Splines in the Space of Shells

Behrend Heeren[1], Martin Rumpf[1], Peter Schröder[2], Max Wardetzky[3] and Benedikt Wirth[4,5]

[1]Institute for Numerical Simulation, University of Bonn, Germany
[2]Caltech, USA
[3]Institute of Numerical and Applied Math, University of Göttingen, Germany
[4]Institute for Computational and Applied Mathematics, University of Münster, Germany
[5]Cells-in-Motion Cluster of Excellence (EXC1003 – CiM), University of Münster, Germany

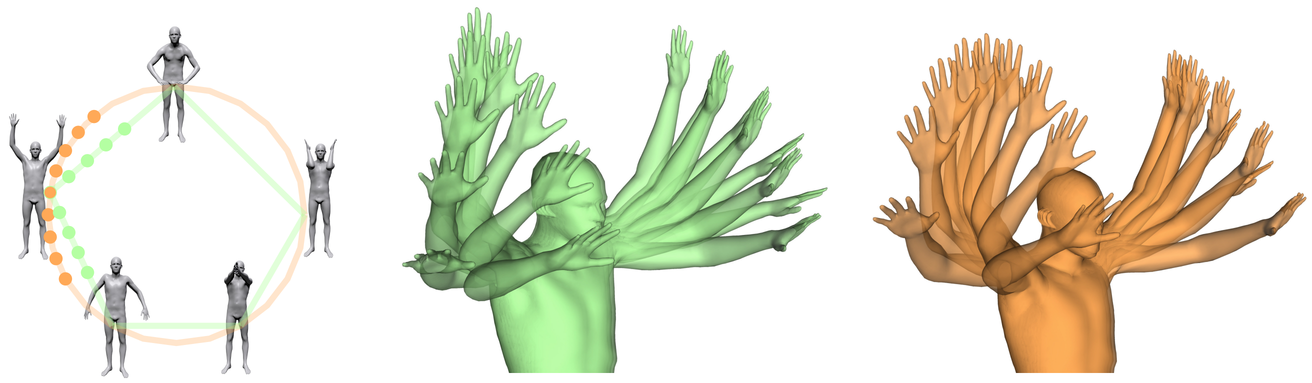

Figure 1: *A (periodic) spline in shell space (orange) allows for a temporally smooth interpolation of a given set of shell keyframe poses (gray, left). Compared to a piecewise geodesic interpolation (green), our spline interpolation yields a more natural motion and a balanced distribution of acceleration along the curve. We only display a few shells on the spline path around one of the keyframe poses as indicated on the left.*

Abstract

Cubic splines in Euclidean space minimize the mean squared acceleration among all curves interpolating a given set of data points. We extend this observation to the Riemannian manifold of discrete shells in which the associated metric measures both bending and membrane distortion. Our generalization replaces the acceleration with the covariant derivative of the velocity. We introduce an effective time-discretization for this novel paradigm for navigating shell space. Further transferring this concept to the space of triangular surface descriptors—edge lengths, dihedral angles, and triangle areas—results in a simplified interpolation method with high computational efficiency.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

We introduce the concept of navigating in the space of shell surfaces, briefly denoted as *shell space*, via interpolating splines. Using splines for smoothly interpolating keyframe poses of animated characters in Euclidean space goes back at least to the pioneering works of Kochanek and Bartels [KB84] and Lasseter [Las87] in the 1980s. Realizing shape interpolation through *curves in the space of shells* is relatively recent. Our point of departure is a combination of two observations: (i) the classical concept of splines can be readily extended from flat Euclidean domains to curved Riemannian manifolds (see Section 2 and the sketch in Fig. 2) and (ii) shell space can be regarded as a Riemannian manifold as first suggested by Kilian et al. [KMP07]. Specifically, we work with the Riemannian metric introduced in [HRWW12, HRS*14] that minimizes viscous dissipation when moving from one shape to another (Section 3).

Our model is based on a variational perspective. In Euclidean

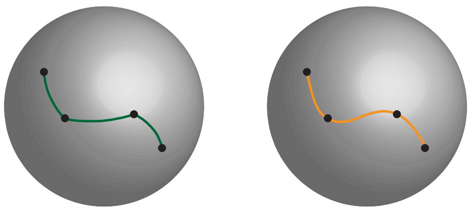

Figure 2: *Piecewise geodesic (left) vs. spline (right) interpolation on Riemannian manifolds (sphere for illustration). Notice C^1-discontinuities for geodesic interpolation. Since shell space is a Riemannian manifold, splines offer smooth keyframe interpolation.*

space, natural cubic splines minimize the total squared second derivative $\int_0^1 \|\ddot{s}\|^2 \,\mathrm{d}t$ among all curves $(s(t))_{t\in[0,1]}$ that pass through a given set of interpolation points. This property is related to the minimization of bending energy. We build on this observation and establish how splines can be variationally defined on shell space where keyframe poses act as interpolation points (see Section 4). We assume a fixed correspondence between consecutive keyframe shapes, given by fixed connectivity in the discrete case. This assumption is natural for the interpolation problem between different poses of the same model. Our main contribution is the translation of the time-continuous variational perspective of splines to a variational formulation of computationally tractable *time-discrete* splines (Section 5).

Using splines in shell space comes at a price, though. The Riemannian metric on shell space is closely related to elastically *deforming* shells and encapsulates both bending and stretching contributions. For triangle meshes the bending contributions depend on dihedral angles, which are nonlinear functions of vertex positions. Consequently splines in shell space lead to highly nonlinear optimization problems (Section 6), whose computational complexity must be reduced to yield practical algorithms.

To address this challenge we switch from vertex positions as primary variables to the *$L\Theta A$-representation* in the spirit of [WDAH10] and [FB11] (Section 7). In this formulation, the primary variables are the edge lengths of triangles (L), dihedral angles between adjacent triangles (Θ), and triangle areas (A). Splines in $L\Theta A$-space can be evaluated through solving a simple linear system. However, the resulting curve in $L\Theta A$-space will in general not be realizable as a sequence of triangle meshes in 3D Euclidean space. Instead we project to the closest curve of realizable triangle meshes in a least squares sense. Overall, this formulation greatly outperforms nonlinear optimization for computing splines based on vertex positions. We support these observations by various numerical experiments (see Section 8).

2. Related work

The idea of using *curves* for navigating shell space (e.g., for shape interpolation) is recent. Kilian et al. [KMP07] introduced geodesics in shell space for interpolating between two given poses using a metric that is derived from in-plane membrane deformations of surfaces. This approach was extended in [HRWW12, HRS*14] to a metric that incorporates full elastic responses including bending contributions and was used for geodesic interpolation and extrapolation as well as for parallel transport. We build on their Riemannian metric. Using this metric, Brandt et al. [BvTH16] suggested a scheme that largely accelerates geodesic computation in shell space by resorting to dimensionality reduction in the spatial domain. In a different development, Winkler et al. [WDAH10] and Fröhlich and Botsch [FB11] used edge length and dihedral angle coordinates to represent meshes for efficiently computing interpolation paths between two given poses. We build on these ideas for our $L\Theta A$-approximation.

In a different development, *spacetime constraints* were introduced early on in graphics by asking for a trajectory that minimizes the work required to satisfy user specified keyframes [WK88]. When restricted to linearized equations of motion, spacetrime constraints, were later found to be generalizations of traditional cubic B-splines and were coined *wiggly splines* due to their "penchant for oscillation" [KA08]. Using explicit representations of wiggly splines computational cost can be reduced and stability be increased [SvTSH15]. Different from these works we consider splines that arise from a Riemannian metric on shell space.

For the remainder of this section we focus on methods that deal with data interpolation (and approximation) on Riemannian manifolds, as this is central to our construction.

Cubic splines minimize the total squared second derivative in Euclidean domains. Analogously, Riemannian cubic polynomials were introduced by Noakes et al. [NHP89] in a variational setting on Riemannian manifolds. This is the *intrinsic* perspective we adopt. In this approach, splines are stationary paths of the elastic functional $s \mapsto \int_0^1 \|\nabla_{\dot{s}}\dot{s}\|^2 \,\mathrm{d}t$, where $\|\nabla_{\dot{s}}\dot{s}\|$ denotes the length of the *covariant derivative* of a curve's tangent vector along itself. Using this approach, Trouvé and Vialard [TV12] presented a spline interpolation method on Riemannian manifolds and Hinkle et al. [HMFJ12] introduced a family of higher order Riemannian polynomials to perform polynomial regression.

Alternatively, one may work with an *extrinsic* variational formulation, i.e., the minimization of $s \mapsto \int_0^1 \|\ddot{s}\|^2 \,\mathrm{d}t$ in ambient space. The restriction of the curve to the manifold is then realized as a constraint. Wallner [Wal04] showed existence of minimizers in this setup for finite dimensional manifolds, and Pottmann and Hofer [PH05] proved that these minimizers are C^2. We do not use an extrinsic formulation since our metric on shell space does not arise from an ambient Euclidean metric.

In addition to variational formulations there are numerous *subdivision schemes* to produce smooth interpolating curves on manifolds. Exploiting the fact that subdivision schemes for curves in linear spaces are mostly based on repeated local averages (see, e.g., [Dyn92, Dyn02]), one obtains a Riemannian extension either by geodesic averaging or by projecting the affine averages onto the manifold. Wallner and Dyn [WD05] showed that the Riemannian extension of cubic subdivision yields C^2 curves, and the authors of [RDS*05] proposed a Deslauriers-Dubuc interpolation scheme.

Some applications call for approximating rather than for interpolating curves with respect to a given set of data points. This can be achieved by replacing hard interpolation by soft penalty constraints [Rei67]. An instance of approximating schemes are Beziér curves. Having the notion of geodesics at hand one can easily transfer this concept to Riemannian manifolds—Beziér curves are simply generated by applying the de Casteljau algorithm with linear interpolation replaced by geodesic interpolation [PN07]. This was done in [ERS*14] for the shape space of images and in [BvTH16] for the space of shells.

3. Review: Shortest paths in shell space

Let $(\mathcal{S},g)$ be a (complete) Riemannian manifold. The *path energy* of a curve $s:[0,1]\to\mathcal{S}$ is defined as

$$\mathcal{E}[s]=\int_0^1 g_{s(t)}\left(\dot s(t),\dot s(t)\right)\mathrm{d}t\,. \tag{1}$$

A minimizer of $\mathcal{E}$ among all curves s with $s(0)=s_A$ and $s(1)=s_B$ is known as a *geodesic*. For s_A and s_B close enough to each other the connecting geodesic is unique and can thus serve as a well-defined interpolating path. The Euler–Lagrange equation to (1) is the geodesic equation $\nabla_{\dot s}\dot s=0$.

Rather than discretizing the Euler–Lagrange equation it is numerically more robust to discretize $\mathcal{E}$ and then minimize the resulting discrete energy. To this end, let $(s_0,\dots,s_K)$ for $K\in\mathbb{N}$ be an ordered set of points in $\mathcal{S}$, which is referred to as a *discrete path*. If $s:[0,1]\to\mathcal{S}$ is a smooth curve passing through $(s_0,\dots,s_K)$ with $s_0=s(0)$ and $s_K=s(1)$, then one has the estimate

$$\mathrm{dist}_g^2(s_0,s_K)\le K\sum_{k=1}^K\mathrm{dist}_g^2(s_{k-1},s_k)\le\mathcal{E}[s]\,,$$

where dist_g denotes the Riemannian distance and equality holds on both sides if and only if s is a geodesic and $s_k=s(\frac{k}{K})$ for all k. Replacing the (squared) Riemannian distance in the sum by a local approximation $\mathcal{W}:\mathcal{S}\times\mathcal{S}\to\mathbb{R}$ that satisfies

$$\mathcal{W}[s,\tilde s]=\mathrm{dist}_g^2(s,\tilde s)+O(\mathrm{dist}_g^3(s,\tilde s))\,, \tag{2}$$

we arrive at the *time-discrete path energy*

$$E^K[s_0,\dots,s_K]=K\sum_{k=1}^K\mathcal{W}[s_{k-1},s_k]\,. \tag{3}$$

In analogy to the continuous case a *discrete geodesic* is defined as a minimizer of (3) for fixed end points $s_0=s_A$ and $s_K=s_B$ and can be viewed as a discrete interpolating path between s_A and s_B. Under suitable assumptions on $\mathcal{S}$, g, and $\mathcal{W}$, discrete geodesics converge to continuous geodesics for $K\to\infty$, see [RW15]. E.g., for an embedded manifold $\mathcal{S}\subset\mathbb{R}^d$, one may simply choose the squared extrinsic distance $\mathcal{W}[s,\tilde s]=\|s-\tilde s\|^2$ as an approximation of the intrinsic one.

Alternatively, instead of first equipping $\mathcal{S}$ with a metric g, one can directly pick an application-motivated function $\mathcal{W}$ provided that $\mathcal{W}$ induces a Riemannian metric via its Hessian. For physics-based shell spaces, $\mathcal{W}[s,\tilde s]$ can for instance be thought of as an elastic deformation energy that reflects the amount of energy needed to deform s into $\tilde s$. We adopt this perspective from [TPBF87,GHDS03,HRWW12] in which $\mathcal{W}$ is composed of a *membrane contribution* due to tangential stretching and a *bending contribution* due to changing shell curvature,

$$\mathcal{W}[s,\tilde s]=\mathcal{W}_{\mathrm{mem}}[s,\tilde s]+\eta\,\mathcal{W}_{\mathrm{bend}}[s,\tilde s]\,,$$

where $\delta=\sqrt{\eta}$ describes the physical thickness of the material layer represented by the shell. As shown in [HRS*14], $\mathcal{W}$ induces a proper Riemannian metric in the space of shells modulo rigid body motions.

Using the above concepts, one may interpolate a sequence of given shells by computing (discrete) geodesics between any two subsequent shells. Examples of such piecewise geodesic interpolating paths are the green shape sequences in Figures 1 to 4. However, those paths exhibit rather abrupt changes in the velocity at keyframe poses. Therefore, in the following we present an interpolation method that leads to smoothly interpolating paths.

4. Splines on Riemannian manifolds

Given a sequence of $J\ge 2$ different time points $t_j\in[0,1]$ and associated data points as keyframe poses $s^j\in\mathcal{S}$, $j=1,\dots,J$, we seek a smooth curve $s:[0,1]\to\mathcal{S}$ that satisfies the *interpolation constraints*

$$s(t_j)=s^j\,,\quad j=1,\dots,J\,. \tag{4}$$

Unfortunately, the piecewise geodesic interpolation paths from the previous section are not smooth at the times t_j, and for $J>2$ there is in general no interpolating geodesic, i.e., a curve s satisfying (4) as well as $\nabla_{\dot s}\dot s(t)=0$ for all $t\in[0,1]$. However, instead of requiring the interpolating curve to satisfy $\nabla_{\dot s}\dot s=0$ exactly, we can penalize a deviation from this constraint via the so-called *elastic energy*

$$\mathcal{F}[s]=\int_0^1 g_{s(t)}\left(\nabla_{\dot s}\dot s(t),\nabla_{\dot s}\dot s(t)\right)\mathrm{d}t\,. \tag{5}$$

Accordingly, we define a *Riemannian spline* through the points (t_j,s^j) as a minimizer s of $\mathcal{F}[s]$ under the interpolation constraint (4). In addition we may optionally impose one of the two boundary conditions

$$\begin{aligned}&\dot s(0)=v_0,\quad \dot s(1)=v_1\quad\text{for given } v_0,v_1\in T\mathcal{S}\,, &&\text{(Hermite b. c.)}\\ &s(0)=s(1),\quad \dot s(0)=\dot s(1)\,, &&\text{(periodic b. c.)}\end{aligned}$$

where $T\mathcal{S}$ denotes the tangent bundle on $\mathcal{S}$. The case without additional conditions is referred to as natural boundary condition.

For a better intuition and to motivate our terminology it is helpful to consider the Euclidean setting $\mathcal{S}=\mathbb{R}^d$, in which $\nabla_{\dot s}\dot s(t)=\ddot s(t)$ and $\mathcal{F}[s]=\int_0^1\|\ddot s(t)\|^2\,\mathrm{d}t$. In that setting a result by de Boor [dB63] states that there is a unique minimizer s of $\mathcal{F}[s]$ with (4) and natural, Hermite, or periodic boundary conditions. Furthermore, s is given by the unique cubic spline satisfying (4) and the boundary conditions. Hence, we here essentially generalize cubic spline interpolation to Riemannian manifolds. The orange shape sequences in Figures 1, 4, and 7 show computed (time-discrete) Riemannian splines with periodic, natural, and Hermite boundary conditions, respectively.

Remark: On general manifolds there may be situations where global minimizers of (5) do not exist. To ensure well-posedness one can regularize the problem and consider minimizers of the functional $s\mapsto\mathcal{F}[s]+\sigma\mathcal{E}[s]$ with $\sigma>0$. Under certain additional assumptions on $\mathcal{S}$ and g one can then show existence of minimizers of $\mathcal{F}+\sigma\mathcal{E}$ (see Section 9).

5. Variational time-discretization of spline curves

In this section we introduce a consistent time-discretization of the elastic energy $\mathcal{F}$ introduced in Section 4 mimicking the variational time-discretization for the path energy introduced in [HRWW12] and revisited in Section 3. As a motivation, first consider discrete splines in Euclidean space, where the covariant derivative of the velocity field of a curve $s:[0,1]\to\mathbb{R}^d$ coincides with $\ddot{s}$. For a uniform sampling $s_k = s(t_k)$ for $t_k = k\tau$, $k=0,\dots,K$, and time step $\tau = 1/K$ one obtains the standard second order difference quotient approximation

$$\|\ddot{s}(t_k)\|^2 \approx \left\|\frac{2s(t_k)-s(t_{k-1})-s(t_{k+1})}{\tau^2}\right\|^2 = 4K^4\left\|s_k - \frac{s_{k-1}+s_{k+1}}{2}\right\|^2. \quad (6)$$

Taking into account $\mathcal{W}[s,\tilde{s}] = \|s-\tilde{s}\|^2$, the term on the right hand side simplifies to $4K^4\,\mathcal{W}[s_k,\frac{1}{2}(s_{k+1}+s_{k-1})]$. Using the simple numerical quadrature $\int_0^1 f(t)\,\mathrm{d}t \approx \tau\sum_{k=1}^{K-1} f(t_k)$ yields

$$\int_0^1 \|\ddot{s}(t)\|^2\,\mathrm{d}t \;\approx\; 4K^3\sum_{k=1}^{K-1}\mathcal{W}[s_k,\tfrac{1}{2}(s_{k+1}+s_{k-1})]\,.$$

Notice that this formulation implicitly incorporates (time-discrete) natural boundary conditions.

The local average $\frac{1}{2}(s_{k-1}+s_{k+1})$ can be considered as the midpoint of a straight line, i.e., a geodesic, connecting s_{k-1} and s_{k+1}. This observation leads to a translation of the above formulation to Riemannian manifolds. Indeed, given a discrete path $(s_0,\dots,s_K)$ in a *general* Riemannian manifold $\mathcal{S}$ with $\mathcal{W}$ as in (2), we may replace the local average in (6) by the midpoint of a (shortest) geodesic. Accordingly, we define the *time-discrete elastic energy* by

$$F^K[s_0,\dots,s_K] = 4K^3\sum_{k=1}^{K-1}\mathcal{W}[s_k,\tilde{s}_k]\,, \quad (7)$$

for a discrete path $(s_0,\dots,s_K)$. Here, $\tilde{s}_k$ is defined by requiring that $(s_{k-1},\tilde{s}_k,s_{k+1})$ be a time-discrete geodesic connecting s_{k-1} and s_{k+1} for $k=1,\dots,K-1$, i.e.,

$$\tilde{s}_k = \arg\min_s\Big(\mathcal{W}[s_{k-1},s]+\mathcal{W}[s,s_{k+1}]\Big)\,. \quad (8)$$

Observe that for embedded manifolds $\mathcal{S}\subset\mathbb{R}^d$ the choice $\mathcal{W}[s,\tilde{s}] = \|s-\tilde{s}\|^2$ leads to a consistent notion of discrete spline curves on $\mathcal{S}$. Under suitable assumptions on the manifold $\mathcal{S}$ and on $\mathcal{W}$ it can be shown that in general F^K is a consistent first order approximation of $\mathcal{F}$.

In order to compute a spline curve in shell space, one has to minimize the energy F^K among all discrete curves $(s_0,\dots,s_K)$ subject to the interpolation conditions $s_{k_j} = s^j$ for $t_j = k_j\tau$ and $j=1,\dots,J$ and the additional built-in constraints (8). Notice that we have implicitly assumed that the interpolation times are multiples of the time step $\tau = K^{-1}$. This restriction, however, can easily be removed using varying time step sizes and suitable adaptations of F^K.

Remark: Bounded time-discrete elastic energy does not necessarily imply bounded time-discrete path energy in general. Indeed, a discrete geodesic has zero elastic energy but positive path energy. As a consequence, minimizing time-discrete elastic energy might not be a well-posed problem. However, one can consider the functional $F^K+\sigma E^K$ instead (see Section 4). Again one can show existence of discrete minimizers for every $\sigma>0$ subject to the interpolation conditions as well as the convergence of the discrete energy against its original continuous version (see Section 9). In our applications we stick to the elastic energy only, i.e., we set $\sigma=0$, since we never observed instabilities or blow-ups.

6. Splines on shell space

The above approach can be applied to arbitrary (complete) Riemannian manifolds, and we now restrict it to discrete shell space. We assume a given correspondence between discrete shells through fixed connectivity. In this case we obtain $\mathcal{S}\hat{=}\mathbb{R}^{3N}$, where N is the number of vertices of one triangle mesh, and we identify a discrete shell with its vector of nodal positions $\mathbf{s}\in\mathbb{R}^{3N}$. To compute distances on shell space, we use $\mathcal{W}$ given by the *Discrete Shells* energy [GHDS03], which induces a Riemannian metric on the space of triangle meshes modulo rigid motions [HRS*14].

For a discrete shell $\mathbf{s}\in\mathbb{R}^{3N}$ let $\mathcal{V}$, $\mathcal{E}$ and $\mathcal{T}$ denote the set of vertices, edges and triangles, respectively. Let $L[\mathbf{s}] = (l_e[\mathbf{s}])_e\in\mathbb{R}^{|\mathcal{E}|}$ and $A[\mathbf{s}] = (a_t[\mathbf{s}])_t\in\mathbb{R}^{|\mathcal{T}|}$ be the vector of edge lengths and triangle areas, respectively. If $e = t_1\cap t_2$ is the common edge of triangles t_1 and t_2 we associate to e the area measure $d_e = \frac{1}{3}(a_{t_1}+a_{t_2})$. The dihedral angle θ_e of e is defined as the angle between the face normals of t_1 and t_2. Finally, let $\Theta[\mathbf{s}] = (\theta_e[\mathbf{s}])_e\in\mathbb{R}^{|\mathcal{E}|}$ denote the vector of dihedral angles. Having these definitions in hand, the *discrete shell* energy for deforming $\mathbf{s}$ into $\tilde{\mathbf{s}}$ is

$$\mathcal{W}[\mathbf{s},\tilde{\mathbf{s}}] = \mu\mathcal{W}_L[\mathbf{s},\tilde{\mathbf{s}}]+\lambda\mathcal{W}_A[\mathbf{s},\tilde{\mathbf{s}}]+\eta\,\mathcal{W}_\Theta[\mathbf{s},\tilde{\mathbf{s}}] \quad (9)$$

with physical parameters $\mu,\lambda,\eta\geq 0$ and

$$\mathcal{W}_L[\mathbf{s},\tilde{\mathbf{s}}] = \sum_{e\in\mathcal{E}} d_e[\mathbf{s}]\left(\frac{l_e[\mathbf{s}]-l_e[\tilde{\mathbf{s}}]}{l_e[\mathbf{s}]}\right)^2,$$

$$\mathcal{W}_A[\mathbf{s},\tilde{\mathbf{s}}] = \sum_{t\in\mathcal{T}} a_t[\mathbf{s}]\left(\frac{a_t[\mathbf{s}]-a_t[\tilde{\mathbf{s}}]}{a_t[\mathbf{s}]}\right)^2,$$

$$\mathcal{W}_\Theta[\mathbf{s},\tilde{\mathbf{s}}] = \sum_{e\in\mathcal{E}} l_e[\mathbf{s}]^2\,\frac{(\theta_e[\mathbf{s}]-\theta_e[\tilde{\mathbf{s}}])^2}{d_e[\mathbf{s}]}\,.$$

In all numerical experiments we used $\mu=\lambda=1$, since local change of length (controlled by μ) and local change of area (controlled by λ), respectively, should be penalized equally. However, the optimal bending parameter $\eta=\delta^2$ depends on the application since δ represents the physical thickness of the shell (see Figure 10). The colored font is for later reference when we introduce our $L\Theta A$ approximation for efficiently computing splines.

In Figure 3 we show a discrete spline curve for $\mathcal{W}$ defined as in (9) and for three different ellipsoids with varying half axes as keyframe poses to be interpolated. In particular in the halfaxis plot the smoothness of the spline curve at the intermediate keyframe pose becomes visible. Figure 4 depicts a discrete spline curve for three different keyframe poses of a cactus-type discrete shell and compares it to the piecewise geodesic interpolation. The boundary keyframe poses $\mathbf{s}_0$ and $\mathbf{s}_{20}$ are given as deformations of the cactus rest pose $\mathbf{s}_{10}$ in two orthogonal directions, which leads to a sharp corner at $\mathbf{s}_{10}$ when performing piecewise geodesic interpolation.

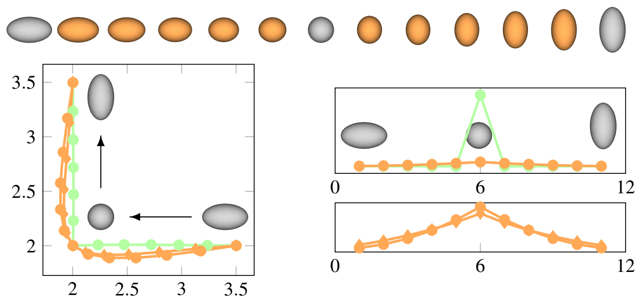

Figure 3: *Top row: Spline curve interpolation between $J = 3$ fixed ellipsoids ($K = 12$, $\eta = 10^{-1}$). Bottom, left: Representation of ellipsoids in $\mathbb{R}^2$ by their half axes in x- and y-directions. Plotting the evolution of eccentricity along the deformation one can see the difference between the piecewise geodesic (green), the nonlinear spline (orange circles), and the $L\Theta A$-space approximation (orange diamonds). Bottom, right: Comparison of piecewise geodesic (green) vs. nonlinear spline (orange, top) and comparison of the nonlinear spline and the $L\Theta A$-space curve (bottom, ordinate rescaled by* 10*).*

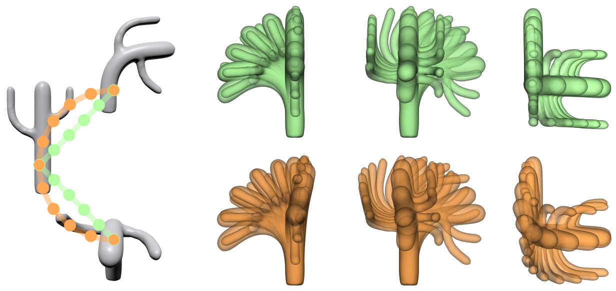

Figure 4: *Left: $J = 3$ fixed keyframe poses (gray). Right: Different views of piecewise geodesic (green) and spline interpolation (orange), respectively, using $K = 10$, $\eta = 10^{-3}$.*

Limitations. As in the Euclidean case, a particular feature of spline interpolation is the so-called overshooting—a consequence of the smoothness requirement. Figure 5 shows an example of this effect for the case of (discrete) Riemannian splines in shell space. For certain applications this feature might be undesired, though.

The biggest limitation, however, is computational time. Indeed, since the mappings $\mathbf{s} \mapsto \{l_e[\mathbf{s}], d_e[\mathbf{s}], \theta_e[\mathbf{s}], a_t[\mathbf{s}]\}$ are nonlinear functions, the minimization of (7) is a nonlinear problem in $\mathbb{R}^{3N(K+1-J)}$ with $K-1$ nonlinear constraints given by (8). In our numerical experiments this leads to slow convergence even for relatively small N and K. For example, the computation of the ellipsoid sequence ($N = 500$, $K = 12$) shown in Figore 3 or the short cactus sequence ($N = 5261$, $K = 20$) shown in Figure 4 can take several minutes or even hours, respectively. In order to overcome this limitation we introduce an effective change of variables in the next section.

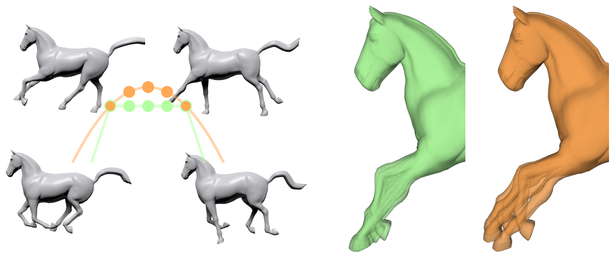

Figure 5: *Overshooting of the left front leg in spline interpolation (orange) compared to the piecewise geodesic interpolation (green) between 4th and 5th keyframe pose of the horse sequence, see video in supplementary material.*

7. A simplified spline model

In order to remedy the problem of high computational cost, we introduce a change of variables in order to turn the nonlinear optimization problem for computing splines in shell space into a linear one. We heavily build on the two-step approximation scheme proposed in [FB11] for this change of coordinates. Indeed, for a discrete shell $\mathbf{s} \in \mathbb{R}^{3N}$ we consider the vectors of edge lengths $L = L[\mathbf{s}]$, dihedral angles $\Theta = \Theta[\mathbf{s}]$, and triangle areas $A = A[\mathbf{s}]$ as primary degrees of freedom. The key observation is that with these degrees of freedom, the energy in (9) become *quadratic* provided that the purple colored terms in (9) are *not* part of the optimization. To achieve this, we work with what we call *reference meshes*, i.e., we replace the purple colored terms in (9) by a priori given meshes $(\hat{\mathbf{s}}_0, \ldots, \hat{\mathbf{s}}_K)$. In the simplest case, this a priori information is given by using a constant reference mesh $\hat{\mathbf{s}}_k = \mathbf{s}$, $k = 0, \ldots, K$, for one particular keyframe pose $\mathbf{s} \in \mathbb{R}^{3N}$. Alternatively, one may use a piecewise time-discrete geodesic between the given keyframe poses to obtain $(\hat{\mathbf{s}}_0, \ldots, \hat{\mathbf{s}}_K)$. Physically, the latter choice is more appropiate since the reference mesh is supposed to approximate the (undeformed) argument $\mathbf{s}$ in (9). We discuss the physical meaning and the impact of the choice of reference meshes in Section 9 (see also Figure 9).

Collecting all new primary variables in one variable $\mathbf{z} = (L, \Theta, A)$ living in the $L\Theta A$ configuration space $\mathbb{R}^{|\mathcal{E}|} \times \mathbb{R}^{|\mathcal{E}|} \times \mathbb{R}^{|\mathcal{T}|}$, we get an approximation of (9) via

$$\widehat{\mathcal{W}}_{L\Theta A}[\mathbf{z}, \tilde{\mathbf{z}}] = \mu \widehat{\mathcal{W}}_L[\mathbf{z}, \tilde{\mathbf{z}}] + \lambda \widehat{\mathcal{W}}_A[\mathbf{z}, \tilde{\mathbf{z}}] + \eta \widehat{\mathcal{W}}_\Theta[\mathbf{z}, \tilde{\mathbf{z}}] \qquad (10)$$

where the $\hat{\cdot}$ indicates that the functional is now quadratic but dependent on the reference meshes. We refer to this as the $L\Theta A$-energy. Note that the parameters $\mu, \lambda, \eta \geq 0$ have the same physical interpretation as in (9). As in the nonlinear setup we set $\mu = \lambda = 1$ and chose $\eta = \delta^2$ depending on the application (see Figure 10). We discuss in particular the impact of λ in the last paragraph of this section.

The discrete interpolation problem can now be re-formulated as follows. We construct a spline curve in the $L\Theta A$-space defined as a minimizer of

$$\hat{F}_{L\Theta A}[\mathbf{z}_0, \ldots, \mathbf{z}_K] = 4K^3 \sum_{k=1}^{K} \widehat{\mathcal{W}}_{L\Theta A}(\mathbf{z}_k, \tilde{\mathbf{z}}_k) \,, \qquad (11)$$

where $(\mathbf{z}_{k-1}, \tilde{\mathbf{z}}_k, \mathbf{z}_{k+1})$ is a geodesic in the $L\Theta A$-space, i.e.,

$$\tilde{\mathbf{z}}_k = \arg\min_{\mathbf{z}} \left(\widehat{\mathcal{W}}_{L\Theta A}[\mathbf{z}_{k-1}, \mathbf{z}] + \widehat{\mathcal{W}}_{L\Theta A}[\mathbf{z}, \mathbf{z}_{k+1}] \right) .$$

Since $\widehat{\mathcal{W}}_{L\Theta A}$ is *quadratic*, one obtains the explicit solution $\tilde{\mathbf{z}}_k$ as a linear combination of $\mathbf{z}_{k-1}$ and $\mathbf{z}_{k+1}$ with coefficients depending on $\hat{\mathbf{s}}_{k-1}$ and $\hat{\mathbf{s}}_k$ only. In particular, if we make use of a constant reference mesh then $\tilde{\mathbf{z}}_k = \frac{1}{2}(\mathbf{z}_{k-1} + \mathbf{z}_{k+1})$. This can be inserted into (11) so that we end up with an unconstrained optimization problem. Hence a minimizer of $\hat{F}_{L\Theta A}$ is a (weighted) cubic spline in the linear space $\mathbb{R}^{|\mathcal{E}|} \times \mathbb{R}^{|\mathcal{E}|} \times \mathbb{R}^{|\mathcal{T}|}$.

Notice that there is no *spatial* coupling between any two different edge lengths in a minimizer $(\mathbf{z}_0, \ldots, \mathbf{z}_K)$ of (11), i.e., an edge length l_e^k of the k^{th} pose interacts only with lengths l_e^j of the same edge e and poses $j \neq k$. The same applies for dihedral angles and triangle areas. As a consequence, the Euler-Lagrange equation for $\hat{F}_{L\Theta A}$ splits into numerous independent $(K+1)$-dimensional linear systems, i.e., one for each edge length, dihedral angle, and triangle area, which can be solved efficiently and in parallel. Moreover, if one chooses a constant reference mesh, then the matrices representing these linear systems are all given by

$$\begin{pmatrix} 1 & -2 & 1 & & & \\ -2 & 3 & -4 & 1 & & \\ 1 & -4 & 6 & -4 & 1 & \\ & 1 & -4 & 6 & -4 & 1 \\ & & & & \ddots & \end{pmatrix} \in \mathbb{R}^{K+1,K+1},$$

which coincides (for interior quantities) to the 2nd order finite difference approximation of 4th derivatives.

Mesh reconstruction. Intermediate values for edge lengths, dihedral angles, and triangle areas are generally not realizable as a triangle mesh. Hence we consider a reconstruction in a least squares sense, similar to [FB11]. For given optimal values $\mathbf{z}_k = (L_k, \Theta_k, A_k)$ we define $\mathbf{s}_k$ as the minimizer of the nonlinear mapping

$$\mathbf{s} \mapsto \widehat{\mathcal{W}}_{L\Theta A}(\mathbf{z}[\mathbf{s}], \mathbf{z}_k) , \tag{12}$$

where $\widehat{\mathcal{W}}_{L\Theta A}$ is defined as in (10). We find the minimizer via the Gauss–Newton method (see Section 6 in [FB11] for details, the "target" values are given by $\mathbf{z}_k$). Furthermore we use exactly the same physical parameters $\mu, \lambda, \eta \geq 0$ as in the optimization of (11). The reconstruction can be seen as a projection of the point $\mathbf{z} \in \mathbb{R}^{|\mathcal{E}|} \times \mathbb{R}^{|\mathcal{E}|} \times \mathbb{R}^{|\mathcal{T}|}$ onto the submanifold which is given by all sets of edge lengths, dihedral angles and triangle areas that are actually realizable as an embedded triangle mesh. Necessary conditions for points to lie in this submanifold are given by the Gauss-Codazzi equations, see, e.g., [WLT12] for a discrete version. Computationally, the reconstruction is the hardest part in the $L\Theta A$-space approximation method. Fortunately it can be parallelized.

Negative descriptors. Optimal $\mathbf{z}$-variables obtained as solutions of linear systems may have negative lengths or areas. This happens rarely in practice and is most easily addressed by setting corresponding edge or area weights to zero.

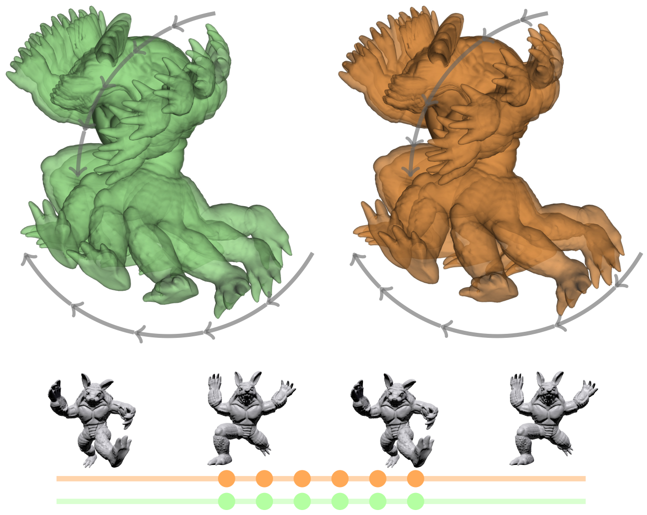

Figure 6: *Piecewise geodesic (green) and spline curve interpolation (orange) between Armadillo keyframe poses (gray) induce similar trajectories but differences in speed, see video in supplementary material.*

Role of area term. Different from [FB11] we use an additional area term, i.e., $\lambda > 0$ in (9) and (10). While we empirically found that this term is not required for meshes with a reasonable aspect ratio, there are examples (e.g., the running horse application) where we observe artifacts in the reconstruction if $\lambda = 0$. Moreover, when optimizing (7) using (9), the stability of the optimization is increased and the convergence time is decreased if $\lambda > 0$.

8. Implementation

Constrained optimization. The nonlinear minimization of (7) under constraint (8) is solved via gradient descent with stepsize control in all primal variables $\mathbf{s}_k$. In the appendix we provide the requisite partial derivatives $F^K_{,k} \in \mathbb{R}^{3N}$ of the constrained energy (7) with respect to the primal variables $\mathbf{s}_k \in \mathbb{R}^{3N}$ for $0 < k < K$.

Multiresolution optimization. The efficiency of the nonlinear optimization can be increased by a multiresolution scheme [FB11, BSPG06], which is based on a separation of low frequency shape information from high frequency detail. As in [KMP07] we decimate all input meshes simultaneously to a complexity of at most 1000 vertices, where we use the sum of per-mesh quadric error metrics [GH97] to prioritize halfedge collapses. In particular, the coarse meshes still have the same connectivity since each halfedge collapse is performed on all meshes simultaneously. The nonlinear optimization is then performed on the reduced meshes. Afterwards, the solution of the coarse level is prolongated to the original resolution (for details, we refer to [FB11, Fig. 11]). The multiresolution scheme is used for the optimization of (7) as well as for minimizing (12).

***L*Θ*A*-space approximation.** Our $L\Theta A$-space approximation consists of three steps:

	#nodes	G.-N.	decim.	reconstr.
cactus	5.2k	500	140	80
horse	8.5k	700	190	130
armadillo	166k	/	7800	3800

Table 1: *Performance statistics for mesh reconstruction (times in ms) measured on Dell Intel(R) Core(TM) i7-2600 3.40GHz. From left to right: number of vertices of high resolution mesh, time for one Gauss–Newton iteration on the original resolution, for the multiresolution preprocessing, i.e., the generation of coarse meshes from fine meshes (decimation), and for the reconstruction (see [FB11, Tab. 1]). On the coarse mesh, one Gauss–Newton step takes about 50ms. On the original Armadillo model, the Gauss–Newton step failed due to memory restrictions.*

1. Simultaneous mesh decimation for all keyframe poses,
2. $L\Theta A$-space optimization based on solution of multiple but small linear systems and shell reconstruction on coarse level by means of Gauss–Newton,
3. prolongation to fine level.

For the reconstruction—the computationally dominant part—we reproduce the computation times stated in [FB11], as listed in Table 1. To enable real-time computations we take the prolongation of the coarse solution as the final solution, which is visually sufficient. In some examples, after reconstruction and prolongation we observe local mesh degenerations in single shapes, e.g., at the fingers in Figure 1 and the tail in Figure 5, which are removed in a post-processing step. One may also perform the $L\Theta A$-optimization directly for the high resolution keyframe poses. In this case it pays off computationally if the reconstruction step is initialized with the meshes from the multiresolution approach. In our experiments, the resulting shells may differ quantitatively but not qualitatively from the computationally more efficient prolongation of coarse grid discrete spline curves.

Total runtimes (without parallelization) of the $L\Theta A$-space approximation are shown in Table 2. In addition, the reconstruction (which amounts to more than 90% of the cost) as well as the prolongation can easily be parallelized, thereby reducing runtime proportionally to the number of available threads.

model (K, γ)	decim.	coarse opt./recon.	prol.	fine opt./recon.
cactus (20, .05)	0.1	1.7	1.7	83
cactus (170, .05)	0.3	22	14	762
horse (40, .10)	0.2	22	5	238
armad. (50, .006)	5.2	80	200	/

Table 2: *Total runtime (in s) of $L\Theta A$-space scheme for computing spline curves of length $K+1$: mesh decimation with γ indicating the fraction of remaining nodes, optimization and reconstruction on coarse level, prolongation using detail transfer, and optimization and reconstruction on fine level using the previous result as initialization for the reconstruction step. Measured without parallelization on Dell Intel(R) Core(TM) i7-2600 3.40GHz.*

Rigid body motions. Shell deformation energies typically do not penalize rigid body motions (translations or rotations in $\mathbb{R}^3$) so that the kernel of $\mathcal{W}$ as defined in (9) has dimension 6. For computational stability, the arbitrary rigid body motion can be fixed by prescribing zeroth and first momentum for each shape. Alternatively, one may restrict to a subspace of shell space in which a certain subset of vertices is fixed for physical or modelling reasons (such as the base of the cactus in Figure 4 or the outer boundary of the faces in Figure 11).

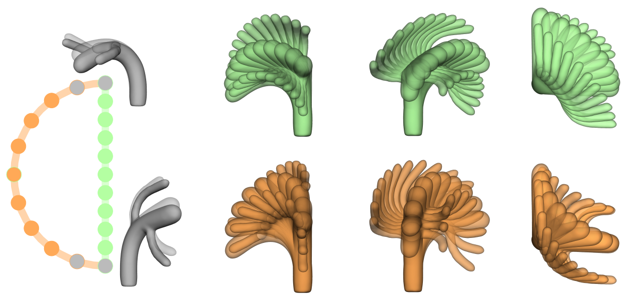

Figure 7: *Interpolation of two keyframe poses (gray) via a spline curve with natural boundary conditions (green; equivalent to a geodesic curve) and with Hermite boundary conditions (orange) for $K = 10$, $\eta = 10^{-3}$. The Hermite boundary conditions are emulated by fixing two additional keyframe poses $\mathbf{s}_1$ and $\mathbf{s}_9$ (light gray, to emphasize the difference they are taken from the piecewise geodesic curve in Figure 4).*

Boundary conditions. Our exposition has so far focussed on natural boundary conditions. Periodic boundary conditions are incorporated by identifying $\mathbf{s}_k = \mathbf{s}_{(k+K) \bmod K}$ for $k = 0, \dots, K$. Hermite boundary conditions, which prescribe endpoint velocities and are useful for blending purposes, are incorporated in the time-discrete setup by fixing $\mathbf{s}_0$ and $\mathbf{s}_1$ as well as $\mathbf{s}_{K-1}$ and $\mathbf{s}_K$, respectively. A comparison of Hermite and natural boundary conditions is shown in Figure 7.

9. Discussion

In this article we introduce the notion of Riemannian splines in the space of shells for computing globally smooth interpolation paths between multiple prescribed keyframe poses. The interpolation paths are obtained either by solving a nonlinear constrained optimization problem or by resorting to a simplified model, the $L\Theta A$-approximation. Compared to piecewise geodesic interpolation there are two striking differences: (i) The trajectory of the spline curve exhibits a time continuous acceleration $(\nabla_{\dot{s}}\dot{s})$ and thus is visually smooth, whereas a piecewise geodesic curve suffers from corners at the keyframe poses (see Figure 1, 4, 8). (ii) The spline curve balances acceleration along the path leading to a variation in speed (see Figure 6), whereas the piecewise geodesic approximately has constant speed. Furthermore, Riemannian splines can incorporate different kinds of boundary conditions, which leads to flexibility in animation, compare, e.g., Figures 4 and 7.

Comparison of approaches. The nonlinear model is consistent with the notion of Riemanian splines and follows rigorously from the underlying physical model. The simplified $L\Theta A$-model

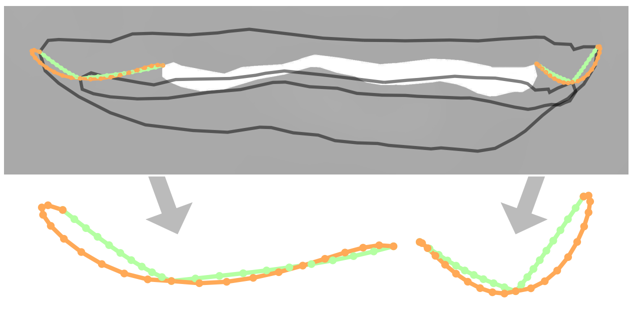
Figure 8: *Zoomed contours of the mouth in keyframe pose $\mathbf{s}_{10}$, $\mathbf{s}_{20}$, and $\mathbf{s}_{30}$ in Figure 11 with trajectories of the corners of the mouth plotted for the spline curve (orange) and the piecewise geodesic curve (green).*

shows the same qualitative behavior as the nonlinear model and is significantly more efficient. Indeed, the simplified energy is quadratic, and geometric compatibility conditions between the triangle lengths, angles, and areas are ignored during the energy minimization, resulting in many but small easily solvable decoupled problems. Yet, using reference meshes in order to obtain a quadratic energy might seem unsatisfactory from a physical point of view since strains are measured with respect to these (artificial) reference meshes. A more physical approach could be recovered by a fixpoint iteration that alternates between computing an interpolation path via minimizing (11) and updating the reference meshes. As exemplified in Figure 9 such a fixpoint iteration is expected to converge but typically only leads to very minor interpolation corrections so that we do not advocate it. The decoupling between edge

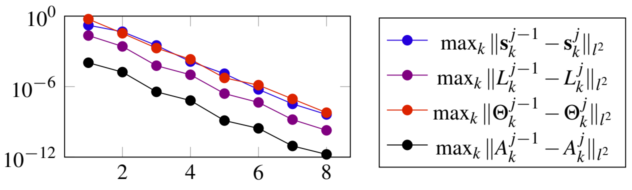

Figure 9: *Maximum difference in cactus sequence from Figure 4 for $j = 1, \ldots, 8$ fixpoint iterations, measured in different norms.*

lengths, dihedral angles, and triangle areas represents a more severe deviation from the original model. Perhaps the simplest intuitive example for this fact is an interpolation between a triangulated half sphere and its inverted version (i.e., the interpolation pushes the half sphere inside out)—while the nonlinear model (7) takes the changing triangle edge lengths into account to achieve an optimal interpolation, the $L\Theta A$-approximation does not alter edge length during interpolation since the two end shells have identical corresponding edge lengths. This may result in different interpolation paths already for simple cases (cf. Figure 3).

Dependence on physical parameters. Given a set of input shells the resulting spline depends on the physical parameters λ, μ, and η of the model (see [HRWW12, Figure 11] for the dependence of discrete geodesics on λ and μ). Let us focus here on the dependence of splines on the thickness parameter η. In the limit $\eta \to 0$ the energy $\mathcal{W}$ measures the deviation from isometric deformations so that a spline stays near the submanifold of isometrically deformed shells. For larger values of η splines may leave this submanifold as shown in Figure 10 for input shells which are all isometric to a planar sheet.

Robustness. While the $L\Theta A$-model enables fast computations and works for large K and N, the Gauss–Newton iteration inside the $L\Theta A$-scheme is quite sensitive to mesh quality and physical parameters. For example, it sometimes requires a fine parameter tuning in order to enforce convergence to non-degenerate meshes in all reconstructions. In contrast, the constrained optimization of the nonlinear model is very robust and prevents any mesh degenerations.

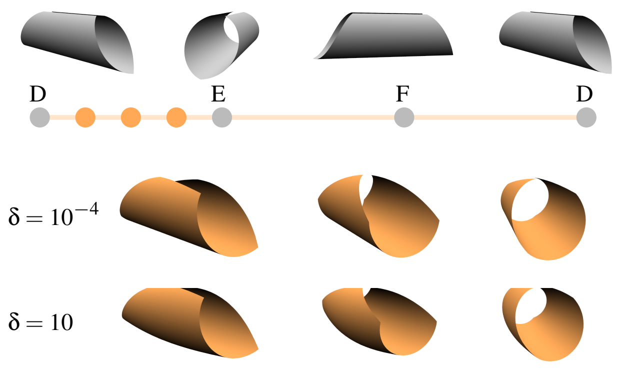

Figure 10: *Periodic spline $D \to E \to F \to D$ with $K = 12$ and different bending weights $\eta = \delta^2$, where δ represents the thickness of the thin sheet (see also Fig. 10 in [HRS*14]). The discrete segments $E \to F$ and $F \to D$ are not shown due to the symmetry of the problem. Note that all input shells D, E, F are isometric deformations of a regular and flat hexagon H. For $\delta = 10^{-4}$ the spline (almost) stays in the subspace of isometric deformations of H, whereas this is no longer the case for $\delta = 10$, as can be seen by the bending of the middle shape in the bottom row.*

Mathematical well-posedness. Since the $L\Theta A$-model mainly acts in the Euclidean $L\Theta A$-configuration space, it enjoys all nice mathematical properties of standard Euclidean spline interpolation. In contrast, the existence and convergence analysis for the nonlinear model (i.e., the minimization of (5) in the continuous and (7) in the discrete case) is considerably more involved. Nevertheless it can be performed exploiting similar variational tools as have been used in [RW15]. Concerning the existence of continuous Riemannian splines the major problem is that paths with bounded elastic energy $\mathcal{F}$ may exhibit arbitrarily large path energy $\mathcal{E}$. To see this, consider a geodesic curve that is traversed with constant speed; such a curve has zero elastic energy but arbitrarily high path energy. For instance, consider a cylinder of radius 1 and three input points s^0, $s^1 = s^0$, and s^2 being opposite of s^0 with $t_0 = 0$, $t_1 = \frac{1}{n}$, and $t_2 = 1$. Then, there is a curve fulfilling the interpolation constraints (4), which winds around the cylinder one time during the time interval $[0, t_1]$ and $n-1$ times during the time interval $[t_1, 1]$. This curve has constant speed $2\pi n$ along the cylinder and thus zero spline energy $\mathcal{F}$ but a path energy $\mathcal{E}$ of $4\pi^2 n^2$. This problem is circumvented by the energy $\mathcal{F}+\sigma\mathcal{E}$, and the existence of minimizers of the time continuous and the time-discrete model can be proved via the general procedure of the direct method in the calculus of variations. Finally, the convergence of discrete splines to a continuous spline path follows from the Γ-convergence of the discrete elastic energy to the continuous elastic energy. The analytical details of a proof of this convergence are beyond the scope of this paper and will appear in a separate publication.

Acknowledgments

Benedikt Wirth's research was supported by the Alfried Krupp Prize for Young University Teachers awarded by the Alfried Krupp von Bohlen und Halbach-Stiftung. Behrend Heeren and Martin Rumpf acknowledge support by the FWF in Austria under the grant S117 (NFN) and by the Hausdorff Center. The authors thank Dennis Tiege for rendering.

References

[BSPG06] Botsch M., Sumner R., Pauly M., Gross M.: Deformation transfer for detail-preserving surface editing. In *Vision, Modeling & Visualization* (2006), pp. 357–364. 6

[BvTH16] Brandt C., von Tycowicz C., Hildebrandt K.: Geometric flows of curves in shape space for processing motion of deformable objects. *Computer Graphics Forum 35(2)* (2016). 2, 3

[dB63] de Boor C.: Best approximation properties of spline functions of odd degree. *J. Math. Mech. 12* (1963), 747–749. 3

[Dyn92] Dyn N.: Subdivision schemes in CAGD. In *Advances in Numerical Analysis*, Light W., (Ed.), vol. 2. Oxford Univ. Press, 1992, pp. 36–104. 2

[Dyn02] Dyn N.: Interpolatory subdivision schemes. In *Tutorials on Multiresolution in Geometric Modelling*, Iske A., Quak E., Floater M. S., (Eds.). Springer, Berlin, 2002, pp. 25–50. 2

[ERS*14] Effland A., Rumpf M., Simon S., Stahn K., Wirth B.: Bézier curves in the space of images. In *Proceedings Scale Space and Variational Methods in Computer Vision* (2014), Lecture Notes in Computer Science, Springer. 3

[FB11] Fröhlich S., Botsch M.: Example-driven deformations based on discrete shells. *Computer Graphics Forum 30 8* (2011), 2246–2257. 2, 5, 6, 7

[GH97] Garland M., Heckbert P.: Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH 97* (1997), Annual Conference Series, pp. 209–216. 6

[GHDS03] Grinspun E., Hirani A. N., Desbrun M., Schröder P.: Discrete shells. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2003), pp. 62–67. 3, 4

[HMFJ12] Hinkle J., Muralidharan P., Fletcher P., Joshi S.: Polynomial regression on Riemannian manifolds. In *ECCV* (2012). 2

[HRS*14] Heeren B., Rumpf M., Schröder P., Wardetzky M., Wirth B.: Exploring the geometry of the space of shells. *Computer Graphics Forum 33*, 5 (2014), 247–256. 1, 2, 3, 4, 8

[HRWW12] Heeren B., Rumpf M., Wardetzky M., Wirth B.: Time-discrete geodesics in the space of shells. *Computer Graphics Forum 31*, 5 (2012), 1755–1764. 1, 2, 3, 4, 8

[KA08] Kass M., Anderson J.: Animating oscillatory motion with overlap: Wiggly splines. *ACM Trans. Graph. 27*, 3 (2008), 28:1–28:8. 2

[KB84] Kochanek D. H. U., Bartels R. H.: Interpolating splines with local tension, continuity, and bias control. *Computer Graphics 18*, 3 (1984), 33–41. 1

[KMP07] Kilian M., Mitra N. J., Pottmann H.: Geometric modeling in shape space. *ACM Trans. Graph. 26* (2007), 1–8. 1, 2, 6

[Las87] Lasseter J.: Principles of traditional animation applied to 3d computer animation. *Computer Graphics 21*, 4 (1987), 35–44. 1

[NHP89] Noakes L., Heinzinger G., Paden B.: Cubic splines on curved spaces. *IMA J. Math. Control Inform. 6* (1989), 465–473. 2

[PH05] Pottmann H., Hofer M.: A variational approach to spline curves on surface. *Computer Aided Geometric Design 22* (2005), 693–709. 2

[PN07] Popiel T., Noakes L.: Bézier curves and C^2 interpolation in riemannian manifolds. *J. Approx. Theory 148*, 2 (2007), 111–127. 3

[RDS*05] Rahman I. U., Drori I., Stodden V. C., Donoho D. L., Schröder P.: Multiscale representations for manifold-valued data. *Multiscale Model. Simul. 4*, 4 (2005), 1201–1232. 2

[Rei67] Reinsch C. H.: Smoothing by spline functions. *Numer. Math. 10* (1967), 177–183. 2

[RW15] Rumpf M., Wirth B.: Variational time discretization of geodesic calculus. *IMA Journal of Numerical Analysis 35*, 3 (2015), 1011–1046. 3, 8

[SvTSH15] Schulz C., von Tycowicz C., Seidel H.-P., Hildebrandt K.: Animating articulated characters using wiggly splines. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2015), pp. 101–109. 2

[TPBF87] Terzopoulos D., Platt J., Barr A., Fleischer K.: Elastically deformable models. *Computer Graphics 21*, 4 (1987), 205–214. 3

[TV12] Trouvé A., Vialard F.-X.: Shape splines and stochastic shape evolutions : A second order point of view. *Quartely of Applied Mathematics 70* (2012), 219–251. 2

[Wal04] Wallner J.: Existence of set-interpolating and energy-minimizing curves. *Comput. Aided Geom. Design 21*, 9 (2004), 883–892. 2

[WD05] Wallner J., Dyn N.: Convergence and C1 analysis of subdivision schemes on manifolds by proximity. *Comput. Aided Geom. Design 22* (2005), 593–622. 2

[WDAH10] Winkler T., Drieseberg J., Alexa M., Hormann K.: Multi-scale geometry interpolation. *Computer Graphics Forum 29(2)* (2010), 309 – 318. 2

[WK88] Witkin A., Kass M.: Spacetime constraints. *Computer Graphics 22*, 4 (1988), 159–168. 2

[WLT12] Wang Y., Liu B., Tong Y.: Linear surface reconstruction from discrete fundamental forms on triangle meshes. *Computer Graphics Forum 31*, 8 (2012), 2277–2287. 6

Appendix

Here we derive the partial derivatives $F^K_{,k}$ of (7) with respect to the primal variables $\mathbf{s}_k$ for $0<k<K$. In the sequel, comma subscripts denote differentiation with respect to the coordinate following the comma. Notice that we set $F^K_{,k}=0$ if the k^{th} variable corresponds to a fixed keyframe pose. In particular, in all our examples we have $F^K_{,0}=0$ and $F^K_{,K}=0$.

F^K depends on $\mathbf{s}_k$ and on the *constraint variables* $\tilde{\mathbf{s}}_k$, where $\tilde{\mathbf{s}}_k$ depends on $\mathbf{s}_{k-1}$ and $\mathbf{s}_{k+1}$ via (8). Hence a straightforward differentiation of F^K yields

$$F^K_{,k}[s_0,\dots,s_K]=\mathcal{W}_{,1}[\mathbf{s}_k,\tilde{\mathbf{s}}_k]+\mathcal{W}_{,2}[\mathbf{s}_{k-1},\tilde{\mathbf{s}}_{k-1}]\partial_{\mathbf{s}_k}\tilde{\mathbf{s}}_{k-1} + \mathcal{W}_{,2}[\mathbf{s}_{k+1},\tilde{\mathbf{s}}_{k+1}]\partial_{\mathbf{s}_k}\tilde{\mathbf{s}}_{k+1} \tag{13}$$

where $\partial_x f$ denotes the partial derivative of f with respect to variable x, e.g., $\partial_{\mathbf{s}_k}F^K=F^K_{,k}$. Furthermore we assume that $\mathcal{W}_{,j}$ is a row

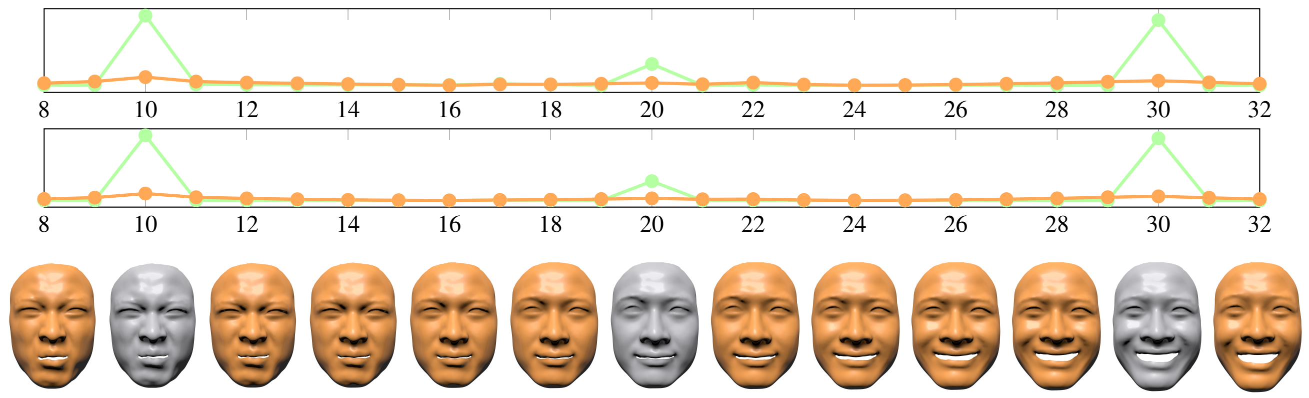

Figure 11: *Extract from a spline curve through five fixed keyframe poses (at $k = 0, 10, 20, 30, 40$, three of which are shown in gray) computed via the $L\Theta A$-scheme for $\eta = 1$, $K = 40$, and periodic boundary conditions. The energy plots show the $L\Theta A$-space energy $k \mapsto \widehat{\mathcal{W}}_{L\Theta A}[\mathbf{z}_k, \tilde{\mathbf{z}}_k]$ (top row) and the nonlinear energy $k \mapsto \mathcal{W}[\mathbf{s}_k, \tilde{\mathbf{s}}_k]$ (second row) for the piecewise geodesic (green) and the spline (orange) interpolation. The piecewise geodesic has pronounced energy concentrations at the keyframe poses.*

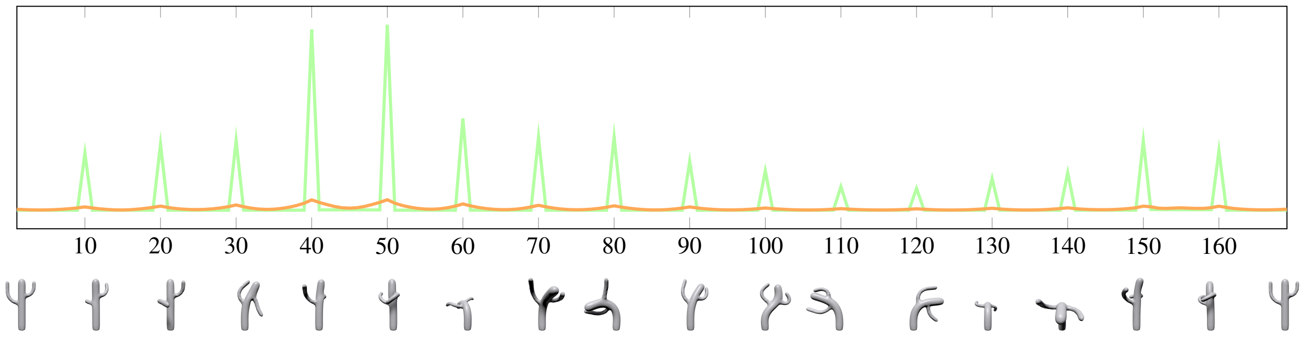

Figure 12: *Energy plot of the $L\Theta A$-space energy $k \mapsto \widehat{\mathcal{W}}_{L\Theta A}[\mathbf{z}_k, \tilde{\mathbf{z}}_k]$ for a spline (orange) and piecewise geodesic (green) interpolation between 17 fixed keyframe poses (gray), computed by $L\Theta A$-space approximation with $\eta = 10^{-3}$, $K = 170$, and natural boundary conditions.*

vector, i.e., $\mathcal{W}_{,j} \in \mathbb{R}^{1,3N}$ for $j = 1, 2$. In (13) we encounter matrix-valued inner derivatives $\partial_{\mathbf{s}_k} \tilde{\mathbf{s}}_j \in \mathbb{R}^{3N,3N}$, which are hard to evaluate. In the following we make use of a standard trick to get rid of these inner derivatives.

Differentiating the Euler–Lagrange equation to (8),

$$0 = \mathcal{W}_{,2}[\mathbf{s}_{j-1}, \tilde{\mathbf{s}}_j] + \mathcal{W}_{,1}[\tilde{\mathbf{s}}_j, \mathbf{s}_{j+1}],$$

wrt. $\mathbf{s}_i$, $i \in \{j-1, j+1\}$, yields the matrix-valued equations

$$0 = \mathcal{W}_{,21}[\mathbf{s}_{j-1}, \tilde{\mathbf{s}}_j] + A_j \partial_{\mathbf{s}_{j-1}} \tilde{\mathbf{s}}_j,$$
$$0 = \mathcal{W}_{,12}[\tilde{\mathbf{s}}_j, \mathbf{s}_{j+1}] + A_j \partial_{\mathbf{s}_{j+1}} \tilde{\mathbf{s}}_j,$$

where the symmetric matrix $A_j \in \mathbb{R}^{3N,3N}$ is given by

$$A_j = \mathcal{W}_{,22}[\mathbf{s}_{j-1}, \tilde{\mathbf{s}}_j] + \mathcal{W}_{,11}[\tilde{\mathbf{s}}_j, \mathbf{s}_{j+1}].$$

Rearranging and multiplying by some column vector $\mathbf{q} \in \mathbb{R}^{3N,1}$ gives

$$(\partial_{\mathbf{s}_{j-1}} \tilde{\mathbf{s}}_j)^T A_j \mathbf{q} = -\mathcal{W}_{,21}[\mathbf{s}_{j-1}, \tilde{\mathbf{s}}_j]^T \mathbf{q}, \tag{14}$$
$$(\partial_{\mathbf{s}_{j+1}} \tilde{\mathbf{s}}_j)^T A_j \mathbf{q} = -\mathcal{W}_{,12}[\tilde{\mathbf{s}}_j, \mathbf{s}_{j+1}]^T \mathbf{q}. \tag{15}$$

If we now define $\mathbf{p}_j \in \mathbb{R}^{3N,1}$ as the solution of the linear system

$$A_j \mathbf{p}_j = -\mathcal{W}_{,2}[\mathbf{s}_j, \tilde{\mathbf{s}}_j]^T, \tag{16}$$

we can plug this into (14) and (15), respectively, and get

$$(\partial_{\mathbf{s}_{j-1}} \tilde{\mathbf{s}}_j)^T \mathcal{W}_{,2}[\mathbf{s}_j, \tilde{\mathbf{s}}_j]^T = \mathcal{W}_{,21}[\mathbf{s}_{j-1}, \tilde{\mathbf{s}}_j]^T \mathbf{p}_j,$$
$$(\partial_{\mathbf{s}_{j+1}} \tilde{\mathbf{s}}_j)^T \mathcal{W}_{,2}[\mathbf{s}_j, \tilde{\mathbf{s}}_j]^T = \mathcal{W}_{,12}[\tilde{\mathbf{s}}_j, \mathbf{s}_{j+1}]^T \mathbf{p}_j.$$

Applying transposition and an index shift $j \mapsto k+1$ and $j \mapsto k-1$, respectively, we get

$$\mathcal{W}_{,2}[\mathbf{s}_{k+1}, \tilde{\mathbf{s}}_{k+1}] \, \partial_{\mathbf{s}_k} \tilde{\mathbf{s}}_{k+1} = \mathbf{p}_{k+1}^T \mathcal{W}_{,21}[\mathbf{s}_k, \tilde{\mathbf{s}}_{k+1}],$$
$$\mathcal{W}_{,2}[\mathbf{s}_{k-1}, \tilde{\mathbf{s}}_{k-1}] \, \partial_{\mathbf{s}_k} \tilde{\mathbf{s}}_{k-1} = \mathbf{p}_{k-1}^T \mathcal{W}_{,12}[\tilde{\mathbf{s}}_{k-1}, \mathbf{s}_k],$$

which can finally be inserted in (13) to obtain

$$F^K_{,k}[s_0, \dots, s_K] = \mathcal{W}_{,1}[\mathbf{s}_k, \tilde{\mathbf{s}}_k] + \mathbf{p}_{k-1}^T \mathcal{W}_{,12}[\tilde{\mathbf{s}}_{k-1}, \mathbf{s}_k] + \mathbf{p}_{k+1}^T \mathcal{W}_{,21}[\mathbf{s}_k, \tilde{\mathbf{s}}_{k+1}]. \tag{17}$$

Hence for every evaluation of the gradient $DF^K = (F^K_{,1}, \dots, F^K_{,K-1})$ one has to solve the $K-1$ nonlinear constraint equations (8) to get $\tilde{\mathbf{s}}_1, \dots, \mathbf{s}_{K-1}$ as well as $K-1$ linear equations (16) to get the dual variables $\mathbf{p}_1, \dots, \mathbf{p}_{K-1}$ ($\mathbf{p}_0 = \mathbf{p}_K = 0$ in (17)).