathbf{u}_0, \mathbf{u}_1)$ is the inverse of the class of $(\mathbf{u}_1, \mathbf{u}_0)$, in other words, they form a group.

It is natural and appealing to take the geometric composition laws we have described as the starting point to defining the translation group and plane rotation groups, and derive from them the corresponding algebraic rule for composing equivalence class invariants ([3]). For translations, we find from congruent triangles that we add the invariants, the displacements in each coordinate, component-wise. The invariants for plane rotations described as equivalence classes of pairs of unit vectors in the plane are formally similar to those we use for unit quaternions: $(\mathbf{u}_0, \mathbf{u}_1) = (\mathbf{u}'_0, \mathbf{u}'_1)$ if and only if $\mathbf{Q}(\mathbf{u}_0, \mathbf{u}_1) = \mathbf{Q}(\mathbf{u}'_0, \mathbf{u}'_1)$, as long as we interpret these vectors as unit vectors in $\mathbb{R}^2$ by identifying the vector part with its non-vanishing component. Then $\mathbf{Q}$ reduces to a map taking $(\mathbf{u}_0, \mathbf{u}_1)$ to $\langle \cos\theta, \sin\theta \rangle$, where θ is the oriented angle between $\mathbf{u}_0$ and $\mathbf{u}_1$. Using congruent and similar triangles, we can derive the composition law for the invariants of plane rotation from the geometry, just as Wessel ([47, 48]) did in 1797. If the invariants of the plane rotation taking $\mathbf{u}_0$ to $\mathbf{u}$ are $\langle \mathbf{u}_0 \cdot \mathbf{u}, \mathbf{u}_0 \times \mathbf{u} \rangle = \langle \cos\theta_1, \sin\theta_1 \rangle$ and the invariants of the rotation taking $\mathbf{u}$ to $\mathbf{u}_1$ are $\langle \mathbf{u} \cdot \mathbf{u}_1, \mathbf{u} \times \mathbf{u}_1 \rangle = \langle \cos\theta_2, \sin\theta_2 \rangle$, then the invariants of the plane rotation taking $\mathbf{u}_0$ to $\mathbf{u}_1$ are $\langle \cos\theta_1 \cos\theta_2 - \sin\theta_1 \sin\theta_2, \cos\theta_1 \sin\theta_2 + \cos\theta_2 \sin\theta_1 \rangle$, i.e.,

$$\mathbf{u}_0 \cdot \mathbf{u}_1 = (\mathbf{u}_0 \cdot \mathbf{u})(\mathbf{u} \cdot \mathbf{u}_1) - (\mathbf{u}_0 \times \mathbf{u})(\mathbf{u} \times \mathbf{u}_1),$$

$$\mathbf{u}_0 \times \mathbf{u}_1 = (\mathbf{u}_0 \cdot \mathbf{u})(\mathbf{u} \times \mathbf{u}_1) + (\mathbf{u} \cdot \mathbf{u}_1)(\mathbf{u}_0 \times \mathbf{u}).$$

We now ask if we can perform the corresponding analysis for reflecting pairs in three-dimensions? The answer is yes, and the resulting composition law contains the plane rotation version as a special case, since the non-commutative three-dimensional rotation group contains many commutative plane rotation subgroups. If we denote by $\mathbf{u}$ the common intermediate unit vector in our above construction for composing reflecting pairs, and call the first vector of the first pair $\mathbf{u}_0$, and the second vector of the second pair $\mathbf{u}_1$, what we are looking for is an expression for the invariants of the composed pair, $\mathbf{Q}(\mathbf{u}_0, \mathbf{u}_1)$ in terms of the invariants of the pairs being composed, $\mathbf{Q}(\mathbf{u}_0, \mathbf{u})$ and $\mathbf{Q}(\mathbf{u}, \mathbf{u}_1)$ (Figure 3). Using Lagrange's formula and familiar vector identities, it is not difficult to derive two new vector identities: For any three unit vectors $\mathbf{u}_0, \mathbf{u}, \mathbf{u}_1 \in \mathbb{R}^3$,

$$\mathbf{u}_0 \cdot \mathbf{u}_1 = (\mathbf{u}_0 \cdot \mathbf{u})(\mathbf{u} \cdot \mathbf{u}_1) - (\mathbf{u}_0 \times \mathbf{u}) \cdot (\mathbf{u} \times \mathbf{u}_1),$$

$$\mathbf{u}_0 \times \mathbf{u}_1 = (\mathbf{u}_0 \cdot \mathbf{u})(\mathbf{u} \times \mathbf{u}_1) + (\mathbf{u} \cdot \mathbf{u}_1)(\mathbf{u}_0 \times \mathbf{u}) + (\mathbf{u} \times \mathbf{u}_1) \times (\mathbf{u}_0 \times \mathbf{u}).$$

Once derived or if motivated by other considerations, these formulas may also be easily confirmed using an adapted basis in which $\mathbf{u} = \langle 1, 0, 0 \rangle$, $\mathbf{u}_0 = \langle \cos\theta, -\sin\theta, 0 \rangle$ with $\cos\theta = \mathbf{u}_0 \cdot \mathbf{u}$, and calculating the remaining components, by differentiating to show the right hand sides are constant as $\mathbf{u} = \mathbf{u}(t)$ is rotated from $u(0) = \mathbf{u}_0$ where the formulas hold identically, or checking them on standard basis vectors and appealing to multilinearity. In terms of the map, $\mathbf{Q}$, if we denote the scalar and vector parts of $\mathbf{Q}(\mathbf{u}_0, \mathbf{u}) = [q_0, \mathbf{q}_0]$, and of $\mathbf{Q}(\mathbf{u}, \mathbf{u}_1) = [q_1, \mathbf{q}_1]$, the formula becomes

$$\mathbf{Q}(\mathbf{u}_0, \mathbf{u}_1) = [q_0 q_1 - \mathbf{q}_0 \cdot \mathbf{q}_1, q_0 \mathbf{q}_1 + q_1 \mathbf{q}_0 + \mathbf{q}_1 \times \mathbf{q}_0].$$

This is quaternion multiplication ([15,16]), also known as the Rodrigues composition formula ([1, 2, 35, 45]), motivated and derived here from the geometry of rotation using reflecting pairs. If $\mathbf{u}_0, \mathbf{u}$, and $\mathbf{u}_1$ are co-planar, so $\mathbf{q}_1 \times \mathbf{q}_0 = \mathbf{0}$, this reduces to the complex multiplication formula above, as claimed.

If we keep the plus and minus classes distinct, equivalence classes of reflecting pairs is isomorphic to the unit quaternions, $\mathbf{SU}(2)$, and if we identify them, the classes we obtain are isomorphic to the the three-dimensional rotation group, $\mathbf{SO}(3)$.

As an example, if we take $\mathbf{u}_0 = \langle 1, 0, 0\rangle$, $\mathbf{u} = \langle 0, 0, -1\rangle$, and $\mathbf{u}_1 = \langle 0, 1, 0\rangle$, $\mathbf{Q}(\mathbf{u}_0, \mathbf{u}) = [q_0, \mathbf{q}_0] = [0, \langle 0, 1, 0\rangle]$, $\mathbf{Q}(\mathbf{u}, \mathbf{u}_1) = [q_1, \mathbf{q}_1] = [0, \langle 1, 0, 0\rangle]$, and either directly or using the composition rule, $\mathbf{Q}(\mathbf{u}_0, \mathbf{u}_1) = [0, \langle 0, 0, 1\rangle]$, which in Hamilton's notation would be written '$\mathbf{ij} = \mathbf{k}$'. Note that when we interpret $\mathbf{ij}$, the half-turn rotation about the y-axis corresponding to $\mathbf{j}$, reflection in $\mathbf{u}_0$ then in $\mathbf{u}$, is performed first, in the order of function composition. This is followed by the half-turn rotation about the x-axis corresponding to $\mathbf{i}$, reflection in $\mathbf{u}$ then in $\mathbf{u}_1$, to obtain the half-turn rotation about the z-axis corresponding to $\mathbf{k}$, reflection in $\mathbf{u}_0$ then in $\mathbf{u}_1$. (See <`http://www.math.utah.edu/~palais/transvection.html`> for interactive visualizations.) Note the different geometric interpretations of the quaternion $\mathbf{i}$ as a half-turn rotation about the $x-$axis and the complex number i, which we think of geometrically as a quarter-turn rotation in the x–y plane! Consequently, since $\mathbf{i}^2 = -1$ in both cases, the complex number -1 corresponds to a half-turn rotation in the x–y plane, while the unit quaternion $-1 = [-1, \langle 0, 0, 0\rangle]$ corresponds to reflection in two opposite unit vectors, i.e, the identity.

This realization makes it easy to 'see' unit quaternions and hence $\mathbf{S}^3$ no less concretely than conventional vectors in $\mathbb{R}^3$. In other words, just as we think of the vector $\mathbf{v} = \langle v_1, v_2, v_3\rangle \in \mathbb{R}^3$ as any pair of points $((t_1, t_2, t_3), (h_1, h_2, h_3))$ satisfying $h_j - t_j = v_j,\ j = 1, 2, 3$, and the rotation in the plane by an angle θ, $z = e^{i\theta} = (\cos\theta, \sin\theta)$ as any pair of unit vectors in $\mathbb{R}^2$, $(\mathbf{u} = \langle u_1, u_2\rangle, \mathbf{v} = \langle v_1, v_2\rangle)$ satisfying $(\mathbf{u}\cdot\mathbf{v}, \mathbf{u}\times\mathbf{v}) = (u_1v_1 + u_2v_2, u_1v_2 - u_2v_1) = (\cos\theta, \sin\theta)$, we may think of the unit quaternion $\mathbf{Q} = [q, \mathbf{q}]$ *as* any pair of unit vectors in $\mathbb{R}^3$, $(\mathbf{u}_0, \mathbf{u}_1)$ satisfying $[\mathbf{u}\cdot\mathbf{v}, \mathbf{u}\times\mathbf{v}] = [q, \mathbf{q}]$.

In addition to composition, many other standard quaternion constructions can be understood in this framework. Negation, as we have seen, corresponds to reversing either unit vector of an ordered pair, and gives the well-known double-covering of the three-dimensional rotation group by the unit quaternions a simple visual interpretation. Since we can invert two reflections by performing them in the reverse order, we obtain the inverse of a geometric quaternion by reversing the order of its pair. Since the scalar product is symmetric with respect to reversing order, but the vector product is anti-symmetric, this correctly corresponds to conjugation of the associated unit quaternion in its standard representation in $\mathbf{S}^3 \subset \mathbb{R}^4$. In Figure 3, $(\mathbf{u}, \mathbf{u})$ is one representative of the identiy, $+1$, $(\mathbf{u}, -\mathbf{u})$ is a representative of -1. Since $\mathbf{Q}_1 = \mathbf{Q}(\mathbf{u}, \mathbf{u}_1) = [q_1, \mathbf{q}_1]$, we have $-\mathbf{Q}_1 = \mathbf{Q}(-\mathbf{u}, \mathbf{u}_1) = [-q_1, -\mathbf{q}_1] = -[q_1, \mathbf{q}_1]$, is the opposite quaternion representing the same rotation, and $\mathbf{Q}_1^* = \mathbf{Q}(\mathbf{u}_1, \mathbf{u}) = [q_1, -\mathbf{q}_1]$ is the inverse or conjugate quaternion, which along with $-\mathbf{Q}_1^*$ corresponds to the inverse rotation.

We emphasize that in order for this picture to correspond to the correct rotation, and to take advantage of its efficiency, we must interpret any such $(\mathbf{u}_0, \mathbf{u}_1)$ as a reflecting pair, so if this rotation $\mathbf{R} = \rho_{\mathbf{u}_1}\rho_{\mathbf{u}_0}$ has axis $\mathbf{n}$ and angle θ, we see that $[\mathbf{u}\cdot\mathbf{v}, \mathbf{u}\times\mathbf{v}] = [q, \mathbf{q}] = [\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\mathbf{n}]$. In the case of vectors in $\mathbb{R}^n$ or plane rotations, there are distinguished representatives based at the identity, $(0, 0, 0)$, or $(1, 0)$, respectively, while in the case of spatial rotations there is not.

By Euler's theorem on the axis of a rotation ([13, 29, 30]) *every* rotation is a transvection defined by some pair of unit vectors, so by identifying the two classes $\mathbf{Q}$ and $-\mathbf{Q}$ this construction can be modified to give a realization of the three-dimensional rotation group $\mathbf{SO}(3)$ as equivalence classes of ordered pairs of lines through the origin in $\mathbb{R}^3$. Our construction can also be extended to the full quaternion algebra by eliminating the restriction that $\mathbf{u}_0$ and $\mathbf{u}_1$ are unit vectors. Since both scalar and vector product are bilinear, the operations of addition or scalar multiplication are realized by finding representatives of two pairs sharing a common first (or second) vectors, and adding (or scalar multiplying) the other vectors.

This same 'shared first vector' construction also arises in an efficient new algorithm for interpolating rotations. The standard method of interpolating rotations $\mathbf{R}_0$ and $\mathbf{R}_1$ is to interpolate corresponding quaternions $\mathbf{Q}_0$ and $\mathbf{Q}_1$ along the shorter arc of the great circle joining them on $\mathbf{S}^3$. For this reason the process is called 'SLERP' ([43, 44]), for spherical linear interpolation. (We choose $\mathbf{Q}_1$ to be the closer to $\mathbf{Q}_0$ of the two opposite quaternions corresponding to $\mathbf{R}_1$.) If we set $\theta = \arccos(\mathbf{Q}_0 \cdot \mathbf{Q}_1)$ the desired path $\mathbf{Q}(\lambda)$, $0 \leq \lambda \leq 1$ joining $\mathbf{Q}_0 = \mathbf{Q}(0)$ and $\mathbf{Q}_1 = \mathbf{Q}(1)$ is given by $\mathbf{Q}(\lambda) = a_0(\lambda)\mathbf{Q}_0 + a_1(\lambda)\mathbf{Q}_1$, where

$$a_0(\lambda) = \frac{\sin((1-\lambda)\theta)}{\sin\theta},$$

and

$$a_1(\lambda) = \frac{\sin(\lambda\theta)}{\sin\theta}.$$

Note that we can replace $\sin(\arccos(\mathbf{Q}_0 \cdot \mathbf{Q}_1)$ by $\sqrt{1-(\mathbf{Q}_0 \cdot \mathbf{Q}_1)^2}$.

Just as in the composition and linear combination constructions, we can always find unit vectors $\mathbf{u}, \mathbf{u}_0$, and $\mathbf{u}_1$ such that $\mathbf{Q}(\mathbf{u}, \mathbf{u}_0) = \mathbf{Q}_0$ and $\mathbf{Q}(\mathbf{u}, \mathbf{u}_1) = \mathbf{Q}_1$. Then if we perform spherical interpolation from $\mathbf{u}_0$ to $\mathbf{u}_1$ on $\mathbf{S}^2 \in \mathbb{R}^3$ by the analogous formula $\mathbf{u}(\lambda) = a_0(\lambda)\mathbf{u}_0 + a_1(\lambda)\mathbf{u}_1$, bilinearity guarantees that the resulting pairs correspond to the standard interpolating path: $\mathbf{Q}(\mathbf{u}, \mathbf{u}(\lambda)) = \mathbf{Q}(\lambda)$. Using Lagrange's identity, we can also show that $\theta = \arccos(\mathbf{u}_0 \cdot \mathbf{u}_1)$, i.e. $\mathbf{u}_0 \cdot \mathbf{u}_1 = \mathbf{Q}_0 \cdot \mathbf{Q}_1$, and again, we can replace $\sin(\arccos(\mathbf{u}_0 \cdot \mathbf{u}_1)$ by $\sqrt{1-(\mathbf{u}_0 \cdot \mathbf{u}_1)^2}$. In Figure 3, $\mathbf{u}(\lambda)$ would be the arc of the great circle joining $\mathbf{u}_0$ to $\mathbf{u}_1$ (not shown).

The advantage of this method is not simply that we have reduced from four to three the number of components used when we interpolate and form dot products. That is true, but not particularly significant. The greater saving is that to rotate a vector $\mathbf{v}$ by each interpolated quaternion $\mathbf{Q}(\lambda)$ in its standard representation in $\mathbb{R}^4$, we must convert $\mathbf{Q}(\lambda)$ to a matrix $\mathbf{R} = \mathbf{R}(\mathbf{Q}(\lambda))$using the Euler–Rodrigues quaternion-to-matrix formula, then multiply $\mathbf{Rv}$, or conjugate quaternion multiply $\mathbf{Q}(\lambda)[0, \mathbf{v}]\mathbf{Q}^*(\lambda) = [0, \mathbf{Rv}]$, which requires even more computation. In contrast, the reflecting pair representation of the interpolating path is ready to act as rotation by composition of two reflections with no overhead. We do not even need to compute (and implicitly normalize) a bisector as we did in the trackball application. For any $\mathbf{v} \in \mathbb{R}^3$, we simply form $\mathbf{w} = 2(\mathbf{v} \cdot \mathbf{u})\mathbf{u} - \mathbf{v}$, and obtain $\mathbf{Rv} = 2(\mathbf{w} \cdot \mathbf{u}(\lambda))\mathbf{u}(\lambda) - \mathbf{w}$. As in the first section, the advantage over converting to a matrix is greatest when there are few vectors to rotate, i.e., when we only need to rotate the viewing frame for a scene.

Finally, the reflecting pair realization of $\mathbf{SU}(2)$ makes it easy to see and understand the fact that the path in the space of three-dimensional rotations from the identity to itself consisting of two full rotations about an axis can be continuously deformed to the constant path at the identity. This fact is the essence of several well-known topological demonstrations that go by various names such as Dirac's Belt (or String) Trick, Feynman's Philippine Wineglass (or Plate) trick, ([12, 19, 20, 33]) and the Orientation Entanglement property described in the treatment of spinors in a classic general relativity book [31]. It is also the basis of the fact that while the homotopy group of loops of plane rotations is isomorphic (via winding number) to $\mathbf{Z}$, on the other hand, since the loop consisting of two full rotations about an axis is homotopic in $\mathbf{SO}(3)$ to a constant loop, the group of homotopy classes of loops of spatial rotations is isomorphic to $\mathbf{Z}/(2)$.

A path consisting of two full turns about an axis $\mathbf{u}$ can be described by a set of reflecting pairs in which the first unit vector $\mathbf{u}_0$ is fixed on the equator of the axis $\mathbf{u}$ on the unit sphere in $\mathbb{R}^3$, and the second $\mathbf{u}_1(t)$ follows a path circling the same equator once from $\mathbf{u}_0$ back to itself. To see this better, note that when the vectors $\mathbf{u}_1(t) = \mathbf{u}_0$, $\mathbf{Q}(\mathbf{u}_0, \mathbf{u}_1(t)) = [1, \langle 0, 0, 0\rangle]$, and $\rho_{\mathbf{u}_1(t)}\rho_{\mathbf{u}_0} = \mathbf{I}$. When they are a quarter turn apart, $\mathbf{Q}(\mathbf{u}_0, \mathbf{u}_1(t)) = [0, \mathbf{u}]$, is a pure vector quaternion, corresponding to a half-turn rotation and represented by a symmetric matrix. When we have gone halfway around the equator, $\mathbf{u}_1(t) = -\mathbf{u}_0$, $\mathbf{Q}(\mathbf{u}_0, \mathbf{u}_1(t)) = [-1, \langle 0, 0, 0\rangle]$ and again $\rho_{\mathbf{u}_1(t)}\rho_{\mathbf{u}_0} = \mathbf{I}$, so we have already completed one full rotation about $\mathbf{u}$. As $\mathbf{u}_1(t)$ completes its revolution around the equator back to $\mathbf{u}_0$, all we have to do to deform this path $\rho_{\mathbf{u}_1(t)}\rho_{\mathbf{u}_0}$ to the constant path at the identity is to deform the path $\mathbf{u}_1(t)$ on $\mathbf{S}^2$ to the constant path $\mathbf{u}_0$. In other words, the reflecting pairs picture reduces the simple-connectedness of $\mathbf{S}^3$ to the simple-connectedness of $\mathbf{S}^2$. Similarly, the fact that one full turn about $\mathbf{u}$ is *not* homotopic to the identity can be understood by letting $\mathbf{u}_0$ be as above, but $\mathbf{u}_1(t)$ only complete a half-turn about the equator from $\mathbf{u}_0$ to $-\mathbf{u}_0$. The corresponding path in the rotation group completes one full turn about $\mathbf{u}$ from the identity back to itself. That this cannot be deformed to the constant path at the identity is apparent from the fact that the endpoint of $\mathbf{u}_1(t)$ cannot be deformed away from $-\mathbf{u}_0$ continuously without the corresponding terminal rotation leaving the identity. In other words, the reflecting pairs picture reduces the fact that $\mathbf{SO}(3)$ is not simply-connected to the fact that $\mathbf{S}^0 = \{+1, -1\}$ is not connected. This formulation also enables us to see that the demonstration of the orientation entanglement property is simply the union of two belt trick homotopies, one reflected in the other. (It is interesting to note that if belt tricks are concatenated periodically instead of reflected, the trick can no longer be performed!)

Geometric descriptions of composition of rotations in terms of their axes and angles, foreshadowing the reflecting pair equivalence class construction, may be found in papers as early as that of Euler ([14]) and Schwarz ([37]) and as recent as Hestenes ([22, 23]), Conway-Smith ([9]), and Penrose ([35]). See Stillwell's excellent article ([46]) as well as Altmann ([1, 2]) for discussions of some historical aspects. Original formulations of the algebraic composition formula (i.e., quaternion multiplication) from different points of view appear in the works of Rodrigues ([36]), Hamilton ([17]), and Cayley ([5]). There are many other nice treatments of quaternions from various points of view, including algebra analysis, geometry and visualization, and many applications in visualization, and many areas of physics and engineering ([10, 12, 18, 19, 20, 25, 26, 28, 29, 42, 43, 46]).

3. Appendix: Perspective Rendering and Implementation of a Virtual Trackball

Objects and scenes are rendered using elementary transformations that map a data point $\mathbf{dp} \in \mathbb{R}^3$ to a pixel on the screen. These mappings depend on the viewing data, i.e., the viewpoint, $\mathbf{vp} \in \mathbb{R}^3$, from which the point is viewed, and the viewframe, an orthonormal basis including the normalized viewing direction, $\mathbf{vd} = -\frac{\mathbf{vp}}{||\mathbf{vp}||}$ specifying the orientation of the viewer at the viewpoint. In our model of a scene or single object, instead of moving the data points, it is equivalent to performing the inverse motion on the viewing frame.

To render a data point, we first map it to its image $\mathbf{ip}$ along the line from the viewpoint to $\mathbf{dp}$ that lies in the image plane orthogonal to the view direction (and passing through the origin.) Thus $\mathbf{ip} = \mathbf{vp} + t(\mathbf{dp} - \mathbf{vp})$, for the value of t satisfying $\mathbf{ip} \cdot \mathbf{vp} = 0$, which leads to $t = -\frac{||\mathbf{vp}||^2}{(\mathbf{dp}-\mathbf{vp})\cdot\mathbf{vp}}$ (see Figure 4).

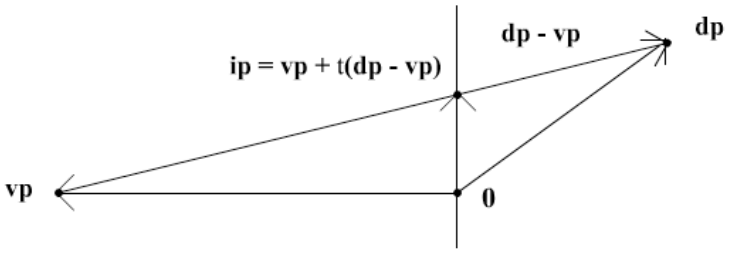

Figure 4. Perspective projection of a data point on the image plane

Note that the numerator is constant as long as the distance from the viewpoint to the origin is fixed and need not be recomputed. Because this map from $\mathbb{R}^3$ to the image plane does not depend on any particular basis for the image plane, we may use it to initialize the remaining vectors in the viewframe, $\mathbf{i}_r, \mathbf{i}_u$. These vectors form an orthonormal basis for the image plane, corresponding to the horizontal right, and vertical up directions, in the image plane and on the screen. For example, we may let the initial $\mathbf{i}_u$ be the projection of $\langle 0, 0, 1\rangle$ into the image plane as determined above, and $\mathbf{i}_r = \mathbf{vd} \times \mathbf{i}_u$.

Our next transformation simply computes the components of $\mathbf{ip}$ in this basis, $c_j = \mathbf{ip} \cdot \mathbf{i}_j,\ j = 1, 2$. Finally, we assign a pixel to the vector $\langle c_1, c_2\rangle \in \mathbb{R}^2$ according to an affine transformation taking the origin $\langle 0, 0\rangle$ to the pixel $\langle pc_1, pc_2\rangle$ at the center of the viewing window,and scaled according to the correspondence above: $\langle px_1, px_2\rangle = \langle pc_1 + s\ c_1, pc_2 - s\ c_2\rangle$. Here, s is a scaling factor and the signs are different because pixels are indexed right and downward instead of the right and up convention in Cartesian coordinates. The rendering map $\mathbf{dp} \to \langle px_1, px_2\rangle$ obtained by composing the three transformations described above provides the basic information required by the standard graphics routines used to draw paths and polygons.

When we implement a 'virtual trackball' with our mouse, we imagine it is located at the head of a unit vector $\mathbf{um} \in \mathbb{R}^3$ on the front surface of a transparent unit sphere centered at the origin, in which our object or scene is embedded. As we move the mouse, it incrementally drags this sphere, and our scene about its center at the origin. The pixel at which the mouse is located on screen corresponds to the vector in the image plane **im** that is collinear with the viewpoint **vp** and the unit mouse vector **um**. The rendering map is not invertible, however we invert the latter two transformations taking a vector in the image plane to pixel coordinates to identify **im**: $\mathbf{im} = c_1\mathbf{i}_r + c_2\mathbf{i}_u$, where $\langle c_1, c_2\rangle = \frac{1}{s}\langle px_1 - pc_1, pc_2 - px_2\rangle$. Now we can find **um** by identifying the point on the line joining **vp** and **um** whose norm is 1, (see Figure 5).

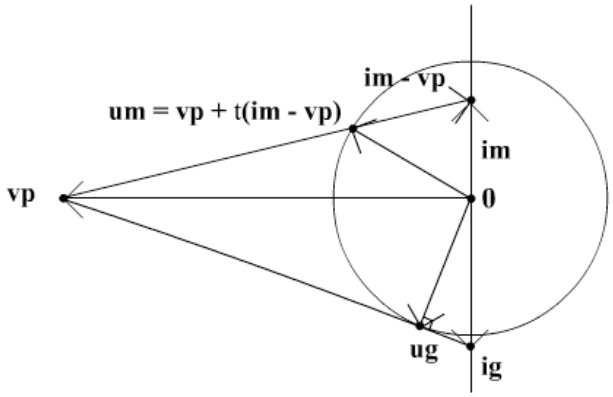

Figure 5. Perspective projection of the mouse on the virtual trackball

Thus $\mathbf{um} = \mathbf{vp} + t(\mathbf{im} - \mathbf{vp})$, for the value of t satisfying $\mathbf{um} \cdot \mathbf{um} = 1$, or $q(t) = at^2 + bt + c = 0$, where $a = ||\mathbf{im} - \mathbf{vp}||^2$, $b = 2(\mathbf{vp} \cdot (\mathbf{im} - \mathbf{vp}))$, and $c = ||\mathbf{vp}||^2 - 1$. Note again that c does not change as long as the distance from **vp** to the origin is invariant, i.e., that it is only rotated about the origin.

We must consider the possibility of no solution or multiple solutions of the quadratic equation above. Geometrically we see that the cone with vertex at the viewpoint and tangent to the unit sphere, is tangent along a latitude circle about the **vd** axis. If **um** is on this grazing horizon, the line joining the viewpoint and **um** intersects the unit sphere exactly once, between the image plane and the viewpoint. This line also intersects the image plane on a circle of radius rg, which may be found using similar triangles and Pythagoras, or by setting the discriminant of q equal to zero. Both approaches give $rg^2 = \frac{||\mathbf{vp}||^2}{||\mathbf{vp}||^2 - 1} > 1$. If $||\mathbf{im}||^2 > rg^2$, the line joining the viewpoint and **im** does not intersect the unit sphere, and we set $\mathbf{um} = \frac{\mathbf{im}}{||\mathbf{im}||}$, the unit vector in the direction of **im** in the image plane, which we cannot truly 'see' as it is 'behind' the visible horizon of the sphere. If $||\mathbf{im}||^2 < rg^2$, $q(t) = 0$ has two solutions, corresponding to the two intersections of the line containing **vp** and such an **im** with the unit sphere using the form that avoids the potential instability of the conventional solution formula. As the mouse crosses the grazing circle back and forth, there is a discontinuity in the mapping to the unit sphere that can cause the virtual trackball to jump unnaturally. Since we initialize and terminate each drag of the ball with a mouse-down and mouse-up event, respectively, this unwanted behavior can be easily prevented by imitating a fictitious mouse-up and mouse-down event each time we cross the horizon.

Once a mouse-down event initializes our first unit vector $\mathbf{u}_I = \mathbf{um}_0$, then until the next mouse-up event, in each cycle we acquire the new mouse pixels and obtain the second unit vector $\mathbf{u}_F = \mathbf{um}_1$ specifying the rotation we wish to perform. After performing the inverse rotation on the viewing frame using the simple transvection algorithm described in the paper, we update our viewpoint using the new view direction and the known view distance from the viewpoint to the origin. The final step we must perform is to transform our previous second mouse unit vector $\mathbf{u}_F$ to become the next initial unit vector. We cannot just let the new $\mathbf{u}_I$ be the old $\mathbf{u}_F$ since, due to the rotation of the viewing frame, that unit vector no longer corresponds to the point on the screen from which it was obtained, and if we did so and left the mouse still, the object would rotate uncontrollably. Fortuitously, we can check that the unit vector that now corresponds to this pixel is just the old $\mathbf{u}_I$! In other words, we only have to compute $\mathbf{u}_I$ once per drag, and during each frame of our interactive animation, we only need to compute one square root to obtain the new $\mathbf{u}_F$. If we do this, and leave the mouse down but static, our object does not move. (We could also rotate or not depending upon a mouse-moved event.) When a mouse-up event occurs, we have the option of stopping the motion, or if we leave $\mathbf{u}_F$ fixed, the latest rotation will be repeated indefinitely, simulating inertial motion of our object.

A working example, sufficient for many purposes, of the implementation described above, may be found along with codes and figures, at:
<http://www.math.utah.edu/∼palais/transvection.html>. In it, we define a function that reflects a vector v in the vector u and puts the result in w:

```
function reflect(v,u,w,normfac) {
c = normfac*dot3(v,u);
for (i=0; i < 3; i++) w[i] = c*u[i]-v[i];
}
```

(where `dot3` implements the dot product in $\mathbb{R}^3$) and use it to define a function that does two reflections to perform a rotation:

```
function rotate(v,b,normfac,u_I){
reflect(v,b,w,normfac);
reflect(w,u_I,v,2);
}
```

Then, literally, every time we move the mouse and acquire a new unit vector $\mathbf{u}_F$, the only lines of code we need to execute in order to perform the specified rotation on the viewframe and viewpoint are:

```
function transvectviewframe(){
for (i=0; i < 3; i++) s[i] = u_I[i]+u_F[i];
normfac = 2.0/dot3(s,s);
rotate(vd,s,normfac,u_I);
rotate(iu,s,normfac,u_I);
cross(vd, iu, ir);
for (i=0; i < 3; i++) vp[i] = -viewdistance*vd[i];
}
```

where `cross` implements the cross product in $\mathbb{R}^3$. If there is a simpler, more efficient code to perform this rotation, given $\mathbf{u}_I$ and $\mathbf{u}_F$, the authors would be interested in learning of it. An analysis of operation counts using various methods can be found at: <http://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation>.

Two University of Utah students have carried out implementations of our transvection trackball algorithm; one, Chuanbin Peng, in C++/OpenGL/FLTK and compiled under Microsoft Visual Studio (UNIX/Linux environments), the other, Eric Brown, for the Nintendo DS video games platform at Sensory Sweep Studios. The authors have implemented it in Pascal, Java, and NI LabVIEW.

We have also implemented numerous utility algorithms documented at the link above. Most notably, conversion from a unit quaternion $\mathbf{Q} = [q, \mathbf{q}]$ to a reflecting pair $(\mathbf{u}_0, \mathbf{u}_1)$ may be thought of in purely linear algebraic terms as the problem of finding unit vectors having prescribed scalar q and vector products. A useful result is that if $\mathbf{u}$ is any unit vector orthogonal to $\mathbf{q}$, then $(\mathbf{u}, q\mathbf{u} + \mathbf{q} \times \mathbf{u})$ and $(q\mathbf{u} - \mathbf{q} \times \mathbf{u}, \mathbf{u})$ are reflecting pairs equivalent to $\mathbf{Q}$. We can check by performing the reflecting pair to quaternion map, i.e., taking their scalar and vector products and applying vector identities. To convert a rotation matrix ρ to a reflecting pair, we note that its axis $\mathbf{v}$ is defined by $(\rho - \mathbf{I})\mathbf{v} = \mathbf{0}$, so we may take $\mathbf{u}_0$ to be any non-zero column of $\rho^{\mathbf{T}} - \mathbf{I}$. Then since $\mathbf{u}_1$ should be the bisector of $\mathbf{u}_0$ and its image under ρ, $\mathbf{u}_1$ is the corresponding column of the anti-symmetric part, A, of ρ: $\frac{1}{2}((\rho^{\mathbf{T}} - \mathbf{I}) + \rho(\rho^{\mathbf{T}} - \mathbf{I})) = \frac{1}{2}(\rho^{\mathbf{T}} - \rho) = \mathbf{R}$. Taking non-generic cases when $\mathbf{R} = \mathbf{0}$ into account also leads to a new proof of Euler's theorem (32,33]).

Shoemake's Arcball ([40, 41]) implements a rather different model for mouse-controlled rotation than other virtual trackballs. Rather than performing incremental, path-dependent rotations, the rotation performed depends only on the point at which the mouse was clicked (down) and the current location of the mouse, and is therefore path-independent. Also, the current unit vector does not specify the direction to which the initial unit vector has been dragged, but the bisector of those directions. Because of this, the object is rotated twice the angle between these vectors on the virtual trackball, and a full-turn of an object can be achieved without lifting (un-clicking) the mouse. The codes for Arcball may be found on the Web. We have also implemented a version using transvection simply by replacing reflection in the bisector with reflection in the current mouse unit vector, and instead of reflecting in the previous mouse unit vector, converted to the new coordinate system, we reflect in the initial mouse unit vector, converted at each step. While the former changes on the screen but remains constant relative to the changing coordinate system, the latter is fixed on the screen, but changes relative to the changing coordinate system. It is also linked from:

<http://www.math.utah.edu/∼palais/transvection.html>

(The original Arcball also draws a great circle arc from the initial mouse point to the point where it has been rotated, and we have implemented this approximately.)

For treatments of more sophisticated aspects of graphics, including clipping, sorting and drawing order, lighting and shading, etc., see e.g., ([15,40]).

REFERENCES

[1] S. Altmann, *Rotations, Quaternions, and Double Groups.*, Dover, New York, 1986.

[2] S. L. Altmann, *Hamilton, Rodrigues and the quaternion scandal* Mathematics Magazine 62: 291-308..

[3] M. Audin, *Geometry.*, Springer, New York, 2002

[4] É. Cartan, *Lecons sur la geometrie des éspaces de Riemann*, Gauthier-Villars, Paris, 1928 p. 266.

[5] A. Cayley, *Collected Mathematical Papers*, Cambridge Univ. Press, 1889.

[6] M. Chen, J. Mountford, and A. Sellen, *A Study in Interactive 3-D Rotation Using 2-D Control Devices.* Computer Graphics (ACM Siggraph '88 Proceedings), 22 (4), 1988, 121–129.

[7] M. Chen and K. C. Smith, *A Technique for Specifying Rotations in Three Dimensions Using a 2-D Input Device.* Proceedings IEEE Montech '87 – Compint '87, 1987,118–120).

[8] M. Chen, *3-D rotation using a 2-D input device.* Develop (The Apple Technical Journal) 14, 40-53 (1993).

[9] J. H. Conway and D. H. Smith, *On Quaternions and Octonions: Their Geometry, Arithmetic, and Symmetry*, A. K. Peters, Wellesley, 2003

[10] H. S. M. Coxeter, *Regular Polytopes*, 3rd ed. New York: Dover, pp. 112–113 and 296, 1973.

[11] J.J. Craig, *Introduction to Robotics: Mechanics and Control, 2nd ed.*, Addison-Wesley, Reading, 1986.

[12] Y. Dang, L. H. Kauffman, and D. J. Sandin, *Hypercomplex Iterations: Distance Estimation and Higher Dimensional Fractals*, World Scientific, New Jersey, 2002.

[13] L. Euler, *Formulae Generales Pro Translatione Quacunque Corporum Rigidorum.* Presented to the St. Petersburg Academy on October 9, 1775. Novi Commentarii Academiae Scientiarum Petropolitanae **20**, 1776, pp. 189–207 [E478]. Reprinted in Opera Omnia: Series 2, Volume **9**, pp. 84–98, and in *Theoria motus corporum solidorum seu rigidorum*, ed. nova, 1790, pp. 449–460 [E478a].

[14] L. Euler, *Nova methodus motum corporum rigidorum determinandi.* Presented to the St. Petersburg Academy on October 16, 1775. Novi Commentarii academiae scientiarum Petropolitanae 20, 1776, pp. 208-238. [E479]. Reprinted in Opera Omnia: Series 2, Volume **9**, pp. 99–125, and in *Theoria motus corporum solidorum seu rigidorum*, ed. nova, 1790, pp. 460–481 [E479a].

[15] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice in C (2nd Edition)*, Addison-Wesley, Reading, 1995

[16] W. R. Hamilton, *On Quaternions.* Proceedings of the Royal Irish Academy, Nov. 11, 1844, vol. 9, vol. 9 (1847), 1-16.

[17] W. R. Hamilton, *The Mathematical Papers of Sir William Rowan Hamilton, Vol. III*, Cambridge Univ. Press, 1967.

[18] A. J. Hanson, *Visualizing Quaternions.*, Morgan Kaufmann/Elsevier, City, 2006.

[19] J. C. Hart, G. K. Francis, and L. H. Kauffman, *Visualizing quaternion rotation*, ACM Transactions on Graphics Volume 13 , Issue 3 July 1994 (cf. also Air On Dirac Strings: dirac.mpg at:
<http://www.evl.uic.edu/hypercomplex/html/dirac.html>)

[20] J. C. Hart, *Quaternion Demonstrator*, <http://graphics.stanford.edu/courses/cs348c-95-fall/software/quatdemo/>
[21] S. Helgason, *Differential Geometry, Lie Groups, and Symmetric Spaces*, Academic Press, New York, 1978.
[22] K. Henriksen, S. Sporring, and K. Hornbaek, *Virtual Trackballs Revisited.* IEEE Transactions on Visualization & Computer Graphics, March/April 2004 (Vol. 10, No. 2) pp. 206-216.
[23] D. Hestenes, *New Foundations for Classical Mechanics, Kluwer, Dordrecht, 1999*
[24] D. Hestenes and G. Sobczyk, *Clifford Algebra to Geometric Calculus*, Kluwer, Dordrecht, 1984.
[25] B.K.P. Horn, *Solution of Absolute Orientation using Unit Quaternions*, Journal of the Optical Society A, Vol. 4, No. 4, pp. 629–642, April 1987.
[26] B.K.P. Horn, *Notes on Unit Quaternions and Rotations*, people.csail.mit.edu/bkph/articles/Quaternions.pdf.
[27] C. Hu, M. Q. H. Meng, M. Mandal, and P. X. Liu, *Robot Rotation Decomposition Using Quaternions*, Proc. 2006 IEEE International Conference on Mechatronics and Automation (ICMA 2006), 2006, p.1158–1163.
[28] J. Hultquist, *A Virtual Trackball*, Graphics Gems, A.S. Glassner, ed. , vol. I, chapter 9, pp. 462-463, Academic Press, 1990.
[29] J. V. Jose and E. J. Saletan, *Classical Dynamics: A Contemporary Approach.*, Cambridge Univ. Press, Cambridge, 1998
[30] J. Kuipers, *Quaternions and Rotation Sequences*, Princeton Univ. Press, Princeton, 1999
[31] C. W. Misner, K. S. Thorne, and J. A. Wheeler, *Gravitation*, W. H. Freeman, 1973
[32] B. Palais and R. Palais, *Euler's Fixed Point Theorem: The Axis of a Rotation.* To appear in J. Fixed Point Theory and Applicatons.
[33] B. Palais and R. Palais, and S. Rodi, *Euler's Theorem on The Axis of a Rotation: Proofs Old and New .* To appear
[34] B. Palais, *Dirac Belt Trick: About this object.* Virtual Math Museum, virtualmathmuseum.org, 2008.
[35] R, Penrose, *The Road to Reality: A Complete Guide to the Laws of the Universe.*, Jonathan Cape, London, 2004
[36] O. Rodrigues, *Des lois géométriques qui regissent les déplacements d'un système solide dans l'espace.* Journal de Mathématiques Pures et Appliquées 5, 1840, 380-440.
[37] H. A. Schwarz, *Zur Theorie der hypergeometrischen Reihe*, J. reine angew. Math. 75, 292-335, 1873.
[38] J.M. Selig, *Introductory Robotics*, Prentice Hall, Englewood Cliffs, 1992.
[39] J. M. Selig, *Active Versus Passive Transformations in Robotics*, IEEE Robotics & Automation Magazine, Los Alamitos, March 2006 pp. 79–84.
[40] P. Shirley, M. Ashikhmin, M. Gleicher, S. R. Marschner, E. Reinhard, K. Sung, W. B. Thompson, and P. Willemsen, *Fundamentals of Computer Graphics, Second Edition*, A K Peters, Wellesley, 2005)
[41] K. Shoemake, *Arcball Rotation Control.* Graphics Gems IV, Academic Press, Boston, pp. 175–192, 1994.
[42] K. Shoemake, *ARCBALL: A user interface for specifying three-dimensional orientation using a mouse./*, Proc. Graphics Interface, Morgan Kaufmann, San Francisco, pp. 151–156. 1992.

[43] K. Shoemake, *Animating Rotation with Quaternion Curves.* Computer Graphics (SIGGRAPH '85 Proceedings held in San Francisco, CA), Vol. 19, No 3, pp. 245–254, July, 1985

[44] K. Shoemake, *Quaternion Calculus and Fast Animation, Computer Animation: 3-D Motion Specification and Control.* SIGGRAPH 1987 Tutorial, pp. 101–121, 1987.

[45] R. A. Spurrier, *Comment on "Singularity-Free Extraction of a Quaternion from a Direction-Cosine Matrix".* Journal of Spacecraft and Rockets, Vol. 15, July 1978, p. 255.

[46] J. Stillwell, *The Story of the 120-Cell*, Notices. Amer. Math. Soc. 48, 17–24, 2001.

[47] J. P. Ward, *Quaternions and Cayley Numbers.*, Kluwer, Dordrecht, 1997

[48] C. Wessel, *Om directionens analytiske betegning*, Proc. Royal Danish Academy of Sciences and Letters, 1799

[49] C. Wessel, *On the Analytical Representation of Direction*, Kgl. Danske Videnskabernes Selskab, Copenhagen,1999 (Add see also my Flash animation of derivation of complex multiplication from rotation)

[50] J. Wolf, *Spaces of Constant Curvature*, Publish or Perish, Berkeley, 1977

Dept. of Math. The Univ. of Utah Salt Lake City, UT 84112

Dept. of Math. 103 MSTB Univ. of California at Irvine Irvine, CA 92697

E-mail address:

palais@math.utah.edu

palais@uci.edu