Spline Joints for Multibody Dynamics

Sung-Hee Lee* Demetri Terzopoulos†
University of California, Los Angeles

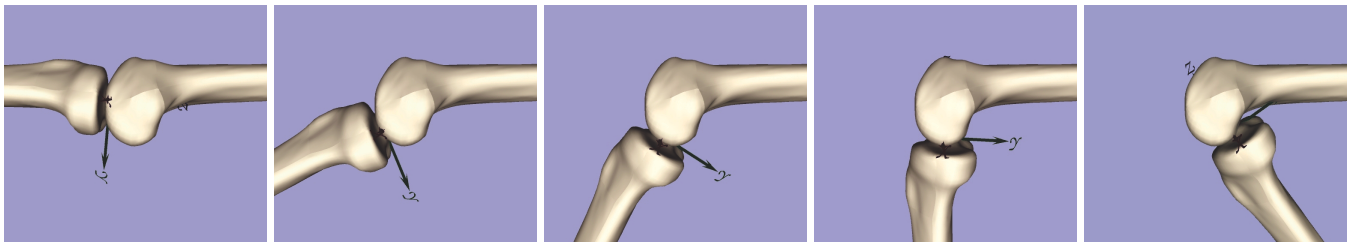


Figure 1: *A spline joint can much more accurately model complex biological joints than is possible using conventional joint models.*

Abstract

Spline joints are a novel class of joints that can model general scleronomic constraints for multibody dynamics based on the minimal-coordinates formulation. The main idea is to introduce spline curves and surfaces in the modeling of joints: We model 1-DOF joints using splines on SE(3), and construct multi-DOF joints as the product of exponentials of splines in Euclidean space. We present efficient recursive algorithms to compute the derivatives of the spline joint, as well as geometric algorithms to determine optimal parameters in order to achieve the desired joint motion. Our spline joints can be used to create interesting new simulated mechanisms for computer animation and they can more accurately model complex biomechanical joints such as the knee and shoulder.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: Biological Joints, Multibody Dynamics, Scleronomic Joints, Splines

1 Introduction

Traditionally, only a few types of elementary joints have been used to model articulated multibody systems for physics-based animation, robotics, and human movement research. These are the *lower pairs* [Reuleaux 1876]; i.e., the prismatic, helical, cylindrical, planar, and spherical joints, and their compounds, such as the universal joint. The lower pairs, which are used for modeling most mechanical or biological systems, are characterized by one or more fixed axes of rotation or translation.

When it comes to designing practical machines, using only the lower pair joints seems reasonable, not because they are ideal choices for every mechanism, but because it is difficult to manufacture more complex types of joints. For the same reason, the creation of more sophisticated joints has been largely neglected in multibody dynamics research. Not surprisingly, therefore, most dynamics simulators and game physics engines, such as ADAMS (www.mscsoftware.com), the Open Dynamics Engine (www.ode.org), and SD/FAST (www.sdfast.com), provide only fairly simple types of joint models limited to fixed joint axes.

By contrast, more complex joints are common in biological systems. Due to the complicated shapes of bones, biological joints usually produce non-trivial movement patterns. For example, the femorotibial joint (Fig. 1) undergoes both rotation and sliding as it is flexed and extended [Kapandji 1974]; it cannot be accurately approximated by a lower pair. While it may be reasonable to approximate biological joints, such as those in the neck, by lower pairs when one is interested only in macroscopic joint articulation [Lee and Terzopoulos 2006], the more accurate analysis and simulation of biological joint motion is important in biomechanics research and medical applications such as virtual surgery simulation or prosthetics design [Delp et al. 1990].

We propose a technique for modeling general scleronomic joints for multibody dynamics based on the minimal-coordinates (or *reduced*-coordinates) formulation. The scleronomic joint imposes bilateral, time-invariant contact constraints. Thus, the relative configuration of the connected bodies is determined wholly by the joint coordinates. While maximal-coordinate dynamics approaches provide the means to simulate arbitrary scleronomic joint constraints, to our knowledge ours is the first generalized approach to modeling arbitrary scleronomic joints for multibody dynamics based on the minimal-coordinates approach, which offers important advantages.

In designing a general scleronomic joint, our key idea is to use splines to model arbitrary, complex joint motions. Hence, we call these *spline joints*. Specifically, we formulate the 1-DOF (degree of freedom) spline curve joint as the product of exponentials of a twist multiplied by a spline basis function, thus defining an arbitrary C^2-continuous spline motion curve ($\mathbb{R} \mapsto \mathrm{SE}(3)$) that is free of singularities. We furthermore present geometric data-fitting and smoothing algorithms for 1-DOF spline joint design. Since higher-dimensional, analytically differentiable splines on SE(3) are not yet known, we formulate an *n*-DOF spline joint as the product of six exponentials of a basis twist multiplied by an *n*-parameter spline.

*www.cs.ucla.edu/∼sunghee
†www.cs.ucla.edu/∼dt

ACM Reference Format
Lee, S., Terzopoulos, D. 2008. Spline Joints for Multibody Dynamics. *ACM Trans. Graph. 27*, 3, Article 22 (August 2008), 8 pages. DOI = 10.1145/1360612.1360621 http://doi.acm.org/10.1145/1360612.1360621.

An advantage of spline joints is that one can employ existing dynamics algorithms without modification. Our technique exploits minimal-coordinates-based dynamics algorithms in modeling general scleronomic constraints. Unlike maximal-coordinates approaches, the spline joint does not suffer from the "drift problem". Hence, it does not require a stabilization process, and this allows larger simulation time steps. As the derivatives of the joint up to the second order are required for dynamics simulation, we provide efficient recursive algorithms to compute the analytic joint Jacobian and Hessian. The former also enables the easy computation of the inverse kinematics of the joint.

In addition to enabling the more accurate modeling of biological joints, such as the knee shown in Fig. 1, spline joints can also be applied in computer graphics to create mechanisms more interesting than those made of simple joints. One has more freedom in designing complex joints for kinematic or dynamic computer animation because they need not be easy to manufacture. We apply our spline joint models to create some interesting animated mechanisms.

2 Related Work

Researchers have endeavored to create complex models of biological joints, but no prior effort has provided a suitably complex joint model that can be used in dynamics simulation. Delp et al. [1990] and Maciel et al. [2002] modeled the knee as a revolute joint whose joint axis translates along a parametric curve. Shao and Ng-Thow-Hing [2003] proposed a general framework for modeling complex joints by composing elementary joint components. Their method is useful for forward kinematics, but they too did not consider inverse kinematics or dynamics.

Our work is related to research on spline curves for rotation. The splines on $SO(3)$ that smoothly interpolate rotations are rather involved. Shoemake [1985] proposed the interpolation of rotations using quaternions. Various techniques have been developed to achieve optimal spline curves in $SO(3)$ that minimize the tangential acceleration [Gabriel and Kajiya 1985; Barr et al. 1992; Kim et al. 1995; Ramamoorthi and Barr 1997; Park and Ravani 1997]. Since the interest is in interpolating the rigid motion of a single body, they all rightfully divide the problem on $SE(3)$ into an interpolating rotation in $SO(3)$ and translation in $\mathbb{R}^3$. In contrast, we are interested in the articulated motion of a jointed, multibody system. Hence, we treat rigid-body motion as a screw motion, without decomposing it into rotation and translation. We adopt the spline algorithms on $SO(3)$ proposed by Kim et al. [1995] and extend their methods to $SE(3)$ for use in our spline curve joints. Their approach provides a simple form of the derivatives of the spline curve, which makes it easier to compute the dynamics of spline joints.

Kry and Pai [2003] introduced a continuous surface contact simulation technique in a minimal-coordinates dynamics framework. Since they handle general topological surfaces obtained by subdivision, the computation of derivatives is more involved. By contrast, our method features the efficient computation of derivatives thanks to the structure of the joint representation. Tändl and Kecskeméthy [2007] use the Frenet frame of spline curves in $\mathbb{R}^3$ for the dynamics simulation of simple mechanisms. The orientation of the Frenet frame is determined entirely by the spline curve, whereas the orientation of the frame in our model is independent of its position along the curve, which enables us to model arbitrary rotations along the motion curve. Moreover, we use C^2-continuous splines, whereas they must use C^4-continuous splines.

Our spline joint can easily be incorporated into current dynamics algorithms. Featherstone [1987] developed an articulated-body dynamics algorithm for multiple-DOF joints, such as the universal joint. One can use this dynamics algorithm without modification to simulate spline joints (see Appendix A).

3 Geometric Preliminaries

This section briefly introduces geometric tools derived from Lie group theory that we will use in this paper. Readers who are familiar with differential geometry can skip this section. Additional details can be found in [Murray et al. 1994].

Given a moving body frame $\mathbf{T}(t) = (\mathbf{R}, \mathbf{p}) \in \mathrm{SE}(3)$, where $\mathbf{R} \in \mathrm{SO}(3)$ denotes rotation and $\mathbf{p} \in \mathbb{R}^3$ translation, its *generalized velocity* expressed in the instantaneous body frame (hence dubbed the *body velocity*) is defined as a *twist*

$$\hat{\mathbf{v}} = \mathbf{T}^{-1}\dot{\mathbf{T}} = \begin{bmatrix} [\omega] & \upsilon \\ 0 & 0 \end{bmatrix}, \tag{1}$$

which is an element of $\mathrm{se}(3)$, the Lie algebra of $\mathrm{SE}(3)$, where ω and υ are, respectively, the angular and linear velocities of $\mathbf{T}$ expressed in the body frame. The 3×3 skew-symmetric matrix of ω is denoted as $[\omega]$. We also represent the twist $\hat{\mathbf{v}}$ as a vector $\mathbf{v} = [\omega^T, \upsilon^T]^T$. The $\vee$ operator maps a twist to the corresponding twist coordinate; i.e., $\hat{\mathbf{v}}^\vee = \mathbf{v}$.

Given $\mathbf{T} \in \mathrm{SE}(3)$ and $\mathbf{g} = [\omega^T, \upsilon^T]^T \in \mathrm{se}(3)$, the adjoint mapping $\mathrm{Ad}_{\mathbf{T}} : \mathrm{se}(3) \mapsto \mathrm{se}(3)$ is defined as $\mathrm{Ad}_{\mathbf{T}}\ \hat{\mathbf{g}} = \mathbf{T}\hat{\mathbf{g}}\mathbf{T}^{-1}$, or in matrix form as

$$\mathrm{Ad}_{\mathbf{T}}\,\mathbf{g} = \begin{bmatrix} \mathbf{R} & 0 \\ [\mathbf{p}]\mathbf{R} & \mathbf{R} \end{bmatrix} \begin{bmatrix} \omega \\ \upsilon \end{bmatrix}. \tag{2}$$

The adjoint mapping is used in the coordinate transformation of twists. As we will see in Section 4, the body velocity of frame $\{i-1\}$ expressed in frame $\{i\}$ is written as

$${}^{i}\mathbf{v}_{i-1} = \mathrm{Ad}_{\mathbf{G}_i^{-1}}\mathbf{v}_{i-1}, \tag{3}$$

where $\mathbf{G}_i$ is the configuration of frame $\{i\}$ with respect to $\{i-1\}$.

Another useful operator is the Lie bracket $\mathrm{ad}_{\hat{\mathbf{g}}} : \mathrm{se}(3) \mapsto \mathrm{se}(3)$ and it occurs when (2) is differentiated. The Lie bracket is defined as $\mathrm{ad}_{\hat{\mathbf{g}}_1}\hat{\mathbf{g}}_2 = \hat{\mathbf{g}}_1\hat{\mathbf{g}}_2 - \hat{\mathbf{g}}_2\hat{\mathbf{g}}_1$, or

$$\mathrm{ad}_{\mathbf{g}_1}\mathbf{g}_2 = \begin{bmatrix} [\omega_1] & 0 \\ [\upsilon_1] & [\omega_1] \end{bmatrix} \begin{bmatrix} \omega_2 \\ \upsilon_2 \end{bmatrix}. \tag{4}$$

The generalized force $\mathbf{f} = [\mu^T, \eta^T]^T$ is an element of $\mathrm{se}^*(3)$, the dual space of $\mathrm{se}(3)$, where $\mu \in \mathbb{R}^3$ represents a moment and $\eta \in \mathbb{R}^3$ a force. In matrix form, the corresponding dual adjoint mappings $\mathrm{Ad}^*_{\mathbf{T}} : \mathrm{se}^*(3) \mapsto \mathrm{se}^*(3)$ and $\mathrm{ad}^*_{\mathbf{g}} : \mathrm{se}^*(3) \mapsto \mathrm{se}^*(3)$ are the transposes of $\mathrm{Ad}_{\mathbf{T}}$ and $\mathrm{ad}_{\mathbf{g}}$; i.e.,

$$\mathrm{Ad}^*_{\mathbf{T}} = \mathrm{Ad}^T_{\mathbf{T}}, \qquad \mathrm{ad}^*_{\mathbf{g}} = \mathrm{ad}^T_{\mathbf{g}}. \tag{5}$$

One can easily verify that $\mathrm{Ad}^{-1}_{\mathbf{T}}\mathbf{g} = \mathrm{Ad}_{\mathbf{T}^{-1}}\mathbf{g}$ and $\mathrm{ad}_{\mathbf{g}}\mathbf{g} = 0$.

For all $\mathbf{g} \in \mathrm{se}(3)$, $e^{\hat{\mathbf{g}}}$ is an element of $\mathrm{SE}(3)$. There exists a closed-form formula of the exponential map $\exp : \mathrm{se}(3) \mapsto \mathrm{SE}(3)$ [Murray et al. 1994]. Note that the derivative of the exponential map is not trivial; $e^{\hat{\mathbf{g}}(t)}\frac{d\hat{\mathbf{g}}(t)}{dt} \neq \frac{d}{dt}e^{\hat{\mathbf{g}}(t)} \neq \frac{d\hat{\mathbf{g}}(t)}{dt}e^{\hat{\mathbf{g}}(t)}$ in general. However, in the case where the rigid motion is due to a constant twist, its derivative takes the following simple form:

$$\frac{d}{dt}e^{\hat{\mathbf{s}}\rho(t)} = e^{\hat{\mathbf{s}}\rho(t)}\hat{\mathbf{s}}\dot{\rho}(t) = \hat{\mathbf{s}}e^{\hat{\mathbf{s}}\rho(t)}\dot{\rho}(t). \tag{6}$$

q; $\mathbf{q}$	Generalized coordinate; vector of generalized coords
$\breve{q}_i$	The i^{th} knot
$\dot{x}$	Differentiation of x with respect to time t
x'	Differentiation of x with respect to $\mathbf{q}$
$\mathbf{T}_i$	The configuration of frame $\{i\}$ w.r.t. the inertial frame
$\mathbf{G}_i$	The configuration of frame $\{i\}$ w.r.t. its parent frame
$\breve{\mathbf{G}}_i$	The i^{th} control frame
$\mathbf{z}_i$	The i^{th} control twist
$\mathbf{v}_i$	The body velocity of $\mathbf{T}_i$
$\mathbf{u}_i$	The body velocity of $\mathbf{G}_i$
$\mathbf{S}_i$	The joint Jacobian of frame $\{i\}$

Table 1: *Frequently used symbols.*

Finally, using the notations defined above, the Newton-Euler equations of the motion of a rigid body are expressed in a simple form as follows [Park et al. 1995]:

$$\mathbf{f} = \mathbf{J}\dot{\mathbf{v}} - \mathrm{ad}^*_{\mathbf{v}}\mathbf{J}\mathbf{v}, \tag{7}$$

where $\mathbf{f} \in \mathrm{se}^*(3)$ is the generalized force applied to the rigid body and $\mathbf{v} \in \mathrm{se}(3)$ is the generalized velocity. The *generalized inertia* $\mathbf{J} \in \mathbb{R}^{6\times 6}$ of the rigid body has the following structure:

$$\mathbf{J} = \begin{bmatrix} \mathscr{I} & m[\mathbf{r}] \\ m[\mathbf{r}]^T & m\mathbf{I} \end{bmatrix}, \tag{8}$$

where m is the mass, $\mathscr{I} \in \mathbb{R}^{3\times 3}$ is the rotational inertia matrix, $\mathbf{r} \in \mathbb{R}^3$ is the position of the center of mass, and $\mathbf{I}$ is the identity matrix. Eq. (7) is coordinate-invariant; i.e., it holds with respect to any coordinate frame.

Table 1 presents a list of symbols that we will use frequently.

4 Dynamics of General Scleronomic Joints

With the geometric tools introduced in Section 3, we will now derive the kinematics and dynamics equations for general scleronomic joints, including the lower pair joints.

Assuming that link i of a multibody system is connected to its parent link $i-1$ via a joint, the configuration $\mathbf{T}_i \in \mathrm{SE}(3)$ of the body frame $\{i\}$ of i with respect to the inertial reference frame is

$$\mathbf{T}_i = \mathbf{T}_{i-1}\mathbf{G}_i, \tag{9}$$

where $\mathbf{T}_{i-1}$ is the configuration of $\{i-1\}$, and $\mathbf{G}_i$ denotes the relative configuration of $\{i\}$ with respect to $\{i-1\}$, which we will call the *joint transformation*. It is determined by the action of the joint and, for scleronomic joints, it is determined entirely by the *joint coordinate* $\mathbf{q}_i \in \mathbb{R}^n$, where n is the number of DOFs of the joint; i.e., $\mathbf{G}_i = \mathbf{G}_i(\mathbf{q}_i)$. The *generalized joint velocity* generated by the action of the joint is

$$\hat{\mathbf{u}}_i = \mathbf{G}_i^{-1}\dot{\mathbf{G}}_i. \tag{10}$$

Substituting (9) into (1), we can express the body velocity $\mathbf{v}_i$ as the velocity of link $i-1$ plus that of the joint:

$$\mathbf{v}_i = {}^i\mathbf{v}_{i-1} + \mathbf{u}_i. \tag{11}$$

For scleronomic joints,

$$\mathbf{u}_i(\mathbf{q}_i) = \mathbf{S}_i(\mathbf{q}_i)\dot{\mathbf{q}}_i, \tag{12}$$

$$\text{where} \quad \mathbf{S}_i(\mathbf{q}_i) = \left(\mathbf{G}_i^{-1}\frac{d\mathbf{G}_i}{d\mathbf{q}_i}\right)^{\vee}. \tag{13}$$

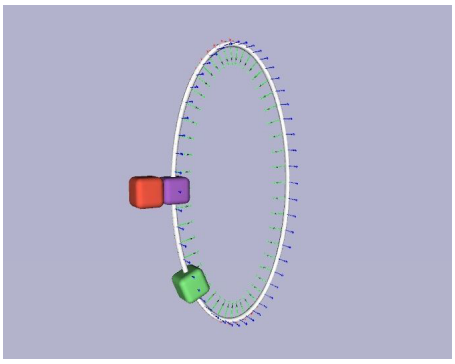


Figure 2: *An elliptic joint. The child link (green) is constrained to slide along the ellipse attached to the parent link (purple).*

The $6\times n$ *joint Jacobian* $\mathbf{S}_i$ is a mapping from the time derivatives of the joint coordinates $\dot{\mathbf{q}}_i$ to the generalized joint velocity $\mathbf{u}_i$. Using (13), the time derivative of $\mathbf{v}_i$ can be expressed as

$$\dot{\mathbf{v}}_i = {}^i\dot{\mathbf{v}}_{i-1} + \mathrm{ad}_{\mathbf{v}_i}\mathbf{S}_i\dot{\mathbf{q}}_i + \dot{\mathbf{q}}_i^T\nabla\mathbf{S}_i\dot{\mathbf{q}}_i + \mathbf{S}_i\ddot{\mathbf{q}}_i, \tag{14}$$

where $\nabla\mathbf{S}_i$ is the *joint Hessian*. Finally, from (7), the Newton-Euler equations of the motion of link i and the joint force or torque $\boldsymbol{\tau}$ are as follows:

$$\mathbf{f}_i = \mathbf{J}_i\dot{\mathbf{v}}_i - \mathrm{ad}^*_{\mathbf{v}_i}\mathbf{J}_i\mathbf{v}_i + {}^i\mathbf{f}_{i+1} - \mathbf{f}_{e,i}, \tag{15}$$

$$\boldsymbol{\tau}_i = \mathbf{S}_i^T\mathbf{f}_i, \tag{16}$$

where $\mathbf{f}_i \in \mathrm{se}^*(3)$ is the generalized force applied by link $i-1$ to link i and $\mathbf{f}_{e,i}$ is the external force (e.g., gravity) on link i. Note that ${}^i\mathbf{f}_{i+1} = \mathrm{Ad}^*_{\mathbf{G}_{i+1}^{-1}}\mathbf{f}_{i+1}$ expresses $\mathbf{f}_{i+1}$ relative to frame $\{i\}$.

Based on the above Lie group theoretic formulation of the kinematics and dynamics equations of general scleronomic joints, we derive in Appendix A an $\mathrm{O}(n)$ recursive forward dynamics algorithm for simulating such joints.

4.1 Creating New Joints

According to (9)–(16), the joint transformation as well as its Jacobian and Hessian are necessary for the kinematic analysis and dynamic simulation of the system. Hence, to create a new joint, it suffices to define a twice-differentiable joint transformation and its derivatives. Before defining new joints, consider the joint transformation, Jacobian, and Hessian of some simple joints.

The **helical joint** with pitch h has a joint transformation in the form of an exponential of a twist of a screw parameter $\mathbf{s} = [0,0,1,0,0,h]^T$ multiplied by a joint coordinate $q \in \mathbb{R}$; i.e., $\mathbf{G}(q) = \mathbf{G}(0)e^{\hat{\mathbf{s}}q}$. Using (13), we can derive the joint Jacobian and Hessian as $\mathbf{S} = \mathbf{s}$ and $\nabla\mathbf{S} = \mathbf{0}$. Thus, the joint Jacobian of the helical joint is constant and it is actually the screw parameter of the joint.

The **universal joint** can be modeled as the product of two revolute joints, $\mathbf{G} = \mathbf{G}(0)e^{\hat{\mathbf{s}}_0 q_0}e^{\hat{\mathbf{s}}_1 q_1}$. Here, the joint Jacobian is $\mathbf{S} = \left[\mathrm{Ad}^{-1}_{e^{\hat{\mathbf{s}}_1 q_1}}\mathbf{s}_0\ \mathbf{s}_1\right] := [\mathbf{s}_0^*\ \mathbf{s}_1]$, where $\mathbf{s}_0^*$ is the instantaneous screw parameter at given q_1, and an element of the Hessian is $d\mathbf{s}_0^*/dq_1 = \mathrm{ad}_{\mathbf{s}_0^*}\mathbf{s}_1$. Note that the joint Jacobian is not constant, but is a function of the joint coordinates, which is typically the case for multi-DOF joints.

Next, consider the creation of a non-conventional joint; for example, an **elliptic joint** (Fig. 2) that constrains the child link to slide along an ellipse with its orientation tangential to the ellipse. The

joint transformation $\mathbf{G}(q)$ can be written as follows:

$$\mathbf{G}(q) = \mathbf{H}\begin{bmatrix} e^{[\mathbf{k}]\phi(q)} & (a\sin q, -b\cos q, 0)^T \\ 1 & 0 \end{bmatrix}, \tag{17}$$

$$\phi(q) = \text{atan2}(b\sin q, a\cos q), \tag{18}$$

where a and b are the semi-axes of the ellipse, and $\mathbf{k}$ is a unit vector in $\mathbb{R}^3$. Once the joint transformation is defined, we must compute the Jacobian and its derivative to perform dynamic simulation. From (10) and (12),

$$\mathbf{S} = [\phi'\mathbf{k},\ e^{-[\hat{\mathbf{k}}]\phi}[a\cos q,\ b\sin q,\ 0]^T]^T, \tag{19}$$

$$\frac{d\mathbf{S}}{dq} = [\phi''\mathbf{k},\ e^{-[\hat{\mathbf{k}}]\phi}[(b\phi' - a)\sin q,\ (b - a\phi')\cos q,\ 0]^T]^T. \tag{20}$$

Since the desired joint motion is complex, defining such a joint transformation in closed form becomes difficult.

Ideally, it should be easy to compute the derivatives of the joint transformation. Since splines can effectively model complex curves and surfaces, it is natural to apply them in modeling complex joints.

5 Spline Joints

We now introduce the *spline joint*, a novel class of scleronomic joints whose joint transformation is modeled using splines. Any splines may be used for the joint transformation, as long as the following two requirements are satisfied:

1. The spline joint $\mathbf{G}(\mathbf{q})$ must be C^2-continuous in order to maintain the boundedness of the joint acceleration.
2. The joint Jacobian $\mathbf{S}(\mathbf{q})$ must not vanish anywhere in the domain, because the generalized joint velocity will vanish where $\mathbf{S}(\mathbf{q}) = \mathbf{0}$, even for non-zero joint coordinate velocity $\dot{\mathbf{q}}$.

Conceptually, we create the spline joint by defining the joint transformation matrix in SE(3) as a function of joint coordinates (an n-dimensional vector) using splines, and subsequently deriving its derivatives up to order two. However, since SE(3) is a curved space, it is by no means trivial to apply splines to the modeling of joint transformations.

We employ the spline curves of Kim et al. [1995] in the modeling of 1-DOF spline joints. Their spline curves in SO(3) can be straightforwardly extended to SE(3) in which our spline curve joints are defined. However, higher-dimensional splines (e.g., spline surfaces) on SO(3) are not yet known.[1] Hence, to create multi-DOF spline joints, we apply splines defined in Euclidean space to the twist coordinates.

5.1 Spline Curve Joint

The general interpolation scheme for SO(3) developed by Kim et al. [1995] achieves C^{k-2}-continuous rotation curves for k^{th}-order splines by introducing a cumulative form of the splines basis functions and the product of their exponentials. The merit of their method from the perspective of joint modeling is that the derivatives are easy to compute. The extension from the spline curve on SO(3) to SE(3) is straightforward and it retains all the important features such as local support and C^{k-2} continuity.

[1] Alexa [2002] defined an SE(3) curve as an exponential of a spline curve in Euclidean space. Similarly, a spline surface on SE(3) can be defined as an exponential of a spline surface in Euclidean space, but in general the derivatives of such a surface cannot be computed analytically.

While arbitrary C^2-continuous splines may be used for our purposes, consider B-splines of order $k > 3$ as a concrete example. Let $\mathbf{G}(q) \in \text{SE}(3)$ be a joint transformation parameterized by $q \in \mathbb{R}$. Using the cumulative B-spline basis, $\mathbf{G}(q)$ can be expressed as the product of exponentials of a constant *control twist* $\mathbf{z}$ multiplied by a spline basis function

$$\mathbf{G}(q) = \breve{\mathbf{G}}_0 \prod_{j=1}^{m} e^{\hat{\mathbf{z}}_j \tilde{B}_{j,k}(q)}, \tag{21}$$

where $\mathbf{z}_j = \log(\breve{\mathbf{G}}_{j-1}^{-1}\breve{\mathbf{G}}_j)$. The cumulative basis function is defined as the sum of B-spline basis functions: $\tilde{B}_{j,k}(q) = \sum_{\ell=j}^{m} B_{\ell,k}(q)$. The *control frame* $\breve{\mathbf{G}}_j$ corresponds to control point j in the regular B-spline function. Kim et al. [1995] show that the $\tilde{B}_{j,k}(q)$ are non-constant only for $\breve{q}_j < q < \breve{q}_{j+k-1}$:

$$\tilde{B}_{j,k}(q) = \begin{cases} 0 & \text{if } q \le \breve{q}_j \\ \sum_{\ell=j}^{j+k-2} B_{\ell,k}(q) & \text{if } \breve{q}_j < q < \breve{q}_{j+k-1} \\ 1 & \text{if } q \ge \breve{q}_{j+k-1} \end{cases} \tag{22}$$

and that their derivatives take the simple form

$$\frac{d}{dq}\tilde{B}_{j,k}(q) = \frac{k-1}{\breve{q}_{j+k-1} - \breve{q}_j} B_{j,k-1}(q). \tag{23}$$

From (22), $e^{\hat{\mathbf{z}}_1\tilde{B}_{1,k}(q)} = \breve{\mathbf{G}}_0^{-1}\breve{\mathbf{G}}_1$, ..., $e^{\hat{\mathbf{z}}_{j-k+1}\tilde{B}_{j-k+1,k}(q)} = \breve{\mathbf{G}}_{j-k}^{-1}\breve{\mathbf{G}}_{j-k+1}$, and $e^{\hat{\mathbf{z}}_{j+1}\tilde{B}_{j+1,k}(q)} = \cdots = e^{\hat{\mathbf{z}}_m\tilde{B}_{m,k}(q)} = \mathbf{I}$. This greatly reduces the number of multiplications and we can express the joint transformation as the product of only $k-1$ exponentials:

$$\mathbf{G}(q) = \breve{\mathbf{G}}_{j-k+1} \prod_{\ell=j-k+2}^{j} e^{\hat{\mathbf{z}}_\ell \tilde{B}_{\ell,k}(q)}. \tag{24}$$

We can also see that the joint transformation is locally defined by k control frames, $\breve{\mathbf{G}}_{j-k+1}, \ldots, \breve{\mathbf{G}}_j$. In the case of the cubic B-spline, the joint transformation can be expressed as the product of just three exponentials.

5.2 Multi-DOF Spline Joint

Unfortunately, the spline curve on SO(3) does not naturally extend to the spline surface, let alone the spline surface on SE(3). An alternative is to model the joint transformation as the product of exponentials of six *basis twists* $\hat{\mathbf{e}}_1, \ldots, \hat{\mathbf{e}}_6$:

$$\mathbf{G}(\mathbf{q}) = \mathbf{H} \prod_{j=1}^{6} e^{\hat{\mathbf{e}}_j \phi_j(\mathbf{q})}, \tag{25}$$

where $\phi_j(\mathbf{q}) : \mathbb{R}^n \mapsto \mathbb{R}$ is an n-DOF spline in Euclidean space. Even though (25) is similar in structure to (21), the two are quite different; in the spline curve joint, the spline is defined in the SE(3) world space, whereas in multi-DOF joints the spline is defined in each dimension of se(3).

It is generally advantageous to use the first three basis twists to represent the translation component of the joint transformation, while the last three represent the rotation. This is equivalent to representing the rotation using Euler angles, which suffers from singularities in certain configurations. Therefore, this approach cannot cover the whole of SO(3). However, in many cases we can choose suitable Euler angles to avoid singularities in the presence of joint limits.

5.3 Derivatives of the Spline Joint Transformation

For both the spline curve joint and the multi-DOF spline joint, the joint transformation $\mathbf{G}(q)$ takes the form of a product of exponentials. Having defined the joint transformation, we must then compute its Jacobian and Hessian for the purposes of dynamics simulation. An important feature of spline joints is that the twists are constant, which enables us to compute the analytic joint Jacobian and its derivatives efficiently as follows. The n-DOF spline joint ($n \geq 1$) has the following form:

$$\mathbf{G}(\mathbf{q}) = \mathbf{H} \prod_{j=1}^{m} e^{\hat{\mathbf{g}}_j \phi_j(\mathbf{q})}, \quad \mathbf{q} \in \mathbb{R}^n, \tag{26}$$

where $\phi_j \in \mathbb{R}$ is some function of the joint coordinates $\mathbf{q}$ and where $\hat{\mathbf{g}}_j \in \mathrm{se}(3)$, a constant, is a control twist for the spline curve joint or a basis twist for the n-DOF joint. Given the joint transformation, the joint Jacobian consists of n twists; i.e., $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_n]$, where

$$\mathbf{s}_i = \left(\mathbf{G}^{-1} \frac{\partial \mathbf{G}}{\partial q_i} \right)^{\vee} = \sum_{k=1}^{m} \mathrm{Ad}^{-1}_{e^{\hat{\mathbf{g}}_{k+1}\phi_{k+1}} \cdots e^{\hat{\mathbf{g}}_m \phi_m}} \mathbf{g}_k \frac{\partial \phi_k}{\partial q_i}. \tag{27}$$

Eq. (27) can be computed efficiently in a recursive manner:

$$\mathbf{s}_i = \mu^i_m, \tag{28}$$

where $\mu^i_0 = 0$ and

$$\mu^i_k = \mathbf{g}_k \frac{\partial \phi_k}{\partial q_i} + \mathrm{Ad}^{-1}_{e^{\hat{\mathbf{g}}_k \phi_k}} \mu^i_{k-1} \quad \text{for } 1 \leq k \leq m. \tag{29}$$

Its derivatives are also expressed recursively:

$$\frac{\partial \mathbf{s}_i}{\partial q_j} = \frac{\partial \mu^i_m}{\partial q_j}, \tag{30}$$

where

$$\frac{\partial \mu^i_k}{\partial q_j} = \mathbf{g}_k \frac{\partial^2 \phi_k}{\partial q_i \partial q_j} + \mathrm{Ad}^{-1}_{e^{\hat{\mathbf{g}}_k \phi_k}} \left(\frac{\partial \mu^i_{k-1}}{\partial q_j} + \mathrm{ad}_{\mu^i_{k-1}} \mathbf{g}_k \frac{\partial \phi_k}{\partial q_j} \right). \tag{31}$$

Appendix B gives pseudo code for computing the joint derivatives.

6 Designing Spline Curve Joints

A spline joint can be designed either by directly specifying the control frames or by providing a series of joint transformations for the spline joint to interpolate. For example, to model a biological joint as a spline joint, one can measure the relative transformations of the two bones in sample configurations and model a spline joint that optimally interpolates between the estimated transformations. We will present a geometric data fitting algorithm to determine the control frames in this scenario. We will focus on the 1-DOF spline curve joint. Data fitting for the multi-DOF spline joint is left for future work, as will be discussed in Section 8.

6.1 Natural Parameterization

Given a set of control frames, we want to construct the spline joint by assigning knot values $\breve{q}_0, \ldots, \breve{q}_m$ to each control frame. While we can assign arbitrary ascending numbers to the knot values, a more intuitive choice would be setting the knot values in such a way that the distance between two frames equals the distance in the joint coordinate space; i.e., $||\delta \mathbf{G}|| = ||\delta q||$ or, equivalently, $||\mathbf{s}(q)|| = 1$. Note that since a twist contains both linear and angular motions, one should define a suitable metric for a particular application. While it is difficult to ensure $||\mathbf{s}(q)|| = 1$ for all q, it is easy to get approximate results by setting $\breve{q}_j = \breve{q}_{j-1} + ||\log(\mathbf{G}^{-1}_{j-1}\mathbf{G}_j)||$.[2]

[2]Here we employ the "unweighted" metric, but any reasonable metric can be used. For example, Kaufman et al. [2005] used the inertia-weighted metric for dynamics simulation. The unweighted metric is in some sense a geometrically reasonable one and it may give a better sense of distance to users, while the inertia-weighted metric may be better for modeling controllers.

6.2 Data Fitting Algorithm

Given a series of desired joint transformations $\mathbf{G}^k_d$ at q^k for $k = 1, \ldots, N$, the goal of the data fitting process is to compute optimal control frames $\breve{\mathbf{G}}_0, \ldots, \breve{\mathbf{G}}_m$ such that

$$\underset{\breve{\mathbf{G}}_0 \ldots \breve{\mathbf{G}}_m}{\operatorname{argmin}} \sum_{k=1}^{N} d(\mathbf{G}^k_d, \mathbf{G}(q^k)), \tag{32}$$

where $d(\cdot,\cdot)$ is a metric on SE(3). We can solve this nonlinear optimization problem by iteratively updating control frames such that the value of the objective function decreases. Imagine that the control frames $\breve{\mathbf{G}}_j$ (and spline) are moving in space and that at each iteration we update $\breve{\mathbf{G}}_j$ using the body velocity $\breve{\mathbf{u}}_j = \breve{\mathbf{G}}_j^{-1} \dot{\breve{\mathbf{G}}}_j$, so that $\mathbf{G}(q^k)$ approaches $\mathbf{G}^k_d$. To this end, we need the mapping from the velocities of the control frames to the velocities of the joint transformations at q^k.

For simplicity, let us consider the cubic B-spline case: $\mathbf{G}(q) = \breve{\mathbf{G}}_{j-3} e^{\gamma} e^{\beta} e^{\alpha}$, where $\gamma = \tilde{B}_{j-2}\hat{\mathbf{z}}_{j-2}$, $\beta = \tilde{B}_{j-1}\hat{\mathbf{z}}_{j-1}$, and $\alpha = \tilde{B}_j \hat{\mathbf{z}}_j$ for $\breve{q}_j \leq q < \breve{q}_{j+1}$. From the relation $\dot{\mathbf{z}}_j = \breve{\mathbf{u}}_j - \mathrm{Ad}^{-1}_{e^{\mathbf{z}_j}} \breve{\mathbf{u}}_{j-1}$, we make the following approximation:

$$\left(e^{-\alpha} \frac{d}{dt} e^{\alpha} \right)^{\vee} \approx \tilde{B}_j \left(\breve{\mathbf{u}}_j - \mathrm{Ad}^{-1}_{e^{\alpha}} \breve{\mathbf{u}}_{j-1} \right). \tag{33}$$

Then, the velocity of the joint transformation at q can be expressed neatly as the sum of the velocities of control frames weighted by the (regular) B-spline basis functions:

$$\begin{aligned} \mathbf{u}(q) \approx\ & B_{j-3} \mathrm{Ad}^{-1}_{e^{\gamma} e^{\beta} e^{\alpha}} \breve{\mathbf{u}}_{j-3} + B_{j-2} \mathrm{Ad}^{-1}_{e^{\beta} e^{\alpha}} \breve{\mathbf{u}}_{j-2} \\ & + B_{j-1} \mathrm{Ad}^{-1}_{e^{\alpha}} \breve{\mathbf{u}}_{j-1} + B_j \breve{\mathbf{u}}_j. \end{aligned} \tag{34}$$

Using (34), we construct a matrix that relates the $\breve{\mathbf{u}}_j$ to the $\mathbf{u}(q^k)$:

$$\mathbf{M} \breve{\mathbf{u}}_c = \mathbf{u}_d, \tag{35}$$

where $\mathbf{M}$ is a banded matrix, $\breve{\mathbf{u}}_c = (\breve{\mathbf{u}}_0^T, \ldots, \breve{\mathbf{u}}_m^T)^T \in \mathbb{R}^{6(m+1)}$, and the desired velocities $\mathbf{u}_d = (\mathbf{u}(q^1)^T, \ldots, \mathbf{u}(q^N)^T)^T \in \mathbb{R}^{6N}$.

For $\mathbf{G}(q^k)$ to approach $\mathbf{G}^k_d$, we construct $\mathbf{M}$, specify $\mathbf{u}_d$ by setting

$$\mathbf{u}(q^k) = \log(\mathbf{G}(q^k)^{-1} \mathbf{G}^k_d), \tag{36}$$

solve the linear system (35) for $\breve{\mathbf{u}}_c$, and update $\breve{\mathbf{G}}_j \leftarrow \breve{\mathbf{G}}_j e^{\breve{\mathbf{u}}_j}$, for $j = 0, \ldots, m$, thus decreasing the value of the objective function in (32). We iterate these steps until convergence in order to solve the optimization problem.

6.3 Smoothing Algorithm

When the desired joint transformations are noisy, the resulting spline joint can produce unnatural motion. In this case, we need to smooth the spline curve in order to improve the quality of motion. An approach defining the smoothness of the spline joint is through the joint Hessian: If it is zero, the joint axis is fixed and produces simple motion like that of the lower pair joints. As in the data fitting algorithm, we compute the relationship between the rate of change of the joint Hessian at each point and the velocities of the

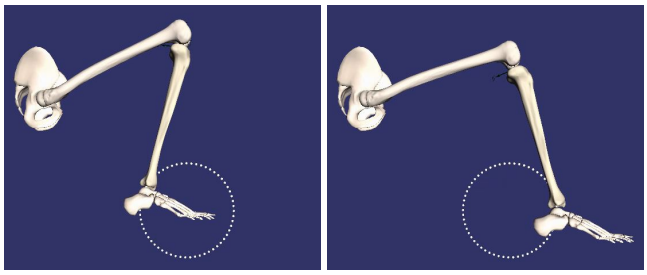


Figure 3: *Knee spline joint inverse kinematics example.*

control frames, and then we update the control frames in such a way that the joint Hessians approach zero.

We give a slightly different approximation to (33) for simplicity:[3]

$$\left(e^{-\alpha}\frac{d}{dt}e^{\alpha}\right)^{\vee} \approx \dot{\alpha} = \tilde{B}_j\left(\breve{\mathbf{u}}_j - \mathrm{Ad}^{-1}_{e^{\mathbf{z}_i}}\breve{\mathbf{u}}_{j-1}\right). \tag{37}$$

Using (37), we can write the time derivative of the joint Hessian (a vector) as

$$\begin{aligned}\dot{\mathbf{h}} \approx &\left(-\mathbf{N}'_{j-2}\mathrm{Ad}^{-1}_{e^{\hat{\mathbf{z}}_{j-2}}}\right)\breve{\mathbf{u}}_{j-3} + \left(\mathbf{N}'_{j-2} - \mathbf{N}'_{j-1}\mathrm{Ad}^{-1}_{e^{\hat{\mathbf{z}}_{j-1}}}\right)\breve{\mathbf{u}}_{j-2}\\ &+ \left(\mathbf{N}'_{j-1} - \mathbf{N}'_{j}\mathrm{Ad}^{-1}_{e^{\hat{\mathbf{z}}_{j}}}\right)\breve{\mathbf{u}}_{j-1} + \mathbf{N}'_{j}\breve{\mathbf{u}}_{j},\end{aligned} \tag{38}$$

where

$$\mathbf{N}_{j-2} = \tilde{B}'_{j-2}\mathrm{Ad}^{-1}_{e^{\beta}e^{\alpha}}, \tag{39}$$

$$\mathbf{N}_{j-1} = \tilde{B}'_{j-1}\mathrm{Ad}^{-1}_{e^{\alpha}} + \tilde{B}_{j-1}\mathrm{Ad}^{-1}_{e^{\beta}e^{\alpha}}\mathrm{ad}_{\gamma'}, \tag{40}$$

$$\mathbf{N}_{j} = \tilde{B}'_{j}\mathbf{I} + \tilde{B}_{j}\mathrm{Ad}^{-1}_{e^{\alpha}}\mathrm{ad}_{(\beta' + \mathrm{Ad}^{-1}_{e^{\beta}}\gamma')}. \tag{41}$$

One can derive the analytic derivatives of the $\mathbf{N}_j$, but numerical differentiation with respect to q is also simple and effective.

As in the data fitting case, we construct a matrix that transforms the velocities of the control frame to the rate of change of the joint Hessian:

$$\mathbf{N}\breve{\mathbf{u}}_c = \dot{\mathbf{h}}_d. \tag{42}$$

We can incorporate the smoothness criterion into the data fitting process by minimizing $(1-w)||\mathbf{u}_d - \mathbf{M}\breve{\mathbf{u}}_c||^2 + w||\dot{\mathbf{h}}_d - \mathbf{N}\breve{\mathbf{u}}_c||^2$, which can be accomplished by solving for $\breve{\mathbf{u}}_c$ in

$$\left((1-w)\mathbf{M}^T\mathbf{M} + w\mathbf{N}^T\mathbf{N}\right)\breve{\mathbf{u}}_c = (1-w)\mathbf{M}^T\mathbf{u}_d + w\mathbf{N}^T\dot{\mathbf{h}}_d,$$

where w is a weight and $\dot{\mathbf{h}}_d$ can simply be set to $-c\mathbf{h}$ for $0 < c \leq 1$.

The smoothing algorithm can also be used to smooth the spline curve while interpolating desired joint transformations by providing more control frames than are necessary for interpolation; i.e., since the number of control frames exceeds the minimum number required for interpolation, the extra control frames provide additional degrees of freedom to achieve smoothness.

7 Examples and Results

Fig. 1 shows the femorotibial joint modeled as a spline curve joint. We created the desired joint transformations by posing the tibia by eye in a commercial modeling package, then applying the data fitting and smoothing algorithms to generate the control frames. As the accompanying video shows, data-fitting without smoothing creates a somewhat unnatural motion. We achieve a more natural motion after applying the smoothing algorithm.

[3] Note that $e^{-\alpha}\frac{de^{\alpha}}{dt} = \dot{\alpha} + \frac{1}{2!}[\,[\,\dot{\alpha},\alpha\,],\alpha\,] + \frac{1}{3!}[\,[\,[\,\dot{\alpha},\alpha\,],\alpha\,],\alpha\,]\cdots$, where the Lie bracket $[\,\alpha,\dot{\alpha}\,] := \alpha\dot{\alpha} - \dot{\alpha}\alpha$. Therefore, the approximation approaches the exact solution when $\dot{\alpha}$ is either small or parallel to α.

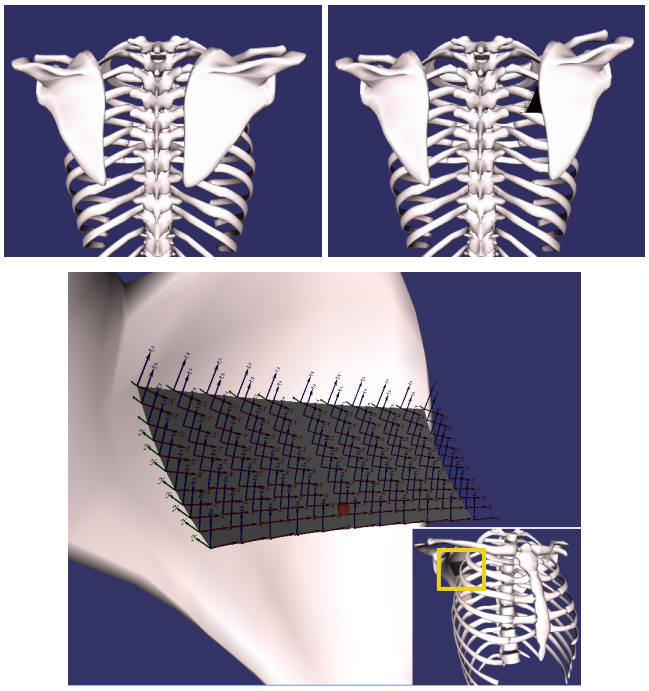
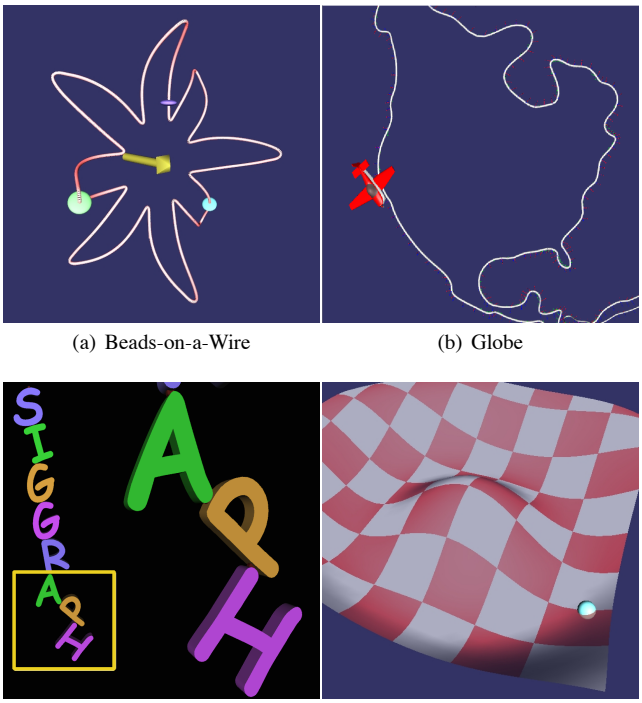


Figure 4: *Sample poses (top) and closeup internal view (bottom) of the scapulothoracic joint, modeled as a spline surface joint.*

Fig. 3 demonstrates that inverse kinematics algorithms can be applied to the spline joint. Given a target position and orientation of the foot, an iterative inverse kinematics algorithm uses the analytic Jacobian to compute the joint coordinates of the leg required to reach the target configuration. The hip and the ankle are modeled as revolute joints.

The scapulothoracic joint in the shoulder is a notorious joint to model in biomechanics. Fig. 4 demonstrates that it can be modeled as a spline surface joint. The scapula is connected to the rib cage via a 2-DOF spline surface joint and to the clavicle via damped springs. Our approach is contrary to conventional methods in which the clavicle and the scapula form a kinematic hierarchy via a ball joint and auxiliary constraints enforce the contact between the scapula and the thorax. Since we model the spline surface such that the pose of the scapula satisfies the constraint between the scapula and the clavicle (i.e., the distance between the acromion process and the sternoclavicular joint is approximately the same as the length of the clavicle), it is easy to enforce the constraint between the scapula and the clavicle using springs. The gray surface between the scapula and the rib cage in Fig. 4 is the surface swept by the reference frame of the scapula. The orientations of the frame at sample positions are drawn on the surface. The red box on the surface represents the current position and orientation.

Fig. 5 illustrates some of our experiments in creating interesting mechanisms using spline joints. In Fig. 5(a), each of the three beads is connected to the parent link through a flower-shaped spline joint. In this example, the beads slide along the spinning wire frame in a physically realistic manner in gravity (we can easily enable them to



(a) Beads-on-a-Wire (b) Globe

(c) SIGGRAPH Joint (d) Deforming Spline Surface Joint

Figure 5: *Example mechanisms modeled with spline joints.*

spin freely about the curve by inserting a simple rotational joint between the bead and the spline joint). Similarly, Fig. 5(b) illustrates a mechanism in which toy airplanes slide along the continental coastlines using spline joints. Note that for the "Globe" example, each control frame is computed such that two of its axes are in the tangent plane of the globe and one of those two axes is also tangent to the coastal curve. The "Beads-on-a-Wire" example is created in a similar manner. These examples demonstrate that we can create spline joints of arbitrary shape.

Fig. 5(c) is a snapshot from the "SIGGRAPH Joint 1" demo in the accompanying video. In this example, we use spline joints to constrain each letter to the surface of the parent letter. The chain of letters, which is constrained at the top of the letter S, free-falls in gravity. As we modeled only a single spline joint between a pair of letters, the contact point of one of the two links is constant while that of the other is changing. In the second "SIGGRAPH Joint 2" example in the accompanying video, we allow sliding on both surfaces by modeling two spline curve joints between the letters. The first curve is created from the tangent frames while the second is created from the inverse of the original control frames.

Fig. 5(d) is an example of a "Deforming Spline Surface Joint". The cyan sphere is constrained to slide along a 2-DOF spline surface joint in a physically realistic manner, subject to gravity. At the same time, the spline surface joint deforms kinematically.

We performed our experiments on a 2.6 GHz Intel Core 2 CPU system. The spline joint permits large numerical time steps using the forward Euler time integration method. In our experiments, the maximum time step ranges from 20 msec ("Globe") to 100 msec ("Beads-on-a-Wire"). Also, the experiments confirm that the spline joint does not raise the complexity of the dynamics formulation beyond $O(n)$. With our unoptimized implementation, the compute time per time step is 6 msec for the 1-DOF system "Knee" and 49 msec for the 8-DOF system "SIGGRAPH Joint 1". Since we use local-support basis splines, the number of control points does not adversely affect the computational complexity.

8 Conclusion and Future Work

We introduced spline joints to model general scleronomic joints for multibody dynamics based on the minimal-coordinates formulation. Spline joints enable the accurate modeling of biological joints, as well as the creation of interestingly intricate mechanisms for computer graphics. Spline joints can easily be incorporated into existing dynamics algorithms. Our technique opens up a range of possibilities in modeling dynamic structures and it nicely avoids the slower maximal-coordinates-based approach that, anyway, would not simplify the representation of the scleronomic constraint.

Many complex biological joints of animals (including humans) can be more accurately modeled using our technique. For example, we demonstrated that the femorotibial joint and the scapulothoracic joint can be modeled more accurately using spline joints. We modeled these knee and shoulder examples by eye; more accurate modeling should be possible using medical imaging techniques. In the case of human joints other than the knee and scapula, the condyloid joint types (e.g., the wrist) and saddle joint types (e.g., the thumb) should benefit most from our technique.

Since we use twice differentiable splines in spline joints, our technique can best be used for modeling "smooth" joints. However, we believe that sharp corners can be reasonably approximated in most cases by smooth splines using multiple control frames near the discontinuities. If one wants to model sharp corners or cusps using non-C^2-continuous splines, it would be necessary to compute and apply an appropriate impulse at the discontinuities.

As demonstrated in the "SIGGRAPH Joint 2" example, surface-to-surface contact can be simulated by multiplying one spline surface joint with a second "inverted" spline surface joint. Moreover, it is possible to enforce additional constraints, such as a no-slip constraint, by computing constraint forces using Lagrange multipliers.

We did not consider the data fitting problem for multi-DOF joints. A naive approach would be to apply least-squares fitting to each of 6 splines because, unlike the 1-DOF case, each spline is defined in Euclidean space. Presumably this coordinate-wise fitting will give reasonable results; however, it may not be optimal in terms of the given metric on SE(3). A good problem for future work would be to develop an optimal data fitting algorithm for multi-DOF spline joints that respects the metric on SE(3).

In the deforming spline surface joint example (Fig. 5(d)), we deformed the surface spline kinematically and the surface itself did not participate in the dynamic simulation. In future work, however, there is no reason why we cannot employ dynamic NURBS [Terzopoulos and Qin 1994] to create an physically accurate D-NURBS surface joint which would deform elastically under the weight of the ball or any other applied forces.

A Recursive Forward Dynamics Algorithm

For use with our spline joints, we derive a Lie Group theoretic recursive dynamics formulation that extends the one in [Park et al. 1995]. Note that the algorithm is essentially the same as the articulated body method for multiple DOF joints developed in [Featherstone 1987]. Our recursive forward dynamics algorithm is $O(n)$ for tree-type articulated structures.

The articulated inertia $\tilde{\mathbf{J}}_i$ and its associated bias force $\mathbf{b}_i$ are defined

such that they satisfy the following relation:

$$\mathbf{f}_i = \tilde{\mathbf{J}}_i \dot{\mathbf{v}}_i + \mathbf{b}_i. \tag{43}$$

Substituting (14) into (43) and pre-multiplying (43) with $\mathbf{S}_i^T$, we can decompose $\ddot{\mathbf{q}}_i$ into two parts, one induced by τ_i and the other induced by neighboring joints, as follows:

$$\ddot{\mathbf{q}}_i = \Omega_i^{-1}\left(\tau_i - \mathbf{S}_i^T \tilde{\mathbf{J}}_i ({}^i\dot{\mathbf{v}}_{i-1} + \mathbf{c}_i) - \mathbf{S}_i^T \mathbf{b}_i\right), \tag{44}$$

where $\mathbf{c}_i = \dot{\mathbf{q}}_i^T \nabla \mathbf{S}_i \dot{\mathbf{q}}_i + \mathrm{ad}_{\mathbf{v}_i}\mathbf{u}_i$ and $\Omega_i = \mathbf{S}_i^T \tilde{\mathbf{J}}_i \mathbf{S}_i$. We want to derive recursive equations for $\tilde{\mathbf{J}}_i$ and $\mathbf{b}_i$. This is accomplished by replacing $\mathbf{f}_{i+1}$ in (15) with (43) and substituting (14) for $\dot{\mathbf{v}}_{i+1}$. Hence, we derive the recursive forward dynamics algorithm for general scleronomic joints as follows:

- Given τ_i and $\mathbf{f}_{e,i}$
- Update $\mathbf{G}_i, \mathbf{S}_i, \nabla\mathbf{S}_i, \mathbf{v}_i$ and $\mathbf{c}_i$
- Backward recursion:

$$\begin{aligned}
\tilde{\mathbf{J}}_i &= \mathbf{J}_i + \mathrm{Ad}^*_{\mathbf{G}_{i+1}^{-1}} \tilde{\mathbf{J}}_{i+1} \mathrm{Ad}_{\mathbf{G}_{i+1}^{-1}} \\
&\quad - \mathrm{Ad}^*_{\mathbf{G}_{i+1}^{-1}} \tilde{\mathbf{J}}_{i+1} \mathbf{S}_{i+1} \Omega_{i+1}^{-1} \mathbf{S}_{i+1}^T \tilde{\mathbf{J}}_{i+1} \mathrm{Ad}_{\mathbf{G}_{i+1}^{-1}} \\
\mathbf{b}_i &= -\mathrm{ad}^*_{\mathbf{v}_i} \mathbf{J}_i \mathbf{v}_i - \mathbf{f}_{e,i} + \mathrm{Ad}^*_{\mathbf{G}_{i+1}^{-1}} (\tilde{\mathbf{J}}_{i+1} \mathbf{c}_{i+1} + \mathbf{b}_{i+1}) \\
&\quad + \mathrm{Ad}^*_{\mathbf{G}_{i+1}^{-1}} \tilde{\mathbf{J}}_{i+1} \mathbf{S}_{i+1} \Omega_{i+1}^{-1} \left(\tau_{i+1} - \mathbf{S}_{i+1}^T (\tilde{\mathbf{J}}_{i+1} \mathbf{c}_{i+1} + \mathbf{b}_{i+1})\right) \\
\Omega_i &= \mathbf{S}_i^T \tilde{\mathbf{J}}_i \mathbf{S}_i
\end{aligned}$$

- Forward recursion:

$$\begin{aligned}
\ddot{\mathbf{q}}_i &= \Omega_i^{-1}\left(\tau_i - \mathbf{S}_i^T \tilde{\mathbf{J}}_i ({}^i\dot{\mathbf{v}}_{i-1} + \mathbf{c}_i) - \mathbf{S}_i^T \mathbf{b}_i\right) \\
\dot{\mathbf{v}}_i &= {}^i\dot{\mathbf{v}}_{i-1} + \mathbf{c}_i + \mathbf{S}_i \ddot{\mathbf{q}}_i
\end{aligned}$$

B Spline Joint Jacobian and Hessian

Elements of the joint Jacobian $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_n]$ and the Hessian are computed using the following algorithm:

Require: q

1: $\mathbf{s}_i = \mathbf{g}_1 \frac{\partial \phi_1}{\partial q_i}, \frac{\partial \mathbf{s}_i}{\partial q_j} = \mathbf{g}_1 \frac{\partial^2 \phi_1}{\partial q_i \partial q_j}$ for $i, j = 1, \ldots, n$
2: **for** $k = 2$ to m **do**
3: $\quad \frac{\partial \mathbf{s}_i}{\partial q_j} = \mathbf{g}_k \frac{\partial^2 \phi_k}{\partial q_i \partial q_j} + \mathrm{Ad}^{-1}_{e^{\mathbf{g}_k \phi_k}} \left(\frac{\partial \mathbf{s}_i}{\partial q_j} + \mathrm{ad}_{\mathbf{s}_i} \mathbf{g}_k \frac{\partial \phi_k}{\partial q_j}\right)$
4: $\quad \mathbf{s}_i = \mathbf{g}_k \frac{\partial \phi_k}{\partial q_i} + \mathrm{Ad}^{-1}_{e^{\mathbf{g}_k \phi_k}} \mathbf{s}_i$

Refer to Section 5.3 for the definitions of the symbols. The complexity of this algorithm is $O(n^2 m)$, where n is the number of DOFs of the joint and m is the number of exponentials. In most cases, m is small (e.g., $m = 3$ for the 1-DOF cubic spline joint) and $n \leq 2$, so this algorithm is efficient.

Acknowledgements

SHL was supported in part by a UCLA Dissertation Year Fellowship. The knee mesh data was created by Marco Viceconti and is available from www.tecno.ior.it/VRLAB. We acknowledge the helpful comments of the anonymous referees.

References

ALEXA, M. 2002. Linear combination of transformations. *ACM Transactions on Graphics 21*, 3 (July), 380–387.

BARR, A. H., CURRIN, B., GABRIEL, S., AND HUGHES, J. F. 1992. Smooth interpolation of orientations with angular velocity constraints using quaternions. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, 313–320.

DELP, S. L., LOAN, J. P., HOY, M. G., ZAJAC, F. E., TOPP, E. L., AND ROSEN, J. M. 1990. An interactive graphics-based model of the lower extremity to study orthopaedic surgical procedures. *IEEE Transactions on Biomedical Engineering 37*, 8 (Aug.), 757–767.

FEATHERSTONE, R. 1987. *Robot Dynamics Algorithms*. Kluwer Adademic Publishers, Boston.

GABRIEL, S., AND KAJIYA, J. 1985. Spline interpolation in curved space. *In SIGGRAPH 85 Course Notes for "State of the Art in Image Synthesis"*.

KAPANDJI, I. 1974. *The Physiology of the Joints*. Churchill Livingstone, Edinburgh.

KAUFMAN, D. M., EDMUNDS, T., AND PAI, D. K. 2005. Fast frictional dynamics for rigid bodies. *ACM Transactions on Graphics 24*, 3 (Aug.), 946–956.

KIM, M.-J., SHIN, S. Y., AND KIM, M.-S. 1995. A general construction scheme for unit quaternion curves with simple high order derivatives. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, 369–376.

KRY, P. G., AND PAI, D. K. 2003. Continuous contact simulation for smooth surfaces. In *ACM Transactions on Graphics*, vol. 22, 106–129.

LEE, S.-H., AND TERZOPOULOS, D. 2006. Heads up! Biomechanical modeling and neuromuscular control of the neck. *ACM Transactions on Graphics 25*, 3 (July), 1188–1198.

MACIEL, A., NEDEL, L. P., AND FREITAS, C. M. D. S. 2002. Anatomy-based joint models for virtual human skeletons. *Proceedings of the Computer Animation 2002 Conference*, 220–224.

MURRAY, R., LI, Z., AND SASTRY, S. 1994. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, New York.

PARK, F. C., AND RAVANI, B. 1997. Smooth invariant interpolation of rotations. *ACM Transactions on Graphics 16*, 3 (July), 277–295.

PARK, F. C., BOBROW, J. E., AND PLOEN, S. R. 1995. A lie group formulation of robot dynamics. *The International Journal of Robotics Research 14*, 6, 609–618.

RAMAMOORTHI, R., AND BARR, A. H. 1997. Fast construction of accurate quaternion splines. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, 287–292.

REULEAUX, F. 1876. *Kinematics of Machinery: Outlines of a Theory of Machines*. MacMillan and Co, London.

SHAO, W., AND NG-THOW-HING, V. 2003. A general joint component framework for realistic articulation in human characters. In *Proceedings of the 2003 ACM Symposium on Interactive 3D Graphics*, 11–18.

SHOEMAKE, K. 1985. Animating rotation with quaternion curves. In *Computer Graphics (Proceedings of SIGGRAPH 85)*, 245–254.

TÄNDL, M., AND KECSKEMÉTHY, A. 2007. A comparison of B-spline curves and Pythagorean hodograph curves for multibody dynamics simulation. *Proceedings of Twelfth World Congress in Mechanism and Machine Science* (June), 380–387.

TERZOPOULOS, D., AND QIN, H. 1994. Dynamic NURBS with geometric constraints for interactive sculpting. *ACM Transactions on Graphics 13*, 2 (Apr.), 103–136.