Energy-Minimizing Splines in Manifolds

Michael Hofer*
Vienna Univ. of Technology

Helmut Pottmann†
Vienna Univ. of Technology

Abstract

Variational interpolation in curved geometries has many applications, so there has always been demand for geometrically meaningful and efficiently computable *splines* in manifolds. We extend the definition of the familiar cubic spline curves and splines in tension, and we show how to compute these on parametric surfaces, level sets, triangle meshes, and point samples of surfaces. This list is more comprehensive than it looks, because it includes variational motion design for animation, and allows the treatment of obstacles via barrier surfaces. All these instances of the general concept are handled by the same geometric optimization algorithm, which minimizes an energy of curves on surfaces of arbitrary dimension and codimension.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

Keywords: variational curve design, splines in manifolds, geometric optimization, motion design, obstacle avoidance

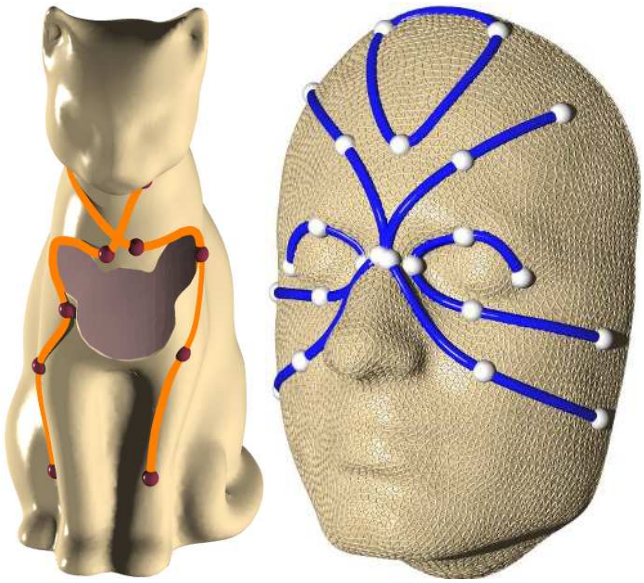

Figure 1: Energy-minimizing splines on triangle meshes with and without boundary.

1 Introduction

Computing energy minimizing curves which are restricted to a surface and which fulfill interpolation and/or approximation conditions is a basic problem of Geometric Computing. Its applications go far beyond the most obvious task of designing a curve on a surface in ordinary Euclidean space — there are many nonlinear geometric settings of higher dimension which have manifestations in the familiar case of three dimensions. An efficient method for variational curve design on surfaces, where neither the dimension of the surface, nor the dimension of ambient space are restricted to two or three, provides a basic tool usable e.g. for energy minimizing rigid body motions, or the handling of obstacles.

1.1 Previous Work

Spline curves in Euclidean spaces which minimize the L^2 norm of the second derivative and related functionals are well understood and comprise one of the basics of Geometric Modeling. For a comprehensive discussion of the theory and computational issues refer to [Hoschek and Lasser 1993]. The classical energy functionals are quadratic, which makes minimization easy. Geometric functionals which require nonlinear minimization techniques and their application to curve design have been studied e.g. by [Brunnett et al. 1993] and [Moreton and Sequin 1993]. Variational curve design has also been performed within the framework of subdivision [Kobbelt and Schröder 1998], an approach we only touch briefly in Remark 6.

Variational design of curves on surfaces has received much less attention. A substantial amount of research focuses on the quaternion 3-sphere because of its well-known relationship with rigid body motions [Barr et al. 1992; Brunnett and Crouch 1994; Jüttler and Wagner 2002; Park and Ravani 1997; Ramamoorthi and Barr 1997]. Minimizing the L^1 or L^2 norm of the *first* derivative of curves is a classical problem of Differential Geometry and leads to the geodesic lines of surfaces. Applications of this in Computer Vision and Image Processing are given in [Caselles et al. 1997; Khaneja et al. 1998; Malladi 2002; Memoli and Sapiro 2001; Sapiro 2001]. A series of contributions addresses intrinsically cubic splines in manifolds, i.e., the minimizers of the covariant acceleration [Camarinha et al. 2001; Gabriel and Kajiya 1985; Noakes et al. 1989; Noakes 2003].

In this paper we concentrate on an *extrinsic* formulation of energy-minimizing curves in embedded manifolds; its definition uses the ambient space. We minimize classical quadratic energy functionals involving first and second derivatives, but with the nonlinear side condition that the solution curves are confined to surfaces. For *parametric* surfaces in $\mathbb{R}^3$, this is the topic of H. Bohl's thesis [1999], who proves the existence of a solution and computes it by quasi-Newton iteration. Pottmann and Hofer [2003] characterize interpolating and approximating minimizers on surfaces of arbitrary finite dimension and codimension. The main results of that paper are briefly summarized in Section 2.

The optimization procedure used in this paper is related to research on active curves [Blake and Isard 1998; Kass et al. 1987], geometric flows of curves on surfaces [Cheng et al. 2002]), and snakes on surfaces [Lee and Lee 2002].

*e-mail: hofer@geometrie.tuwien.ac.at
†e-mail: pottmann@geometrie.tuwien.ac.at

1.2 Overview

We discuss computation and applications of splines on surfaces based on a variational principle, where the dimension of neither surface nor ambient space is restricted. By discretization the problem is reduced to minimizing a quadratic function in a so-called *constraint manifold*: A sequence $\mathbf{p}_1,\ldots,\mathbf{p}_M$ of points on a given surface S, with S contained in $\mathbb{R}^n$, is seen as one point $(\mathbf{p}_1,\ldots,\mathbf{p}_M)$, having $D = n \cdot M$ coordinates. The constraint manifold consists of these new points and is a surface in $\mathbb{R}^D$.

The minimization algorithm we propose has been designed having in mind the high dimensionality of the problem. It works by gathering as little discrete curvature information as possible and is applicable to all kinds of surface representations: the parametric form, implicit representations, a mesh, or a dense point cloud. These two properties make it widely applicable. We illustrate applications of the concept of energy-minimizing splines in Computer Graphics by means of selected applications.

This paper is organized as follows: After a review of known results in Section 2 and discussing computation in Section 3, we show applications to variational design and the smoothing of rigid body motions in Section 4. Section 5 shows how to compute energy minimizing splines in the presence of obstacles, which is done by creating a sequence of auxiliary barrier surfaces and working with those. The same principle applies to motion planning.

2 Characterization of Energy-Minimizing Splines in Manifolds

Let S be a k-dimensional regular surface, embedded in Euclidean $\mathbb{R}^n$, $k < n$. Moreover, a sequence of points $\mathbf{p}_i \in S$, $i = 1,\ldots,N$ and real numbers $u_1 < \cdots < u_N$ are given. We are seeking a curve $\mathbf{c}(u)$ on S, which interpolates the given data, $\mathbf{c}(u_i) = \mathbf{p}_i$, and minimizes some energy functional.

Surface Counterparts of Cubic Splines

Recall that a cubic C^2 spline curve arises as minimizer of the energy

$$E_2(\mathbf{x}) = \int_{u_1}^{u_N} \ddot{\mathbf{x}}(u)^2 du, \tag{1}$$

under interpolation conditions, $\mathbf{x}(u_i) = \mathbf{p}_i$. The case where the admissible curves $\mathbf{x}(u)$ are restricted to the given manifold S has recently been studied in [Pottmann and Hofer 2003]. We summarize the most important results, since they provide a basis for the numerical approach and give us further insight into certain applications discussed later.

When we speak of a curve segment in this section, we always mean its part between two interpolation points $\mathbf{p}_i, \mathbf{p}_{i+1}$; it is parameterized over the interval $[u_i, u_{i+1}]$. Moreover, the following assumptions are made: S is C^4 and *without boundary* (i.e., S is closed or extends to infinity; boundaries are discussed in Section 5). Admissible curves lie on S, have C^4 segments, satisfy the interpolation conditions $\mathbf{x}(u_i) = \mathbf{p}_i$, and are at least C^1 at the data points.

In [Pottmann and Hofer 2003] it has been proved that *among all admissible curves* $\mathbf{x}$*, a minimizer of the functional* E_2 *of Equ. (1) is* C^2 *and possesses segments whose fourth derivative vectors are orthogonal to S. Moreover, at the end points of the solution curve, the second derivative vector is orthogonal to S.*

This result refers to the case of natural end conditions. Other end conditions and closed curves require modifications analogous to spline theory [Hoschek and Lasser 1993].

The minimizers possess a characterization which is very similar to the familiar cubic C^2 splines: The fourth derivative of a cubic vanishes and so does the tangential component of the fourth derivative of spline segments in S (see Fig. 2). These segments cannot be computed explicitly in general, but only numerically, see Sect. 3. Existence of minimizers follows from results of [Bohl 1999].

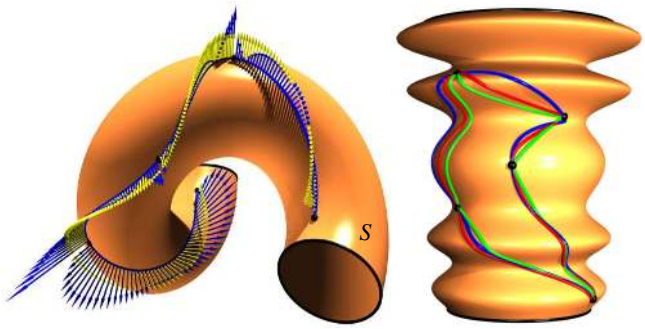

Figure 2: (Left) Counterpart to a C^2 cubic spline curve on a surface: the fourth derivative vectors (blue) are in direction of the surface normals (yellow). (Right) Curves minimizing E_2 (blue), E_t (red), and E_1 (green) interpolating the same points on a surface.

Geodesics

Let us prescribe just two points $\mathbf{p}_1, \mathbf{p}_2$ on a surface S and look for a curve segment on S, which joins $\mathbf{p}_1$ and $\mathbf{p}_2$ and minimizes the L^2 norm of the first derivative,

$$E_1(\mathbf{x}) = \int_{u_1}^{u_N} \dot{\mathbf{x}}(u)^2 du. \tag{2}$$

It is well known that the second derivative vectors $\ddot{\mathbf{c}}$ of the minimizer $\mathbf{c}$ are orthogonal to S and that $\mathbf{c}$ is a *geodesic* (shortest path on S) parameterized with constant speed. Note that minimizing the functional (1) also optimizes the parametrization. There are no simple global uniqueness results for geodesics, so we cannot expect such results for the more involved case of splines.

Splines in Tension in Manifolds

In the unrestricted (classical) case, minimizing a linear combination of energies (1) and (2),

$$E_t(\mathbf{x}) = \int_{u_1}^{u_N} (\ddot{\mathbf{x}}^2 + w\dot{\mathbf{x}}^2) du, \quad w = const > 0, \tag{3}$$

leads to the well known *splines in tension* [Schweikert 1966]. Their counterparts in manifolds have the property that the vector field $\mathbf{c}^{(4)}(u) - w\ddot{\mathbf{c}}(u)$ is orthogonal to S; otherwise, they are characterized in the same way as minimizers of E_2. Increasing w moves the spline closer to the curve which minimizes E_1, i.e., the curve composed by geodesic segments which connect the points $\mathbf{p}_i$. Hence, the parameter w controls the tension of the curve.

In [Pottmann and Hofer 2003], the authors provide analogous results for the surface counterparts of *smoothing splines*, which approximate the given data, or to *quintic splines* (minimizers of the L^2 norm of the third derivative). The algorithms below can handle these cases as well, but we will concentrate on cubic splines and splines in tension.

3 Computing Energy-Minimizing Splines in Manifolds

We are minimizing *quadratic functionals* such as E_2 or E_t. After discretization we obtain *quadratic functions*. The restriction of

the curves to a surface yields constraints, and thus the numerical computation of splines in manifolds requires an algorithm for the constrained minimization of a quadratic function. For this purpose we propose a projected gradient algorithm, whose stepsize control is guided by a geometric interpretation of an error estimate.

3.1 The Basic Geometric Optimization Algorithm

Consider minimization of a quadratic function

$$F : \mathbb{R}^D \to \mathbb{R}, \quad F(\mathbf{x}) = \mathbf{x}^T \cdot Q \cdot \mathbf{x} + 2\mathbf{q}^T \cdot \mathbf{x} + q,$$

with a symmetric positive definite matrix Q, under the constraint that $\mathbf{x}$ lies in some surface $\Phi \subset \mathbb{R}^D$. We assume that Φ has dimension m and that it is smooth in the area we work in.

The geometric approach to this minimization problem views the matrix Q as matrix of the inner product $\langle \mathbf{x}, \mathbf{y} \rangle := \mathbf{x}^T \cdot Q \cdot \mathbf{y}$, of a Euclidean metric in $\mathbb{R}^D$. F assumes its minimum in the point $\mathbf{p} = -Q^{-1} \cdot \mathbf{q}$. Up to an unimportant additive constant, F then equals

$$F(\mathbf{x}) = (\mathbf{x} - \mathbf{p})^T \cdot Q \cdot (\mathbf{x} - \mathbf{p}).$$

This is the squared distance $\|\mathbf{x} - \mathbf{p}\|^2$ of points $\mathbf{p}$ and $\mathbf{x}$ in the Euclidean norm mentioned above. Here, and in this entire section, 'distance', 'orthogonality', and related concepts refer to the metric defined by the matrix Q. The minimum of F on the surface Φ is attained at the point $\mathbf{p}^*$, which is closest to $\mathbf{p}$. $\mathbf{p}^*$ is the normal footpoint of $\mathbf{p}$ on Φ.

We propose the following iterative procedure for minimizing F. It consists of repeated application of the following two steps (see Fig. 3): $\mathbf{x}_c$ (the current point) is initialized with an initial guess $\mathbf{x}_0$ for the minimizer.

1. Compute the tangent space T^m of Φ at the current iterate $\mathbf{x}_c$ and project the point $\mathbf{p}$ orthogonally into T^m, which results in the point $\mathbf{p}_T$.

2. Compute an appropriate stepsize s and project $\mathbf{x}_s := \mathbf{x}_c + s(\mathbf{p}_T - \mathbf{x}_c)$ onto Φ; this yields the next iterate $\mathbf{x}_+$.

Convergence Analysis

The choice of the stepsize s requires a discussion of error estimates. We briefly summarize results here and omit proofs, which can be found in [Pottmann and Hofer 2004].

Under the assumption that Φ has derivatives up to third order, the current error $\mathbf{e}_c := \mathbf{x}_c - \mathbf{p}^*$ and the error at the next iteration step $\mathbf{e}_+ = \mathbf{x}_+ - \mathbf{p}^*$ are related via

$$\|\mathbf{e}_+\| \leq |1 - s + sd\kappa_{\max}| \, \|\mathbf{e}_c\| + C \, \|\mathbf{e}_c\|^2. \tag{4}$$

d equals the distance of $\mathbf{p}$ and $\mathbf{p}^*$. $\kappa_{\max}$ is the largest (in the sense of absolute values) normal curvature of the surface Φ in the point $\mathbf{p}^*$, with respect to the normal vector $\mathbf{n} := \mathbf{p} - \mathbf{p}^*$. C is a constant.

Since $\mathbf{p} - \mathbf{x}_c$ is the negative gradient of the squared distance function $\frac{1}{2}\|\mathbf{x} - \mathbf{p}\|^2$ at $\mathbf{x}_c$, the method is a projected gradient algorithm.

Remark 1. The normal curvature κ of a curve $\mathbf{l}$ in Φ through $\mathbf{p}^*$, with respect to the normal vector $\mathbf{n}$, has the following interpretation: Connecting $\mathbf{p}$ with $\mathbf{l}$ yields a cone (Fig. 3). By developing this cone into the plane, $\mathbf{l}$ is transformed into a planar curve $\widetilde{\mathbf{l}}$, whose (ordinary) curvature is precisely κ [do Carmo 1976; Spivak 1975].

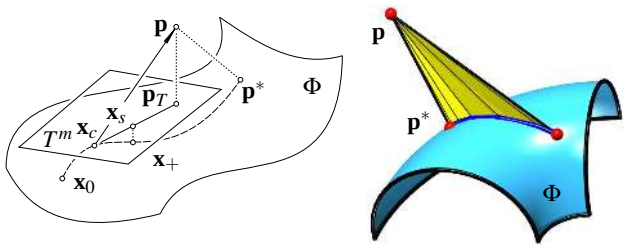

Figure 3: (Left) Footpoint computation with a projected gradient algorithm. (Right) Footpoint cone.

Remark 2. The projection $\mathbf{x}_s \mapsto \mathbf{x}_+$ is of marginal importance as regards local convergence. A sufficiently smooth projection only influences the constant factor C in Equ. (4). All projections discussed later meet this condition.

We proceed with the discussion of (4). Convergence of the algorithm depends on the constant

$$C_1 := |1 - s + sd\kappa_{\max}|.$$

We have linear convergence if $C_1 < 1$. The goal is a strategy for selecting the stepsize s such that C_1 becomes as small as possible.

The normal vector $\mathbf{n}$ has been chosen such that it points from $\mathbf{p}^*$ to $\mathbf{p}$. We have $d \geq 0$, but we cannot assume that $\kappa_{\max} \geq 0$. Two cases have to be discussed:

(a) $d\kappa_{\max} > 1$: Then $1 - s + sd\kappa_{\max} > 1$ and no choice of s would guarantee convergence. Fortunately, this case does not arise in our setting: it can be shown to occur only if $\mathbf{p}^*$ is a local minimizer, but not a global one.

(b) $d\kappa_{\max} < 1$: In this case any choice of s with

$$0 < s < \frac{2}{|d\ \kappa_{\max} - 1|} \tag{5}$$

gives a constant $C_1 < 1$, i.e., *linear convergence*.

Note that $d = 0$ together with $s = 1$ yields the optimal situation $C_1 = 0$, i.e., a *quadratically convergent algorithm*.

Stepsize Selection

To show the idea behind our selection of stepsize, we first look at the simple case that Φ is a *curve* with curvature $\kappa = \kappa_{\max}$ (in the sense of Remark 1) at the footpoint $\mathbf{p}^*$. If we choose

$$s = \frac{1}{|d\kappa - 1|}, \tag{6}$$

we get $C_1 = 0$ in (4), which means *quadratic convergence*.

In practice, however, we do not know d and κ. In the following we show how to extend this idea from the curve to the surface case, and how to estimate the unknown values d and κ from data collected at previous iteration steps.

Remark 3. Equation (6) becomes intuitively clear for a planar curve, which for purposes of second order analysis is replaced by its osculating circle of radius $1/\kappa$ at the footpoint (Fig. 4). The point $\mathbf{x}_s$ is chosen as close as possible to $\mathbf{p}^*$, but still on the tangent at $\mathbf{x}_c$. It is elementary to verify Equ. (6) for this choice of $\mathbf{x}_s$.

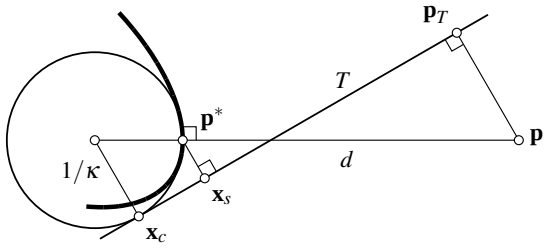

Figure 4: Visualization of stepsize selection.

In order to utilize the curve case for a general surface Φ, we consider a curve $\mathbf{c}_f$ on Φ, containing the individual iterates in our projected gradient algorithm. Computing $\mathbf{p}$'s footpoint in $\mathbf{c}_f$ yields the same point $\mathbf{p}^*$ as computing the footpoint on Φ. This means that we could just as well have applied the entire iteration to $\mathbf{c}_f$, which however is not known.

In view of Remark 1, we now estimate the stepsize s via developing the *footpoint cone* Γ_f, which connects $\mathbf{p}$ and $\mathbf{c}_f$. An approximation of this development in a neighborhood of the current iterate $\mathbf{x}_c$ can be computed from the mutual distances of the three points $\mathbf{x}_c$, the previous iterate $\mathbf{x}_-$, and $\mathbf{p}$.

Algorithm 1 *estimates the stepsize s via an approximate planar development of a part of the footpoint cone Γ_f (see Fig. 5):*

1. *Using the distances $\|\mathbf{p}_T - \mathbf{x}_c\|$, $\|\mathbf{p}_T - \mathbf{p}\|$, $\|\mathbf{x}_- - \mathbf{p}\|$, $\|\mathbf{x}_- - \mathbf{x}_c\|$ we develop the triangles $\mathbf{p}\mathbf{x}_c\mathbf{p}_T$ and $\mathbf{p}\mathbf{x}_c\mathbf{x}_-$ into a plane as shown by Fig. 5. Development is indicated by a tilde.*

2. *In the planar coordinate system of Fig. 5, let $\widetilde{\mathbf{x}}_-$ have coordinates (ξ,η). The circle k, which passes through $\widetilde{\mathbf{x}}_-$ and touches the line $\widetilde{\mathbf{x}}_c\widetilde{\mathbf{p}}_T$ at $\widetilde{\mathbf{x}}_c$, has center $\mathbf{m} = (0,\rho)$ and radius $\rho = (\xi^2+\eta^2)/(2\xi)$. We let $\kappa := 1/\rho$ and $d := |\,\|\widetilde{\mathbf{p}} - \mathbf{m}\| - |\rho|\,|$. We plug these values into (6) to get the stepsize s.*

Note that there are two ways to attach the triangles $\widetilde{\mathbf{p}}\widetilde{\mathbf{x}}_c\widetilde{\mathbf{p}}_T$ and $\widetilde{\mathbf{p}}\widetilde{\mathbf{x}}_c\widetilde{\mathbf{x}}_-$ to each other. One way is shown by Fig. 5. The other possibility is obtained by reflecting the first one in the line $\widetilde{\mathbf{p}}\widetilde{\mathbf{x}}_c$. Of course, we take the possibility which gives the smaller distortion of the spatial distance $\|\mathbf{p}_T - \mathbf{x}_-\|$.

Summary of the Optimization Algorithm

Algorithm 2 *computes the minimizer $\mathbf{p}^*$ of a quadratic function with positive definite matrix Q (and unconstrained minimizer $\mathbf{p}$) under the constraint that $\mathbf{p}^*$ lies on a given m-dimensional surface $\Phi \subset \mathbb{R}^D$: Starting with an initial guess $\mathbf{x}_c = \mathbf{x}_0$ for the minimizer, iteratively apply the following two steps.*

1. *Compute a basis $\{\mathbf{c}_1,\ldots,\mathbf{c}_m\}$ of Φ's tangent space at the current iterate $\mathbf{x}_c$, and the Gramian matrix $G = (g_{ij}) = (\langle \mathbf{c}_i, \mathbf{c}_j\rangle)$ of that basis with respect to the inner product $\langle \mathbf{x},\mathbf{y}\rangle := \mathbf{x}^T \cdot Q \cdot \mathbf{y}$. Further compute the vector $\mathbf{r} = (r_1,\ldots,r_m)$ with $r_j = \langle \mathbf{p} - \mathbf{x}_c, \mathbf{c}_j\rangle$. Define the tangent vector $\mathbf{t} = \sum_i v_i \mathbf{c}_i$ where $\mathbf{v} = (v_1,\ldots,v_m)$ is the solution of the linear system $G\cdot\mathbf{v} = \mathbf{r}$.*

2. *With distance computations based on the norm $\|\mathbf{x}-\mathbf{y}\|^2 = \langle \mathbf{x}-\mathbf{y}, \mathbf{x}-\mathbf{y}\rangle$ and the stepsize strategy of Algorithm 1, compute a stepsize s and the point $\mathbf{x}_s = \mathbf{x}_c + s\mathbf{t}$. Project $\mathbf{x}_s$ onto Φ to obtain the next iterate $\mathbf{x}_+$.*

The projection in step 2 depends on the chosen representation of Φ, and shall be discussed in connection with the individual applications.

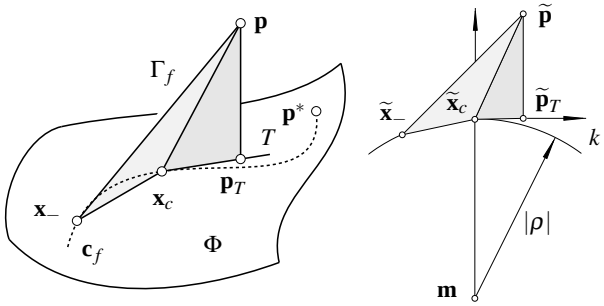

Figure 5: (Left.) Distances in $\mathbb{R}^D$. (Right.) Approximate planar development of a part of the footpoint cone.

In the very first iteration step, Algorithm 1 does not work and we must be content with a safely small stepsize s, whose validity can be tested with a standard strategy, e.g., the Armijo rule [Kelley 1999]. Such a strategy should be added to each step anyway, in particular if large curvature changes unbalance our stepsize selection. In the applications shown later in this paper, such changes have been detected by means of the footpoint cone: we observe the change of curvature as iteration progresses.

Later during iteration, points $\mathbf{x}_-$ and $\mathbf{x}_c$ will be too close to be used for curvature estimation; in such a case we revert to the last available valid estimate for κ.

The optimization algorithm presented here is very well suited for high dimensional applications, since it reduces curvature related computations (Algorithm 1) to the minimum which is necessary to achieve fast convergence.

Remark 4. The stepsize selection (5) can be seen as one step in a Newton iteration for computing $\mathbf{p}$'s footpoint on a circle, which approximates the footpoint curve $\mathbf{c}_f$ at $\mathbf{x}_c$. Other curves can be used for that purpose as well. E.g. a method proposed by [Hartmann 1999] for computing footpoints on parametric and implicit surfaces in $\mathbb{R}^3$ uses a certain parabola. The difference to our method is that the auxiliary curve used in each round of iteration does not use the approximate footpoints computed in previous rounds. Its extension to other surface representations is possible by using the ideas of the present paper. It turns out that the performance of Hartmann's method is similar to ours, but it is stable only for small d.

3.2 Splines in Manifolds

Computing energy minimizing curves on surfaces requires discretization. We describe a solution to the interpolation problem: The unknown curve $\mathbf{c}$ must pass through given points, so we assume that we are given parameter values $u_1,\ldots,u_N$ and points $\mathbf{p}_1,\ldots,\mathbf{p}_N$, such that $\mathbf{c}(u_i) = \mathbf{p}_i$. The curve itself is represented by a point sequence which contains the points $\mathbf{p}_i$ (cf. Fig. 6):

$$\mathbf{p}_1, \mathbf{q}_{1,1}, \mathbf{q}_{1,2}, \ldots, \mathbf{q}_{1,M_1}, \mathbf{p}_2, \mathbf{q}_{2,1}, \ldots, \mathbf{p}_N,$$

with M_i new points in between $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$. We assume that evaluating the unknown curve $\mathbf{c}$ at parameter values $u_i + j(u_{i+1} - u_i)/(M_i+1)$ yields $\mathbf{q}_{i,j}$. In our implementation, the parameter values u_i of the data points $\mathbf{p}_i$ are estimated such that Δu_i equals the arc length of the segment between $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ on the initial curve. In case of heavy changes of the length of certain segments during the optimization, we recompute the parameters.

The new points $\mathbf{q}_{i,j}$ are the unknowns in the minimization problem. Their number equals $M := M_1 + \ldots + M_{N-1}$. We rename them $(\mathbf{x}_1,\ldots,\mathbf{x}_M)$ in order to establish the connection with the previous section and collect them in a new point $X \in \mathbb{R}^{Mn}$. The constraint

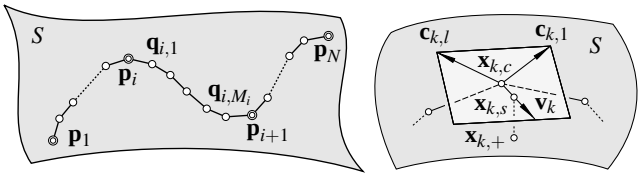

Figure 6: (Left.) Notation for the discretized spline. (Right.) Elementary view of an iteration step of the optimization algorithm.

manifold Φ is the set of X's, such that the single $\mathbf{x}_k$'s are contained in the given surface. It is a surface whose dimension equals M times the dimension of the original surface.

Our optimization algorithm as described in the previous section employs Φ's tangent spaces and orthogonal projection onto them. It is therefore necessary to give bases of these spaces. Suppose that for each point $\mathbf{x}_k$, the original surface's tangent space is spanned by vectors $\mathbf{c}_{k,l}$. Then the long vector $(0,\ldots,0,\mathbf{c}_{k,l},0,\ldots,0)$ with $\mathbf{c}_{k,l}$ in the place of $\mathbf{x}_k$ is tangent to Φ; and Φ's tangent space is spanned by those vectors, as k runs in $1,\ldots,M$ and l runs in $1,\ldots,n$.

The next step is to describe the quadratic function F. We use the difference of successive points as a discrete first derivative, and a difference of such differences as a discrete second derivative. By replacing integration by summation, any of the functionals E_1, E_2, E_t is thus converted into a quadratic function, and we directly apply Algorithm 2.

Note that this essentially means projecting gradients of an energy function, where the projection is defined via the orthogonality given by the energy itself. This is a discrete version of a *Sobolev gradient*. The efficiency of Sobolev gradient methods for geometric optimization problems has been pointed out by Renka and Neuberger (see [Neuberger 1997]).

An elementary view of the algorithm is the following: We displace the (non fixed) vertices $\mathbf{x}_{k,c}$ of the polygon to the current iterate in their respective tangent spaces of S, such that the displaced polygon minimizes the chosen energy E; then these displacement vectors are scaled by the stepsize s, and the resulting points $\mathbf{x}_{k,s}$ are projected to points $\mathbf{x}_{k,+}$ of S (see Fig. 6).

Among the many possible ways to compute an *initial guess* of the spline, we list the following two. Other choices will be mentioned later in connection with special representations of S or specific applications.

1. Compute the unrestricted energy minimizing spline through the given points $\mathbf{p}_i$ (with estimates for their parameter values u_i as suggested in the literature [Hoschek and Lasser 1993]) and project it onto the surface S. The projection has to be performed efficiently and therefore depends on the representation of S. This is discussed below.

2. Use an approximation to a piecewise geodesic curve.

The latter choice can be built into a *homotopy method*: For the minimization of E_2 or E_t (with weight w), we start with the minimization of E_t with high weight w', and during the iteration we perform an appropriate reduction of w' down to the desired value w ($w=0$ for E_2).

Remark 5. The algorithm can be used also for *smoothing* or *fairing* of a given curve $\mathbf{c}^0$ (see Fig. 7): We fix some points of it, define $\mathbf{c}^0$ as initial position and run a few (small) steps in the optimization. Of course, this works for any representation of S and any of the applications discussed below (see Fig. 11).

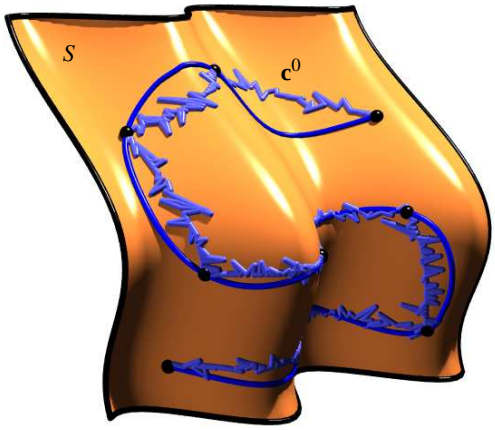

Figure 7: Smoothing curves on surfaces.

Remark 6. Inserting only *one* point $\mathbf{q}_i \in S$ between two consecutive points $\mathbf{p}_i, \mathbf{p}_{i+1}$ can be seen as one round of *variational curve subdivision* in a manifold. This is a generalization of [Kobbelt and Schröder 1998]. The presented version improves and stabilizes the algorithm in [Hofer et al. 2003] by controlling the stepsize factor s.

Algorithm 2 is sufficiently general so as to work for any representation of the surface S. In the following, we will restrict ourselves to surfaces in $\mathbb{R}^3$, and discuss the computation of splines for various representations of S.

Parameterized Surfaces

If S is given in a parametric representation $\mathbf{s}(r_1,r_2)$, the bases $\mathbf{c}_{i,j}$ in the tangent spaces of S at $\mathbf{x}_i$ can be taken as the first partial derivatives $\mathbf{s}_{,1}$, $\mathbf{s}_{,2}$ at the point $\mathbf{x}_i$ with respect to the surface parameters.

An initial position of the curve to be optimized can be chosen by means of the surface parameterization, for example by choosing $\mathbf{s}(r_1(t),r_2(t))$ with piecewise linear functions $r_i(t)$. Throughout the computation, it is important to respect periodicity conditions in case of closed surfaces.

The projection onto the constraint manifold Φ is done via linearization of the parametric representation. Let $\mathbf{r}_{i,c}$ be the vectors of parameter values of the polygon vertices $\mathbf{x}_{i,c}$ in the current iterate. Then, the step vector $s\mathbf{t}$ of Algorithm 2 is composed of a sequence of tangential displacement vectors $\mathbf{v}_i = v_{i,1}\mathbf{s}_{,1} + v_{i,2}\mathbf{s}_{,2}$ of the vertices $\mathbf{x}_{i,c}$. Now, the parameter vectors of the next iterates $\mathbf{x}_{i,+}$ are taken to be $\mathbf{r}_{i,+} = \mathbf{r}_{i,c} + (v_{i,1}, v_{i,2})$. Thus, the projection is actually not present, in accordance with the fact that the optimization problem may be seen as an unconstrained problem over the parameter domain. Examples are shown by Fig. 2.

Implicit Surfaces

Implicitly defined surfaces play a prominent role in Computer Graphics, and even more so since a variety of simulation and shape optimization problems can be solved efficiently with the level set method [Osher and Fedkiw 2002].

For an implicitly defined surface $F(\mathbf{x})=0$, the tangent space of S in the point $\mathbf{x}_i$ is spanned by two independent vectors $\mathbf{c}_{i,1}, \mathbf{c}_{i,2}$, which are orthogonal to the gradient $\nabla F(\mathbf{x}_i)$ at $\mathbf{x}_i$.

The projection of a displaced vertex $\mathbf{x}_{i,s}$ onto S requires a few steps of a Newton iteration for the search of a zero of F along the line $\mathbf{x}_{i,s} + \lambda \nabla F(\mathbf{x}_i)$.

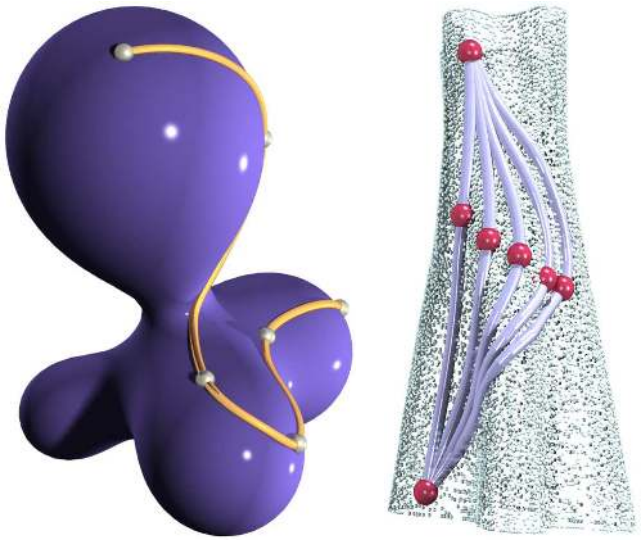

Figure 8: Energy minimizing splines on (left) an implicit surface, and (right) a point cloud.

Triangle Meshes and Point Cloud Surfaces

Our algorithm is also applicable for curve design on a triangle mesh or even a point cloud (see Fig. 8). In view of the energies we use for curve design, the mesh or point cloud should be representing a smooth surface. [Adamson and Alexa 2003; Alexa et al. 2003] use a moving least squares method to associate a smooth surface with a given point cloud. This surface is locally represented in implicit form. Its smoothness depends on the smoothness of the weight function used by that method. Thus, point clouds appear as a special case of implicitly defined surfaces. Noisy triangle meshes can be smoothed in the same way. In both cases, for the actual implementation we need to compute tangent planes and – for the projection – intersection points with a straight line. For smoothed meshes this is standard. For computing the tangent plane of a point cloud or a noisy mesh, we use the associated surface. Intersection with and projections onto point clouds are treated in [Adamson and Alexa 2003; Alexa et al. 2003; Levin 2004; Pauly et al. 2003].

Currently the algorithm does not handle spline curves which go over sharp edges, unless one of the given interpolation points is on that edge. One way to handle edges would be to break spline curves on edges.

4 Variational Design of Rigid Body Motions

The design of smooth motions of a rigid body in $\mathbb{R}^3$ is one of the most prominent applications of splines in manifolds. Unlike most contributions to motion design, we do not use the quaternion unit sphere as a model of SO(3), but consider the group of Euclidean congruence transformations as a surface, following [Belta and Kumar 2002] and [Hofer et al. 2003].

4.1 The Group of Euclidean Motions Embedded in the Affine Group

Consider a rigid body moving in $\mathbb{R}^3$. We use Cartesian coordinates and denote points of the moving system Σ^0 by $\mathbf{x}^0, \mathbf{y}^0, \ldots$, and points of the fixed system by $\mathbf{x}, \mathbf{y}$, and so on. A rigid body transformation α maps points $\mathbf{x}^0 \in \Sigma^0$ to positions $\mathbf{x}$ in the fixed system via

$$\mathbf{x} = \mathbf{a}_0 + A \cdot \mathbf{x}^0. \tag{7}$$

We speak of the image also as 'position' Σ; it is determined by the pair $(\mathbf{a}_0, A)$, consisting of a vector $\mathbf{a}_0$ (the position of the origin) and the rotation matrix A. If $\mathbf{a}_0(u)$ and $A(u)$ depend smoothly on u, which can be thought of as time, we speak of a smooth motion, or sometimes simply of a motion. Our goal is the design of motions which interpolate given positions $\Sigma(u_i)$ at time instances u_i.

If we do not impose any restriction on the matrix A in (7), we get an *affine map* $\alpha(u)$ and an *affine position* $\Sigma(u)$. Let us denote the three column vectors of A as $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$. Now we associate with the affine map α a *point in 12-dimensional affine space* $\mathbb{R}^{12}$, represented by the vector $\mathbf{A} = (\mathbf{a}_0, \ldots, \mathbf{a}_3)$. Because of the orthogonality condition imposed on A, the image points of rigid body transformations α lie in a 6-dimensional manifold $M^6 \subset \mathbb{R}^{12}$.

A meaningful *metric* in $\mathbb{R}^{12}$ can be introduced by means of a collection X of points $\mathbf{x}_1^0, \mathbf{x}_2^0, \ldots, \mathbf{x}_K^0$ in the moving system (body), which are called *feature points*. The squared distance $\|\alpha - \beta\|^2$ of affine mappings α and β from each other is defined as sum of squared distances $\sum_i \|\alpha(\mathbf{x}_i^0) - \beta(\mathbf{x}_i^0)\|^2$ of the corresponding feature point positions. One does not have to use unit point masses at a discrete number of feature points. Instead, we could work with another mass distribution (positive measure) on the moving body.

It can be shown (see e.g. [Hofer et al. 2003]) that this distance measure introduces a Euclidean metric in $\mathbb{R}^{12}$, which only depends on the barycenter $\mathbf{s}_x = (1/K)\sum_i \mathbf{x}_i^0$ and on the inertia matrix

$$J := \sum_i \mathbf{x}_i^0 \cdot \mathbf{x}_i^{0\,T} \tag{8}$$

of the feature points. By a well-known result from mechanics, we can replace the set of feature points by the six vertices of their inertia ellipsoid, without changing the barycenter and the inertia matrix of X. To do so, we choose the barycenter as the origin in the moving system and the eigenvectors of J as coordinate axes. Then the six points have coordinates $(\pm f_1, 0, 0), (0, \pm f_2, 0), (0, 0, \pm f_3)$, where $2f_i^2$ are the eigenvalues of J. Now, $\|\alpha - \beta\|^2$, i.e., the above defined squared distance of the points $\mathbf{A} = (\mathbf{a}_0, \ldots, \mathbf{a}_3)$ and $\mathbf{B} = (\mathbf{b}_0, \ldots, \mathbf{b}_3)$ from each other is given by the formula

$$\|\mathbf{A} - \mathbf{B}\|^2 = 6(\mathbf{a}_0 - \mathbf{b}_0)^2 + 2\sum_{i=1}^{3} f_i^2 (\mathbf{a}_i - \mathbf{b}_i)^2. \tag{9}$$

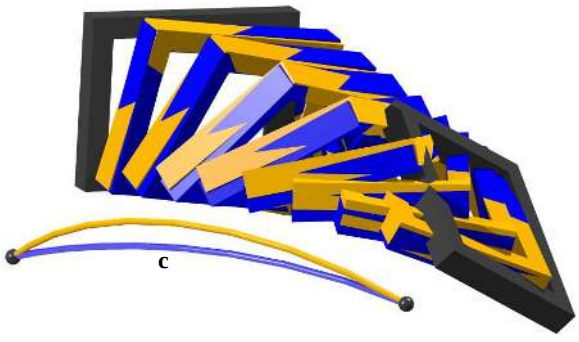

Figure 9: Comparing shortest motions between two positions in the sense of the metric (9) (blue) and in the quaternion sense (orange). The former is the free motion of a rigid body and tends to generate shorter paths $\mathbf{c}$ for points close to the moving object.

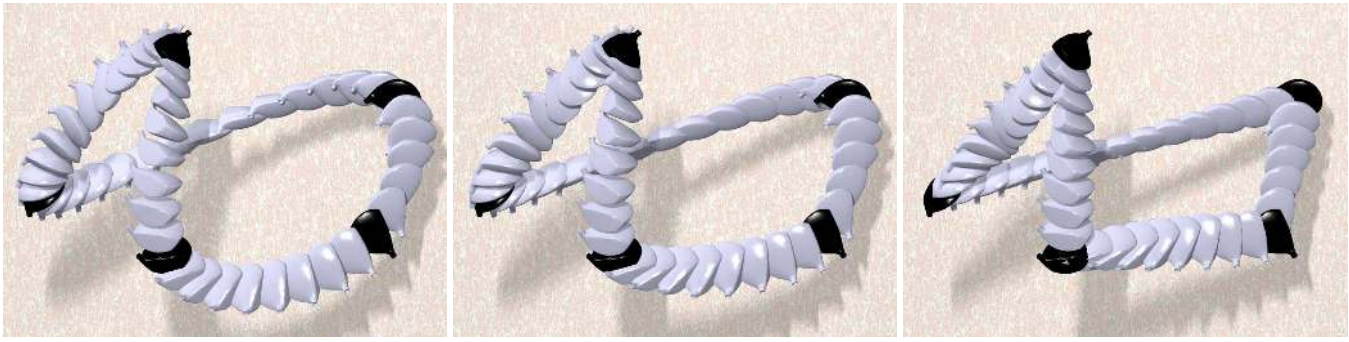

Figure 10: Energy minimizing cyclic motions: (Left) E_2, (Center) E_t, (Right) E_1.

Figure 11: (Top) Input motion and (Bottom) smoothed version, obtained through application of a few steps of the algorithm for minimization of E_2.

4.2 Computation of Energy-Minimizing Motions

Variational motion design is seen as curve design with energy minimizing splines on $M^6 \subset \mathbb{R}^{12}$, using the metric (9). For motions, the meaning of minimizing one of the functionals E_1, E_2, or E_t, is that the total energy of the feature point trajectories is minimized. It makes sense to base motion design on trajectories of points on the moving body. We view this as an advantage over the known purely intrinsic formulations, which are neglecting shape and mass properties of the moving body (see also Fig. 9).

Recall that the numerical computation of energy minimizing splines is based on a discretization. This means that the final motion is discretely resolved by a certain number of positions, including the input positions. The optimization algorithm in each step computes a tangent vector $s\mathbf{t}$ and requires projecting $\mathbf{x}_s = \mathbf{x}_c + s\mathbf{t}$ onto Φ. Actually both $\mathbf{x}_c$ and $\mathbf{x}_s$ are sequences of positions, and the tangent vector $\mathbf{t}$ is a sequence of tangent vectors to M^6. We project the sequence $\mathbf{x}_s$ by projecting each single position orthogonally onto M^6. This is a known algebraic problem of degree four. For details, see e.g. [Belta and Kumar 2002; Horn 1987].

For a geometric characterization of the motions which are computed in this way, we use the concept of a balanced force system from statics. A system of forces $\mathbf{f}_i$, attached to points $\mathbf{p}_i$, is in balance, if both, the sum of force vectors and the sum of moment vectors, vanish, i.e., $\sum_i \mathbf{f}_i = 0$, and $\sum_i \mathbf{p}_i \times \mathbf{f}_i = 0$. At an arbitrary time instant u of a sufficiently smooth motion, the k-th derivative vectors at the feature point positions define the k-th derivative force system $\mathbf{S}_k$. Now motions arising from the minimization of E_2 are characterized as follows: *The energy minimizing spline motion is C^2, at each time instant $u \neq u_i$ the 4-th derivative force system $\mathbf{S}_4(u)$ is in balance, and at the end positions, the systems $\mathbf{S}_2(u_1), \mathbf{S}_2(u_N)$ of second derivatives are in balance. In particular, the trajectory of the barycenter of the feature points is an interpolating cubic C^2 spline.* For a proof, one uses the results of [Pottmann and Hofer 2003] and shows that the k-th derivative vector of a curve $\mathbf{C}$ on M^6 at a point $\mathbf{C}(u)$ is orthogonal to M^6 if the k-th derivative force system of the corresponding position in $\mathbb{R}^3$ is in balance. The spline property of the barycenter trajectory follows immediately from (9) and the definition of E_2; therefore, the motion computation described above in $\mathbb{R}^{12}$ can be decomposed into the computation of this special trajectory (in $\mathbb{R}^3$) and the computation of the rotational part (in $\mathbb{R}^9$).

Analogous results hold for the other spline functionals. The case of E_1 belongs to geodesics on M^6. The corresponding motions have a balanced force system $\mathbf{S}_2(u)$ of second derivatives. The trajectory of the barycenter of the feature point set is a straight line traced with constant speed. These motions are well-known in mechanics as free motions of a body [Arnol'd 1989], see Fig. 9.

5 Splines in the Presence of Obstacles

So far, we did not mention curve design on surfaces with boundaries, such as surface patches in CAD applications. In this section we consider an even more general problem, namely variational design of curves which avoid a finite number of obstacles O_i.

Although this problem has a variety of practical applications, there are not many contributions dealing with it. We point to work on interpolation with cubic spline functions under linear inequality

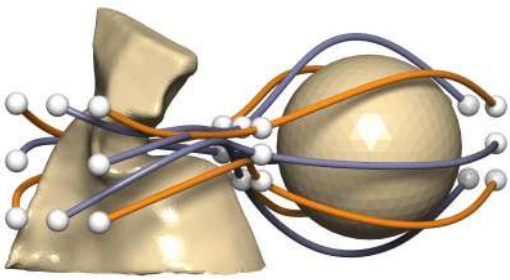

Figure 12: Variational curve design in the presence of obstacles.

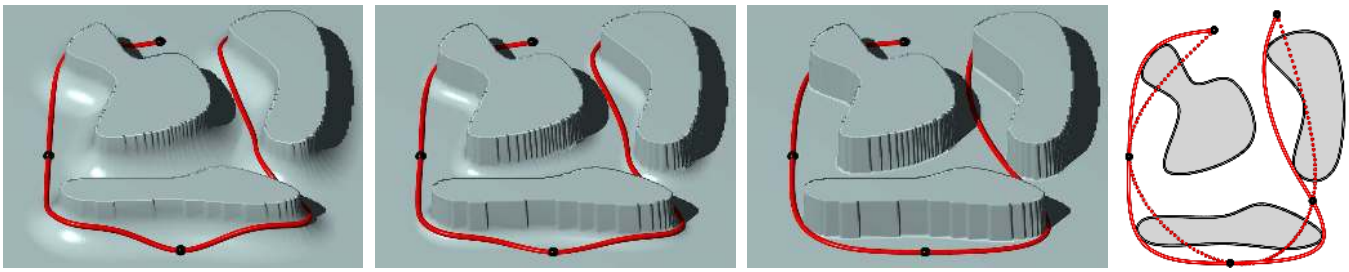

Figure 13: Spline computation in the presence of obstacles: Barrier surface with decreasing rounding radius during the iteration, and resulting planar spline curve.

constraints [Opfer and Oberle 1988], CAD related work on constrained curve design without energy minimization [Meek et al. 2003], and in particular to Bohl's contribution to splines on parametric surfaces [1999]. We will see that we can get rid of the obstacle constraints by introducing an appropriate unbounded auxiliary surface M and working with that.

Let us start with curve design in the plane $\mathbb{R}^2$. As a geometric interpretation of the *barrier method* from constrained optimization [Fletcher 1987], we embed $\mathbb{R}^2$ as the plane $z = 0$ into $\mathbb{R}^3$ and remove the interior of each obstacle O_i. We now attach, to each obstacle boundary $\mathbf{b}_i$, a cylindrical surface with z-parallel rulings and smooth out the edge along $\mathbf{b}_i$ which is generated by this procedure. This results in a smooth surface M. Given input points $\mathbf{p}_i$, which do not lie in any obstacle, are lifted onto M in z-direction; only points near obstacle boundaries will be changed by this lifting.

Now we design an interpolating curve $\bar{\mathbf{c}}$ on M; its projection onto $z = 0$ is the desired curve. During the optimization, M is not kept fixed. The blending radius is reduced so that it tends to zero, see Fig. 13. Therefore, the final curve $\mathbf{c}$ is a minimizer of the chosen energy under the given constraints. The general results of Sect. 3 imply that $\mathbf{c}$ *is* C^2 *and piecewise cubic where it does not lie in a boundary curve. There can be parts of* $\mathbf{c}$ *along a boundary curve* $\mathbf{b}_i$*; the parametrization of* $\mathbf{c}$ *along* $\mathbf{b}_i$ *has its 4th derivative vectors orthogonal to* $\mathbf{b}_i$.

In view of the obvious extension to $\mathbb{R}^3$, we construct the blending areas of M with help of distance fields to the obstacles. We use an algorithm from [Tsai 2002] for distance field computation, since it stores the normal footpoints; this is helpful for the projection onto M. Let $d_i(x,y)$ be the signed distance of the point (x,y) from the obstacle O_i. Then, the corresponding blending surface is given implicitly by the equation $f(d_i(x,y),z) = 0$, where f describes the shape of the blend profile (Fig. 13).

Note that the initial curve already determines the *combinatorial type* of the final spline curve. In our current implementation we use as initial curve a curve composed of geodesics (in the presence of obstacles) between consecutive points $\mathbf{p}_i, \mathbf{p}_{i+1}$. These are computed by propagating the distance field originating in $\mathbf{p}_i$ within M. As soon as the propagating wave reaches $\mathbf{p}_{i+1}$ and $\mathbf{p}_{i-1}$, we trace back with a gradient descent method, see Fig. 14. The distance propagation is done with an adapted version of Zhao's sweeping algorithm [Zhao 2004], where grid nodes inside obstacles get a flag and are not used for the propagation.

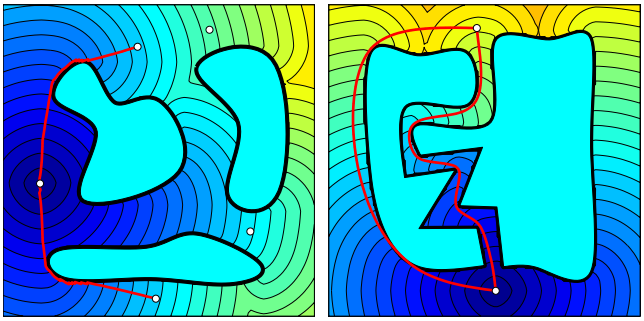

Figure 14: (Left) Computing the initial position via backward gradient flow on the distance field in the presence of obstacles. (Right) The path through the tunnel is shorter but has higher energy.

The spline through the channel shown by Fig. 14, right, also shows that corners do not cause problems: at a convex corner, the blend is smooth; at a concave corner, a blend as defined above would have a sharp edge, but a smooth spline never reaches a concave corner. Moreover, this figure also illustrates a second, combinatorially different solution; it is longer than the path through the channel, but has a smaller energy E_2.

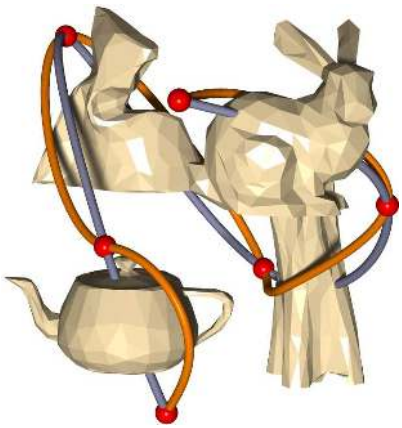

Figure 15: One curve avoids the solids.

Figures 12 and 15 show 3D examples of spline curves in the presence of obstacles. Figure 16 presents an example for variational motion design in the presence of obstacles, where we have combined the methodologies from Sect. 4 and 5: the path of the barycenter $\mathbf{s}_x$ of the moving body is a spline, which does not come closer to the obstacles than a distance r; here r is the maximum distance of points on the moving body to $\mathbf{s}_x$. This method guarantees a smooth motion and avoids the obstacles in a conservative way. Consideration of the precise shape of the moving object can in principle be done with our algorithm using a configuration space approach. However, the efficient computation of the corresponding barrier surface remains a challenging problem for future research.

Fig. 1, left, shows a spline on a surface with boundary. Here, the obstacle is the set of all points in $\mathbb{R}^3$, whose distance from the surface exceeds ε. The spline is designed within the remaining thin layer, and is finally projected onto the surface. This method is sim-

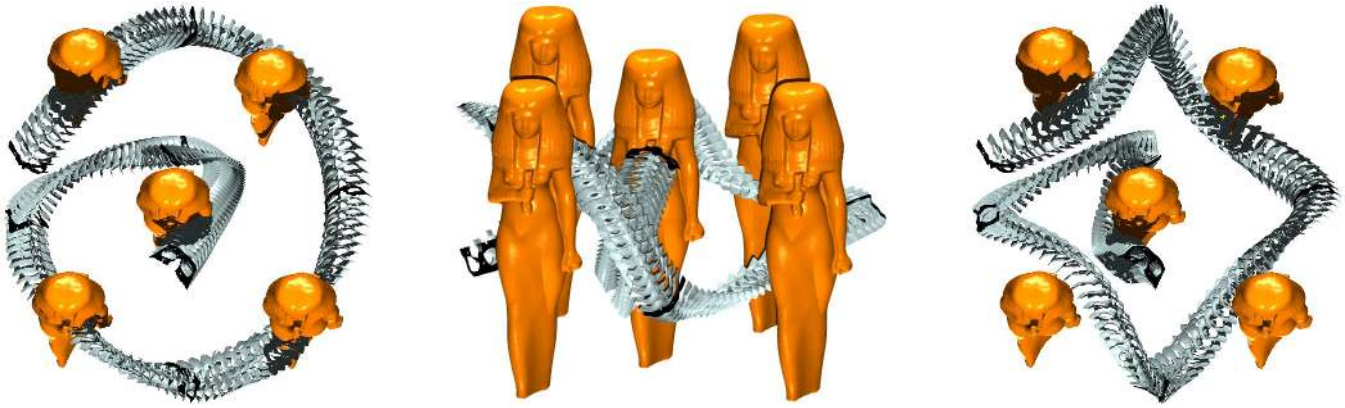

Figure 16: Motion design in the presence of obstacles: Motion minimizing E_2 (Left) unconstrained (Center, Right) avoiding 5 obstacles.

ilar to the computation of geodesics in [Memoli and Sapiro 2001]. It is independent of the type of surface representation we use, since for all of them distance fields can be computed efficiently.

Performance

We implemented our algorithms in *Matlab* and tested them on a PC with 1.8GHz. All spline curves and motions computed in this paper were computed within 0.5–5 seconds (not counting distance field computation). The table below shows computation time t per position for variational motion design. The case $M = 45$ (M is the number of optimized positions of the discretized motion), corresponds to Fig. 10.

M	45	94	149	191	247	288
t	.0016s	.0018s	.0025s	.0032s	.0042s	.005s

6 Conclusion

We have discussed the computation of energy minimizing spline curves in a k-dimensional manifold based on a geometrically motivated algorithm for constrained minimization of a quadratic function. Splines in manifolds possess a variety of interesting applications in Computer Graphics and related areas. This has been illustrated by means of computing splines on various surface representations, by motion design, and by spline computation in the presence of obstacles.

The essential component of the computational approach is not restricted to curves; its use for the generation of spline surfaces in manifolds seems straightforward. This is one of the many possible extensions of our results.

The concept of *image manifold* [Kimmel et al. 2000] makes it possible to develop image sensitive drawing tools. Such an approach includes images defined on surfaces. An example where not color or texture information, but the surface normals comprise the image, is given by Fig. 17, where design in the image manifold means feature-sensitive design in the original surface. This type of image manifold has been employed in [Pottmann et al. 2004] for the development of feature sensitive morphological operators on surfaces.

Acknowledgements

This research was supported by the Austrian Science Fund (FWF) under grant P16002-N05 and in part by the innovative project '3D Technology' of Vienna University of Technology. The golf club model (Fig. 10), and the Isis model (Fig. 16) are courtesy of *Cyberware*. The teapot model and the bunny model (Fig. 15) have been created by Martin Newell and the Stanford Computer Graphics Laboratory, respectively. The authors would like to thank the anonymous reviewers for their valuable comments. We gratefully acknowledge the enduring support of Johannes Wallner.

References

ADAMSON, A., AND ALEXA, M. 2003. Approximating and intersecting surfaces from points. In *Geometry Processing 2003*, Eurographics, 245–254.

ALEXA, M., BEHR, J., COHEN, D., FLEISHMAN, S., LEVIN, D., AND SILVA, C. T. 2003. Computing and rendering point set surfaces. *IEEE Trans. Visualization Comp. Graph. 9*, 1, 3–15.

ARNOL'D, V. I. 1989. *Mathematical Methods of Classical Mechanics*. Springer.

BARR, A. H., CURRIN, B., GABRIEL, S., AND HUGHES, J. F. 1992. Smooth interpolation of orientations with angular velocity constraints using quaternions. *Computer Graphics (Proceedings of ACM SIGGRAPH 92), 26*, 2, 313–320.

BELTA, C., AND KUMAR, V. 2002. An SVD-projection method for interpolation on SE(3). *IEEE Trans. Robotics Automation 18*, 3, 334–345.

BLAKE, A., AND ISARD, M. 1998. *Active Contours*. Springer.

BOHL, H. 1999. *Kurven minimaler Energie auf getrimmten Flächen*. PhD thesis, Univ. Stuttgart.

BRUNNETT, G., AND CROUCH, P. E. 1994. Elastic curves on the sphere. *Adv. Comput. Math. 2*, 23–40.

BRUNNETT, G., HAGEN, H., AND SANTARELLI, P. 1993. Variational design of curves and surfaces. *Surveys Math. Ind. 3*, 1–27.

CAMARINHA, M., LEITE, F. S., AND CROUCH, P. E. 2001. On the geometry of Riemannian cubic polynomials. *Diff. Geom. Applic. 15*, 107–135.

CASELLES, V., KIMMEL, R., AND SAPIRO, G. 1997. Geodesic active contours. *Int. J. Comp. Vision 22*, 61–79.

CHENG, B. T., BURCHARD, P., MERRIMAN, B., AND OSHER, S. 2002. Motion of curves constrained on surfaces using a level set approach. *Journal of Computational Physics 175*, 2, 604–644.

DO CARMO, M. P. 1976. *Differential Geometry of Curves and Surfaces*. Prentice-Hall.

FLETCHER, R. 1987. *Practical Methods of Optimization*. Wiley.

GABRIEL, S., AND KAJIYA, J. 1985. Spline interpolation in curved space. In *ACM SIGGRAPH 85 course notes for State of the Art Image Synthesis*, ACM Press / ACM SIGGRAPH.

HARTMANN, E. 1999. On the curvature of curves and surfaces defined by normalforms. *Comp. Aided Geom. Des. 16*, 355–376.

HOFER, M., POTTMANN, H., AND RAVANI, B. 2003. Geometric design of motions constrained by a contacting surface pair. *Comp. Aided Geom. Des. 20*, 8–9, 523–547.

HORN, B. K. P. 1987. Closed form solution of absolute orientation using unit quaternions. *J. Optical Soc. A 4*, 629–642.

HOSCHEK, J., AND LASSER, D. 1993. *Fundamentals of Computer Aided Geometric Design*. A. K. Peters.

JÜTTLER, B., AND WAGNER, M. 2002. Kinematics and animation. In *Handbook of Computer Aided Geometric Design*, G. Farin, M. S. Kim, and J. Hoschek, Eds. Elsevier, 723–748.

KASS, M., WITKIN, A., AND TERZOPOULOS, D. 1987. Snakes: Active contour models. *Intl. J. of Comp. Vision 1*, 321–331.

KELLEY, C. T. 1999. *Iterative Methods for Optimization*. SIAM.

KHANEJA, N., MILLER, M. I., AND GRENANDER, U. 1998. Dynamic programming generation of curves on brain surfaces. *IEEE PAMI 20*, 11, 1260–1265.

KIMMEL, R., MALLADI, R., AND SOCHEN, N. 2000. Images as embedded maps and minimal surfaces: movies, color, texture, and volumetric medical images. *Int. J. of Comp. Vision 39*, 111–129.

KOBBELT, L., AND SCHRÖDER, P. 1998. A multiresolution framework for variational subdivision. *ACM Transactions on Graphics 17*, 4, 209–237.

LEE, Y., AND LEE, S. 2002. Geometric snakes for triangular meshes. *Computer Graphics Forum 21*, 3, 229–238.

LEVIN, D. 2004. Mesh-independent surface interpolation. In *Geometric Modeling for Scientific Visualization*, G. Brunnett, B. Hamann, H. Müller, and L. Linsen, Eds. Springer, 37–50.

MALLADI, R. 2002. *Geometric Methods in Bio-Medical Image Processing*. Springer.

MEEK, D. S., ONG, B. H., AND WALTON, D. J. 2003. Constrained interpolation with rational cubics. *Comp. Aided Geom. Design 20*, 5, 253–275.

MEMOLI, F., AND SAPIRO, G. 2001. Fast computation of weighted distance functions and geodesics on implicit hypersurfaces. *J. Comput. Phys. 173*, 2, 730–764.

MORETON, H., AND SEQUIN, C. 1993. Scale invariant minimum cost curves: fair and robust design implements. *Computer Graphics Forum 12*, 473–484.

NEUBERGER, J. W. 1997. *Sobolev Gradients and Differential Equations*. Springer.

NOAKES, L., HEINZINGER, G., AND PADEN, B. 1989. Cubic splines on curved spaces. *IMA J. Math. Cont.& Inf. 6*, 465–473.

NOAKES, L. 2003. Null cubics and Lie quadratics. *J. Math. Physics 44*, 3, 1436–1448.

OPFER, G., AND OBERLE, H. J. 1988. The derivation of cubic splines with obstacles by methods of optimization and optimal control. *Numer. Math. 52*, 17–31.

OSHER, S., AND FEDKIW, R. 2002. *The Level Set Method and Dynamic Implicit Surfaces*. Springer.

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

PAULY, M., KEISER, R., KOBBELT, L., AND GROSS, M. 2003. Shape modeling with point-sampled geometry. *ACM Transactions on Graphics 22*, 3, 641–650.

POTTMANN, H., AND HOFER, M. 2003. A variational approach to spline curves on surfaces. Tech. Rep. 115, TU Wien, Inst. Geometry, *http://www.geometrie.tuwien.ac.at/ig/papers/vscs.pdf*.

POTTMANN, H., AND HOFER, M. 2004. Algorithms for constrained minimization of quadratic functions – geometry and convergence analysis. Tech. Rep. 121, TU Wien, Geometry Preprint. *http://www.geometrie.tuwien.ac.at/ig/papers/foot.pdf*.

POTTMANN, H., STEINER, T., HOFER, M., HAIDER, C., AND HANBURY, A. 2004. The isophotic metric and its application to feature sensitive morphology on surfaces. In *Proceedings of ECCV 2004*. Springer.

RAMAMOORTHI, R., AND BARR, A. 1997. Fast construction of accurate quaternion splines. In *Proceedings of ACM SIGGRAPH 1997*, ACM Press / ACM SIGGRAPH, New York, ACM, 287–292.

SAPIRO, G. 2001. *Geometric Partial Differential Equations and Image Analysis*. Cambridge Univ. Press.

SCHWEIKERT, D. 1966. An interpolating curve using a spline in tension. *J. Math. Phys. 45*, 312–317.

SPIVAK, M. 1975. *A Comprehensive Introduction to Differential Geometry*. Publish or Perish.

TSAI, R. 2002. Rapid and accurate computation of the distance function using grids. *J. Comput. Phys. 178*, 1, 175–195.

ZHAO, H.-K. 2004. A fast sweeping method for Eikonal equations. *Mathematics of Computation*. to appear.

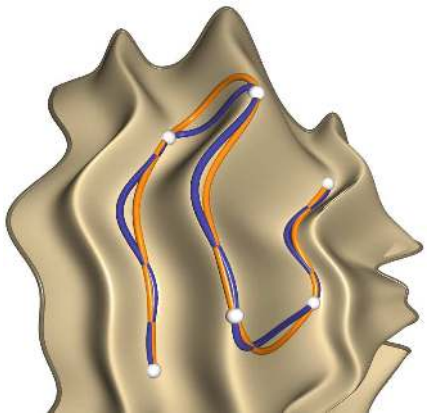

Figure 17: Adding feature sensitivity to curve design.