# Weighted Averages on Surfaces

Daniele Panozzo[1] Ilya Baran[2,3,4] Olga Diamanti[1] Olga Sorkine-Hornung[1]

[1]ETH Zurich [2]Belmont Technology, Inc. [3]Adobe Research [4]Disney Research, Zurich

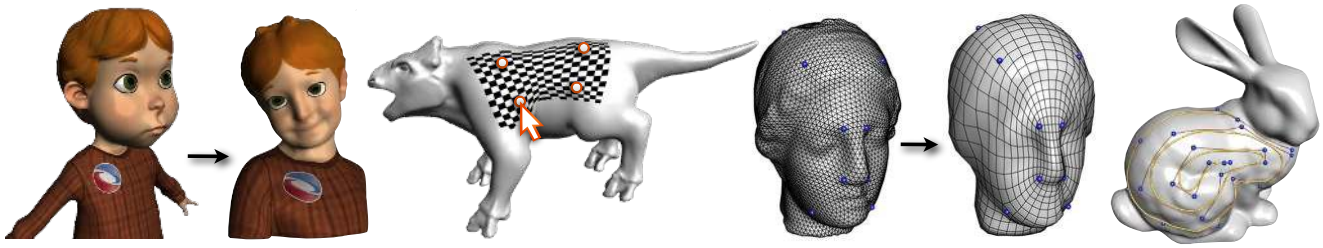


**Figure 1:** *Interactive control for various geometry processing and modeling applications made possible with weighted averages on surfaces. From left to right: texture transfer, decal placement, semiregular remeshing and Laplacian smoothing, splines on surfaces.*

## Abstract

We consider the problem of generalizing affine combinations in Euclidean spaces to triangle meshes: computing weighted averages of points on surfaces. We address both the *forward problem*, namely computing an average of given anchor points on the mesh with given weights, and the *inverse problem*, which is computing the weights given anchor points and a target point. Solving the forward problem on a mesh enables applications such as splines on surfaces, Laplacian smoothing and remeshing. Combining the forward and inverse problems allows us to define a correspondence mapping between two different meshes based on provided corresponding point pairs, enabling texture transfer, compatible remeshing, morphing and more. Our algorithm solves a single instance of a forward or an inverse problem in a few microseconds. We demonstrate that anchor points in the above applications can be added/removed and moved around on the meshes at interactive framerates, giving the user an immediate result as feedback.

**Keywords:** surface geometry, weighted averages, correspondence

**Links:** DL PDF WEB VIDEO

## 1 Introduction

Computing weighted averages, or affine combinations of points in Euclidean space is a fundamental operation. Given $n$ anchor points and corresponding weights, their weighted average can be easily computed by coordinate-wise weighted averaging. In this paper, we explore a generalization of weighted averages to points on triangulated surfaces (meshes) and develop a fast method for finding them. The natural way to generalize weighted averages to an arbitrary metric space is the Fréchet mean [Cartan 1929; Fréchet 1948]: it is defined as the point that minimizes the sum of weighted squared distances to the anchors. How to find this point, however, is not obvious from the definition, and this task has so far received little attention in the literature.

The Fréchet mean is typically studied with Riemannian metrics, such as geodesic distance. Computing geodesic distance between two *arbitrary* points on a triangle mesh, even despite the latest advancements, is relatively expensive. We therefore focus on a different class of metrics, which we call *Euclidean-embedding metrics*, that are derived by embedding the mesh in a (possibly high-dimensional) Euclidean space and computing Euclidean distance in that space. A number of known metrics, such as diffusion distance, commute-time distance and biharmonic distance [Lipman et al. 2010] are Euclidean-embedding metrics. We adapt the construction of Rustamov and colleagues [2009] to obtain a Euclidean-embedding metric that mimics geodesic distance.

We show that for a Euclidean-embedding metric, the Fréchet mean takes a special form: it is the result of taking the Euclidean weighted average of the points in the embedding space and projecting it (i.e., finding the closest point) onto the embedded mesh. However, the embedded mesh is not a smooth surface, and the Euclidean projection operator exhibits discontinuities near mesh edges. The Fréchet mean therefore also behaves discontinuously. We introduce a new projection operator which can be seen as a generalization of Phong projection [Kobbelt et al. 1999] to Euclidean spaces of dimension higher than three, and use this operator instead of the Euclidean projection. We show experimentally that our Phong projection behaves in a qualitatively similar way to Euclidean projection onto a smooth surface, although no smooth surface is actually constructed.

Armed with this Phong projection operator, we develop fast algorithms for computing the *forward problem* and the *inverse problem*. The forward problem is to find the weighted average of several anchors, given the anchors and the weights. The inverse problem is to compute weights for a given set of anchors, such that the weighted average is a given target point. In the Euclidean space, the inverse problem is known as generalized barycentric coordinates. Unlike the forward problem, the inverse problem has been previously studied from a computational point of view for geodesic distances [Rustamov 2010], and we give a solution for our setup.

Weighted averages are a fundamental building block that can be used for a variety of tasks in computer graphics and geometric modeling. Using the forward problem, we construct splines on meshes

and remesh surfaces semiregularly. Using both the forward and the inverse problem together allows us to define a dense correspondence between two meshes, which we can use to transfer surface-varying data, such as a texture, from one mesh to another or set up a morph between them. While all of the above tasks have specialized higher-quality algorithms, our methods are several orders of magnitude faster. We can perform all mentioned tasks interactively, providing more feedback to the user, and setting a new point on the performance-vs-quality tradeoff curve.

Our main technical contributions are:

1. We develop Phong projection in higher dimensions as a smoother alternative to Euclidean projection.
2. We present new generalized barycentric coordinates for *scattered* points.
3. We present an efficient algorithm for solving the forward problem.

## 2 Related work

The mathematical basis for weighted averages in a general metric space is the Fréchet mean [Cartan 1929; Fréchet 1948], also known as the Karcher mean [Karcher 1977] and the Riemannian center of mass. It is typically used with geodesic distance or other Riemannian metrics. However, geodesic distance is not $C^1$ and is sensitive to noise on meshes (Figure 10). Moreover, computing the geodesic distance is relatively costly [Surazhsky et al. 2005], since it needs to be done exactly (small errors in the distance can change the minimum location unpredictably), and between arbitrary points on the mesh (not just vertices). This may preclude recent fast approximate methods [Sethian 1996; Xin et al. 2012; Crane et al. 2013]. Instead, we use a metric obtained by embedding the mesh in a high-dimensional Euclidean space, which is smooth and very fast to evaluate.

Methods for computing the Fréchet mean have been developed on spheres [Buss and Fillmore 2001] and rotation groups [Pennec 1998]. Methods for computing other kinds of weighted averages have been proposed for different subgroups of matrices [Alexa 2002; Pálfia 2009]. Somewhat surprisingly, to the best of our knowledge, no efficient algorithms for computing the Fréchet mean have been previously presented for general surfaces.

For the inverse problem, there generally exists more than one set of weights for which given anchors average to a given point. For most applications, however, a particular weight vector is needed. Weights that vary smoothly and have other desirable properties are known as *generalized barycentric coordinates*. Their construction is a well-studied problem in Euclidean space [Floater 2003; Ju et al. 2005; Joshi et al. 2007; Lipman et al. 2007]. Langer and colleagues [2006] define generalized barycentric coordinates on a sphere and Rustamov [2010] on arbitrary surfaces as we do. However, the generalized barycentric coordinate schemes that Rustamov uses are defined for polygons, while we have a set of anchors in no particular order. We therefore use a simple scheme generalizing a recent idea by Waldron [2011] and Moving Least Squares (MLS). MLS interpolation has been generalized to surfaces [Jin et al. 2009] using geodesic distance, which has to be recomputed whenever an anchor changes. Rustamov's method suffers from the same drawback.

Prior methods have used interpolated tangent planes and normals at vertices to obtain smoother behavior on triangle meshes in 3D. The classical example of such a method is Phong shading [Phong 1975]. Kobbelt and colleagues [1999] use projection along the Phong normal to construct multiresolution mesh hierarchies. Registration techniques [Chen and Medioni 1991] and other methods [Sander et al. 2000] establish correspondences between nearby surfaces by shooting rays in the interpolated normal direction from one surface to the other. Phong tessellation [Boubekeur and Alexa 2008] uses tangent planes at mesh vertices to replace triangles with quadratic patches for smoother display. We develop an analog of Phong projection [Kobbelt et al. 1999] in higher dimensions for our purposes (Section 3.2).

Since weighted averages can be used for a variety of tasks, we briefly review the relevant literature for the most important applications of our framework.

**Cross-parametrization.** Recently, a lot of effort has been spent on establishing mappings between surfaces, due to the large number of practical applications that benefit from it, such as texture transfer and morphing [Eckstein et al. 2001; Tzur and Tal 2009]. The computed map may be optimized to be as isometric as possible [Ovsjanikov et al. 2010] or as conformal as possible [Kim et al. 2011]. In [Schreiner et al. 2004], progressive meshes are used to find the mapping, while in [Kraevoy and Sheffer 2004] the meshes are simplified and parametrized on a common base domain. Other similar methods have been developed [Sumner and Popović 2004; Yeh et al. 2011]. None of these techniques are fast enough to be interactive. Our method, while not specialized for this task, can be used to define a cross-parametrization between two surfaces given compatible anchors. When the anchors are changed (i.e., removed, added or displaced), our map can be updated at interactive rates, whereas all prior methods require minutes of computation or more on complicated cases.

**On-surface deformations.** Weighted averages on surfaces can similarly be used to construct a mapping from a surface to itself and thus deform a signal defined on it. An alternative method tailored to this problem was presented by Ritschel and colleagues [2010], who show various applications of on-surface deformations, allowing artists to control shadows, caustics and deform textures interactively. They simulate a piece of elastic cloth sliding over the surface and use a specialized GPU solver to achieve interactive frame rates. Constrained parametrization [Hormann et al. 2008] could also be used for the same goal, but handling surfaces with genus greater than one is complicated and computationally expensive.

**Splines on surfaces.** Buss and Fillmore [2001] demonstrated spherical splines as an application of their averaging operator. Spline curves on general surfaces have been studied in a variational setting [Hofer and Pottmann 2004], but the necessary optimization is relatively costly. Jin and colleagues [2009] defined curves on surfaces as iso-contours of an interpolated scalar field, but again, the computational requirements are significant. Wallner and Pottmann [2006] introduced an averaging-based definition of splines over smooth surfaces, where an averaging operator that produces points on the surface is defined. Their approach is to simply compute the weighted average in 3D space and project the result onto the surface. This is similar to our method, but without a high-dimensional embedding and without Phong projection the results can be less robust (Figure 3, left) and discontinuous on discrete meshes.

## 3 Method

Our method consists of several building blocks. We start by introducing notation and terminology for the Fréchet mean and Euclidean-embedding metrics and establishing some basic facts (Section 3.1). We then introduce Phong projection for higher dimensions (Section 3.2), describe our method for computing the forward problem (Section 3.3), and then show how to solve the inverse problem of generalized barycentric coordinates (Section 3.4).

Finally we describe how we construct the specific Euclidean-embedding metric that we use (Section 3.5).

### 3.1 Preliminaries

We refer to the $n$ points $\mathbf{x}_i$ whose average is to be computed as *anchors*. In $\mathbb{R}^k$, the forward problem is solved simply by coordinate-wise averaging using the given weights $w_i$, $\sum w_i = 1$:

$$\hat{\mathbf{x}} = \sum_{i=1}^{n} w_i \mathbf{x}_i. \tag{1}$$

The generalization of weighted averages to metric spaces is called the Fréchet mean. The Fréchet mean of $n$ anchors and weights $w_i$ over a metric space $\mathcal{M}$ with metric $d$ is defined as:

$$\hat{\mathbf{x}} = \operatorname*{argmin}_{\mathbf{x}\in\mathcal{M}} F(\mathbf{x}), \quad \text{where} \quad F(\mathbf{x}) = \sum_{i=1}^{n} w_i \, d(\mathbf{x}, \mathbf{x}_i)^2. \tag{2}$$

As long as $d$ is smooth, the gradient of $F(\mathbf{x})$ at the minimum $\hat{\mathbf{x}}$ must be zero:

$$\sum_{i=1}^{n} w_i \, \nabla d(\hat{\mathbf{x}}, \mathbf{x}_i)^2 = 0. \tag{3}$$

For Euclidean space, $\nabla d(\mathbf{x}, \mathbf{x}_i)^2 = \mathbf{x} - \mathbf{x}_i$ and it is easy to check that the Fréchet mean definition reduces to the usual one (1).

In general, the Fréchet mean is not always well-defined: $F(\mathbf{x})$ may have multiple minima. For example, on a sphere, consider anchors that are vertices of a platonic solid, and identical weights. If $d$ is the Euclidean metric of the ambient 3D space then $F$ is constant, and if $d$ is the geodesic metric on the sphere then $F$ has multiple minima due to symmetry. Even if $F$ has a unique minimum, its location may not be continuous in $w_i$ and $\mathbf{x}_i$, and Equation (3) can be satisfied at a local minimum. Karcher [1977] and Kendall [1990] among others have studied the conditions for which the Fréchet mean is well-behaved on Riemannian manifolds. At a high-level, if the anchors are close together, Gaussian curvature is not too high, and the weights are nonnegative, $\hat{\mathbf{x}}$ is well-defined and continuous in all variables. Our focus is on fast computation, and we leave it up to the user to ensure that there are enough close-by anchors; our experiments show that this is not difficult (see the accompanying video for an example of anchor placement).

While the Fréchet mean is usually used with geodesic distances, geodesics between arbitrary points on a mesh are slow to compute.

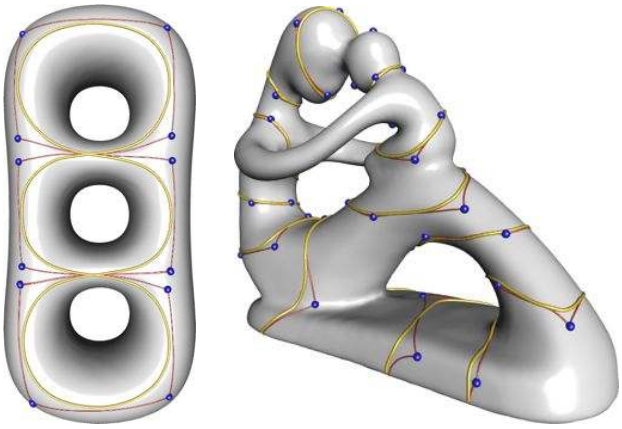


**Figure 2:** *Cubic B-splines on surfaces are shown as yellow curves. The red curve is a linear spline and illustrates the control polygon.*

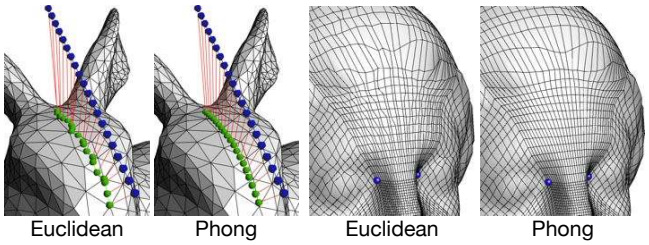



**Figure 3:** *Left: A line segment in 3D space projected onto a mesh using Euclidean and Phong projection. Right: Quad remeshing (see Section 4) using Euclidean and Phong projection in $\mathbb{R}^D$. Notice the jagginess of the Euclidean projections.*

A different class of metrics, that we call Euclidean-embedding metrics, has been gaining popularity in recent literature. A Euclidean-embedding metric is defined by an embedding $\mathbf{e} : \mathcal{M} \to \mathbb{R}^D$ and the distance between two points on the surface is the Euclidean distance between their embeddings $d(\mathbf{x}_1, \mathbf{x}_2) = \|\mathbf{e}(\mathbf{x}_1) - \mathbf{e}(\mathbf{x}_2)\|$. On a mesh, the embedding is defined on the vertices and is linear over each face interior. Euclidean-embedding metrics are very fast to evaluate, and if $\mathbf{e}$ is smooth, there is no $C^1$-discontinuous cut locus, unlike with geodesic distance. In this work, we always compute distances using a Euclidean embedding; see Section 3.5 for details on how this embedding is constructed.

For a Euclidean-embedding metric, the Fréchet mean has a simpler form. Letting $\mathbf{y}_i = \mathbf{e}(\mathbf{x}_i)$ and substituting the metric into the definition of $F$, we obtain:

$$F(\mathbf{x}) = \sum_{i=1}^{n} w_i \|\mathbf{e}(\mathbf{x}) - \mathbf{y}_i\|^2. \tag{4}$$

Letting $\bar{\mathbf{y}} = \sum_{i=1}^{n} w_i \mathbf{y}_i$ and using that $\sum w_i = 1$, we can simplify $F$ to:

$$\begin{aligned} F(\mathbf{x}) &= \sum_{i=1}^{n} \left( w_i \, \mathbf{e}(\mathbf{x})^T \mathbf{e}(\mathbf{x}) - 2\, w_i \, \mathbf{e}(\mathbf{x})^T \mathbf{y}_i + w_i \, \mathbf{y}_i^T \mathbf{y}_i \right) = \\ &= \mathbf{e}(\mathbf{x})^T \mathbf{e}(\mathbf{x}) - 2\, \mathbf{e}(\mathbf{x})^T \bar{\mathbf{y}} + \sum_{i=1}^{n} w_i \, \mathbf{y}_i^T \mathbf{y}_i = \\ &= \|\mathbf{e}(\mathbf{x}) - \bar{\mathbf{y}}\|^2 - \bar{\mathbf{y}}^T \bar{\mathbf{y}} + \sum_{i=1}^{n} w_i \, \mathbf{y}_i^T \mathbf{y}_i, \end{aligned} \tag{5}$$

where only the first term depends on $\mathbf{x}$. Thus, we have just shown that minimizing $F$ over the original surface in $\mathbb{R}^3$ is equivalent to minimizing the distance to $\bar{\mathbf{y}}$, the weighted average of the anchors in the embedding space $\mathbb{R}^D$. The computation of the Fréchet mean for a Euclidean-embedding metric thus consists of computing the Euclidean weighted average $\bar{\mathbf{y}}$ in the embedding space and projecting it onto the embedded mesh. Because a mesh is not smooth, Euclidean projection, and therefore the exact Fréchet mean, is not a smooth function of the weights or the anchor locations. We therefore use a different projection operator that provides a better-behaved weighted average.

**Smooth projection.** The Euclidean projection operator $\hat{\mathbf{y}} = \mathcal{P}_E(\bar{\mathbf{y}})$ that projects $\bar{\mathbf{y}}$ onto a mesh $\mathcal{M}$ is *discontinuous* at the medial axis of the mesh. Because, unlike for smooth surfaces, the medial axis of a mesh extends all the way to the mesh edges, the projection operator is discontinuous no matter how close $\bar{\mathbf{y}}$ is to the mesh. Even away from the medial axis, $\mathcal{P}_E$ has undesirable behavior, as shown in Figure 3.

Assuming that the mesh is approximating a smooth surface, one would like a projection operator with smoother behavior. One can use the fact that projection onto a smooth surface is always along the direction perpendicular to the surface. Kobbelt and colleagues [1999] propose projecting along a continuous normal field in 3D, as follows (see also [Botsch and Sorkine 2008] , Section IID): If $\hat{\mathbf{y}}$ is a projection of $\bar{\mathbf{y}}$, and $\mathbf{n}$ is the normal at $\hat{\mathbf{y}}$, then $\mathbf{n} \times (\hat{\mathbf{y}} - \bar{\mathbf{y}}) = \mathbf{0}$. For a mesh embedded in 3D one can take estimated surface normals at the mesh vertices and define a continuous surface normal on the mesh triangles using barycentric interpolation (as is commonly done for Phong shading). This simulates the behavior of projecting onto a smooth surface without actually having a smooth surface. Suppose that at vertex $i$ of a triangle $\mathbf{t}$ (for $i = 1, 2, 3$) the vertex position is $\mathbf{v}_i$ and the normal is $\mathbf{n}_i$. To project $\bar{\mathbf{y}}$, [Kobbelt et al. 1999] look for a point $\hat{\mathbf{y}}$ on the triangle $\mathbf{t}$ with barycentric coordinates $\xi_i$ that satisfy:

$$(\xi_1\mathbf{n}_1 + \xi_2\mathbf{n}_2 + \xi_3\mathbf{n}_3) \times (\xi_1\mathbf{v}_1 + \xi_2\mathbf{v}_2 + \xi_3\mathbf{v}_3 - \bar{\mathbf{y}}) = \mathbf{0}. \quad (6)$$

For a mesh embedded in $D$-dimensional space, the normal is not a vector, but a $(D-2)$-dimensional subspace, hence the above construction does not directly generalize. To make Phong projection work for a $D$-dimensional space, we use the fact that the normal at $\hat{\mathbf{y}}$ is the subspace orthogonal to the tangent plane at $\mathbf{y}$, and so a similar construction can be developed by interpolating the tangent planes at the triangle's vertices instead of normals. Note that it is similar, but not equivalent to the previous one in 3D, since linearly interpolating bases of tangent planes is different than interpolating the normals. Interpolating bases can, however, be more easily extended to work in higher dimensional spaces. Since the same tangent plane can be represented by an infinite number of bases, we must carefully select which basis we use for the linear interpolation to avoid degeneracies. The next section describes our Phong projection more formally; the supplemental material provides a more extended description.

### 3.2 Phong projection

We denote our triangle mesh as $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$ with embedded vertices $\mathcal{V} \subset \mathbb{R}^D$. Each vertex has an associated tangent plane (computed e.g. using the Loop [1987] limit stencil in $\mathbb{R}^D$), represented by two basis vectors, which we assume to be orthonormal. Denote the tangent planes at the vertices of a triangle $\mathbf{t} = (\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ as $T_1, T_2, T_3 \in \mathbb{R}^{2\times D}$. Let $\Psi(\xi_1, \xi_2, \xi_3) \in \mathbb{R}^{2\times D}$ be an interpolated tangent plane basis for the point $\xi_1\mathbf{v}_1 + \xi_2\mathbf{v}_2 + \xi_3\mathbf{v}_3$ inside $\mathbf{t}$, where $\xi_i$ are barycentric coordinates. $\Psi$ will be precisely defined later (Definition 3).

**Definition 1.** *A point $\hat{\mathbf{y}} = \xi_1\mathbf{v}_1 + \xi_2\mathbf{v}_2 + \xi_3\mathbf{v}_3$ on the triangle $\mathbf{t}$ having vertices $\mathbf{v}_i$ is a Phong projection of $\bar{\mathbf{y}} \in \mathbb{R}^D$ onto $\mathbf{t}$ if:*

$$\Psi(\xi_1, \xi_2, \xi_3)(\xi_1\mathbf{v}_1 + \xi_2\mathbf{v}_2 + \xi_3\mathbf{v}_3 - \bar{\mathbf{y}}) = \mathbf{0}, \quad (7)$$

$$\xi_1 + \xi_2 + \xi_3 = 1, \quad (8)$$

$$\xi_i \geq 0. \quad (9)$$

As previously discussed, a Phong projection is a point $\hat{\mathbf{y}}$ on the triangle, where the interpolated tangent plane is perpendicular to the line connecting $\hat{\mathbf{y}}$ and $\bar{\mathbf{y}}$. Note that this characterization corresponds to the Euclidean projection onto a smooth manifold.

**Definition 2.** *The Phong projection of a point $\bar{\mathbf{y}}$ onto a mesh $\mathcal{M}$ is the closest Phong projection with respect to every triangle of $\mathcal{M}$.*

Unlike Euclidean projection, the Phong projection onto a triangle may not exist or multiple points on a single triangle may be Phong

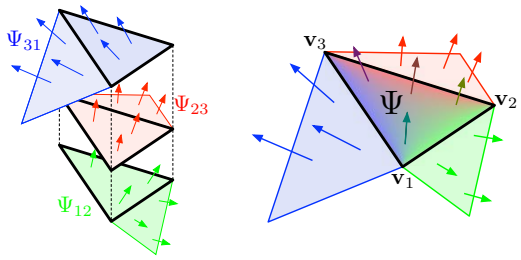



**Figure 4:** *We define a continuous blend for each pair of adjacent triangles (flaps) on the mesh (left). To compute $\Psi$ over the black triangle, we blend the interpolants on the overlapping flaps using the weights $1/\xi_1, 1/\xi_2, 1/\xi_3$ (right).*

projections of $\bar{\mathbf{y}}$. In some cases (Figure 5) there are points that have no Phong projection onto any triangle of a closed mesh. While such situations can be constructed, in all our experiments, for reasonably tessellated meshes, Phong projection does exist for points away from the medial surface. The supplemental material sketches out how a formal version of this claim might be proved.

We now turn to defining $\Psi$. We call two tangent plane bases $T, K \in \mathbb{R}^{2\times D}$ *equivalent* and write $T \equiv K$ if they represent the same tangent plane. We seek a continuous function $\Psi$ that interpolates the vertex tangent planes to the triangle interiors, i.e. $\Psi(1, 0, 0) \equiv T_1$, etc. Defining $\Psi$ is a non-trivial task, since the bases that describe a tangent plane are not unique, and the interpolant should be independent of the particular basis choice and also consistent on edges and vertices shared by multiple triangles. Furthermore, we want to keep the blending linear so that it can be easily inverted, but linearly blending bases of 2D subspaces is not guaranteed to always generate another basis of a 2D subspace. For example, $T$ and $-T$ may both be bases, but blending them with equal weights yields the zero matrix; our goal is to construct $\Psi$ while avoiding such scenarios.

We first define $\Psi$ for each mesh edge, based on the tangent planes of its end vertices. We then extend each on-edge $\Psi$ to interpolate tangent planes over the incident flap, i.e., both triangles adjacent to the edge. Hence, for each point inside a triangle, with barycentric coordinates $(\xi_1, \xi_2, \xi_3)$, we obtain three possible tangent planes, and we linearly blend between these tangent planes using blending weights $1/\xi_i$. See Figure 4 for an illustration of the construction. Starting from on-edge interpolation and using $1/\xi_i$ weights ensures continuity of $\Psi$ on mesh vertices and edges, as proved in the supplemental document.

The main intuition for our construction is that we have some degrees of freedom when choosing the bases representing the vertex tangent planes $T_i$, and we can choose the bases locally, per-triangle $\mathbf{t}$, when constructing $\Psi$ within $\mathbf{t}$. We will choose them such that they are "as close as possible" to each other in the sense of vectors in $\mathbb{R}^D$; this ensures (under some mild assumptions on $\mathcal{M}$) that their linear blending cannot degenerate. When considering two orthonormal bases $T_1, T_2 \in \mathbb{R}^{2\times D}$ for tangent planes of edge end vertices $\mathbf{v}_1, \mathbf{v}_2$, we can choose the basis $T_2$ for $\mathbf{v}_2$ and the basis $\mathrm{Ort}(T_2T_1^T)T_1$ for $\mathbf{v}_1$, where

$$\mathrm{Ort}(A) = \operatorname*{argmin}_{B \in O(2)} \|B - A\|_F. \quad (10)$$

Ort can be computed using polar decomposition, see the supplemental document. In this way, the tangent plane basis of $\mathbf{v}_1$ is brought as close as possible relative to that of $\mathbf{v}_2$, and linearly blending between them is "safe". We will thus have

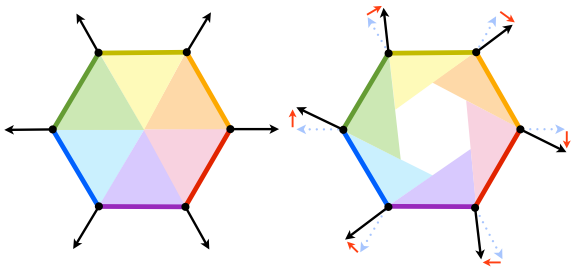


**Figure 5:** *In some cases, Phong projection may not exist. On the left, we discretize a circle as a hexagon with exact normals. The points in each colored region project to the corresponding edge. On the right, we slightly rotate all the normals clockwise. The white hexagon in the middle now consists of points that do not have a Phong projection.*

$\Psi_{12}(\xi_1, \xi_2, 0) \equiv \xi_1 R_{12} T_1 + \xi_2 T_2$, where $R_{12} = \mathrm{Ort}(T_2 T_1^T)$, and similarly $\Psi_{23}, \Psi_{31}$ for the other edges of a triangle $\mathbf{t}$.

We have an additional degree of freedom to exploit: the tangent plane bases at $\mathbf{v}_1, \mathbf{v}_2$ can both be rotated in-plane by the same $2 \times 2$ orthogonal matrix $E_{12}$ without changing the planes themselves and without affecting their linear blending, up to equivalence (and the same holds for the other edges). We choose the matrices $E_{12}, E_{23}, E_{31}$ such that the bases we linearly blend in the end for each point inside $\mathbf{t}$ are as close as possible to each other. Formally:

**Definition 3.** *Tangent plane interpolation:*

$$\Psi(\xi_1, \xi_2, \xi_3) = \frac{\xi_1 \xi_2 \xi_3}{\xi_1 \xi_2 + \xi_2 \xi_3 + \xi_3 \xi_1} \left( \frac{1}{\xi_3} \Psi_{12}(\xi_1, \xi_2, \xi_3) + \right. \quad (11)$$
$$\left. + \frac{1}{\xi_1} \Psi_{23}(\xi_1, \xi_2, \xi_3) + \frac{1}{\xi_2} \Psi_{31}(\xi_1, \xi_2, \xi_3) \right),$$

where

$$\Psi_{12}(\xi_1, \xi_2, \xi_3) = \xi_1 E_{12} R_{12} T_1 + \xi_2 E_{12} T_2 + \xi_3 \frac{1}{2} (E_{23} + E_{31} R_{31}) T_3.$$

The formulas for the other edge blends $\Psi_{23}, \Psi_{31}$ are obtained by cyclic permutation of the indices. $E_{ij} \in \mathbb{R}^{2 \times 2}$ are computed as

$$E_{12}, E_{23}, E_{31} = \underset{E_{12}, E_{23}, E_{31} \in O(2)}{\mathrm{argmin}} \sum_{1 \le i < j \le 6} \|A_i - A_j\|_F^2, \quad (12)$$

where $A_1 = E_{12} R_{12} T_1$, $A_2 = E_{12} T_2$, $A_3 = E_{23} R_{23} T_2$, $A_4 = E_{23} T_3$, $A_5 = E_{31} R_{31} T_3$, $A_6 = E_{31} T_1$. The $E$ matrices can be iteratively optimized by fixing two out of the three and solving a Procrustes problem for the third, which is guaranteed to decrease the energy; this optimization converges quickly. We refer to the supplemental for a more detailed explanation of the construction of $\Psi$ and for a formal proof of its continuity and interpolation properties.

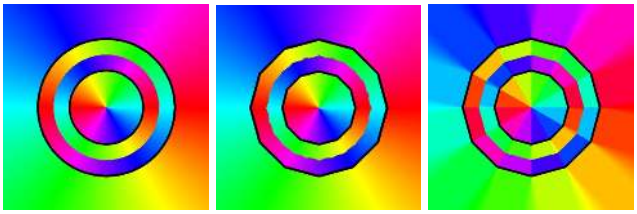
Phong projection has several useful properties:

1. Phong projection takes each point on the mesh to itself.

   *Proof.* When $\hat{\mathbf{y}}$ is on the mesh and $\xi_1, \xi_2, \xi_3$ are its barycentric coordinates, Equation (7) is trivially satisfied. □

2. Phong projection onto a plane is the same as Euclidean projection.

   *Proof.* For a discretized plane, all tangent planes and triangle planes are the same. $\Psi$ will be constant and identical to them: the resulting projection will thus simply be the Euclidean projection onto the triangle. □



(a) Eucl. projection onto a smooth surface (b) Phong projection onto a mesh (c) Euclidean projection onto a mesh

**Figure 6:** *Comparison of the qualitative behavior of (a) projection onto a smooth surface, (b) Phong projection, and (c) Euclidean projection onto a mesh. The smooth surface is a torus, which is discretized in (b) and (c). Each image shows a plane through the center of the torus; each point is colored based on the angle between the vector from that point to its projection and the $x$-axis. Phong projection behaves similarly to projecting onto the smooth torus, while Euclidean projection shows clear discontinuities.*

3. If the Euclidean projection of a point $\bar{\mathbf{y}}$ onto a tangent plane at vertex $\mathbf{v}$ is $\mathbf{v}$ itself, then $\mathbf{v}$ is a Phong projection of $\bar{\mathbf{y}}$.

   *Proof.* Assume w.l.o.g. that $\mathbf{v}$ is the first vertex in the triangle. Then $\Psi(1, 0, 0)$ is by definition the tangent plane at $\mathbf{v}$, and Equation 7 corresponds to Euclidean projection. □

4. Assume that a point $\hat{\mathbf{y}}$, which is on an edge shared by triangles $\mathbf{t}_1$ and $\mathbf{t}_2$, is a Phong projection of a point $\bar{\mathbf{y}}$ onto $\mathbf{t}_1$. Then $\hat{\mathbf{y}}$ is also a Phong projection of $\bar{\mathbf{y}}$ onto $\mathbf{t}_2$.

   *Proof.* Equation (7) is identical in both triangles up to an index permutation, and $\Psi$ is the same due to continuity. □

5. Given a point $\hat{\mathbf{y}}$ on a triangle, the set of points in ambient space for which $\hat{\mathbf{y}}$ is a Phong projection is an affine subspace.

   *Proof.* Equation (7) is linear in $\bar{\mathbf{y}}$, so its solution set is an affine subspace. □

The third and fourth properties explain why Phong projection is consistent on mesh vertices and edges, and are the reasons for our involved definition, and the fifth property allows us to solve the inverse problem with relative ease.

Phong projection on a triangle is computed by solving Equations (7) and (8), using Newton's method starting from $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$; the solution is discarded if Equation (9) is not satisfied.

We illustrate the advantage of Phong projection in Figures 3 and 6.

### 3.3 The forward problem

To compute the weighted average of $n$ anchors over a Euclidean-embedding metric, we first use the embedding to map them to $\mathbb{R}^D$ and compute their weighted average $\bar{\mathbf{y}}$ in Euclidean space. The remaining task is then to find the Phong projection of $\bar{\mathbf{y}}$ onto the embedded mesh. The simplest brute-force approach is to compute the Phong projection onto every triangle and pick the closest point, but this is very expensive. Acceleration structures, such as KD-trees or bounding volume hierarchies are typically used to speed up Euclidean projection, but they scale poorly to higher dimensions.

Instead, we use a simple local search approach, based on the fact that Euclidean and Phong projections are fairly close. We start with an initial guess vertex, for example, the vertex closest to the projection from a previous computation, or the anchor with the highest weight, if a better initial guess is not available. We then walk over the edge graph, greedily choosing edges that get us closer to $\bar{\mathbf{y}}$,

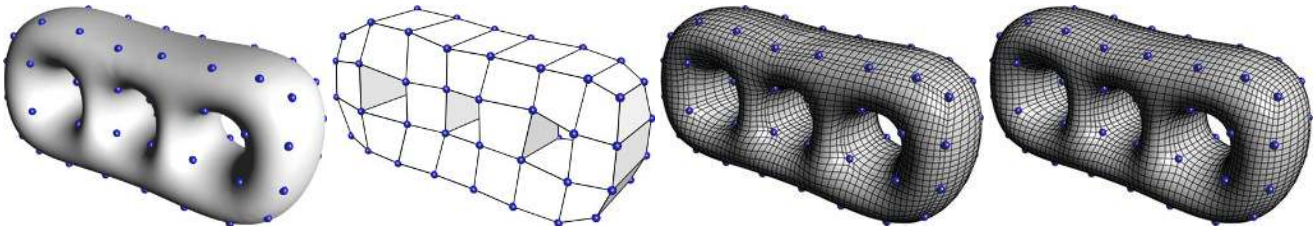


**Figure 7:** *Example of remeshing using weighted averages. From left to right: the original mesh, a coarse quad mesh, a uniformly subdivided quad mesh (without Laplacian smoothing), and the final result after 5 steps of Laplacian smoothing on the surface.*

stopping when we hit a vertex closer to $\bar{\mathbf{y}}$ than all of its neighbors. From that vertex, we do a breadth-first search over triangles, trying to find a Phong projection on each triangle and stopping as soon as we find a triangle for which a Phong projection exists.

This method is not guaranteed to find the Phong projection, but it fails when there are multiple close projection candidates and the weighted average is therefore meaningless. We compared the results of this method to the brute force solution for all of the forward problems in Figure 7 and the projections are all exactly the same. For the forward problems in Figure 12, the covering by anchors is not perfect and our method produced different results on 2.5 percent of the pixels, with a maximum error of 7 times the average length of a mesh edge.

### 3.4 The inverse problem

For many applications, we need to be able to compute, for given anchors and a given point on the surface $\hat{\mathbf{x}}$, a set of weights $\mathbf{w} = (w_1, \ldots, w_n)^T$ for which $\hat{\mathbf{x}}$ is the weighted average (the weights must sum to one). In Euclidean space, finding such weights is known as the generalized barycentric coordinates problem. Rustamov [2010] solves this problem for the Fréchet mean with geodesic distances, but we need to be able to invert our formulation to be consistent with the forward problem.

There are many ways to define the generalized barycentric coordinates, since Equation (3) has multiple solutions. The particular solution needs to satisfy three properties: locality, interpolation, and smoothness. Locality means that the anchors that are far away from $\hat{\mathbf{x}}$ should have very low weights. This is important for most applications shown in Section 4 because it restricts the effect of every anchor to a small region around it. Interpolation means that when $\hat{\mathbf{x}}$ coincides with an anchor, the weight for all other anchors should be zero. Finally smoothness means that small changes to anchors and $\hat{\mathbf{x}}$ result in small changes to the weights. Many existing generalized barycentric coordinates satisfy these properties, but they assume that the anchors are given as a simple polygon. For our applications we cannot use such schemes because the polygon would have to change its connectivity in order to remain simple as the anchors move, violating smoothness.

Instead, we propose a new set of generalized barycentric coordinates, obtained by simply treating the Phong projection and partition of unity as linear constraints and solving a quadratic optimization problem that penalizes weights for far-away anchors. Given a point on a triangle $\mathbf{t}$ with barycentric coordinates $\xi$, we solve:

$$\begin{aligned} \arg\min_{\mathbf{w}} \ & \|\mathbf{D}\,\mathbf{w}\|^2 \qquad (13) \\ s.t. \ & \Psi(\xi_1,\xi_2,\xi_3)(\xi_1\mathbf{v}_1 + \xi_2\mathbf{v}_2 + \xi_3\mathbf{v}_3 - \sum_i w_i\mathbf{y}_i) = \mathbf{0}, \\ & \sum_i w_i = 1. \end{aligned}$$

$\mathbf{D}$ is a diagonal matrix that contains a weighting term for every anchor. This is a standard linearly-constrained minimum-norm problem, which we solve using the pseudoinverse.

We are left with the problem of defining $\mathbf{D}$. Note that for far-away anchors whose weights should be low, $\mathbf{D}_{i,i}$ should be high. Additionally, we penalize anchors far away from the tangent plane more because they are likely to be less reliable. Given the locations (in the embedded mesh) of the anchors, $\mathbf{y}_i = \mathbf{e}(\mathbf{x}_i)$, and of the input surface point, $\hat{\mathbf{y}} = \mathbf{e}(\hat{\mathbf{x}})$, we split the vector $\mathbf{y}_i - \hat{\mathbf{y}}$ into the component $\mathbf{v}_T$ in the tangent plane at $\hat{\mathbf{y}}$ and the component $\mathbf{v}_O$ orthogonal to the tangent plane. Using $\mathbf{D}_{i,i} = \|\mathbf{v}_T\|^2 + \gamma\|\mathbf{v}_O\|^2$ guarantees interpolation: if a given point on the surface is also an anchor $i$, then $\mathbf{D}_{i,i} = 0$ and the weight combination that assigns $w_i = 1$ and $w_j = 0$ for all other anchors satisfies the constraints and sets the objective in (13) to zero; the quadratic nature of the optimization problem guarantees uniqueness of this solution. We have found that this definition of $\mathbf{D}$ works well for a wide range of $\gamma$ (we use 10 for all examples).

This formulation is a weighted variant of Waldron's method [2011], which, without $\mathbf{D}$, produces weights that are not interpolatory. If one removes the Phong projection constraints, our weights reduce to Moving Least Squares (see the supplemental material for the derivation). Our weights are $\mathcal{C}^1$ on a smooth manifold, since the corresponding MLS formulation generates $\mathcal{C}^1$ weights and we are only adding additional linear constraints (on piecewise-linear meshes the weights are, of course, $\mathcal{C}^0$ on the mesh edges).

Maximum Entropy Coordinates [Hormann and Sukumar 2008] satisfy all our desiderata, if the inverse of the squared distances is used as a prior. They lead to similar results to our weights, but are too expensive to compute for our purposes, since they require minimizing a nonlinear energy with Newton's method.

The accompanying video shows examples of our weights on a surface.

### 3.5 Euclidean-embedding metric

We now discuss how we construct a Euclidean-embedding metric suitable for our purposes. Lipman et al. [2010] compare several alternative metrics: diffusion distance, commute-time distance, and their proposed biharmonic distance. These metrics are defined in terms of eigenvectors of the Laplace-Beltrami operator, which can behave poorly, as small features can collapse in the embedding (Figure 8). Instead, we use a metric that is based on Rustamov and colleagues' [2009] boundary embedding construction.

Rustamov et al. [2009] embed a mesh by computing the geodesic distances between all pairs of points and using Metric Multidimensional Scaling (MMDS) [Cox and Cox 2000] . This approach does not scale, as constructing and working with the distance matrix for a mesh with 100,000 vertices is impractical: even storing such a matrix would require just under 20 GB in single precision.

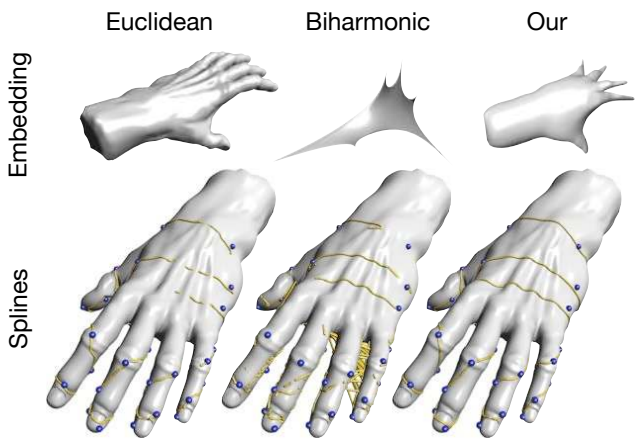



**Figure 8:** *Top: The original surface (Euclidean embedding) and the first three dimensions of the biharmonic embedding and our embedding for the hand mesh. The biharmonic embedding tends to collapse thin features, making it unsuitable for weighted averages computations on these regions. Bottom: Cubic B-splines using different Euclidean-embedding metrics. Discontinuities in the weighted averages computed with Euclidean and Biharmonic embeddings can be seen as both the yellow curve passing through the surface (to connect to a far-away point) and as artifacts on the fingers. All of these results were computed using Phong projection.*

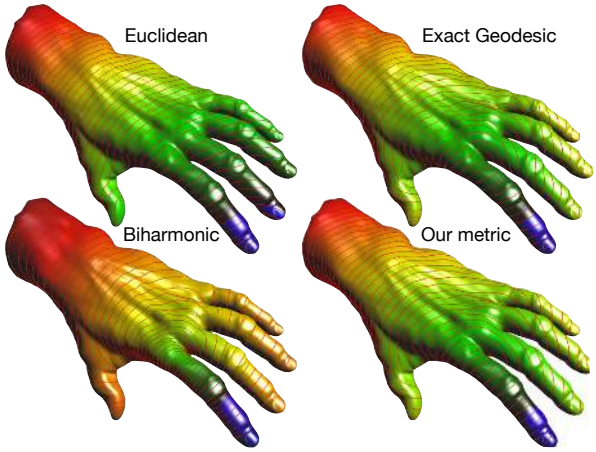



**Figure 9:** *Distance and isolines from a point on the tip of the index finger using different metrics. Due to the collapse shown in Fig. 8, the isolines of biharmonic distance are aligned with the fingers and do not discriminate between different points around each finger.*

The IsoCharts method [Zhou et al. 2004] employs Landmark MDS [de Silva and Tenenbaum 2002] to efficiently compute a Euclidean embedding into higher dimensions for spectral mesh segmentation and parameterization applications. We take a conceptually similar landmark approach and adapt it to our purposes to ensure a smooth embedding. We sample a representative subset $\mathcal{S}$ of the mesh vertices, compute approximate pairwise geodesic distances only for these vertices and embed $\mathcal{S}$ into $\mathbb{R}^D$ using MMDS. We then use least-squares meshes [Sorkine and Cohen-Or 2004] to embed the remaining mesh vertices, creating a mesh in $\mathbb{R}^D$ whose triangles are shaped similarly to the original mesh.

We find the subset of mesh vertices $\mathcal{S}$ by decimating the mesh using an algorithm from OpenMesh [Botsch et al. 2002] that iteratively collapses the halfedge that changes the face normals least. This way, the subsequent LS-mesh upsampling will have an easier time correctly approximating areas with low curvature. When a user-specified number of points (we use 1000) is left, the decimation stops and these vertices are used as $\mathcal{S}$.

To embed the samples $\mathcal{S}$, we use the fast marching method [Sethian 1996] to construct the dissimilarity matrix $\mathbf{Q}$: $\mathbf{Q}_{i,j} = d_{\text{geo}}(\mathbf{s}_i, \mathbf{s}_j)$ . We use the implementation of MMDS in MATLAB 11, minimizing the following stress criterion:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\left(1 - \frac{\|\mathbf{e}(\mathbf{s}_i) - \mathbf{e}(\mathbf{s}_j)\|}{\mathbf{Q}_{i,j}}\right)^2, \tag{14}$$

where $\mathbf{e}(\mathbf{s}_i) \in \mathbb{R}^D$ is the embedding of $\mathbf{s}_i$. We use $D$=8 for all examples. We have not observed any improvement in quality above eight dimensions, and as 8 floats fit into an AVX register, we would gain little performance improvement from using fewer dimensions.

Denote the set of all original mesh vertices by $\tilde{\mathcal{V}} = \{\tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_N\}$, and the final embedded version of each $\tilde{\mathbf{v}}_i$ by $\mathbf{v}_i \in \mathbb{R}^D$. To complete the embedding $\mathbf{v} \in \mathbb{R}^{N\times D}$ of the remaining vertices $\tilde{\mathcal{V}}\backslash\mathcal{S}$, we compute a smooth mesh in the $D$-dimensional space as a least-squares mesh:

$$\begin{aligned} &\mathbf{v} = \underset{\mathbf{v}}{\operatorname{argmin}}\ \operatorname{tr}\left(\mathbf{v}^T\left(\mathbf{L}\mathbf{M}^{-1}\mathbf{L}\right)\mathbf{v}\right) \\ &s.t.\ \ \mathbf{v}_i = \mathbf{e}(\mathbf{s}_i),\ \ \forall i \in \mathcal{S}, \end{aligned} \tag{15}$$

where $\mathbf{L}$ and $\mathbf{M}$ are the stiffness matrix (cotangent Laplacian) and the lumped mass matrix of the original mesh, respectively [Botsch and Sorkine 2008]. The optimization in (15) amounts to solving a sparse linear system and we use MATLAB's built-in solver.

We compare our metric with Euclidean, biharmonic and exact geodesic distance [Surazhsky et al. 2005] in Figure 9. Our metric is smooth everywhere, while still providing a good approximation of geodesic distance. In Figure 8, we show results of splines defined using our algorithm with three different metrics. The overall precomputation usually takes a few minutes (Table 1).

## 4 Results

Our method was timed on a quad core 2.6 GHz Intel Core i7 processor, using all four cores to solve weighted averages problems in parallel. The software is written in C++, manually optimized using AVX intrinsics and OpenMP, and compiled using the Intel Compiler. The GPU was only used for rendering and did not participate in timings. Table 1 shows the timings for forward and inverse problems on our meshes. The inverse problem timings appear sublinear in the number of anchors due to a constant time component for constructing the constraint matrix and solving. The performance of the forward problem is more complicated, depending on mesh structure and anchor placement more than on vertex count.

Below we present some of the possible applications of weighted averages on surfaces.

### 4.1 Forward problem

The following applications only make use of the forward problem.

**Splines on surfaces.** Most splines can be defined in terms of weighted averages: for example, a point on a B-spline is the weighted average of control points using the basis functions as

| Mesh | Faces | Anchors | Forw. | Inverse | Pre. |
|---|---|---|---|---|---|
| pig | 8016 | 27 | 0.69 $\mu$s | 0.87 $\mu$s | 1.1 m |
| 3Holes | 28800 | 64 | 0.64 $\mu$s | 1.26 $\mu$s | 1.2 m |
| lizard | 39036 | 93 | 0.85 $\mu$s | 1.53 $\mu$s | 1.9 m |
| chubby | 40808 | 93 | 0.85 $\mu$s | 1.57 $\mu$s | 2.0 m |
| dog | 50528 | 93 | 1.54 $\mu$s | 1.60 $\mu$s | 2.7 m |
| horse | 58616 | 100 | 1.64 $\mu$s | 1.62 $\mu$s | 2.0 m |
| boy | 149996 | 93 | 1.71 $\mu$s | 1.65 $\mu$s | 7.7 m |

**Table 1:** *We timed our method by running 100,000 random points, finding their weights with respect to a fixed set of anchors, and then solving the forward problems with those weights (to arrive back at the original points). The computations were done in four parallel threads; the table shows the average per vertex times. In this way, we can solve approximately 300,000 forward and inverse problems per second on reasonably-sized meshes. For these timings we do not use an initial guess.*

weights. By replacing averages in Euclidean space with the Fréchet mean and using our computation, splines on surfaces become easy to define and fast to compute. Figure 2 shows examples using cubic B-splines. As shown in our video, splines can be edited interactively by dragging control points, and we use the results of the previous frame as the initial guesses for the forward problem.

**Semiregular remeshing and Laplacian smoothing on the surface.** Given a coarse base quadrilateral mesh, it is possible to produce a finer quad mesh by splitting every quadrilateral into four. If the coarse base mesh is defined on a surface, we can perform the regular subdivision directly on the mesh by finding the position of every newly inserted vertex using weighted averages. When the refined mesh is generated, a few steps of Laplacian smoothing (moving each vertex to the average of its neighbors on the surface) produces a high-quality quadrilateral mesh. The points of the original mesh can be moved interactively (again, the previous frame is used as an initial guess), and as an anchor is dragged, the resulting fine quadrangulation is generated at 8.5 fps for the final (rightmost) example in Figure 7 (35 fps without Laplacian smoothing).

### 4.2 Inverse problem

**Local surface parametrization.** As shown by Rustamov [2010], the solution to the inverse problem can be used to compute decals [Schmidt et al. 2006] on the surface. As Figure 10 shows, our method produces smooth results even in the presence of noise.

### 4.3 Forward and inverse problems

By combining the inverse and forward problems and using corresponding sets of anchors, we can establish a correspondence between two surfaces, or from a surface to itself. To map a point from the first surface to the second, we use the inverse problem to find weights with respect to the anchor set and then solve the forward problem on the second surface with those weights. Such correspondences are useful for many applications. We measure the symmetrized $L^2$ stretch distortion [Schreiner et al. 2004] of every mapping. We present the result for the whole surface, parts of which are incorrectly transferred due to small areas not covered by the anchors, presence of holes (Figure 12) and ambiguity of the Fréchet mean projection. Since we do not have a robust way of determining where this happens, we also report the average on the best 95% of the triangles, discarding the rest as outliers.

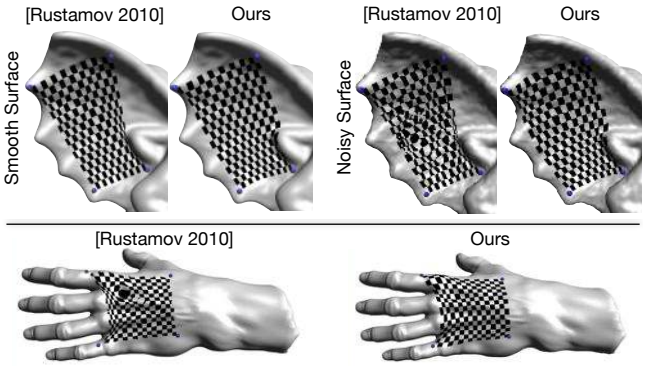



**Figure 10:** *The inverse problem with four anchors defines a mapping from mesh vertices onto a checkerboard, which is used as a texture. For a smooth mesh, our method and Rustamov's [2010] perform similarly, but when the mesh is corrupted by a small amount of noise, our metric is more robust than exact geodesic distance. The symmetrized $L^2$ stretch [Schreiner et al. 2004] of the mapping on the smooth wing is slightly worse with our method (0.75 compared to 0.88 for [Rustamov 2010]), the ideal measure being 1. Similarly for the hand, our mapping has a stretch of 0.74 compared to 0.83.*

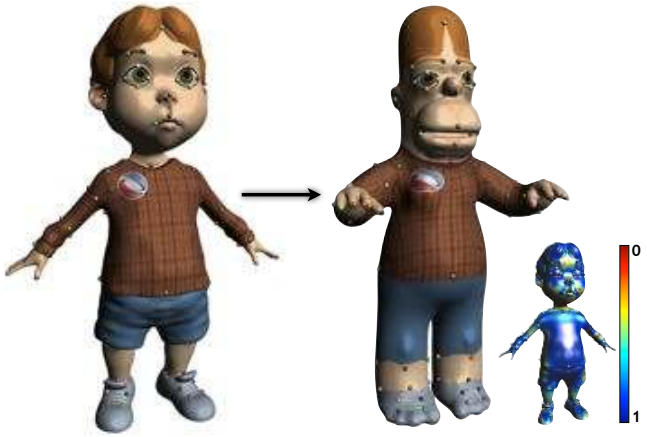



**Figure 11:** *Together, the forward and inverse problem are used to establish a correspondence between surfaces. Here this correspondence is used to transfer textures from the boy to the chubby man. The 93 anchors are shown in corresponding colors. For this texture, 531,326 texels were transferred in 1.18s. The color plot shows the symmetrized $L^2$ stretch. The distortion for the whole surface is 0.05, with 95% of the triangles having an average distortion of 0.40. 10k faces are flipped during the mapping, covering an area of 2.8% of the target surface.*

**Texture transfer.** An obvious use of surface correspondence is to transfer texture (or skinning weights, scattering coefficients, or any other quantity) from a textured model onto another shape that has a texture atlas. For each point on the target shape's texture that is mapped onto some point on the target shape, we find that point, find which point on the source shape it corresponds to using weighted averages, and then sample the source texture at that point. For medium resolution textures, our algorithm is interactive, providing the user with quick feedback when specifying the anchors. We currently sample the original texture bilinearly, but mipmapping and supersampling could be used for a higher-quality transfer. An implementation of our algorithm in a GPU fragment shader could compute this mapping on the fly, but we leave this as

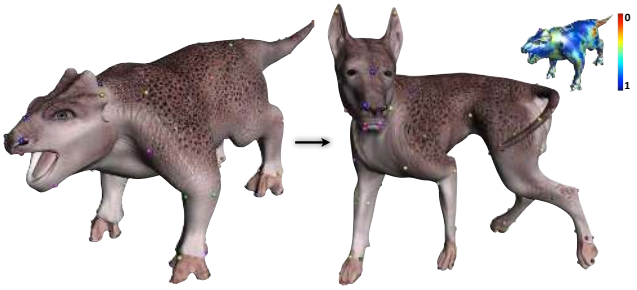


**Figure 12:** *Here the texture is transferred "flipped" because the 93 user-placed anchors on the dog are a mirror-reflection. The correspondence set up by weighted averages does not require the orientations to be preserved. The holes in the dog's eyes do not impede the transfer because only the inverse problem is solved on the dog. The color plot shows the symmetrized $L^2$ stretch. The distortion for the whole surface is 0.12, with 95% of the triangles having an average distortion of 0.32. 2.8k faces are flipped during the mapping, covering an area of 2.5% of the target surface.*

future work. For target shapes without readily available texture atlases we used [Sorkine et al. 2002] to compute them. Figures 11 and 12 demonstrate two texture transfers, between humanoids and quadrupeds, respectively. By changing the anchor positions, the texture can be deformed, as shown in Figure 13.

**Mesh morphing.** Another straightforward application of a mapping between two surfaces is mesh morphing. We show an example in Figure 14, where a lizard is morphed into a horse. The target horse has the connectivity of the lizard model and the vertex positions are the vertex positions on the lizard mapped through our correspondence onto the horse. We computed the morphing sequence using patch-based linear rotation-invariant coordinates [Baran et al. 2009]. We used 98 anchors to define the map.

### 4.4 Limitations

Under certain conditions (such as a Riemannian metric of nonpositive Gaussian curvature or anchors sufficiently close together), $F$ has no local minima other than the global minimum, but for real-world meshes, we cannot expect this. Our initial guesses generally get us to the global minimum, but we cannot guarantee this, and synthetic counterexamples can be constructed. Additionally, for sets of anchors that do not cover the surface well, there are likely to be regions on the surface that are not weighted averages of the anchors for any weights (Figure 15). For these regions, the weights produced by the inverse problem are meaningless. To detect this problem and guide the user in positioning the anchors (e.g., for the target surface in a correspondence), for every mesh vertex, we compute the inverse problem followed by the forward problem and only consider the correspondence to be defined if we end up back where we started. We also do not use the correspondence on mesh vertices for which there are large negative weights with respect to the current anchors.

When we have corresponding anchors on two surfaces, $A$ and $B$ and define a correspondence from $A$ to $B$ by solving the inverse problem on $A$ and the forward problem on $B$, we can also define the symmetric correspondence from $B$ to $A$. A subtle but important point is that even though both correspondences may be bijective, they will not, in general, be inverses of each other. In fact, we do not know an efficient method for inverting these correspondences. Furthermore, when a new anchor point is added, the mapping might change locally. It might be possible to optimize the insertion of the new anchor to reduce the amount of distortion introduced. In practice, this is not a serious problem since correspondences are placed interactively.

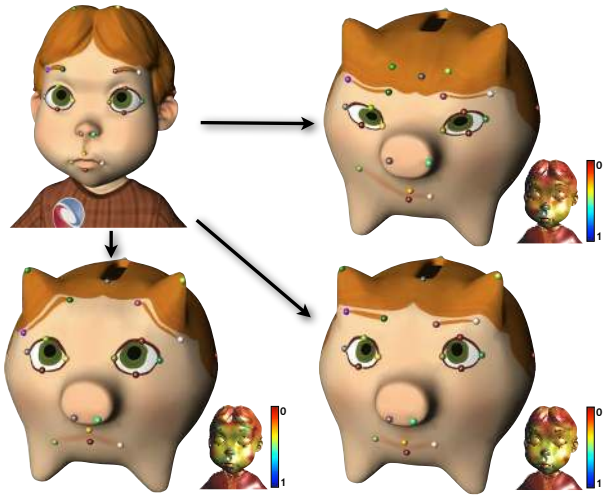


**Figure 13:** *Example of texture transfer and deformation between a humanoid character and a piggybank model. This transfer uses 27 anchors. The full-resolution transfer computes the correspondence for 2,878,322 texels and takes 3.6 seconds. The user places the markers using a lower-resolution texture for interactive feedback. The color plots show the $L^2$ stretch of the mapping.*

Compared to dedicated user-guided cross-parameterization methods [Kraevoy and Sheffer 2004; Schreiner et al. 2004] our method requires several times as many anchors and does not optimize stretch or conformality. However, in many cases, it is more valuable to be able to quickly see and adjust the mapping than to wait for a more automatic algorithm to execute.

## 5 Conclusion and future work

We presented an efficient method for computing and inverting weighted averages on surfaces. We demonstrated its usefulness as a building block for several important applications, enabling solutions that are much faster than previously possible. We have by no means exhausted the space of possible applications. For example, we plan to experiment with symmetric painting and mass transport interpolation over a surface [Bonneel et al. 2011].

Exploring the space of metrics based on Euclidean embeddings is another interesting challenge. Our method is an attempt at a smooth approximation of geodesic distance that can be computed extremely quickly between arbitrary points. Better embedding algorithms and embedding into more powerful norms (such as $L_\infty$) provide interesting directions for future research.

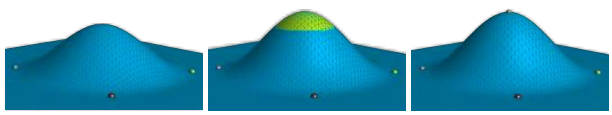


**Figure 15:** *For a relatively small bump (left) weighted averages of the four anchors (one is obscured) cover the bump. As the bump grows larger, no weighted average of the anchors reaches the top (middle). Placing an additional anchor at the bump top (right) allows the entire surface to be covered by weighted averages again.*

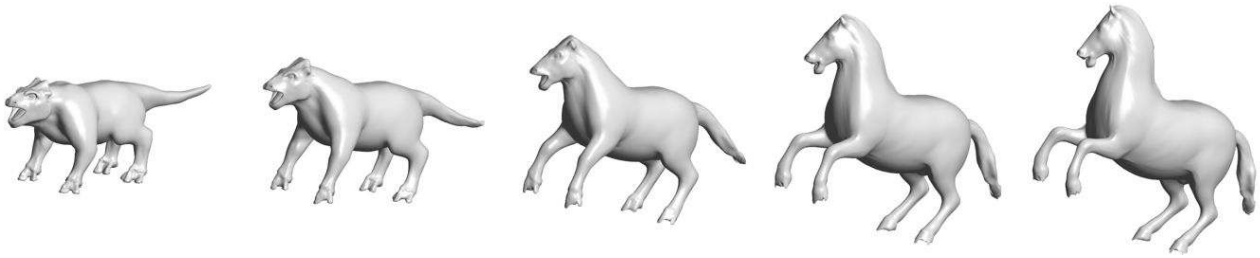


**Figure 14:** *A morph between a lizard and a horse, using a dense correspondence computed with weighted averages. The symmetrized $L^2$ stretch for the whole surface is 0.09, with 95% of the triangles having an average distortion of 0.46. 2.1k faces are flipped during the mapping, covering an area of 2.6% of the target surface.*

## Acknowledgements

We thank the anonymous reviewers for their insightful comments and Emily Whiting for narrating the accompanying video. The Boy model in Figure 11 was kindly provided by Maurizio Nitti. This work was supported in part by the ERC grant iModel (StG-2012-306877), by an SNF award 200021_137879 and a gift from Adobe Research.

## References

ALEXA, M. 2002. Linear combination of transformations. *ACM Trans. Graph. 21*, 3, 380–387.

BARAN, I., VLASIC, D., GRINSPUN, E., AND POPOVIĆ, J. 2009. Semantic deformation transfer. *ACM Trans. Graph. 28*, 3.

BONNEEL, N., VAN DE PANNE, M., PARIS, S., AND HEIDRICH, W. 2011. Displacement interpolation using lagrangian mass transport. *ACM Trans. Graph. 30*, 6.

BOTSCH, M., AND SORKINE, O. 2008. On linear variational surface deformation methods. *IEEE Trans. Visualization and Computer Graphics 14*, 1, 213–230.

BOTSCH, M., STEINBERG, S., BISCHOFF, S., AND KOBBELT, L. 2002. OpenMesh - a generic and efficient polygon mesh data structure. In *Proc. OpenSG Symposium*.

BOUBEKEUR, T., AND ALEXA, M. 2008. Phong tessellation. *ACM Trans. Graph. 27*, 5, 141:1–141:5.

BUSS, S. R., AND FILLMORE, J. P. 2001. Spherical averages and applications to spherical splines and interpolation. *ACM Trans. Graph. 20*, 2, 95–126.

CARTAN, É. 1929. Groupes simples clos et ouverts et géométrie riemannienne. *J. Math. Pures Appl. 8*, 1–33.

CHEN, Y., AND MEDIONI, G. 1991. Object modeling by registration of multiple range images. In *Proc. IEEE International Conference on Robotics and Automation*, 2724–2729.

COX, T. F., AND COX, M. A. A. 2000. *Multidimensional Scaling, Second Edition*. Chapman & Hall/CRC, Sept.

CRANE, K., WEISCHEDEL, C., AND WARDETZKY, M. 2013. Geodesics in heat. *ACM Trans. Graph.*. to appear.

DE SILVA, V., AND TENENBAUM, J. B. 2002. Global versus local methods in nonlinear dimensionality reduction. In *Proc. NIPS*, 705–712.

ECKSTEIN, I., SURAZHSKY, V., AND GOTSMAN, C. 2001. Texture mapping with hard constraints. *Comput. Graph. Forum 20*, 3, 95–104.

FLOATER, M. S. 2003. Mean value coordinates. *Computer Aided Geometric Design 20*, 1, 19–27.

FRÉCHET, M. 1948. Les éléments alétoires de nature quelconque dans un espace distancié. *Ann. Inst. H. Poincaré 10*, 4, 215–310.

HOFER, M., AND POTTMANN, H. 2004. Energy-minimizing splines in manifolds. *ACM Trans. Graph. 23*, 3, 284–293.

HORMANN, K., AND SUKUMAR, N. 2008. Maximum entropy coordinates for arbitrary polytopes. In *Proc. SGP*, 1513–1520.

HORMANN, K., POLTHIER, K., AND SHEFFER, A. 2008. Mesh parameterization: Theory and practice. In *SIGGRAPH ASIA 2008 Course Notes*.

JIN, J., GARLAND, M., AND RAMOS, E. A. 2009. MLS-based scalar fields over triangle meshes and their application in mesh processing. In *Proc. ACM I3D*, 145–153.

JOSHI, P., MEYER, M., DEROSE, T., GREEN, B., AND SANOCKI, T. 2007. Harmonic coordinates for character articulation. *ACM Trans. Graph. 26*, 3, 71:1–71:9.

JU, T., SCHAEFER, S., AND WARREN, J. 2005. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph. 24*, 3, 561–566.

KARCHER, H. 1977. Riemannian center of mass and mollifier smoothing. *Communications on pure and applied mathematics 30*, 5, 509–541.

KENDALL, W. 1990. Probability, convexity, and harmonic maps with small image I: uniqueness and fine existence. *Proceedings of the London Mathematical Society 3*, 2, 371.

KIM, V. G., LIPMAN, Y., AND FUNKHOUSER, T. 2011. Blended intrinsic maps. *ACM Trans. Graph. 30*, 4.

KOBBELT, L., VORSATZ, J., AND SEIDEL, H.-P. 1999. Multiresolution hierarchies on unstructured triangle meshes. *Comput. Geom. Theory Appl. 14*, 1-3, 5–24.

KRAEVOY, V., AND SHEFFER, A. 2004. Cross-parameterization and compatible remeshing of 3D models. *ACM Trans. Graph. 23*, 3, 861–869.

LANGER, T., BELYAEV, A., AND SEIDEL, H.-P. 2006. Spherical barycentric coordinates. In *Proc. SGP*, 81–88.

LIPMAN, Y., KOPF, J., COHEN-OR, D., AND LEVIN, D. 2007. GPU-assisted positive mean value coordinates for mesh deformations. In *Proc. SGP*, 117–124.

LIPMAN, Y., RUSTAMOV, R. M., AND FUNKHOUSER, T. A. 2010. Biharmonic distance. *ACM Trans. Graph. 29*, 3.

LOOP, C. 1987. *Smooth subdivision surfaces based on triangles*. Master's thesis, Department of Mathematics, University of Utah.

OVSJANIKOV, M., MÉRIGOT, Q., MÉMOLI, F., AND GUIBAS, L. J. 2010. One point isometric matching with the heat kernel. *Comput. Graph. Forum 29*, 5, 1555–1564.

PÁLFIA, M. 2009. The Riemann barycenter computation and means of several matrices. *Int. J. Comput. Math. Sci. 3*, 3, 128–133.

PENNEC, X. 1998. Computing the mean of geometric features: Application to the mean rotation. Rapport de Recherche RR–3371, INRIA - Epidaure project, Sophia Antipolis, France, March.

PHONG, B. 1975. Illumination for computer generated pictures. *Communications of the ACM 18*, 6, 311–317.

RITSCHEL, T., THORMÄHLEN, T., DACHSBACHER, C., KAUTZ, J., AND SEIDEL, H.-P. 2010. Interactive on-surface signal deformation. *ACM Trans. Graph. 29*, 4.

RUSTAMOV, R., LIPMAN, Y., AND FUNKHOUSER, T. 2009. Interior distance using barycentric coordinates. *Comput. Graph. Forum 28*, 5.

RUSTAMOV, R. 2010. Barycentric coordinates on surfaces. *Comput. Graph. Forum 29*, 5, 1507–1516.

SANDER, P. V., GU, X., GORTLER, S. J., HOPPE, H., AND SNYDER, J. 2000. Silhouette clipping. In *Proc. ACM SIGGRAPH*, 327–334.

SCHMIDT, R., GRIMM, C., AND WYVILL, B. 2006. Interactive decal compositing with discrete exponential maps. *ACM Trans. Graph. 25*, 3, 605–613.

SCHREINER, J., ASIRVATHAM, A., PRAUN, E., AND HOPPE, H. 2004. Inter-surface mapping. *ACM Trans. Graph. 23*, 3.

SETHIAN, J. A. 1996. A fast marching level set method for monotonically advancing fronts. In *Proc. Nat. Acad. Sci*, 1591–1595.

SORKINE, O., AND COHEN-OR, D. 2004. Least-squares meshes. In *Proc. Shape Modeling International*, 191–199.

SORKINE, O., COHEN-OR, D., GOLDENTHAL, R., AND LISCHINSKI, D. 2002. Bounded-distortion piecewise mesh parameterization. In *Proc. IEEE Visualization*, 355–362.

SUMNER, R. W., AND POPOVIĆ, J. 2004. Deformation transfer for triangle meshes. *ACM Trans. Graph. 23*, 3, 399–405.

SURAZHSKY, V., SURAZHSKY, T., KIRSANOV, D., GORTLER, S. J., AND HOPPE, H. 2005. Fast exact and approximate geodesics on meshes. *ACM Trans. Graph. 24*, 3, 553–560.

TZUR, Y., AND TAL, A. 2009. FlexiStickers: Photogrammetric texture mapping using casual images. *ACM Trans. Graph. 28*, 3.

WALDRON, S. 2011. Affine generalised barycentric coordinates. *Jaen Journal on Approximation 3*, 2.

WALLNER, J., AND POTTMANN, H. 2006. Intrinsic subdivision with smooth limits for graphics and animation. *ACM Trans. Graph. 25*, 2, 356–374.

XIN, S.-Q., YING, X., AND HE, Y. 2012. Constant-time all-pairs geodesic distance query on triangle meshes. In *Proc. ACM I3D*.

YEH, I.-C., LIN, C.-H., SORKINE, O., AND LEE, T.-Y. 2011. Template-based 3D model fitting using dual-domain relaxation. *IEEE Trans. Vis. Comput. Graph. 17*, 8, 1178–1190.

ZHOU, K., SYNDER, J., GUO, B., AND SHUM, H.-Y. 2004. Isocharts: stretch-driven mesh parameterization using spectral analysis. In *Proc. SGP*, ACM, New York, NY, USA, 45–54.