A rational quaternion spline of arbitrary continuity

J.K. Johnstone* and J.P. Williams†
July 1999

Abstract

Quaternion splines are a classical tool for orientation control in computer animation and robotics. In this paper, we design a rational quaternion spline. This quaternion spline has many desirable properties: it is a fully general NURBS curve of arbitrary continuity, most naturally with $C^2$ continuity; it is of high quality, demonstrably better than earlier curves; its construction is an efficient generalization of classical interpolation techniques in Euclidean space, leading to simple implementation and easy incorporation into existing NURBS-based modelers; and it is coordinate-frame invariant.

The vast majority of quaternion splines have been non-rational curves. There are many advantages to the design of rational quaternion splines, such as computational efficiency (both in designing the quaternion spline and in later using the spline) and compatibility with existing NURBS technology. The few other rational quaternion splines have been limited to $C^1$ continuity and have had other difficulties.

We believe that our rational quaternion spline will be of considerable use in interactive animation, interactive robot motion control, and motion analysis. Its design is also suggestive of a general approach for the design of rational curves on arbitrary surfaces.

Keywords: quaternion, quaternion spline, orientation, animation, rational curves, NURBS, interpolation.

---

* Geometric Modeling Lab, 125 Campbell Hall, Department of Computer and Information Sciences, UAB, Birmingham, AL 35294.
† Imaging and Visualization Group, Siemens Corporate Research, Princeton, NJ.
1 Introduction

1.1 A rational quaternion spline

Quaternion splines are a classical tool for orientation control in computer animation and robotics. The design of quaternion splines has received a great deal of attention since Shoemake introduced them in 1985 [Shoemake 85]. However, quaternion splines have traditionally been non-rational curves [Shoemake 85, Duff 85, Pletinckx 89, Schlag 91, Barr 92, Nielson 92, Nielson 93, Nam 95, Park 95, Kim 95, Rama 97]. The only exceptions are a quaternion spline constructed from spherical bicaps by Wang and Joe [Wang 93] and a quadratic quaternion spline of Nielson [Nielson 92, Nielson 93], both of which have restricted $C^1$ continuity. In this paper, we design a rational quaternion spline that can be made with arbitrary continuity. It is also of superior quality compared to both rational and nonrational quaternion splines.

A rational quaternion spline has many advantages. Rational curves (usually represented as NURBS) are the most efficient of curves, due to the inherent efficiency of polynomial computations and the elegant and powerful theory of rational Bezier and B-spline curves. As a result of this elegance, rational Bezier curves and NURBS are the de facto standard in modeling systems, with a large, established suite of algorithms for their manipulation. Thus, the design of a rational quaternion spline will allow the curve to be incorporated immediately into existing software and geometric models, and subsequent computations with the curve will be simple and efficient. Our algorithm for the quaternion spline’s construction also is efficient, largely because of the rationality of the computations. This will allow interactive design for interactive animation, interactive robot motion control, and efficient motion analysis. Since the design of a motion matching a designer’s intent is as much of an art as a science, interactive design and refinement is a desirable feature.

1.2 Design on a surface

The main challenge of quaternion spline design is that it involves the design of a curve on a surface. A unit quaternion is a point on the unit sphere $S^3$ in 4-space, $x_1^2 + x_2^2 + x_3^2 + x_4^2 - 1 = 0$. In particular, the unit quaternion $(\cos \frac{\theta}{2}, v \sin \frac{\theta}{2})$, $v \in \mathbb{R}^3$, $\|v\| = 1$, represents the orientation of an object where the object in its canonical orientation has been rotated by $\theta$ radians about the axis $v$. (See Section 3 for a full review of quaternions.) If the orientations of a rigid object in a finite series of keyframes are represented by quaternions, the design of a smooth curve on $S^3$ interpolating these quaternions represents a smooth motion of the object through the desired orientations. This interpolating curve on $S^3$ is called a quaternion spline. Thus, quaternion spline design is curve design on a surface.

The design of a curve on a surface is more challenging than the conventional design of a curve in Euclidean space, since surfaces are Riemannian spaces with a significantly different geometry than Euclidean space. For example, in moving from Euclidean to Riemannian geometry, straight lines are replaced by geodesics. Indeed, the constraint of the quaternion spline to $S^3$ is the major reason for the nonrationality of the existing quaternion spline methods, since constraint to a sphere is mostly easily done through a nonrational analogue of linear interpolation, sleping [Shoemake 85, Duff 85, Pletinckx 89, Schlag 91, Nielson 92, Nielson 93, Kim 95, Nam 95], or by inherently nonrational constrained optimization [Barr 92, Rama 97].

Since the modeling of curves in Euclidean space is much better understood than the modeling of curves in Riemannian space, another promising approach to the design of a quaternion spline is to somehow reduce the problem to the design of a curve in Euclidean space, unconstrained to a surface. An approach for achieving this reduction is as follows. Let $f : \mathbb{R}^m \rightarrow S^3$ be a map to

---

1 A curve is rational if its parameterization can be expressed as a rational map. A map $(x_1, \ldots, x_n) \mapsto (f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n))$ is rational if its components $f_i$ are all rational polynomials (quotients of polynomials).
S^3. Then any curve C in m-space yields a curve f(C) on S^3 (Figure 1). We can now concentrate on the Euclidean problem of designing the curve C. The problem of constraining the curve to the surface has been effectively reduced to the problem of constructing a map to the surface.

**Definition 1.1** The design of a curve on a surface through the design of a curve in m-space and a map from m-space to the surface will be called the Euclidean-space approach to curve design on the surface.

Since we want the quaternion spline to be rational, the curve C and the map f must be rational. It is simple to construct a rational curve C, since the classical methods for curve design in Euclidean space yield rational curves [Farin 97]. The construction of a good rational map to S^3 is more challenging.

We are now ready to give our general algorithm for the design of a rational quaternion spline through the quaternions \( \{p_i\}_{i=1,...,k} \subset S^3 \). Refer to Figures 2-7.

**(1a)** Design a rational map to S^3, \( f: \mathbb{R}^4 \to S^3 \).

**(1b)** Compute the inverse map \( f^{-1}: S^3 \to \mathbb{R}^4 \).

**(2a)** Compute \( \{f^{-1}(p_i)\}_{i=1,...,k} \), mapping the quaternions to Euclidean space.

**(2b)** Design a rational curve C in Euclidean space, interpolating \( \{f^{-1}(p_i)\}_{i=1,...,k} \).

**(2c)** Compute \( f(C) \), mapping the curve back to the surface. This is the desired rational quaternion spline interpolating the quaternions \( \{p_i\}_{i=1,...,k} \).
Figure 2: Step 1a

Figure 5: Step 2a

Figure 3: Step 1b \((g = f^{-1})\)

Figure 6: Step 2b

Figure 4: Quaternions

Figure 7: Step 2c
We prefer rational maps to $S^3$ with domain $\mathbb{R}^4$, the resident Euclidean space, since then $f^{-1}(p_i)$ is a curve in step 2a (as opposed to a point if the domain is $\mathbb{R}^3$ or a surface if $\mathbb{R}^3$). This leads to a useful flexibility in the interpolation step 2b, since we can interpolate curves rather than points, which can be used to design better curves on the surface.

Note that the inverse map $f^{-1}$ need not be rational. Also note that (1a-b) are preprocessing steps, (2b) is a well understood interpolation problem and (2a) and (2c) are simple operations. Thus, this algorithm is easily implemented and integrated into existing modeling/animation systems.

Since $f$ is a rational map, the curve on $S^3$ inherits the continuity of the curve $C$ in $\mathbb{R}^4$. Since it is simple to define cubic interpolating curves in Euclidean space with $C^2$ continuity, or curves of higher degree with even higher continuity, the curve on $S^3$ can easily have $C^2$ continuity and indeed arbitrary continuity. Other quaternion splines have considerable difficulty with continuity.

The rest of the paper will be structured as follows. Section 2 reviews related work on quaternion splines and Section 3 gives the basic theory of quaternions. In Section 4, we build a good rational map to $S^3$ and its inverse. In Section 5, we discuss the design of an interpolating curve in Euclidean space, especially the interpolation of curves as opposed to points. In Section 6, we show how to map a curve in $\mathbb{R}^4$ back to a curve on $S^3$ while preserving the Bezier structure, for the important case of the cubic Bezier curve. In Section 7, we discuss a region of instability for the inverse map and show how to avoid this region. Section 8 gives examples of quaternion splines designed using the new method. In Section 9, we show how to analyze the quality of a quaternion spline, and use this measure to show the superiority of our method over several other methods. We end with some conclusions and ideas for future work in Section 10.

2 Related work

There is a rich literature on quaternion splines. [Shoemake 85] introduced them as a solution for keyframe animation in 1985. He used spherical linear interpolation (slerping) between points on $S^3$. Many others have also used slerping [Duff 85, Pletinckx 89, Schlag 91, Nielson 92, Nielson 93, Kim 95, Nam 95]. The spherical linear interpolation between points $P_0$ and $P_1$ on $S^3$ is:

$$P(t) = \frac{\sin((1-t)\theta)P_0 + \sin(t\theta)P_1}{\sin \theta}$$

where $\cos \theta = P_0 \cdot P_1$. This represents an arc of the great circle between $P_0$ and $P_1$. The de Casteljau algorithm can be used to build traditional Bezier curves. A spherical analog to the de Casteljau algorithm based upon this spherical linear interpolation is used to build quaternion splines, mimicking Bezier [Shoemake 85, Kim 95], B-spline [Duff 85, Nielson 92, Nielson 93, Kim 95], Hermite [Kim 95, Nam 95], cardinal spline [Pletinckx 89], and Catmull-Rom [Schlag 91] curves. Since slerping is a nonrational operation, all of the methods based on slerping generate nonrational curves. Most of these curves are defined only by a geometric construction, and have no closed form algebraic definition. Computation of derivatives of these curves is complicated, as is the imposition of $C^2$ continuity. Kim et. al. [Kim 95] propose solutions to the latter two problems, using Lie algebra and its exponential map. (Our use of conventional Bezier curves completely removes these problems with continuity or derivative calculation.) [Park 97] also uses Lie algebra to design the quaternion spline. They work on the SO(3) manifold, rather than $S^3$ manifold.

[Barr 92] uses constrained optimization to develop optimal quaternion splines, optimizing the constraint that the curve lie on $S^3$. They also introduce low covariant acceleration as a desirable property of a quaternion spline, and incorporate it into their constraints. This ability to incorporate extra constraints into the optimization is a nice feature of their algorithm. Their quaternion spline is nonrational with no closed-form expression, and their numerical optimization can be expensive. Their approach is refined for added efficiency in [Rama 97].
Figure 8: Any change of orientation can be expressed as a rotation about a fixed axis

[Wang 93, Wang 94] and [Nielson 93] are close in spirit to this paper, since they design rational quaternion splines. However, these curves are limited in scope. [Wang 93, Wang 94] and [Nielson 93] develop quadratic curves with $G^1$ continuity, and [Wang 94] develops sextic curves with $C^2$ continuity. At least $C^2$ continuity is desirable, especially for animation. The quaternion splines of [Wang 93] are built from biarcs (great circles of the sphere), and those of [Nielson 93] from circular arcs. Both Wang and Nielson’s methods also involve heuristic, data-dependent choices that can be difficult to make, such as the choice of a spherical biarc from a one-parameter family of valid spherical biarcs, or a center of projection. Our method generalizes the work of Wang and Nielson, by creating rational curves of arbitrary even degree (all rational curves on $S^3$ have even degree [Wang 94]) and arbitrary continuity, based on traditional NURBS and without any data-dependent choices.

The Euclidean-space approach has been used before for the design of curves on surfaces [Dietz 93, Wang 94]. The classical solution to the design of trim curves on a surface is an example, where the trim curve is designed in the parameter space of the surface and then mapped back to the surface using the parameterization. Trim curves are not a good solution to quaternion splines, however. It turns out that a parameterization of $S^3$ is not a good map to the surface for quaternion spline design (see Section 9). Its domain is also wrong ($\mathbb{R}^3$ rather than $\mathbb{R}^4$) as discussed in Sections 1 and 5.

3 Quaternions and orientation

The unit quaternion is a preferred representation for the orientation of a rigid object in computer animation. The quaternion was invented by Hamilton in 1843 as a 4-dimensional generalization of complex numbers. It was soon recognized that the quaternion can also be used to represent an orientation. To see this, we appeal to a fundamental result of Euler from 1752 [Goldstein 50] (Figure 8).

**Theorem 3.1 (Euler)** A rigid body can be moved from an arbitrary initial orientation to an arbitrary final orientation by a single rotation of the body about a fixed axis.

This shows that the orientation of a rigid body can be represented by a rotation axis $v$, $\|v\| = 1$, and the rotation angle $\theta$ about this axis (Figure 8) required to rotate the body into the given
orientation from a canonical orientation (say, the orientation in which the exact geometry of the body was originally specified). \(v\) and \(\theta\) are encoded in a unit quaternion by \((\cos \frac{\theta}{2}, v \sin \frac{\theta}{2})\). Notice that \(v\) and \(\theta\) can be easily extracted from this encoding.

The quaternion's representation of orientation has two great benefits. First, since it is a unit vector, it can be interpreted geometrically as a point on \(S^3\). Second, we can measure the amount of rotation using this geometric interpretation, since the metric on \(S^3\) is the same as the metric of the rotation group [Misner 73].

**Theorem 3.2** The metrics on \(S^3\) and the rotation group \(SO(3)\) are equivalent.

In short, the speed of a body's rotation through 3-space can be directly measured, and directly controlled, through the length of the quaternion spline. This is an important tool for motion control in animation or robotics.

The rotation of an object by a quaternion need not be computed using its literal interpretation as rotation about an axis. Quaternion algebra offers an easier solution, which we now review. The quaternion \((a, b, c, d)\) is actually shorthand for \(a + bi + cj + dk\), where \(i\), \(j\) and \(k\) satisfy the relationship

\[i^2 = j^2 = k^2 = ijk = -1.\] (1)

A quaternion \((a, b, c, d)\) is often expressed as a scalar component and a vector component, \((s, v)\), where \(s = a\) and \(v = (b, c, d)\). Using this notation and applying (1), the formula for quaternion multiplication is

\[s_1, v_1 \times s_2, v_2 = [s_1 \times s_2 - v_1 \cdot v_2, s_1 \times v_2 + s_2 \times v_1 + v_1 \times v_2]\] (2)

Then, a point \(p \in \mathbb{R}^3\) is rotated by the unit quaternion \([s, v]\) to the point \([s, -v] \times [0, p] \times [s, v]\). This is a quaternion, but it can be interpreted as a point in 3-space since its scalar part is necessarily 0.

Mathematically, an orientation can be represented by either of two antipodal quaternions (by flipping the rotation axis \(v\)). That is, \(S^3\) is a double covering of \(SO(3)\). However, when using quaternions for motion control, only one of the two quaternion representations is appropriate in a given context: the quaternion with the smaller angular gap on \(S^3\) to the previous quaternion. This is true because a small angular gap between consecutive quaternions creates less spinning than a large angular gap. Consider the difference between a small rotation and the complementary rotation that represents an almost complete pirouette. If the designer's intent is to include the pirouette, then this should be made explicit in the motion control by adding intermediate quaternions that imply this spin. A motion should not introduce spinning unless it is explicitly designed in.

The quaternion has several advantages over other representations for orientation, such as the rotation matrix and Euler angles. Unlike Euler angles, quaternions do not experience gimbal lock, can be combined easily, and have an effectively unique representation for each orientation. Unlike rotation matrices, quaternions have a concise representation, 4 scalars rather than 9.\(^2\) And unlike both Euler angles and rotation matrices, quaternions have a natural geometric interpretation through identification with \(S^3\), which is a crucial element in algorithmic development.

### 4 A map to and from \(S^3\)

The first step in our algorithm for the design of a rational quaternion spline is to design a good rational map from 4-space to the surface \(S^3\). Recall that we prefer a map with domain \(\mathbb{R}^4\), so that

---

\(^2\)A quaternion is still slightly larger than it needs to be since \(v\), being a unit vector, is fully determined by only 2 of its elements. That is, \(v\) and \(\theta\) could have been encoded in a quaternion using only 3 scalars. However, the redundancy of 4 scalars is necessary for the quaternion's geometric interpretation on a unit sphere and the metric equivalence of Theorem 3.2.
the data points will be mapped to one-dimensional curves in Euclidean space and the curve design in Euclidean space will enjoy more flexibility.

Our approach is as follows. We shall first develop a normal form for all rational maps of \( \mathbb{R}^4 \) to \( S^3 \). We will then choose a particular map. Finally, we will compute the inverse of this map.

### 4.1 Rational maps of \( \mathbb{R}^4 \) to \( S^3 \)

Consider a rational map from \( \mathbb{R}^4 \) to \( S^3 \):

\[(x_1, \ldots, x_4) \mapsto \left( \frac{f_1(x_1, \ldots, x_4)}{f_5(x_1, \ldots, x_4)}, \ldots, \frac{f_4(x_1, \ldots, x_4)}{f_5(x_1, \ldots, x_4)} \right)\]

where \( f_1, \ldots, f_5 \) are polynomials. Since the image lies on \( S^3 \), \( f_1^2 + \ldots + f_4^2 = f_5^2 \) and \( (f_1, f_2, f_3, f_4, f_5) \)

is a Pythagorean quintuple.

**Definition 4.1** \((a_1, \ldots, a_5) \in K^{n+1} \) is a Pythagorean quintuple over \( K \) if \( a_1^2 + \ldots + a_4^2 = a_5^2 \).

Thus, the study of rational maps from \( \mathbb{R}^4 \) to \( S^3 \) is equivalent to the study of Pythagorean quintuples of polynomials. Pythagorean quintuples involve the sum of four squares. In the number theory literature, there is an extensive study of the sum of four squares (driven by the search for a proof that every positive integer is the sum of the squares of four integers [Dickson 52]). An important result was developed by Euler, \(^4\) which we can use to build Pythagorean quintuples, and then a characterization of maps to \( S^3 \).

**Lemma 4.2 (Euler’s Four Squares Theorem [Herstein 75])**

\[
(a_1^2 + a_2^2 + a_3^2 + a_4^2)(\hat{a}_1^2 + \hat{a}_2^2 + \hat{a}_3^2 + \hat{a}_4^2) = \\
(a_1 \hat{a}_1 - a_2 \hat{a}_2 - a_3 \hat{a}_3 - a_4 \hat{a}_4)^2 + (a_1 \hat{a}_2 + a_2 \hat{a}_1 + a_3 \hat{a}_4 - a_4 \hat{a}_3)^2 + \\
(a_1 \hat{a}_3 - a_2 \hat{a}_4 + a_3 \hat{a}_1 + a_4 \hat{a}_2)^2 + (a_1 \hat{a}_4 + a_2 \hat{a}_3 - a_3 \hat{a}_2 + a_4 \hat{a}_1)^2
\]

where \( a_1, a_2, a_3, a_4, \hat{a}_1, \hat{a}_2, \hat{a}_3, \hat{a}_4 \) are elements of a commutative ring.

**Corollary 4.3** \((a_1^2 + a_2^2 + a_3^2 + a_4^2, 2a_1 a_4, 2a_2 a_4, 2a_3 a_4, a_1^2 + a_2^2 + a_3^2 + a_4^2) \) is a Pythagorean quintuple for any polynomials \( a_1, a_2, a_3, a_4 \).

**Proof:** Let \((\hat{a}_1, \hat{a}_2, \hat{a}_3, \hat{a}_4) = (a_1, -a_2, -a_3, a_4)\).

The following lemma establishes a weaker version of the necessary condition associated with Corollary 4.3. This in turn leads to the desired necessary and sufficient condition for rational maps of \( \mathbb{R}^4 \) to \( S^3 \).

**Lemma 4.4** A quintuple of polynomials is Pythagorean only if it can be expressed in the form

\[
a(a_1^2 + a_2^2 + a_3^2 + a_4^2, 2a_1 a_4, 2a_2 a_4, 2a_3 a_4, a_1^2 + a_2^2 + a_3^2 + a_4^2)
\]

for some polynomials \( a_1, a_2, a_3, a_4, \frac{1}{a} \).#

\(^3\)This term derives from the Pythagorean Theorem on right triangles, which involves Pythagorean triples.

\(^4\)This result shows that the product of a sum of four squares and a sum of four squares is another sum of four squares. This reduces the problem of showing that every integer is the sum of four squares to the simpler problem of showing that every prime is the sum of four squares, since every integer can be expressed as the product of primes.
**Proof:** Let \((p_1, p_2, p_3, p_4, p_5)\) be a Pythagorean quintuple of polynomials. If \(p_1 = p_5\), let \((a_1, a_2, a_3, a_4, \alpha) = (p_1, 0, 0, 0, \frac{1}{p_1})\). Thus, we may assume without loss of generality that \(p_1 \neq p_5\). Let \((a_1, a_2, a_3, a_4, \alpha) = (p_2, p_3, p_4, p_5 - p_1, \frac{1}{2p_5 - p_1})\). Then \(a_1, a_2, a_3, a_4, \alpha\) generate the Pythagorean quintuple \((p_1, \ldots, p_5)\) as in (3). In particular,
\[
\alpha(a_1^2 + a_2^2 + a_3^2 - a_4^2) = \frac{p_2^2 + p_3^2 + p_4^2 - (p_5 - p_1)^2}{2(p_5 - p_1)}
\]
and applying \(p_1^2 + p_2^2 + p_3^2 + p_4^2 = p_5^2\),
\[
\alpha(a_1^2 + a_2^2 + a_3^2 - a_4^2) = \frac{-2p_1^2 + 2p_1p_5}{2(p_5 - p_1)} = p_1.
\]
Also,
\[
\alpha(2a_i a_4) = \frac{2p_i + 1(p_5 - p_1)}{2(p_5 - p_1)} = p_{i+1}
\]
for \(i = 1, 2, 3\). Finally,
\[
\alpha(a_1^2 + a_2^2 + a_3^2 + a_4^2) = \frac{p_2^2 + p_3^2 + p_4^2 + (p_5 - p_1)^2}{2(p_5 - p_1)} = p_5
\]
Thus, \((p_1, p_2, p_3, p_4, p_5) = \alpha(a_1^2 + a_2^2 + a_3^2 - a_4^2, 2a_1 a_4, 2a_2 a_4, 2a_3 a_4, a_1^2 + a_2^2 + a_3^2 + a_4^2)\). ■

**Corollary 4.5** A map is a rational map of \(\mathbb{R}^4\) to \(S^3\) if and only if it is of the form:
\[
(x_1, x_2, x_3, x_4) \mapsto \frac{1}{a_1^2 + a_2^2 + a_3^2 + a_4^2}, (a_1^2 + a_2^2 + a_3^2 - a_4^2, 2a_1 a_4, 2a_2 a_4, 2a_3 a_4)
\]
(4)
where \(a_1, a_2, a_3, a_4\) are polynomials over \(x_1, x_2, x_3, x_4\) (or some permutation of this form).\(^5\)

**Proof:** Consider a rational map of \(\mathbb{R}^4\) to \(S^3\), \((x_1, x_2, x_3, x_4) \mapsto (f_1, f_2, f_3, f_4)\), where \(f_1, \ldots, f_5\) are polynomials over \(x_1, \ldots, x_4\). Then \((f_1, \ldots, f_5)\) is a Pythagorean quintuple so it can be expressed in the normal form \((f_1, \ldots, f_5) = \alpha(a_1^2 + a_2^2 + a_3^2 - a_4^2, 2a_1 a_4, 2a_2 a_4, 2a_3 a_4, a_1^2 + a_2^2 + a_3^2 + a_4^2)\) for some polynomials \(a_1, a_2, a_3, a_4, \frac{1}{\alpha}\) by Lemma 4.4. Thus, the map is of the form (4), since the leading \(\alpha\) cancels when expressed in \((f_1, f_2, f_3, f_4, f_5)\). Notice that it is legal to cancel the \(\alpha\), since \(\alpha = \frac{1}{\beta}\) for some polynomial \(\beta\), which implies that \(\alpha\) is never zero.

Now consider a map of the form (4). This is a rational map of \(\mathbb{R}^4\) to \(S^3\) by Corollary 4.3. ■

The most natural choice for the polynomials \(a_i\) in Corollary 4.5 is \(a_i = x_i\) and the identity permutation.

\(^5\)This result generalizes to rational maps from \(\mathbb{R}^n\) to \(S^{n-1}\), \(n \geq 2\), using an identical proof. That is, a map is a rational map of \(\mathbb{R}^n\) to \(S^{n-1}\), \(n \geq 2\), if and only if it is of the form:
\[
(x_1, \ldots, x_n) \mapsto \left(\frac{a_1^2 + \cdots + a_{n-1}^2 - a_n^2}{a_1^2 + \cdots + a_n^2}, \frac{2a_1 a_n}{a_1^2 + \cdots + a_n^2}, \ldots, \frac{2a_{n-1} a_n}{a_1^2 + \cdots + a_n^2}\right)
\]
where \(a_1, \ldots, a_n\) are polynomials over \(x_1, \ldots, x_n\).
Definition 4.6 The most natural map to $S^3$ is $M : \mathbb{R}^4 - \{0\} \rightarrow S^3$ defined by:
\[
M(x_1, x_2, x_3, x_4) = \frac{1}{x_1^2 + x_2^2 + x_3^2 + x_4^2}(x_1^2 + x_2^2 - x_3^2 - x_4^2, 2x_1x_2, 2x_2x_3, 2x_3x_4)
\]
(5)

We shall use the map $M$ as our rational map to $S^3$. We fully analyze this map in [Johnstone 99b]. For example, we show that it is a powerful extension of the inverse map of stereographic projection.

Remark 4.7 Ironically, Euler’s Four Squares Theorem, our main tool in the development of the map $M$ to the quaternion sphere $S^3$, can be viewed as an anticipation of quaternions themselves (a century in advance of their invention by Hamilton!) since it encodes the product formula for quaternions (compare Lemma 4.2):
\[
(a_1 + a_2i + a_3j + a_4k)(\bar{a}_1 + \bar{a}_2i + \bar{a}_3j + \bar{a}_4k) = (a_1\bar{a}_1 - a_2\bar{a}_2 - a_3\bar{a}_3 - a_4\bar{a}_4) + (a_1\bar{a}_2 + a_2\bar{a}_1 + a_3\bar{a}_4 - a_4\bar{a}_3)i + (a_1\bar{a}_3 - a_2\bar{a}_4 + a_3\bar{a}_1 + a_4\bar{a}_2)j + (a_1\bar{a}_4 + a_2\bar{a}_3 - a_3\bar{a}_2 + a_4\bar{a}_1)k
\]

4.2 The inverse map

To map the quaternions to Euclidean space, the inverse of our map to $S^3$ is needed (step 1b of the algorithm in Section 1.2). Since the manifold $\mathbb{R}^4$ is one dimension larger than $S^3$, one would expect the preimage of a point of $S^3$ to be a curve in $\mathbb{R}^4$. This is indeed the case. In fact, the inverse of a point is simply a line. The proof of the following theorem is simplified by working in projective space.

Definition 4.8 Real projective $n$-space $P^n$ is the space $\{ (x_1, x_2, \ldots, x_{n+1}) : x_i \in \mathbb{R}, \text{ not all zero} \}$ under the equivalence relation
\[
(x_1, \ldots, x_{n+1}) = k(x_1, \ldots, x_{n+1}), \quad k \neq 0 \in \mathbb{R}.
\]
(6)

The point $(x_1, \ldots, x_{n+1})$ in projective $n$-space, $x_{n+1} \neq 0$, is equivalent to the point $(\frac{x_1}{x_{n+1}}, \ldots, \frac{x_n}{x_{n+1}})$ in $n$-space. The point $(x_1, \ldots, x_n, 0)$ in projective $n$-space represents the point at infinity in the direction $(x_1, \ldots, x_n)$. To translate from $n$-space to projective $n$-space, the point $(x_1, \ldots, x_n)$ is typically transformed into the point $(x_1, \ldots, x_n, 1)$. See [Harris 92] for more details on projective space.

Theorem 4.9 $M^{-1} : S^3 \rightarrow \mathbb{R}^4$ is defined as follows:
\[
M^{-1}(x_1, x_2, x_3, x_4) = \begin{cases} 
 t(x_2, x_3, x_4, 1 - x_1), & t \in \mathbb{R}, \ t \neq 0 \\
 H & \text{if } (x_1, x_2, x_3, x_4) \neq (1, 0, 0, 0) \\
 & \text{if } (x_1, x_2, x_3, x_4) = (1, 0, 0, 0)
\end{cases}
\]

where $H$ is the hyperplane $x_4 = 0$ minus the origin. That is, the preimage of $(x_1, x_2, x_3, x_4) \neq (1, 0, 0, 0)$ on $S^3$ is a line through the origin, minus the origin.

Proof: We work in projective space, where the map $M$ becomes
\[
(x_1, x_2, x_3, x_4, x_5) \rightarrow (x_1^2 + x_2^2 + x_3^2 - x_4^2, 2x_1x_2, 2x_2x_3, 2x_3x_4, x_1^2 + x_2^2 + x_3^2 + x_4^2).
\]
(7)

Let $p = (p_1, p_2, p_3, p_4, 1) \in S^3 \subset P^4$. We want to determine the conditions on $q = (q_1, q_2, q_3, q_4, q_5)$ so that $M(q_1, q_2, q_3, q_4, q_5) = p$. Suppose that $M(q_1, q_2, q_3, q_4, q_5) = p$. Using (6), we have
\[
q_1^2 + q_2^2 + q_3^2 - q_4^2 = kp_1
\]
(8)
\begin{align*}
2q_1q_4 &= kp_2 \\
2q_2q_4 &= kp_3 \\
2q_3q_4 &= kp_4 \\
q_1^2 + q_2^2 + q_3^2 + q_4^2 &= k
\end{align*}

for some \( k \neq 0 \). \( q_5 \) is arbitrary, since it does not appear in these equations. Subtracting (8) from (12), we have \( 2q_1^2 = k(1-p_1) \) or

\[
q_4 = \pm \sqrt{\frac{k(1-p_1)}{2}}
\]

**Case 1:** Suppose \( p_1 = 1 \). Then \( p = (1,0,0,0,1) \) since \( p \in S^3 \). \( q_4 = 0 \) by (13) and \( M(q) = (q_1^2 + q_2^2 + q_3^2, 0, 0, 0, q_1^2 + q_2^2 + q_3^2) = (1, 0, 0, 0, 1) \) for any values of \( q_1, q_2, q_3 \), not all zero. That is, \( M(q) = p \) if and only if \( q \in H \) where \( H \) is the hyperplane \( x_4 = 0 \) minus the origin. Equivalently, \( M^{-1}(p) = H \).

**Case 2:** Suppose \( p_1 \neq 1 \). Then \( p_1 < 1 \) since \( p \in S^3 \). From (13), \( q_4 \neq 0 \) and \( q_4 \) is a real number. From (9-11), \( q_i = \frac{kp_{i+1}}{2q_4} \) for \( i = 1, 2, 3 \):

\[ q = \left( \frac{kp_2}{2q_4}, \frac{kp_3}{2q_4}, \frac{kp_4}{2q_4}, q_4, q_5 \right) \]

Using (6),

\[ q = \frac{2q_4}{k} q = (p_2, p_3, p_4, \frac{2q_4^2}{k}, \frac{2q_4q_5}{k}) \]

and then (13),

\[ q = (p_2, p_3, p_4, 1 - p_1, \pm \sqrt{\frac{2(1-p_1)}{k}q_5}) \]

Since \( q_5 \) and \( k \) are arbitrary,

\[ q = (p_2, p_3, p_4, 1 - p_1, k') \quad k' \in \mathbb{R} \]

We have shown that \( M(q) = p \) only if \( q = (p_2, p_3, p_4, 1 - p_1, k') \).

On the other hand, if \( q = (p_2, p_3, p_4, 1 - p_1, k') \), \( k' \in \mathbb{R} \) and \( p_1 \neq 1 \), then

\[ M(q) = (p_2^2 + p_3^2 + p_4^2 - (1 - p_1)^2, 2p_2(1-p_1), 2p_3(1-p_1), 2p_4(1-p_1), p_2^2 + p_3^2 + p_4^2 + (1 - p_1)^2) \]

Using \( p_2^2 + p_3^2 + p_4^2 = 1 (p \in S^3) \),

\[ M(q) = (2p_1(1-p_1), 2p_2(1-p_1), 2p_3(1-p_1), 2p_4(1-p_1), 2(1-p_1)) \]

or \( M(q) = (p_1, p_2, p_3, p_4, 1) \) using (6). Thus, \( M(q) = p \) if and only if \( q = (p_2, p_3, p_4, 1 - p_1, k') \), \( k' \in \mathbb{R} \). Translating back from projective space, \( M(q) = p \) if and only if \( q = t(p_2, p_3, p_4, 1 - p_1), t \neq 0 \in \mathbb{R} \). Equivalently, \( M^{-1}(p) = t(p_2, p_3, p_4, 1 - p_1), t \neq 0 \in \mathbb{R} \). (It is understandable that the preimage does not contain the origin, since \( M \) is undefined there.)

**Definition 4.10** The special point \((1,0,0,0)\) is called the pole of the map \( M^{-1} \). \((x_2, x_3, x_4, 1 - x_1)\) is called the defining point of the preimage \( M^{-1}(x_1, x_2, x_3, x_4) \).

Notice the beautiful simplicity of the defining point.
5 Designing the curve in Euclidean space

Once the quaternions have been mapped into Euclidean space, the next step is to design the curve in Euclidean space. This involves interpolating a curve through the inverse image lines $M^{-1}(p_i)$. We will reduce this problem to point interpolation, by choosing one point per line and applying conventional interpolation through a set of points. Each point should be chosen wisely, taking full advantage of our flexibility to choose any point on the line. A more precise statement of step (2b) of our algorithm for curve design on a surface is now:

(i) Intelligently choose $k$ points $\{q_i\}$ on the $k$ lines $\{M^{-1}(p_i)\}$.

(ii) Design a rational curve $C$ in Euclidean space interpolating $\{q_i\}_{i=1,...,k}$, using classical techniques.

To understand our choice of points $\{q_i\}$ on the lines $\{M^{-1}(p_i)\}$, we must motivate the desirability of designing a short curve in Euclidean space. Other factors being equal, we prefer shorter quaternion splines. We want to minimize any unnecessary spinning of the object: loop-the-loops or gyration should not be introduced unless explicitly designed in the motion. Since the metric of $S^3$ is the same as the metric of the rotation matrix group $SO(3)$ (Theorem 3.2), the removal of unnecessary object rotation is equivalent to the removal of unnecessary length in the quaternion spline. Thus, we prefer shorter quaternion splines. Then, although curve length in Euclidean space is not equivalent to length of the associated quaternion spline, shorter curves in Euclidean space are associated with shorter quaternion splines.

Therefore, we would prefer the curve in Euclidean space to move efficiently between the lines $\{M^{-1}(p_i)\}$ and a good choice of points $\{q_i\}$ is a set of closest points:

- On the first inverse line, let $q_1$ be the intersection of the line $M^{-1}(p_1)$ and $S^3$.
- On subsequent lines, let $q_i$ be the closest point to $q_{i-1}$.

6 Mapping back to $S^3$

After the curve is designed in $\mathbb{R}^4$, it must be mapped back to a curve on $S^3$ using $M$. The following theorem shows how this mapping is done, segment by segment, for the important case of a cubic Bezier curve in $\mathbb{R}^4$. We concentrate on the cubic Bezier curve since classical design of an interpolating curve in $\mathbb{R}^4$ will generate a cubic polynomial curve [Farin 97]. The image of other curves under $M$ can be computed similarly.

**Theorem 6.1** Let $c(t)$ be a cubic Bezier curve in 4-space with control points $b_i = (b_{i1}, b_{i2}, b_{i3}, b_{i4})$, $i = 0, ..., 3$. The image of $c(t)$ under $M$ is a rational sextic Bezier curve with weights

$$w_k = \sum_{0 \leq i \leq 3, 0 \leq j \leq 3, i + j = k} \frac{\binom{3}{i} \binom{3}{j}}{\binom{6}{k}} (b_{i1}b_{j1} + b_{i2}b_{j2} + b_{i3}b_{j3} + b_{i4}b_{j4})$$

\[ (14) \]

\[ ^{6} \text{For example in the extreme case, motion along an inverse line } M^{-1}(p_i) \text{ in } \mathbb{R}^4 \text{ causes no associated motion on } S^3. \text{ However, in general, the arc length of the two curves, one in Euclidean space and the other on } S^3, \text{ are strongly related. Moreover, the avoidance of motion in Euclidean space always leads to the avoidance of motion on } S^3. \]
and control points

\[
c_k = \frac{1}{w_k} \sum_{0 \leq i \leq 3} \sum_{0 \leq j \leq 3} \binom{3}{i} \binom{3}{j} \begin{pmatrix} b_{i1}b_{j1} + b_{i2}b_{j2} + b_{i3}b_{j3} - b_{i4}b_{j4} \\ 2b_{i1}b_{j1} \\ 2b_{i2}b_{j1} \\ 2b_{i3}b_{j1} \\ b_{i1}b_{j1} + b_{i2}b_{j2} + b_{i3}b_{j3} + b_{i4}b_{j4} \end{pmatrix} \quad (15)
\]

for \( k = 0, \ldots, 6 \).

**Proof:** Let \( B_i^n(t) \) be the \( i \)th Bernstein polynomial of degree \( n \). Then \( c(t) = \sum_{i=0}^3 B_i^3(t) b_i \). This proof is a simple application of the product rule of Bernstein polynomials [Farin 97]:

\[
B_i^m(t)B_j^n(t) = \binom{m}{i} \binom{n}{j} \binom{m+n}{i+j} B_{i+j}^{m+n}(t).
\]

We will again work in projective space, where we recall that the map \( M \) becomes

\[
(x_1, x_2, x_3, x_4, x_5) \rightarrow (x_1^2 + x_2^2 + x_3^2 - x_4^2, 2x_1x_4, 2x_2x_4, 2x_3x_4, x_1^2 + x_2^2 + x_3^2 + x_4^2).
\]

Let \( M(c(t)) = (m_1(t), m_2(t), m_3(t), m_4(t), m_5(t)) \). \( m_5(t) \) can be simplified using the product rule of Bernstein polynomials, as follows:

\[
m_5(t) = \sum_{i=0}^3 B_i^3(t) b_{i1}^2 + \ldots + \sum_{i=0}^3 B_i^3(t) b_{i4}^2 = \sum_{i=0}^3 \sum_{j=0}^3 \binom{3}{i} \binom{3}{j} \binom{6}{i+j} B_{i+j}^6(t) (b_{i1}b_{j1} + \ldots + b_{i4}b_{j4})
\]

Letting \( k = i + j \),

\[
m_5(t) = \sum_{k=0}^6 B_k^6(t) \sum_{0 \leq i \leq 3} \sum_{0 \leq j \leq 3} \binom{3}{i} \binom{3}{j} \binom{6}{k} (b_{i1}b_{j1} + \ldots + b_{i4}b_{j4})
\]

Computing the other coordinates analogously yields

\[
M(c(t)) = \sum_{k=0}^6 B_k^6(t) \sum_{0 \leq i \leq 3} \sum_{0 \leq j \leq 3} \binom{3}{i} \binom{3}{j} \binom{6}{k} \begin{pmatrix} b_{i1}b_{j1} + b_{i2}b_{j2} + b_{i3}b_{j3} - b_{i4}b_{j4} \\ 2b_{i1}b_{j1} \\ 2b_{i2}b_{j1} \\ 2b_{i3}b_{j1} \\ b_{i1}b_{j1} + b_{i2}b_{j2} + b_{i3}b_{j3} + b_{i4}b_{j4} \end{pmatrix}
\]

which is a sextic rational Bézier curve with weights (14) and control points (15). ■

Notice how the map \( M \) reveals itself in the formula for the control points and weights, its numerator in the control points and its denominator in the weights. The remaining components of the formulae (14-15) are a reflection of the product rule for Bernstein polynomials.
7 Avoiding the pole

The design of our curve is essentially complete. However, we must refine one of the steps in the interests of stability. The mapping of the data into Euclidean space is unstable if any of the data is close to the pole \((1,0,0,0)\). In this section, we shall show how the quaternions \(\{p_i\}_{i=1,...,k}\) can be moved away from the pole for robust curve design.

7.1 Ill-conditioning near the pole

\(M^{-1}\) is not well-behaved near its pole \((1,0,0,0)\). Recall that the image of a point \(p \neq (1,0,0,0)\) under \(M^{-1}\) is a line through the origin and the defining point of \(M^{-1}(p)\) (Definition 4.10). As a point \(p\) on \(S^3\) approaches the pole \((1,0,0,0)\), the defining point of \(M^{-1}(p)\) approaches the origin:

\[
\lim_{(x_1,x_2,x_3,x_4) \to (1,0,0,0)} (x_2, x_3, x_4, 1-x_1) = (0,0,0,0)
\]

Consequently, the defining point becomes an ill-conditioned specification of \(M^{-1}(p)\): small motions of \(p\) can cause large motions of \(M^{-1}(p)\). This is analogous to the solution of a linear system when the condition number of the matrix becomes large, and the linear system becomes very sensitive to perturbation of the matrix.

Example 7.1 If \(p\) on \(S^3\) is at distance \(d\) from the pole, the defining point of \(M^{-1}(p)\) is also at distance \(d\) from the origin. Then a motion of \(d+\epsilon\) of \(p\) (corresponding to a motion of the defining point directly towards the origin) can yield a 90-degree change of orientation of the line \(M^{-1}(p)\).

In other words, near the pole on \(S^3\), there is little correspondence between position of \(p\) and position of \(M^{-1}(p)\). This leads to quaternion splines that jump wildly about the pole or create cusps (Figure 9). We conclude that we want to move data points away from the pole. The following definition gives an empirical notion of how far the points must be moved away.

Definition 7.2 A point \(p \in S^3\) is too close to the pole if the angle formed by the vectors \(p\) and \((1,0,0,0)\) is smaller than 30 degrees. Experimental evidence indicates that points closer to the pole can lead to undesirable behaviour in the curve.
7.2 Moving data away from the pole

We shall use a single rotation about the origin to move the data points away from the pole $(1, 0, 0, 0)$. Once the quaternion spline has been robustly designed, it will be rotated back by the same amount, yielding the desired curve.\(^7\) Let us call the locus of points on $S^3$ such that no data point is within 30 degrees the **empty region** of $S^3$. Our problem reduces to finding a point $p$ in an empty region of $S^3$; then a rotation of $p$ to the pole will move all data points a safe distance from the pole. We first show that there is an empty region to find, and then how to find a point in an empty region.

7.3 Existence of an empty region

First, is there an empty region? It is theoretically possible that the data points are so densely packed on the sphere that there is no empty region. However, the requisite number of data points is huge. The 'surface area' of $S^3$ is $2\pi^2$ [Kendall 61] while a region 30 degrees wide about a point has area $\pi/6$. Thus, theoretically only about $12\pi \approx 38$ points are needed to cover the sphere, leaving no empty region.\(^8\) However, this requires a gap of at least 60 degrees between consecutive quaternions, a very large change of orientation. The typical gap between consecutive quaternions for reasonable motion control is about 20 degrees. Using a gap of 20 degrees, 434,783 points\(^9\) are needed to cover $S^2$, and even more to cover $S^3$. Moreover, in order to leave no empty region, the quaternions must be spread across the entire sphere. It is far more common for the data to be restricted to a small region of the sphere. Thus, there will be an empty region on $S^3$ in all but the most pathological cases (over 500,000 quaternions uniformly scattered across $S^3$).

7.4 Finding an empty region

There are many ways to find a point in an empty region of $S^3$. We shall offer three methods, a very simple randomized method, a heuristic method, and an optimal method. Since there are so many empty regions on $S^3$, a random choice of point on $S^3$ is effective (where we continue to choose a point on $S^3$ at random until a point in an empty region is found). This is an extremely simple method: the only care that must be taken is with repeatability and coordinate-frame invariance. If given the same data twice, we would like the same quaternion spline to be constructed, so we would like to choose the same point in an empty region to rotate to the pole. Fortunately, we can take advantage of the predictability of pseudo-random number generators. If we use a pseudo-random number generator with the same seed, such as the C/C++ 'rand' function, exactly the same sequence of 'random' points will be generated (see Remark 7.3) each time it is called, leading to the same quaternion spline if the method is repeated.

**Remark 7.3** To find a random point on $S^3$, we find four random numbers in $[-1,1]$ defining a 4-vector, and then normalize this vector. A random number in $[-1,1]$ can be generated in C using the 'rand' function as follows: \((\text{float}) \ (\text{rand}() \ \% \ 32767) / 16383 - 1.0.\)

To make the method coordinate-frame invariant, the data is first rotated into a canonical frame (before the random choice of empty region) as follows: $p_0$ is rotated to the pole, then $p_1$ is rotated to the $\{x_2 = 0, x_3 = 0\}$ plane (without moving $p_0$), and finally $p_2$ is rotated to the $\{x_3 = 0\}$ hyperplane (without moving $p_0$ or $p_1$).\(^{10}\) Of course, this rotation is also reversed at the end.

\(^7\)Rotation of a rational Bezier curve can be achieved simply by rotation of its control polygon.

\(^8\)Actually, quite a few more points are needed, since the discs about the 38 points do not abut perfectly.

\(^9\)The surface area of $S^2$ is $4\pi$, while the surface area of a region on $S^2$ 10 degrees wide centered about a point is $\int_{-\pi}^{\pi} \int_{0}^{\pi} \sin \phi \ d\phi \ d\theta = 0.000089$ [Lang 79].

\(^{10}\)Only 3 points are necessary to define a unique frame of reference, since the added degree of freedom is locked down through the points lying on $S^3$. 

15
There is also an heuristic approach to finding a point of an empty region, which is not guaranteed to find a point but usually works well. This will find not just any empty point, but usually a very good empty point. Rather than directly looking for an empty region, we try to move the data far from the pole by moving the best-fitting plane of the data as far as possible from the pole. Since the best-fitting plane is a good representative of most of the points, the points tend to move far away from the pole. However, this method is not guaranteed since outliers may still be moved close to the pole. It is efficient, since the best-fitting plane can be computed quickly from the covariance matrix of the data [Ballard 82]. This method is equivalent to choosing the normal of the best-fitting plane as the empty point.

Finally, one could find the optimal empty point: the point in the middle of the largest empty region which will move the data as far away as possible from the pole. This is a generalization of the classical largest empty circle problem for points in a plane [Preparata 85], and a similar technique can be used. Let \( f(p) \) be the distance of \( p \in S^3 \) from the nearest quaternion (measuring distance on the surface). The maximum of \( f(p) \) is attained at some vertex of the Voronoi diagram of the quaternions on \( S^3 \). This is the desired optimal empty point. The Voronoi diagram must be built on a surface, which is itself an interesting problem. Luckily, only the Voronoi vertices need to be computed, not the entire Voronoi diagram. A Voronoi diagram is built out of point bisectors, and the bisector of two points on \( S^3 \) is a great circle of \( S^3 \). A simplistic approach is to compute a superset of the Voronoi vertices by intersecting all bisectors of two quaternions, and then choose the one furthest from all quaternions.

Some discussion is necessary about the issue of coordinate-frame invariance. The observant reader may have noticed that we have established coordinate-frame invariance for our quaternion splines on \( S^3 \), but should actually be more interested in coordinate-frame invariance of the objects in 3-space that are undergoing the motion. Fortunately, these invariances are equivalent, due to the following lemma.

**Lemma 7.4** Rotation of an object in 3-space by a constant amount is equivalent to rotation of the associated quaternion on \( S^3 \) by a constant amount.

**Proof:** Let the original orientation of the object and the amount of rotation be represented by the quaternions \( q = (q_1, q_2, q_3, q_4) \) and \( c = (c_1, c_2, c_3, c_4) \), respectively. Then the orientation of the object after the rotation is \( cq \), so the associated quaternion of the object has changed from \( q \) to \( cq \). Using the formula for quaternion multiplication (2), we notice that \( cq \) can also be interpreted as the rotation of \( q \) on \( S^3 \) by a constant amount (depending on \( c \)):

\[
c \ast q = \begin{pmatrix}
c_1 & -c_2 & -c_3 & -c_4 \\
c_2 & c_1 & -c_4 & c_3 \\
c_3 & c_4 & c_1 & -c_2 \\
c_4 & -c_3 & c_2 & c_1
\end{pmatrix}
\begin{pmatrix}
q_1 \\
q_2 \\
q_3 \\
q_4
\end{pmatrix}
\]

This matrix is a rotation matrix in 4-space. \( \blacksquare \)

Finally, note that the maps \( M \) and \( M^{-1} \) are not affine invariant. Thus, in order to make our quaternion spline construction coordinate-frame invariant, a rotation to a canonical frame is necessary as a preprocessing step, regardless of whether we want to rotate away from the pole or not. In other words, our rotation of the data serves a dual purpose: avoiding the pole and making the quaternion spline construction coordinate-frame invariant.
Figure 10: Examples

8 Examples

To consolidate our understanding of the algorithm before moving on to its analysis, we will now provide some examples. We can visualize quaternion splines, which reside in 4-space, using the following trick. Points of $x_3 = 0$ are mapped by $M^{-1}$ to points of $x_2 = 0$ in Euclidean space, which are then mapped back by $M$ to points of $x_3 = 0$ on $S^3$ again. Therefore, we can visualize quaternion splines in 3 dimensions using data samples with $x_3 = 0$. We only apply this restriction to $x_3 = 0$ to the examples that we want to visualize in this paper. The quaternion splines that we analyze in Section 9 reside in 4-space, with fully general quaternion data. Figure 10 shows six rational quaternion splines designed using the method of this paper (on input of 5, 5, 6, 10, 66, and 100 quaternions, respectively). Figure 11 illustrates the coordinate-frame invariance of the method: rotated data sets generate the same curve, rotated.

9 Analysis

In this section, we consider the quality of our rational quaternion spline and compare it to other quaternion splines. [Barr 92] proposes that covariant acceleration be used as a measure of the
quality of a quaternion spline: the smaller the covariant acceleration, the more desirable the curve.

**Definition 9.1** The **covariant acceleration** of a curve segment \(c(t), t \in I\), is \(\int_I \|c''(t)\| c(t)\| dt\), where \(a \backslash b = a - \left(\frac{\partial a}{\partial b}\right)b\).

Their motivation is as follows. The acceleration of a curve on a surface can be decomposed into normal and covariant components: normal acceleration is necessary as it keeps the curve on the surface, but covariant acceleration is not. Low covariant acceleration can also be motivated by appealing to the design of short quaternion splines, as discussed in Section 5. The shortest curve through a set of points is necessarily a geodesic, which has zero covariant acceleration [Thorpe 79].

We will use covariant acceleration as a measure of curve quality. One should note that covariant acceleration is only a guide to quality and should not be viewed as an absolute measure. For example, geodesics with their minimal covariant acceleration are not necessarily desirable as quaternion splines, as the rigidity of their perfection can force them into strange shapes not conducive to motion control. The best judge of the quality of a quaternion spline is the motion that it generates. We have found that our quaternion splines generate very natural motions, and have shown their effectiveness for animation in [Johnstone and Williams 95].

### 9.1 Choice of point on \(M^{-1}(p_i)\)

We first explore the advantage of the flexibility to choose any points on the lines \(M^{-1}(p_i)\). We compare the following strategies for choosing the point on \(M^{-1}(p_i)\): the closest point (the method outlined in Section 5 and the preferred method), the intersection of \(M^{-1}(p_i)\) with \(S^3\), and the defining point of \(M^{-1}(p_i)\), which is basically an arbitrary choice of point on the line in Euclidean space. We find that these choices are progressively worse, as expected. All of these data sets have 100 points.
<table>
<thead>
<tr>
<th>Data set</th>
<th>Covariant acceleration of our curve</th>
<th>Improvement factor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Closest</td>
<td>On $S^4$</td>
</tr>
<tr>
<td>1</td>
<td>23.9</td>
<td>394</td>
</tr>
<tr>
<td>2</td>
<td>30.0</td>
<td>166</td>
</tr>
<tr>
<td>3</td>
<td>14.0</td>
<td>31.5</td>
</tr>
</tbody>
</table>

Table 1: Curve quality with various choices of point on $M^{-1}(p_i)$

<table>
<thead>
<tr>
<th>Data set</th>
<th>Covariant acceleration</th>
<th>Improvement factor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Our curve</td>
<td>Trim curve</td>
</tr>
<tr>
<td>1</td>
<td>23.9</td>
<td>764</td>
</tr>
<tr>
<td>2</td>
<td>25.0</td>
<td>457</td>
</tr>
<tr>
<td>3</td>
<td>41.3</td>
<td>352</td>
</tr>
<tr>
<td>4</td>
<td>15.9</td>
<td>90.6</td>
</tr>
<tr>
<td>5</td>
<td>14.8</td>
<td>42.4</td>
</tr>
</tbody>
</table>

Table 2: A comparison with trim curves

9.2 Trim curves

We next compare our rational quaternion spline with trim curves. Recall from Section 1 that trim curves also use modeling in Euclidean space, in this case parameter space, but use the surface parameterization as the map to the surface. This comparison should capture the advantage of using a well-designed map to the surface, as well as the advantage of flexibility in choosing an inverse point along a line. Typical results from these tests are presented in Table 9.1, ordered by the amount of improvement. These data sets, like all of the others we use in this section, are generated randomly, with restrictions on distance between consecutive quaternions. This shows the consistent improvement when using the new approach, usually about an order of magnitude.

9.3 Park and Ravani's method

We now compare to a nonrational quaternion spline. It is meaningless to compare with Barr et al.'s methods [Barr 92, Rama 97] since they explicitly minimize covariant acceleration. Therefore, we choose to compare to another good nonrational spline, Park and Ravani's method [Park 97]. The results are in Table 9.3.

9.4 Execution times

As one would expect from its reliance on classical NURBS technology and rational computations, our method is very efficient. Table 9.4 gives some typical results, running on a 225MHz SGI Octane workstation. The first six results are for the examples in Figure 10. Times of 0 are possible since the

<table>
<thead>
<tr>
<th>Data set</th>
<th>Covariant acceleration</th>
<th>Improvement factor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Our curve</td>
<td>Lie curve</td>
</tr>
<tr>
<td>1</td>
<td>0.364</td>
<td>652</td>
</tr>
<tr>
<td>2</td>
<td>0.643</td>
<td>555</td>
</tr>
<tr>
<td>3</td>
<td>13.3</td>
<td>645</td>
</tr>
</tbody>
</table>

Table 3: A comparison with Park and Ravani
<table>
<thead>
<tr>
<th>Data set</th>
<th># of data points</th>
<th>Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>0.00</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>0.00</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>0.00</td>
</tr>
<tr>
<td>5</td>
<td>66</td>
<td>0.03</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>0.03</td>
</tr>
<tr>
<td>7</td>
<td>200</td>
<td>0.12</td>
</tr>
<tr>
<td>8</td>
<td>500</td>
<td>0.27</td>
</tr>
</tbody>
</table>

Table 4: Execution times

profiler we use measures in intervals of 30 microseconds. Clearly, curve design is entirely interactive and responsive.

10 Conclusions and future work

We have developed a rational quaternion spline of high quality. It is a B-spline of arbitrary continuity, and is easily built using the classical construction of interpolating B-splines in Euclidean space. The dependence of the method on existing B-spline technology leads to great efficiency in both the implementation and the execution of the method, which can promote interactive control for animation or robot motion. Previous quaternion splines have required new tools (e.g., sleeping, constrained optimization, biarcs) that prevent them from being incorporated into existing NURBS modelers.

We believe that there is additional promise in the Euclidean space approach for curve design on surfaces. In particular, there are two forms of inherent flexibility in the method that deserve further study: map flexibility (the choice of map to the surface) and interpolation flexibility (the choice of point on the inverse image curve during Euclidean space interpolation). In the future, we also want to explore a direct interpolation of the set of inverse image curves by a curve. A full understanding of the relationship between curves in Euclidean and Riemannian space is also an interesting challenge.

References


