EUROGRAPHICS 2008 / G. Drettakis and R. Scopigno
(Guest Editors)

Volume 27 (2008), Number 2

Manifold-valued Thin-Plate Splines with Applications in Computer Graphics

Florian Steinke[1], Matthias Hein[2], Jan Peters[1], and Bernhard Schölkopf[1]

[1]Max Planck Institute for Biological Cybernetics, Tübingen, Germany
[2]Saarland University, Saarbrücken, Germany

Abstract

We present a generalization of thin-plate splines for interpolation and approximation of manifold-valued data, and demonstrate its usefulness in computer graphics with several applications from different fields. The cornerstone of our theoretical framework is an energy functional for mappings between two Riemannian manifolds which is independent of parametrization and respects the geometry of both manifolds. If the manifolds are Euclidean, the energy functional reduces to the classical thin-plate spline energy. We show how the resulting optimization problems can be solved efficiently in many cases. Our example applications range from orientation interpolation and motion planning in animation over geometric modelling tasks to color interpolation.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modelling, Splines

1. Introduction

Thin-plate splines (TPS) are a standard tool in computer graphics and also in many other disciplines both for interpolation and approximation. So far, most work has been focused on Euclidean output data, for example in trajectory design with control points in $\mathbb{R}^3$ or in implicit surface reconstruction. The current paper generalizes thin-plate splines to the case where the output space is a Riemannian manifold.

In computer graphics data living on manifolds occur quite naturally. Some basic types are directions, angles, and orientations, as well as smooth surfaces of objects and colors. More generally any data in Euclidean space which underlie smooth constraints can be seen as lying on a manifold. Therefore manifold-valued interpolation/approximation is of general interest in computer graphics.

1.1. Related Work

Thin-plate splines are characterized as the minimizers of a differential energy, the squared Frobenius norm of the Hessian subject to data interpolation/approximation constraints. Research in manifold-valued splines has up to now mainly focused on curves, in which case thin-plate splines are equivalent to well-known cubic splines. Cubic splines have been generalized to manifold-valued data in [GK85, NHP89, BCGH92] by replacing the standard derivative with the intrinsic covariant derivative on the manifold. Recently, an approach using the second extrinsic derivative has been investigated in [HP04] and then generalized to a network of curves in [WPH07]. For a broad overview of existing techniques for interpolating curves on manifolds, with an eye on motion planning, see [NP07].

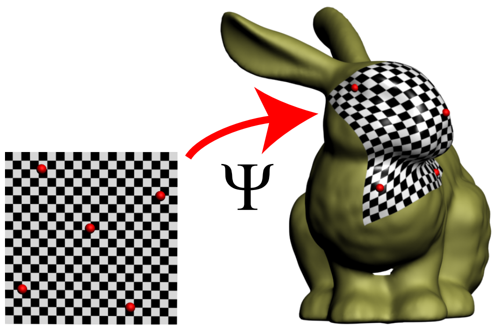


Figure 1: *Manifold-valued thin-plate spline Ψ mapping a 2D region onto a 3D bunny model using five markers.*

The energy of mappings between Riemannian manifolds has been first studied by Eells and Sampson [ES64]. They define an energy based on the first-order differential of the mapping. The local extrema of this energy are the so called harmonic maps. Since distortion-free (isometric) mappings are harmonic, discrete harmonic mappings with Euclidean output are commonly used, e.g., in [ZRS05]. A method for interpolation/approximation of manifold-valued data based on the harmonic energy has been proposed in [MSO04].

1.2. Roadmap

The aim of this paper is to generalize thin-plate splines from the Euclidean to the manifold setting, or equivalently generalize cubic splines on curved spaces to the case where one has multivariate input. We will define a suitable energy for multivariate mappings between two manifolds, which reduces to the thin-plate spline energy if both manifolds are Euclidean. The parametrization independent energy will only use intrinsic geometric properties of the manifold. We will show that similar to the difference between cubic and linear splines, our method leads to a smoother solution than using the harmonic energy which is based on the first order derivative. Special attention will be given to the boundary and appropriate boundary conditions. This will allow us to smoothly extrapolate the mapping outside of the data range, without fixing the boundary a priori. Note that extrapolation is not possible in the formulation of cubic splines on curved spaces in [GK85, NHP89, BCGH92, HP04] since start and end points of the curve need to be fixed.

Particular emphasis will be placed on the efficient implementation of the corresponding optimization problem. We believe that the theoretical soundness, the relatively easy and efficient implementation, and a wide range of possible applications provide the potential that the manifold-valued generalization of thin-plate splines will become a standard tool, just as their Euclidean equivalent.

After a sketch of the theoretical framework in Section 2 we will describe in Section 3 the implementation of the optimization problem in detail. We demonstrate the method on several examples, namely interpolation of rotations (Section 4.1), learning of task-space tracking (Section 4.2), mapping two dimensional regions onto smooth surfaces (Section 4.3), and color interpolation (Section 4.4).

2. Theoretical framework

We would like to define an energy functional for a mapping $\phi : M \to N$ between two Riemannian manifolds M and N. Three objectives should hold for the energy functional.

1. independence of the parametrization of M and N,
2. intrinsic formulation, that is it should only depend on the geometry of M and N,
3. penalization of the second order differential. In particular for $M \subseteq \mathbb{R}^m$ and $N \subseteq \mathbb{R}^n$ it should reduce to the thin-plate spline energy,

$$S_{\text{ThinPlate}}(\phi) = \int_{M\subseteq\mathbb{R}^m} \sum_{\alpha,\beta=1}^{m} \sum_{\mu=1}^{p} \left(\frac{\partial^2\phi^\mu}{\partial x^\alpha \partial x^\beta}\right)^2 dx.$$

The first objective means that the energy should not depend on the coordinate representation of the manifold, e.g., the energy of curves on the sphere should be the same if we represent the sphere in spherical or stereographic coordinates. This can be achieved by formulating the energy in the covariant language of differential geometry. The second requirement is that the energy should only depend on the geometry of M and N, that is only *intrinsic* properties of M and N should matter. In particular, if M and N are isometrically embedded in Euclidean space like the sphere $\mathbb{S}^2$ in $\mathbb{R}^3$ or SO_3 in $\mathbb{R}^{3\times 3}$, no properties of the ambient spaces should be taken into account, since the embedding is not unique. The third objective is motivated by the fact that an energy functional only penalizing the first order differential leads only to piecewise differentiable solutions as is shown below.

We call the resulting energy functional *Eells* energy after James Eells, who pioneered the study of harmonic maps between Riemannian manifolds. The derivation requires some heavy machinery from differential geometry, for better readability we have moved it into the Appendix A-C. Here, we present a particular simple form of the energy functional in the case where the input manifold M is Euclidean and the output manifold N is a submanifold in Euclidean space $\mathbb{R}^p$. These conditions on input and output manifold hold in most of our example applications covering many fields of computer graphics. Let $i : N \to \mathbb{R}^p$ be the isometric embedding of N into $\mathbb{R}^p$, which is just the identity map if N is a submanifold of $\mathbb{R}^p$. Then we denote by $\Psi = i \circ \phi$ the composition of the map $\phi : M \to N$ with the inclusion map i.

Introducing standard Cartesian coordinates in $M \subset \mathbb{R}^m$ and $\mathbb{R}^p$, the Eells energy can be written as

$$S_{\text{Eells}}(\Psi) = \int_{M\subseteq\mathbb{R}^m} \sum_{\alpha,\beta=1}^{m} \sum_{\mu=1}^{p} \left[\left(\frac{\partial^2\Psi^\mu}{\partial x^\alpha \partial x^\beta}\right)^\top\right]^2 dx, \quad (1)$$

where $\top$ denotes the projection onto the tangent space of N. It can be shown that this form of the energy is equivalent to the intrinsic formulation defined on N, although we are apparently using extrinsic properties, i.e. properties related to $\mathbb{R}^p$. The crucial difference to standard multivariate splines is the projection $\top$ onto the tangent space. This way, we penalize only intrinsic variations.

It is instructive to discuss the difference in the case of curves. For the interpolation of curves on manifolds an extrinsic energy has been proposed by Hofer and Pottmann [HP04]. Their extrinsic energy is given by $\int_{M\subseteq\mathbb{R}} \sum_{\mu=1}^{p} \left(\frac{\partial^2\Psi^\mu}{\partial t^2}\right)^2 dt$ whereas the Eells energy reduces to the cubic spline energy for curves on manifolds $\int_{M\subseteq\mathbb{R}} \sum_{\mu=1}^{p} \left[\left(\frac{\partial^2\Psi^\mu}{\partial t^2}\right)^\top\right]^2 dt$, as proposed in [GK85, NHP89]. One can decompose the acceleration $\frac{\partial^2\Psi^\mu}{\partial t^2}$ into its tangential and normal component. Thus, in the extrinsic energy, apart from the desired tangential, intrinsic acceleration one penalizes also the normal, extrinsic acceleration. The set of minimizers of both energies can therefore differ substantially. Since geodesics have zero intrinsic acceleration they are clearly minimizing the Eells energy. This is quite desirable since geodesics correspond to the most “simple” curves on manifolds. However, geodesics will not necessarily minimize the extrinsic energy of [HP04] due to the penalization of the normal component of the acceleration of the curve.

The domains M that we consider usually possess a boundary. Thus, we have to specify the behavior at the boundary via boundary conditions. Using variational techniques we find the extremal equation of the Eells energy, see Theorem 1 in Appendix A. We deduce sufficient boundary conditions (BC), see Equation 12, which for the case $M \subseteq \mathbb{R}^m$ can be written in extrinsic notation, see Theorem 2, as

$$\sum_{\alpha=1}^{m} N^{\alpha}\left(\frac{\partial^2\Psi^{\mu}}{\partial x^{\alpha}\partial x^{\beta}}\right)^{\top} = 0, \qquad \sum_{\alpha,\beta=1}^{m} N^{\alpha}\frac{\partial}{\partial x^{\beta}}\left(\frac{\partial^2\Psi^{\mu}}{\partial x^{\beta}\partial x^{\alpha}}\right)^{\top} = 0, \tag{2}$$

where N^{α} is the normal vector field at the boundary of M. Consideration of these boundary conditions is novel even for cubic splines on curved spaces. They allow the smooth extrapolation of the solution.

3. Implementation

Given the data points (x_i, y_i) with $x_i \in M$ and $y_i \in N$, we generally compute approximating splines. Denoting the squared geodesic distance in N by $d^2(.,.)$, we minimize the functional

$$S_{Eells}(\Psi) + C\sum_{i=1}^{k} d^2(y_i, \Psi(x_i)) \tag{3}$$

over all $\Psi : M \to \mathbb{R}^p$ subject to the conditions $\Psi(x) \in N$ and boundary conditions (2). For large C objective (3) enforces interpolation.

If N is Euclidean, it can be shown that the minimizer of (3) is a weighted sum of basis functions, piecewise cubic polynomials for curves, or thin-plate basis functions in the multivariate setting [HL93]. However, addition is not well defined for points on general manifolds, thus we cannot hope for such a nice expansion if N is a general manifold. Instead, we resort to discretization of the input space M which is still very efficient due to sparsity of all involved matrices.

Below we describe the necessary discretization steps, taking special care of boundaries of the domain M. We show how to minimize the resulting optimization problem efficiently using a geometrically motivated constrained Newton approach.

3.1. Discrete Formulation of the Optimization Problem

In our model applications shown below, we use the spaces $[0,1]$, $[0,1]^2$, and $\mathbb{S}^1$ as input manifolds M. We cover these with a regular grid with spacing h, allowing us to use standard symmetric finite difference approximations for first and second order derivatives. At the boundaries of M we use a virtual point scheme: for each discretization point on the boundary we compute the boundary normal, for points in corners there may be several. We construct for any boundary point in M two new "virtual" points outside the domain of M by translating each boundary point along the normal vector by h and $2h$. The virtual points are added to the interior discretization points, yielding the set X_d of all d discretization points. Domains with non-constant metric or non-trivial boundary could be dealt with using an interpolation scheme between non-uniformly spaced discretization points in M, or employing techniques from [BCOS01] who discretize PDEs on general manifolds.

We represent $\Psi : M \to \mathbb{R}^p$ by its function values at the discretization points, i.e. as a vector $\Psi \in \mathbb{R}^{dp}$. Discrete expressions for the first and second derivatives in direction α, β are stored in sparse block diagonal $dp \times dp$ matrices $\mathbf{D}_{\alpha}, \mathbf{D}_{\alpha,\beta}$. Rows corresponding to virtual points are left blank. The matrices $\mathbf{F}_{int}$, $\mathbf{F}_{bd}$ filter out rows corresponding to interior points, or boundary points respectively. From the k data points (x_i, y_i) we build a vector $\mathbf{y} \in \mathbb{R}^{kp}$, and a block diagonal $kp \times dp$ interpolation matrix $\mathbf{S}$ such that $\mathbf{S}\Psi$ yields weighted k-nearest neighbor estimates of $\Psi(x_i)$. The $dp \times dp$ matrix $\mathbf{P}_t^{\Psi}$ is the orthogonal projection of $\mathbb{R}^{dp}$ vectors onto the tangent spaces of the output manifold N at positions encoded in Ψ. Lastly, $\mathbf{N}^{\alpha}$ are diagonal matrices storing the α components of the boundary normals at the discretization points on the boundary. In matrix notation problem (3) then reads

$$\begin{aligned}
\min_{\Psi} \quad & \sum_{\alpha,\beta} \Psi^T \mathbf{D}_{\alpha,\beta}^T (\mathbf{P}_t^{\Psi})^T \mathbf{F}_{int} \mathbf{P}_t^{\Psi} \mathbf{D}_{\alpha,\beta}\Psi + C d^2(\mathbf{S}\Psi, \mathbf{y}) \\
s.t. \quad & \sum_{\alpha} \mathbf{N}^{\alpha}\mathbf{F}_{bd}\mathbf{P}_t^{\Psi}\mathbf{D}_{\alpha,\beta}\Psi = 0 \quad \forall\, \beta = 1,..,p \\
& \sum_{\alpha,\beta} \mathbf{N}^{\alpha}\mathbf{F}_{bd}\mathbf{D}_{\beta}\mathbf{P}_t^{\Psi}\mathbf{D}_{\beta,\alpha}\Psi = 0 \\
& \Psi(x) \in N \quad \forall\, x \in X_d.
\end{aligned} \tag{4}$$

To exemplify the notation we sketch it for mappings $\Psi : [0,1] \to \mathbb{S}^2 \subset \mathbb{R}^3$. Here, $X_d = \{-2h, -h, .., 1+2h\}$ where $-2h, -h, 1+h, 1+2h$ are the virtual points. We stack the different output components of Ψ above each other, $\Psi = (\Psi^1(-2h), \Psi^1(-h), .., \Psi^3(1+2h))^T$. $\mathbf{D}_1, \mathbf{D}_{11}, \mathbf{F}_{bd}, \mathbf{F}_{int}, \mathbf{S}, \mathbf{N}^1$ are block-diagonal matrices with three identical sub-blocks, one for each output dimension. It is $\mathbf{D}_{1,ij}^{sub} = \delta_{i,j+1} - \delta_{i,j-1}$, except for the blank rows $1, d$. $\mathbf{F}_{bd,ij}^{sub} = \delta_{i=1,j=3} + \delta_{i=2,j=d-2}$ since the left (right) boundary point has index 3 $(d-2)$ in X_d. Corresponding surface normals in M are stored as $\mathbf{N}^{1,sub} = \mathrm{diag}(-1,1)$. $\mathbf{P}_t^{\Psi}$ is assembled from individual projections $\mathbf{P}_t^{\Psi(x)} \in \mathbb{R}^{3\times 3}$, $x \in X_d$.

3.2. Optimization

Optimization problem (4) closely resembles a linear constrained quadratic problem, that could be solved in one Newton step. However, the geodesic distance function $d^2(.,.)$, the

Algorithm 1 Optimization routine

```
1: $\Psi \leftarrow \text{ProjectOn}N(\text{free TPS solution})$
2: **repeat**
3:     $P_t^\Psi \leftarrow$ tangent space projection at current $\Psi$
4:     Compute geodesic distance $\mathbf{S}\Psi$ to $\mathbf{y}$,
       $\tilde{\mathbf{y}} \leftarrow \mathbf{S}\Psi - \frac{1}{2}\nabla d^2(\mathbf{S}\Psi, \mathbf{y})$
5:     Determine direction $\delta\Psi$: Set $\Psi_0 \leftarrow \Psi$
       $\mathbf{A} \leftarrow \sum_{\alpha,\beta} \mathbf{D}_{\alpha,\beta}^T (\mathbf{P}_t^\Psi)^T \mathbf{F}_{int} \mathbf{P}_t^\Psi \mathbf{D}_{\alpha,\beta} + C2\mathbf{S}^T\mathbf{S}$
       $\mathbf{C} \leftarrow \begin{pmatrix} \sum_\alpha \mathbf{N}^\alpha \mathbf{F}_{bd} \mathbf{P}_t^\Psi \mathbf{D}_{\alpha,\beta} \\ \sum_{\alpha,\beta} \mathbf{N}^\alpha \mathbf{F}_{bd} \mathbf{D}_\beta \mathbf{P}_t^\Psi \mathbf{D}_{\beta,\alpha} \\ \mathbf{1} - P_t^\Psi \end{pmatrix}$
       $\mathbf{b} \leftarrow 2\mathbf{S}^T\tilde{\mathbf{y}}$, $\mathbf{d} \leftarrow \begin{pmatrix} 0 & 0 & \Psi_0^T(\mathbf{1} - P_t^\Psi) \end{pmatrix}^T$
       and solve system (6) for $\mathbf{x} = \Psi$. $\delta\Psi \leftarrow \Psi - \Psi_0$
6:     $t^* = \text{argmin}_{t>0} \text{Energy}\big(\text{ProjectOn}N(\Psi_0 + t\delta\Psi)\big)$
7:     $\Psi \leftarrow \text{ProjectOn}N(\Psi_0 + t^*\delta\Psi)$
8: **until** $\|\Psi - \Psi_0\|_\infty <$ threshold
```

constraint $\Psi(x) \in N$, and the dependence of $\mathbf{P}_t^\Psi$ on Ψ rule out such a direct approach. Instead, as outlined in algorithm 1, we solve the problem iteratively, approximating (4) in each step with the closest linearly constrained quadratic problem. This numerical scheme turned out to be robust and efficient, typically converging in few iterations.

Given a current solution Ψ_0, we replace the non-linear constraint $\Psi(x) \in N$ at each point with a geometrically motivated linear alternative: We constrain $\delta\Psi(x) = \Psi(x) - \Psi_0(x)$ to lie in the tangent space of N at $\Psi_0(x)$. Furthermore, the non-linear squared geodesic distance function $d^2(\mathbf{S}\Psi, \mathbf{y})$ is approximated by a Euclidean distance term. We compute the geodesic distance from $\mathbf{S}\Psi$ to $\mathbf{y}$, and place a virtual target $\tilde{\mathbf{y}}$ in the tangent plane of $\mathbf{S}\Psi$ in direction of $\mathbf{y}$ at the computed distance value. This way, the Euclidean term $\|\mathbf{S}\Psi - \tilde{\mathbf{y}}\|^2$ has the same value and gradient as the replaced $d^2(\mathbf{S}\Psi, \mathbf{y})$. Formally, $\tilde{\mathbf{y}} = \mathbf{S}\Psi_0 - \frac{1}{2}\nabla d^2(\mathbf{S}\Psi_0, \mathbf{y})$.

In each iteration we then solve a constrained quadratic objective of the form

$$\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x} \quad s.t. \quad \mathbf{C}\mathbf{x} = \mathbf{d}. \tag{5}$$

Introducing Lagrange multipliers λ, the minimum is achieved by the solution of

$$\begin{pmatrix} \mathbf{A} & \mathbf{C}^T \\ \mathbf{C} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{d} \end{pmatrix}. \tag{6}$$

Since all involved matrices are extremely sparse, these problems are amenable to efficient sparse solvers. For medium to large problems we used the exact solver CHOLMOD [CDHR06], for very large problems preconditioned conjugate gradient methods could be used. We add a small ridge to increase numerical stability.

Having solved the quadratic problem, the vectors $\delta\Psi$ indicate a direction of descent. We perform a line search using Goldstein's rule. For each proposed step size we project the corresponding Ψ back onto the manifold, and evaluate its energy there. The optimization is terminated when the maximal change of Ψ in one iteration is less than a small threshold.

As initial solution for the iterative scheme, we use the free solution, projected onto the manifold N. I.e., we first compute the minimizer of (4) pretending the output space was $\mathbb{R}^p$, in which case the problem is equivalent to the normal thin-plate spline solution. For complex output manifolds the projection of the free Ψ onto N may introduce large distortions increasing the danger of local minima. Where necessary, we thus move Ψ slowly towards N in an iterative manner targeting a low Eells energy already for intermediate solutions. The squared distance of $\Psi(x)$ to the closest point on N is added to objective (4) while the constraint $\Psi(x) \in N$ is dropped. We solve for a new Ψ in each iteration with increasing weight on the distance term. To compute tangent space projections for points $\Psi(x) \notin N$ as required during this process, we use the iso-distance manifold to N through $\Psi(x)$.

Note that it is mainly the constraint $\Psi(x) \in N$ that couples the different components of Ψ in (4). In the case where N is Euclidean or where N is the direct sum of several Riemannian manifolds, all involved matrices are block diagonal and each component of Ψ can be computed separately. However, the proposed algorithm is also quite efficient even for coupled dimensions. Sparse solvers typically scale linearly with the number of nonzero entries. As this number is proportional to the number of output dimensions in our case, the overall scaling of our algorithm is linear in the number of output dimensions. It also scales sub-quadratically with the number of discretization points, see Section 4.5. Since the number of discretization points grows exponentially with the number of input dimensions, our approach is limited to a small number of input dimensions. However, for many problems in computer graphics, this is not a problem, since usually the input dimension is low.

3.3. Operations on the output manifold

Our algorithm requires only three operations from the embedded output manifold N: (1) Projection of any point in the embedding space onto the manifold. (2) Projection onto the tangent space at any point of N. (3) Computation of the geodesic distance and its derivative between any two points of the manifold N.

The implementation of these operations depends largely on the way the manifold N is represented. For many manifolds and their standard embeddings these steps are straightforward. For $N = \mathbb{S}^2 \subset \mathbb{R}^3$, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$, it is $\text{ProjectOn}N(\mathbf{x}) = \mathbf{x}/\|\mathbf{x}\|$, the projection onto the tangent space $P_t^{\mathbf{x}} = \mathbf{1} - \frac{\mathbf{x}\mathbf{x}^T}{\mathbf{x}^T\mathbf{x}}$, the geodesic distance $d^2(\mathbf{x}, \mathbf{y}) = \text{acos}(\frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|})^2$, and its derivative $\nabla_{\mathbf{x}} d^2(\mathbf{x}, \mathbf{y}) = 2P_t^{\mathbf{x}} \frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|}\sqrt{d^2(\mathbf{x}, \mathbf{y})}$.

Rotations respectively orientations as members of SO_3

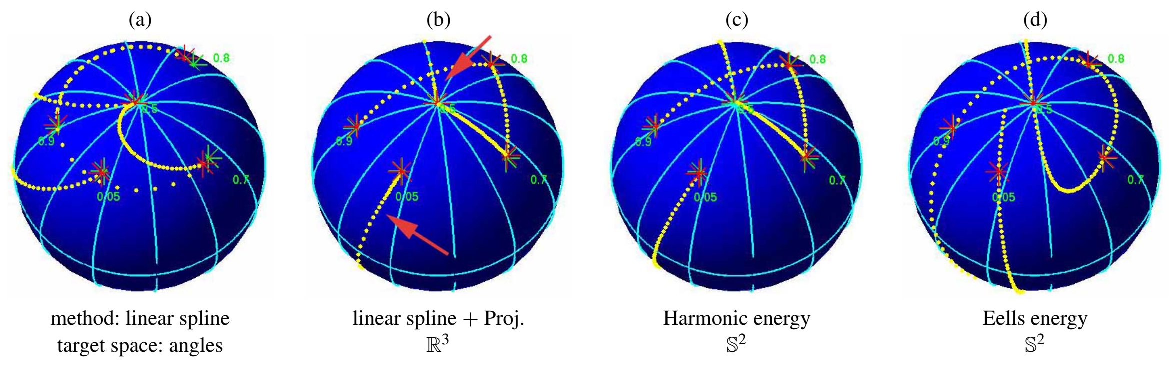


Figure 2: *The interval* $[0,1]$ *is mapped onto the unit sphere* $\mathbb{S}^2$ *in 3D. Green markers show the given data points* $y_i \in \mathbb{S}^2$*, respective training times* $x_i \in [0,1]$ *are given as numbers close-by. Red markers indicate* $\Psi(x_i)$ *for the approximating spline* $\Psi : [0,1] \to \mathbb{S}^2$*. Yellow dots mark the* Ψ*-images of equally spaced points in* $[0,1]$.

can be isometrically embedded in $\mathbb{R}^9$ as orthonormal 3×3 matrices. The induced distance is the absolute value of the rotation angle. Different object specific metrics implementing a non-trivial inertia tensor are proposed in [HP04] and can be dealt with similarly. The three operations are in this case: (1) The closest orthonormal matrix to any 3×3 matrix can be found via the polar decomposition. (2) the tangent space of a point $\mathbf{O}$ of the Lie group SO_3 is given by $\{\mathbf{OJ} \,|\, \mathbf{J} \in \Theta\}$ where Θ are the skew-symmetric matrices. (3) the geodesic distance between $\mathbf{O}_1$ and $\mathbf{O}_2$ is given as the Frobenius norm of the matrix logarithm $\log(\mathbf{O}_1\mathbf{O}_2^T)$.

We also experimented with surfaces in $\mathbb{R}^3$, which were given as densely sampled meshes. One approach would be to convert this discrete representation into a continuous differentiable one, e.g. by fitting an implicit surface description [OBA*03, MSO04]. However, we resorted to a much simpler scheme that worked well for densely sampled surfaces. We extract surface normals for each vertex, and identify the manifold close to a point x with the tangent plane of the closest vertex to x. Projection onto N and normal extraction can then be done efficiently using fast nearest-neighbor search. In order to determine geodesics, we first compute the shortest path on the given mesh using Dijkstra's algorithm. With this as initialization, we then compute a harmonic mapping (9) from $[0,1]$ to the surface fixing the end points to the points for which the geodesic distances should be computed. This can be done with an implementation almost identical to the one described above, just replacing second order derivatives with first order expressions. It is known that minimizers of the harmonic energy are geodesics [ES64]. Note that small geodesic distance in N implies also small Euclidean distance in $\mathbb{R}^p$. In many of the examples below, we worked with high weights C for the distance terms in (3), such that already after one iteration of the optimization algorithm the points were close with respect to the geodesic distance in N. From then on, we worked directly with the computationally cheaper Euclidean distance since the distances and gradients are in this case almost identical.

4. Experiments

4.1. Interpolation on the Sphere and on SO_3

We consider the approximation of a curve on the sphere $\mathbb{S}^2 \subseteq \mathbb{R}^3$ with 6 data points, see Figure 2. Before discussing manifold-valued splines, we present some naive approaches, highlighting the difficulties of the problem. A first idea could be to parameterize the surface of the sphere using spherical coordinates, and to interpolate the coordinates of the data points using linear splines (For visualization purposes we use linear splines corresponding to first order differential energies here). This is computationally attractive since the coordinates form a linear space, and the splines can be computed using basis function expansions. However, as shown in Figure 2(a), no path can go through the parametrization boundary $-\pi,\pi$, and the geometry is heavily distorted by the non-linear parametrization mapping from $\mathbb{S}^2$ to $(-\pi,\pi) \times (0,\pi)$. Alternatively, shown in Figure 2(b), one could first compute a linear spline in $\mathbb{R}^3$ and then project it onto the sphere. While the trajectory can now surround the sphere, the metric is still distorted. The yellow points are equally spaced in the input, however, close to the red arrows they are not equally spaced in the output.

Manifold adapted approaches are much better suited. In Figure 2(c), the harmonic energy (9) is used in the objective (3), instead of the Eells energy. Note that that the yellow points are now equally spaced between any two data points, up to small distortions resulting from the 2D visualization. However, since the minimizers of the harmonic energy are piecewise geodesic [MLH06], the curve is not differentiable. It also does not extend outside of the first/last marker. Using the Eells energy both these problems are avoided, see Figure 2(d). The curves are smooth and they extrapolate linearly, or more precisely geodesically.

Similar effects are demonstrated in the video of this paper showing an interpolation from $\mathbb{S}^1$ to SO_3 visualized as the periodic rotation of a teapot. We use an embedding of SO_3 into $\mathbb{R}^{3\times3}$ opposed to the quaternion based approach of [BCGH92] which is otherwise very similar.

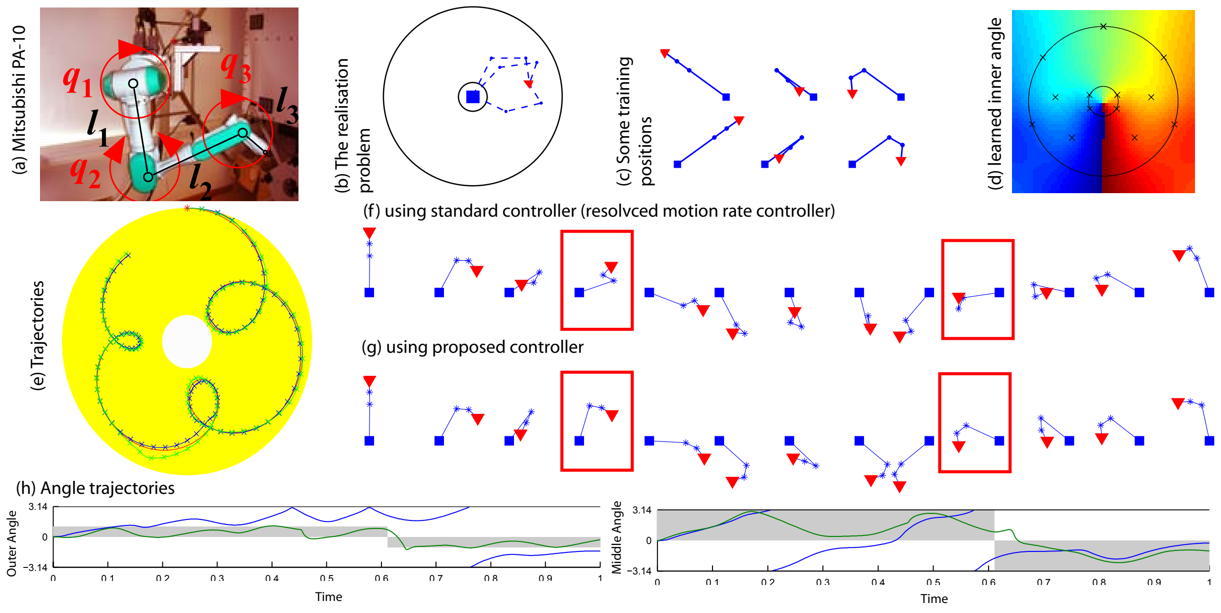


Figure 3: *(a) Example system: Mitsubishi PA-10 with three planar DoFs where two have no joint limits (the others are locked). (b) Many postures of a three link arm in two dimensions yield the same tip position. (c) Some training postures. (d) The inner most angle of the arm generalized to the unit square in task space, $\mathbb{R}^2$. Angle $-\pi$ corresponds to dark blue, π to dark red, training points are marked as black crosses. (e) The desired task space trajectory (red) is followed by both the resolved motion rate controller [SHV06] (blue) and our controller (green). The reachable space is yellow. (f,g) Postures during the trajectory. (h) Inner and outer angle plotted over time. The gray areas show the region of the training values for the current x position (right hand side positive angles, left hand side negative ones).*

4.2. Learning of Task-Space Tracking

Consider a skeleton based model in animation. As a running example we use a model of a robot arm, see Figure 3 (a). Most movement tasks are not defined through the model's joint angles $\mathbf{q} \in \mathbb{S}_1^n = \mathbb{S}^1 \times \cdots \times \mathbb{S}^1$ but rather by the motion of an end-effector $\mathbf{x} \in \mathbb{R}^m$, the fingertip. Thus, task-space planning and control requires the inverse kinematic mapping of the task onto the joint space.

Most interesting models are redundant $n > m$, i.e. there is a whole set of joint angles which all put the finger tip at the same location. Some of these will look natural, others won't. A controller that just focuses on keeping the end effector on the desired trajectory may thus lead to rather undesirable postures. In practice it may be quite hard to specify all (soft) constraints for a high-dimensional system explicitly, and it may be much easier to specify a number of example postures. We therefore propose to generate joint-space trajectories that stay close to previously observed postures.

Typically, redundancy resolution is achieved by pulling the robot towards a rest posture as implemented for example in the 3DSMax HI controller. Learning of postures has been proposed by [GMHP04] who use Gaussian process regression. However, since some joints can rotate by 360° our manifold-valued thin-plate splines are much better suited for such a situation.

Formally, we assume that we are given a desired path $\mathbf{x}_d(t) \in \mathbb{R}^m$ of the finger tip. At time t, we aim at determining $\delta\mathbf{q}$ in the model's joint angles $\mathbf{q} \in \mathbb{S}_1^n$ such that the new posture $\mathbf{q} + \delta\mathbf{q}$ with tip position $\mathbf{x}(\mathbf{q} + \delta\mathbf{q})$ is close to the desired position $\mathbf{x}_d(t)$ and at the same time is similar to training postures in this region of task space. For generalising locally preferred postures $\mathbf{q}_1, .., \mathbf{q}_k$ at positions $\mathbf{x}_1, .., \mathbf{x}_k$ to all reachable positions in task space, we use our generalized thin-plate splines to learn a mapping $\mathbf{q}_{\text{pred}} : \mathbb{R}^m \to \mathbb{S}_1^n$. We then choose $\delta\mathbf{q}$ such that it solves the optimization problem

$$\begin{aligned} \min_{\delta\mathbf{q}} \quad & \|\mathbf{x}(\mathbf{q}+\delta\mathbf{q}) - \mathbf{x} - \delta\mathbf{x}_d - \kappa[\mathbf{x}_d(t) - \mathbf{x}]\|^2 \\ & + \lambda_1 \|\delta\mathbf{q}\|^2 + \lambda_2 d^2_{\mathbb{S}_1^3}(\mathbf{q}+\delta\mathbf{q}, \mathbf{q}_{\text{pred}}(x)). \end{aligned} \tag{7}$$

Firstly, this cost tries to keep the finger tip on the desired trajectory with a feedback term with gain κ. Secondly, we prefer small steps $\delta\mathbf{q}$, and lastly try to minimise the distance between $\mathbf{q} + \delta\mathbf{q}$ and suitably generalised training examples $\mathbf{q}_{\text{pred}}$. The trade-off between the different objectives is controlled by the weighting coefficients λ_1 and λ_2. Taking the derivative of (7) with respect to $\delta\mathbf{q}$ and equating to zero we arrive at the following control law,

$$\begin{aligned} \delta\mathbf{q} = (\mathbf{J}\mathbf{J}^T)^{-1}\mathbf{J}^T \Big[\quad & \lambda_1 \left(\delta\mathbf{x}_d - \kappa[\mathbf{x}_d(t) - \mathbf{x}]\right) \\ & + \lambda_2 \nabla d^2_{\mathbb{S}_1^3}(\mathbf{q}+\delta\mathbf{q}, \mathbf{q}_{\text{pred}}(x)) \Big] \end{aligned} \tag{8}$$

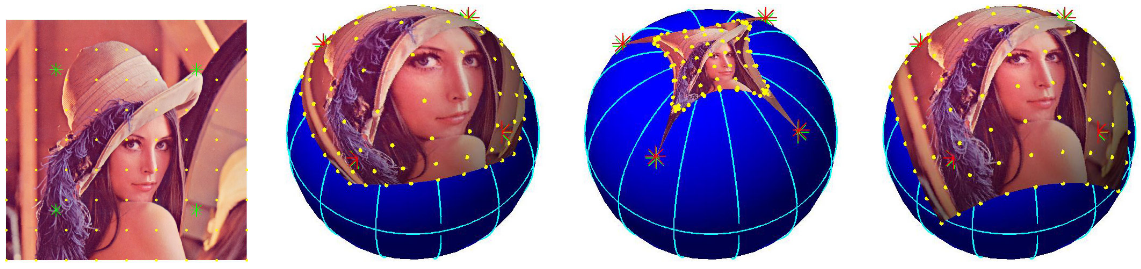

(a) original in $\mathbb{R}^2$
(b) thin-plate to $\mathbb{R}^3$ + proj.
(c) harmonic $\mathbb{S}^2$
(d) thin-plate to $\mathbb{S}^2$

Figure 4: *The Lena image (a) is used to visualize a mapping from the unit square in $\mathbb{R}^2$ to the unit sphere $\mathbb{S}^2$ in $\mathbb{R}^3$. Green markers show the given data point pairs, red stars on $\mathbb{S}^2$ denote positions of the input markers in $\mathbb{R}^2$ mapped to the sphere by the approximating spline.*

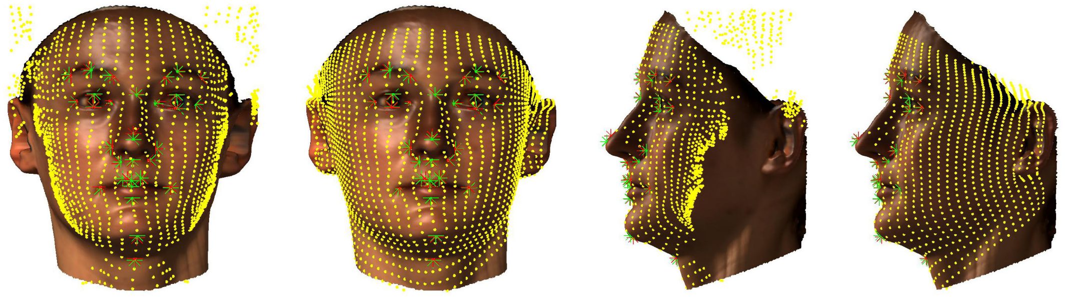


Figure 5: *Thin-plate splines mapping a regular grid in $\mathbb{R}^2$ (yellow points) onto a face manifold in $\mathbb{R}^3$. Green and red markers as in Figure 4.*

where $\mathbf{J}$ is the forward kinematic Jacobian $\mathbf{J}(\mathbf{q}) = \frac{\partial \mathbf{x}}{\partial \mathbf{q}}(\mathbf{q})$.

The presented method is evaluated on the three link ($n = 3$) arm model, see Figure 3 (a). For better visualization we chose a planar configuration ($m = 2$). Many postures $\mathbf{q}$ yield the same end effector location $\mathbf{x}$, see Figure 3(b). Training postures in Figure 3(c) are bent to the right for points $\mathbf{x}$ right of the base, to the left otherwise. From 15 examples (black crosses in Figure 3(d)) we learn the function $\mathbf{q}_{\text{pred}}(x)$; its first component is color coded in Figure 3(d). Note the direct transition from $-\pi$ to π would be impossible with normal thin-plate splines, since they are not aware of the fact that π and $-\pi$ actually encode the same angle. While the standard resolved motion rate controller [NCM*05, SHV06] ($\lambda_2 = 0$) results in intuitively quite unnatural poses (red boxes in Figure 3(f,g)) despite a null-space term, ours stays close to the more natural training set. Also, when plotting the middle and outer angles — for which the training data imply a kind of soft constraints, see gray areas in Figure 3(h) — our controller consistently stays closer while full-filling the task to follow $\mathbf{x}_d(t)$ equally well as the default approach, see Figure 3(e).

The above example is also visualized in the video of this paper, where we compare against two alternatives. The resolved motion rate controller [SHV06] (red) has no preferred posture, the 3DSMAX HI inverse kinematics controller (blue) has a single one. In contrast, our approach (green) features location dependent preferred postures as learned from a set of training examples. While it performs similar to the 3DSMAX controller in the right half of the task space, it smoothly adapts to the reverse curvature of the arm when entering the left half of task space.

4.3. Geometric Modelling

Here, we experiment with mappings from $[0,1]^2 \in \mathbb{R}^2$ to smooth surfaces in 3D. Such smooth mappings generated from few data points are useful for many geometric modelling tasks such as surface parametrization, remeshing, or texture mapping.

In parametrization, one typically computes mappings from the surface to $\mathbb{R}^2$, see [SPR06] for an overview. However, there are also many applications where the inverse mapping is required. For this case, one could try to invert the forward mapping, but this may be costly and the estimated forward mapping need not even be invertible. Alternatively, one could directly estimate the inverse mapping from the $\mathbb{R}^2$ domain onto the manifold using our manifold-valued thin-plate splines. Similarly, one could use a regular grid mapped onto the surface of an object, to reorganize the mesh according to a 2D coordinate system. Such mappings should minimizes a reasonable measure of distortion. The Eells energy can be seen as such a measure of distortion. It follows from Proposition 2.21 in [EL83], see also [HSS08], that mappings have zero Eells energy if and only if they are totally geodesic, which means that geodesics in M are mapped to geodesics in N. Every distortion free mapping is totally geodesic. The converse is generally not true but together with enough training points one can find among all possible totally geodesic mappings the one which is distortion free. Therefore for a given interpolation problem we can see the Eells energy as a measure of the deviation from a distortion free mapping. This is in contrast to the harmonic energy where totally geodesic maps are local extrema of the harmonic energy, but the value of the energy is zero if and only if the map is constant, that is M is mapped to a single point in N. Therefore the Eells energy is a much better measure of distortion than the harmonic energy.

In Figure 4 we compare different approaches for splines from $\mathbb{R}^2$ to surfaces in $\mathbb{R}^3$ more closely. In Figure 4(b), we first compute thin-plate splines in $\mathbb{R}^3$, which yields a plane cutting through the given 4 markers in this case. We then project the plane onto the sphere. Observe the extreme fish-eye distortion resulting from projection. In Figure 4(c), we show results for variational splines using the harmonic energy. This approach is commonly used in geometric modelling, e.g., [ZRS05], although mostly targeting linear spaces. However, the mapped image does not fill the convex hull of the training points. This is why the harmonic energy is traditionally only used for input domains without boundary, or when the output boundary can be fixed a priori. [ZRS05] discusses a method to avoid this behavior, but one could alternatively use our proposed manifold-valued thin-plate splines, see Figure 4(d). Since the Eells energy does not try to minimize the distances between the points, but the variation of distances, it is much less prone to contraction of the image. Furthermore, our generalized thin-plate splines extrapolate nicely out of the convex hull of the marker points.
Similar effects are observed in Figure 5 showing a thin-plate spline from $\mathbb{R}^2$ to a face manifold guided by 30 markers. These were placed on feature points of the face such as eyes and mouth, their input position in $\mathbb{R}^2$ was determined by projecting the 3D points onto the surface of a vertical cylinder through the head. A more complex output manifold taken from the Stanford 3D Scanning Repository is used in Figure 1.

4.4. Color Interpolation

Another potential field of application for manifold-valued splines is color processing, since perceptually colors have a circular structure [She80]. This property is used in the HSV color space, where H, the hue value, is a circular variable.

For smoothing color values over an image, it makes sense to take into account the presence of edges. Edges can be included via a non-uniform metric in the input space. A one pixel distance could be termed large, if it crosses an edge, and small otherwise. This way our smoothing spline which varies slowly in units measured by the metric could express sharp changes over edges, whereas it would vary slowly within objects. A derivation and implementation hints are given in the Appendix D.

The effects are demonstrated in Figure 6, where we aim at coloring a black and white image of a circle (a). We interpolate given H color values (b) over the image, fixing the S and V channel values to 1. A uniform metric (c) misses to take into account the shape of the circle. We then compute the norm of the (a)-image gradients in (d), a simple edge detection scheme. Using these values as multiplier in the metric, we arrive at an interpolation that is much better suited to the image structure (e).

The same technique is used for image compression in Figure 7. The compression consists of the following steps: first, we transform the RGB image into HSV color space. We sample randomly 500 pixels of H values, corresponding to $2-3\%$ of all values. We also store the S and V components for the whole image. During decompression we interpolate the H channel of the image using manifold-valued thin-plate splines. The mapping $\Psi : \mathbb{R}^2 \to \mathbb{S}^1$ is learned using an edge-adapted metric as above, where the edges are extracted from the stored S and V channel. The HSV color image is finally transformed back to normal RGB values. Some experimental results are summarized below. RGB values range from 0 to 1, the error is the RGB root mean squared error over the whole image.

	Horse	Flower
Image size	135 x 200	133 x 100
Error interpol. in $\mathbb{R}$	0.029	0.144
Error interpol. in $\mathbb{S}^1$	0.028	0.042

While the overall compression rate and quality is certainly not state-of-the-art in well-developed image compression, the example may nevertheless show that manifold-valued thin-plate splines are able to capture important regularities in natural datasets such as color images. It might be possible to include such knowledge into a more sophisticated state-of-the-art compression scheme in the future.

4.5. Performance

Some run-times for our naive Matlab implementation on a dual core 2.2 GHz notebook are given below. For the Lena problem, Figure 4, the run-time for one optimization step empirically scaled like $O(d^{1.3})$ with the number of discretization points d, an average of 3.5 optimization steps were needed. Significant speed-ups and memory savings could be achieved with an adaptive discretization scheme as is often used in finite element methods.

	line	Lena	Lena	Lena	flower
# discret. points	105	118	726	2.7k	14k
Time [s]	0.14	0.21	2.1	13	122

The discretization error of Ψ for different spacings h between discretization points reduced like $O(h^{1.7})$ in the Lena example, when compared to results for a very fine discretization that was assumed to be identical with the continuous solution.

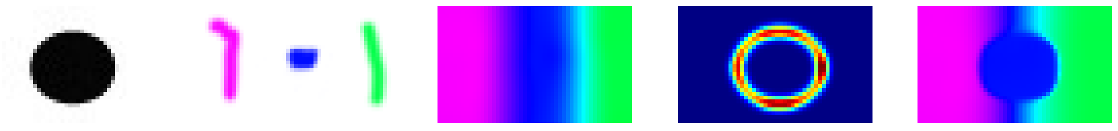
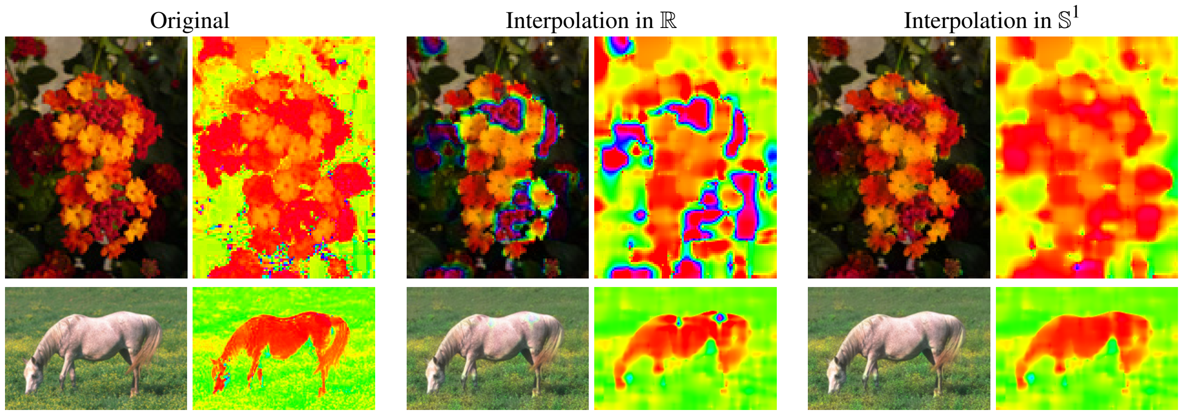


Figure 6: *Image (A) is colored by interpolating the colors in (B) in HSV color space, the H channel is modelled as $\mathbb{S}^1$. (C) shows results for the Eells energy with a uniform metric. However, we can extract edges from the original image (A) and use them as a scalar metric (D). The Eells interpolation then does not interpolate across edges (E), as the metric implies a large distance between the inner and the outer area of the circle.*



Original
Interpolation in $\mathbb{R}$
Interpolation in $\mathbb{S}^1$

Figure 7: *The original images (left) are compressed via a HSV space method. During compression we randomly discard* 98% *of the H channel of the original images (left column right), but we keep the full S and V information. At decompression time, we interpolate the H values either using normal splines from the image pixels to* $[0,1] \in \mathbb{R}$ *(middle column), or the Eells energy for splines targeting the circle* $\mathbb{S}^1$ *(right column). We obtain the H images shown in the right columns. When combining the interpolated H channels with the additionally stored S and V channels we obtain the images shown to the left of the H images.*

5. Conclusion

We have presented a generalization of thin-plate splines to manifold-valued output with a wide range of applications in computer graphics and elsewhere. In this paper we have focused on Euclidean input manifolds. However, the theoretical framework can be easily extended to general maps between Riemannian manifolds e.g. in order to compute dense point correspondences between two faces or other objects.

Appendix A: Derivation of the Eells energy

Let M be our m-dimensional input manifold and N the n-dimensional output manifold. Both are Riemannian manifolds with metric g in M and h in N. We will use abstract index notation, i.e., the tensor type is indicated by the position of "abstract" indices. They should not be mixed up with the indices for the components. A twice covariant tensor h is written as h_{ab} and the coordinate representation is $h_{ab} = h_{\mu\nu}\, dx^\mu_a \otimes dx^\nu_b$. In general, we use Greek letters for components (α,β,γ for components of M and μ,ν,ρ for components in N) and Latin ones for abstract indices (a, b, c for indices in M and r, s, t in N).

The differential $d\phi^r_a : T_xM \longrightarrow T_{\phi(x)}N$ of a mapping $\phi : M \longrightarrow N$ evaluated at x is given as

$$d\phi^r_a(x) = \frac{\partial \phi^\mu}{\partial x^\alpha} dx^\alpha_a \Big|_x \otimes \frac{\partial^r}{\partial y^\mu}\Big|_{\phi(x)} = {}^M\nabla_a \phi^\mu \Big|_x \otimes \frac{\partial^r}{\partial y^\mu}\Big|_{\phi(x)},$$

where x^α and y^μ are coordinates in M and N and ${}^M\nabla$ is the covariant derivative of M. The differential $d\phi^r_a$ measures the change of the output $\phi(x) \in N$ as one varies x in the input manifold M. This object can be used to define the most simple differential energy, the so called *harmonic energy*,

$$\begin{aligned} S_{\text{harmonic}}(\phi) &= \int_M \|d\phi\|^2_{T^*_x M \otimes T_{\phi(x)}N}\, dV(x) \\ &= \int_M g^{\alpha\beta}(x) h_{\mu\nu}(\phi(x)) \frac{\partial \phi^\mu}{\partial x^\alpha}\frac{\partial \phi^\nu}{\partial x^\beta} dV(x), \quad (9) \end{aligned}$$

where $dV = \sqrt{\det g}\, dx$ is the natural volume element of M. For standard multivariate regression, that is $M = \mathbb{R}^m$ and $N = \mathbb{R}$, the harmonic energy reduces to the energy functional of linear splines $S(\phi) = \int_{\mathbb{R}^d} \|\nabla\phi\|^2\, dx$. It is well-known that using this energy functional for interpolation/approximation leads to piecewise linear solutions. For curves on manifolds it leads to piecewise geodesic solutions, see [MLH06].

The problem with the definition of higher-order differentials of mappings between manifolds is that the first order differential $d\phi$ "lives" in the cotangent and tangent space,

T_x^*M and $T_{\phi(x)}N$, of two different manifolds. Thus we cannot simply use the connection ${}^M\nabla$ of M. The solution is to "pull-back" the connection ${}^N\nabla$ of N for the covariant derivatives of elements in $T_{\phi(x)}N$. The *pull-back connection* $\nabla' : T_xM \otimes T_{\phi(x)}N \to T_{\phi(x)}N$ is defined as

$$\nabla'_{\frac{\partial}{\partial x^\alpha}} \frac{\partial^r}{\partial y^\mu} :={}^N \nabla_{\phi_* \frac{\partial}{\partial x^\alpha}} \frac{\partial^r}{\partial y^\mu} = \frac{\partial \phi^\nu}{\partial x^\alpha} {}^N\Gamma^\rho_{\nu\mu} \frac{\partial^\mu}{\partial y^\rho},$$

where ${}^N\Gamma^\omega_{\nu\mu}$ are the Christoffel-symbols of N.

With this definition, we have a notion of the derivative of a vector field on N with respect to a variation in M, where M and N are connected via ϕ. The covariant derivatives of the differential $d\phi^r_a$ of $\phi : M \to N$ can thus be defined as

$$\begin{aligned} \nabla'_b d\phi^r_a &:= {}^M\nabla_b {}^M\nabla_a \phi^\mu \otimes \frac{\partial^r}{\partial y^\mu} + {}^M\nabla_a \phi^\mu \otimes \nabla'_b \frac{\partial^r}{\partial y^\mu} \\ &= \Big[\frac{\partial^2 \phi^\mu}{\partial x^\beta \partial x^\alpha} - \frac{\partial \phi^\mu}{\partial x^\gamma} {}^M\Gamma^\gamma_{\beta\alpha} + \frac{\partial \phi^\rho}{\partial x^\alpha} \frac{\partial \phi^\nu}{\partial x^\beta} {}^N\Gamma^\mu_{\nu\rho} \Big] \qquad (10) \\ &\qquad dx^\beta_b \otimes dx^\alpha_a \otimes \frac{\partial^r}{\partial y^\mu}. \end{aligned}$$

Up to here the presented notions can be found in [EL83].

Equivalent to the the thin-plate spline case, we now use the inner product in $T_x^*M \otimes T_x^*M \otimes T_{\phi(x)}N$, yielding the *Eells energy*,

$$\begin{aligned} S_{\text{Eells}}(\phi) &= \int_M \| \nabla'_b d\phi^r_a \|^2_{T_x^*M \otimes T_x^*M \otimes T_{\phi(x)}N} \, dV(x) \\ &= \int_M g^{ac} g^{bf} h_{rs} \nabla'_b d\phi^r_a \nabla'_d d\phi^s_c dV(x). \qquad (11) \end{aligned}$$

Appendix B: Variation of the Eells energy

Variation of the Eells energy provides us with necessary conditions for a minimizer and most importantly with boundary conditions for M.

Theorem 1 Let $\phi(t,x) : (-\varepsilon, \varepsilon) \times M \to N$ be a variation of the mapping $\phi = \phi(0,x)$ and $W^r = \frac{\partial}{\partial t} \phi^r_t \big|_{t=0}$ the variational vector field at $t = 0$. The variation of the Eells energy is given as,

$$\begin{aligned} &\frac{d}{dt} S_{\text{Eells}}(\phi_t) \Big|_{t=0} \\ =&2 \int_M g^{ab} g^{cd} h_{rs} W^r \left[\nabla'_c \nabla'_a \nabla'_b d\phi^s_d + R^N_{uvw}{}^s \, d\phi^w_a \, d\phi^u_c \, \nabla'_b d\phi^v_d \right] dV \\ &+ 2 \int_{\partial M} h_{rs} \, g^{ab} N^d \left[\nabla'_a W^r \nabla'_d d\phi^s_b - W^r \, \nabla'_a \nabla'_b d\phi^s_d \right] d\tilde{V} \end{aligned}$$

where $d\tilde{V}$ is the volume element of the boundary ∂M and R^N_{serb} is the curvature tensor of N.

A necessary condition for a minimizer of the Eells energy is $\frac{d}{dt} S_{\text{Eells}}(\phi_t) \Big|_{t=0} = 0$ for all vector fields W. A set of boundary conditions which achieves that the boundary terms vanish are

$$N^a \nabla'_a d\phi^r_b = 0, \qquad N^c g^{ab} \nabla'_a \nabla'_b d\phi^r_c = 0. \qquad (12)$$

The proof is basically build on the two following Lemmas (proofs can be found in [HSS08]). The first one is a generalization of the Green's theorem for the pull-back connection and the second one computes the commutator of the derivative with respect to the variation and the pull-back connection.

Lemma 1 Let $R \in \otimes^{p+1} T^*M \otimes \phi^{-1}TN$ and $S \in \otimes^p T^*M \otimes \phi^{-1}TN$, where $\phi^{-1}TN$ is the bundle of $T_{\phi(x)}$, $x \in M$. Then with ∇' being the pull-back connection,

$$\int_M \langle R, \nabla' S \rangle = \int_{\partial M} \langle R, N \otimes S \rangle - \int_M \langle \text{trace}_g \nabla' R, S \rangle,$$

where N is the covector associated to the normal vector of M and the trace is taken with respect to the first two indices.

Lemma 2 Let $\phi(t,x) : (-\varepsilon, \varepsilon) \times M \to N$ be a variation of the mapping $\phi = \phi(0,x)$. Then

$$\frac{\partial}{\partial t} \nabla'_a d\phi^r_b = \nabla'_a \nabla'_b \frac{\partial \phi^r}{\partial t} + R^N_{suv}{}^r \, d\phi^s_a \, \frac{d\phi^u}{dt} \, d\phi^v_b. \qquad (13)$$

Proof We use the commutator from lemma 2 and obtain,

$$\begin{aligned} \frac{d}{dt} S_{\text{Eells}}(\phi_t) &= 2 \int_M g^{ab} g^{cd} h_{rs} \left(\frac{\partial}{\partial t} \nabla'_a (d\phi_t)^r_c \right) \nabla'_b (d\phi_t)^s_d dV \\ =&2 \int_M g^{ab} g^{cd} h_{rs} \nabla'_a \nabla'_c \frac{\partial \phi^r}{\partial t} \, \nabla'_b (d\phi_t)^s_d dV \\ &+ 2 \int_M g^{ab} g^{cd} h_{rs} R^N_{uvw}{}^r \, (d\phi_t)^u_a \frac{\partial \phi^v_t}{\partial t} (d\phi_t)^w_c \, \nabla'_b (d\phi_t)^s_d \, dV \end{aligned}$$

One has $\nabla'_b (d\phi_t)^s_d \Big|_{t=0} = \nabla'_b d\phi^s_d$. We apply two times the Green's theorem of Lemma 1 and obtain

$$\begin{aligned} \frac{d}{dt} S_{\text{Eells}}(\phi_t) \Big|_{t=0} &= 2 \int_M g^{ab} g^{cd} h_{rs} \nabla'_a \nabla'_c W^r \, \nabla'_b d\phi^s_d \, dV \\ &+ 2 \int_M g^{ab} g^{cd} h_{rs} R^N_{uvw}{}^r \, d\phi^u_a W^v \, d\phi^w_c \, \nabla'_b d\phi^s_d \, dV \\ =&2 \int_{\partial M} N^b g^{cd} h_{rs} \nabla'_c W^r \, \nabla'_b d\phi^s_d \, d\tilde{V} \\ &- 2 \int_{\partial M} g^{ab} N^d h_{rs} W^r \, \nabla'_a \nabla'_b d\phi^s_d \, d\tilde{V} \\ &+ 2 \int_M g^{ab} g^{cd} h_{rs} W^r \, \nabla'_c \nabla'_a \nabla'_b d\phi^s_d \, dV \\ &+ 2 \int_M g^{ab} g^{cd} h_{rs} R^N_{uvw}{}^r \, d\phi^u_a W^v \, d\phi^w_c \, \nabla'_b d\phi^s_d \, dV \\ =&2 \int_M g^{ab} g^{cd} h_{rs} W^r \left[\nabla'_c \nabla'_a \nabla'_b d\phi^s_d + R^N_{uvw}{}^s \, d\phi^w_a \, d\phi^u_c \, \nabla'_b d\phi^v_d \right] dV \\ &+ 2 \int_{\partial M} h_{rs} \, g^{ab} N^d \left[\nabla'_a W^r \nabla'_d d\phi^s_b - W^r \, \nabla'_a \nabla'_b d\phi^s_d \right] d\tilde{V}, \end{aligned}$$

where we have used in the last step $R_{uvws} = R_{wsuv}$. □

Appendix C: From intrinsic to extrinsic representation

The expression of the Eells energy in coordinates is quite complicated as one can see from the explicit form of $\nabla'_b d\phi^r_a$ in Eq. (10). However, the expression dramatically simplify if N is an isometrically embedded submanifold of $\mathbb{R}^p$ and M is a subset of Euclidean space $\mathbb{R}^m$. We have to stress that the following reformulation of the energy is completely equivalent to the one in Eq. (11).

Let $i : N \longrightarrow \mathbb{R}^p$ be the isometric embedding and denote by $\Psi : M \longrightarrow \mathbb{R}^p$ the composition $\Psi = i \circ \phi$. Let z^μ be standard Cartesian coordinates in $\mathbb{R}^p$. Then the differential of Ψ is given as $d\Psi^r_a = \frac{\partial \Psi^\mu}{\partial x^\alpha} dx^\alpha_a \otimes \frac{\partial^r}{\partial z^\mu}$. As above we can also define an pull-back connection $\tilde{\nabla} : T_x M \otimes T_{\psi(x)}\mathbb{R}^p \longrightarrow T_{\psi(x)}\mathbb{R}^p$ for the mapping Ψ, $\tilde{\nabla}_{\frac{\partial}{\partial x^\alpha}} \frac{\partial^r}{\partial z^\mu} := {}^{\mathbb{R}^p}\nabla_{\Psi_* \frac{\partial}{\partial x^\alpha}} \frac{\partial^r}{\partial z^\mu} = 0$, which is trivial due to the flatness of the connection of $\mathbb{R}^p$. Because of this property the expressions for the corresponding covariant derivatives expression will simplify significantly. The following theorem shows how intrinsic expressions in ϕ can be expressed in terms of the extrinsic ones in Ψ.

Theorem 2 The following equivalences between intrinsic and extrinsic objects hold,

$$\begin{aligned} d\phi^r_a =& d\Psi^r_a, \qquad \nabla'_b d\phi^r_a = \left(\tilde{\nabla}_b d\Psi^r_a\right)^\top, \\ \nabla'_c \nabla'_b d\phi^r_a =& \tilde{\nabla}_c \left(\tilde{\nabla}_b d\Psi^r_a\right)^\top - d\Psi^s_c \, {}^N\Pi^r_{su} \left(\tilde{\nabla}_b d\Psi^u_a\right)^\top, \end{aligned}$$

where $^\top$ denotes the projection onto the tangent space $T_{\Psi(x)}N$ and ${}^N\Pi^r_{su}$ is the second fundamental form of N. If M is a domain in $\mathbb{R}^m$ we derive for the Eells energy (11) the expression given in Eq. (1), for the corresponding boundary conditions (12) the form (2).

The proof can be found in [HSS08]. Note that if M is a domain in $\mathbb{R}^m$ one has

$$\tilde{\nabla}_b d\Psi^r_a = \frac{\partial^2 \Psi^\mu}{\partial x^\alpha \partial x^\beta} \, dx^\alpha_a \otimes dx^\beta_b \otimes \frac{\partial^r}{\partial y^\mu}.$$

Appendix D: Non-uniform Metric in *M*

Consider the metric $g_{ij}(x) = \Omega(x)\delta_{ij}$ on M. For non-constant Ω, the Christoffel symbols ${}^M\Gamma^\kappa_{\beta\alpha}$ in the coordinate expression of $\nabla'_b d\Psi^r_c$ in Eq. (10) do not vanish. We compute

$$\begin{aligned} \nabla'_b d\Psi^r_c \;=\; & \Big[\frac{\partial^2 \Psi^\mu}{\partial x^\beta \partial x^\alpha} - \frac{\partial_\beta \Omega}{2\Omega} \frac{\partial \Psi^\mu}{\partial x^\alpha} - \frac{\partial_\alpha \Omega}{2\Omega} \frac{\partial \Psi^\mu}{\partial x^\beta} \\ & + \sum_\gamma \frac{\partial_\gamma \Omega}{2\Omega} \frac{\partial \Psi^\mu}{\partial x^\gamma} \Big] dx^\beta_b \otimes dx^\alpha_c \otimes \frac{\partial^r}{\partial y^\mu}. \end{aligned}$$

This is a linear expression in Ψ. For implementation we just have to change the second derivative matrices $\mathbf{D}_{\alpha,\beta}$ described in Section 3, by adding first order terms in form of weighted combinations of $\mathbf{D}_\alpha$ matrices.

References

[BCGH92] BARR A. H., CURRIN B., GABRIEL S., HUGHES J. F.: Smooth interpolation of orientations with angular velocity constraints using quaternions. *Computer Graphics 26*, 2 (1992), 313–320.

[BCOS01] BERTALMIO M., CHENG L., OSHER S., SAPIRO G.: Variational problems and partial differential equations on implicit surfaces. *J. of Comp. Phys. 174*, 2 (2001), 759–780.

[CDHR06] CHEN Y., DAVIS T., HAGER W., RAJAMANICKAM S.: Algorithm 8xx: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. *ACM Trans. Math. Softw., submitted in* (2006).

[EL83] EELLS J., LEMAIRE L.: *Selected topics in harmonic maps*. AMS, Providence, RI, 1983.

[ES64] EELLS J., SAMPSON J. H.: Harmonic mappings of Riemannian manifolds. *Amer. J. Math. 86*, 1 (1964), 109–160.

[GK85] GABRIEL S., KAJIYA K.: Spline interpolation in curved space. In *SIGGRAPH'85 Course Notes on State of the Art Image Synthesis* (1985), vol. 6, ACM Press, pp. 1–14.

[GMHP04] GROCHOW K., MARTIN S., HERTZMANN A., POPOVIĆ Z.: Style-based inverse kinematics. *ACM Transactions on Graphics 23*, 3 (2004), 522–531.

[HL93] HOSCHEK J., LASSER D.: *Fundamentals of Computer Aided Geometric Design*. A. K. Peters, 1993.

[HP04] HOFER M., POTTMANN H.: Energy-minimizing splines in manifolds. *ACM Transactions on Graphics 23* (2004), 284–293.

[HSS08] HEIN M., STEINKE F., SCHÖLKOPF B.: *Energy functionals for manifold-valued mappings and their properties*. Tech. Rep. 167, Max Planck Institute for Biological Cybernetics, Tübingen, Germany, 2008. available at *http://www.kyb.tuebingen.mpg.de/techreports.html*.

[MLH06] MACHADO L., LEITE F. S., HÜPER K.: Riemannian means as solutions of variational problems. *LMS J. Comput. Math. 9* (2006), 86–103.

[MSO04] MÉMOLI F., SAPIRO G., OSHER S.: Solving variational problems and partial differential equations mapping into general target manifolds. *J.Comp.Phys. 195*, 1 (2004), 263–292.

[NCM*05] NAKANISHI J., CORY R., MISTRY M., PETERS J., SCHAAL S.: Comparative experiments on task space control with redundancy resolution. In *Proc. IEEE/RSJ IROS* (2005).

[NHP89] NOAKES L., HEINZINGER G., PADEN B.: Cubic Splines on Curved Spaces. *IMA Journal of Mathematical Control and Information 6* (1989), 465–473.

[NP07] NOAKES L., POPIEL T.: Geometry for robot path planning. *Robotica* (2007), 1–11.

[OBA*03] OHTAKE Y., BELYAEV A., ALEXA M., TURK G., SEIDEL H.-P.: Multi-level partition of unity implicits. *ACM Transactions on Graphics 22* (2003), 463–470.

[She80] SHEPARD R.: Multidimensional Scaling, Tree-Fitting, and Clustering. *Science 210*, 4468 (1980), 390.

[SHV06] SPONG M. W., HUTCHINSON S., VIDYASAGAR M.: *Robot Modeling and Control*. Wiley, 2006.

[SPR06] SHEFFER A., PRAUN E., ROSE K.: Mesh Parameterization Methods and Their Applications. *Foundations and Trends in Computer Graphics and Vision 2*, 2 (2006), 105–171.

[WPH07] WALLNER J., POTTMANN H., HOFER M.: Fair webs. *The Visual Computer 23*, 1 (2007), 83–94.

[ZRS05] ZAYER R., RÖSSL C., SEIDEL H.: Setting the boundary free: A composite approach to surface parameterization. *Symposium on Geometry Processing* (2005), 91–100.