Pré-Publicações do Departamento de Matemática
Universidade de Coimbra
Preprint Number 04–38

A TWO-STEP ALGORITHM OF SMOOTH SPLINE GENERATION ON RIEMANNIAN MANIFOLDS

JANUSZ JAKUBIAK, FÁTIMA SILVA LEITE AND RUI C. RODRIGUES

Abstract: This paper presents a simple geometric algorithm to generate splines of arbitrary degree of smoothness in Euclidean spaces. Unlike other existing methods, this simple geometric algorithm does not require a recursive procedure and, consequently, introduces a significant reduction in calculation time. The algorithm is then extended to other complete Riemannian manifolds, namely to matrix Lie groups and spheres.

Keywords: Spline functions, interpolation, Riemannian manifolds, Lie groups, rigid body motions.

1. Introduction

Motivated by many engineering applications, interpolation problems through arbitrary points in Euclidean spaces have been studied for a long time and numerous interpolation algorithms have been designed. The most well known geometric method to generate polynomial splines in Euclidean spaces is the de Casteljau algorithm, which is based on recursive linear interpolations. For an overview of existing works we refer to [1] and literature cited therein.

With the development of robotics and computer graphics it has become important to find planning methods of spatial motion of rigid bodies, where not only position, but also orientation, are required to change smoothly. Shoemake in [2] introduced the idea of interpolating in $SO(3)$, using quaternions. His work was further enhanced in [3, 4]. A generalization of the de Casteljau algorithm to general Riemannian manifold was proposed in [5]. The fundamental idea was to replace line segments in an Euclidean space with geodesics in a manifold. Explicit formulas for the de Casteljau algorithm on compact Lie groups and spheres were further developed in [6]. Algorithms for interpolating rigid body motion in 3D space with minimum acceleration

Received December 10, 2004.
Work supported in part by project POSI/SRI/41618/2001 and Control Training Site HPMT-CT-2001-00278, while the first author was visiting ISR-Coimbra.

were presented in [7, 8]. Another application of the algorithm to the Euclidean group $SE(3)$ in cases of different choice of metrics was presented in [9]. Another approach was proposed in [10], where splines in $SO(n)$ are defined as a projection of optimal splines in $SL^+(n)$.

When the spline is required to minimize a certain cost functional, its generation depends on a variational approach. Such an approach for Riemannian manifolds was undertaken in [11] and then continued in [12, 13]. In [14] a method of finding an optimal spline in $SE(3)$ was proposed, however the solutions with minimal average velocity or acceleration impose additional constraints on the boundary conditions.

One of our objectives in this paper is to present a geometric algorithm which generates smooth interpolating splines using the minimum number of steps. In addition it is important that the computation of each spline segment (between two consecutive points) relies on local data only, so that the change of values at a particular knot, doesn't imply recalculation of the whole curve, but only neighbor segments. The idea of the algorithm follows [15,16], where a spline segment in $\mathbb{R}^3$ is obtained as a convex combination of lines and arcs. In [17] a geometric algorithm exploiting three supporting lines was presented. Herein the spline is designed as a combination of two polynomials defined by boundary values and a smoothing function that ensures satisfactory smoothness of the spline at connecting points.

The paper is organized as follow: section 2 contains the statement of the problem. Section 3 presents the algorithm for the Euclidean space $\mathbb{R}^d$ and its main properties. A generalization of this algorithm to Riemannian manifolds, in particular Lie groups and spheres is shown in section 4. Section 5 summarizes the paper.

2. Problem statement

Throughout the paper M denotes a complete Riemannian manifold and T_pM is the tangent space to M, at a point $p \in M$. A *k-frame* P in M at p is an ordered set $P = \left(p, \dot{p}, \ddot{p}, \cdots, p^{(k)}\right)$, consisting of the point p and k vectors, $\dot{p}, \ddot{p}, \cdots, p^{(k)}$, tangent to M at p. The general problem to be studied in this paper, has the following statement:

Problem $\mathcal{C}_k$ - Given n distinct points $p_0, p_1, \cdots, p_{n-1}$ in M, n k-frames $P_0, P_1, \cdots, P_{n-1}$ in M at the points $p_0, p_1, \cdots, p_{n-1}$ respectively, and a partition $\Delta : 0 = t_0 < t_1 < \cdots < t_{n-1} = T$ of the time interval $[0, T]$, generate a $\mathcal{C}^m$-smooth curve $(m \geq k \geq 1)$ $s : [0, T] \subset \mathbb{R} \to M$ on M, which satisfies the following interpolation conditions:

$$\Big(s(t_i), \dot{s}(t_i), \ddot{s}(t_i), \cdots, s^{(k)}(t_i)\Big) = P_i, \quad 0 \le i \le n-1. \tag{1}$$

3. Interpolation algorithm on Euclidean spaces

We start with the case when M is the d-dimensional Euclidean space $\mathbb{R}^d$. Given a k-frame $P = \big(p, \dot{p}, \ddot{p}, \cdots, p^{(k)}\big)$ in $\mathbb{R}^d$ and $\tau \in \mathbb{R}$, there always exists a $\mathcal{C}^\infty$-smooth function $x : \mathbb{R} \to \mathbb{R}^d$ satisfying

$$\Big(x(\tau), \dot{x}(\tau), \ddot{x}(\tau), \cdots, x^{(k)}(\tau)\Big) = P. \tag{2}$$

Indeed, if L is any linear differential operator of order k and constant coefficients, it follows from the theory of existence and uniqueness of solutions of differential equations that $L(x) = 0$ has a unique $\mathcal{C}^\infty$-smooth solution which satisfies the initial conditions (2). In particular, for $L = \dfrac{d^{k+1}}{dt^{k+1}}$, such a solution is a polynomial of degree k in $\mathbb{R}^d$, given by

$$x(t) = a_0 + \sum_{j=1}^{k} a_j\,(t-\tau)^j, \qquad \text{with } a_0 = p, a_j = \frac{p^{(j)}}{j!},$$

for $j = 1, \cdots, k$.

Now, we show that given two k-frames in $\mathbb{R}^d$, $P_{i-1} = (p_{i-1}, \dot{p}_{i-1}, \cdots, p_{i-1}^{(k)})$ and $P_i = (p_i, \dot{p}_i, \cdots, p_i^{(k)})$, there exists a smooth curve $t \mapsto s_i(t)$ in $\mathbb{R}^d$, defined on the interval $[t_{i-1}, t_i]$, which satisfies the following boundary conditions

$$\begin{aligned} &\Big(s_i(t_{i-1}), \dot{s}_i(t_{i-1}), \cdots, s_i^{(k)}(t_{i-1})\Big) = P_{i-1}, \\ &\Big(s_i(t_i), \dot{s}_i(t_i), \cdots, s_i^{(k)}(t_i)\Big) = P_i. \end{aligned} \tag{3}$$

Without loss of generality, the interval $[t_{i-1}, t_i]$ may be replaced by $[0,1]$, just by choosing the re-parametrization $(t \to \tau)$ defined by $t = (1-\tau)\,t_{i-1} + \tau\,t_i$. For the sake of simplicity, in what follows we work on the $[0,1]$ interval.

Lemma 3.1. *If ϕ is a real valued smooth function satisfying*

$$\begin{array}{lclclcl} \phi(0) & = & 0, & \quad & \phi(1) & = & 1 \\ \dot{\phi}(0) & = & 0, & & \dot{\phi}(1) & = & 0 \\ & \vdots & & & & \vdots & \\ \phi^{(k)}(0) & = & 0, & & \phi^{(k)}(1) & = & 0 \end{array} \tag{4}$$

and l_i, r_i are $\mathcal{C}^\infty$-smooth vector valued functions in $\mathbb{R}^d$ satisfying

$$\begin{aligned} \left(l_i(0), \dot{l}_i(0), \cdots, l_i^{(k)}(0)\right) &= P_{i-1}, \\ \left(r_i(1), \dot{r}_i(1), \cdots, r_i^{(k)}(1)\right) &= P_i, \end{aligned} \tag{5}$$

then the curve $t \mapsto s_i(t)$ defined by

$$s_i(t) = (1 - \phi(t))\, l_i(t) + \phi(t) r_i(t) \tag{6}$$

is smooth and satisfies the boundary conditions

$$\begin{aligned} \left(s_i(0), \dot{s}_i(0), \cdots, s_i^{(k)}(0)\right) &= P_{i-1}, \\ \left(s_i(1), \dot{s}_i(1), \cdots, s_i^{(k)}(1)\right) &= P_i. \end{aligned} \tag{7}$$

Proof: It is clear from (4) and (6) that

$$s_i(0) = l_i(0) \qquad \text{and} \qquad s_i(1) = r_i(1).$$

Now, applying Leibniz formula for the derivative of a product, we get,

$$s_i^{(j)}(t) \;=\; l_i^{(j)}(t) + \sum_{s=0}^{j} \binom{j}{s} \phi^{(s)}(t)\, \left(r_i(t) - l_i(t)\right)^{(j-s)}. \tag{8}$$

Using this formula, together with the conditions (4), it follows immediately that, for $j = 0, 1, \cdots, k$,

$$s_i^{(j)}(0) = l_i^{(j)}(0) \qquad \text{and} \qquad s_i^{(j)}(1) = r_i^{(j)}(1),$$

which by (5) proves that s_i satisfies the boundary conditions (7). The smoothness of s_i results from its definition. ■

Remark 3.1. One question that still needs to be addressed is the existence of a smooth function $\phi : [0,1] \to [0,1]$ satisfying the required boundary conditions (4). For later developments it will be important to guarantee that such a function is also strictly increasing in the interval $[0,1]$ and satisfies extra higher order derivative constraints at the boundary points. Next, we exhibit a function which satisfies all these requirements.

Lemma 3.2. *The function ϕ defined by*

$$\phi(t) = \gamma \sum_{i=0}^{m} \frac{a_{m+1+i}}{m+1+i}\, t^{m+1+i}, \tag{9}$$

where $a_{m+1+i} = (-1)^i\binom{m}{i}$ *and* $\gamma^{-1} = \displaystyle\sum_{i=0}^{m} \frac{a_{m+1+i}}{m+1+i}$, *satisfies the following conditions:*

1) $\dot\phi(t) = \gamma\, t^m(1-t)^m$*;*
2) ϕ *is strictly increasing in* $[0,1]$*;*
3) ϕ *satisfies the boundary conditions (4);*
4) $\phi^{(j)}(0) = \phi^{(j)}(1) = 0$, *for* $j = k+1, \cdots, m$*;*
5) $\phi^{(m+1)}(0) = (-1)^m \phi^{(m+1)}(1) = \gamma\, m! \neq 0$.

Proof: The anti-derivatives of the function

$$g(t) = \gamma\, t^m(1-t)^m = \gamma \sum_{i=0}^{m} (-1)^i \binom{m}{i} t^{m+i}$$

are of the form

$$\begin{aligned}\int g(t)dt &= \gamma \sum_{i=0}^{m} \frac{(-1)^i\binom{m}{i}}{m+1+i}\, t^{m+1+i} + \gamma_1 \\ &= \gamma \sum_{i=0}^{m} \frac{a_{m+1+i}}{m+1+i}\, t^{m+1+i} + \gamma_1 = \phi(t) + \gamma_1,\end{aligned}$$

for γ_1 a real constant. So, ϕ is an anti-derivative of g, which proves *1)*. Since the constant γ can be rewritten in terms of the Gamma function as:

$$\gamma^{-1} = \sum_{i=0}^{m} \frac{a_{m+1+i}}{m+1+i} = \frac{2^{-1-2m}\sqrt{\pi}\, m!}{Gamma[\frac{3}{2}+m]} > 0,$$

g is positive on $]0,1[$ and, consequently, ϕ is strictly increasing in $[0,1]$. By construction, $\phi(0) = 0$ and $\phi(1) = 1$. Also $\phi^{(j)}(0) = \phi^{(j)}(1) = 0$, for $j = 1, \cdots m$, because g, together with its derivatives up to the order $m-1$, vanish at $t = 0$ and at $t = 1$. Since $k \leq m$, this proves both *3)* and *4)*. Finally, *5)* follows from a straightforward calculation of the m-th derivative of g, evaluated at $t = 0$ and $t = 1$. ∎

Lemma 3.3. *If* ϕ *is the function defined in the previous lemma, then the curve*

$$s_i(t) = (1-\phi(t))\, l_i(t) + \phi(t) r_i(t), \tag{10}$$

satisfies the the boundary conditions (7), and in addition also satisfies:

$$
\begin{aligned}
& s_i^{(j)}(0) = s_i^{(j)}(1) = 0, \textit{ for } j = k+1, \cdots, m, \\
& s_i^{(m+1)}(0) = \gamma\, m!\, (r_i(0) - l_i(0)), \qquad (11) \\
& s_i^{(m+1)}(1) = -\gamma\, m!\, (r_i(1) - l_i(1)).
\end{aligned}
$$

This results from using the conditions 3) – 5) of the previous lemma in the formula (8) for the derivatives of s_i. Except for some degenerated cases, $s_i^{(m+1)}(0)$ and $s_i^{(m+1)}(1)$ are non zero. The curve $t \mapsto s_i(t)$ defined by (10) is a convex combination of two vector-valued functions, the *left component* l_i and the *right component* r_i. The following theorem follows from the previous considerations and results.

Theorem 3.1. *The curve* $s : [0, T] \subset \mathbb{R} \to \mathbb{R}^d$ *defined by*

$$
s(t) = s_i\left(\frac{t - t_{i-1}}{t_i - t_{i-1}}\right), \quad t \in [t_{i-1}, t_i],\ i = 1, \cdots, n,
$$

where s_i *is defined in the previous lemma, is* $\mathcal{C}^m$*-smooth and satisfies the interpolation conditions*

$$
\Big(s(t_i), \dot{s}(t_i), \ddot{s}(t_i), \cdots, s^{(k)}(t_i)\Big) = P_i, \qquad 0 \le i \le n-1. \tag{12}
$$

That is, s *is a solution of the interpolation problem* $\mathcal{P}_k$ *for* $M = \mathbb{R}^d$. *Moreover,* $s^{(j)}(t_i) = 0$, *for* $j = k+1, \cdots, m$, $i = 0 \cdots, n-1$, *and, in general,* $s^{(m+1)}(t_i) \neq 0$.

Remark 3.2. (1) The curve $t \mapsto s(t)$ defined in this theorem is a polynomial spline. Each spline segment is a polynomial of degree $2m+k+1$.

(2) The left and right components of each segment of the spline are chosen so that the interpolation conditions are satisfied. They are uniquely defined from the given data. For $k = 1$, each of these components are one degree polynomials (geodesics) in $\mathbb{R}^d$.

(3) Since ϕ has been chosen strictly increasing in $[0, 1]$ and, for every i, s_i is a convex combination of the left and right components, the resulting spline function has the convex hull property. That is, for every $t \in [0, T]$, $s(t)$ lies in the convex hull of the given data.

(4) The smoothness properties of the generated spline depend highly on the choice of the function ϕ. For a fixed m, the resulting spline using

the function given in Lemma 3.2, is $\mathcal{C}^m$-smooth. For this reason, we say that ϕ is a *smoothing function* for the spline. Clearly, given m, the coefficients of the smoothing function ϕ can be immediately obtained and stored in a look-up table, in such a way that the calculation of the spline at each point will be reduced to a single application of formula (10).

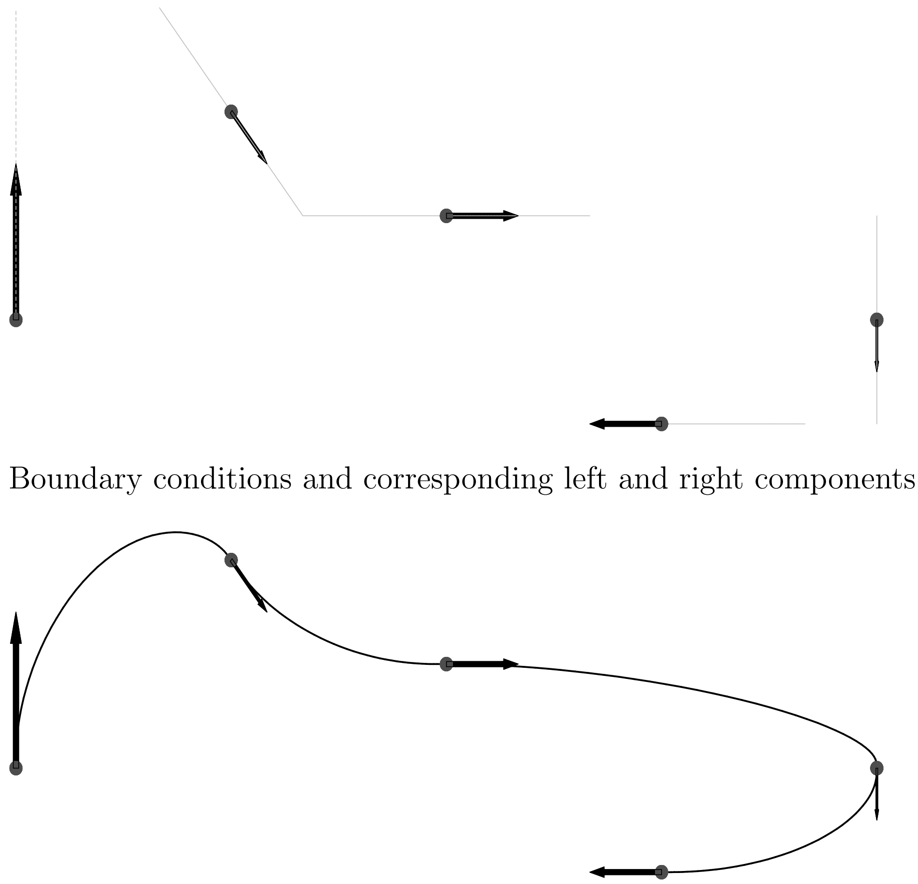
We give below examples of the function ϕ, for small values of m.

$$\begin{aligned}
&m=1, \quad \phi(t) = 3\,t^2 - 2\,t^3 \\
&m=2, \quad \phi(t) = 10\,t^3 - 15\,t^4 + 6\,t^5 \\
&m=3, \quad \phi(t) = 35\,t^4 - 84\,t^5 + 70\,t^6 - 20\,t^7 \\
&m=4, \quad \phi(t) = 126\,t^5 - 420\,t^6 + 540\,t^7 - 315\,t^8 + 70\,t^9
\end{aligned}$$

Fig. 1 illustrates the generation of an interpolating $\mathcal{C}^2$- smooth spline on the plane, with prescribed points and velocities. Before we proceed, we would like to compare the presented algorithm with the classical de Casteljau algorithm. For a required degree of smoothness m, the choice of the smoothing function $\phi(t)$ as described in lemma 3.2 produces a spline which, besides satisfying the required interpolating conditions, also has vanishing higher order derivatives, up to order m, at the knot points. If these extra conditions were also prescribed à priori, the de Casteljau algorithm could be used to generate a spline interpolating curve. However, contrary to our approach which involves two steps only, the de Casteljau algorithm requires a number of steps which increases with m. The reduction in complexity, and so in calculation time, is the major advantage of our algorithm. Besides this, the spline segments are calculated independently, so that if the data at one particular instant of time changes, there is no need to recalculate the whole spline, but only the two neighboring segments. Another important feature of our construction, also shared by the de Casteljau algorithm, is that it can be generalized to certain non-Euclidean spaces, which play an important role in many engineering applications, such as path planning of mechanical systems.

Remark 3.3. *The algorithm of spline construction can be also applied when the number of prescribed derivatives varies from point to point. In such a case the interpolation conditions* (1) *are given as*

$$\Big(s(t_i), \dot{s}(t_i), \ddot{s}(t_i), \cdots, s^{(k_i)}(t_i)\Big) = P_i, \quad 0 \le i \le n-1,$$



(a) Boundary conditions and corresponding left and right components

(b) Resulting spline

FIGURE 1. Example of spline generation

where $k_i \leq k$. As a result, the polynomials defining left and right spline components may be of a different order.

4. Interpolation algorithm on complete Riemannian manifolds

The general construction of the previous section is not applicable to any Riemannian manifold, since explicit formulas for the analogues of polynomial functions, which play the role of left and right components of each spline segment, are not known. One exception is when $k = 1$, since for this case both components are geodesics. Consequently, for $k = 1$, the previous construction can be generalized to any complete Riemannian manifold, as long as one can reduce the calculation of geodesics to a manageable form. Such is the case when M is a connected and compact Lie group or a sphere. For $k = 1$, the problem $\mathcal{P}_k$ stated in section 1 may be restated as:

Problem $\mathcal{P}_1$ - Given n distinct points $p_0, p_1, \cdots, p_{n-1}$ in M, n vectors $v_0, v_1, \cdots, v_{n-1}$ tangent to M at $p_0, p_1, \cdots, p_{n-1}$ respectively and a partition $\Delta : 0 = t_0 < t_1 < \cdots < t_{n-1} = T$ of the time interval $[0, T]$, generate a $\mathcal{C}^m$-smooth curve $(m \geq 1)$ $t \mapsto s(t)$ on M, defined on the interval $[0, T]$, which satisfies the following interpolation conditions:

$$s(t_i) = p_i, \quad \dot{s}(t_i) = v_i, \qquad 0 \leq i \leq n-1. \tag{13}$$

4.1. Interpolation algorithm on Lie groups

For a connected and compact matrix Lie group G with Lie algebra $\mathcal{L}$, any vector tangent to G at a point p is of the form Vp, for some generator $V \in \mathcal{L}$. So, the above data consists of n points $p_0, p_1, \cdots, p_{n-1}$ and n infinitesimal generators $V_0, V_1, \cdots, V_{n-1}$. We assume that G is equipped with a bi-invariant Riemannian metric. Each pair $(p, V) \in G \times \mathcal{L}$ determines a unique geodesic $t \mapsto x(t)$, which passes through p at $t = \tau$ with velocity equal to Vp. This geodesic is defined in terms of the exponential mapping by $x(t) = e^{(t-\tau)V}\, p$. So, following the construction of the previous section, the left and right components of the spline segment $t \mapsto s_i(t)$ which satisfies the boundary conditions

$$\begin{aligned} s_i(0) &= p_{i-1}, & s_i(1) &= p_i, \\ \dot{s}_i(0) &= v_{i-1} = V_{i-1}\, p_{i-1}, & \dot{s}_i(1) &= v_i = V_i\, p_i, \end{aligned} \tag{14}$$

are given respectively by

$$l_i(t) = e^{tV_{i-1}}\, p_{i-1} \quad \text{and} \quad r_i(t) = e^{(t-1)V_i}\, p_i. \tag{15}$$

In order to find an explicit formula for s_i, one uses the analogue of a convex combination of two geodesic curves. Geometrically, to obtain a point $s_i(\tau)$ one joins the point $l_i(\tau)$ (at $t = 0$) to the point $r_i(\tau)$ (at $t = 1$) by a geodesic parameterized by the function ϕ. This is similar to what has been done in the generalization of the de Casteljau algorithm for $\phi(t) = t$ (cf [5], [6]). We claim that if ϕ is the $\mathcal{C}^m$-smooth scalar function of Lemma 3.2, then the curve $t \mapsto s_i(t)$, defined by

$$s_i(t) = e^{\phi(t) B_i(t)}\, l_i(t), \tag{16}$$

where

$$B_i(t) = Log\left(r_i(t)\, l_i^{-1}(t)\right), \tag{17}$$

satisfies the required boundary conditions. In this paper we prove this result only for the cases when $m = 2$ and $G = SO(d)$. In order to check the derivative conditions it is convenient to use the following lemmas, which may be found in Sattinger and Weaver [18].

Lemma 4.1. *If $t \mapsto A(t)$ is a smooth curve in $\mathcal{L}$ and ad denotes the adjoint operator on $\mathcal{L}$ defined by $ad\, A(B) = [A, B]$, the following identities hold:*

$$\frac{d}{dt} e^{A(t)} = \Omega_A^L(t) e^{A(t)}, \quad \Omega_A^L(t) = \int_0^1 \mathrm{e}^{u\, ad A(t)}\, \dot{A}(t) du. \tag{18}$$

$$\frac{d}{dt} e^{A(t)} = e^{A(t)} \Omega_A^R(t), \quad \Omega_A^R(t) = \int_0^1 \mathrm{e}^{-u\, ad A(t)}\, \dot{A}(t) du. \tag{19}$$

Lemma 4.2. *If $t \mapsto A(t)$ and $t \mapsto B(t)$ are smooth curve in $\mathcal{L}$, then*

$$\frac{d}{dt}\Big(e^{A(t)} B(t) e^{-A(t)}\Big) = e^{A(t)} \Big\{ ad\, \Omega_A^R(t)(B(t)) + \dot{B}(t) \Big\}\, e^{-A(t)}, \tag{20}$$

$$\begin{aligned} \frac{d^2}{dt^2}\Big(e^{A(t)} B(t) e^{-A(t)}\Big) = e^{A(t)} \big\{ & ad^2\, \Omega_A^R(t)(B(t)) \\ & + ad\, \Omega_A^R(t)(\dot{B}(t)) + ad\, \dot{\Omega}_A^R(t)(B(t)) \\ & + \ddot{B}(t) + \dot{B}(t) \big\} e^{-A(t)}. \end{aligned} \tag{21}$$

Proof: First notice that, as a consequence of the alternative formulas (18) and (19), the following relations, between Ω_A^L and Ω_A^R, hold:

$$\begin{aligned} \Omega_A^L(t) &= e^{A(t)} \Omega_A^R(t) e^{-A(t)}, \\ \Omega_{-A}^L(t) &= -\Omega_A^R(t), \\ \Omega_{-A}^R(t) &= -e^{A(t)} \Omega_A^R(t) e^{-A(t)}. \end{aligned}$$

Now, applying Lemma 4.1 and the previous formulas, we get

$$\begin{aligned}\frac{d}{dt}\left(e^{A(t)}B(t)e^{-A(t)}\right) &= \Omega^L_A(t)e^{A(t)}B(t)e^{-A(t)} \\ &\quad + e^{A(t)}\dot{B}(t)e^{-A(t)} + e^{A(t)}B(t)e^{-A(t)}\Omega^R_{-A}(t) \\ &= e^{A(t)}\Omega^R_A(t)B(t)e^{-A(t)} + e^{A(t)}\dot{B}(t)e^{-A(t)} \\ &\quad - e^{A(t)}B(t)\Omega^R_A(t)e^{-A(t)} \\ &= e^{A(t)}\left\{\Omega^R_A(t)B(t) - B(t)\Omega^R_A(t) + \dot{B}(t)\right\}e^{-A(t)} \\ &= e^{A(t)}\left\{[\Omega^R_A(t), B(t)] + \dot{B}(t)\right\}e^{-A(t)}.\end{aligned}$$

The second formula is obtained from the first, replacing $B(t)$ by $[\Omega^R_A(t), B(t)] + \dot{B}(t)$. ■

Higher order derivatives are obtained similarly, but the expressions become more complicated as the order increases. Due to Leibniz formula for the derivative of a product of two functions, if one of the factors vanishes at a particular point, together with its derivatives, up to a certain order j, the same happens to the product function. Since the smoothing function ϕ satisfies $\phi^{(j)}(0) = 0,\ j = 0, \cdots, m$, the former argument can be used repeatedly to conclude that the functions $A(t) = \phi(t)\, B_i(t)$, and $\Omega^L_A(t)$ satisfy the following conditions at $t = 0$

$$\begin{aligned} &A^{(j)}(0) = 0, \quad j = 0, \cdots, m, \\ &{\Omega^L_A}^{(j)}(0) = 0, \quad j = 0, \cdots, m-1. \end{aligned} \tag{22}$$

Similarly, since the function $\psi = \phi - 1$ satisfies $\psi^{(j)}(1) = 0,\ j = 0, \cdots, m$, then the functions $C(t) = \psi(t)\, B_i(t)$, and $\Omega^L_C(t)$ satisfy the following conditions at $t = 1$

$$\begin{aligned} &C^{(j)}(1) = 0, \quad j = 0, \cdots, m, \\ &{\Omega^L_C}^{(j)}(1) = 0, \quad j = 0, \cdots, m-1. \end{aligned} \tag{23}$$

Theorem 4.1. *The curve $t \mapsto s_i(t)$ defined by (15)-(16)-(17), satisfies the following conditions:*

$$\begin{aligned} &s_i(0) = p_{i-1}, && s_i(1) = p_i, \\ &\dot{s}_i(0) = v_{i-1} = V_{i-1}\, p_{i-1}, && \dot{s}_i(1) = v_i = V_i\, p_i. \end{aligned} \tag{24}$$

Proof: The conditions $\phi(0) = 0$ and $l_i(0) = p_{i-1}$ imply that $s_i(0) = p_{i-1}$. Now, making $A(t) = \phi(t)\, B_i(t)$, one gets

$$\dot{s}_i(t) = \Omega_A^L(t)\, e^{A(t)}\, l_i(t) + e^{A(t)}\, \dot{l}_i(t).$$

But the conditions $\phi(0) = \dot{\phi}(0) = 0$ imply that $A(0) = \dot{A}(0) = 0$ and consequently $\Omega_A^L(0) = 0$ and

$$\dot{s}_i(0) = \dot{l}_i(0) = V_{i-1}p_{i-1}. \tag{25}$$

In order to prove the boundary conditions at $t = 1$ we proceed as before, but instead of working with the expression (16) we use the following alternative expression

$$s_i(t) = e^{-(1-\phi(t))B_i(t)}\, r_i(t). \tag{26}$$

This formula is easily derived from the relationship between the left and the right components that follows from (16). Indeed, since $r_i(t) = e^{B_i(t)} l_i(t)$, then

$$e^{-(1-\phi(t))B_i(t)}\, r_i(t) = e^{\phi(t)B_i(t)} e^{-B_i(t)}\, r_i(t) = e^{\phi(t)B_i(t)}\, l_i(t).$$

Now, using (26) and the values of ϕ and r_i at $t = 1$, one gets

$$s_i(1) = p_i, \qquad \dot{s}_i(1) = \dot{r}_i(1) = V_i p_i.$$

■

In the Euclidean case, higher order derivatives of s_i, from $k + 1$ to m, vanish at $t = 0$ and $t = 1$. This, however, doesn't happen necessarily in the Riemannian case. Assuming that $k = 1$ and $m = 2$, we check that the second derivative of s_i doesn't vanish at $t = 0$. Indeed, using the fact that the left component $l_i(t)$ is a geodesic in G, it follows from (16), that

$$\dot{s}_i(t) = \left(\Omega_A^L(t) + e^{ad\, A(t)} V_{i-1}\right) s_i(t), \tag{27}$$

and using Lemma 4.2 we get

$$\begin{aligned} \ddot{s}_i(t) = \big\{ &\dot{\Omega}_A^L(t) + e^{ad\, A(t)}[\Omega_A^L(t), V_{i-1}] \\ &+ \left(\Omega_A^L(t) + e^{ad\, A(t)} V_{i-1}\right)^2 \big\} s_i(t). \end{aligned} \tag{28}$$

Evaluating at $t = 0$ and taking into account that $\phi(0) = \dot{\phi}(0) = 0$, it follows that $\ddot{s}_i(0) = V_{i-1}^2 p_{i-1}$, which is not zero, except when $V_{i-1} = 0$. This is not so bad, since what is natural to expect on a Riemannian manifold is that higher order covariant derivatives of the velocity vector field vanish at $t = 0$ and $t = 1$. Here the covariant derivative of a vector field W_t along a curve in G (a Riemannian manifold imbedded in some high-dimensional Euclidean space $\mathbb{R}^l$) at each point, may be viewed as a new vector field along that curve, which results from differentiating W_t as a vector field along a curve in $\mathbb{R}^l$ and then projecting it, at each point, onto the tangent space to G at that point. When G is the special orthogonal Lie group $SO(d)$, its Lie algebra $\mathcal{L}$ consists of the vector space of skew-symmetric matrices (equipped with the commutator), denoted by $so(d)$. As usual, $\mathcal{L}$ is identified with the tangent space at the identity and, for each $p \in SO(d)$, the tangent space at p is defined by $T_pSO(d) = \{Ap : A^T = -A\}$. The left and right-invariant Riemannian metric on $SO(d)$ is induced by the Frobenius norm on the general Lie algebra $gl(d)$, defined by

$$< A, B >= trace(A^T B).$$

With respect to this metric, $T_p^{\perp}SO(d) = \{Sp : S^T = S\}$. So, in this case, the covariant derivative of the velocity vector field along the spline curve $t \mapsto s_i(t)$ defined by (15)-(16)-(17) can be easily obtained from (28), taking into account that in this case $\dot{\Omega}_A^L(t) + e^{ad\,A(t)}[\Omega_A^L(t), V_{i-1}] \in so(n)$ and $\left(\Omega_A^L(t) + e^{ad\,A(t)}V_{i-1}\right)^2$ is symmetric. We have

$$\frac{D\dot{s}_i}{dt}(t) = \left\{\dot{\Omega}_A^L(t) + e^{ad\,A(t)}[\Omega_A^L(t), V_{i-1}]\right\} s_i(t).$$

Since $\dot{\Omega}_A^L(0) = \Omega_A^L(0) = 0$, it also follows that $\frac{D\dot{s}_i}{dt}(0) = 0$. Working with the alternative formula (26), one obtains $\frac{D\dot{s}_i}{dt}(1) = 0$.

Remark 4.1. As for the Euclidean space, if one wants to construct a spline curve $t \mapsto s(t)$ which solves Problem $\mathcal{P}_1$ with the additional constraints $\frac{D\dot{s}}{dt}(t_i) = 0$ at the knot points, the generalized de Casteljau algorithm also produces a solution which is $\mathcal{C}^2$-smooth but that algorithm is computationally very expensive.

To present an exemplary spline produced by the algorithm we present its application to the design of a smooth motion of three dimensional object in space. A location of the object is then defined by six variables: position and orientation of the object, which can be also represented as an element of $SO(3) \times \mathbb{R}^3$ space. Following [19], we can calculate exponentials and logarithms of rotational part with a use of Rodrigues formula. Let there are given $\Omega \in SO(3)$ and $\hat{\omega} \in so(3)$, where $\hat{\omega}$ denotes skew-symmetric matrix corresponding to the angular velocity $\omega \in \mathbb{R}^3$. The exponential of $\hat{\omega}$ is given by

$$\Omega = e^{\hat{\omega}} = I_3 + \frac{\hat{\omega}}{||\omega||} \sin ||\omega|| + \frac{\hat{\omega}^2}{||\omega||^2} (1 - \cos ||\omega||)$$

while the logarithm of Ω is defined by

$$\hat{\omega} = \log \Omega = \frac{\alpha}{2 \sin \alpha} \left(\Omega - \Omega^T \right)$$
$$\alpha = \arccos \frac{1}{2} (\operatorname{tr} \Omega - 1)$$

The task of the algorithm is to find a spline connecting frames p_0 and p_1

$$p_0 = \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right) \quad p_1 = \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 10 \\ -3 \\ 0 \end{bmatrix} \right)$$

with the boundary velocities chosen in such a way that the rotations in the initial and in the final point are round different, orthogonal axes

$$v_0 = \left(\begin{bmatrix} \cos \frac{\pi}{2} & -\sin \frac{\pi}{2} & 0 \\ \sin \frac{\pi}{2} & \cos \frac{\pi}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 3 \\ 0 \end{bmatrix} \right)$$
$$v_1 = \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \frac{3\pi}{4} & -\sin \frac{3\pi}{4} \\ 0 & \sin \frac{3\pi}{4} & \cos \frac{3\pi}{4} \end{bmatrix}, \begin{bmatrix} 3 \\ -3 \\ 0 \end{bmatrix} \right)$$

Fig. 2 presents resulting postures of the object taken in time intervals equal to 1/6.

4.2. Interpolation algorithm on spheres

The d-dimensional sphere S^d, equipped with the Riemannian metric which is induced by the Euclidean metric of the imbedded space $\mathbb{R}^{d+1}$, is also a complete Riemannian manifold. In this case, given a point $p \in S^d$ and a vector v tangent to the sphere at that point, there exists a unique geodesic $t \mapsto x(t)$ that passes through p at time τ, with velocity v:

$$x(t) = \cos \left((t - \tau) \parallel v \parallel \right) p + \frac{\sin \left((t - \tau) \parallel v \parallel \right)}{\parallel v \parallel} v. \tag{29}$$

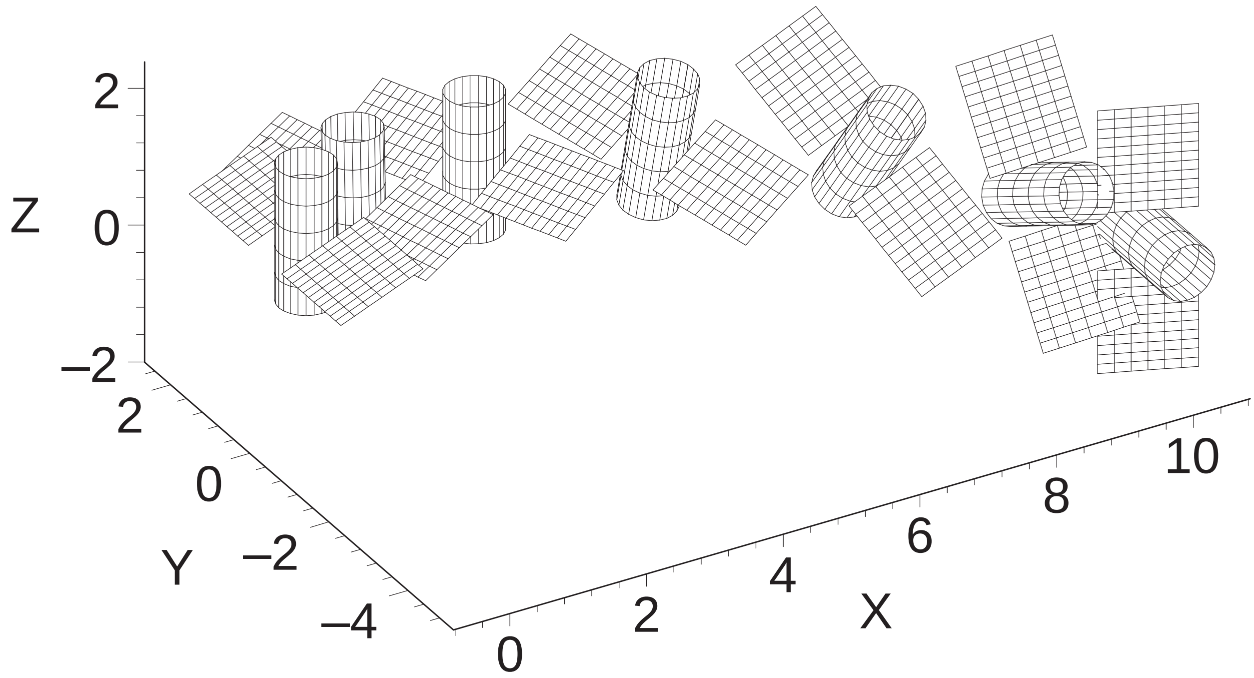


FIGURE 2. An example of a spline applied to a rigid body motion

Also, given two (not antipodal) points $p, q \in S^d$, the geodesic arc $t \mapsto y(t)$ which joins p (at $t = 0$) to q (at $t = 1$) is given by:

$$y(t) = \frac{\sin((1-t)\theta_{p,q})}{\sin\theta_{p,q}}\, p + \frac{\sin(t\theta_{p,q})}{\sin\theta_{p,q}}\, q, \tag{30}$$

where $\theta_{p,q} = \cos^{-1}(p^T q)$ is the angle between the vectors p and q. The first formula (29) is used to generate the left and right components of the spline segment $t \mapsto s_i(t)$ that joins the points p_{i-1} (at $t = 0$) to p_i (at $t = 1$) with prescribed initial and final velocity v_{i-1} and v_i respectively. As before, the left and right components $t \mapsto l_i(t)$ and $t \mapsto r_i(t)$ are required to satisfy the following conditions:

$$\begin{aligned} l_i(0) &= p_{i-1}, & r_i(1) &= p_i, \\ \dot{l}_i(0) &= v_{i-1}, & \dot{r}_i(1) &= v_i. \end{aligned} \tag{31}$$

So, according to (29), they are defined by

$$\begin{aligned} l_i(t) &= \cos\left(t \parallel v_{i-1} \parallel\right) p_{i-1} + \frac{\sin\left(t \parallel v_{i-1} \parallel\right)}{\parallel v_{i-1} \parallel} v_{i-1}, \\ r_i(t) &= \cos\left((t-1) \parallel v_i \parallel\right) p_i + \frac{\sin\left((t-1) \parallel v_i \parallel\right)}{\parallel v_i \parallel} v_i. \end{aligned} \tag{32}$$

Theorem 4.2. *The curve $t \mapsto s_i(t)$ on the sphere S^d defined by*

$$s_i(t) = \frac{\sin\left((1-\phi(t))\theta_{l_i,r_i}(t)\right)}{\sin\left(\theta_{l_i,r_i}(t)\right)} l_i(t) + \frac{\sin\left(t\theta_{l_i,r_i}(t)\right)}{\sin\left(\theta_{l_i,r_i}(t)\right)} r_i(t), \tag{33}$$

where l_i and r_i are given by (32) and ϕ is given in Lemma 3.2, satisfies the following boundary conditions

$$\begin{aligned} s_i(0) = p_{i-1}, &\quad s_i(1) = p_i, \\ \dot{s}_i(0) = v_{i-1}, &\quad \dot{s}_i(1) = v_i. \end{aligned} \tag{34}$$

Moreover, $\frac{D\dot{s}_i}{dt}(0) = \frac{D\dot{s}_i}{dt}(1) = 0.$

Proof: Before computing the spline derivatives, define the functions

$$f(t) = \frac{\sin(1-\phi(t))\theta_{rl}(t)}{\sin\theta_{rl}(t)}, \qquad g(t) = \frac{\sin\phi(t)\theta_{rl}(t)}{\sin\theta_{rl}(t)},$$

and calculate their derivatives at the boundary points $t = 0$ and $t = 1$, to obtain the following:

$$\begin{array}{llll} f(0) = 1, & f(1) = 0, & g(0) = 0, & g(1) = 1 \\ \dot{f}(0) = 0, & \dot{f}(1) = 0, & \dot{g}(0) = 0, & \dot{g}(1) = 0, \\ \ddot{f}(0) = 0, & \ddot{f}(1) = 0, & \ddot{g}(0) = 0, & \ddot{g}(1) = 0. \end{array} \tag{35}$$

The first two derivatives of $s_i(t) = f(t)l_i(t) + g(t)r_i(t)$ are given by:

$$\begin{aligned} \dot{s}_i(t) =& \dot{f}(t)l_i(t) + f(t)\dot{l}_i(t) + \dot{g}(t)r_i(t) + g(t)\dot{r}_i(t), \\ \ddot{s}_i(t) =& \ddot{f}(t)l_i(t) + 2\dot{f}(t)\dot{l}_i(t) + f(t)\ddot{l}_i(t) + \ddot{t}g(t)r_i(t) \\ &+ 2\dot{g}(t)\dot{r}_i(t) + g(t)\ddot{r}_i(t). \end{aligned} \tag{36}$$

Now, the boundary conditions (34) result from the conditions satisfied by the functions ϕ and its derivatives at $t = 0$ and $t = 1$, together with (31) and (35). To check the conditions for the covariant derivative of the velocity vector, just notice that

$$\begin{aligned}\ddot{s}_i(0) &= \ddot{l}_i(0) = - \parallel v_{i-1} \parallel p_{i-1} \perp T_{p_{i-1}}S^d,\\ \ddot{s}_i(1) &= \ddot{r}_i(1) = - \parallel v_i \parallel p_i \perp T_{p_{i-1}}S^d,\end{aligned}$$

so that, the projection onto the corresponding tangent spaces vanishes. ■

As an illustration of the algorithm we generate a spline segment on the S^2 sphere. The prescribed boundary conditions are

$$\begin{aligned} p_1 &= \begin{bmatrix}1 & 0 & 0\end{bmatrix} & v_1 &= \begin{bmatrix}0 & -0.8 & 1.2\end{bmatrix}\\ p_2 &= \begin{bmatrix}\frac{1}{2} & \frac{\sqrt{3}}{2} & 0\end{bmatrix} & v_2 &= \begin{bmatrix}-\frac{\sqrt{3}}{4} & \frac{1}{4} & -1\end{bmatrix}.\end{aligned}$$

Figure 2 shows the initial and final velocities and the resulting spline. In the graph on the right hand side, it is shown how the norm of the velocity changes in time. One can notice a high acceleration in the middle part of the graph, which corresponds to the growth of ϕ in this time interval.

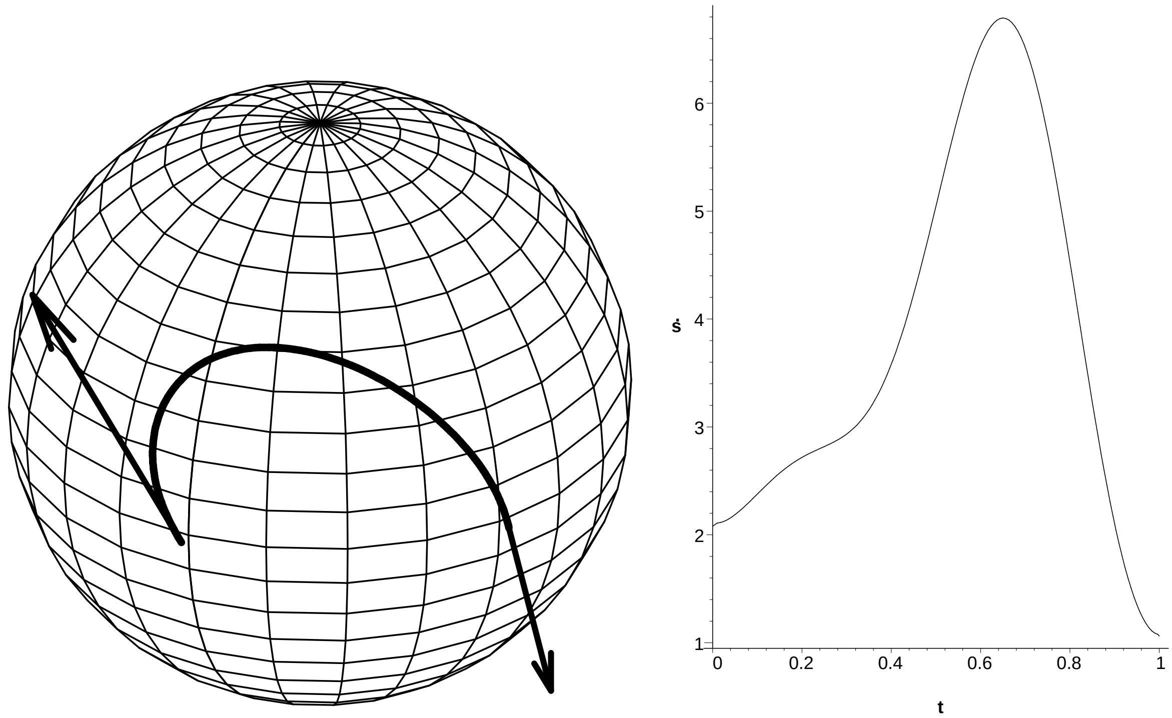


FIGURE 3. Exemplary spline on an S^2 sphere: a spline and a velocity change

5. Conclusions

The spline generation algorithm presented in this paper provides a simple geometric method of finding interpolating curves on Riemannian manifolds with any prescribed degree of smoothness. This algorithm requires the calculation of three geodesics only to generate each spline segment, while for the most used method to generate polynomial splines, the de Casteljau algorithm, that number increases with the degree of smoothness. This reduction in complexity is particularly important for non-Euclidean spaces, like rotation groups, since the implementation of these algorithms based on on the construction of geodesics implies computing matrix exponentials and matrix logarithms, which are both difficult and time consuming. Another interesting feature of the presented algorithm is the role played by the smoothing function ϕ. With appropriate choices of the boundary conditions on ϕ, one can ensure the required smoothness of the whole spline without further changes in the algorithm. The requirement of vanishing higher derivatives in the boundary points enable simple joining of consecutive spline segment without spoiling spline smoothness. Drawbacks of this approach, resulting from zero acceleration condition at the boundaries, are visible in longer trajectories and higher acceleration in the intermediate phase of the movement, but in tasks when frame data are changing, ability of finding a spline which bases on two frames only and the effectiveness of the algorithm may be a right compromise.

References

[1] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design*, 3rd ed. San Diego: Academic Press, 1993.

[2] K. Shoemake, "Animating rotation with quaternion curves," *ACM SIGGRAPH'85*, vol. 19, no. 3, pp. 245–254, 1985.

[3] Q. J. Ge and B. Ravani, "Computer aided geometric design of motion interpolants," in *Proc. ASME Design Automation Conf.*, Miami, 1991, pp. 33–41.

[4] M.-S. Kim and K.-W. Nam, "Interpolating solid orientations with circular blending quaternions curves," *Computer Aided Design*, vol. 27, no. 5, pp. 385–398, 1995.

[5] F. Park and B. Ravani, "Bézier curves on Riemannian manifolds and Lie groups with kinematic applications," *ASME J. Mechan. Design*, vol. 117, pp. 36–40, 1995.

[6] P. Crouch, G. Kun, and F. Silva Leite, "The De Casteljau algorithm on Lie groups and spheres," *Journal of Dynamical and Control Systems*, vol. 5, no. 3, pp. 397–429, 1999.

[7] M. Žefran and V. Kumar, "Planning of smooth motions on se(3)," in *Proc. Int. Conf. Robotics Automation*, Minneapolis, MN, USA, 1996, pp. 121–126.

[8] ——, "Two methods for interpolating rigid body motions," in *Proc. Int. Conf. Robotics Automation*, Leuven, Belgium, 1998, pp. 2922–2927.

[9] C. Altafini, "The De Casteljau algorithm on SE(3)," in *Nonlinear Control in Year 2000*, A. I. and F. Lamnabhi-Lagarrigue and W. Respondek, Ed. Springer-Verlag, 2000.
[10] C. Belta and V. Kumar, "An SVD-based projection method for interpolation on SE(3)," *IEEE Trans. Robot. Automat.*, vol. 18, no. 3, pp. 334–345, 2002.
[11] L. Noakes, G. Hirzinger, and B. Paden, "Cubic splines on curved spaces," *IMA J. Math. Control Information*, vol. 6, pp. 465–473, 1989.
[12] M. Camarinha, F. Silva Leite, and P. Crouch, "Splines of class C^k on non-Euclidean spaces," *IMA J. Math. Control Information*, vol. 12, no. 4, pp. 399–410, 1995.
[13] P. Crouch and F. Silva Leite, "The dynamic interpolation problem: on Riemannian manifolds, Lie groups and symmetric spaces," *Journal of Dynamical and Control Systems*, vol. 1, no. 2, pp. 177–202, 1995.
[14] M. Zefran, V. Kumar, and C. Croke, "Generation of smooth three-dimensional rigid body motions," *IEEE Trans. Robot. Automat.*, vol. 14, pp. 579–589, 1998.
[15] M. S. Nagy and T. P. Vendel, "Generating curves and swept surfaces by blended circles," *Computer Aided Geometric Design*, vol. 17, pp. 197–206, 2000.
[16] R. C. Rodrigues, F. Silva Leite, and S. Rosa, "On the generation of a trigonometric interpolating curve in R^3," in *Proceedings of 11th Int. Conf. Advanced Robotics ICAR*, Coimbra, Portugal, 2003, pp. 1629–1634.
[17] R. C. Rodrigues, F. Silva Leite, and J. Jakubiak, "A new geometric algorithm to generate interpolating curves on riemannian manifolds," (submitted for publication).
[18] D. Sattinger and O. Weaver, *Lie groups and algebras with applications to physics, geometry and mechanics*, ser. Appl. Math. Sci. 61. Springer-Verlag, 1986.
[19] R. M. Murray, Z. X. Li, and S. S. Sastry, *A mathematical introduction to robotics manipulation.* London: CRC Press, 1994.

Janusz Jakubiak
Institute of Engineering Cybernetics, Wrocław University of Technology, ul. Janiszewskiego 11/17,50-370 Wrocław, Poland

E-mail address: `Janusz.Jakubiak@pwr.wroc.pl`

Fátima Silva Leite
Departamento de Matemática, Universidade de Coimbra, 3001-454 Coimbra, Portugal

E-mail address: `fleite@mat.uc.pt`
URL: `http://www.mat.uc.pt/~fleite`

Rui C. Rodrigues
Departamento de Física e Matemática, Instituto Superior de Engenharia, Rua Pedro Nunes, 3030-199 Coimbra, Portugal

E-mail address: `ruicr@isec.pt`
URL: `http://www.isec.pt/~ruicr`