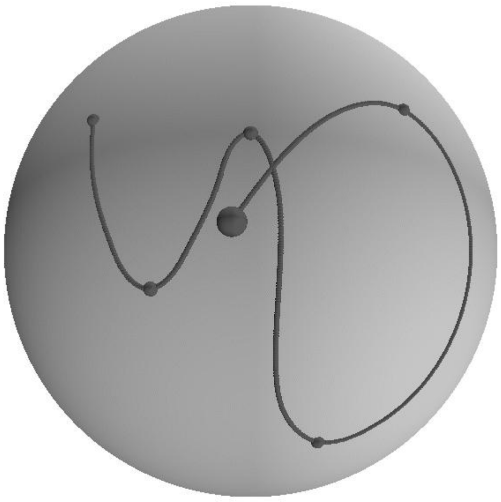
Quaternions, Interpolation and Animation

Erik B. Dam erikdam@diku.dk
Martin Koch myth@diku.dk
Martin Lillholm grumse@diku.dk

Technical Report DIKU-TR-98/5
Department of Computer Science
University of Copenhagen
Universitetsparken 1
DK-2100 Kbh Ø
Denmark

July 17, 1998



Abstract

The main topics of this technical report are quaternions, their mathematical properties, and how they can be used to rotate objects. We introduce quaternion mathematics and discuss why quaternions are a better choice for implementing rotation than the well-known matrix implementations. We then treat different methods for interpolation between series of rotations. During this treatment we give complete proofs for the correctness of the important interpolation methods *Slerp* and *Squad*. Inspired by our treatment of the different interpolation methods we develop our own interpolation method called *Spring* based on a set of objective constraints for an optimal interpolation curve. This results in a set of differential equations, whose analytical solution meets these constraints. Unfortunately, the set of differential equations cannot be solved analytically. As an alternative we propose a numerical solution for the differential equations. The different interpolation methods are visualized and commented. Finally we provide a thorough comparison of the two most convincing methods (*Spring* and *Squad*). Thereby, this report provides a comprehensive treatment of quaternions, rotation with quaternions, and interpolation curves for series of rotations.

Contents

Chapter 1

Introduction

To animate means to "bring to life." Animation is a visual presentation of change. Traditionally this has been used in the entertainment business, for example Donald Duck moving in a cartoon. More serious applications have later been developed for physics (visualization of particle systems) and chemistry (displaying molecules).

This paper treats a small part of the world of animation — animation of rotation. As a background for the following chapters, we will in this section give an overview of how animation was done traditionally (i. e. before the computer), and how it is done now. The presentation is based on [Foley et al., 1990] and [Lasseter, 1987].

An animation is based on a story — a manuscript. The manuscript is used to make a *storyboard*, in which it is decided how to split the story into individual *scenes*. For each scene a sketch is made with some text describing the scene. Based on the storyboard, a series of *key frames* is produced showing the characters of the cartoon in key positions. The frames between the key frames can then be made from these key positions. Traditionally, the most experienced artists produced the key frames (and were therefore named *key framers*), leaving the frames in-between to the less experienced artists (who became known as *in-betweeners*). The animators produce a rough draft of the animation, which is presented at a *pencil test*. Once the draft is satisfactory, the final version is produced and transferred to celluloid.

This method of animation is called *key framing* and has since been used in computer animation systems. Already in 1968 animation of 3D models was known, and the idea of using computers for key frame animation was used in 1971 [Burtnyk & Wein, 1971].

Computers are natural replacements for the in-betweeners. Given two key frames, the frames in-between can be generated by interpolation. Admittedly there are several problems with this approach:

- A translation between two key frames can easily be obtained by simple linear interpolation. When the movement consists of more key frames it is necessary to use more advanced curves (for example *splines*) to produce a smooth movement across key frames.
- Ordinary physics cannot be used to describe how the eye perceives moving objects in a cartoon. Objects will change shape as they move: A ball will morph into an oval when bouncing fast (see figure 1.1). This will not happen automatically if a computer is used to animate the motion of the ball.
- Animation of rotational movement has also been attempted using key frames and interpolation. Rotation is more complex than translation, however. The problems involved in interpolating rotations will be treated in this paper.

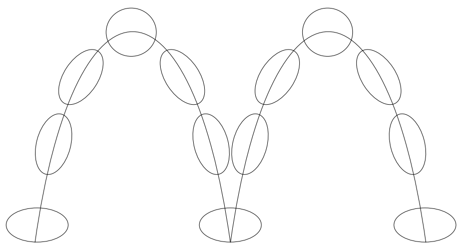


Figure 1.1: *The cartoon version of a bouncing ball.*

Computer animation consists of much more than pure motion. Apart from the problems of interpolation of the movement there are complicated issues concerning light, sound, colors, camera angles, camera motion, shadows, physical properties of the objects being modelled etc.

We limit this paper to treat methods for representation and implementation of rotation. The methods are mostly based on *quaternions*, a kind of four-dimensional complex numbers. Through a series of attempts to define “nice” rotation, we derive a mathematical description of rotation through a series of key frames.

We will not discuss the matters mentioned in the first two bullets above or the other aspects mentioned (light, sound etc.)

The main foundation for this paper is the articles [Shoemake, 1985], [Barr et al., 1992], and [Watt & Watt, 1992]. We do not require any knowledge of these articles. It will, however, be an advantage for the reader to be familiar with the common transformation methods using matrices and to have basic knowledge of interpolation curves in the plane (in particular *splines*). Some basic mathematics knowledge will also be advantageous (group theory, differentional calculus, calculus of variations and differential geometry).

Chapter 2

Geometric transformations

In this chapter we will briefly discuss selected transformations of objects in 3D. The key topic will be rotation but since interpolation between positions offers useful parallels to interpolation of rotation, we include translation. Note that these parallels serve only as inspiration for the rotational case — mainly because the space of translations is Euclidean while the space of rotations is not. This difference will be discussed in depth in the following chapters.

2.1 Translation

Translation is the most obvious kind of transformation: A point in space is moved from one position to another. Let a point $P \in \mathbb{R}^3$ be denoted by a 3-tuple (x, y, z), $x, y, z \in \mathbb{R}$ and the translation by a vector $(\Delta x, \Delta y, \Delta z)$. Then the new position P' is calculated by simple addition: $P' = (x + \Delta x, y + \Delta y, z + \Delta z)$. The definition is non-ambiguous i. e. there exists only one translation vector that takes P to P'.

2.2 Rotation

Rotation in 3D is not as simple as translation and it can be defined in many ways. We have chosen the following definition:

We will use the definition given by Euler's ($*$1707 – † 1783) theorem [Euler, 1752] — written in modern notation (compare with figure 2.1):

Proposition 1.
Let O, $O' \in \mathbb{R}^3$ be two orientations. Then there exists an axis $l \in \mathbb{R}^3$ and an angle of rotation $\theta \in \;]-\pi, \pi]$ such that O yields O' when rotated θ about l.

Note that the proposition states existence and does not state uniqueness.

We will distinguish between *orientations* and *rotations*. An orientation of an object in $\mathbb{R}^3$ is given by a normal vector. A rotation is defined by an axis and an angle of rotation.

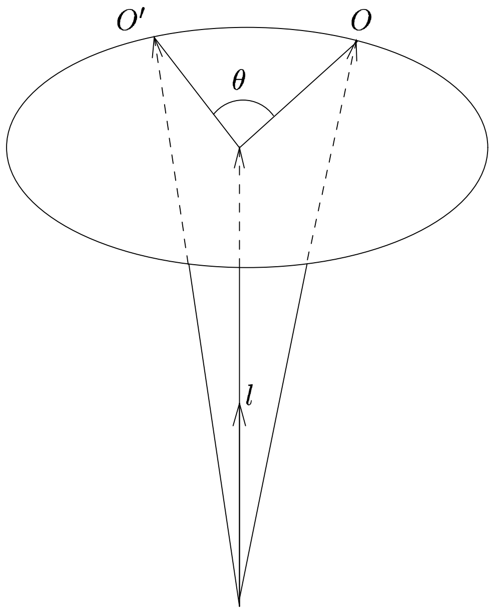


Figure 2.1: *Let O, $O' \in \mathbb{R}^3$ be two orientations. Then there exists an axis $l \in \mathbb{R}^3$ and an angle of rotation $\theta \in \ ]-\pi, \pi]$ such that O yields O' when rotated θ about l.*

Chapter 3

Two rotational modalities

Euler's theorem (proposition 1) gives a simple definition of rotations. In most of the literature, Euler *angles* are used to define rotation. From these two fundamental definitions, rotation can be discussed mathematically in numerous ways. We will term the combination of a definition and a corresponding mathematical representation a *rotational modality*. In this report we will discuss the following two modalities:

- Rotation defined by Euler angles represented by general transformation matrices.
- Rotation defined by Euler's theorem represented by quaternions.

The aim is of this chapter is to reach an implementation of a general rotation with each modality. A comparison of how the modalities implement a general rotation is given in chapter 4. Conversion between representations of rotation is discussed in appendix B. Finally, general conventions for rotation used in this report can be found in appendix A.

In sections 3.1 and 3.2, Euler angles and their matrix representation are described. The description is brief — the reader is assumed familiar with these topics.

Section 3.3 gives an in-depth treatment of quaternions starting of with the basics of quaternion mathematics. After the introduction, it is established how quaternions can be used to represent rotation as defined by Euler's theorem.

3.1 Euler angles

The space of orientations can be parameterized by Euler angles. When Euler angles are used, a general orientation is written as a series of rotations about three mutually orthogonal axes in space. Usually the x, y, and z axes in a Cartesian coordinate system are used. The rotations are often called *x-roll*, *y-roll* and *z-roll*.

Euler originally developed Euler angles as a tool for solving differential equations. Later Euler angles have become the most widely used method of parametererizing the space of orientations.

As we shall see below, this choice gives rise to a number of problems. If we choose to consider a rotation as the action performed to obtain a given orientation, Euler angles can be used to parameterize the space of rotations. To describe a general rotation as described in section 2.2, three Euler angles $(\theta_1, \theta_2, \theta_3)$ are required, where θ_1, θ_2, and θ_3 are the rotation angles about the x, y, and z axes, respectively.

The conversion from a general rotation to Euler angles is ambiguous since the same rotation can be obtained with different sets of Euler angles (see [Foley et al., 1990]). Furthermore, the resulting rotation depends on the order in which the three rolls are performed. This gives rise to further ambiguity but fits well with the fact that rotations in space do not generally commute (see appendix B). Some of the ambiguity in the conversion to Euler angles can be eliminated by adopting a convention of which order the rolls should be performed. In this paper, we use the convention described in appendix A. Introducing a convention does not, however, eliminate the ambiguity altogether (see chapter 4).

3.2 Rotation matrices

Rotation matrices are the typical choice for implementing Euler angles. For each type of roll, there is a corresponding rotation matrix, i. e. an x rotation matrix, a y rotation matrix, and a z rotation matrix. The matrices rotate by multiplying them to the position vector for a point in space, and the result is the position vector for the rotated point. A rotation matrix is a 3×3 matrix, but usually homogeneous 4×4 matrices are used instead (see [Foley et al., 1990] for further detail). A general rotation is obtained by multiplying the three roll-matrices corresponding to the three Euler angles. The resulting matrix embodies the general rotation and can be applied to the points that are to be rotated.

The three standard rotation matrices are given in homogeneous coordinates in appendix B.

Matrix multiplication is not generally commutative. This fits well with the fact that rotations in space do not commute.

Finally it should be noted that using homogeneous transformation matrices gives the only implementation that effectively embodies all standard transformations: Translation, scaling, shearing, and various projection transformations.

3.3 Quaternions

The second rotational modality is rotation defined by Euler's theorem and implemented with quaternions. Since quaternions are not nearly as well-known as transformation matrices, and since no good overview of the field exists, we will give a historical overview and then provide a thorough treatment of quaternion mathematics.

3.3.1 Historical background

Quaternions were invented by Sir William Rowan Hamilton (∗1809 – † 1865) in 1843. Hamilton's aim was to generalize complex numbers to three dimensions, i. e. numbers of the form $a+\mathbf{i}b+\mathbf{j}c$, where $a,b,c \in \mathbb{R}$ and $\mathbf{i}^2 = \mathbf{j}^2 = -1$. Hamilton never succeeded in making this generalization, and it has later been proven that the set of three-dimensional numbers is not closed under multiplication. In 1966 Kenneth O. May gave the following elegant proof of this:

Proposition 2.
The set of three-dimensional complex numbers is not closed under multiplication.

Proof (freely adopted from Kenneth O. May 1966):
Assume that the usual rules of arithmetic for complex numbers hold, and that $\mathbf{i}^2 = \mathbf{j}^2 = -1$.

The proof is by contradiction, so we assume that a closed multiplication exists. Since multiplication is closed, there exist $a,b,c \in R$ that satisfy $\mathbf{ij} = a + \mathbf{i}b + \mathbf{j}c$. Multiplying this with $\mathbf{i}$ yields $-\mathbf{j} = -b + \mathbf{i}a + \mathbf{ij}c$. Substituting the first equation in the second equation yields $-\mathbf{j} = -b + \mathbf{i}a + (a + \mathbf{i}b + \mathbf{j}c)c$, i. e. $0 = (ac-b) + \mathbf{i}(a+bc) + \mathbf{j}(c^2+1)$. Thus $ac-b=0$, $a+bc=0$ and $c^2+1=0$. The equation $c^2+1=0$ gives the contradiction, since c is real by assumption.
$\square$

One of Hamilton's motivations for seeking three-dimensional complex numbers was to find a description of rotation in space corresponding to the complex numbers, where a multiplication corresponds to a rotation and a scaling in the plane.

While walking by the Royal Canal in Dublin on a Monday in October 1843, Hamilton realized that *four* numbers are needed to describe a rotation followed by a scaling. One number describes the size of the scaling, one the number of degrees to be rotated, and the last two numbers give the plane[1] in which the vector should be rotated. After this insight, Hamilton found a closed multiplication for four-dimensional complex numbers of the form $\mathbf{i}x + \mathbf{j}y + \mathbf{k}z$, where $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$. Hamilton dubbed his four-dimensional complex numbers *quaternions*. The parallel to ordinary complex numbers stems from the imaginary parts.

A quaternion is usually written $[s, \mathbf{v}], s \in \mathbb{R}, \mathbf{v} \in \mathbb{R}^3$. Here s is called the *scalar part*, and $\mathbf{v} = (x,y,z)$ is the *vector part*.

[1]The xy plane can be rotated to any plane in xyz space through the origin by giving the rotation angles about the x and y axes.

Historical aside

Hamilton presented quaternion mathematics at a series of lectures at the Royal Irish Academy. The lectures gave rise to a book [Hamilton, 1853], the full title of which (with typography as in the book) is:

> LECTURES ON QUATERNIONS: CONTAINING A SYSTEMATIC STATEMENT OF
>
> A New Mathematical Method
>
> OF WHICH THE PRINCIPLES WERE COMMUNICATED IN 1843 TO THE ROYAL IRISH ACADEMY; AND WHICH HAS SINCE FORMED THE SUBJECT OF SUCCESSIVE COURSES OF LECTURES, DELIVERED IN 1848 AND SUBSEQUENT YEARS IN THE HALLS OF TRINITY COLLEGE, DUBLIN: WITH NUMEROUS ILLUSTRATIVE DIAGRAMS, AND WITH SOME GEOMETRICAL AND PHYSICAL APPLICATIONS.

In the book (page 271) Hamilton writes (again imitating the book's typography):

285. We know then how to interpret in two apparently different ways, which are, however, easily perceived to have an essential connection with each other, the following SYMBOL OF OPERATION,

$$q()q^{-1};$$

where q may be called (as before) the *operator quaternion* while the symbol (suppose r) of the *operand quaternion* is conceived to occupy the place marked by the parentheses. For we may either consider the effect of the operation, thus symbolized, to be (as in 282, 283) a *conical rotation of the axis of the operand round the axis of the operator, through double the angle thereof*, in such a manner as to *transport the vertex of the representative angle* of the operand *to a new position* on the unit sphere, without changing the *magnitude* of that angle, nor the *tensor*[2] of the quaternion thus operated on: or else, at pleasure, may regard (by 285) the operation as causing one extremity of the *representative arc* of the same *operand* (r) to *slide along the doubled arc* of the same *operator* (q), without any change in the *length* of the arc so sliding, nor of its *inclination* to the great circle along which its extremity thus slides.

The historically interested reader is referred to [Hamilton, 1899] and [Hallenberg et al., 1993] (only available in Danish).

3.3.2 Basic quaternion mathematics

In this section we will state the notation used for quaternions and establish quaternion mathematics including addition, multiplication, subtraction, and multiplication with a scalar. Finally we define the conjugate and the inverse of a quaternion.

[2]In modern usage = norm.

Notation

We use $\equiv$ to mean equal by definition. Closed intervals on the real line are denoted by $[a,b] \equiv \{\ x \mid a \leq x \leq b, a, b, x \in \mathbb{R}\}$. A semi-open interval is, for example, denoted by $]a,b] \equiv \{\ x \mid a < x \leq b, a, b, x \in \mathbb{R}\}$. The set of n times differentiable functions from A to B with continuous derivatives we denote $C^n(A,B)$.

Definition 1.
The set of quaternions is denoted H.

Quaternions consist of a scalar part $s \in \mathbb{R}$ and a vector part $\mathbf{v} = (x,y,z) \in \mathbb{R}^3$. We will use the following forms:

Definition 2.
Let $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1, \mathbf{ij} = \mathbf{k}$ and $\mathbf{ji} = -\mathbf{k}$. Then $q \in H$ can be written:

$$\begin{aligned} q \equiv\ & [s, \mathbf{v}] && , \ s \in \mathbb{R},\ \mathbf{v} \in \mathbb{R}^3 \\ \equiv\ & [s, (x,y,z)] && , \ s,x,y,z \in \mathbb{R} \\ \equiv\ & s + \mathbf{i}x + \mathbf{j}y + \mathbf{k}z && , \ s,x,y,z \in \mathbb{R} \end{aligned}$$

We will identify the set of quaternions $\{[s, \mathbf{0}] \mid s \in \mathbb{R}\}$ with $\mathbb{R}$ and the set $\{[0, \mathbf{v}] \mid \mathbf{v} \in \mathbb{R}^3\}$ with $\mathbb{R}^3$.

Definition 3.
Let $q, q' \in H$ where $q = [s,(x,y,z)]$ and $q' = [s',(x',y',z')]$. The addition operator, $+$, is defined

$$q + q' \equiv [s,\mathbf{v}] + [s',\mathbf{v}'] \equiv [s,(x,y,z)] + [s',(x',y',z')] \equiv (s + \mathbf{i}x + \mathbf{j}y + \mathbf{k}z) + (s' + \mathbf{i}x' + \mathbf{j}y' + \mathbf{k}z')$$

Proposition 3. (Quaternion addition)
Let $q, q' \in H$, where $q = [s,\mathbf{v}]$ and $q' = [s',\mathbf{v}']$. Then $q + q' = [s + s', \mathbf{v} + \mathbf{v}']$

Proof of proposition 3

$$\begin{aligned} q + q' &\equiv [s,\mathbf{v}] + [s',\mathbf{v}'] \\ &\equiv (s + \mathbf{i}x + \mathbf{j}y + \mathbf{k}z) + (s' + \mathbf{i}x' + \mathbf{j}y' + \mathbf{k}z') \\ &= (s + s') + \mathbf{i}(x + x') + \mathbf{j}(y + y') + \mathbf{k}(z + z') \\ &\equiv [s + s', \mathbf{v} + \mathbf{v}'] \end{aligned}$$

□

Definition 4.
Let $q, q' \in H$ where $q = s + \mathbf{i}x + \mathbf{j}y + \mathbf{k}z$ and $q' = s' + \mathbf{i}x' + \mathbf{j}y' + \mathbf{k}z'$. Multiplication is defined

$$qq' \equiv [s,\mathbf{v}][s',\mathbf{v}'] \equiv [s,(x,y,z)][s',(x',y',z')] \equiv (s + \mathbf{i}x + \mathbf{j}y + \mathbf{k}z)(s' + \mathbf{i}x' + \mathbf{j}y' + \mathbf{k}z')$$

Proposition 4. (Multiplication of quaternions)
Let $q, q' \in H$, where $q = [s,\mathbf{v}]$ and $q' = [s',\mathbf{v}']$. Then $qq' = [ss' - \mathbf{v} \cdot \mathbf{v}', \mathbf{v} \times \mathbf{v}' + s\mathbf{v}' + s'\mathbf{v}]$, where $\cdot$ and $\times$ denote the scalar and vector product in $\mathbb{R}^3$, respectively.

Proof of proposition 4
From definition 2 the following identities can be obtained from simple algebra: $\mathbf{jk} = \mathbf{i}, \mathbf{kj} = -\mathbf{i}, \mathbf{ik} = -\mathbf{j}$ and $\mathbf{ki} = \mathbf{j}$. These identities are used in:

$$\begin{aligned}
qq' &\equiv [s, \mathbf{v}][s', \mathbf{v}'] \\
&\equiv (s + \mathbf{i}x + \mathbf{j}y + \mathbf{k}z)(s' + \mathbf{i}x' + \mathbf{j}y' + \mathbf{k}z') \\
&= ss' - (xx' + yy' + zz') + \mathbf{i}(sx' + s'x + yz' - zy') + \\
&\quad \mathbf{j}(sy' + s'y + zx' - xz') + \mathbf{k}(sz' + s'z + xy' - yx') \\
&\equiv [ss' - \mathbf{v} \cdot \mathbf{v}', \mathbf{v} \times \mathbf{v}' + s\mathbf{v}' + s'\mathbf{v}]
\end{aligned}$$

□

Corollary 1. (to proposition 4)
Quaternion multiplication is not generally commutative.

Proof of proposition 1
We give a counter-example: $\mathbf{ij} = \mathbf{k}$, but $\mathbf{ji} = -\mathbf{k}$.

□

Below we will give a number of propositions without proof. The proofs are all based on the principle used above: The constituent quaternions are written $s + \mathbf{i}x + \mathbf{j}y + \mathbf{k}z$. Then, using simple algebra and collection of terms, the result can be written as a quaternion using definition 2.

We state the following properties of quaternion multiplication:

Proposition 5.
Let $p, q, q' \in H$ and $r \in \mathbb{R}$. Then:

$$\begin{aligned}
(pq)q' &= p(qq') && (\textit{Quaternion multiplication is associative.}) \\
p(q + q') &= pq + pq' && (\textit{Quaternion multiplication distributes} \\
(q + q')p &= qp + q'p && \textit{across addition.})
\end{aligned}$$

Multiplying quaternions by a scalar is most easily introduced by identifying $r \in \mathbb{R}$ with the quaternion $[r, \mathbf{0}]$:

Definition 5.
Let $q \in H$ and $r \in \mathbb{R}$. Multiplication by a scalar is defined

$$rq \equiv [r, \mathbf{0}]q$$

Proposition 6. (Multiplication with a scalar)
Let $q \in H$, where $q = [s, \mathbf{v}]$ and let $r \in \mathbb{R}$. Then $rq = qr = [r, \mathbf{0}][s, \mathbf{v}] = [rs, r\mathbf{v}]$.

Note that the proposition gives that multiplication with a scalar is commutative.

We will use the notation $\dfrac{q}{r}$ to mean $\dfrac{1}{r}q$, where $q \in H$ and $r \in \mathbb{R}$.

We can now introduce subtraction in the usual manner:

Definition 6.
Given $q, q' \in H$, subtraction is defined $q - q' \equiv q + (-1)q'$

The definition gives the expected:

Proposition 7. (Quaternion subtraction)
Let $q, q' \in H$, where $q = [s, \mathbf{v}]$ and $q' = [s', \mathbf{v}']$. Then $q - q' = q + (-1)q' = [s - s', \mathbf{v} - \mathbf{v}']$.

Corresponding to the definition of the conjugate of a complex number, we define the conjugate of a quaternion:

Definition 7.
Let $q \in H$. Then q^* is called the conjugate of q and is defined by $q^* \equiv [s, \mathbf{v}]^* \equiv [s, -\mathbf{v}]$.

The definition gives rise to the following properties:

Proposition 8.
Let $p, q \in H$. Then:

i) $(q^*)^* = q$ *ii)* $(pq)^* = q^*p^*$ *iii)* $(p + q)^* = p^* + q^*$ *iv)* $qq^* = q^*q$.

The norm of a quaternion is obtained using conjugation:

Definition 8.
Let $p \in H$ and let the mapping $\|\cdot\| : H \curvearrowright \mathbb{R}$ be defined by $\|q\| \equiv \sqrt{qq^*}$. This mapping is called the norm and $\|q\|$ is the norm of q.

That this mapping is a norm in the usual sense is shown in the corollary to proposition 9. The norm mapping has a number of interesting properties that are summarized in:

Proposition 9.
Let $q, q' \in H$ and let $\|\cdot\| : H \curvearrowright \mathbb{R}$ be given as is definition 8. The following equations hold:

$$\begin{aligned} \|q\| &= \sqrt{s^2 + \mathbf{v} \cdot \mathbf{v}} = \sqrt{s^2 + x^2 + y^2 + z^2} && (3.1)\\ \|q^*\| &= \|q\| && (3.2)\\ \|qq'\| &= \|q\|\|q'\| && (3.3) \end{aligned}$$

Proof of proposition 9
The equations 3.1 and 3.2 can be seen directly. Equation 3.3 follows from:

$$\|qq'\| = \sqrt{qq'(qq')^*} = \sqrt{qq'q'^*q^*} = \sqrt{q\|q'\|^2q^*} = \sqrt{qq^*\|q'\|^2} = \sqrt{\|q\|^2\|q'\|^2} = \|q\|\|q'\|$$

□

We will later need the inner product of two quaternions. We also want to show that the norm mapping is indeed a norm in the usual mathematical sense. From equation 3.1 in proposition 9 it follows that the norm of a quaternion q can be written as it is usually obtained from the inner product (if $q \in H$ is identified with the corresponding vector in $\mathbb{R}^4$). This property is formalized by:

Definition 9.
Let $q, q' \in H$, $q = [s, \mathbf{v}] = [s, (x, y, z)]$, $q' = [s', \mathbf{v}'] = [s', (x', y', z')]$. The inner product is defined as $\bullet : H \times H \curvearrowright \mathbb{R}$ where $q \bullet q' = ss' + \mathbf{v} \cdot \mathbf{v}' = ss' + xx' + yy' + zz'$.

Note that the definition yields $q \bullet q = s^2 + x^2 + y^2 + z^2$, which gives rise to:

Corollary 2. (to proposition 9)
The norm of a quaternion q can be obtained by $\|q\| = \sqrt{q \bullet q}$. Furthermore, $\| \cdot \|$ is a norm in the usual mathematical sense.

Proof of corollary 2
That $\bullet$ computes the norm squared follows directly from proposition 9 and definition 9. Now let $q = [s, (x, y, z)] \in H$. If we identify q with $(s, x, y, z) \in \mathbb{R}^4$, the above method of computing the norm is identical to the usual Euclidean norm on $\mathbb{R}^4$. Thus the quaternion norm is a norm in the usual sense.

$\square$

We will later need the following generalization of the two- and three-dimensional cases:

Proposition 10.
Let $q, q' \in H$. Define q, q' as the corresponding four-dimensional vectors and let α be the angle between them. Then $q \bullet q' = \|q\| \|q'\| \cos \alpha$.

3.3.3 The algebraic properties of quaternions.

In this section we prove that the set of quaternions $H \setminus \{[0, (0, 0, 0)]\}$ is a non-Abelian group under quaternion multiplication. At the end of the section we give a summary of some other algebraic properties of quaternions.

Definition 10.
The set of quaternions $H \setminus \{[0, (0, 0, 0)]\}$ is written $\overset{\circ}{H}$

We will base the discussion on the definition of a group:

Definition 11.
Let G be a set with an operator $\cdot : G \times G \curvearrowright G$ defined by $(a, b) \to a \cdot b \equiv ab$. G is a group if

i)	*$a(bc) = (ab)c$ for all $a, b, c \in G$*	*(The operator is associative)*
ii)	*Exactly one $I \in G$ exists such that $Ia = aI = a$ for all $a \in G$.*	*(I is the neutral element)*
iii)	*For every $a \in G$ there exists an element $a^{-1} \in G$, such that $aa^{-1} = a^{-1}a = I$.*	*(a^{-1} is the inverse element of a)*

If $ab = ba$ for all $a, b \in G$, G is called an Abelian or commutative group.

That there exists a neutral element and inverse elements in $\overset{\circ}{\mathrm{H}}$ under quaternion multiplication is shown in the following two lemmas:

Lemma 1.
The element $I = [1, \mathbf{0}] \in \overset{\circ}{H}$ is the unique neutral element under quaternion multiplication.

Proof of lemma 1
Let $q \in H$ be given. Proposition 6 gives $qI = Iq = [1s, 1\mathbf{v}] = [s, \mathbf{v}] = q$.

Thus I is a neutral element. I is also the only element that meets the requirements. To see this, assume that J also meets the requirements. Then $IJ = I$, because J is a neutral element. Furthermore, $IJ = J$, since I is a neutral element. This gives us that $I = IJ = J$, so $I = J$ is the only neutral element in $\overset{\circ}{\mathrm{H}}$.

$\square$

Lemma 2.
Let $q \in \overset{\circ}{H}$. Then there exists $q^{-1} \in H$ such that $qq^{-1} = q^{-1}q = I$. Furthermore q^{-1} is unique and given by:

$$q^{-1} = \frac{q^*}{\|q\|^2}$$

Proof of lemma 2
Let $q \in \overset{\circ}{\mathrm{H}}$ be given.

Uniqueness
Let both $p_1, p_2 \in H$ be inverse to q. That p_1 and p_2 are equal follows from

$$p_1 = p_1 I = p_1(qp_2) = (p_1 q)p_2 = Ip_2 = p_2$$

Existence
Let $p = \dfrac{q^*}{\|q\|^2}$. Then

$$\begin{aligned} qp &= q\frac{q^*}{\|q\|^2} = \frac{qq^*}{\|q\|^2} = \frac{\|q\|^2}{\|q\|^2} = 1 \equiv I \\ pq &= \frac{q^*}{\|q\|^2}q = \frac{q^*q}{\|q\|^2} = \frac{qq^*}{\|q\|^2} = \frac{\|q\|^2}{\|q\|^2} = 1 \equiv I \end{aligned}$$

Thus every quaternion in $\overset{\circ}{\mathrm{H}}$ has an inverse.

$\square$

We will write $\frac{p}{q}$ for pq^{-1}. Note that this is generally different from $q^{-1}p$ since quaternion multiplication is not commutative.

We can now state the following:

Proposition 11.
The set $\overset{\circ}{H}$ is a non-Abelian group under quaternion multiplication.

Proof of proposition 11
Note that the set of quaternions is closed under multiplication. This follows directly from Hamilton's definition. The first requirement from the definition of a group follows from proposition 5. The second and third requirements follow from lemmas 1 and 2. The group is not Abelian, since quaternion multiplication is not commutative.

$\square$

Other algebraic properties

The set of quaternions satisfy some other algebraic properties that are worth mentioning. These are given without further ado:

- The set of quaternions is an Abelian group $(H, +)$ under quaternion addition.
- The set of quaternions is a non-Abelian ring $(H, +, \cdot)$, where $+$ and $\cdot$ are quaternion addition and multiplication.

3.3.4 Unit quaternions

This section discusses a subset of the quaternion group — the set of unit quaternions.

Definition 12.
Let $q \in H$. If $\|q\| = 1$, then q is called a unit quaternion. We will use H_1 to denote the set of unit quaternions.

The set of unit quaternions constitutes a unit sphere in four-dimensional space. We shall later see that the set of unit quaternions play an important part in relation to general rotations. The following propositions lead to the important proposition 21. the following:

Proposition 12.
Let $q = [s, \mathbf{v}] \in H_1$. Then there exists $\mathbf{v}' \in \mathbb{R}^3$ and $\theta \in \ ]-\pi, \pi]$ such that $q = [\cos\theta, \mathbf{v}'\sin\theta]$.

Proof of proposition 12
If $q = [1, \mathbf{0}]$ we let $\theta = 0$ and $\mathbf{v}'$ can be freely chosen amongst unit vectors in $\mathbb{R}^3$.

If $q \neq [1, \mathbf{0}]$ we let $k = |\mathbf{v}|$ and $\mathbf{v}' = \frac{1}{k}\mathbf{v}$. Then $\mathbf{v} = k\mathbf{v}'$ where $\mathbf{v}'$ is a unit vector in $\mathbb{R}^3$. Since q is a unit quaternion, we get

$$1 = \|q\|^2 = s^2 + \mathbf{v} \cdot \mathbf{v} = s^2 + k^2\mathbf{v}' \cdot \mathbf{v}' = s^2 + k^2$$

The equation $s^2 + k^2 = 1$ describes a circle in the plane. Since a circle is also described by $\cos^2\theta + \sin^2\theta = 1$, there exists $\theta \in \ ]-\pi, \pi]$ such that $s = \cos\theta$ and $k = \sin\theta$. All in all we get the desired:

$$q = [s, \mathbf{v}] = [s, \mathbf{v}'k] = [\cos\theta, \mathbf{v}'\sin\theta]$$

□

Two important results for unit quaternions are given in:

Proposition 13.
Let $q, q' \in H_1$. The following two equations hold:

i) $\|qq'\| = 1$ *ii)* $q^{-1} = q^*$

Proof of proposition 13
i) $\|qq'\| = \|q\|\|q'\| = 1$, since $\|q\| = \|q'\| = 1$. (by equation 3.3 in proposition 9)
ii) $q^{-1} \equiv q^*/\|q\|^2 = q^*$, since $\|q\| = 1$.

□

The set of unit quaternions H_1 is obviously a subset of $\overset{\circ}{\mathrm{H}}$, but definition 13 and proposition 14 give that H_1 constitute a *subgroup* of $\overset{\circ}{\mathrm{H}}$.

Definition 13.
Let G be a group and $F \neq \varnothing$ be a subset of G. F is a subgroup of G if

i) For all $a, b \in F$: $ab \in F$ (F is closed)

ii) For all $a \in F$: $a^{-1} \in F$

Proposition 14.
The set H_1 of unit quaternions is a subgroup of the group $\overset{\circ}{H}$.

Proof of proposition 14
Let $q, q' \in H_1$. Proposition 13 gives that $\|qq'\| = 1$, i. e. that $qq' \in H_1$, and thus the first subgroup requirement is satisfied. Equation 3.2 in proposition 9 and proposition 13 give that

$$\|q^{-1}\| = \|q^*\| = \|q\| = 1$$

and thereby the second subgroup requirement $q^{-1} \in H_1$.

□

3.3.5 The exponential and logarithm functions

We will later need quaternion versions of the real exponential and logarithm functions. The definitions and a few consequences of them are given here (see [Pervin & Webb, 1992] for further detail).

Definition 14.
Let $q \in H_1$, where $q = [\cos\theta, \sin\theta\mathbf{v}]$ as in proposition 12. The logarithm function log *is defined*

$$\log q \equiv [0, \theta\mathbf{v}]$$

Note that $\log[1, (0,0,0)] = [0, (0,0,0)]$ as in the real case. Note also that $\log q$ is not in general a unit quaternion.

The exponential function is introduced by

Definition 15.
For a quaternion of the form $q = [0, \theta\mathbf{v}]$, $\theta \in \mathbb{R}$, $\mathbf{v} \in \mathbb{R}^3$, $|\mathbf{v}| = 1$, the exponential function exp *is defined by*

$$\exp q \equiv [\cos\theta, \sin\theta\mathbf{v}]$$

Note that the exponential and logarithm functions are mutually inverse, and that exp maps into H_1.

From the above definitions we can define exponentiation for $q \in H_1, t \in \mathbb{R}$:

Definition 16.
Let $q \in H_1, t \in \mathbb{R}$. Exponentiation q^t is defined by

$$q^t \equiv \exp(t \log q)$$

This gives rise to the following:

Proposition 15.
Let $q \in H_1, t \in \mathbb{R}$. Then $\log(q^t) = t \log q$.

Proof of proposition 15

$$\log(q^t) = \log(\exp(t \log q)) = t \log q$$

$\square$

The following rule from $\mathbb{R}$ also holds for unit quaternions:

Proposition 16.
Let $q \in H_1, q = [\cos\theta, \sin\theta\mathbf{v}]$ and $a, b \in \mathbb{R}$. Then

$$q^a q^b = q^{a+b}$$

Proof of proposition 16

$$\begin{aligned}
q^a q^b &= \exp(a \log q)\exp(b \log q) \\
&= \exp(a[0, \theta\mathbf{v}])\exp(b[0, \theta\mathbf{v}]) \\
&= [\cos a\theta, \mathbf{v}\sin a\theta][\cos b\theta, \mathbf{v}\sin b\theta] \\
&= [\cos a\theta\cos b\theta - \sin a\theta\sin b\theta(\mathbf{v}\cdot\mathbf{v}), \mathbf{v}\cos a\theta\sin b\theta + \mathbf{v}\cos b\theta\sin a\theta + (\mathbf{v}\times\mathbf{v})\sin a\theta\sin b\theta] \\
&= [\cos a\theta\cos b\theta - \sin a\theta\sin b\theta, \mathbf{v}(\cos a\theta\sin b\theta + \cos b\theta\sin a\theta)] \\
&= [\cos((a+b)\theta), \sin((a+b)\theta)\mathbf{v}] \\
&= \exp([0, (a+b)\theta\mathbf{v}]) \\
&= \exp((a+b)\log(q)) \\
&= q^{a+b}
\end{aligned}$$

$\square$

Another rule from the real numbers is $(p^a)^b = p^{ab}$. This rule also holds for unit quaternions:

Proposition 17.
Let $p \in H_1$ and $a, b \in \mathbb{R}$. Then $(p^a)^b = p^{ab}$

Proof of proposition 17

$$(p^a)^b = (\exp(a \log p))^b = \exp(b \log(\exp(a \log p))) = \exp(ba \log p) = p^{ab}$$

$\square$

One must be very careful when using exp and log as the corresponding real versions. For example, consider the following incorrect derivation, where p and q are unit quaternions.

$$pq = \exp(\log(pq)) = \exp(\log(p) + \log(q)) = \exp(\log(q) + \log(p)) = \exp(\log(q))\exp(\log(p)) = qp$$

This is inconsistent with the fact that quaternion multiplication is not commutative. The error lies in the second step where the rule ($\log pq = \log p + \log q$) is used — this rule does *not* hold for quaternions.

3.3.6 Rotation with quaternions

Hamilton sought to describe rotations in space, just as complex numbers describe rotations in the plane. That quaternions do, in fact, perform rotation, is shown in the following propositions (proposition 21 in particular).

Proposition 18.
Let $p \in H$, $p = [s, (x, y, z)] = [s, \mathbf{v}]$ and let $q \in \overset{\circ}{H}$. If $r \in \mathbb{R} \setminus \{0\}$ then $(rq)p(rq)^{-1} = qpq^{-1}$.

Proof of proposition 18
Let $r \in \mathbb{R} \setminus \{0\}$. The inverse of rq is $q^{-1}r^{-1}$. Since scalar multiplication is commutative we can write: $(rq)p(rq)^{-1} = rqpq^{-1}r^{-1} = qpq^{-1}rr^{-1} = qpq^{-1}$. Thus qpq^{-1} is unchanged if q is multiplied by any non-zero scalar. $\square$

In the propositions below, we will only consider unit quaternions, since results shown for H_1 generalize to all of $\overset{\circ}{\mathrm{H}}$ by proposition 18.

Proposition 19.
Let $q \in H_1$, $p = [s, \mathbf{v}] \in H$. Then $qpq^{-1} = p'$, where $p' = [s, \mathbf{v}']$ with $|\mathbf{v}| = |\mathbf{v}'|$.

Proof of proposition 19
Below we write $S(q)$ for the scalar part of q.

The proof consists of three steps. We first show $S(p') = S(p)$ for $p \in \{[s, \mathbf{0}] \mid s \in \mathbb{R}\}$ and then for $p \in \{[0, \mathbf{v}] \mid \mathbf{v} \in \mathbb{R}^3\}$. Finally these results are used to show the proposition for $p \in H$.

If p is a scalar represented as a quaternion, $S(p') = S(p)$ follows from simple algebra. Let $p = [s, \mathbf{0}]$, then:

$$qpq^{-1} = q[s, \mathbf{0}]q^{-1} = [s, \mathbf{0}]qq^{-1} = [s, \mathbf{0}]$$

We have used that multiplication with a scalar commutes (proposition 6).

Correspondingly, we will now show that the same result holds for a vector $\mathbf{v}$ represented as a quaternion $[0, \mathbf{v}]$.

The scalar part $S(q)$ of a quaternion q can be computed by $2S(q) = q + q^*$. We show the proposition for a quaternion with 0 in the scalar part $p = [0, \mathbf{v}]$:

$$\begin{aligned}
2S(qpq^{-1}) &= (qpq^{-1}) + (qpq^{-1})^* \\
&= (qpq^*) + (qpq^*)^* \\
&= qpq^* + qp^*q^* && \text{(Propositions 5 and 8)} \\
&= q(p+p^*)q^* && \text{(Proposition 5)} \\
&= q(2S(p))q^* \\
&= 2S(p) && \text{(The above result)} \\
&= 0 && \text{(Since } p = [0, \mathbf{v}]\text{)}
\end{aligned}$$

Now let $p \in H$, $p = [s, \mathbf{v}] = [s, \mathbf{0}] + [0, \mathbf{v}]$.

$$\begin{aligned}
qpq^{-1} &= q([s, \mathbf{0}] + [0, \mathbf{v}])q^{-1} \\
&= q[s, \mathbf{0}]q^{-1} + q[0, \mathbf{v}]q^{-1} && \text{(Proposition 5)} \\
&= [s, \mathbf{0}] + [0, \mathbf{v}'] && \text{(The two above results)} \\
&= [s, \mathbf{v}']
\end{aligned}$$

All in all $S(p') = S(p)$. Since $q \in H_1$, proposition 9, equation 3.3 gives $\|p'\| = \|qp'q^{-1}\| = \|q\|\|p\|\|q^{-1}\| = \|p\|$. Since s is unchanged, it must be the case that $|\mathbf{v}| = |\mathbf{v}'|$.

□

Corollary 3. (to proposition 19)
Let $q \in H_1$, $p = [a, b\mathbf{v}] \in H$ where $a, b \in \mathbb{R}$ and $\mathbf{v} \in \mathbb{R}^3$. If $q[a, \mathbf{v}]q^* = [a, \mathbf{v}']$, then $q[a, b\mathbf{v}]q^* = [a, b\mathbf{v}']$.

Proof of proposition 3

$$\begin{aligned}
qpq^* &= q[a, b\mathbf{v}]q^* \\
&= qb[\tfrac{a}{b}, \mathbf{v}]q^* \\
&= b[\tfrac{a}{b}, \mathbf{v}'] && \text{(Proposition 19)} \\
&= [a, b\mathbf{v}']
\end{aligned}$$

□

We will later need the following useful rule:

Proposition 20.
Let $q, p \in H_1, p = [\cos\theta, \sin\theta\mathbf{v}], t \in \mathbb{R}$. Then $qp^tq^* = (qpq^*)^t$.

Proof of proposition 20
By corollary 3 there exists $\mathbf{v}' \in \mathbb{R}^3$ such that $q[\cos\theta, \sin\theta\mathbf{v}]q^* = [\cos\theta, \sin\theta\mathbf{v}']$. We get

$$\begin{aligned}
qp^tq^* &= q(\exp(t\log p))q^* \\
&= q(\exp(t[0, \theta\mathbf{v}]))q^* && \text{(Definition 14)} \\
&= q(\exp[0, t\theta\mathbf{v}])q^* \\
&= q([\cos t\theta, \sin t\theta\mathbf{v}])q^* && \text{(Definition 15)} \\
&= [\cos t\theta, \sin t\theta\mathbf{v}'] && \text{(Corollary 3)} \\
&= \exp(t[0, \theta\mathbf{v}']) \\
&= \exp(t\log[\cos\theta, \sin\theta\mathbf{v}']) \\
&= \exp(t\log(qpq^*)) \\
&= (qpq^*)^t
\end{aligned}$$

□

We are now ready to show the main theorem of this section (inspired by [Watt & Watt, 1992]).

Proposition 21.
Let $q \in H_1$, $q = [\cos\theta, \sin\theta\mathbf{n}]$. Let $\mathbf{r} = (x, y, z) \in \mathbb{R}^3$ and $p = [0, \mathbf{r}] \in H$. Then $p' = qpq^{-1}$ is p rotated 2θ about the axis $\mathbf{n}$.

Proof of proposition 21
We first show how a vector $\mathbf{r}$ is rotated θ degrees about $\mathbf{n}$, using sine, cosine, and the scalar and vector products. We then show that the same result is obtained through rotation with quaternions.

Assume therefore that $\mathbf{r}$ is to be rotated θ to $R\mathbf{r}$ about an axis given by the unit vector $\mathbf{n}$ (see figure 3.1).

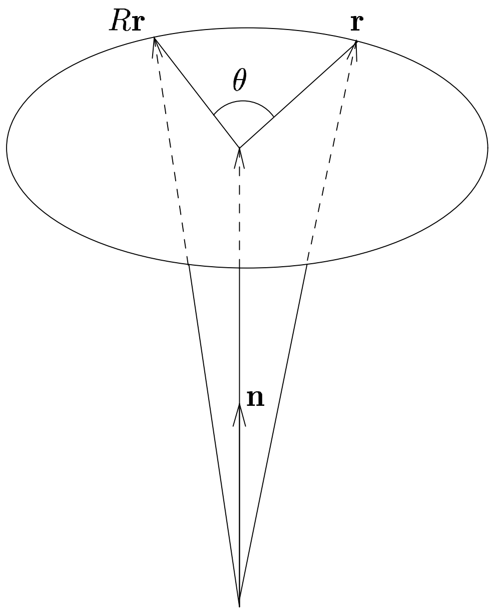


Figure 3.1: *The vector* $\mathbf{r}$ *is rotated* θ *to* $R\mathbf{r}$ *about an axis given by the unit vector* $\mathbf{n}$.

The vector $\mathbf{r}$ can be written as a sum of two components, $\mathbf{r}_{||}$ and $\mathbf{r}_{\perp}$, where $\mathbf{r}_{||}$ is the projection of $\mathbf{r}$ on $\mathbf{n}$, and $\mathbf{r}_{\perp}$ is orthogonal to $\mathbf{n}$ (see figure 3.2). We get

$$\begin{aligned} \mathbf{r}_{||} &= (\mathbf{r} \cdot \mathbf{n})\mathbf{n}, \quad \text{and} \\ \mathbf{r}_{\perp} &= \mathbf{r} - \mathbf{r}_{||} = \mathbf{r} - (\mathbf{r} \cdot \mathbf{n})\mathbf{n} \end{aligned}$$

To see how the rotation affects $\mathbf{r}$, we place a two-dimensional coordinate system in the plane that is orthogonal to $\mathbf{n}$ and contains the points designated by $\mathbf{r}$ and $R\mathbf{r}$. To do this, we need a vector $\mathbf{v}$ that is orthogonal to $\mathbf{r}_{\perp}$ and $\mathbf{n}$:

$$\mathbf{v} = \mathbf{n} \times \mathbf{r}_{\perp} = \mathbf{n} \times (\mathbf{r} - (\mathbf{r} \cdot \mathbf{n})\mathbf{n}) = \mathbf{n} \times \mathbf{r} - \mathbf{n} \times (\mathbf{r} \cdot \mathbf{n})\mathbf{n} = \mathbf{n} \times \mathbf{r} - \vec{\mathbf{0}} = \mathbf{n} \times \mathbf{r}$$

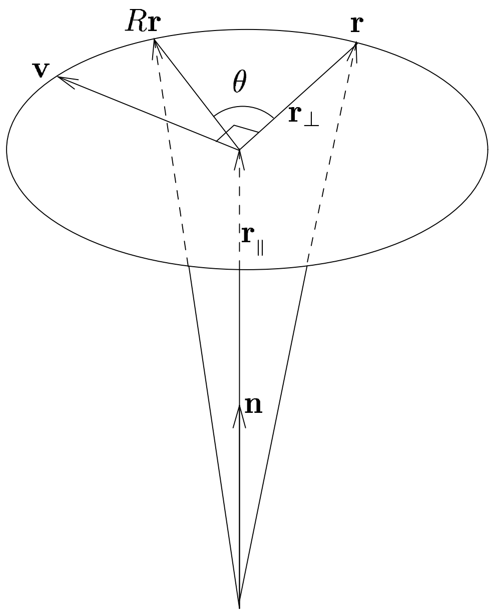


Figure 3.2: *In the two-dimensional coordinate system orthogonal to* $\mathbf{n}$, $(R\mathbf{r})_\perp$ *can be written* $(R\mathbf{r})_\perp = \mathbf{r}_\perp \cos\theta + \mathbf{v}\sin\theta$.

From figure 3.2 we see that $R\mathbf{r}$'s component orthogonal to $\mathbf{n}$, $(R\mathbf{r})_\perp$, is given by

$$(R\mathbf{r})_\perp = \mathbf{r}_\perp \cos\theta + \mathbf{v}\sin\theta.$$

We now get:

$$\begin{aligned} R\mathbf{r} &= (R\mathbf{r})_{||} + (R\mathbf{r})_\perp \\ &= \mathbf{r}_{||} + \mathbf{r}_\perp \cos\theta + \mathbf{v}\sin\theta \\ &= (\mathbf{r}\cdot\mathbf{n})\mathbf{n} + (\mathbf{r} - (\mathbf{r}\cdot\mathbf{n})\mathbf{n})\cos\theta + \mathbf{v}\sin\theta \\ &= (\mathbf{r}\cdot\mathbf{n})\mathbf{n} - (\mathbf{r}\cdot\mathbf{n})\mathbf{n}\cos\theta + \mathbf{r}\cos\theta + \mathbf{v}\sin\theta \\ &= (1-\cos\theta)(\mathbf{r}\cdot\mathbf{n})\mathbf{n} + \mathbf{r}\cos\theta + (\mathbf{n}\times\mathbf{r})\sin\theta \qquad (3.4) \end{aligned}$$

We will now examine the effect of applying a quaternion to a vector, and see that we get the same result as in equation 3.4.

We now look at $R_q(p) = qpq^{-1}$ and remind the reader that $p = [0, \mathbf{r}]$ and that q is a unit quaternion $[s, \mathbf{v}]$:

$$\begin{aligned}
R_q(p) &= [s,\mathbf{v}][0,\mathbf{r}][s,\mathbf{v}]^{-1} \\
&= [s,\mathbf{v}][0,\mathbf{r}][s,-\mathbf{v}] \\
&= [s,\mathbf{v}][\mathbf{v}\cdot\mathbf{r}, s\mathbf{r}-\mathbf{r}\times\mathbf{v}] \\
&= [s(\mathbf{v}\cdot\mathbf{r})-\mathbf{v}\cdot(s\mathbf{r}-\mathbf{r}\times\mathbf{v}), s(s\mathbf{r}-\mathbf{r}\times\mathbf{v})+(\mathbf{v}\cdot\mathbf{r})\mathbf{v}+\mathbf{v}\times(s\mathbf{r}-\mathbf{r}\times\mathbf{v})] \\
&= [0, s^2\mathbf{r}-s(\mathbf{r}\times\mathbf{v})+(\mathbf{v}\cdot\mathbf{r})\mathbf{v}+\mathbf{v}\times(s\mathbf{r})-\mathbf{v}\times(\mathbf{r}\times\mathbf{v})] \\
&= [0, s^2\mathbf{r}+(\mathbf{v}\cdot\mathbf{r})\mathbf{v}-\mathbf{v}\times(\mathbf{r}\times\mathbf{v})-2s(\mathbf{r}\times\mathbf{v})] \\
&= [0, s^2\mathbf{r}+(\mathbf{v}\cdot\mathbf{r})\mathbf{v}-(\mathbf{v}\cdot\mathbf{v})\mathbf{r}+(\mathbf{v}\cdot\mathbf{r})\mathbf{v}+2s(\mathbf{v}\times\mathbf{r})] \qquad (*) \\
&= [0, (s^2-\mathbf{v}\cdot\mathbf{v})\mathbf{r}+2(\mathbf{v}\cdot\mathbf{r})\mathbf{v}+2s(\mathbf{v}\times\mathbf{r})]
\end{aligned}$$

$(*)$ Here we use the identity $\mathbf{v}_1\times(\mathbf{v}_2\times\mathbf{v}_3)=(\mathbf{v}_1\cdot\mathbf{v}_3)\mathbf{v}_2-(\mathbf{v}_1\cdot\mathbf{v}_2)\mathbf{v}_3$

Since q is a unit quaternion, we can write $q=[\cos\theta,(\sin\theta)\mathbf{n}]$, where $|\mathbf{n}|=1$ (by proposition 12 on page 14).

Substituting this into $R_q(p)$, we get:

$$\begin{aligned}
R_q(p) &= [0,\ (\cos^2\theta-\sin^2\theta(\mathbf{n}\cdot\mathbf{n}))\mathbf{r}+2((\sin\theta)\mathbf{n}\cdot\mathbf{r})(\sin\theta)\mathbf{n} \\
&\qquad +2\cos\theta((\sin\theta)\mathbf{n}\times\mathbf{r})] \\
&= [0,\ (\cos^2\theta-\sin^2\theta)\mathbf{r}+(2\mathbf{n}\sin^2\theta)(\mathbf{n}\cdot\mathbf{r}) \\
&\qquad +2\cos\theta\sin\theta(\mathbf{n}\times\mathbf{r})] \\
&= [0,\ \mathbf{r}\cos 2\theta+(1-\cos 2\theta)(\mathbf{n}\cdot\mathbf{r})\mathbf{n}+(\mathbf{n}\times\mathbf{r})\sin 2\theta]
\end{aligned}$$

From the above derivation, we see that the result is the same vector as in equation 3.4 except that the above equation has 2θ instead of θ. Thus, given a unit vector $\mathbf{n}$ and a rotation angle θ, the unit quaternion $[\cos\theta, \sin\theta\mathbf{n}]$ rotates $\mathbf{r}$ through the angle 2θ about $\mathbf{n}$. □

As a consequence of this proposition, we get the following important corollary:

Corollary 4. (to proposition 21)
Any general three-dimensional rotation θ about $\mathbf{n}$, $|\mathbf{n}|=1$ can be obtained by a unit quaternion.

Proof of corollary 4
In the above proposition choose q such that $q=[\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\mathbf{n}]$. Thus the desired rotation is obtained.

□

Composition of rotation is achieved by multiplying the corresponding quaternions. This is formalized in:

Proposition 22.
Let $q_1, q_2 \in H_1$. Rotation by q_1 followed by rotation by q_2 is equivalent to rotation by q_2q_1.

Proof of proposition 22
Given $p \in H$, the result follows directly from

$$\begin{aligned}
q_2(q_1 p q_1^{-1})q_2^{-1} &= (q_2 q_1)p(q_1^{-1} q_2^{-1}) & \\
&= (q_2 q_1)p(q_1^* q_2^*) & \text{(proposition 13)} \\
&= (q_2 q_1)p(q_2 q_1)^* & \text{(proposition 8)} \\
&= (q_2 q_1)p(q_2 q_1)^{-1} & \text{(proposition 13)}
\end{aligned}$$

□

3.3.7 Geometric intuition

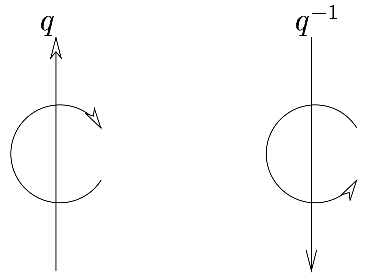
We will make some observations that can aid the intuitive understanding of rotation with quaternions.

The quaternions q and q^{-1}

Let $q = [s, \mathbf{v}] \in H_1$. Then

$$[s, v]^{-1} = q^{-1} = q^* = [s, -v]$$

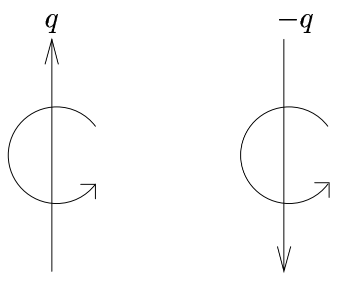
It can be useful to consider the geometric interpretation of this: The inverse of q, q^{-1}, rotates the same number of degrees as q, but the axis points in the opposite direction:



By inverting the axis, the direction of rotation is reversed; a subsequent rotation by q^{-1} cancels out the effect of the rotation q.

The quaternions q and $-q$

The quaternion $-q$ represents exactly the same rotation as q (this follows from proposition 6). This may seem surprising, but should be expected: A rotation through the angle θ about the axis $\mathbf{n}$ can also be expressed as a rotation through the angle $-\theta$ about the axis $-\mathbf{n}$. It is therefore aesthetically pleasing that we find both rotations on the unit quaternion sphere. The same duality is also found in Euler's theorem.



Non-unit quaternions

It follows from proposition 18 that all quaternions on the line $rq,\ r \in \mathbb{R},\ r \neq 0$ represent the same rotation.

3.3.8 Quaternions and differential calculus

In this section we show a number of common results from differential calculus for functions that map into H. The results will later be used to show that certain interpolation curves are differentiable.

Proposition 23.
Let $q = [\cos\theta, \sin\theta\mathbf{v}] \in H_1, t \in \mathbb{R}$. *Then*

$$\frac{d}{dt}q^t = q^t \log(q)$$

Proof of proposition 23
The equation is shown through simple calculation of the two sides of the equation.

The left-hand side:

$$\frac{d}{dt}q^t = \frac{d}{dt}\exp(t\log(q)) = \frac{d}{dt}\exp(t[0,\theta\mathbf{v}]) = \frac{d}{dt}[\cos(t\theta), \sin(t\theta)\mathbf{v}] = \underline{\theta[-\sin(t\theta), \cos(t\theta)\mathbf{v}]}$$

The right-hand side:

$$\begin{aligned} q^t\log(q) &= \exp(t\log(q))\log(q) = [\cos(t\theta), \sin(t\theta)\mathbf{v}][0,\theta\mathbf{v}] \\ &= [-\theta\sin(t\theta)(\mathbf{v}\cdot\mathbf{v}), \theta\cos(t\theta)\mathbf{v} + \theta\sin(t\theta)(\mathbf{v}\times\mathbf{v})] \\ &= [-\theta\sin(t\theta), \theta\cos(t\theta)\mathbf{v}] = \underline{\theta[-\sin(t\theta), \cos(t\theta)\mathbf{v}]} \end{aligned}$$

□

We also want to show the chain rule and the product rule for quaternions. We will first show the product rule. The purpose of this derivation is to ensure that the order of the quaternions in the differentiated expression is correct; it is important to make sure that this is the case since quaternion multiplication is not commutative.

Proposition 24. (The product rule)
Let $f, g \in C^1(\mathbb{R}, H)$. Then

$$\frac{d}{dt}(f(t)g(t)) = (\frac{d}{dt}f(t))g(t) + f(t)(\frac{d}{dt}g(t))$$

Proof of proposition 24

$$\begin{aligned}
\frac{d}{dt}(f(t)g(t)) &= \lim_{\delta\to 0} \frac{f(x+\delta)g(x+\delta) - f(x)g(x)}{\delta} \\
&= \lim_{\delta\to 0} \frac{f(x+\delta)g(x+\delta) - f(x+\delta)g(x) + f(x+\delta)g(x) - f(x)g(x)}{\delta} \\
&= \lim_{\delta\to 0} \left(f(x+\delta)\frac{g(x+\delta)-g(x)}{\delta} + \frac{f(x+\delta)-f(x)}{\delta}g(x) \right) \\
&= f(x)g'(x) + f'(x)g(x)
\end{aligned}$$

□

Proposition 25. (The chain rule)
Let $f \in C^1(H, H)$, $g \in C^1(\mathbb{R}, H)$. Then $\frac{d}{dt}f(g(x)) = f'(g(x))g'(x)$

Proof of proposition 25
We compute the derivative at an arbitrary point $c \in \mathbb{R}$:

$$\begin{aligned}
\frac{d}{dt}f(g(c)) &= \lim_{x\to c} \frac{f(g(x)) - f(g(c))}{x-c} \\
&= \lim_{x\to c} \frac{(f(g(x)) - f(g(c)))(g(x)-g(c))^{-1}(g(x)-g(c))}{x-c} \\
&= \lim_{x\to c} \left(\frac{f(g(x)) - f(g(c))}{g(x)-g(c)} \frac{g(x)-g(c)}{x-c} \right) \\
&= f'(g(c))g'(c)
\end{aligned}$$

□

Finally we state the following result, that has no obvious counterpart in the real numbers:

Proposition 26.
Let $q \in C^1(\mathbb{R}, H_1), r \in C^1(\mathbb{R}, \mathbb{R})$. Since q maps into H_1, $q(t)$ can be written $[\cos\theta(t), \mathbf{v}(t)\sin\theta(t)]$, *and we have*

$$\begin{aligned}
\frac{d}{dt}q(t)^{r(t)} = \Big[& -\sin\Big(r(t)\theta(t)\Big)\Big(r'(t)\theta(t) + r(t)\theta'(t)\Big), \\
& \cos\Big(r(t)\theta(t)\Big)\Big(r'(t)\theta(t) + r(t)\theta'(t)\Big)\mathbf{v}(t) + \sin\Big(r(t)\theta(t)\Big)\mathbf{v}'(t)\Big]
\end{aligned}$$

Proof of proposition 26

$$\begin{aligned}
\frac{d}{dt} q(t)^{r(t)} &= \frac{d}{dt} \exp(r(t) \log(q(t))) \\
&= \frac{d}{dt} \exp(r(t)[0, \mathbf{v}(t)\theta(t)]) \\
&= \frac{d}{dt} \exp[0, r(t)\mathbf{v}(t)\theta(t)] \\
&= \frac{d}{dt} [\cos(r(t)\theta(t)), \sin(r(t)\theta(t))\mathbf{v}(t)] \\
&= \Big[\ -\sin\Big(r(t)\theta(t)\Big)\Big(r'(t)\theta(t) + r(t)\theta'(t)\Big), \\
&\qquad \cos\Big(r(t)\theta(t)\Big)\Big(r'(t)\theta(t) + r(t)\theta'(t)\Big)\mathbf{v}(t) + \sin\Big(r(t)\theta(t)\Big)\mathbf{v}'(t)\Big]
\end{aligned}$$

$\square$

Compare this equation to the derivative of two real-valued functions $u, v \in C^1(\mathbb{R}, \mathbb{R})$:

$$\frac{d}{dt} u^v = v u^{v-1} \frac{d}{dt} u + u^v \log(u) \frac{d}{dt} v.$$

This equation is different from the equation in proposition 26, and it is unlikely that it holds in general for quaternions.

3.4 An algebraic overview

This section contains a short resume of the algebraic properties of the rotational modalities. The mathematical concepts are assumed to be known by the reader, and they are therefore not described in detail. This section elaborates on the algebra previously discussed and is only intended for the interested reader.

The space of three-dimensional rotations is not a simple vector-space but a closed three-dimensional manifold and also a non-Abelian group (see section 3.3.3) known in the literature as $SO(3)$ [Shoemake, 1994b]. Here S is for "special" and the $O(3)$ stems from the definition: $O(n) = \{n \times n \; matrices \mid O^t O = I\}$ — the set of orthonormal $n \times n$ matrices.

The set of unit quaternions constitutes a subgroup of the quaternion group (see section 3.3.4). In the literature, H_1 is also called S^3 [Shoemake, 1985]. This subgroup constitutes a hypersphere in quaternion space. The spherical metric for S^3 is equivalent to the angular metric for $SO(3)$ [Shoemake, 1985].

Furthermore the rotation group can be projected onto the four-dimensional unit sphere of unit quaternions. This projection is two-to-one (see section 3.3.7): For each rotation there are two corresponding unit quaternions — q that is obtained directly and $-q$, the antipodal unit quaternion (see [Shoemake, 1994b], [Foley et al., 1990] and appendix B). This is because $SO(3)$ has the same topology as the three-dimensional projection space called ($\mathbf{R}P^3$), while the set of unit quaternions constitutes a hypersphere (S^3) that is topologically different from $\mathbf{R}P^3$ [Shoemake, 1994b].

Bearing these similarities and the small discrepancy in mind, we can see that by developing a smooth interpolation between unit quaternions, we get a smooth interpolation between general rotations. The problem is not trivial, in particular because H_1 constitutes a non-Euclidean space, which excludes the usual interpolation methods such as splines. Our task is to find an equivalent interpolation curve on the surface of the four-dimensional unit sphere.

Chapter 4

A comparison of quaternions, Euler angles and matrices

In the previous chapter we introduced two rotational modalities:

- Rotation defined by Euler angles represented by general transformation matrices.
- Rotation defined by Euler's theorem represented by quaternions.

In this chapter we will describe advantages and disadvantages of the different modalities.

4.1 Euler angles/matrices — Disadvantages

Traditionally homogeneous matrices have been used to represent Euler angles because the basic rotation matrices for rotation about the x-, y-, and z-axes are simple and well-known. This historically based choice has some disadvantages, though. We will discuss the disadvantages below.

Lack of intuition

Describing a general rotation as rotations about the three basis axes is not natural for an animator. If, for instance, the animator wants to rotate an object 30 degrees about a rotation axis given by the vector $(1, 1, 1)$, it is quite tedious to derive the corresponding Euler angles about the three basis axes.

The order of rotation axes is important

The user of a graphical system must express rotations in respect to a certain convention that defines in which order the three basis rotations are applied. Different conventions yield different results. For example, a rotation of $(\frac{\pi}{2}, \frac{\pi}{4}, 0)$ yields different orientations depending on which convention of x, y, z and y, x, z is being used.

As an example, we examine an object in a coordinate system (see figure 4.1). By rotating the object in figure 4.1.*i* the angle $\frac{\pi}{2}$ about x and then $\frac{\pi}{4}$ about y the result is 4.1.*ii*. If the convention y, x, z is used instead, the rotation is done by $\frac{\pi}{4}$ about y and then $\frac{\pi}{2}$ about x yielding 4.1.*iii*.

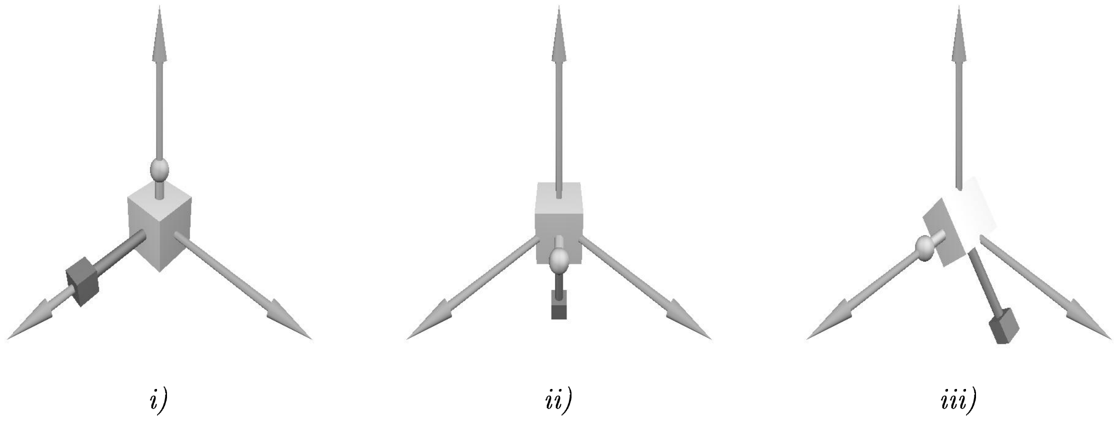


Figure 4.1: *In the coordinate system the x-axis points to the right, the y-axis points up and the z-axis point to the left. Figures ii) and iii) show the result of applying different rotation conventions to the object in figure i).*

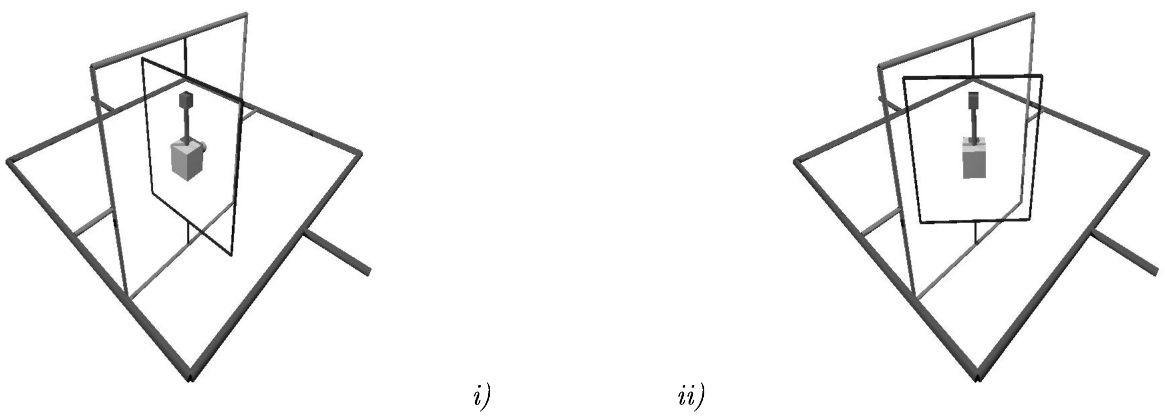
Gimbal lock

Getting an intuitive understanding of how rotation matrices work is quite difficult. In particular, it is difficult to predict how successive rotations about the basis axes affect each other. Considering that the matrix representation of Euler angles has an innate singularity in the parameterization makes this even more difficult. It is possible to create series of rotations, where one degree of freedom in the rotation is lost. This situation is called *gimbal lock*.

Gimbal lock is a concept originating from the air and space industry, where gyroscopes are used [Shoemake, 1985] [Watt & Watt, 1992] [Verplaetse, 1995]. A gyroscope basically consists of three concentric rings. See figure 4.2 for an illustration of this (the example is inspired by [McCool, 1995]).

In 4.2.*i* the inner ring represents the x-axis, the center ring the y-axis and the outer ring represents the z-axis. A rotation about the x-axis can for example result in 4.2.*ii*, where the object is rotated approximately 45 degrees about the x-axis. If we then rotate 90 degrees about the y-axis, we get the situation shown in figure 4.3.

In this situation, the x and the z-rotation acts about the same axis. This is an example of gimbal lock.



i) *ii)*

Figure 4.2: *In the coordinate system the x-axis points up, the y-axis points to the left and the z-axis points to the right. The inner ring represents the x-axis, the center ring the y-axis and the outer ring represents the z-axis. From the starting point i) the x-axis is rotated approximately 45 degrees in ii).*

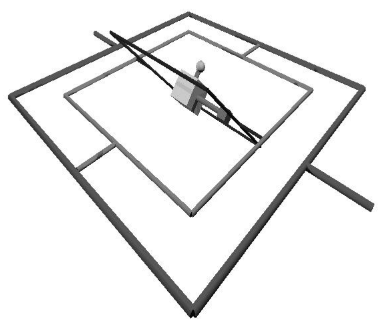


Figure 4.3: *In the coordinate system the x-axis points up, the y-axis points to the left and the z-axis points to the right. In this situation the x and the z-rotation act about the same axis. This phenomenon is called gimbal lock.*

Mathematically gimbal lock corresponds to loosing a degree of freedom in the general rotation matrix (see appendix B):

$$R(\alpha,\beta,\gamma) = \begin{pmatrix} \cos\beta\cos\gamma & \cos\gamma\sin\alpha\sin\beta - \cos\alpha\sin\gamma & \cos\alpha\cos\gamma\sin\beta + \sin\alpha\sin\gamma & 0 \\ \cos\beta\sin\gamma & \cos\alpha\cos\gamma + \sin\alpha\sin\beta\sin\gamma & \cos\alpha\sin\beta\sin\gamma - \cos\gamma\sin\alpha & 0 \\ -\sin\beta & \cos\beta\sin\alpha & \cos\alpha\cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

If we let $\beta = \frac{\pi}{2}$, then a rotation with α will have the same effect as applying the same rotation with $-\gamma$. This can be seen from the following derivation (using the addition formulas for cos and sin):

$$
\begin{aligned}
R(\alpha, \tfrac{\pi}{2}, \gamma) &= \left(\begin{array}{cccc} 0 & \cos\gamma\sin\alpha - \cos\alpha\sin\gamma & \cos\alpha\cos\gamma + \sin\alpha\sin\gamma & 0 \\ 0 & \cos\alpha\cos\gamma + \sin\alpha\sin\gamma & \cos\alpha\sin\gamma - \cos\gamma\sin\alpha & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right) \\
&= \left(\begin{array}{cccc} 0 & \sin(\alpha-\gamma) & \cos(\alpha-\gamma) & 0 \\ 0 & \cos(\alpha-\gamma) & \sin(\alpha-\gamma) & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right)
\end{aligned}
$$

This expression shows that the rotation only depends on the difference $\alpha - \gamma$ and therefore it has only one degree of freedom instead of two. For $\beta = \frac{\pi}{2}$ changes of α and γ result in rotations about the same axis.

Implementing interpolation is difficult

Normally the coordinates of each basis axis are interpolated independently. Thereby the interdependencies between the axes are ignored. As an example, this results in unexpected effects when applying simple linear interpolation. See section 6.1.1 for a treatment of this.

Ambiguous correspondence to rotations

Given a rotation matrix it is difficult to solve the inverse problem: What are the original rotations about the basis axes? In general, there is no unambiguous solution to this problem. See appendix A or [Shoemake & Duff, 1994] for more detail.

In addition to this, a rotation can be represented by many different rotation matrices. All in all the mapping between rotations and rotation matrices is neither injective nor surjective.

The result of composition is not apparent

According to Euler's theorem two successive rotations can be expressed as one. The two rotation matrices must be composed and multiplied followed by extraction of the resulting rotation. To determine this rotation is tedious and in general not possible as mentioned above (see page 90 or [Shoemake & Duff, 1994] for further detail).

The representation is redundant

Homogeneous matrices contain expendable information. If the matrices are to be used exclusively for rotation, the matrices will have zeroes for indices $(4, i)$ and $(i, 4), i \in \{1, 2, 3\}$. In addition to this, the matrix uses 9 places for the 4 degrees of freedom that are necessary to describe a rotation according to Euler's theorem.

On top of this numerical inaccuracies will be problematic. Since rotation matrices must be orthonormal there are 6 constraints (each row must be a unit vector and the columns must be mutually orthogonal) that must be maintained during the computations.

4.2 Euler angles/matrices — Advantages

An advantage of matrix implementations is that the mathematics is well-known and that matrix applications are relatively easy to implement using standard packages. These advantages are more historically than rationally determined though.

The main advantage of the matrix representation is the ability of the homogeneous matrix to represent all the other basic transformations, for example translation, scaling, projection, and shearing.

4.3 Quaternions — Disadvantages

Quaternions only represent rotation

It is possible to implement translation using quaternions (quaternion addition can be used as a translation transformation interpreting the vector part of the quaternion as the translation vector). In [Maillot, 1990] a kind of homogeneous quaternions are defined with a multiplication making both translation and rotation multiplicative.

Even though it is possible to define a homogeneous quaternion and thereby including the translation composition, this extension is not as elegant as the homogeneous matrices. The homogeneous extension appears to be ignored in the literature. Quaternions are used for rotation exclusively, matrices are used for all other transformations.

Quaternion mathematics appears complicated

Quaternions are not included in standard curriculum of modern mathematics. Some might study the quaternion group in algebra, but knowledge of quaternions is, in general, not widespread. Therefore quaternions require a bit of work in the beginning. However, quaternions should pose no problem for someone able to understand matrix algebra.

4.4 Quaternions — Advantages

Obvious geometrical interpretation

Quaternions express rotation as a rotation angle about a rotation axis. This is a more natural way to perceive rotation than Euler angles.

The obvious correspondence between Euler's theorem and rotations represented by quaternions gives a nice intuitive understanding of quaternions. The mapping between rotations and quaternions is therefore unambiguous with the exception that every rotation can be represented by two quaternions. This appears to be a weakness in the quaternion representation. That q and $-q$ correspond to the same rotation is on the other hand mathematically pleasing. This is because rotations themselves come in pairs. Given a rotation, the same rotation is obtained by rotating in the opposite direction about the opposite axis. (see 3.3.7 on page 22).

Coordinate system independency

Quaternion rotation in not influenced by the choice of coordinate system. The user of an animation system does not need to worry about a certain convention of the order of rotation about explicit axes.

Simple interpolation methods

Quaternions allow elegant formulations of a range of interpolation methods. Achieving a smooth interpolation is therefore simpler using quaternions than Euler angles. We will give a comprehensive treatment of this in chapter 6.

Compact representation

The representation of rotation using quaternions is compact in the sense that it is four dimensional and thereby only contains the four degrees of freedom required according to Euler's theorem.

In theory all non-zero quaternions can be used for rotation (by proposition 18). In practical applications only unit quaternions will be used. Thus, only one constraint on the representation must be upheld during computation compared to the six constraints on rotation matrices.

No gimbal lock

Since gimbal lock is innate to the matrix representation of Euler angles, this problem does not appear in the quaternion representation.

Simple composition

Rotations are easily composed when using quaternions. The composition corresponds to multiplication of the involved quaternions. Rotation with q_1 followed by rotation with q_2 is achieved by rotating with the quaternion $q_2 q_1$.

4.5 Conclusion

We have stated a series of advantages and disadvantages for the two rotation modalities. Using Euler angles represented by matrices leads to several problems. Rotation must be expressed as the angles about three explicit axes, with the order being important. It is possible to encounter gimbal lock and finally it is troublesome to uphold the mathematical constraints on the representation during calculations.

The quaternion representation is compact with a more natural geometrical interpretation and a parameterization of rotation that is not dependent on the coordinate system.

The only real advantage of matrices is the possibility of representing all the other transformations.

All in all, quaternions offer the best choice for representation of rotations.

4.6 Other modalities

In the previous sections we have argued that rotations should be represented by quaternions based on Euler's theorem. However, we have only compared to one other modality — rotation matrices based on Euler angles.

Obviously, other modalities are possible. For example the two modalities can be combined. It is possible to define rotation matrices based on Euler's theorem (an example can be found in [Foley et al., 1990] exercise 5.15). Thereby the problems connected to Euler angles are avoided yielding a better correspondence between rotation matrices and the set of rotations. There are still problems with this modality though. For instance, the inverse mapping from rotation matrices to rotations is still ambiguous. Further, the matrix representation is not well-suited for interpolation algorithms. For example the matrices still need to fulfill the constraints imposed by being orthonormal matrices.

We will limit ourselves from discussing other modalities. This is simply because the two we have mentioned are by far the most important. Euler angles and matrices are the most common modality in both the literature and in applications. We claim that quaternions are more appropriate.

Chapter 5

Visualizing interpolation curves

In chapter 6 we discuss a series of interpolation methods that can interpolate between two or more quaternions. We would like to compare these methods from a theoretical perspective. At the same time it is natural to compare the results of the methods in practice, to see if practice reflects our theoretical considerations.

This section contains a short description of the visualization methods used and the motivation for each type of visualization.

5.1 Direct visualization

The most obvious visualization method is to apply the interpolated rotations to the object. This is most easily achieved by defining an object using a three-dimensional visualization tool, and then rotating it with the interpolated rotations. We let this visualization method produce an animated sequence that shows how the object is rotated. The method gives an intuitive feel for how the interpolation curve behaves, but it is difficult to say anything concrete about the smoothness of the interpolation curves or the variation in angular velocity. We therefore need other methods of visualization that can provide us with this information.

5.2 Visualizing an approximation of angular velocity

We want to visualize the angular velocity of the interpolation curve. For example it will be interesting to see if some of the interpolation curves have constant angular velocity.

In the following, q_i denotes the i'th frame, i. e. the i'th quaternion in a discrete quaternion interpolation curve.

To produce a graph of the angular velocity, we must define a function that gives an approximation of the angular velocity. Then all that remains is to plot this function against the interpolation parameter. A number of different approximations to the angular velocity can be defined based in either physics or mathematics. We will base our definition in mathematics, and use that we have defined a norm on quaternions. We can define the distance between two quaternions q_1, q_2 to be $d(q_1, q_2) = \|q_1 - q_2\|$. Then the angular velocity V in the $i'th$ quaternion q_i can be approximated by the centered average:

$$V(q_i) = \frac{d(q_i, q_{i-1}) + d(q_i, q_{i+1})}{2} = \frac{\|q_i - q_{i-1}\| + \|q_i - q_{i+1}\|}{2} \tag{5.1}$$

Plotting V as a function of the interpolation parameter yields a graph of an approximation to the angular velocity.

We will omit the first and last key frames from the angular velocity graph, since no obvious angular velocity can be assigned to them. Thereby the leftmost point on the angular velocity graph is the angular velocity in the first interpolated frame. The remaining key frames we will mark with an asterisk in the velocity graph.

5.3 Visualizing the smoothness of interpolation curves

We would also like to visualize the interpolation curves to see, for example, how smooth they appear.

Since quaternion space is four-dimensional, we cannot visualise the interpolated curves directly. We will always interpolate between unit quaternions, and the interpolated quaternions will always (with a few exceptions in chapter 6 on page 38 and 69) be unit quaternions. This means that we only need three dimensions to visualize the interpolation curves, because they lie on the surface of the unit sphere. In practice it can be difficult to visualise this space effectively since it must be presented via a two-dimensional media (paper or monitor). We can remove another dimension by interpolating between quaternions in the same three-dimensional hyperplane. This can be achieved by fixing one of the quaternion coordinates in all key frames. Thus the interpolated curves should stay inside a two-dimensional space that can be shown in the plane. To keep the association to the four-dimensional unit sphere, we elect to show the curves on the surface of the three-dimensional unit sphere (a two-dimensional sub manifold of $\mathbb{R}^3$). In chapter 6 we will argue that the ideal interpolation curve will lie on the surface of the quaternion unit sphere. Our choice of visualization ensures that we can visually determine if the visualization curve stays on the surface of the unit sphere. See figure 5.3 for an example of this.

The actual visualizations are produced using a ray tracer [POV, 1997]. Here we generate a large sphere that represents the three-dimensional unit sphere. The first key frame is shown as a medium-sized sphere, other key frames are shown using a bit smaller spheres, and interpolated points are shown using small spheres. See figures 5.1 through 5.3 below.

5.4 Some examples of visualization

In this section we show some examples of the visualization of the angular velocity graphs and the interpolation curves. The interpolation methods are described in section 6; we only describe properties of the resulting visualizations here.

The interpolation is performed on the frames given in table 5.1. The key frames are given by a general rotation. As noted above, we choose the rotation angle and axis such that all the rotations lie in the same three-dimensional hyperplane.

In figures 5.1 through 5.3 we discuss different properties of the visualizations. Note that there is no obvious connection between the rotations in figure 5.1 and the points on the surface of the sphere. This is because the table contains general rotations, while the visualizations show the corresponding quaternions. Since we are only interested in the geometric shape of the curves, the absolute positioning of the key frames on the sphere is irrelevant.

Rotation angle $\theta \in \,]-\pi, \pi]$	Rotation axis $v \in \mathbb{R}^3$
1	(1,3,0)
1.9	(-1,0,0)
0	(-2,1,0)
-2	(3,4,0)
-1	(-1,4,0)
1	(1,3,0)

Table 5.1: *Key frames*

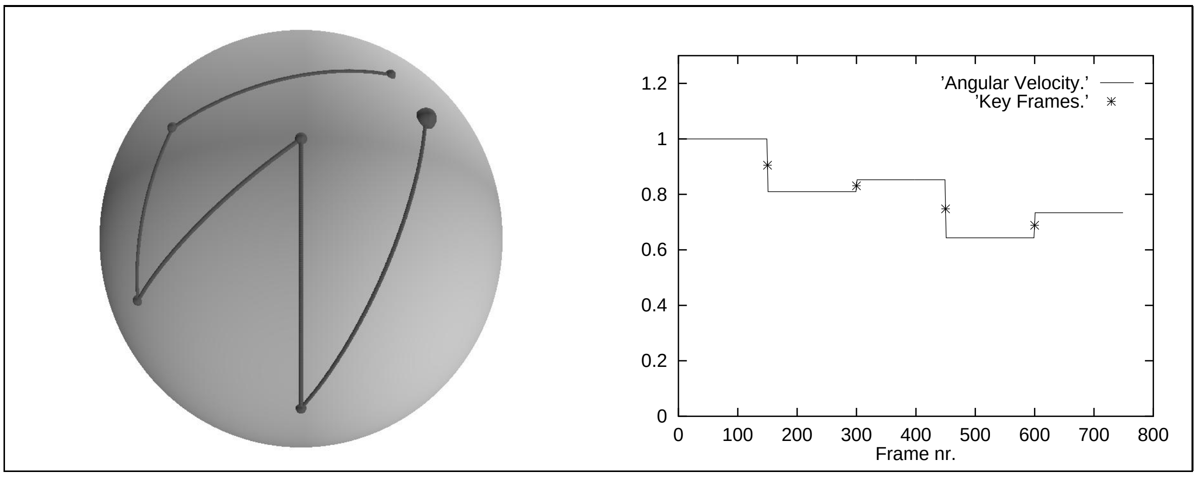


Figure 5.1: *The interpolation curve stays on the surface of the sphere, but it is not differentiable in any key frames; the curve "breaks" when it passes through the key frames. The angular velocity graph is piecewise continuous and shows that the angular velocity is constant between keys. This method of interpolation is called Slerp and is described in section 6.1.5.*

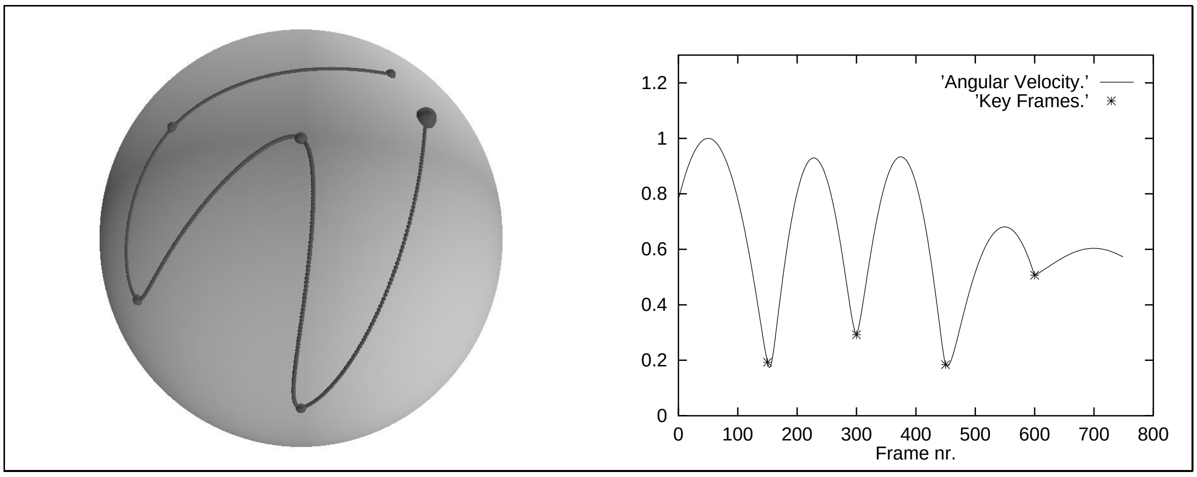


Figure 5.2: *The interpolation curve is now differentiable through all key frames. Compare, for example, the key frame in the middle of the figure with the corresponding key frame in figure 5.1. The angular velocity graph is continuous and assumes local minima at the key frames. This interpolation curve is called Squad, and it is described in section 6.2.1.*

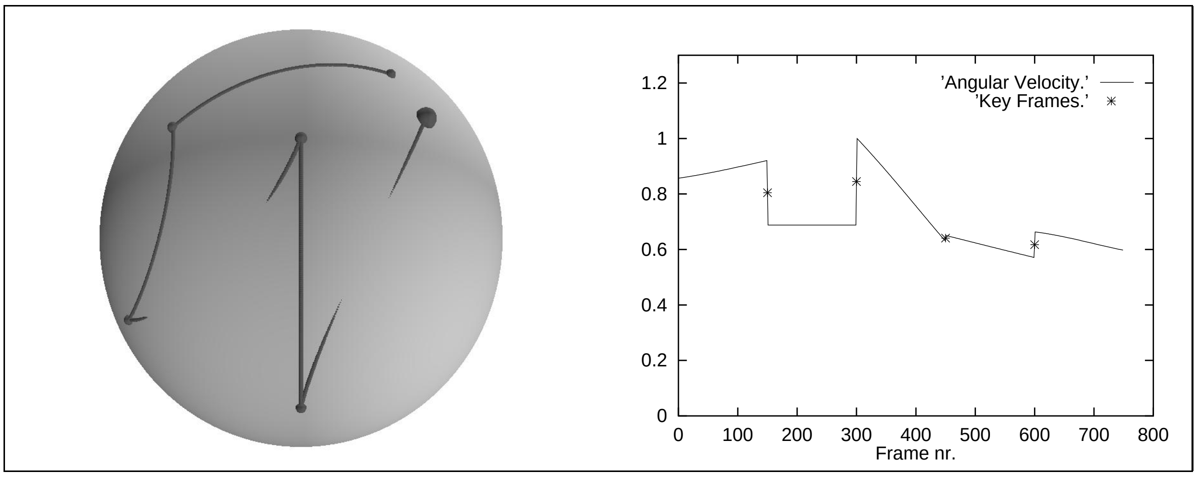


Figure 5.3: *This interpolation curve dips below the surface of the three-dimensional unit sphere. This means that the interpolated points are not unit quaternions, and thus the points do not lie on the surface of the sphere. The angular velocity graph is piecewise linear. This interpolation curve is called LinEuler, and is described in section 6.1.1*

In general we will illustrate the interpolation methods with the last two visualization methods: Sphere and graph. We have included the animated sequences for the sake of completeness, since it is in this context that the interpolation serves its practical purpose.

A few examples of these animated sequences can be seen at **`http://kantine.diku.dk/~myth/gif/`**

Chapter 6

Interpolation of rotation

In the previous chapters we presented and discussed two rotational modalities. In this chapter we will investigate how well-suited the modalities are for interpolation methods.

Gradually we will move from simple, intuitive methods to more advanced, theoretically well-founded interpolation methods. Parallel to the derivation of methods we will discuss what we perceive as criteria defining the optimal interpolation method. Our modest aim is to define and implement this optimal method.

6.1 Interpolation between two rotations

Initially we will limit ourselves to looking at interpolation between two rotations. We will use the rotation representations as inspiration for treating a series of simple methods.

Each of the methods results in an interpolation curve defined as follows. Given an arbitrary set M we interpolate between $x_0 \in M$ and $x_1 \in M$ parameterised by $h \in [0,1]$. The resulting interpolation curve $\gamma : M \times M \times [0,1] \curvearrowright M$ must satisfy the constraints:

$$\begin{aligned} \gamma(x_0, x_1, 0) &= x_0 \\ \gamma(x_0, x_1, 1) &= x_1 \end{aligned}$$

6.1.1 Linear Euler interpolation: *LinEuler*

The most obvious method is simply linear interpolation between two tuples of Euler angles. Calling this interpolation curve *LinEuler*, interpolation between $v_0 = (x_0, y_0, z_0) \in \mathbb{R}^3$ and $v_1 = (x_1, y_1, z_1) \in \mathbb{R}^3$ can be stated algorithmically using $h \in [0,1]$ as the interpolation parameter:

$$LinEuler(v_0, v_1, h) = v_0(1-h) + v_1 h \tag{6.1}$$

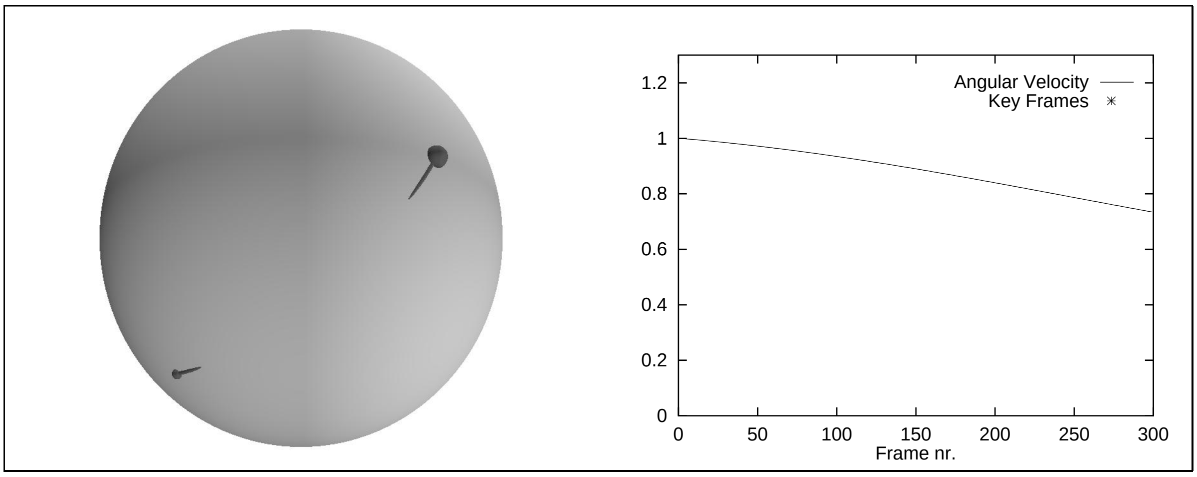


Figure 6.1: *Interpolation curve and velocity graph for Linear Euler interpolation — LinEuler. Between the two key frames are 300 interpolated frames.*

Figure 6.1 is not a very good illustration of the interpolation curve. The curve "lives" in the set of Euler angles while the illustration shows the corresponding quaternions. As described in chapter 5, the illustration is designed to show unit quaternions with the z coordinate equal to zero. The quaternions corresponding to the key frames meet these criteria but the rest of the interpolation curve does not. The curve consists of unit quaternions but their z coordinate is not generally equal to zero. Therefore the curve disappears from the surface of the sphere in the illustration.

This behaviour is neither optimal[1] nor intuitively correct.

The velocity graph shows that the animation corresponding to the interpolation will gradually slow down. Again, this behaviour is neither optimal nor intuitively correct.

6.1.2 Linear Matrix interpolation: *LinMat*

An alternative simple attempt is linear interpolation between rotation matrices — meaning linear interpolation of every single matrix element independently of the others.

This can be stated simply algorithmically. With parameter $h \in [0, 1]$, the interpolation curve between the rotation matrices $M_0 \in \mathbb{R}^4 \times \mathbb{R}^4$ and $M_1 \in \mathbb{R}^4 \times \mathbb{R}^4$ the curve is defined by

$$LinMat(M_0, M_1, h) \quad = \quad M_0(1-h) + M_1 h \qquad (6.2)$$

As with linear Euler interpolation, the curve for linear matrix interpolation does not in general lie on the unit sphere, since linear interpolation between orthonormal matrices will not in general produce orthonormal matrices. Thus, the interpolated matrices are general homogeneous

[1]We perceive "optimal" informally so far. Later we will state a strict definition of the optimal interpolation curve.

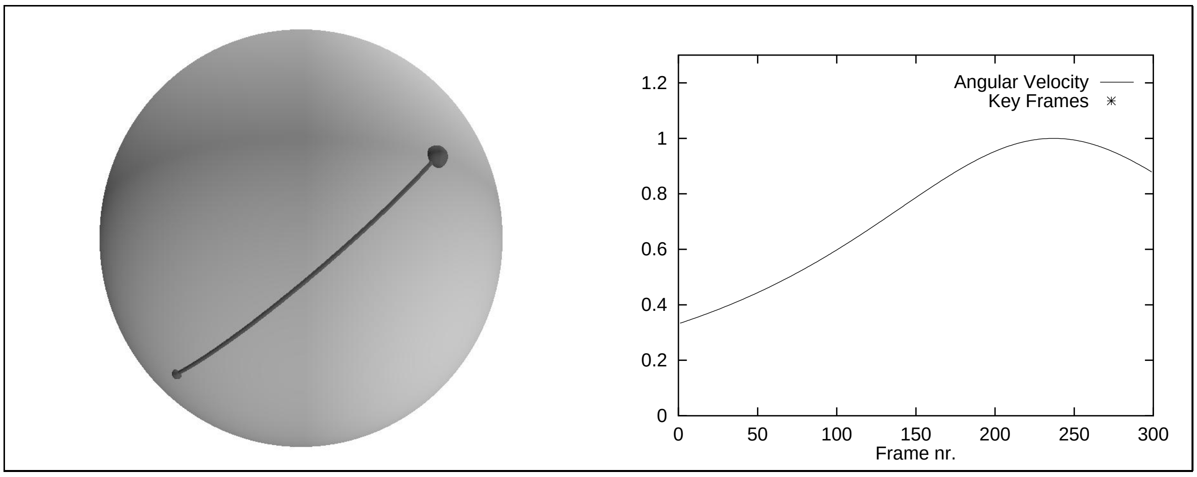


Figure 6.2: *Interpolation curve and velocity graph for linear matrix interpolation — LinMat. Between the two key frames are 300 interpolated frames.*

transformation matrices containing translation, scaling, projection and other transformation elements. Thereby the interpolation can become arbitrarily wrong. For example, it is possible to collapse the entire object into a single point [Shoemake & Duff, 1994].

Our visualization methods cannot show other transformations than rotation. Therefore we are not able to illustrate that the interpolated matrices are not pure rotation matrices. By converting matrices to quaternions we preserve only the rotational part.

In figure 6.2 we have projected the interpolation curve on to the unit quaternion sphere to show the pure rotational part of the interpolation curve. Finally, after these detours, we arrive at a quite nice interpolation curve.

As explained, the nice illustration does not imply that the interpolation method is usable — only a component of the full interpolation curve is shown. The method is only discussed for completeness.

6.1.3 Linear Quaternion interpolation: *Lerp*

Finally, another obvious attempt is linear interpolation between rotation quaternions (called *Lerp* for linear inter**p**olation). For $q_0, q_1 \in H$ and $h \in [0,1]$ this interpolation curve can be stated:

$$Lerp(q_0, q_1, h) \quad = \quad q_0(1-h) + q_1 h \tag{6.3}$$

The interpolation curve for linear interpolation between quaternions gives a straight line in quaternion space. The curve therefore dips below the surface of the unit sphere. Since all quaternions on a line through the origin give the same rotation[2], the curve can be projected on

[2]Except the origin itself. This follows from proposition 18.

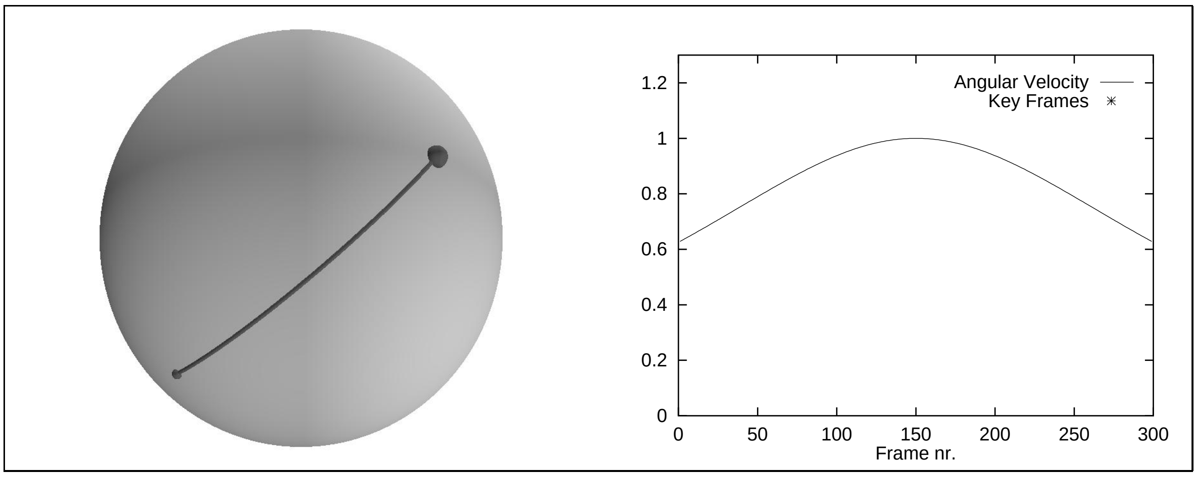


Figure 6.3: *Interpolation curve and velocity graph for linear quaternion interpolation — Lerp. Between the two key frames are 300 interpolated frames.*

to the unit sphere without changing the corresponding rotations. Therefore the interpolation curve is normalized (figure 6.3).

The illustration shows that *Lerp* is the first of the interpolation methods discussed so far to yield a satisfying result. Even though the interpolation curve for *Lerp* resembles the curve for *LinMat* (see figures 6.2 and 6.3) we must emphasize that this does not mean that the interpolations are alike. The illustration for *LinMat* is the result of a series of transformations and not a true image of the interpolation curve.

Even though the interpolation curve for *Lerp* is nice, the velocity graph is not intuitively satisfying. The speedup in the middle is due to the fact that the interpolation curve takes a "short cut" below the surface of the unit sphere. This is not a desired property. The intuitively correct velocity graph for linear interpolation is a constant function.

6.1.4 A summary of linear interpolation

We have attempted simple linear interpolation with the purpose of revealing which rotation representation is most suitable for defining interpolation curves.

The disadvantages of *LinEuler* and *LinMat* are evident. As previously described, Euler angles are not the best definition for rotation and matrices are not an obvious representation. Therefore it is not to be expected that simple linear interpolation between pairs of Euler angles or rotation matrices will result in nice interpolation curves.

In contrast the interpolation curve for *Lerp* is quite nice. The only problem is the varying velocity graph. A constant velocity is not necessarily a requirement for a curve. However, in this case the varying velocity is a problem since it is the result of a flaw in the method. The problem is that the interpolated quaternions are not unit quaternions in general.

For Euler angles and rotation matrices the linear interpolation can result in unacceptable interpolation curves. It does not necessarily follow from this that it is impossible to define a satisfying interpolation curve using these representations. It does, however, imply that it is not possible to define **simple** algorithms yielding satisfying curves. This is a direct consequence of the disadvantages in the Euler angle and rotation matrix representation as discussed in chapter 4.

On the other hand, the very simple *Lerp* is close to being optimal. At this stage we perceive the optimal interpolation curve between two key frames to be the great arc on the quaternion unit sphere between the two corresponding quaternions.

Due to the previous considerations we will refrain from further attempts at deriving interpolation curves based on Euler angles and rotation matrices.

6.1.5 Spherical Linear Quaternion interpolation: *Slerp*

As proven in proposition 18, all quaternions on a line through the origin[3] perform the same rotation. However, we only want to use unit quaternions for rotation since they possess a range of desirable properties[4].

Simple linear quaternion interpolation yields a secant between the two quaternions. Therefore the interpolation function has larger velocity in the middle of the curve (see figure 6.3 and figure 6.4). Apart from this *Lerp* is optimal. An obvious idea is to define an interpolation method yielding the same interpolation curve but where the interpolated quaternions are unit quaternions. Instead of doing simple linear interpolation the curve should follow a great arc on the quaternion unit sphere from one key frame to the other. This is called *great arc interpolation* or **s***pherical* **l***inear* *int***erp***olation* - *Slerp*.

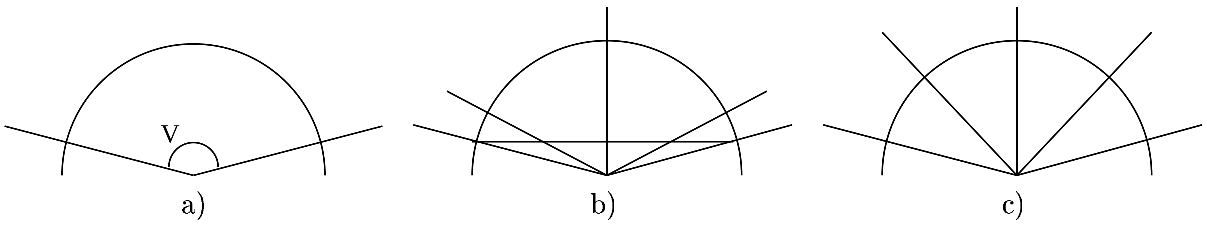


Figure 6.4: *An illustration in the plane of the difference between Lerp and Slerp.* **a)** *The interpolation covers the angle v in three steps.* **b)** *Lerp — The secant across is split in four equal pieces. The corresponding angles are shown.* **c)** *Slerp — The angle is split in four equal angles.*

This interpolation can be stated algorithmically as follows ([Shoemake, 1985], [Shoemake, 1987] and [Shoemake, 1997]). Given $q_0, q_1 \in H_1$ and $h \in [0, 1]$ the following four functions are equivalent expressions for spherical linear interpolation:

[3]Except the origin itself.

[4]The unit quaternion sphere is, as mentioned in chapter 3.4, equivalent to the space of general rotations.

$$\begin{aligned}
Slerp(p,q,h) &= p\ (p^*\ q)^h && (6.4)\\
Slerp(p,q,h) &= (p\ q^*)^{1-h}\ q && (6.5)\\
Slerp(p,q,h) &= (q\ p^*)^h\ p && (6.6)\\
Slerp(p,q,h) &= q\ (q^*\ p)^{1-h} && (6.7)
\end{aligned}$$

Notice the pairwise symmetry yielding the intuitively correct:

$$Slerp(p,q,h) = Slerp(q,p,1-h)$$

The equivalence of the four expressions for *Slerp* is proven in the following proposition inspired by [Shoemake, 1997]. No proof of the equivalence has previously been published (this is confirmed by Ken Shoemake).

Proposition 27.
For $p, q \in H_1, h \in \mathbb{R}$ the following four expressions are equivalent:

$$\begin{array}{ll}
(1) & p\ (p^*\ q)^h\\
(2) & (p\ q^*)^{1-h}\ q\\
(3) & (q\ p^*)^h\ p\\
(4) & q\ (q^*\ p)^{1-h}
\end{array}$$

Proof
First we show (1) = (3). Proposition 20 is used (page 18).

$$\begin{aligned}
p(p^*q)^h &= p(p^*q)^h(p^*p)\\
&= (p(p^*q)^hp^*)p\\
&= (pp^*qp^*)^hp && \text{(Proposition 20)}\\
&= (qp^*)^hp
\end{aligned}$$

Now we show (4) = (2):

$$\begin{aligned}
q(q^*p)^{1-h} &= q(q^*p)^{1-h}(q^*q)\\
&= (q(q^*p)^{1-h}q^*)q\\
&= (q(q^*p)q^*)^{1-h}q && \text{(Proposition 20)}\\
&= (pq^*)^{1-h}q
\end{aligned}$$

Finally we show (2) = (1) using proposition 16 from page 16 and proposition 17:

$$\begin{aligned}
(pq^*)^{1-h}q &= (pq^*)(pq^*)^{-h}q && \text{(Proposition 16)}\\
&= (pq^*)((pq^*)^{-1})^hq && \text{(Proposition 17)}\\
&= pq^*((pq^*)^*)^hq\\
&= pq^*(qp^*)^hq\\
&= p(q^*(qp^*)q)^h && \text{(Proposition 20)}\\
&= p(q^*qp^*q)^h\\
&= p(p^*q)^h
\end{aligned}$$

□

We have thus proved the equivalence of the four expressions for *Slerp*[5]. From now on we will use $Slerp(p,q,h) = p(p^*q)^h$ (equation 6.4).

That *Slerp* does, in fact, perform great arc interpolation on the four-dimensional quaternion sphere is not obvious. Often in the literature it is stated that this follows directly from the Lie group structure of the unit quaternions. We provide a thorough proof in proposition 28 requiring only basic differential geometry.

There are several different ways of proving proposition 28. One approach is to look at the curvature of *Slerp*. It is fairly easy to prove that the curvature equals one throughout the entire interpolation curve. Only great arcs have curvature equal one on a unit sphere. Here we use another approach. The key point in this proof is observing that the curve is a great arc if the second derivative vector is parallel (and with opposite direction) to the position vector of the curve, i. e. $\frac{d^2}{dh^2}Slerp(p,q,h) = c\ Slerp(p,q,h),\ c \leq 0$. This corresponds to the forces acting on an object describing a plane circular motion with constant angular velocity.

Before the proof we need a lemma:

Lemma 3.
Let $p = [s, \mathbf{v}], q_1 = [s_1, (x_1, y_1, z_1)] = [s_1, \mathbf{v}_1], q_2 = [s_2, (x_2, y_2, z_2)] = [s_2, \mathbf{v}_2] \in H$.
Then $(pq_1) \bullet (pq_2) = \|p\|^2\ (q_1 \bullet q_2)$

Proof of lemma 3

$$\begin{aligned}
(pq_1) \bullet (pq_2) &= [ss_1 - \mathbf{v}\cdot\mathbf{v}_1, s\mathbf{v}_1 + s_1\mathbf{v} + \mathbf{v}\times\mathbf{v}_1] \bullet [ss_2 - \mathbf{v}\cdot\mathbf{v}_2, s\mathbf{v}_2 + s_2\mathbf{v} + \mathbf{v}\times\mathbf{v}_2] \\
&= s^2s_1s_2 - ss_1\mathbf{v}\cdot\mathbf{v}_2 - ss_2\mathbf{v}\cdot\mathbf{v}_1 + (\mathbf{v}\cdot\mathbf{v}_1)(\mathbf{v}\cdot\mathbf{v}_2) + \\
&\quad s^2\mathbf{v}_1\cdot\mathbf{v}_2 + ss_2\mathbf{v}\cdot\mathbf{v}_1 + s\mathbf{v}_1\cdot(\mathbf{v}\times\mathbf{v}_2) + \\
&\quad ss_1\mathbf{v}\cdot\mathbf{v}_2 + s_1s_2\mathbf{v}\cdot\mathbf{v} + s_1\mathbf{v}\cdot(\mathbf{v}\times\mathbf{v}_2) + \\
&\quad s(\mathbf{v}\times\mathbf{v}_1)\cdot\mathbf{v}_2 + s_2(\mathbf{v}\times\mathbf{v}_1)\cdot\mathbf{v} + (\mathbf{v}\times\mathbf{v}_1)\cdot(\mathbf{v}\times\mathbf{v}_2) \\
&= s^2s_1s_2 + (\mathbf{v}\cdot\mathbf{v}_1)(\mathbf{v}\cdot\mathbf{v}_2) + \\
&\quad s^2\mathbf{v}_1\cdot\mathbf{v}_2 + s\mathbf{v}_1\cdot(\mathbf{v}\times\mathbf{v}_2) + s_1s_2\mathbf{v}\cdot\mathbf{v} + \\
&\quad s(\mathbf{v}\times\mathbf{v}_1)\cdot\mathbf{v}_2 + (\mathbf{v}\times\mathbf{v}_1)\cdot(\mathbf{v}\times\mathbf{v}_2)
\end{aligned}$$

We simplify using $(\mathbf{v}\times\mathbf{v}_1)\cdot(\mathbf{v}\times\mathbf{v}_2) = (\mathbf{v}\cdot\mathbf{v})(\mathbf{v}_1\cdot\mathbf{v}_2) - (\mathbf{v}\cdot\mathbf{v}_2)(\mathbf{v}_1\cdot\mathbf{v})$:

$$\begin{aligned}
(pq_1) \bullet (pq_2) &= s^2s_1s_2 + (\mathbf{v}\cdot\mathbf{v}_1)(\mathbf{v}\cdot\mathbf{v}_2) + \\
&\quad s^2\mathbf{v}_1\cdot\mathbf{v}_2 + s\mathbf{v}_1\cdot(\mathbf{v}\times\mathbf{v}_2) + s_1s_2\mathbf{v}\cdot\mathbf{v} + \\
&\quad s(\mathbf{v}\times\mathbf{v}_1)\cdot\mathbf{v}_2 + (\mathbf{v}\cdot\mathbf{v})(\mathbf{v}_1\cdot\mathbf{v}_2) - (\mathbf{v}\cdot\mathbf{v}_2)(\mathbf{v}_1\cdot\mathbf{v}) \\
&= (s^2 + \mathbf{v}\cdot\mathbf{v})s_1s_2 + (s^2 + \mathbf{v}\cdot\mathbf{v})\mathbf{v}_1\cdot\mathbf{v}_2 + \\
&\quad s\mathbf{v}_1\cdot(\mathbf{v}\times\mathbf{v}_2) + s(\mathbf{v}\times\mathbf{v}_1)\cdot\mathbf{v}_2 \\
&= \|p\|^2\ (q_1 \bullet q_2) + s\mathbf{v}_1\cdot(\mathbf{v}\times\mathbf{v}_2) + s\mathbf{v}_2\cdot(\mathbf{v}\times\mathbf{v}_1)
\end{aligned}$$

Finally, we use the identity

$$\mathbf{v}\cdot(\mathbf{v}_1\times\mathbf{v}_2) = \begin{vmatrix} x & y & z \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{vmatrix} = xy_1z_2 + x_2yz_1 + x_1y_2z - xy_2z_1 - x_1yz_2 - x_2y_1z$$

[5]Another compelling expression for *Slerp* is $Slerp(p,q,h) = p^{1-h}\ q^h$. This is intuitively analogous to ordinary linear interpolation $p\ (1-h) + q\ h$. The equivalence with equation 6.4 can be shown as follows: $p(p^*q)^h = p(p^{-1}q)^h = pp^{-h}q^h = p^{1-h}q^h$. This is very nice — and yet another example of how easy it is to make erroneous proofs with quaternions. For $q, p \in H$ and $h \in \mathbb{R}$ the equation $(qp)^h = q^h\ p^h$ does not hold in general. This would require commutativity.

We now get:

$$\begin{aligned}(pq_1) \bullet (pq_2) &= \|p\|^2\ (q_1 \bullet q_2) + s\mathbf{v}_1 \cdot (\mathbf{v} \times \mathbf{v}_2) + s\mathbf{v}_2 \cdot (\mathbf{v} \times \mathbf{v}_1) \\ &= \|p\|^2\ (q_1 \bullet q_2) + \\ &\quad s(x_1yz_2 + x_2y_1z + xy_2z_1 - x_1y_2z - xy_1z_2 - x_2yz_1) + \\ &\quad s(x_2yz_1 + x_1y_2z + xy_1z_2 - x_2y_1z - xy_2z_1 - x_1yz_2) \\ &= \|p\|^2\ (q_1 \bullet q_2)\end{aligned}$$

□

Proposition 28.
The curve Slerp$(p, q, h) : H_1 \times H_1 \times [0,1] \curvearrowright H_1$ is a great arc on the unit quaternion sphere between p and q. The position vector function of Slerp has constant angular velocity.

Proof of proposition 28
To show proposition 28 we must prove that the following four conditions are met:

$$\begin{aligned}\mathit{Slerp}(p,q,0) &= p &(6.8)\\ \mathit{Slerp}(p,q,1) &= q &(6.9)\\ \|\mathit{Slerp}(p,q,h)\| &= 1,\ h \in [0..1] &(6.10)\\ \frac{d^2}{dh^2}\mathit{Slerp}(p,q,h) &= c\ \mathit{Slerp}(p,q,h),\ c \leq 0 \in \mathbb{R} &(6.11)\end{aligned}$$

Conditions 6.8 and 6.9 are shown directly using the definitions for exp and log.

$$\begin{aligned}\mathit{Slerp}(p,q,0) &= p\ (p^*\ q)^0 = p\ \exp([0,0]) = p[1,0] = p \\ \mathit{Slerp}(p,q,1) &= p\ (p^*\ q)^1 = p\ \exp(\log(p^*\ q)) \\ &= p\ p^*\ q = p\ p^{-1}\ q = q\end{aligned}$$

Condition 6.10 is met since exp maps into H_1 (definition 15) and since the norm of a product is the product of the norms (proposition 9, equation 3.3):

$$\|\mathit{Slerp}(p,q,h)\| = \|p\|\ \|(p^*\ q)^h\| = 1\ \|\exp(h\ \ \log(p^*\ q))\| = 1$$

To show condition 6.11, we need the second derivative of *Slerp*. Using proposition 23 we find:

$$\begin{aligned}\frac{d}{dt}\mathit{Slerp}(p,q,h) &= \frac{d}{dt}p(p^*q)^h \\ &= p(p^*q)^h \log(p^*q) \\ &= \mathit{Slerp}(p,q,h)\log(p^*q) &(6.12)\\ \frac{d^2}{dh^2}\mathit{Slerp}(p,q,h) &= p\ (p^*\ q)^h\ \log(p^*\ q)^2 \\ &= \mathit{Slerp}(p,q,h)\ \log(p^*\ q)^2\end{aligned}$$

Condition 6.11 holds if $\log(p^*\ q)^2$ is a non-positive real number. Since $p^*, q \in H_1$, then $p^*\ q \in H_1$. By proposition 12 there exists $\theta \in \mathbb{R}$ and $\mathbf{v} \in \mathbb{R}^3, |\mathbf{v}| = 1$ such that $p^*\ q = [\cos\theta, \sin\theta\mathbf{v}]$. Then:

$$\begin{aligned}\log(p^*\ q)^2 &= [0, \theta\mathbf{v}]^2 \\ &= [-\theta^2\ \mathbf{v}\cdot\mathbf{v}, \theta^2\ \mathbf{v}\times\mathbf{v}] \\ &= [-\theta^2, \mathbf{0}]\end{aligned}$$

Thus $\frac{d^2}{dh^2}\mathit{Slerp}(p,q,h) = c\ \mathit{Slerp}(p,q,h)$ where $c = -\theta^2 \leq 0$.

□

Having shown that $\mathit{Slerp}(p,q,h), h \in [0,1]$ spans a great arc between p and q, there are still two possible curves depending on which direction around the unit sphere *Slerp* takes. The following proposition states that *Slerp* behaves as desired.

Proposition 29.
Let $p, q \in H_1$. Then $\mathit{Slerp}(p,q,h),\ h \in [0,1]$, spans the shortest great arc between p and q on the unit quaternion sphere.

Proof of proposition 29
Let $q_{\frac{1}{2}} = \mathit{Slerp}(p,q,\frac{1}{2})$ and let α denote the angle between p and $q_{\frac{1}{2}}$. *Slerp* yields the shortest arc if and only if $\alpha \in\ ]-\frac{\pi}{2}, \frac{\pi}{2}]$. This is equivalent to $\cos(\alpha) \in [0,1]$. We therefore examine the sign of $\cos(\alpha)$.

Let $p, q \in H_1$, where $p = [s, v]$.

$$\begin{aligned}\cos(\alpha) &= p \bullet q_{\frac{1}{2}} && \text{(Proposition 10)} \\ &= p \bullet \mathit{Slerp}\,(p,q,1/2) \\ &= p \bullet (p\ (p^*q)^{\frac{1}{2}})\end{aligned}$$

Since $p^*, q \in H_1$ it follows that $p^*q \in H_1$. By proposition 12 there exists $\mathbf{w} \in \mathbb{R}^3$, $|\mathbf{w}| = 1$ and $\psi \in\ ]-\pi,\pi]$ such that $p^*q = [\cos(\psi), \sin(\psi)\mathbf{w}]$. Using lemma 3 we get:

$$\begin{aligned}\cos\alpha &= p \bullet \left(p\ [\cos(\psi), \sin(\psi)\mathbf{w}]^{1/2}\right) \\ &= p \bullet (p\ \exp((1/2)\log[\cos(\psi), \sin(\psi)\mathbf{w}])) \\ &= p \bullet (p\ \exp([0, (\psi/2)\mathbf{w}])) \\ &= p \bullet (p\ [\cos(\psi/2), \sin(\psi/2)\,\mathbf{w}]) \\ &= (p\ [1,\mathbf{0}]) \bullet (p\ [\cos(\psi/2), \sin(\psi/2)\,\mathbf{w}]) \\ &= ||p||^2([1,\mathbf{0}] \bullet [\cos(\psi/2), \sin(\psi/2)\,\mathbf{w}]) && \text{(Lemma 3)} \\ &= ||p||^2\cos(\psi/2) \\ &= \cos(\psi/2)\end{aligned}$$

Now $\psi \in\ ]-\pi,\pi]$ yields $\cos(\psi/2) \geq 0$ and therefore $\cos(\alpha) \geq 0$. Thus *Slerp* spans the shortest great arc between p and q.

□

We have now proven the equivalence of the four expressions for *Slerp* from proposition 27 and then proven that *Slerp* actually produces the desired great arc. This could conclude our treatment of *Slerp*. However, the literature has traditionally avoided the use of exponentiation in the expression for *Slerp*[6]. We have encountered no problems using the expressions from proposition 27. However, for the sake of completeness, we will include the following expression for *Slerp* without exponentiation:

$$\begin{aligned} \cos(\Omega) &= q_0 \bullet q_1 \\ Slerp(q_0, q_1, h) &= \frac{q_0 \sin((1-h)\Omega) + q_1 \sin(h\Omega)}{\sin(\Omega)} \end{aligned} \tag{6.13}$$

Notice that this expression is not defined for $q_0 = \pm q_1$. The obvious patch is $Slerp(q, q, h) \equiv q$.

The correctness of the expression above (equation 6.13) can be shown in the plane. The interpolation between p_0 and p_1 (as illustrated in figure 6.5) can be written:

$$q(h) = \begin{pmatrix} \cos(v+ht) \\ \sin(v+ht) \end{pmatrix}$$

The expression from equation 6.13 can — through applying the addition formulas for sin and cos successively — be written as:

$$\begin{aligned} Slerp(p_0, p_1, h) &= \frac{p_0 \sin((1-h)t) + p_1 \sin(ht)}{\sin(t)} \\ &= \begin{pmatrix} \frac{\cos(v)\sin((1-h)t) + \cos(v+t)\sin(ht)}{\sin(t)} \\ \frac{\sin(v)\sin((1-h)t) + \sin(v+t)\sin(ht)}{\sin(t)} \end{pmatrix} \\ &= \begin{pmatrix} \frac{\cos(v)(\sin(t)\cos(ht) - \cos(t)\sin(ht)) + (\cos(v)\cos(t) - \sin(v)\sin(t))\sin(ht)}{\sin(t)} \\ \frac{\sin(v)(\sin(t)\cos(ht) - \cos(t)\sin(ht)) + (\sin(v)\cos(t) + \cos(v)\sin(t))\sin(ht)}{\sin(t)} \end{pmatrix} \\ &= \begin{pmatrix} \cos(v)\cos(ht) - \sin(v)\sin(ht) \\ \sin(v)\cos(ht) + \cos(v)\sin(ht) \end{pmatrix} \\ &= \begin{pmatrix} \cos(v+ht) \\ \sin(v+ht) \end{pmatrix} \\ &= q(h) \end{aligned}$$

Thus, the correctness of the expression has been proven in the plane. This result can be generalized directly to four dimensions thereby proving equation 6.13.

***Slerp* summarized**

The interpolation curve for *Slerp* (figure 6.6) forms a great arc on the quaternion unit sphere (as proven in proposition 28). In differential geometry terms, the great arc is a *geodesic* — corresponding to a straight line. Not only does *Slerp* follow a great arc (as proven in proposition 29) it follows the shortest great arc. Thus *Slerp* yields the shortest possible interpolation path between the two quaternions on the unit sphere[7]. Furthermore *Slerp* has constant angular velocity. All in all *Slerp* is the optimal interpolation curve between two rotations.

[6]Since $q^h = \exp(h \log q)$ exponentiation automatically implies the use of the logarithm and exponential functions. These functions are only defined on a limited set of quaternions and they can therefore cause problems in conjunction with numerical inaccuracies.

[7]It should be noted that even though *Slerp* performs the shortest possible arc between p and q this is not nec-

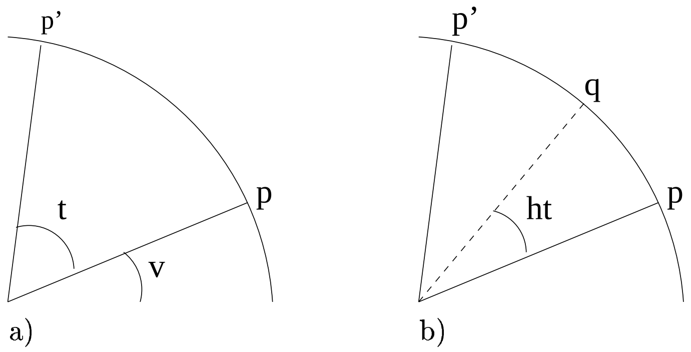


Figure 6.5: *Slerp in the plane.* **a)** *The interpolation goes from p to p' across the angle t.* **b)** *A step in the interpolation, where $h \in [0, 1]$, q moves from p to p'.*

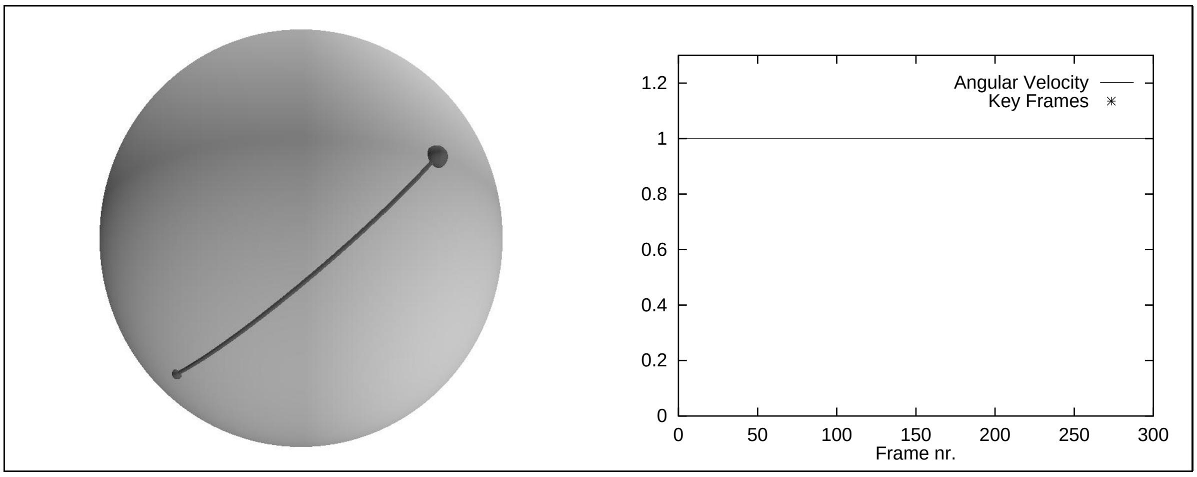


Figure 6.6: *Interpolation curve and velocity graph for spherical linear quaternion interpolation – Slerp. Between the two key frames there are 300 interpolated frames.*

essarily optimal. Since p and $-p$ perform the same rotation (according to equation 18 page 17), the interpolation between $-p$ and q could possibly yield a shorter interpolation path. This can be established simply by comparing the distance between p and q, $||p - q||$, with the distance between $-p$ and q, $||p + q||$.

6.2 Interpolation over a series of rotations: Heuristic approach

When interpolating between two rotations *Slerp* is optimal. In the set of unit quaternions the interpolation curve of *Slerp* is equivalent to a straight line (the great arc). When interpolating between a series of rotations problems emerge: a) The curve is not smooth at the control points, b) The angular velocity is not constant and c) The angular velocity is not continuous at the controls points.

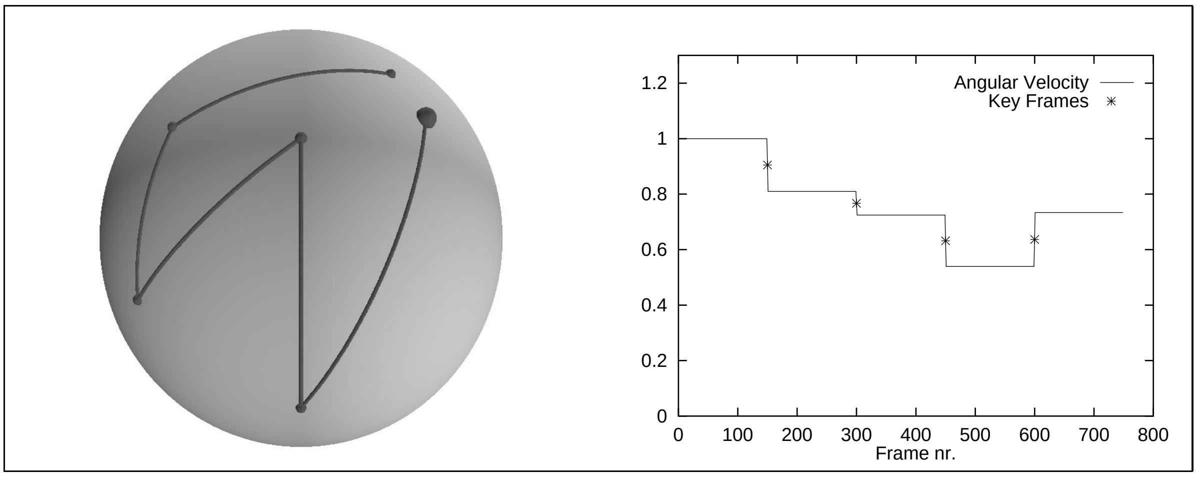


Figure 6.7: *Interpolation curve and angular velocity graph for Slerp. Between the six key frames, 750 interpolated frames have been generated.*

A reparameterization can easily ensure continuity across the entire interpolation. Actually the interpolation parameter is transformed into a number of discrete frames between each pair of key frames. Thus a reparameterization corresponds to assigning each interval a number of frames relative to the size of the interval. The size of an interval can be measured as the angle θ between a pair of key frames q_i and q_{i+1}, given by $\cos\theta = q_i \bullet q_{i+1}$.

Since the number of frames in each subinterval necessarily has to be an integer the angular velocity is, due to rounding, only approximately constant. Compare figure 6.7 and figure 6.8.

It is not equally simple to fix the lack of smoothness. Analogously it is simple to interpolate between two points in the plane with a straight line, but even in the simple Euclidean space it is relatively complicated to create a smooth interpolation between a series of points (see figure 6.9).

When interpolating between a series of control points in the plane different kinds of cubic curves are typically used. For example this can be done with Bézier curves, which can be constructed quite simply.

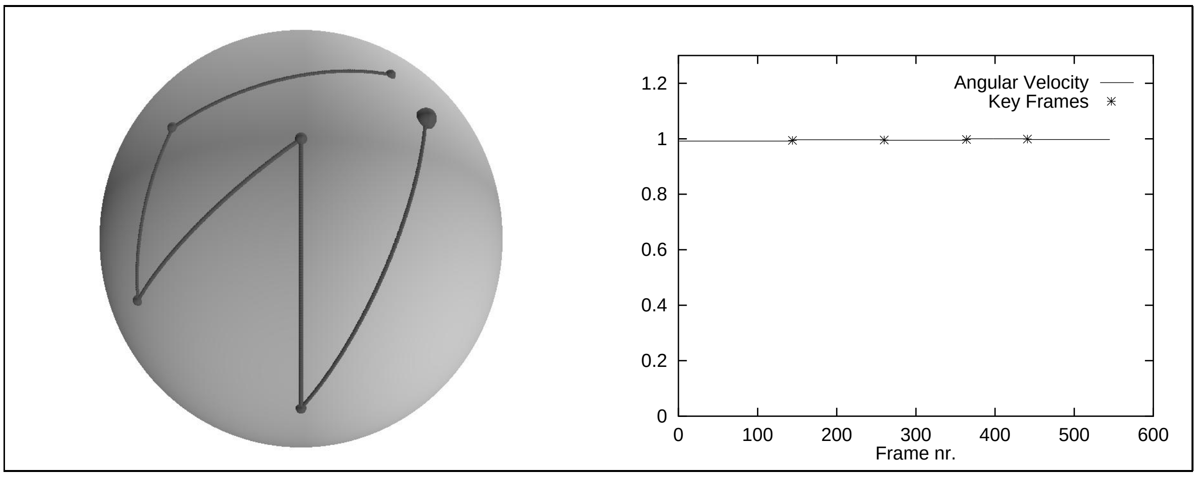


Figure 6.8: *Interpolation curve and angular velocity graph for Slerp. Between the six key frames, 550 interpolated frames have been generated. The frames in the subintervals are distributed according to length of the interval.*

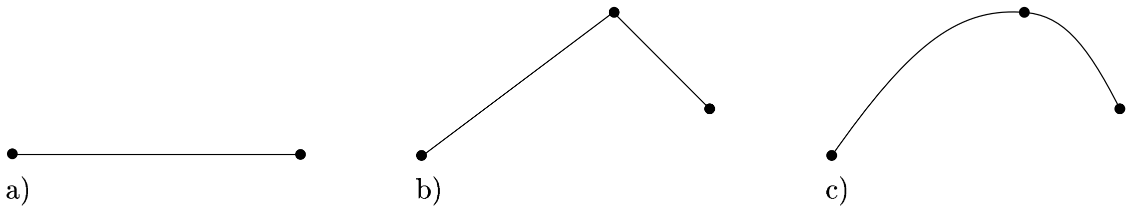


Figure 6.9: **a)** *In the plane simple interpolation between two points is obtained by a straight line.* **b)** *Linear interpolation between a series of points is not differentiable in the control points.* **c)** *To ensure differentiability one can use cubic curves, for example splines.*

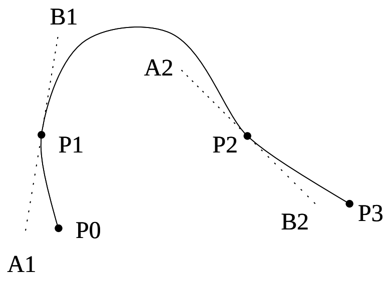


Figure 6.10: *Interpolation between the points* $P1$ *and* $P2$ *with a Bézier curve. The curve is defined as a third-order curve, where the tangent in the control points is defined by auxiliary points. For example the tangent in* $P1$ *is defined by the auxiliary points A1 and B1 (the tangent is B1-P1 or P1-A1). The differentiability is automatically assured since the curve is a third order curve.*

The Bézier curve from figure 6.10 (with auxiliary points B_1 and A_2) that interpolates between the control points P_1 and P_2 can be expressed algorithmically (based on [Watt & Watt, 1992]) as three steps of linear interpolation:

$$\begin{aligned} lin(x_0, x_1, h) &= x_0(1-h) + x_1 h \\ Bezier(P_1, P_2, B_1, A_2, h) &= lin(lin(P_1, P_2, h), lin(B_1, A_2, h), 2h(1-h)) \end{aligned}$$

The auxiliary points can be moved arbitrarily, which yields a change in the shape of the curve. The interpolation curve between P1 and P2 is solely determined from the positions of auxiliary points B1, A2 relative to control points P1 and P2. The tangent at P1 is defined by the vector B1-P1 and the tangent at P2 is defined by the vector A2-P2.

When interpolating between a series of control points, it is often desirable to ensure differentiability in the control points. This constraint can be met by making the tangents coincide in the control points, i.e. ensuring that B1-P1 = P1-A1.

6.2.1 Spherical Spline Quaternion interpolation: *Squad*

The above construction can serve as an inspiration for formulating the spherical cubic equivalent of a Bézier curve. This interpolation curve is called *Squad* (**s***pherical* and **quad***rangle*) and was presented by Shoemake in [Shoemake, 1987].

Shoemake defines *Squad* as (with $h \in [0,1]$):

Definition 17.

$$\begin{aligned} Squad(q_i, q_{i+1}, s_i, s_{i+1}, h) &= Slerp(Slerp(q_i, q_{i+1}, h), Slerp(s_i, s_{i+1}, h), 2h(1-h)) \quad (6.14) \\ s_i &= q_i \exp\left(-\frac{\log(q_i^{-1}q_{i+1}) + \log(q_i^{-1}q_{i-1})}{4}\right) \quad (6.15) \end{aligned}$$

The resulting expression for *Squad* is analogous to the Bézier curve, but involves spherical linear interpolation instead of simple linear interpolation. B1 and A2 are written s_i and s_{i+1}. The expression for s_i (equation 6.15) will be derived below.

Correctness of *Squad*

The definition of *Squad* is complex and therefore neither the continuity nor the differentiability of the resulting interpolation curve is obvious.

Squad was originally presented in [Shoemake, 1987] which has served as general reference for a proof of the differentiability of *Squad*. [Shoemake, 1987] is no longer available[8], and furthermore

[8][Shoemake, 1987] is a set of course notes from SIGGRAPH 1987, and these notes are no longer available from University libraries or ACM.

the original proof of differentiability was flawed[9]. In [Kim et al., 1996] a new proof was presented. However, the differentiability of *Squad* is a consequence of a more general result in this paper and therefore the proof is not very thorough. In addition, the constants s_i from proposition 17 were not derived. After corresponding with Ken Shoemake [Shoemake, 1997] we have therefore derived a complete proof of the differentiability of *Squad*.

Proposition 30.
$\mathit{Squad} \in C^1$

Proof
That *Squad* is continuously differentiable is obvious except at the control points, since all the subexpressions are continuously differentiable in a given sub-interval and therefore *Squad* is continuously differentiable inside each interval.

We must now show continuous differentiability for *Squad* at a given control point q_i. First we must show that the neighboring segments have the control points as their value at the end points, i. e. that $\mathit{Squad}(q_{i-1}, q_i, s_{i-1}, s_i, 1) = \mathit{Squad}(q_i, q_{i+1}, s_i, s_{i+1}, 0)$:

$$\begin{aligned}\mathit{Squad}(q_{i-1}, q_i, s_{i-1}, s_i, 1) &= \mathit{Slerp}(\mathit{Slerp}(q_{i-1}, q_i, 1), \mathit{Slerp}(s_{i-1}, s_i, 1), 0)\\ &= \mathit{Slerp}(q_i, s_i, 0)\\ &= q_i\end{aligned}$$

$$\begin{aligned}\mathit{Squad}(q_i, q_{i+1}, s_i, s_{i+1}, 0) &= \mathit{Slerp}(\mathit{Slerp}(q_i, q_{i+1}, 0), \mathit{Slerp}(s_i, s_{i+1}, 0), 0)\\ &= \mathit{Slerp}(q_i, s_i, 0)\\ &= q_i\end{aligned}$$

Thus *Squad* is continuous and has the correct value at the control points.

We now show that *Squad* is continuously differentiable at a given control point. We do this by deriving the derivative of *Squad* in a given interval. Like above, we must then show that

$$\frac{d}{dt}\mathit{Squad}(q_{i-1}, q_i, s_{i-1}, s_i, 1) = \frac{d}{dt}\mathit{Squad}(q_i, q_{i+1}, s_i, s_{i+1}, 0)$$

To find the derivative of *Squad*, we need the derivative of *Slerp*, which we get from equation 6.12.

We introduce the abbreviation

$$g_i(h) = \mathit{Slerp}(q_i, q_{i+1}, h)^* \mathit{Slerp}(s_i, s_{i+1}, h)$$

Now we will find the derivative of $\mathit{Squad}(q_i, q_{i+1}, s_i, s_{i+1}, h)$ and decide how s_i and s_{i+1} must be defined to ensure differentiability at the control points.

$$\begin{aligned}\frac{d}{dt}\mathit{Squad}(q_i, q_{i+1}, s_i, s_{i+1}, h) &= \frac{d}{dt}\mathit{Slerp}(\mathit{Slerp}(q_i, q_{i+1}, h), \mathit{Slerp}(s_i, s_{i+1}, h), 2h(1-h))\\ &= \frac{d}{dt}\left(\mathit{Slerp}(q_i, q_{i+1}, h) \;\; g_i(h)^{2h(1-h)}\right)\end{aligned}$$

[9]According to Ken Shoemake.

The product rule for differentiation (proposition 24) yields:

$$\begin{aligned}
\frac{d}{dt}\mathit{Squad}(q_i, q_{i+1}, s_i, s_{i+1}, h) &= \frac{d}{dt}\left(\mathit{Slerp}(q_i, q_{i+1}, h) g_i(h)^{2h(1-h)}\right) \\
&= \left(\frac{d}{dt}\left(\mathit{Slerp}(q_i, q_{i+1}, h)\right)\right) g_i(h)^{2h(1-h)} + \\
&\quad \mathit{Slerp}(q_i, q_{i+1}, h)\left(\frac{d}{dt}(g_i(h)^{2h(1-h)})\right) \\
&= \mathit{Slerp}(q_i, q_{i+1}, h)\log(q_i^* q_{i+1}) g_i(h)^{2h(1-h)} + \\
&\quad \mathit{Slerp}(q_i, q_{i+1}, h)\left(\frac{d}{dt} g_i(h)^{2h(1-h)}\right)
\end{aligned}$$

Since $g_i(h)$ is a product of unit quaternions, the function values are on the unit sphere. Therefore, $g_i(h)$ may be written:

$$g_i(h) = [\cos(\theta_{g_i(h)}), \sin(\theta_{g_i(h)})\mathbf{v}_{g_i(h)}]$$

Here $\mathbf{v}_{g_i(h)}$ is a unit vector. We can now use proposition 26 to find the derivative of $g_i(h)^{2h(1-h)}$:

$$\begin{aligned}
\frac{d}{dt} g_i(h)^{2h(1-h)} &= \Big[-\sin\left(2h(1-h)\theta_{g_i(h)}\right)\left(\frac{d}{dt}(2h(1-h))\theta_{g_i(h)} + 2h(1-h)\frac{d}{dt}(\theta_{g_i(h)})\right), \\
&\quad \cos\left(2h(1-h)\theta_{g_i(h)}\right)\left(\frac{d}{dt}(2h(1-h))\theta_{g_i(h)} + 2h(1-h)\frac{d}{dt}(\theta_{g_i(h)})\right)\mathbf{v}_{g_i(h)} + \\
&\quad \sin\left(2h(1-h)\theta_{g_i(h)}\right)\frac{d}{dt}(\mathbf{v}_{g_i(h)})\Big] \\
&= \Big[-\sin\left(2h(1-h)\theta_{g_i(h)}\right)\left((2-4h)\theta_{g_i(h)} + 2h(1-h)\theta_{g_i'(h)}\right), \\
&\quad \cos\left(2h(1-h)\theta_{g_i(h)}\right)\left((2-4h)\theta_{g_i(h)} + 2h(1-h)\theta_{g_i'(h)}\right)\mathbf{v}_{g_i(h)} + \\
&\quad \sin\left(2h(1-h)\theta_{g_i(h)}\right)\mathbf{v}_{g_i'(h)}\Big]
\end{aligned}$$

Having expanded all the subexpressions of the derivative of *Squad*, we will now determine s_i so that the derivative of *Squad* is continuous across each control point, i. e.

$$\frac{d}{dt}\mathit{Squad}(q_{i-1}, q_i, s_{i-1}, s_i, 1) = \frac{d}{dt}\mathit{Squad}(q_i, q_{i+1}, s_i, s_{i+1}, 0)$$

Below we write $\frac{d}{dt}\left(g_{i-1}(h)^{2h(1-h)}\right)(1)$ for the derivative of the expression $g_{i-1}(h)^{2h(1-h)}$ applied to the value 1. Using algebra and rearranging, we get:

$$\begin{aligned}
\frac{d}{dt}\mathit{Squad}(q_{i-1}, q_i, s_{i-1}, s_i, 1) &= \mathit{Slerp}(q_{i-1}, q_i, 1)\log(q_{i-1}^* q_i) + \\
&\quad \mathit{Slerp}(q_{i-1}, q_i, 1)\frac{d}{dt}\left(g_{i-1}(h)^{2h(1-h)}\right)(1) \\
&= q_i \log(q_{i-1}^* q_i) + q_i[0, -2\ \theta_{g_{i-1}}(1)\mathbf{v}_{g_{i-1}}(1)] \\
&= q_i\left(\log\left(q_{i-1}^* q_i\right) - 2\log([\cos(\theta_{g_{i-1}}(1)), \sin(\theta_{g_{i-1}}(1))\mathbf{v}_{g_{i-1}}(1)])\right) \\
&= q_i(\log(q_{i-1}^* q_i) - 2\log(g_{i-1}(1))) \\
&= q_i(\log(q_{i-1}^* q_i) - 2\log(q_i^* s_i))
\end{aligned}$$

$$\begin{aligned}
\frac{d}{dt}\mathit{Squad}(q_i, q_{i+1}, s_i, s_{i+1}, 0) &= \mathit{Slerp}(q_i, q_{i+1}, 0)\log(q_i^* q_{i+1}) + \\
&\quad \mathit{Slerp}(q_i, q_{i+1}, 0)\frac{d}{dt}\left(g_i(h)^{2h(1-h)}\right)(0) \\
&= q_i \log(q_i^* q_{i+1}) + q_i[0, 2\ \theta_{g_i}(0)\mathbf{v}_{g_i}(0)] \\
&= q_i \left(\log\left(q_i^* q_{i+1}\right) + 2\log([\cos(\theta_{g_i}(0)), \sin(\theta_{g_i}(0))\mathbf{v}_{g_i}(0)])\right) \\
&= q_i(\log(q_i^* q_{i+1}) + 2\log(g_i(0))) \\
&= q_i(\log(q_i^* q_{i+1}) + 2\log(q_i^* s_i))
\end{aligned}$$

Thus, s_i must satisfy

$$q_i(\log(q_i^* q_{i+1}) + 2\log(q_i^* s_i)) = q_i(\log(q_{i-1}^* q_i) - 2\log(q_i^* s_i)).$$

Using $q_i^* = q_i^{-1}$ since $q_i \in H_1$ we get:

$$\begin{aligned}
4\log(q_i^* s_i) &= \log(q_{i-1}^* q_i) - \log(q_i^* q_{i+1}) \\
q_i^* s_i &= \exp\left(\frac{\log(q_{i-1}^* q_i) - \log(q_i^* q_{i+1})}{4}\right) \\
s_i &= q_i \exp\left(\frac{\log(q_{i-1}^* q_i) - \log(q_i^* q_{i+1})}{4}\right)
\end{aligned}$$

To rewrite the expression for s_i, we use the identity $(q_1 q_2)^* = q_2^* q_1^*$. Since the constituent quaternions are unit quaternions, the identities $q^* = q^{-1}$, and $\log(q^*) = -\log(q)$ also hold. Finally we have:

$$\begin{aligned}
s_i &= q_i \exp\left(-\frac{\log(q_i^* q_{i-1}) + \log(q_i^* q_{i+1})}{4}\right) \\
&= q_i \exp\left(-\frac{\log(q_i^{-1} q_{i-1}) + \log(q_i^{-1} q_{i+1})}{4}\right) \qquad (6.16)
\end{aligned}$$

Thus *Squad* is continuously differentiable at the control points with s_i defined as above. All in all we have shown that *Squad* is continuous and continuouly differentiable across all segments. Further observe that the derived equation 6.16 for s_i is the same as equation 6.15.

□

The interpolation curve generated by *Squad*

The algorithmic expression for *Squad* yields an interpolation curve for a series of quaternions $q_0, \ldots, q_N$. The expression is not defined in the first and last interval since q_{-1} appears in the expression for s_0 and q_{n+1} appears in the expression for s_n. Therefore it is necessary to define sound values for s_0 and s_n. The simplest solution is to define $s_0 \equiv q_0$ and $s_N \equiv q_N$ — alternatively q_{-1} and q_{n+1} can be defined. The choice of s_0 and q_0 have little impact on the resulting interpolation curve and we will consider the choice an implementation detail.

As for the interpolation curve of *Slerp* (figure 6.8) it is, for implementation purposes, necessary to produce a discrete version of the interpolation parameter and thereby selecting the number of interpolated frames between each key frame. For *Slerp* this process was simple since the arc length of the interpolation curve corresponds to the angle between the two involved quaternions.

Since the interpolation curve for *Squad* is rounded, it is not simple to calculate the arc length between each pair of key frames and thus it is not trivial to determine the number of frames between each pair of key frames. We choose to determine the number of frames between each pair of key frames relative to the distance between the the two key frames. This is not the optimal choice, but a simple and effective heuristic.

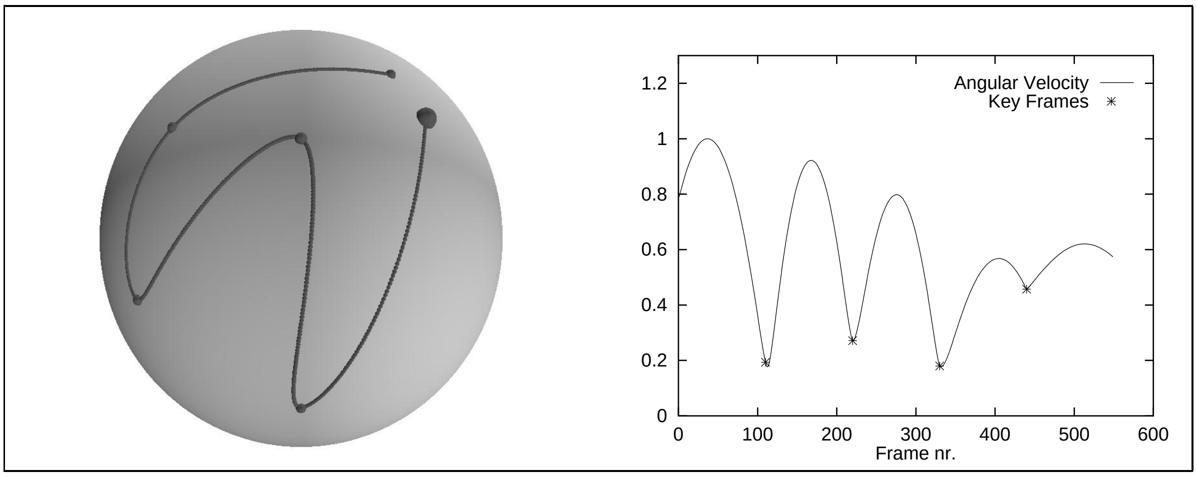


Figure 6.11: *Interpolation curve and angular velocity graph for Squad. Between the six key frames we have interpolated 550 frames. The frames have been distributed according to interval length.*

From figure 6.11 it is clear that *Squad* gives a “nice” interpolation curve. The term “nice” can be read as continuous and differentiable — but even this clarification is qualitatively vague: A continuous and differentiable curve can have any number of more or less wild twists and turns. From the formulation of *Squad* it is far from trivial to determine qualitative properties of the curve. Therefore we want a more objective measure from which we can **define** an interpolation curve. In this context it is no longer adequate to use qualified guesses to derive new methods of interpolation. However, the previously stated methods provide a good foundation for the development of a more general method.

In the next sections we will seek the formulation of a more general method from a more mathematical and physical point of view.

6.3 Interpolation between a series of rotations: Mathematical approach

So far the interpolation methods have been fairly simple and based on the rotation representations. In principle, the interpolation is independent of which rotational modality is used to implement the method. Instead, the optimal interpolation curve should be defined from the desired properties in the space of rotations. This optimal curve can, of course, be written algorithmically for any sensible representation of rotation.

The above point can be exemplified for interpolation between two rotations. The optimal interpolation curve is the equivalent of a straight line in the space of rotations. This curve can be written algorithmically using Euler angles, but the advantage of using quaternions is that the curve can be stated simply — using *Slerp*. This is due to the previously described equivalence between the space of rotations and the unit quaternion sphere.

Therefore, we will base our discussion below on the space of rotations. We will give mathematically based demands for the optimal interpolation curve. The goal is to give an algorithmic description of the optimal interpolation curve.

6.3.1 The interpolation curve

Interpolation between rotations is defined in the space of rotations $SO(3)$. As mentioned earlier, however, $SO(3)$ and the set, H_1, of unit quaternions are topologically equivalent. We therefore choose to define the general interpolation in the space of unit quaternions.

Definition 18.
Given k control points $q_i \in H_1$ and $\mathbf{I} = [t_1, t_k]$, the interpolation curve $\gamma(t) : \mathbf{I} \curvearrowright H_1$, is constrained by $\gamma(t_i) \equiv q_i$ for $t_i \in \mathbf{I}$. We require that $t_1 \leq t_2, \ldots, t_{k-1} \leq t_k$.

6.3.2 Definitions of smoothness

A natural requirement is that the interpolation curve is "nice." This vague term usually means **smooth** in differential geometry. However, several different definitions of smooth exist. We mention the following:

Definition 19.
Let $\gamma(t)$ be the parameterization of a curve in $C^n(\mathbf{I}, \mathbb{R}^n)$. **Smooth** *can then be defined in the following ways:*

[Madsen, 1991]:	*The curve $\gamma(t)$ is* **smooth** *if: $\gamma(t) \in C^1$ and $\forall t \in \mathbf{I} : \gamma'(t) \neq 0$.*
[Schwarz, 1989]:	*The curve $\gamma(t)$ is* **smooth** *if: $\gamma(t) \in C^2$.*
[Jakobsen, 1993]:	*The curve $\gamma(t)$ is* **smooth** *if: $\gamma(t) \in C^\infty$.*

The different definitions express that it is not immediately obvious what "nice" is. We must therefore examine more closely which properties we want the interpolation curve to have.

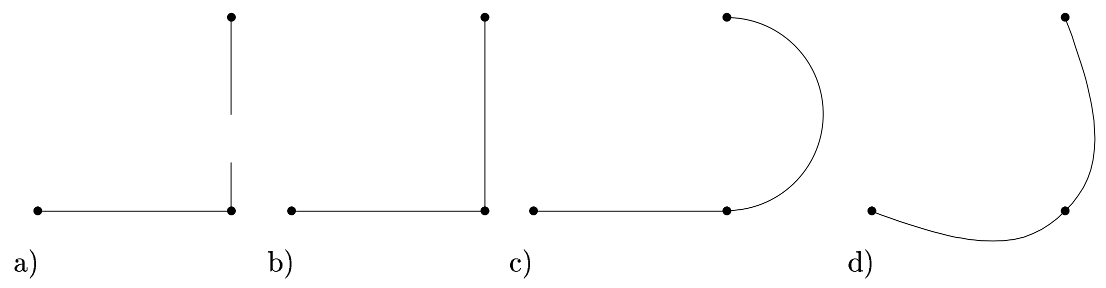


Figure 6.12: *Interpolation between three control points in the plane.* **a)** *Discontinuous interpolation curve.* **b)** *Continuous curve.* **c)** C^1*-curve.* **d)** C^2*-curve.*

The curve must obviously be continuous. The curve must also be differentiable. We do not want either “holes” or “breaks” in an animation. Thus we demand that the interpolation curve must be C^1. However, it is not as obvious whether the curve should be C^2 or, for that matter, C^∞. Figure 6.12 illustrates the different classes of curves in the plane.

Since the control points are symmetric, it is natural to expect that the interpolation curve is symmetric. The illustrations in the plane clearly show that we must demand C^2 over C^1. However, illustrations in the plane are not adequate ground to base this choice on, and we therefore postpone this decision (see section 6.3.7).

In the first definition we find the requirement $\gamma'(t) \neq 0$. Thus, the interpolation function is not allowed to contain singularities, i. e. the interpolation must not “stop.” Offhand, this seems to be a sensible demand. However, it is possible to make sensible but contradictory demands to the speed of the interpolation curve. Consider, for example, animating a pendulum. The control points (that define the angle that the pendulum oscillates through) will lie on a straight line in the space of rotations. At the outer positions, it is to be expected that the pendulum has no velocity, corresponding to a singularity in the interpolation function. We therefore also postpone this decision until section 6.3.7.

This discussion of smoothness does not bring us much further. The definitions above will not even allow us to differentiate between *LinEuler*, *Lerp* and *Slerp*. These curves are all C^2 except at the control points, where they are C^0. Thus, it is not sufficient just to describe which class of functions the interpolation curve should belong to.

6.3.3 The optimal interpolation

Smoothness considerations do not give adequate requirements to the definition of the interpolation curve. We need an objective measure of how “nice” our curve is in rotation space.

We will again seek inspiration in the plane (see figure 6.13). As mentioned earlier, cubic curves are usually used to interpolate between a series of points. There are many kinds of cubic curves; the Bézier curve described in section 6.2 is an example. The most common class of curves is *splines*.

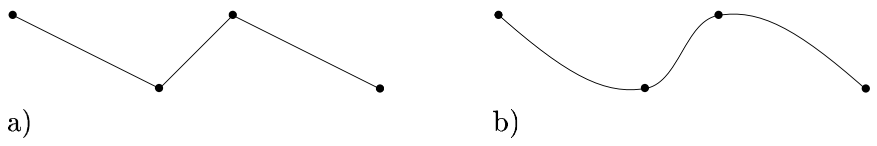


Figure 6.13: *Interpolating with a spline in the plane.* **a)** *Simple linear interpolation.* **b)** *Interpolation with a spline.*

Discussing splines we must, as any serious project dealing with splines, write a bit about ship builders. Traditionally, when ship builders wanted to decide how to shape the curved parts of a ship, a flexible piece of metal was used. The piece of metal was fixed between a series of rivets. The metal piece then adapted itself to the rivets, making a nice soft curve. The ship builders called this tool a *spline*[10].

Viewed physically, the above description of a spline corresponds to the piece of metal achieving a minimum of inner tension forces subject to the constraints (the rivets). Mathematically, the metal piece minimizes curvature.

In the plane, the curve can be described as follows. Given the control points $(x_i, y_i) \in \mathbb{R}^2$, define $\gamma(t) \in C^2(\mathbf{I}, \mathbb{R}^2)$, where t is the natural parameter[11], such that $\gamma(t)$ passes through the control points and at the same time minimizes the expression $\int_I \|\gamma''(t)\|^2 dt$. Thus the square of the curvature is minimized.

This simple formulation gives a non-ambiguous definition of the interpolation curve from a general concept in differential geometry. We therefore choose to view the curve that minimizes (the square of the) curvature as the optimal interpolation curve. As we shall see below, it is not as simple to compute this curve in H_1 as it is in the plane.

6.3.4 Curvature in H_1

The interpolation curve lies on H_1, which is a hypersphere in quaternion space. Normally the curvature for a curve $\gamma(t)$ is defined as $\|\gamma''(t)\|$, assuming that t is the natural parameter.

The interpolation curve *Slerp* yields a great arc on the quaternion unit sphere. A great arc in H_1 is equivalent to a straight line in the plane. Thus it is to expected be that a great arc does not have any curvature. If the curvature is computed by $\|\gamma''(t)\|$, the curvature will not be zero, but one: The curvature of the unit sphere. We therefore want to compute the curvature relative to the quaternion unit sphere, and not relative to quaternion space. We will call this curvature the *local curvature.*

The definition of local curvature for a curve that lies on a surface is based in differential geometry. The local curvature in a point is defined as follows. Given the point on the surface, a coordinate system (a map) is placed in the tangent plane. The local curvature of the curve is now the curvature of the curve projected onto the tangent plane. This is also called *tangential curvature.*

[10]The ambitious project will obviously also note that modern-day architects use a refined version of the ships-builder's spline to draw curves. According to an architect, however, this is a myth.
[11]i. e. the parameter defined from the curve length.

In differential geometry a great arc (the "straight line") is called a geodesic. Projected onto the tangent plane, the geodesic becomes a straight line. Thus we see, as expected, that a great arc does not have any local curvature.

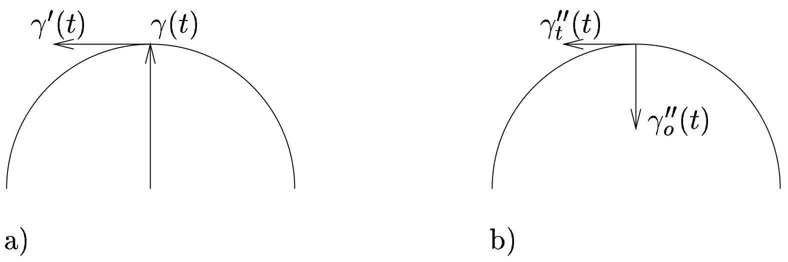


Figure 6.14: *The division of $\gamma''(t)$ — an analogy in the plane.* **a)** *The position vector for the curve $\gamma(t)$ lies on the surface of the quaternion unit sphere. The tangential plane is orthogonal to the position vector.* **b)** *$\gamma''(t)$ can be split in a component $\gamma''_t(t)$ (in the tangential plane) and a component $\gamma''_o(t)$ (parallel with the position vector).*

If a curve $\gamma(t)$ lies on the surface of H_1, we can split $\gamma''(t)$ into two parts (see figure 6.14): a component, $\gamma''_t(t)$, in the tangential plane , and a component, $\gamma''_o(t)$, orthogonal to the tangential plane. The desired part of the curvature is $\gamma''_t(t)$. Thus, the local curvature κ of the curve γ can be obtained:

$$\kappa(\gamma, t) = \|\gamma''_t(t)\| = \|\gamma''(t) - \gamma''_o(t)\|$$

Since $\gamma(t)$ lies on the surface of the unit sphere, $\gamma''_o(t)$ will be parallel to $\gamma(t)$. Thus we can find $\gamma''_o(t)$ by projecting $\gamma''(t)$ onto $\gamma(t)$. Thus:

$$\begin{aligned}\gamma''_t(t) &= \gamma''(t) - \gamma''_o(t) \\ &= \gamma''(t) - \left(\gamma''(t) \bullet \frac{\gamma(t)}{\|\gamma(t)\|}\right) \frac{\gamma(t)}{\|\gamma(t)\|} \\ &= \gamma''(t) - (\gamma''(t) \bullet \gamma(t))\gamma(t)\end{aligned}$$

Definition 20.
Given $\gamma(t) \in C^2(\mathbf{I}, H_1)$, the local curvature $\kappa(\gamma, t)$ is defined:

$$\kappa(\gamma, t) = \|\gamma''(t) - (\gamma''(t) \bullet \gamma(t))\gamma(t)\|$$

Note that t is not necessarily the natural parameter. Thus the above definition is not correct in a strict differential geometric sense[12]. However, we are not only interested in the shape (curvature) of the interpolation curve. We also want a "nice" angular velocity function, i. e. one that minimizes angular acceleration (corresponding, from a physical viewpoint, to minimizing the energy). In the above expression the angular acceleration is automatically included, exactly because we do not reparameterize to the natural parameter.

[12]The curvature for a curve parameterised on the natural parameter can be written $\kappa(\gamma, t) = \|\gamma''(t)\|$. In general for a smooth curve, the correct expression from differential geometry is:

$$\kappa(\gamma, t) = \left\| \frac{\gamma''(t)}{\|\gamma'(t)\|^2} - \frac{\gamma'(t)(\gamma''(t) \bullet \gamma'(t))}{\|\gamma'(t)\|^4} \right\|$$

6.3.5 Minimizing curvature in H_1: Continuous, analytical solution

We defined the optimal interpolation curve in H_1 as the curve that minimizes the square of the curvature, but with the restriction that it must pass through the control points. We then defined the relevant expression for curvature in H_1. We thus get the following formulation of the problem:

Given the control points $q_1, \ldots, q_N \in H_1$ we seek $\gamma(t) \in C^k(\mathbf{I}, H_1)$ such that there exist $t_1, \ldots, t_N \in (I)$ that satisfy $\gamma(t_i) = q_i$, and such that this expression is minimized:

$$K(\gamma) = \int_{t_1}^{t_N} ||\kappa(\gamma, h)||^2 dh \tag{6.17}$$

The problem of minimizing an integral of a function is called a *calculus of variations* problem. Below, we will outline the basic method used for solving problems in the calculus of variations.

A necessary condition for $K(\gamma)$ to attain a minimum is that $K'(\gamma) = 0$. For $\delta \in \mathbb{R}$ and $\psi \in C^k(\mathbf{I}, H_1)$ we look at the "derivative":

$$\lim_{\delta \to 0} \frac{K(\gamma + \delta\psi) - K(\gamma)}{\delta} = 0 \tag{6.18}$$

In the above expression, $\delta\psi$ is called the *variation* of γ, and the function $\gamma + \delta\psi$ is called the *comparison function.* A function $\gamma(h) \in C^k(\mathbf{I}, H_1)$ that satisfies the boundary conditions (i. e. $\gamma(t_i) = q_i$ for $i = 1, \ldots, N$) is called an *admissible* function.

We will now derive the requirements a solution to the variation problem must meet. We therefore assume that both the solution γ and the comparison function $\gamma + \delta\psi$ are admissible, and that $K(\gamma)$ is minimal.

For the comparison function $\gamma + \delta\psi$ to be admissible, it must be the case that

$$\psi(t_1) = \psi(t_2) = \ldots = \psi(t_N) = 0 \tag{6.19}$$

Furthermore, since γ and $\gamma + \delta\psi$ lie on the surface of the unit sphere, we have that $||\gamma||^2 = 1$ and $||\gamma + \delta\psi||^2 = 1$. Thus we have[13]:

$$\begin{aligned} 1 &= ||\gamma + \delta\psi||^2 \\ &= ||\gamma||^2 + \delta^2||\psi||^2 + 2\delta(\gamma \bullet \psi) \\ &= 1 + \delta(\delta||\psi||^2 + 2\gamma \bullet \psi) \\ &\Updownarrow \\ \gamma \bullet \psi &= -\frac{\delta}{2}||\psi||^2 \end{aligned} \tag{6.20}$$

[13]In the derivations below, it is possible to ignore the fact that the constituent expressions contain quaternion functions. This is due to the fact that quaternion multiplication is not used (only the scalar product, $\bullet$, is used). Therefore, there are no problems with commutativity.

We want to use our knowledge of the "derivative" and examine $K(\gamma + \delta\psi) - K(\gamma)$:

$$\begin{aligned}
&K(\gamma+\delta\psi) - K(\gamma) \\
&\quad = \int_{t_1}^{t_N} \|\kappa(\gamma+\delta\psi, h)\|^2 - \|\kappa(\gamma, h)\|^2 \; dh \\
&\quad = \int_{t_1}^{t_N} \|(\gamma+\delta\psi)'' - ((\gamma+\delta\psi)'' \bullet (\gamma+\delta\psi))(\gamma+\delta\psi)\|^2 - \|\gamma'' - (\gamma'' \bullet \gamma)\gamma\|^2 \; dh \\
&\quad = \int_{t_1}^{t_N} \|\gamma'' + \delta\psi'' - (\gamma'' \bullet \gamma + \delta\gamma'' \bullet \psi + \delta\gamma \bullet \psi'' + \delta^2\psi'' \bullet \psi)(\gamma+\delta\psi)\|^2 \\
&\qquad\quad - \|\gamma'' - (\gamma'' \bullet \gamma)\gamma\|^2 \; dh \\
&\quad = \int_{t_1}^{t_N} \|[\gamma'' - (\gamma'' \bullet \gamma)\gamma] + \delta[\psi'' - (\gamma'' \bullet \gamma)\psi - (\gamma'' \bullet \psi)\gamma - (\psi'' \bullet \gamma)\gamma] + \delta^2[\ldots] + \delta^3[\ldots]\|^2 \\
&\qquad\quad - \|\gamma'' - (\gamma'' \bullet \gamma)\gamma\|^2 \; dh
\end{aligned}$$

In the above expression, we have collected terms according to the exponent of δ. The goal of the above derivations is to find the "derivative." In the expression for $K(\gamma+\delta\psi) - K(\gamma)$ terms multiplied by δ^2 and δ^3 can be removed because they disappear when examining the limit, after being divided by δ. Thus the expression can be written:

$$\begin{aligned}
K(\gamma+\delta\psi) - K(\gamma) &= \int_{t_1}^{t_N} \|A + \delta B\|^2 - \|A\|^2 \; dh \\
&= \int_{t_1}^{t_N} \delta^2\|B\|^2 + 2\delta A \bullet B \; dh
\end{aligned}$$

Here $A = \gamma'' - (\gamma'' \bullet \gamma)\gamma$ and $B = \psi'' - (\gamma'' \bullet \gamma)\psi - (\gamma'' \bullet \psi)\gamma - (\psi'' \bullet \gamma)\gamma$. Again we may omit terms multiplied by δ^2, and we can rewrite the expression:

$$\begin{aligned}
K(\gamma+\delta\psi) - K(\gamma) &= \int_{t_1}^{t_N} 2\delta[\gamma'' - (\gamma'' \bullet \gamma)\gamma] \bullet [\psi'' - (\gamma'' \bullet \gamma)\psi - (\gamma'' \bullet \psi)\gamma - (\gamma \bullet \psi'')\gamma] \; dh \\
&= 2\delta \int_{t_1}^{t_N} \gamma'' \bullet \psi'' - (\gamma'' \bullet \gamma)(\gamma'' \bullet \psi) - (\gamma'' \bullet \psi)(\gamma'' \bullet \gamma) - (\gamma \bullet \psi'')(\gamma'' \bullet \gamma) \\
&\qquad - (\gamma'' \bullet \gamma)(\gamma \bullet \psi'') + (\gamma'' \bullet \gamma)(\gamma'' \bullet \gamma)(\gamma \bullet \psi) + (\gamma'' \bullet \gamma)(\gamma'' \bullet \psi)(\gamma \bullet \gamma) \\
&\qquad + (\gamma'' \bullet \gamma)(\gamma \bullet \psi'')(\gamma \bullet \gamma) \; dh
\end{aligned}$$

We can now use the fact that $\gamma \bullet \gamma = \|\gamma\|^2 = 1$, since γ lies on the surface of the unit sphere. Furthermore, equation 6.20 is used to rewrite $\gamma \bullet \psi = -\frac{\delta}{2}\|\psi\|^2$. We then get:

$$\begin{aligned}
K(\gamma+\delta\psi) - K(\gamma) &= 2\delta \int_{t_1}^{t_N} \gamma'' \bullet \psi'' - (\gamma'' \bullet \gamma)(\gamma'' \bullet \psi + \gamma \bullet \psi'') + (\gamma'' \bullet \gamma)^2(-\frac{\delta}{2}\|\psi\|^2) \; dh \\
&= 2\delta \int_{t_1}^{t_N} \gamma'' \bullet \psi'' - (\gamma'' \bullet \gamma)(\gamma'' \bullet \psi + \gamma \bullet \psi'') \; dh
\end{aligned}$$

We have yet again ignored terms multiplied by δ^2.

After numerous rewritings, we have isolated δ as a single factor. This factor will disappear in the expression for the derivative when divided by δ. We therefore now want to isolate terms containing ψ. This is attempted using partial integration. The expression is rewritten as follows:

$$\begin{aligned}
K(\gamma+\delta\psi)-K(\gamma) &= 2\delta\sum_{i=1}^{N-1} L_i \\
L_i &= \int_{t_i}^{t_{i+1}} \gamma'' \bullet \psi'' - (\gamma''\bullet\gamma)(\gamma''\bullet\psi + \gamma\bullet\psi'')\, dh \\
&= \int_{t_i}^{t_{i+1}} (\gamma'' - (\gamma''\bullet\gamma)\gamma)\bullet\psi'' - (\gamma''\bullet\gamma)(\gamma''\bullet\psi)\, dh \\
&= [(\gamma''-(\gamma''\bullet\gamma)\gamma)\bullet\psi']_{t_i}^{t_{i+1}} \\
&\quad -\int_{t_i}^{t_{i+1}} (\frac{d}{dh}\{\gamma''-(\gamma''\bullet\gamma)\gamma\})\bullet\psi' - (\gamma''\bullet\gamma)(\gamma''\bullet\psi)\, dh \\
&= [(\gamma''-(\gamma''\bullet\gamma)\gamma)\bullet\psi']_{t_i}^{t_{i+1}} - [(\frac{d}{dh}\{\gamma''-(\gamma''\bullet\gamma)\gamma\})\bullet\psi]_{t_i}^{t_{i+1}} \\
&\quad +\int_{t_i}^{t_{i+1}} (\frac{d^2}{dh^2}\{\gamma''-(\gamma''\bullet\gamma)\gamma\})\bullet\psi - (\gamma''\bullet\gamma)(\gamma''\bullet\psi)\, dh
\end{aligned}$$

Note that the above expression requires that γ is four time s differentiable. Now we can use that ψ is zero in all the control points (equation 6.19) to see that the second term is zero:

$$L_i = [(\gamma''-(\gamma''\bullet\gamma)\gamma)\bullet\psi']_{t_i}^{t_{i+1}} + \int_{t_i}^{t_{i+1}} (\frac{d^2}{dh^2}\{\gamma''-(\gamma''\bullet\gamma)\gamma\})\bullet\psi - (\gamma''\bullet\gamma)(\gamma''\bullet\psi)\, dh$$

We again consider the whole expression:

$$\begin{aligned}
K(\gamma+\delta\psi)-K(\gamma) &= 2\delta\sum_{i=1}^{N-1} L_i \\
&= 2\delta\sum_{i=1}^{N-1} [(\gamma''-(\gamma''\bullet\gamma)\gamma)\bullet\psi']_{t_i}^{t_{i+1}} \\
&\quad +2\delta\sum_{i=1}^{N-1}\int_{t_i}^{t_{i+1}} (\frac{d^2}{dh^2}\{\gamma''-(\gamma''\bullet\gamma)\gamma\})\bullet\psi - (\gamma''\bullet\gamma)(\gamma''\bullet\psi)\, dh \\
&= 2\delta[(\gamma''-(\gamma''\bullet\gamma)\gamma)\bullet\psi']_{t_1}^{t_n} \\
&\quad +2\delta\sum_{i=1}^{N-1}\int_{t_i}^{t_{i+1}} (\frac{d^2}{dh^2}\{\gamma''-(\gamma''\bullet\gamma)\gamma\} - (\gamma''\bullet\gamma)\gamma'')\bullet\psi\, dh
\end{aligned}$$

The last rewriting uses continuity in the control points. We can find the "derivative"

$$\begin{aligned}
\lim_{\delta\to 0}\frac{K(\gamma+\delta\psi)-K(\gamma)}{\delta} &= 2[(\gamma''-(\gamma''\bullet\gamma)\gamma)\bullet\psi']_{t_1}^{t_N} \\
&\quad +2\sum_{i=1}^{N-1}\int_{t_i}^{t_{i+1}} (\frac{d^2}{dh^2}\{\gamma''-(\gamma''\bullet\gamma)\gamma\} - (\gamma''\bullet\gamma)\gamma'')\bullet\psi\, dh
\end{aligned}$$

Since the derivative must be zero (equation 6.18) for any ψ, we get the following requirements to the solution:

$$\begin{aligned}
\gamma &\in C^4((I), H_1) \\
0 &= \gamma''(t_1) - (\gamma''(t_1) \bullet \gamma(t_1))\gamma(t_1) \\
0 &= \gamma''(t_N) - (\gamma''(t_N) \bullet \gamma(t_N))\gamma(t_N) \\
(\gamma'' \bullet \gamma)\gamma'' &= \frac{d^2}{dh^2}\{\gamma'' - (\gamma'' \bullet \gamma)\gamma\} \\
&= \gamma'''' - (\gamma'' \bullet \gamma)''\gamma - 2(\gamma'' \bullet \gamma)'\gamma' - (\gamma'' \bullet \gamma)\gamma''
\end{aligned}$$

The second and third requirements are equivalent with the local curvature being zero at the end-points. The last requirement can be rewritten using $\gamma \bullet \gamma = \|\gamma\| = 1$:

$$0 = (1)'' = (\gamma \bullet \gamma)'' = 2\gamma'' \bullet \gamma + 2\gamma' \bullet \gamma' \tag{6.21}$$

Now we can state the set of differential equations, that must be solved to find an analytical solution to the desired optimal interpolation curve:

Proposition 31.
Given the control points $q_1, \ldots, q_N \in H_1$ the interpolation curve $\gamma \in C^4(\mathbf{I}, H_1)$, where $\gamma(t_i) = q_i$ for $t_i \in \mathbf{I}$, will minimize $\int_{t_1}^{t_N} \|\kappa(h)\|^2 dh$ if the following requirements are met:

1. $\kappa(t_1) = 0$
2. $\kappa(t_N) = 0$
3. *For each interval $t_i < h < t_{i+1}$:*
   $\gamma'''' + (\gamma' \bullet \gamma')''\gamma + 2(\gamma' \bullet \gamma')'\gamma' + 2(\gamma' \bullet \gamma')\gamma'' = 0$

Now "all" that remains is to solve the above fourth-order differential equation with the given boundary values. This, unfortunately, is not possible with the existing mathematical knowledge. This is confirmed by Jørgen Sand[14] and Gerd Grubb[15].

6.3.6 Minimizing curvature in H_1: Continuous, semi-analytical solution

The strict mathematical derivation of the desired optimal interpolation curve stranded on an unsolvable differential equation.

We can, however, again seek inspiration in the plane. The corresponding differential equation is here $\gamma'''' = 0$, i. e. that the fourth derivative of the curve must be zero between the control points. This corresponds to a third-order curve (a spline). If the solution is constrained to be a third-order curve, the equations can be solved more easily.

[14]Associate Professor at the Institute of Computer Science, University of Copenhagen, specializing in the solution of systems of equations.

[15]Professor at the Institute of Mathematics, University of Copenhagen, specializing in differential equations.

We could therefore restrict which family of functions we would like the interpolation curve to belong to. This could, for example, be a kind of cubic splines. With this restriction on the set of possible solutions, the optimization problem could be solved.

Depending on the choice of definition of the family of curves, this strategy for finding a solution will give a result corresponding to the basis for the construction of *Squad*.

To keep this report at a manageable size, we will not pursue this line of thought any further.

6.3.7 Minimizing curvature in H_1: Discretized, numerical solution

Unfortunately, we cannot give an analytical expression for the optimal interpolation curve. Therefore we will try to solve a discrete version of the problem using a numerical method.

We rewrite the problem in an equivalent discrete version. We then solve the new version of the problem.

Discretization

Given the control points $Q_1, \ldots, Q_k \in H_1$ we sought an analytical solution $\gamma(t) \in C^k(\mathbf{I}, H_1)$ such that there existed $t_1, \ldots, t_k \in (I)$, that $\gamma(t_i) = Q_i$, and such that the following expression was minimized:

$$K(\gamma) = \int_{t_1}^{t_k} \|\kappa(\gamma, t)\|^2 dt \tag{6.22}$$

In the discrete version we will therefore attempt to solve the following problem. Given control points $Q_1, \ldots, Q_k \in H_1$ we seek $q_1, \ldots, q_N \in H_1$ such that $q_{i_t} = Q_t$ for $t = 1, \ldots, k$ and such that the following expression is minimized:

$$E = \sum_{i=1}^{N} l(q_i)\ \|\tilde{\kappa}(q_i)\|^2 \tag{6.23}$$

The integral has been replaced by a sum. The "parameter width" of the interval we integrate over is termed $l(q_i)$. It can be expressed as a centered average of the parameter width in the intervals immediately before and after the i'th quaternion:

$$l(q_i) \quad = \quad \frac{\|q_i - q_{i-1}\| + \|q_i - q_{i+1}\|}{2} \tag{6.24}$$

Another measure of the parameter distance between q_i and q_{i-1} in the approximation for $l(q_i)$ could be θ_i, where $\cos\theta_i = q_i \bullet q_{i+1}$, instead of $\|q_i - q_{i-1}\|$.

In equation 6.23 $\tilde{\kappa}$ is the discrete version of the local curvature:

$$\tilde{\kappa}(q_i) \quad = \quad q_i'' - \frac{q_i'' \bullet q_i}{q_i \bullet q_i}\, q_i \tag{6.25}$$

Note that the denominator $q_i \bullet q_i$ is not omitted as in the definition of local curvature (page 59). This is because the interpolated quaternions cannot be expected to be unit quaternions (this is explained below). Therefore, the denominator is not in general equal to 1; thus it cannot be omitted.

In equation 6.25 the second derivative of the discrete approximation of the interpolation curve is used. A good centered approximation of the second derivative is ([Kincaid & Cheney, 1991], [Barr et al., 1992]):

$$q_i'' = \frac{q_{i-1} - 2q_i + q_{i+1}}{l(q_i)^2} \tag{6.26}$$

Gradient descent

The above equation (equation 6.23) can be minimized using *gradient descent.* In general terms, gradient descent can be described as follows. Consider the function that is to be minimized (commonly termed the *energy function*) as a hilly landscape, where the function value is the height of a hill at each set of coordinates. The gradient of the function in that point points in the steepest possible up-hill direction. Gradient descent is based on an initial guess at the solution. From the initial guess the gradient is computed, and a small step is taken in the opposite direction of the gradient (i. e. down-hill), resulting in a new point. This process is repeated with the new point until the function value does not become smaller by taking a step. In this fashion, the gradient descent method yields an approximate local minimum. It is outside the scope of this report to describe the theory of gradient descent any further, but we will describe the method in enough detail so that readers without prerequisites in the field will still be able to follow the derivations.

When using numerical approximation methods such as gradient descent, it is often difficult to maintain restrictions on the solution space. Instead a term is added that makes the solution more "expensive"[16] if the solution lies outside the desired solution space. Thus, we seek a function that can determine if the discrete version of the interpolation curve lives in H_1, and thus consists of unit quaternions. This can be done by:

$$g(q) = q \bullet q - 1 \tag{6.27}$$

Note that $H_1 = \{q \in H \mid g(q) = 0\}$. This measure for determining if the quaternions are unit quaternions can be combined with the energy function E as follows:

$$F = \sum_{i=1}^{N} l(q_i) \; \|\tilde{\kappa}(q_i)\|^2 + c \; g(q_i)^2 \tag{6.28}$$

Assuming $c \in \mathbb{R}$ suitably large, the energy function F will have a minimum approximately where E has a minimum, and where all q_i are approximately unit quaternions.

[16]We here assume "the cheaper, the better", which is not always the case in real life. Sometimes you have to pay extra for quality. Red wine is a good example of this, although there are, of course, exceptions in this case, too.

We want to find a minimum for F in equation 6.28 using gradient descent. We therefore have to find the gradient. Below, we will write $q_{i,x}$ for the x'th coordinate (regarding a quaternion as a four-dimensional vector) in the i'th quaternion in the discrete interpolation curve. The gradient is $4n$-dimensional. Each coordinate can be written:

$$\frac{\partial F}{\partial q_{i,x}} = \frac{\partial}{\partial q_{i,x}} \left(\sum_{j=1}^{N} l(q_j)\ \|\tilde{\kappa}(q_j)\|^2 + c\ g(q_j)^2 \right)$$

In equations 6.24, 6.25, and 6.26, q_i is a term in the discrete versions of $l(q_{i-1})$, $l(q_i)$, $l(q_{i+1})$, $\tilde{\kappa}(q_{i-1})$, $\tilde{\kappa}(q_i)$ and $\tilde{\kappa}(q_{i+1})$. The corresponding terms must appear in the partial derivative:

$$\begin{aligned}
\frac{\partial F}{\partial q_{i,x}} &= \frac{\partial}{\partial q_{i,x}} \left(l(q_{i-1})\ \|\tilde{\kappa}(q_{i-1})\|^2 + l(q_i)\ \|\tilde{\kappa}(q_i)\|^2 + l(q_{i+1})\ \|\tilde{\kappa}(q_{i+1})\|^2 + c\ g(q_i)^2 \right) \\
&= \frac{\partial}{\partial q_{i,x}} \left(l(q_{i-1})\ \tilde{\kappa}(q_{i-1}) \bullet \tilde{\kappa}(q_{i-1}) + l(q_i)\ \tilde{\kappa}(q_i) \bullet \tilde{\kappa}(q_i) + l(q_{i+1})\ \tilde{\kappa}(q_{i+1}) \bullet \tilde{\kappa}(q_{i+1}) + c\ g(q_i)^2 \right) \\
&= \frac{\partial l(q_{i-1})}{\partial q_{i,x}} \tilde{\kappa}(q_{i-1}) \bullet \tilde{\kappa}(q_{i-1}) + \frac{\partial l(q_i)}{\partial q_{i,x}} \tilde{\kappa}(q_i) \bullet \tilde{\kappa}(q_i) + \frac{\partial l(q_{i+1})}{\partial q_{i,x}} \tilde{\kappa}(q_{i+1}) \bullet \tilde{\kappa}(q_{i+1}) + \\
&\quad 2l(q_{i-1})\tilde{\kappa}(q_{i-1}) \bullet \frac{\partial\ \tilde{\kappa}(q_{i-1})}{\partial q_{i,x}} + 2l(q_i)\tilde{\kappa}(q_i) \bullet \frac{\partial\ \tilde{\kappa}(q_i)}{\partial q_{i,x}} + 2l(q_{i+1})\tilde{\kappa}(q_{i+1}) \bullet \frac{\partial\ \tilde{\kappa}(q_{i+1})}{\partial q_{i,x}} + \\
&\quad 2c\ g(q_i)\ \frac{\partial\ g(q_i)}{\partial q_{i,x}} \qquad (6.29)
\end{aligned}$$

Below we will derive the partial derivatives for the sub-expressions of the above expression.

We introduce the notation 1_x to be the vector with 1 in the x'th coordinate, and 0 in the other coordinates. We can now look forward to deriving the partial derivatives of $g(q_i)$, $l(q_{i-1})$, $l(q_i)$, $l(q_{i+1})$, q''_{i-1}, q''_i, q''_{i+1}, $\tilde{\kappa}(q_i)$, $\tilde{\kappa}(q_{i-1})$ and $\tilde{\kappa}(q_{i+1})$:

$$\begin{aligned}
\frac{\partial\ g(q_i)}{\partial q_{i,x}} &= \frac{\partial}{\partial q_{i,x}} (q_i \bullet q_i - 1) \\
&= 2\ q_i \bullet \frac{\partial\ q_i}{\partial q_{i,x}} \\
&= 2\ q_{i,x} \qquad (6.30) \\
\frac{\partial\ l(q_{i-1})}{\partial q_{i,x}} &= \frac{\partial}{\partial q_{i,x}} \frac{\|q_{i-1} - q_{i-2}\| + \|q_{i-1} - q_i\|}{2} \\
&= \frac{1}{2} \frac{\partial}{\partial q_{i,x}} \left(\sqrt{(q_{i-1} - q_{i-2}) \bullet (q_{i-1} - q_{i-2})} + \sqrt{(q_{i-1} - q_i) \bullet (q_{i-1} - q_i)} \right) \\
&= \frac{1}{2} \left(- \frac{1_x \bullet (q_{i-1} - q_i)}{\sqrt{(q_{i-1} - q_i) \bullet (q_{i-1} - q_i)}} \right) \\
&= \frac{1_x \bullet (q_i - q_{i-1})}{2\ \|q_{i-1} - q_i\|} \qquad (6.31)
\end{aligned}$$

$$
\begin{aligned}
\frac{\partial\, l(q_i)}{\partial q_{i,x}} &= \frac{\partial}{\partial q_{i,x}} \frac{\|q_i - q_{i-1}\| + \|q_i - q_{i+1}\|}{2} \\
&= \frac{1}{2} \frac{\partial}{\partial q_{i,x}} \left(\sqrt{(q_i - q_{i-1}) \bullet (q_i - q_{i-1})} + \sqrt{(q_i - q_{i+1}) \bullet (q_i - q_{i+1})} \right) \\
&= \frac{1}{2} \left(\frac{1_x \bullet (q_i - q_{i-1})}{\sqrt{(q_i - q_{i-1}) \bullet (q_i - q_{i-1})}} + \frac{1_x \bullet (q_i - q_{i+1})}{\sqrt{(q_i - q_{i+1}) \bullet (q_i - q_{i+1})}} \right) \\
&= \frac{1_x \bullet (q_i - q_{i-1})}{2\, \|q_i - q_{i-1}\|} + \frac{1_x \bullet (q_i - q_{i+1})}{2\, \|q_{i+1} - q_i\|} \qquad (6.32) \\
\frac{\partial\, l(q_{i+1})}{\partial q_{i,x}} &= \frac{\partial}{\partial q_{i,x}} \frac{\|q_{i+1} - q_i\| + \|q_{i+1} - q_{i+2}\|}{2} \\
&= \frac{1}{2} \frac{\partial}{\partial q_{i,x}} \left(\sqrt{(q_{i+1} - q_i) \bullet (q_{i+1} - q_i)} + \sqrt{(q_{i+1} - q_{i+2}) \bullet (q_{i+1} - q_{i+2})} \right) \\
&= \frac{1}{2} \left(- \frac{1_x \bullet (q_{i+1} - q_i)}{\sqrt{(q_{i+1} - q_i) \bullet (q_{i+1} - q_i)}} \right) \\
&= \frac{1_x \bullet (q_i - q_{i+1})}{2\, \|q_{i+1} - q_i\|} \qquad (6.33) \\
\frac{\partial\, q''_{i-1}}{\partial q_{i,x}} &= \frac{\partial}{\partial q_{i,x}} \frac{q_{i-2} - 2q_{i-1} + q_i}{l^2(q_{i-1})} \\
&= \frac{l(q_{i-1})^2 \frac{\partial}{\partial q_{i,x}} (q_{i-2} - 2q_{i-1} + q_i) - (q_{i-2} - 2q_{i-1} + q_i) \frac{\partial}{\partial q_{i,x}} l(q_{i-1})^2}{l(q_{i-1})^4} \\
&= \frac{l(q_{i-1})^2 \frac{\partial}{\partial q_{i,x}} q_i - (q_{i-2} - 2q_{i-1} + q_i) 2 l(q_{i-1}) \frac{\partial}{\partial q_{i,x}} l(q_{i-1})}{l(q_{i-1})^4} \\
&= \frac{l(q_{i-1})^2\ 1_x - 2(q_{i-2} - 2q_{i-1} + q_i) l(q_{i-1}) \frac{\partial}{\partial q_{i,x}} l(q_{i-1})}{l(q_{i-1})^4} \qquad (6.34) \\
\frac{\partial\, q''_i}{\partial q_{i,x}} &= \frac{\partial}{\partial q_{i,x}} \frac{q_{i-1} - 2q_i + q_{i+1}}{l(q_i)^2} \\
&= \frac{l(q_i)^2 \frac{\partial}{\partial q_{i,x}} (q_{i-1} - 2q_i + q_{i+1}) - (q_{i-1} - 2q_i + q_{i+1}) \frac{\partial}{\partial q_{i,x}} l(q_i)^2}{l(q_i)^4} \\
&= \frac{l(q_i)^2 \frac{\partial}{\partial q_{i,x}} (-2q_i) - (q_{i-1} - 2q_i + q_{i+1}) 2 l(q_i) \frac{\partial}{\partial q_{i,x}} l(q_i)}{l(q_i)^4} \\
&= -\frac{2 l(q_i)^2\ 1_x + 2(q_{i-1} - 2q_i + q_{i+1}) l(q_i) \frac{\partial}{\partial q_{i,x}} l(q_i)}{l(q_i)^4} \qquad (6.35)
\end{aligned}
$$

$$\begin{aligned}
\frac{\partial\, q''_{i+1}}{\partial q_{i,x}} &= \frac{\partial}{\partial q_{i,x}} \frac{q_i - 2q_{i+1} + q_{i+2}}{l(q_{i+1})^2} \\
&= \frac{l(q_{i+1})^2 \frac{\partial}{\partial q_{i,x}}(q_i - 2q_{i+1} + q_{i+2}) - (q_i - 2q_{i+1} + q_{i+2}) \frac{\partial}{\partial q_{i,x}} l(q_{i+1})^2}{l(q_{i+1})^4} \\
&= \frac{l(q_{i+1})^2 \frac{\partial}{\partial q_{i,x}} q_i - (q_i - 2q_{i+1} + q_{i+2}) 2l(q_{i+1}) \frac{\partial}{\partial q_{i,x}} l(q_{i+1})}{l(q_{i+1})^4} \\
&= \frac{l(q_{i+1})^2\ 1_x - 2(q_i - 2q_{i+1} + q_{i+2}) l(q_{i+1}) \frac{\partial}{\partial q_{i,x}} l(q_{i+1})}{l(q_{i+1})^4} \qquad (6.36)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial\, \tilde{\kappa}(q_{i-1})}{\partial q_{i,x}} &= \frac{\partial}{\partial q_{i,x}} \left(q''_{i-1} - \frac{q''_{i-1} \bullet q_{i-1}}{q_{i-1} \bullet q_{i-1}}\ q_{i-1} \right) \\
&= \frac{\partial\, q''_{i-1}}{\partial q_{i,x}} - \frac{\frac{\partial q''_{i-1}}{\partial q_{i,x}} \bullet q_{i-1}}{q_{i-1} \bullet q_{i-1}}\ q_{i-1} \qquad (6.37)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial\, \tilde{\kappa}(q_i)}{\partial q_{i,x}} &= \frac{\partial}{\partial q_{i,x}} \left(q''_i - \frac{q''_i \bullet q_i}{q_i \bullet q_i}\ q_i \right) \\
&= \frac{\partial\, q''_i}{\partial q_{i,x}} - \frac{q''_i \bullet q_i}{q_i \bullet q_i} \frac{\partial\, q_i}{\partial q_{i,x}} - \frac{\partial}{\partial q_{i,x}} \left(\frac{q''_i \bullet q_i}{q_i \bullet q_i} \right)\ q_i \\
&= \frac{\partial\, q''_i}{\partial q_{i,x}} - \frac{q''_i \bullet q_i}{q_i \bullet q_i}\ 1_x - \frac{\partial}{\partial q_{i,x}} \left(\frac{q''_i \bullet q_i}{q_i \bullet q_i} \right)\ q_i \\
&= \frac{\partial\, q''_i}{\partial q_{i,x}} - \frac{q''_i \bullet q_i}{q_i \bullet q_i}\ 1_x - \left(\frac{\frac{\partial q''_i \bullet q_i}{\partial q_{i,x}}\ q_i \bullet q_i - \frac{\partial q_i \bullet q_i}{\partial q_{i,x}}\ q''_i \bullet q_i}{(q_i \bullet q_i)^2} \right)\ q_i \\
&= \frac{\partial\, q''_i}{\partial q_{i,x}} - \frac{q''_i \bullet q_i}{q_i \bullet q_i}\ 1_x - \left(\frac{\left(\frac{\partial q''_i}{\partial q_{i,x}} \bullet q_i + q''_i \bullet \frac{\partial q_i}{\partial q_{i,x}} \right)\ q_i \bullet q_i - 2\ \left(q_i \bullet \frac{\partial q_i}{\partial q_{i,x}} \right)\ q''_i \bullet q_i}{(q_i \bullet q_i)^2} \right)\ q_i \\
&= \frac{\partial\, q''_i}{\partial q_{i,x}} - \frac{q''_i \bullet q_i}{q_i \bullet q_i}\ 1_x - \left(\frac{\left(\frac{\partial\, q''_i}{\partial q_{i,x}} \bullet q_i + q''_i \bullet 1_x \right)\ q_i \bullet q_i - 2\ \left(q_i \bullet 1_x \right)\ q''_i \bullet q_i}{(q_i \bullet q_i)^2} \right)\ q_i (6.38)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial\, \tilde{\kappa}(q_{i+1})}{\partial q_{i,x}} &= \frac{\partial}{\partial q_{i,x}} \left(q''_{i+1} - \frac{q''_{i+1} \bullet q_{i+1}}{q_{i+1} \bullet q_{i+1}}\ q_{i+1} \right) \\
&= \frac{\partial\, q''_{i+1}}{\partial q_{i,x}} - \frac{\frac{\partial q''_{i+1}}{\partial q_{i,x}} \bullet q_{i+1}}{q_{i+1} \bullet q_{i+1}}\ q_{i+1} \qquad (6.39)
\end{aligned}$$

We now have simple (but long) expressions for all the terms in the gradient. We return to the gradient equation (equation 6.29). All the constituent terms have been derived, and the gradient can be explicitly computed by substituting the derived terms for $\tilde{\kappa}(q_{i-1})$, $\tilde{\kappa}(q_i)$, and $\tilde{\kappa}(q_{i+1})$ (equation 6.25) and the partial derivatives $\frac{\partial \tilde{\kappa}(q_{i-1})}{\partial q_{i,x}}$, $\frac{\partial \tilde{\kappa}(q_i)}{\partial q_{i,x}}$ and $\frac{\partial \tilde{\kappa}(q_{i+1})}{\partial q_{i,x}}$ (equations 6.37, 6.38, and 6.39) into the equation of the gradient (equation 6.29).

The algorithm

We have now derived an explicit discrete expression for the gradient, and we can perform gradient descent according to the following algorithm:

Initialization
: Give a good initial guess $\mathbf{q^0} = (q_1, \ldots, q_N)$. An obvious way of doing this is by using one of the methods described earlier (*Slerp* or *Squad*). The better the initial guess, the faster the method converges.

Iteration
: Write out the gradient using equation 6.29:

$$\nabla F = ((\frac{\partial F}{\partial q_{1,1}}, \frac{\partial F}{\partial q_{1,2}}, \frac{\partial F}{\partial q_{1,3}}, \frac{\partial F}{\partial q_{1,4}}), ..., (\frac{\partial F}{\partial q_{N,1}}, \frac{\partial F}{\partial q_{N,2}}, \frac{\partial F}{\partial q_{N,3}}, \frac{\partial F}{\partial q_{N,4}}))$$

(We set the gradient to zero in all key frames.)

Perform the iteration on the solution guess: $\mathbf{q^{i+1}} = \mathbf{q^i} - e\nabla F$, where e defines the *step length*

Alternatively: Perform the iteration on the solution guess: $\mathbf{q^{i+1}} = \mathbf{q^i} - e\frac{\nabla F}{\|\nabla F\|}$. This method can provide greater numerical stability.

Termination condition
: Repeat the second step until a suitable termination condition has been reached. The termination condition can be dependent on the number of iterations, the total energy, F, the energy F in relation to the number of control points, the size of the gradient, or more advanced strategies.

Alternative methods for computing the norm of the interpolation curve

As noted above, we want the interpolation curve to lie in the space, H_1, of unit quaternions. This is ensured by using the penalty function g from equation 6.28. It can be difficult to determine when the weighting factor, $c \in \mathbb{R}$, of the penalty function (equation 6.28) is "suitably large." In theory, c must be infinite to ensure that $q_i \in H_1$. Several methods exist to handle this problem:

Projection
: There is a simple and well-defined connection between points inside and outside the solution space. As described previously, all quaternions on a line through the origin perform the same rotation. Thus the solution guess can simply be projected into the solution space by normalizing the generated quaternions. This can be done in each iteration or, alternatively, after the last iteration.

Lagrange Multipliers
: The penalty function $g(q_i)$ can also be introduced with a *Lagrange Multiplier* λ_i in the following manner:

$$F = E + \sum_{i=1}^{N} \lambda_i g(q_i)$$

A solution to equation 6.22 is a singularity for F. However, the singularity will not be a minimum, but instead a saddle point. Thus gradient descent is still an applicable method for the constituent quaternions, but gradient *ascent* must be performed on the auxiliary variables λ_i.

By including the penalty function both in equation 6.28 and a Lagrange Multiplier, the method become more numerically robust and converges faster. The details of this method can be found in [Platt & Barr, 1988] and in [Barr et al., 1992].

Polar coordinates

In the above method of solving the problem, we have reformulated the problem such that the restrictions on the solution space are integrated into the energy function that is to be minimized. However, we can restate the restriction on the solution space such that the complexity of the expression to be minimized does not increase. This can be done be representing the quaternions using polar coordinates. Thus a quaternion is written $q = [r, (\rho, \theta, \phi)]$, where r is the radius and ρ, θ, and ϕ are the three necessary rotation angles.

Using this representation, it is very easy to maintain the restriction on the solution space. This can be done by not performing gradient descent on the radius coordinate, that is kept at a constant value of 1. Thus the restriction on the solution space is maintained.

Amongst the above representations, polar coordinates will ensure that the interpolation curve stays in the space, H_1, of unit quaternions. The use of *Lagrange Multipliers* would ensure a more robust and faster converging algorithm. We will not pursue either possibility, since we want the algorithm to be as simple as possible. We regard normalizing the generated quaternions as "cheating" seen from a theoretical viewpoint. We will therefore not discuss any of the three alternative methods any further.

Extensions to the algorithm

Generally, gradient descent is a method that is fairly easily expanded. The desired property of the interpolation curve must simply be described as a zero-crossing for some function. For example we might want to ensure constant angular velocity across the entire interpolation curve. This can be obtained using yet another penalty function:

$$w(q_i) = \|q_{i-1} - q_i\| - \|q_i - q_{i+1}\| \tag{6.40}$$

Thus the penalty function increases with the difference between the previous and the following step length. This penalty can simply be introduced into the algorithm by introducing it in the original energy function (equation 6.28):

$$F = \sum_{i=1}^{N} l(q_i)\ \|\tilde{\kappa}(q_i)\|^2 + c_g\ g(q_i)^2 + c_w\ w(q_i)^2 \tag{6.41}$$

An extra term must be added to the gradient. This is easily derived since $w(q_i) = 2l(q_i)$ and the derivative can thus be obtained using equation 6.32:

$$\frac{\partial\ w(q_i)}{\partial q_{i,x}} = \frac{1_x \bullet (q_i - q_{i-1})}{\|q_i - q_{i-1}\|} + \frac{1_x \bullet (q_i - q_{i+1})}{\|q_{i+1} - q_i\|} \tag{6.42}$$

Adding the above penalty does not ensure constant speed in the interpolation curve. This is due to the fact that the gradient contains other terms that affect the distance between the individual key frames. As stated earlier, local curvature is defined such that both the geometric curvature and the size of the angular acceleration of the interpolation curve are included. In practice, this means that the angular velocity of the interpolation curve will be smaller around the key frames, where the curvature of the interpolation curve is maximal[17].

When the gradient contains terms that act in opposite directions, the algorithm becomes less robust. Thus setting c_w "suitably large" will not necessarily ensure approximately the same distance between the interpolated frames. Instead this could lead to the method becoming numerically unstable, producing unwanted results.

The implementation

The above description of the algorithm is purely theoretical. Since we did not want to analyze the convergence and stability properties of the algorithm theoretically, there is no guarantee that the method works in practice. We will therefore present a number of considerations to take into account when implementing the algorithm.

Multi-step minimization

In practice, the algorithm behaves badly when given many frames. Each frame is only affected by its neighbors, and only key frames are positioned correctly initially. Thus the adaption to the key frames must propagate through all the frames between the keys and a given frame. The more frames that lie in-between, the more iterations are necessary for the system to attain an energy minimum. This give a practical upper bound on the acceptable number of frames between each key frame.

An efficient way of solving this problem is by minimizing in several steps. In the first step relatively few frames are used, and the system attains an energy minimum after few steps. The frames computed in the first step are used as key frames in the second step. In the second step more in-between frames are added, and an energy minimum is again attained after a few steps. This process may be repeated an arbitrary number of times, until the desired number of frames has been reached.

In figure 6.15 the result of the algorithm is seen with all frames placed during the first step. This gives a bad approximation curve even after many iterations. The result bears great resemblance to *Slerp*, which was used to generate the initial guess passed to the optimization algorithm.

If the multi-step algorithm is used, a much nicer interpolation curve is produced after fewer iterations. In figure 6.16 the intermediate result with few frames can be seen. This is used as the initial configuration for the second step, and the final interpolation curve can be seen in figure 6.17. The angular velocity graph is also nicer when the multi-step algorithm is used. If the one-step method is used (see figure 6.15), the velocity curve resembles an electrocardiogram, while the velocity curve for the multi-step algorithm (6.17) is somewhat nicer, but still somewhat

[17]This is natural seen from a physical perspective. It is also necessary to drive slower through a sharp bend in the road than it is on a straight section.

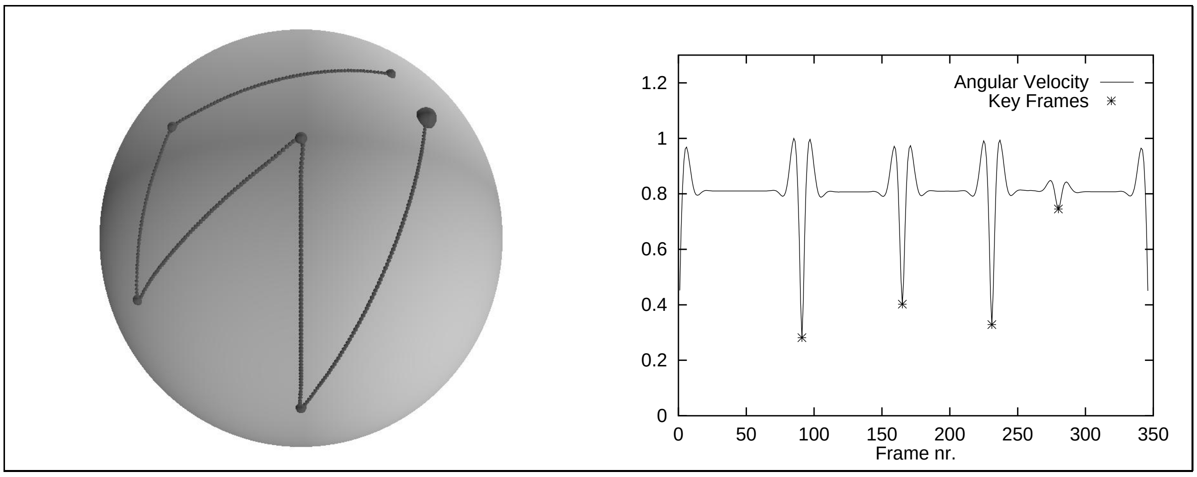


Figure 6.15: *One-step iteration. The interpolation curve contains 350 frames, and is shown after 500 iterations.*

uneven. This is a weakness in our implementation, but we expect that a more robust algorithm with better convergence properties will yield nicer velocity graphs.

All in all, 450 iterations are used for the multi-step method versus 500 in the original version. The result is obviously nicer when the multi-step method is used.

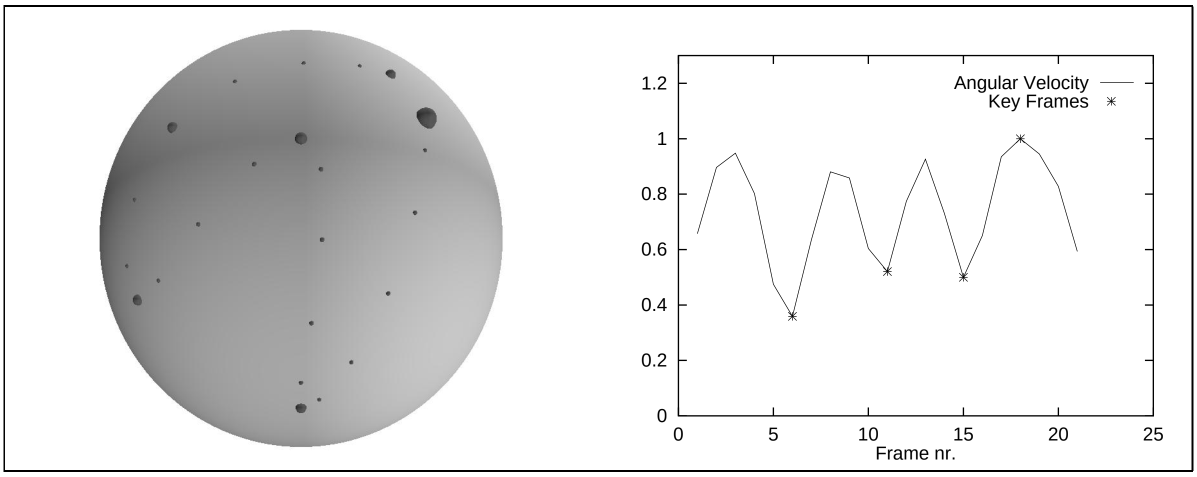


Figure 6.16: *The multi-step algorithm, first step. In the first step only 22 frames and 200 iterations are used.*

Simplifying the parameter width – $l(q_i)$

In equation 6.24 a numerical approximation to the "parameter width" for the parameter of the interpolation curve is given. In practice it turns out that this expression has no influence on the shape of the interpolation curve. The algorithm becomes less numerically stable with the expression, however. We therefore use the simpler expression:

$$l(q_i) = 1 \tag{6.43}$$

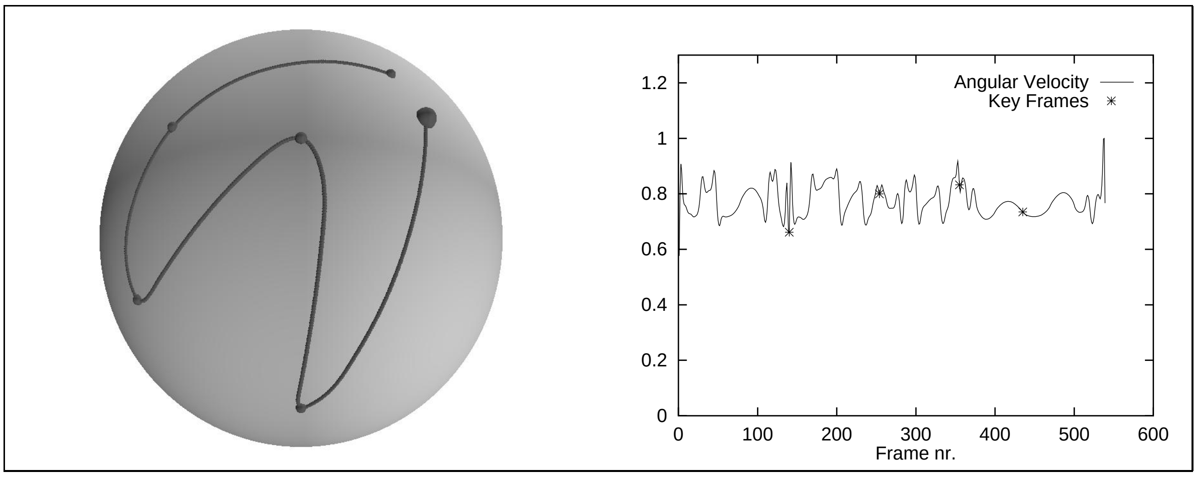


Figure 6.17: *The multi-step algorithm, final result. In the second step, 110 frames are interpolated using 150 iterations. In the third and final step, 550 frames and 100 iterations are used.*

$$\begin{aligned}\frac{\partial\ l(q_i)}{\partial q_{i,x}} &= \frac{\partial\ l(q_{i-1})}{\partial q_{i,x}} \\ &= \frac{\partial\ l(q_{i+1})}{\partial q_{i,x}} \\ &= 0 \end{aligned} \tag{6.44}$$

The expression for the gradient (equation 6.29) is simplified correspondingly:

$$\begin{aligned}\frac{\partial F}{\partial q_{i,x}} &= 2\tilde{\kappa}(q_{i-1}) \bullet \frac{\partial\ \tilde{\kappa}(q_{i-1})}{\partial q_{i,x}} + 2\tilde{\kappa}(q_i) \bullet \frac{\partial\ \tilde{\kappa}(q_i)}{\partial q_{i,x}} + 2\tilde{\kappa}(q_{i+1}) \bullet \frac{\partial\ \tilde{\kappa}(q_{i+1})}{\partial q_{i,x}} \\ &\quad +2c\ g(q_i)\ \frac{\partial\ g(q_i)}{\partial q_{i,x}} \end{aligned} \tag{6.45}$$

Weighting the curvature at the key frames

The analytical version of the minimization of curvature ensures that the curvature is minimized by definition. The discrete numerical approach only gives an approximation of the solution with minimized curvature. The validity of the approximation to the solution depends on the approximations to the constituent mathematical expressions. For example the second derivative of the interpolation curve is approximated with:

$$q_i'' = \frac{q_{i-1} - 2q_i + q_{i+1}}{l(q_i)^2}$$

It is this approximation that is at the core of the approximation of the local curvature of the interpolation curve. Offhand, it is difficult to predict if this approximation introduces "weaknesses" to the numerical solution when interpolating between key frames with certain properties.

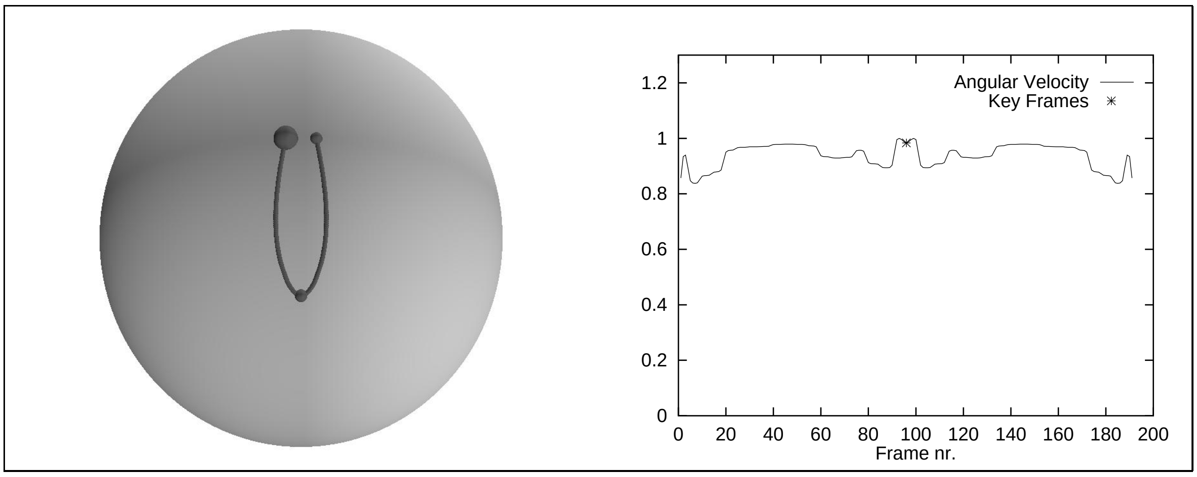


Figure 6.18: *The multi-step algorithm used on a set of key frames with a sharp curve. 200 frames are interpolated.*

In practice the numerical method is somewhat sensitive to sharp curves. For example the curve in figure 6.18 is too "sharp" in the middle key frame.

The approximation to the curvature does not adequately propagate across key frames. Let us reexamine the gradient (the simplified version, equation 6.45):

$$\frac{\partial F}{\partial q_{i,x}} = 2\tilde{\kappa}(q_{i-1}) \bullet \frac{\partial\ \tilde{\kappa}(q_{i-1})}{\partial q_{i,x}} + 2\tilde{\kappa}(q_i) \bullet \frac{\partial\ \tilde{\kappa}(q_i)}{\partial q_{i,x}} + 2\tilde{\kappa}(q_{i+1}) \bullet \frac{\partial\ \tilde{\kappa}(q_{i+1})}{\partial q_{i,x}} + 2c\ g(q_i)\ \frac{\partial\ g(q_i)}{\partial q_{i,x}}$$

The last term makes sure that the quaternions remain unit quaternions. This can be ignored. This leaves three considerations when determining the gradient, and thus the movement of the individual frame during the iteration. These are the changes in curvature in the quaternion itself, and in both its neighbors.

It is tempting to think that it is enough to consider curvature in the quaternion itself. However, if the neighbors are not taken into consideration, the result will be trivial, namely *Slerp*. Here every frame except the key frames has zero curvature. Therefore no compensation will be made for the large curvature in the key frames.

Thus minimizing curvature at the key frames depends only on that the immediate neighbors of each key frame must "take note of" the curvature of their neighbors. This is ensured by the terms $2\tilde{\kappa}(q_{i-1}) \bullet \frac{\partial\ \tilde{\kappa}(q_{i-1})}{\partial q_{i,x}}$ and $2\tilde{\kappa}(q_{i+1}) \bullet \frac{\partial\ \tilde{\kappa}(q_{i+1})}{\partial q_{i,x}}$. As shown by the example in figure 6.18, this is not quite sufficient.

The approximation of the second derivative of the interpolation curve is therefore not a good approximation in this particular case. The approximation is too local, and should have a wider domain. Since this is not a report on numerical calculation methods, we have not attempted to find an optimal approximation expression.

The problem we have described can be eliminated simply, though. Each key frame can simply "ask" its neighbors to be more "considerate." This means that we add a weighting function to each expression in the gradient. The weight is dependent on whether or not a given frame is a neighbor of a key frame. For example, $2\tilde{\kappa}(q_{i-1}) \bullet \frac{\partial\ \tilde{\kappa}(q_{i-1})}{\partial q_{i,x}}$ will be replaced by $c_k 2\tilde{\kappa}(q_{i-1}) \bullet \frac{\partial\ \tilde{\kappa}(q_{i-1})}{\partial q_{i,x}}$, if q_{i-1} is a key frame. In practice, a good value for c_k is about 1.2. The effect of the weighting function can be seen in figure 6.19.

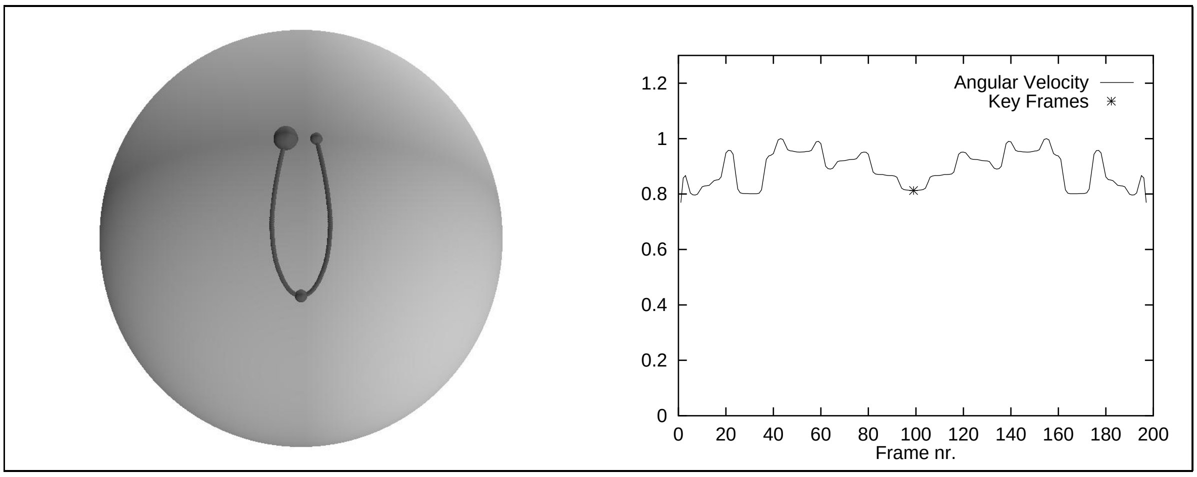


Figure 6.19: *The multi-step algorithm with a set of key frames with a sharp curve. 200 frames are interpolated, and the curvature around the key frames are weighted with a factor 1.2*

Adding a constant to the algorithm "by eye" is at odds with our stated purpose of deriving an interpolation curve from objective criteria. The alternative is to analyze the properties of different numerical approximations. This, like the properties of convergence and stability, is outside the scope of this project.

Remaining details

We have not described the termination requirements for the algorithm described above. In the implementation we have chosen the simplest possible: The number of iterations.

Correspondingly, we have not defined the initial guess that the algorithm uses. We have elected to use *Slerp* between each pair of key frames. This starting point is clearly not optimal. Quicker convergence might be achieved using *Squad*. Since the purpose was to derive a interpolation curve that is superior to *Squad*, it seemed to be more fair to use *Slerp* as the basis for the iteration.

The remaining constants in the algorithm (the step size, e, in the iteration, and the penalty factor, c) can all be calculated from robustness and convergence criteria. As previously mentioned, we will not describe these properties any further, and regard the constants as implementation details (see the introduction to the program in appendix C).

The interpolation curve summarized

The modest purpose of this chapter was to develop the optimal interpolation from objective, general criteria to the interpolation curve.

We first studied the more or less heuristic interpolation curves that are to be found in the literature. These included the naive *LinMat*, *LinEuler*, *Lerp*, the simple *Slerp*, and finally the convincing *Squad*.

Using these simple interpolation curves as a basis we tried to define which class of functions the interpolation curve should belong to (page 56). We were not able to determine which definition of smoothness was suitable to define the class of desired functions.

We then attempted to define an interpolation curve that minimized the integral of the local curvature (defined in equation 6.17) of the interpolation curve. These derivations required that the interpolation curve is four times differentiable with continuous derivatives (i. e. $\gamma(t) \in C^4(\mathbf{I}, H_1)$). This is noted on page 62 in section 6.3.5. Unfortunately, this derivation gave rise to a fourth-order differential equation that we were unable to solve. Thus it is irrelevant to consider the open questions from section 6.3.2: How many times differentiable should the curve be and are singularities allowed?

Thus we settled for a discrete, numerical solution. We have presented a method based on *gradient descent*. We examined and refined the method. The final result were some very pleasing interpolation curves.

As our final interpolation algorithm we will choose the basic algorithm from section 6.3.7 with the relative distribution of frames in the sub-intervals (see section 6.2.1 on page 55) and with the weighting of the curvature around the key frames. This interpolation curve we will name *Spring* (for **Sp**he**r**ical **I**nterpolation using **N**umerical **G**radient descent). In figure 6.20, the effect of using *Spring* can be seen. The result is only marginally better than the version that does not include special treatment of curvature at the key frames. In the next chapter we will see examples where *Spring* much more clearly demonstrates its value.

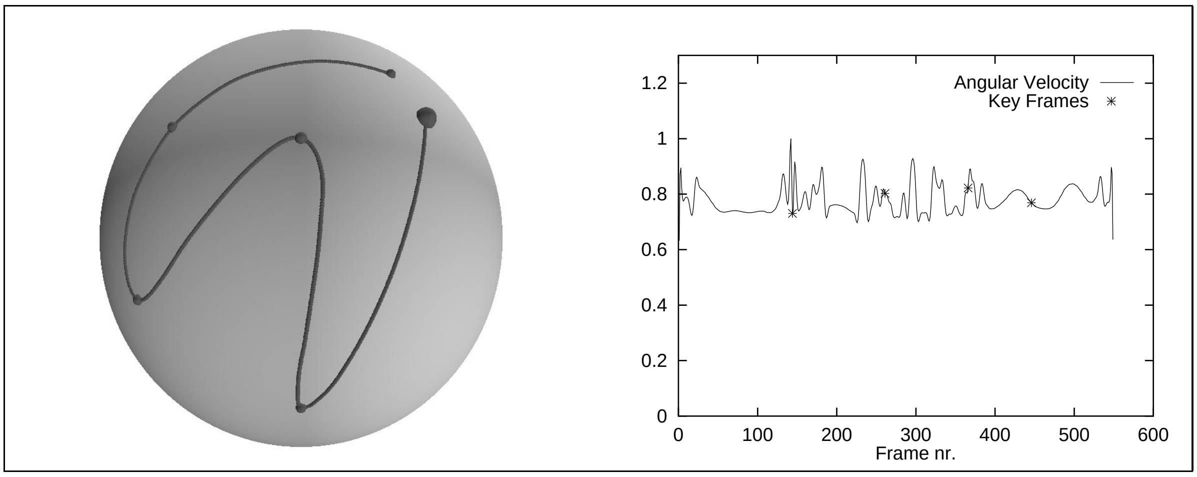


Figure 6.20: *The effect of Spring. 550 frames have been interpolated and curvature is weighted with a factor 1.3*

Chapter 7

Squad and *Spring*

In chapter 6 we treated a series of interpolation algorithms. The most convincing among the known algorithms was *Squad*. In this chapter we will compare *Squad* with our own algorithm *Spring*.

The comparison will be based on a number of illustrative examples.

7.1 Example: A semi circle

First we check if *Squad* and *Spring* can produce nice rounded curves. We have placed the key frames as corners in a spherical square. Around the center key frames the interpolation curve should approximately be a semi circle. Figure 7.1 and 7.2 show that both curves meet this requirement nicely.

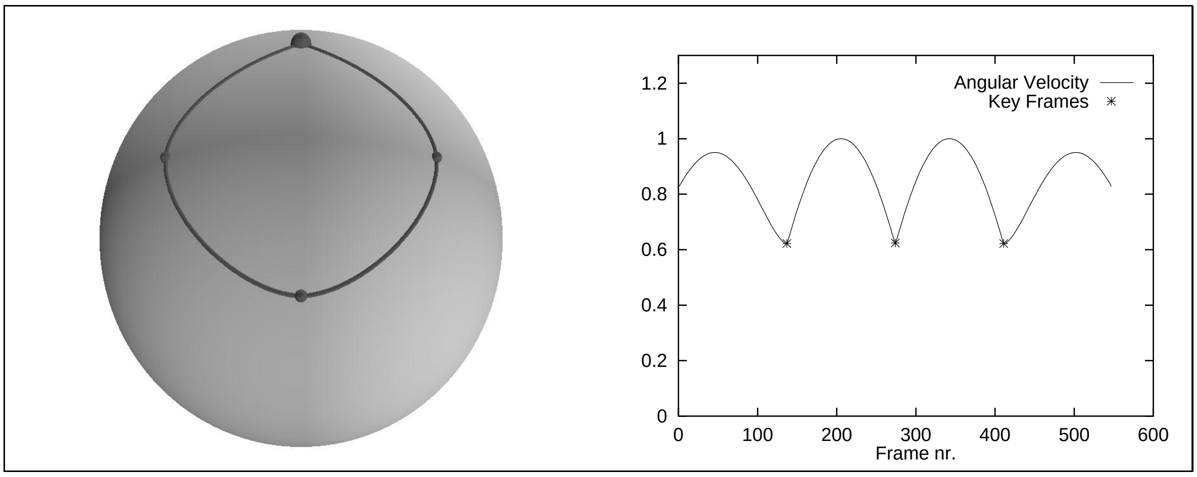


Figure 7.1: *A simple curve with soft rounded corners interpolated with Squad. A total of 550 frames have been used in the interpolation.*

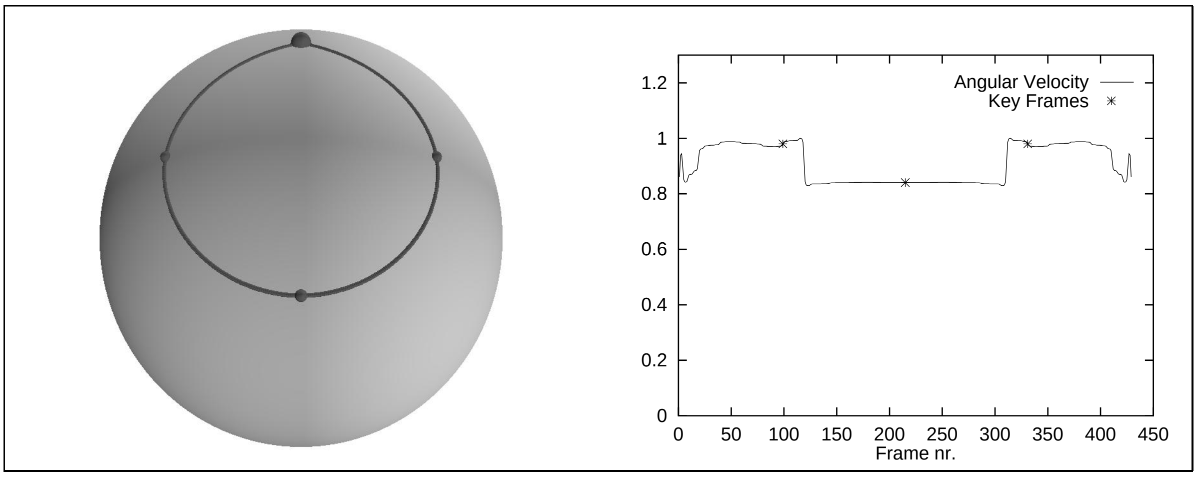


Figure 7.2: *A simple curve with soft rounded corners interpolated with Spring. A total of 430 frames and 700 iterations in the steps have been used. No extra weight on the curvature at the key frames has been added (see section 6.3.7 page 73).*

7.2 Example: A nice soft curve

This next example should be no real challenge for either of the algorithms. The expected interpolation curve is simply a nice rounded curve with no sharp corners. The intersection in the curve should pose no problem.

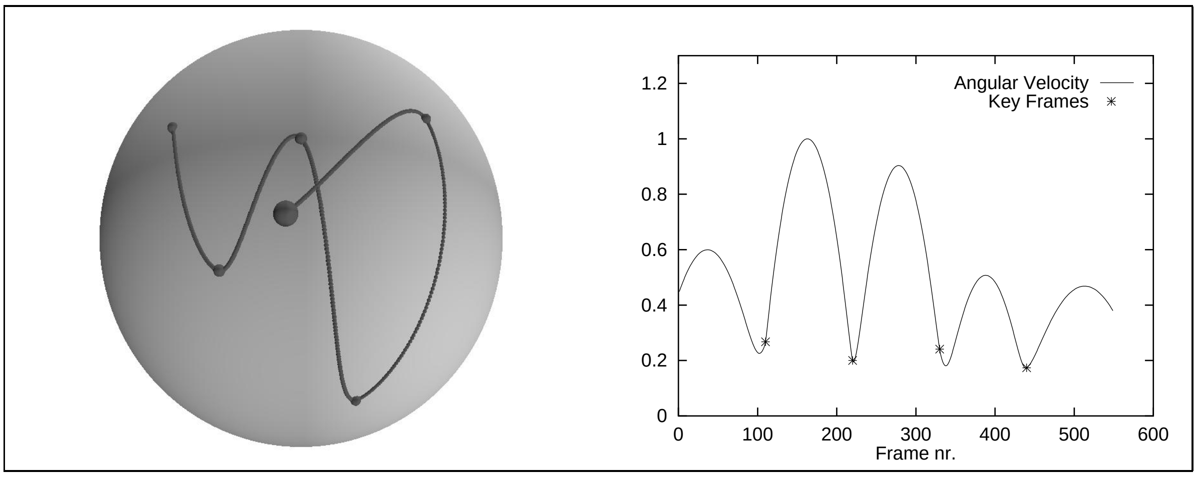


Figure 7.3: *Interpolation with Squad of a soft curve with an intersection. A total of 550 frames have been used in the interpolation.*

Both interpolation curves on figure 7.3 and figure 7.4 are nice. However, the curve has more rounded corners for *Spring* than for *Squad* (even though no extra weight has been added to the curvature at the key frames – see section 6.3.7 page 73). This means that the curve for *Spring* has smaller curvature. Furthermore the velocity graph for *Spring* is more constant than for *Squad*. This implies that *Spring* behaves as desired.

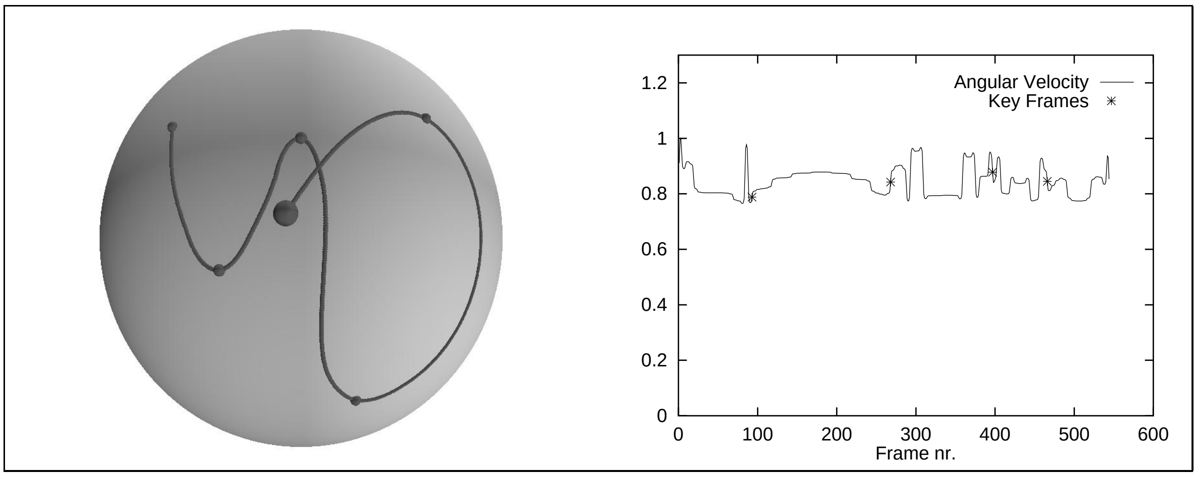


Figure 7.4: *Interpolation with Spring of a soft curve with an intersection. A total of 550 frames and 700 iterations distributed over the interpolation steps. No extra weight is added to the curvature around the key frames (see section 6.3.7 page 73)*

7.3 Example: Interpolation curve with cusp

Squad can produce interpolation curves with quite pointy corners at the key frames.

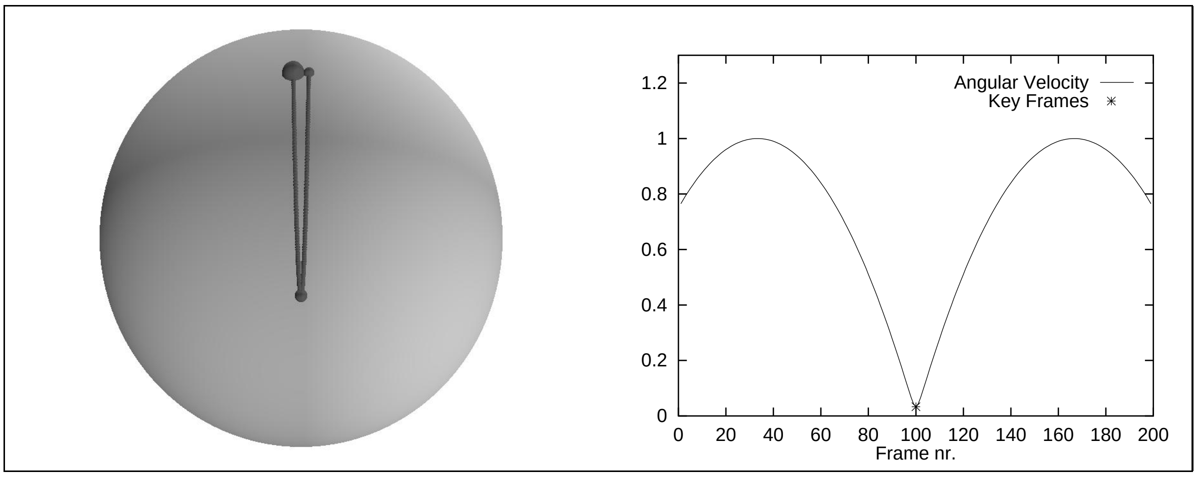


Figure 7.5: *A curve with a cusp interpolated with Squad. 200 frames have been used.*

The interpolation curve for *Squad* (figure 7.5) reveals a nasty pointy curve. However, *Spring* is able to produce a nice smooth rounded interpolation curve (figure 7.6).

That the interpolation curve for *Squad* has a sharp corner does not contradict the proven fact that *Squad* is differentiable (section 6.2.1 page 51). At the key frame with the sharp corner the velocity of the curve is zero. Thereby the function *Squad* remains differentiable although the geometric appearance of the curve is not intuitively differentiable.

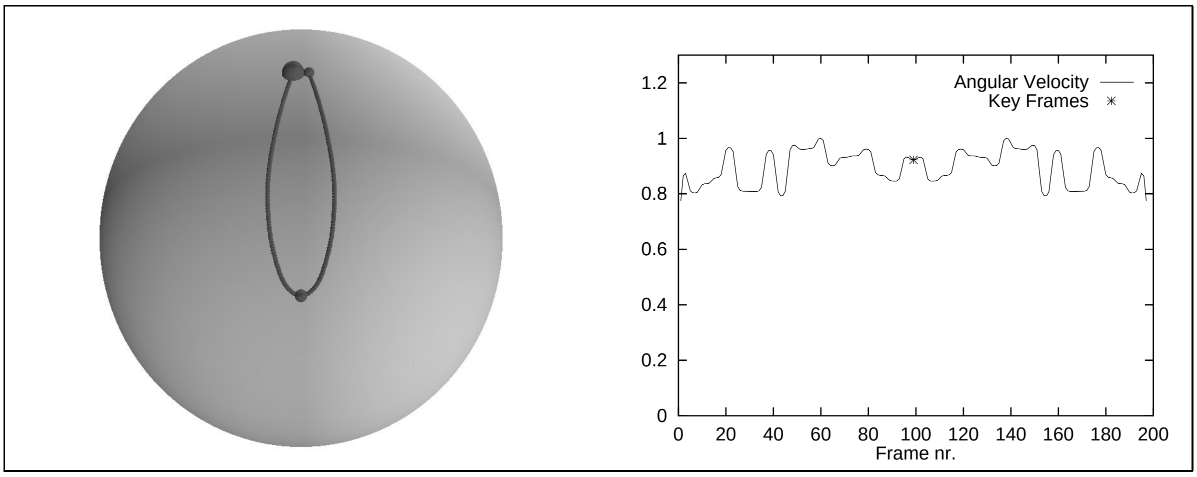


Figure 7.6: *A curve with a cusp interpolated with Spring. A total of 200 frames and 700 iterations distributed over three steps. The relative weight of the curvature around the key frames is 1.3 (see section 6.3.7 page 73).*

7.4 Example: A pendulum

We continue to investigate curves with large curvature. In this example we use a curve with infinite curvature - a pendulum motion. This is achieved with three key frames where the first and last are equal.

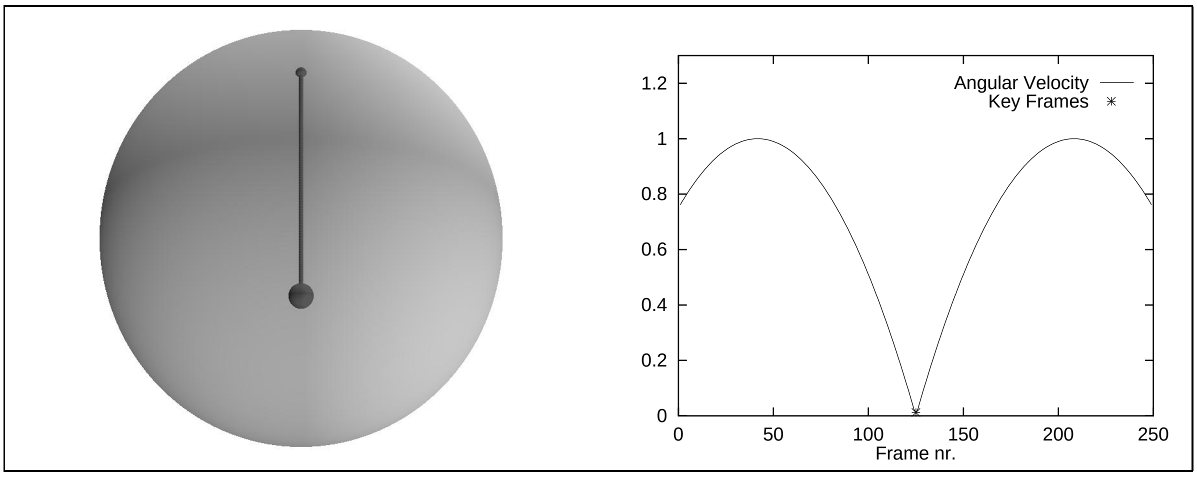


Figure 7.7: *Squad displaying pendulum motion over 250 frames.*

The desired behaviour of the interpolation curve is not intuitively obvious. Since all key frames are on a arc one would expect the curve to remain on this arc.

The figures (7.7 and 7.8) shows the same behaviour for both curves - the pendulum motion. Note that *Lerp* and *Slerp* would produce the same curve.

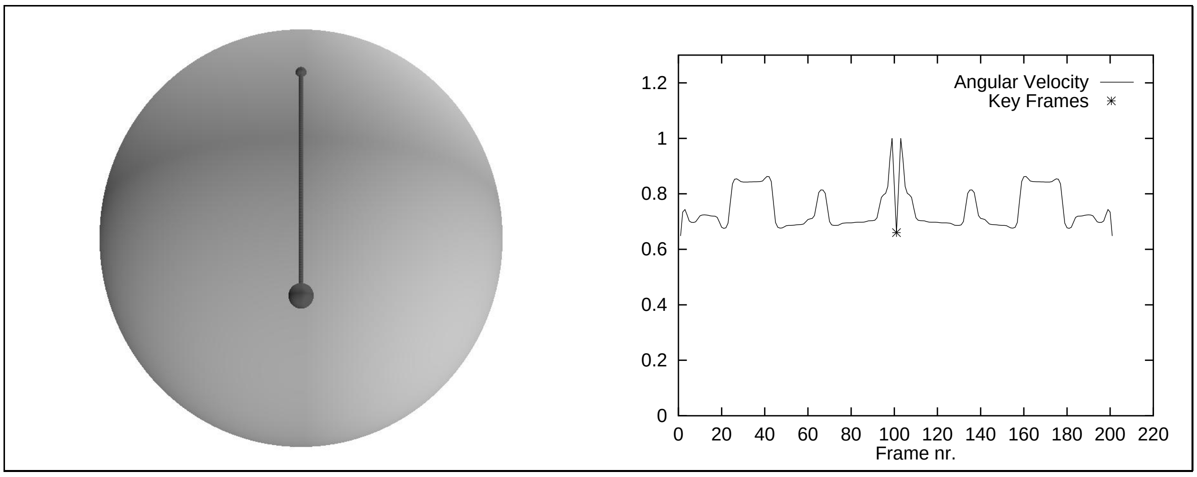


Figure 7.8: *Spring displaying pendulum motion over 200 frames and 700 iterations in three steps. The relative weight of the curvature around the key frames is 1.5 (see section 6.3.7 page 73).*

7.5 Example: A perturbed pendulum

Even though it is very reasonable that the interpolation curve remains on the same arc when the key frames are all on an arc, this is not necessarily correct. In principle the curve has infinite curvature at the center key frame. Since *Spring* is supposed to minimize curvature this is somewhat disappointing.

To understand this, it is necessary to bear the algorithm in mind. Using gradient descent, each frame will move slightly in each step to decrease the curvature. But in which direction should the frames close to the center key frame move? Since the curve is symmetric, the gradient at each frame will be zero. The problem is that this does not imply a local minimum but a local maximum instead.

Thus, the pendulum is an example of how it is possible to confuse *Spring*. We investigate this further by perturbing the first and last key frames slightly so the curve is no longer a pure pendulum. It is only just possible to see in the curve for *Squad* (figure 7.9).

Figure 7.10 shows how the pure arc is no longer a (repelling) fix point for the gradient descent algorithm *Spring*. The minimal perturbation allows the interpolation curve to be nice and rounded at the center key frame.

7.6 Example: Global properties

The final example demonstrates a fundamental difference between *Squad* and *Spring*. In each interval the interpolation curve for *Squad* is defined exclusively from the two previous and the two following key frames — i. e. a local definition. In contrast, the interpolation curve for *Spring* is globally defined.

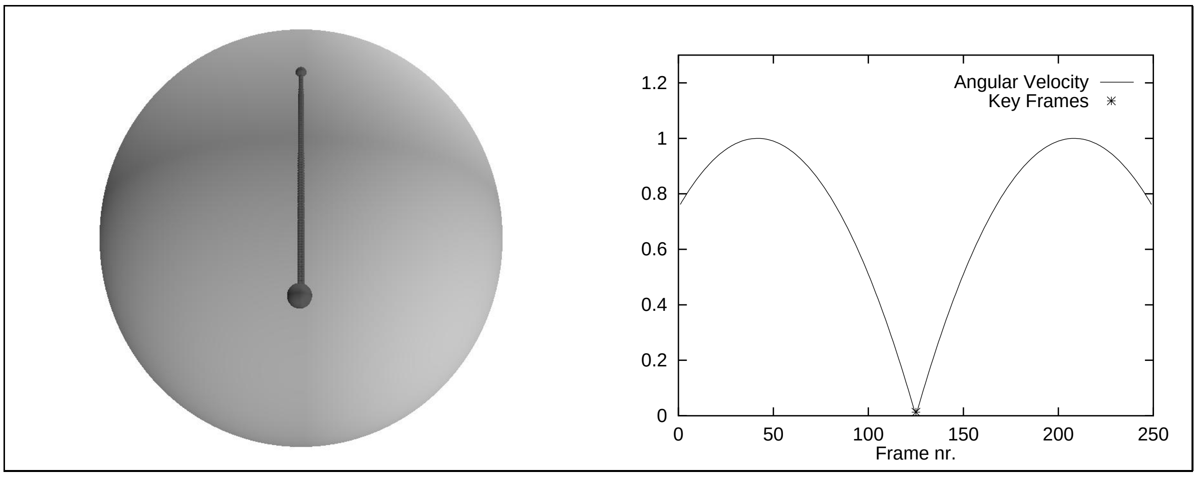


Figure 7.9: *The perturbed pendulum interpolated by Squad using 250 frames.*

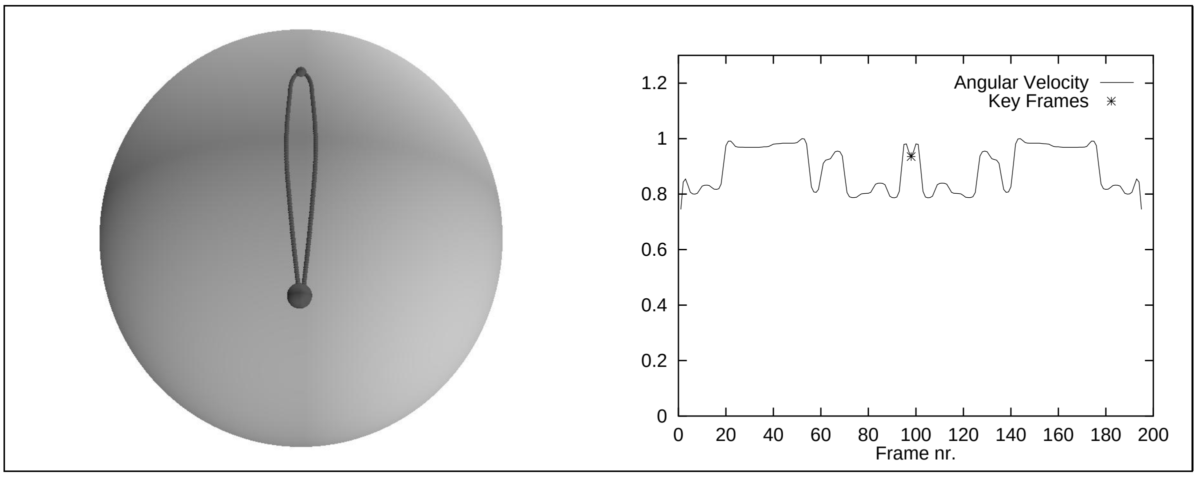


Figure 7.10: *The perturbed pendulum interpolated by Spring using 200 frames and 700 iterations in three steps. The relative weight of the curvature around the key frames is 1.5 (see section 6.3.7 page 73)*

Figure 7.11 shows the interpolation curve for *Squad* on a set of five key frames. The first three key frames lie approximately on an arc and therefore the interpolation curve is an arc in the first interval. Likewise the interpolation curve form an arc in the last interval.

In contrast the interpolation curve for *Spring* (figure 7.12) is nice and smooth. The global structure of the algorithm allows the curve to distribute the curvature evenly across all the intervals. Instead of having excessive curvature at the center key frame, a part of the curvature is propagated to the outer intervals.

It should be noted that it is necessary to add a relatively large weight to the curvature around the key frames in this example. However, we have no doubt that an algorithm with a better numerical approximation for the constituent expressions (in particular q_i'' in equation 6.26) would produce the same result — without having to fit the parameters of the program to the example.

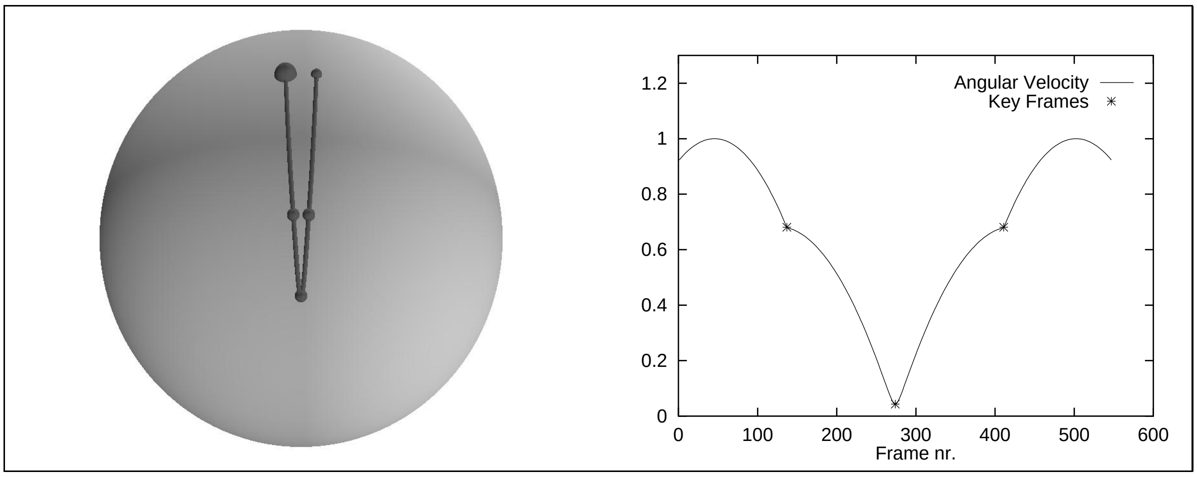


Figure 7.11: *Squad producing pointy curve using 550 frames.*

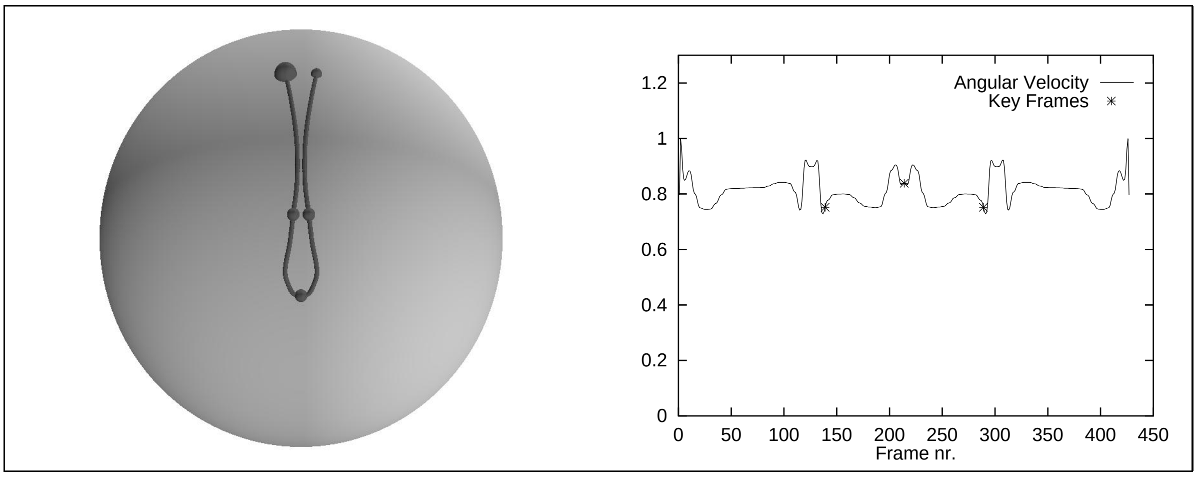


Figure 7.12: *Spring avoiding the pointy curve using 430 frames and 900 iterations. The relative weight of the curvature around the key frames is* 4 *(see section 6.3.7 page 73).*

7.7 Conclusion

The fundamental differences between the to methods are displayed below.

Property for the	**Characteristics for *Squad***	**Characteristics for *Spring***
Algorithm	Simple Analytical Continuous Local	Complex Numerical Discrete Global
Interpolation curve	Nice at simple curves. Sharp corners at key frames with large curvature	Nice at all examples. The parameters of the program must be fitted to some examples.

The choice between *Squad* and *Spring* is not obvious. If a simple algorithm yielding nice results in most cases is needed, then the simple *Squad* will suffice. If really nice interpolation is mandatory in all cases the more complex *Spring* will be more appropriate.

Chapter 8

The Big Picture

In this final chapter we will first attempt to discuss our work in relation to the available literature on quaternions and interpolation of rotations. Using this as a starting point we will point out relevant topics for future work.

8.1 Comparison to previous work

This paper has covered five main topics:

- Rotation modalities (Section 3).
- Quaternion mathematics (Section 3.3).
- Curves for interpolation of rotations — Heuristic approach (Section 6.2).
- Curves for interpolation of rotations — Analytic minimization of the local curvature (Section 6.3.5).
- Curves for interpolation of rotations – Numerical minimization of the tangential curvature (Section 6.3.7).

For each of the main topics we will summarize our contributions compared to the previous work.

Quaternion mathematics

Quaternion mathematics has been treated several times in the literature ([Hamilton, 1853], [Hamilton, 1899], [Pervin & Webb, 1992], [Shoemake, 1994b], [Maillot, 1990], and [Kim et al., 1996]). [Pervin & Webb, 1992] should be noted for a treatment of the basic mathematical properties — including the logarithm and exponential functions. In [Kim et al., 1996], a general framework for differentiation is given (though the reader is referred to a differential geometry text for the proof), and the derivative of exp is derived based on this framework. None of the articles have given a complete treatment of the necessary mathematics. Differentiation of the quaternion functions is very central in the study of the smoothness of the interpolation curves.

This report includes a comprehensive treatment of quaternion math. In particular, we have derived all the differentiation equations necessary for proving the desired properties of the interpolation curves (in section 3.3.8).

Curves for interpolation of rotations — Heuristic approach

Interpolation curves in the plane have inspired several interpolation curves for rotations. The two most important are *Slerp* [Shoemake, 1985] and *Squad* [Shoemake, 1987]. Shoemake attempts to prove the differentiability of *Squad* but the proof is flawed[1]. In [Kim et al., 1996], a more general result is given that entails the differentiability of *Squad*. Our result is derived using less advanced mathematics, and may be more easily accessible.

This is the first report which contains a comprehensive treatment of the two most important heuristic quaternion curves (section 6.2). All the known expressions for *Slerp* are stated and the correctness of their properties is proven. For *Squad* we apply the derived differentiation equations to prove the differentiability of the function.

Other heuristic approaches

Like *Squad*, a number of quaternion interpolation curves have been made from general spherical cubic curves. These curves are fitted to the control points, thus yielding relatively nice interpolation curves. Most often these curves are inspired by cubic curves in the plane.

Examples of this are the fairly simple spherical Catmull-Rom B-spline in [Schlag, 1994] and a spherical Bézier curve in [Shoemake, 1985].

In this report, we have not investigated this approach.

Curves for interpolation of rotations — Analytic approach

The first attempt to derive an optimal interpolation curve from a set of objective criteria can be seen in [Barr et al., 1992]. The paper states an expression minimizing the tangential curvature (in this paper also called the *local curvature*). However, the paper makes no attempt to state or solve the differential equation, that corresponds to the optimal curve.

In this paper we state a set of objective criteria for the optimal interpolation curve (section 6.3.5) — inspired by [Barr et al., 1992]. We then derive the fourth order differential equation whose solution will minimize the local curvature. Unfortunately, we are not able to solve the differential equation analytically.

[1]As mentioned earlier [Shoemake, 1987] is unavailable but Shoemake himself has stated [Shoemake, 1997] that the proof was flawed.

Minimization of the local curvature — Numerical approach

A brief description of a numerical solution to the problem of minimizing the tangential curvature is sketched in [Barr et al., 1992]. [Platt & Barr, 1988] contains a more sophisticated method (using *Augmented Lagrangian constraints* in a *Finite elements* method). A method yeilding faster convergence is presented in [Ramamoorthi & Barr, 1997].

We give a complete algorithm for finding a discrete solution to the numerical version of the minimization problem (section 6.3.7).

Complete treatment

To our knowledge, there exists no other complete treatment of quaternion mathematics and the applications in interpolation of rotations.

This report combines a comprehensive overview including both a thorough treatment of the basic quaternion mathematics and as well, the most important methods for interpolating orientations is space. This is where quaternions really show their strength (see sections 3.3.6, 3.4, and chapter 6).

8.2 Future work

Obviously, it would be very nice to derive an analytic solution of the differential equation that minimizes the local curvature of the interpolation curve. Since differential equations are centuries old this is not very likely to happen in the immediate future. Therefore it would be more realistic to establish a more robust numerical solution with a faster convergence. This is beyond the scope of this report. An excellent starting point for this approach would be [Platt & Barr, 1988] and [Ramamoorthi & Barr, 1997].

Another direction could be the development of more specialized applications. As an example, the movements of a camera should not necessarily be interpolated in the same manner as the moving object during animation. A stable horizon is possibly a desired feature for the camera (i. e. the camera must not tilt upside down). Relevant introductions to this line of work are [Shoemake, 1994b] and [Shoemake, 1994a].

Another example of specialization is applications where the interpolation need not be smooth. For instance, in the plane the interpolation of a fly or a UFO need not be smooth either. An example of extraction of certain properties of interpolation curves in 3D (*tension, continuity* and *bias control*) can be found in [Kochanek & Bartels, 1984].

Finally, yet another example of specialization of the interpolation curve can be studied in [Barr et al., 1992]. In this method, the angular velocity can be given explicitly at the first and last key frame (the authors call this *Angular Velocity Constraints*). An obvious generalization of this would be the ability to supply the angular velocity at an arbitrary key frame.

Acknowledgements

We would like to thank Ken Shoemake, who patiently and enthusiastically helped with answers to questions posed via e-mail. We also owe thanks to Jørgen Sand[2], who offered comprehensive suggestions concerning differential equations and the calculus of variations, and Gerd Grubb[3], who helped with differential equations.

For thorough proof reading of the Danish version we would like to thank Tommy Højfeld Olesen. We would like to thank Theo Engell for the illustration in the introduction. Finally we would like to thank our advisor Knud Henriksen.

[2]Associate Professor at the Institute of Computer Science, the University of Copenhagen, specializing in the solution of systems of equations.

[3]Professor at the Institute of Mathematics, the University of Copenhagen, specializing in differential equations.

Appendix A

Conventions

In this report we have used following conventions:

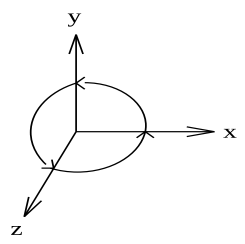
Coordinate system

We use a **right-handed** coordinate system. In computer graphics it is common to use a left-handed coordinate system. This allows the z-axis to point “into” the screen which seems natural. Since we primarily use coordinates for mathematical derivations we have chosen to use the mathematical standard — the right-handed coordinate system.

Rotation

Still using the mathematical standard we rotate counter-clockwise. The direction of rotation about an axis is obtained by the right-hand rule: Hold the axis with right hand and the thumb pointing in the positive direction of the axis. A positive rotation will now rotate in the direction of the fingers (apart from the thumb).

This is illustrated below:



Rotation about	z	*brings*	x	*into*	y
Rotation about	y	*brings*	z	*into*	x
Rotation about	x	*brings*	y	*into*	z

Euler angles

Rotation by Euler angles is defined by a rotation about each of the three coordinate axes. To make this unambiguous it is necessary to define the order of rotation.

The specific order of rotation is of no importance in this paper and we arbitrarily choose x, y, z. Other conventions are described in [Craig, 1986].

Appendix B

Conversions

In this appendix we show the conversions between different representations for rotation: Euler angles, matrices, and quaternions.

B.1 Euler angles to matrix

Rotation about the x-axis by the angle α followed by rotation about the y-axis by the angle β concluded by rotation about the z-axis by the angle γ is written in matrix[1] notation:

$$\begin{aligned}
R(\alpha,\beta,\gamma) &= R_z(\gamma)R_y(\beta)R_x(\alpha) \\
&= \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} \cos\beta\cos\gamma & \cos\gamma\sin\alpha\sin\beta - \cos\alpha\sin\gamma & \cos\alpha\cos\gamma\sin\beta + \sin\alpha\sin\gamma & 0 \\ \cos\beta\sin\gamma & \cos\alpha\cos\gamma + \sin\alpha\sin\beta\sin\gamma & -(\cos\gamma\sin\alpha) + \cos\alpha\sin\beta\sin\gamma & 0 \\ -\sin\beta & \cos\beta\sin\alpha & \cos\alpha\cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}$$

B.2 Matrix to Euler angles

The rotation matrix derived above is the starting point for the conversion from matrix to Euler angles. The conversion to Euler angles requires the inverse trigonometric functions. Neither arcsin nor arccos will, by itself, yield values on the entire interval from $-\pi$ to π. We therefore want to establish both the sin and cos values for all the angles. Using both the sin and cos values the original angles can be determined in the interval $]-\pi,\pi]$ as $v = \operatorname{sgn}(\sin v)\arccos(v)$, where sgn is defined: $\operatorname{sgn}(0) = 0$ and $\operatorname{sgn}(x) = \frac{x}{|x|}$.

[1]Using homogeneous matrices the rotation matrices are 4×4.

We can directly determine $\sin\beta$ as $-R_{31}$. Isolating $\cos\beta$ yields the following equations:

$$\begin{aligned}
\cos^2\beta &= (-R_{11}R_{32}R_{31} - R_{33}R_{21})/R_{12} \\
\cos^2\beta &= (-R_{11}R_{33}R_{31} + R_{32}R_{21})/R_{13} \\
\cos^2\beta &= (-R_{32}R_{31}R_{21} + R_{11}R_{33})/R_{22} \\
\cos^2\beta &= (-R_{31}R_{33}R_{21} - R_{11}R_{32})/R_{23}
\end{aligned}$$

From this it is not possible to determine the sign of $\cos\beta$. This means that we might as well determine β directly from $sin\beta$. Either way it is only possible to determine β in the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$ corresponding to the assumption that $\cos\beta$ is positive.

Determining cos and sin for α and γ requires $\cos\beta$. If the assumption that $\cos\beta$ is positive does not hold, then also α and γ are determined incorrectly. Unfortunately this is the best that can be done.

The equations for cos and sin for each of the constituent angles are shown below. From this the angles can be determined as stated above.

$$\begin{aligned}
\beta &= \arcsin(-R_{31}) \quad (\text{Assuming } \beta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]) \\
\cos\alpha &= \frac{R_{33}}{\cos\beta} \\
\sin\alpha &= \frac{R_{32}}{\cos\beta} \\
\cos\gamma &= \frac{R_{11}}{\cos\beta} \\
\sin\gamma &= \frac{R_{21}}{\cos\beta}
\end{aligned}$$

Obviously this requires that $\cos\beta \neq 0$. If $\cos\beta = 0$ then $\beta = \pm\frac{\pi}{2}$. This corresponds to gimbal lock (see section 4.1) and therefore it is impossible to distinguish α from γ. Thus we arbitrarily define $\gamma \equiv 0$. In this situation the angles can be determined:

$$\begin{aligned}
\beta &= \arcsin(-R_{31}) \quad (\text{assuming} \beta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]) \\
\cos\alpha &= R_{22} \\
\sin\alpha &= -R_{23} \\
\gamma &= 0
\end{aligned}$$

B.3 Quaternion to matrix

Rotation of the vector $p = (x, y, z)$ with the quaternion q is done by the operation $q\ [0, p]\ q^{-1}$. We want to determine the corresponding matrix which multiplied on $[x\ y\ z\ 1]^T$ from the left will yield the same result.

The product of two quaternions $q_v = [w, (a, b, c)]$ and $q_h = [s, (x, y, z)]$ (written in $\mathbf{i}, \mathbf{j}, \mathbf{k}$-notation) is:

$$\begin{aligned} q_v q_h &= (w + \mathbf{i}a + \mathbf{j}b + \mathbf{k}c)(s + \mathbf{i}x + \mathbf{j}y + \mathbf{k}z) \\ &= (ws - ax - by - cz) + \mathbf{i}(as + wx - cy + bz) + \\ &\quad \mathbf{j}(bs + wy + cx - az) + \mathbf{k}(cs + wz - bx + ay) \end{aligned}$$

Written as columns using sloppy notation[2] this equals:

$$q_v q_h = \begin{bmatrix} a \\ b \\ c \\ w \end{bmatrix} \times_q \begin{bmatrix} x \\ y \\ z \\ s \end{bmatrix} = \begin{bmatrix} wx - cy + bz + as \\ cx + wy - az + bs \\ -bx + ay + wz + cs \\ -ax - by - cz + ws \end{bmatrix}$$

From this we can write the matrices corresponding to multiplying from the left and from the right with a quaternion. First we determine V_{q_v} such that $V_{q_v} q_h = q_v q_h$, where q_v and $q_v q_h$ are written as columns:

$$V_{q_v} = \begin{bmatrix} w & -c & b & a \\ c & w & -a & b \\ -b & a & w & c \\ -a & -b & -c & w \end{bmatrix}$$

Then we write H_{q_h}, such that $H_{q_h} q_v = q_v q_h$:

$$H_{q_h} = \begin{bmatrix} s & z & -y & x \\ -z & s & x & y \\ y & -x & s & z \\ -x & -y & -z & s \end{bmatrix}$$

We are now ready to write the matrix M, such that $Mp = q\ [0,p]\ q^{-1}$. Using $q = [s, (x, y, z)]$ and $q^{-1} = [s, (-x, -y, -z)]$ we get:

$$\begin{aligned} M &= V_q H_{q^{-1}} \\ &= \begin{bmatrix} s & -z & y & x \\ z & s & -x & y \\ -y & x & s & z \\ -x & -y & -z & s \end{bmatrix} \begin{bmatrix} s & -z & y & -x \\ z & s & -x & -y \\ -y & x & s & -z \\ x & y & z & s \end{bmatrix} \\ &= \begin{bmatrix} 1 - 2(y^2 + z^2) & 2xy - 2sz & 2sy + 2xz & 0 \\ 2xy + 2sz & 1 - 2(x^2 + z^2) & -2sx + 2yz & 0 \\ -2sy + 2xz & 2sx + 2yz & 1 - 2(x^2 + y^2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

[2]The quaternion $[s, (x, y, z)]$ is written as the column $[x, y, z, s]^T$, and $\times_q$ denotes quaternion multiplication

B.4 Matrix to Quaternion

Conversion from a rotation matrix to the corresponding unit quaternion uses the matrix M derived above. First we find s:

$$\begin{aligned} M_{11} + M_{22} + M_{33} + M_{44} &= 4 - 4(x^2 + y^2 + z^2) \\ &= 4 - 4(1 - s^2) \quad \text{Da } s^2 + x^2 + y^2 + z^2 = 1 \\ &= 4s^2 \end{aligned}$$

This yields s^2. Now the other values follow:

$$\begin{aligned} s &= \pm \frac{1}{2} \sqrt{M_{11} + M_{22} + M_{33} + M_{44}} \\ x &= \frac{M_{32} - M_{23}}{4s} \\ y &= \frac{M_{13} - M_{31}}{4s} \\ z &= \frac{M_{21} - M_{12}}{4s} \end{aligned}$$

The sign of s cannot be determined. Depending on the choice of sign for s the signs for x, y and z change as well. This means choosing between a quaternion and the corresponding negative quaternion. These quaternions yield the same rotation but the interpolation curve can be influenced by this choice.

Since the entire interpolation is calculated on quaternions this will pose no practical problem. Therefore we simply choose the positive square root.

B.5 Between quaternions and Euler angles

These conversions can simply be achieved by going via matrices using the conversions stated above.

Therefore the limitation on the β angle from the conversion between matrix and Euler angles will hold for the conversion from quaternions to Euler angles as well. It is not possible to avoid this limitation (or a similar one) using a direct conversion between quaternions and Euler angles.

Appendix C

Implementation

This chapter contains a brief description of the program we have developed for visualization including the standard packages we have used.

`quat` displays in a window on the screen how an object (the letter "R") looks when it is rotated in space. The desired key orientations are supplied through a resource file together with various parameters. The interpolation method is supplied as a command line parameter. The available methods are:

`lineuler`	Linear interpolation between Euler angles (section 6.1.1).
`linmat`	Linear interpolation between rotation matrices (section 6.1.2).
`lerp`	Linear interpolation between quaternions (section 6.1.3)
`slerp`	Spherical linear interpolation between quaternions (section 6.1.5).
`squad`	Spherical spline interpolation between quaternions (section 6.2.1)
`slerpsvupti`	Minimization of the tangential curvature using a gradient descent method with *Slerp* as initial solution (section 6.3.7).
`squadsvupti`	Minimization of the tangential curvature using a gradient descent method with *Squad* as initial solution (section 6.3.7).
`justdoit`	Minimization of the tangential curvature using a gradient descent method applied three times with *Slerp* as initial solution (section 6.3.7).

The curious names `slerpsvupti`, `squadsvupti` and `justdoit` are used for historical reasons.

We use the graphics library SPHIGS [Sklar & Brown, 1993] for displaying the animation on the screen. However, we have added the ability to export the animation as a series of PPM files. The PPM files are converted to an animated GIF using the program `convert` [ImageMagic, 1997]. A few examples of this can be seen at `http://kantine.diku.dk/~myth/gif`

The program `quat` produces a visualization of the interpolated quaternions and an approximation of the velocity (see chapter 5). The velocity is displayed as a two dimensional graph generated by the program `gnuplot` [Williams & Kelley, 1993]. The visualization of the interpolation curve is made using the ray tracer POV-ray [POV, 1997].

C.1 The basic structure of `quat`

We have used C++ for writing **`quat`**. The object oriented language allows us to implement classes for each mathematical concept: matrix, vector, quaternion and so on. Apart from this separate objects handle interpolation and visualization.

The source code can be obtained from the authors.

Bibliography

[Barr et al., 1992] Alan H. Barr, Bena Currin, Steven Gabriel, & John F. Hughes. Smooth interpolation of orientations with angular velocity constraints using quaternions. *Computer Graphics*, 26(2):313–320, July 1992.

[Burtnyk & Wein, 1971] Nester Burtnyk & Marceli Wein. Computer generated keyframe animation. *SMPTE*, (80):149–153, March 1971.

[Craig, 1986] John J. Craig. *Introduction to Robotics. Mechanics and Control*, chapter 2. Addison-Wesley, 1986.

[Euler, 1752] Leonhard Euler. Decouverte d'un nouveau principe de méchanique. *Opera omnia (1957)*, Ser. secunda(Vol. 5):81–108, 1752. Orell Füsli Turici.

[Foley et al., 1990] James D. Foley, Andries van Dam, Steven K. Feiner, & John F. Hughes. *Computer Graphics Principles and Practice.* Addison-Wesley, Reading, Massachusetts, 2nd. edition, 1990.

[Hallenberg et al., 1993] Niels Hallenberg, Martin Koch, & Ole Fogh Olsen. Vektorernes opståen og udvikling (*The construction and development of vector calculus*). 1993.

[Hamilton, 1853] Sir W. R. Hamilton. *Lectures on Quaternions.* Hodges Smith & Co., Dublin, 1853.

[Hamilton, 1899] Sir W. R. Hamilton. *Elements of Quaternions*, volume 1-2. Longmans, Green and Co., 1899.

[ImageMagic, 1997] ImageMagic. *Convert.* E. I. du Pont de Nemours and Company, 1997. Part of the ImageMagic version 3.9.0 library. `http://www.wizards.dupont.com/cristy-/ImageMagick.html`.

[Jakobsen, 1993] Hans Plesner Jakobsen. *Course notes for Mathematics 3GE (differential geometry).* Matematisk Notetryk, Institute of Mathematics, University of Copenhagen, Denmark, Copenhagen, 1993.

[Kim et al., 1996] Myoung-Jun Kim, Myung-Soo Kim, & Sung Yong Shin. A compact differential formula for the first derivative of a unit quaternion curve. *The Journal of Visualization and Computer Animation*, 7:43–57, 1996.

[Kincaid & Cheney, 1991] David Kincaid & Ward Cheney. *Numerical Analysis.* Brooks/Cole Publishing Company, Pacific Grove, California, 1991.

[Kochanek & Bartels, 1984] Doris H. U. Kochanek & Richard H. Bartels. Interpolating splines with local tension, continuity, and bias control. *Computer Graphics*, 18:33–41, July 1984.

[Lasseter, 1987] John Lasseter. Principles of traditional animation applied to 3D computer animation. *Computer Graphics*, 21(4):35–44, July 1987.

[Madsen, 1991] Tage Gutmann Madsen. *Course notes for Mathematics 1MA (calculus)*. Matematisk Notetryk, Institute of Mathematics, University of Copenhagen, Denmark, Copenhagen, 1991.

[Maillot, 1990] Patrick-Gilles Maillot. Using quaternions for coding 3d transformations. In Andrew Glassner, editor, *Graphics Gems 1*, chapter 10, pages 498–515. Academic Press, Inc., 1990.

[McCool, 1995] Michael McCool. Orientation interaction tester. `http://www.cgl.uwaterloo.ca/Gallery/image_html/gimbal.jpg.html`, June 1995.

[Pervin & Webb, 1992] Edward Pervin & Jon A. Webb. *Quaternions in Computer Vision and Robotics*. Carnegie-Mellon University, 1992.

[Platt & Barr, 1988] John C. Platt & Alan H. Barr. Constraint methods for flexible models. *Computer Graphics*, 22(4):279–288, August 1988.

[POV, 1997] POV-team. *Persistence of Vision Ray Tracer*, Febuary 1997. `http://www.povray.org`.

[Ramamoorthi & Barr, 1997] Ravi Ramamoorthi & Alan H. Barr. Fast construction of accurate quaternion splines. *Computer Graphics*, pages 287–292, 1997.

[Schlag, 1994] John Schlag. Using geometric constructions to interpolate orientation with quaternions. *Graphics Gems IV*, pages 230–236, 1994.

[Schwarz, 1989] H. R. Schwarz. *Numerical Analysis, A Comprehensive Introduction*. John Wiley & Sons, Chicester, 1989.

[Shoemake & Duff, 1994] Ken Shoemake & Tom Duff. Matrix animation and polar decomposition. `ftp://ftp.cis.upenn.edu/pub/graphics/shoemake/polar-decomp.ps.Z`, 1994.

[Shoemake, 1985] Ken Shoemake. Animating rotation with quaternion curves. *Computer Graphics*, 19(3):245–254, 1985.

[Shoemake, 1987] Ken Shoemake. Quaternion calculus and fast animation. *SIGGRAPH Course Notes*, 10:101–121, 1987. **Not available**.

[Shoemake, 1994a] Ken Shoemake. Fiber bundle twist reduction. *Graphics Gems IV*, pages 230–236, 1994.

[Shoemake, 1994b] Ken Shoemake. Quaternions. `ftp://ftp.cis.upenn.edu/pub/graphics/-shoemake/quatut.ps.Z`, 1994.

[Shoemake, 1997] Ken Shoemake. `Re: Siggraph 1987 tutorial`, June 1997. E-mail correspondence June/July 1997.

[Sklar & Brown, 1993] David Frederick Sklar & Christopher R. Brown. *Simple Programmer's Hierarchical Graphics Standard (SPHIGS) for ANSI-C Version 1.0*, March 1993. A detailed description can be found in [Foley et al., 1990].

[Verplaetse, 1995] Christopher Verplaetse. Can a pen remember what it has written using inertial navigation?: An evaluation of current accelerometer technology. `http://verp.www.media.mit.edu/projects/SmartPen/smartpen.html`, May 1995.

[Watt & Watt, 1992] Alan Watt & Mark Watt. *Advanced Animation and Rendering Techniques Theory and Practice*, chapter 15. Addsion-Wesley, Wokingham, England, 1992.

[Williams & Kelley, 1993] Thomas Williams & Colin Kelley. *GNUPLOT — An Interactive Plotting Program Version 3.4*, June 1993. `http://science.nas.nasa.gov/~woo/gnuplot/gnuplot.html`.