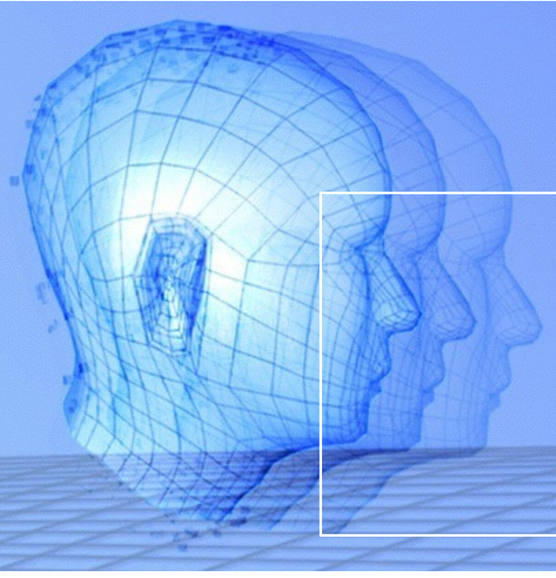


International Convention on SPM / SMI 2014

SPM / SMI 2014 Poster

Edge Length Interpolation

Carlos Rojas, Alex Tsui, S. He, L. Simons, S. Li, N. Amenta
Computer Science, University of California, Davis

Problem

- Input: set of meshes with same triangulation
  - Interpolate edge lengths for target mesh
  - Embed target mesh in $\mathbb{R}^3$

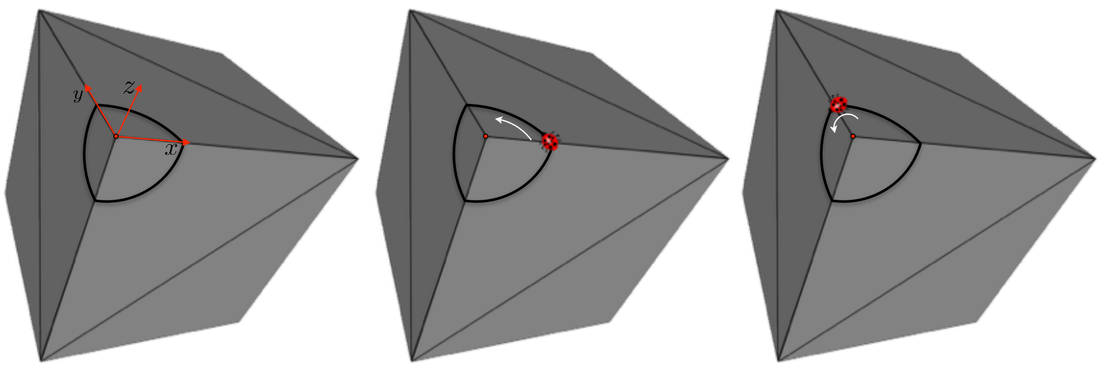
Why Interpolate Edge Lengths?

- They are shape invariant to rotation and translation.
- Vectors of edge lengths can be seen as points in a Euclidean shape space so we can easily perform statistical shape analysis.

Embedding Method

- Solve for the dihedral angles
- Reconstruct vertices from dihedral angles and edge lengths

Solve for the dihedral angles



- Vertex figure: Intersection of polyhedron with infinitesimal sphere at v.
- The ladybug's walk is a series of Z and X rotations in her local coordinate system.
- Minimizing E_v gives optimal dihedral angles.

$$E_v = ||Z_0 X_0 Z_1 X_1 \dots Z_k X_k - I||_F^2$$

Reconstruct vertices from dihedral angles and edge lengths

- Matrix $M_{v,u}$ transforms local coordinate system of v to local coordinate system of u; known from edge lengths and dihedrals
- Compute the orientation of every local coordinate frame G_v in a single global coordinate system by solving

$$G_v M_{v,u} - G_u = 0$$

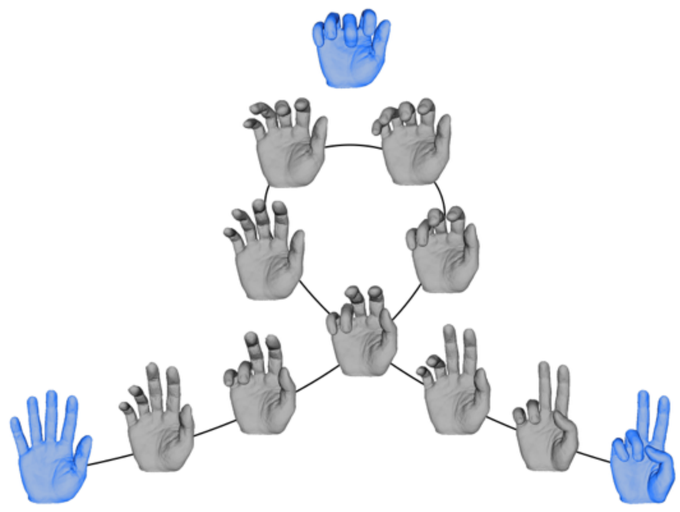
- For every edge u, v let u_v be position of u in the coordinate system of v.
- Global positions of u and v satisfy.

$$v + G_v u_v - u = 0$$

Results

Four edge-length interpolation steps between straight and bent arm.

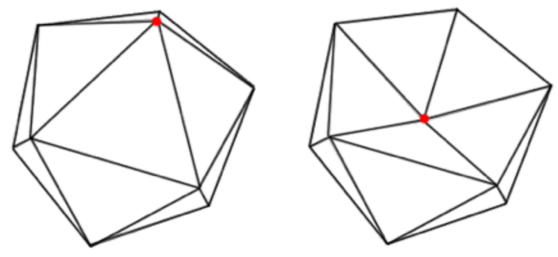
Bilinearly interpolating the edge lengths of the blue centaurs on the four corners produces the centaurs in the center.



The gray poses of the hands are interpolated from the three blue poses by interpolating their edge lengths.

Limitations

- Computation is slow, due to solving a high-dimensional non-linear system.
- Where a creases appear or disappear we see jitter
- Self-intersections are not allowed
- Usually there are multiple correct embeddings as shown in icosohedron below



For more information, please contact Carlos Rojas at crojas@ucdavis.edu

71.11 cm (W) x 91.43 cm (H)

AUTODESK.
Croucher Foundation 裘槎基金會
王寬誠教育基金會 K. C. WONG EDUCATION FOUNDATION

Edge Length Interpolation

Carlos Rojas, Alex Tsui, Stewart He, Lance Simons, Shengren Li, Nina Amenta
University of California, Davis

Abstract

We propose a scheme to interpolate between a set of two or more 3-dimensional triangulated meshes with corresponding connectivity. The principle behind this is that meshes, when viewed as vectors of edge lengths, can be seen as points in a Euclidean shape space. Within this space, mesh interpolation corresponds to traversing straight lines between points, and due to the Euclidean structure of the underlying shape space, we can easily take averages and perform statistical shape analysis. We perform experiments in three important applications to illustrate these ideas. First, we demonstrate interpolation between various shapes and poses. Second, we illustrate statistical analysis on a large dataset of faces, calculating mean shapes and exploring new shapes by moving along the principal modes of variance of the dataset. Finally, we visualize morphs corresponding to conformal mappings, an important class of deformations, which form a subspace of the edge-length space.

Key words: shape space, rigidity, shape interpolation, morphing, conformal mapping

1. Introduction

Let us begin with the mathematical description of the problem, and then discuss its relevance to computer graphics.

1.1. *The problem*

Consider a triangle mesh T and the graph G describing its vertex-edge connectivity. T fixes the vertex locations in $\mathbb{R}^3$, which obviously fixes the edge lengths, and these edge lengths are invariant under a rigid motion (rotation and translation) of the vertices. We might wonder if, conversely, fixing the edge lengths fixes the vertex positions up to a rigid motion?

In one sense this is nearly true: almost all closed manifold triangle meshes are rigid. Euler conjectured in 1766 that all closed polyhedral surfaces with rigid faces are rigid. Cauchy [8] proved that *convex* polyhedra with rigid faces are rigid, and Gluck [13] showed that most genus-zero triangulated polyhedra, even those that are not convex, are rigid. Only comparatively recently did Robert Connelly find an example of a manifold, embedded triangle mesh that was *not* rigid [10]. A recent ebook by Igor Pak [22] gives an excellent overview of this area; we touch on this again in Section 4, where we consider shape spaces.

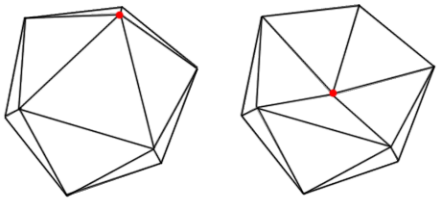


Fig. 1. Two different rigid embeddings of the same triangulated graph G that achieve the same set of edge lengths. The top vertex of the icosohedron on the left is "pushed" into the interior of the non-convex icosohedron on the right.

We can therefore be pretty certain that any embedding of G that we find will be rigid. The difficulty with solving for the vertex positions up to a rigid motion is that there might be, and in fact usually are, not one but a finite number

of discrete possible rigid embeddings $T_1, \ldots, T_k$ achieving the given set of mesh edge lengths; see Figure 1. So the problem which takes the abstract graph G and a set of edge lengths as input, and produces an embedding T as output, generally has many solutions.

In this paper we consider an easier problem. We are given an input set of two or more embeddings of G, and we want to interpolate between them by linearly interpolating their edge lengths. This is not always possible; in Figure 1, the edges adjacent to the red vertex would have to get smaller and then longer again when interpolating from one conformation to the other. But in many interesting cases - often when the embeddings are different poses of the same figure, when they are different instances from the same class, such as faces, bones or other anatomy, or in general when they are mild deformations of the same shape - it seems to be. The problem of constructing an interpolating mesh is easier than the general embedding problem since the input meshes can be used to construct a good approximation with which to begin an optimization algorithm. Our embedding algorithm has two steps, a non-linear optimization that finds a set of dihedral angles that respect the given edge lengths, and a linear step that finds a set of vertex positions respecting both the edge lengths and the dihedrals.

1.2. *Relevance to graphics*

This computational problem is useful in computer graphics because the edge-length vector associated with an embedding T of G forms a description of the object shape that has some obvious nice properties. First, shape can be defined technically as the geometric quantities that are invariant under some set of transformations. The edge-length vector is inherently invariant under translation and rotation, so it cleanly captures shape modulo rigid motion, with a small loss of information (due to the multiple embeddings issue). Second, the edge-length vector defines the graph metric on the surface, in which geodesic distance is measured by graph distance on the mesh. Interpolating the edge lengths produces deformations that are as-isometric-as-possible in this simple discrete sense.

We consider three applications of this basic idea of representing a shape by its graph G and its vector of edge lengths. In Section 3, we consider interpolating poses by interpolating their edge length vectors and using the method described above to embed the sequence of vectors into a sequence of 3D meshes. The as-isometric-as-possible property gives us interpolations between poses that naturally reflect the articulated motion, thus combining in one framework the morphs usually produced by a skeletal framework like linear blend skinning and the free-form detail changes accomplished with blend shapes. In Section 4 we argue that shape distances measured by Euclidean differences of edge-length vectors reflect meaningful shape differences that are either missed, or require complex calculations, in other shape space definitions. Finally in Section 5 we visualize a discrete conformal mapping as a morph. Discrete conformal mappings modify a shape by modifying its edge lengths, and we again use our embedding algorithm to map the edge length vectors to meshes in $\mathbb{R}^3$. The embedding algorithm thus partially answers the question of how to invert conformal mappings to parameter spaces.

This paper presents a novel approach to a fundamental problem in geometric computation, with ties to the interesting mathematical basis of mesh deformation.

1.3. *Prior work*

Shape spaces have been studied extensively in the context of the statistical analysis of anatomical shape. Even though it is well-established that shape spaces invariant to rotation are non-Euclidean [15], Euclidean spaces are used in practice because they allow the full range of statistical methods. In morphometrics [5,26] a Euclidean space is typically computed by aligning landmark points representing the specimens via Generalized Procrustes Analysis (GPA) and then working in the $3n$-dimensional coordinate space. Statistical Shape Analysis [12] takes this approach farther by deriving statistically useful feature spaces from this fundamental space. In the framework known as Euclidean Distance Matrix Analysis (EDMA) [18], the coordinates of the Euclidean shape space are the n^2 elements of the distance matrix of the landmarks. It is also possible [28] to use only the lengths of a sufficiently large set of "trusses" to ensure the rigidity of the point set. In $\mathbb{R}^2$ the choice of truss edges seems somewhat arbitrary, but our key observation is that the edges of a mesh representing the object surface are a natural choice in $\mathbb{R}^3$!

Allen et al's work on the space of human body shapes [2] was a landmark in the use of shape spaces in data-driven modeling in computer graphics. After fitting corresponding template meshes to their scanned data, they constructed a Euclidean shape space using PCA vectors computed from the template vertex positions. Later work considered human figures captured in different poses [1,3]. Interpolating facial expressions (blend shapes) using the Euclidean shape space of vertex positions is a venerable [23] mainstay of facial animation.

Kilian et al. [16] emphasized shape spaces in computer graphics, and their paper is the most relevant antecedent to

this work. They considered the computation of geodesics in two curved spaces: an approximation to the curved Kendall shape space, which is as-rigid-as-possible, and an approximation of a non-Euclidean as-isometric-as-possible space. The computational and mathematical issues concerning geodesic computation, averaging and parallel transport in those spaces, which are interesting but computationally intensive, are all easy in a Euclidean space.

Sederberg [24] proposed morphing two-dimensional polygons by interpolating their edge lengths and the angles. Winkler et al. [32] use a similar idea in 3D, interpolating the dihedral angles as well as the edge lengths, and fitting the model together by a multi-resolution least-squares computation. Our approach differs in that we use the fact that a global zero-error solution for a desired set of edge lengths usually exists, and we solve for the correct dihedrals. Their multi-resolution method is much faster than ours, however. Another multi-resolution method due to Chu et al. [9] finds a decomposition of the mesh based on a set of input poses, using mean-shift clustering, and then reconstructs an interpolated pose.

2. Algorithm

The straight-forward way to approach the embedding problem would be to solve for the vertex positions directly, given the edge lengths. But there are two major difficulties with this idea. Since we know that there will be many globally minimal solutions for any set of input edge lengths, we are guaranteed that any formulation of the problem will be non-linear and high-dimensional. It is very important, therefore, to start from a good approximate solution. It is not clear how to choose this approximate solution from the vertex positions of the meshes to be interpolated, since the vertex positions move, perhaps quite a bit, in a non-linear and mesh-dependent way during the interpolation. A second problem is that the vertex positions depend on an arbitrary rotation and translation, so that some regularization is required to select a canonical set. Generally this is done by fixing the position of one triangle, but the optimization then has to propagate this arbitrary choice to the rest of the mesh. This further complicates an already fragile calculation.

Instead, we work in two steps. First we solve for the dihedral angles of the mesh. These are also invariant under rotation and translation, so we avoid having to choose a canonical representation when doing the nonlinear optimization, and they generally change smoothly during the morph so we can select good estimated solutions by linearly interpolating the dihedrals of the input meshes. Once we have the dihedrals, the mesh is determined up to rotation and translation, and we find the vertex positions by solving two linear systems, one for rotation and the other for translation, roughly as in [20].

2.1. *Solving for dihedrals*

Consider the *spherical vertex figure* formed by intersecting the polyhedron with an infinitesimally small sphere centered at a vertex v (see Figure 2). The boundary of the intersection is a spherical polygon with sides $S_1, S_2, \ldots S_k$, all great circle arcs, and vertices $u_1, \ldots, u_k$, where $k = \text{degree}(v)$. The arc lengths of the sides are the angles of the triangular faces at v, and the angles at the vertices u_i of the spherical polygon are the dihedrals. Now consider the

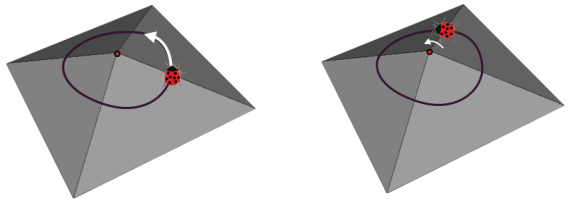


Fig. 2. The vertex figure is the intersection of an infinitesimal sphere with the mesh at a vertex. Consider a ladybug taking a closed walk along the vertex figure. We equip her with a local coordinate system that is centered at the vertex, so that she is always at position $(1, 0, 0)$ and facing $(0, 1, 0)$. To cross a triangle, the ladybug rotates around the z-axis of her local coordinate system, which passes through the origin. To rotate at a corner of the vertex figure, she rotates around the x-axis of her local coordinate system, on which she sits.

vertex figure on the unit sphere, and consider a ladybug walking along it, controlled by a local coordinate system, as in Figure 2. Each step in her walk is either a rotation X_i around the x-axis (through the bug, at a vertex of the spherical polygon) or a rotation Z_i around the z-axis (walking an edge of the spherical polygon). So her entire path is a series of matrix multiplications (right-multiplication of matrices is rotation in the local coordinate system) such that:

$$Z_0 X_0 Z_1 X_1 \ldots Z_k X_k = I \tag{1}$$

where I is the identity matrix; after the walk she is back in her original position and orientation. The Z_i are all constant, determined by the input edge lengths; our job is to determine the X_i, each of which is defined by a rotation angle α_i.

For an arbitrary set of dihedral angles X_i, the path does not close; there is some error both in the position and the orientation of the bug. We define an the error function at vertex v:

$$E_v = ||Z_0 X_0 Z_1 X_1 \dots Z_k X_k - I||_F^2 \quad (2)$$

where F indicates the Frobenius norm, that is, the error is the sum of the squares of the elements of the matrix. The overall energy function for the mesh is:

$$E(A) = \sum_v E_v \quad (3)$$

where A is the vector of dihedral angles α_i.

We use the non-linear solver `fmin_l_bfgs_b` from the SciPy package [7], since it works well on problems with many local minima. This implementation of the BFGS technique requires taking the first and second derivatives of the energy function with respect to each variable at each iteration. The error will be the same wherever we break the cycle, so we might as well assume it is broken at X_k. The other X_i do not depend on X_k, so to compute the derivatives with respect to α_k we can rewrite each term of the energy function

$$M_k = Z_0 X_0 Z_1 X_1 \dots Z_k \quad (4)$$
$$E_v = ||M_k X_k - I||_F^2 \quad (5)$$

where M_k is a rotation matrix independent of α_k, and

$$X_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha_k & -\sin\alpha_k \\ 0 & \sin\alpha_k & \cos\alpha_k \end{bmatrix} \quad (6)$$

A little calculation, using the fact that M_k is orthonormal, shows that the degree two terms in $\sin\alpha_k$ and $\cos\alpha_k$ cancel out, and we get

$$E_v(\alpha_k) = 6 - 2[m_{11}^{kj} + (m_{22}^{kj} + m_{33}^{kj})\cos\alpha_k + (m_{23}^{kj} - m_{32}^{kj})\sin\alpha_k]. \quad (7)$$

leaving us with the simple first and second derivatives with respect to α_k:

$$\frac{\partial E_v}{\partial \alpha_k} = -2((m_{23}^j - m_{32}^j)\cos\alpha_k - (m_{22}^j + m_{33}^j)\sin\alpha_k) \quad (8)$$

and

$$\frac{\partial^2 E_v}{\partial \alpha_k^2} = 2((m_{22}^j + m_{33}^j)\cos\alpha_k + (m_{23}^j - m_{32}^j)\sin\alpha_k). \quad (9)$$

A specific α_k appears in two of the vertex terms, say u and v (with the same sign!), so that

$$\frac{\partial E}{\partial \alpha_k} = \frac{\partial E_v}{\partial \alpha_k} + \frac{\partial E_u}{\partial \alpha_k} \quad (10)$$

meaning that

$$\frac{\partial E}{\partial \alpha} = -2([(m_{23}^v - m_{32}^v) + (m_{23}^u - m_{32}^u)]\cos\alpha_k - [(m_{22}^v + m_{33}^v) + (m_{22}^u + m_{33}^u)]\sin\alpha_k) \quad (11)$$

and similarly for the second derivative.

We initialize the search for the vector A of dihedral angles by averaging the dihedrals of the input meshes at each edge. This works surprisingly well in most of the cases we tested, but it is not guaranteed to converge to a global minimum (a set of dihedrals that form a closed mesh), or to the minimum we desire for a smooth interpolation. We discuss a few situations where it failed in Section 6.

2.2. *Solving for vertex positions*

Given the dihedral angles, it is clear that the mesh is rigid and we can compute a set of vertex positions that can realize the mesh. The straightforward linear system to achieve this is very ill-conditioned, so we use a more robust two-step approach adapted from the Frenet frames method of Lipman et al. [20]. A similar construction that takes edge lengths and dihedrals as input and produces a mesh output was recently proposed by Wang et al. [31].

We first define an orthonormal coordinate frame at each vertex v, by arbitrarily putting the local x-axis on the edge with minimum ID adjacent to v, putting the local y-axis in the direction of, and on the plane of, the triangle to the left of that edge, and letting the z-axis be their cross-product. The first step is to find, for every edge uv, a local rotation $M_{u,v}$ that transforms the local coordinate system of u to the local coordinate system of v. This is possible because we know the dihedral angles of the edges adjacent to u and v.

The next step is to use these local changes to compute the orientations of every local coordinate frame in a single global coordinate system. Let G_v be the global rotation at v. Then we have the system of equations

$$G_v M_{v,u} - G_u = 0$$

for every edge u, v. We treat this as nine linear equations, one for each element of G_u, ignoring the constraint that the G_u must be orthonormal. We solve the over-constrained system by least-squares, fixing the rotation at vertex v_0.

While a perfect solution to this set of equations should exist and should be a set of rotation matrices, because of numerical error and because the linear system is ill-conditioned, the G_v are not orthonormal in practice. We therefore orthogonalize the resulting G_u matrices, using Singular Value Decomposition [21].

Fig. 3. Using edge-length interpolation to combine skeletal motion and blend shapes for animation. The original arm mesh (upper left) is bent using linear blend skinning, producing a self-intersection (upper middle). The bent arm pose is then cleaned up interactively to remove the self-intersection (upper right). Four intermediate steps of edge-length interpolation between the original straight arm and the cleaned-up bent arm interpolate both the skeleton-derived motion and the edits.

	vertices	time	error
hand	10,000	9.1 min	0.172
centaur	15,768	9.6 min	2.601
face	58,327	20.6 min	0.077
arm	2,502	3.9 min	0.053

Table 1
Time is mean total running time per embedding in minutes. Error is the error of the non-linear optimization from Equation 3. For the arm, face and hand the statistics are over all frames of the video.

Given the global rotations G_u, we then construct the global vertex positions. For every edge u, v, let u_v be the position of u in the coordinate system at v. Then the real positions of u and v satisfy

$$v + G_v u_v - u = 0$$

Again, we initialize the position of v_0 and solve by least-squares. A robust direct solver for sparse, ill-conditioned systems is required in both these steps [6]; we used the `spsolve` function in SciPy.

2.3. *Timing and Performance*

Timings and errors are shown in Table 1. In rendering dense sequences of embeddings for video, we found glitches associated with some specific shape transitions, leading to the longer running times and higher errors. We discuss this further in Section 6.

3. Pose Interpolation

Pose interpolation should be as-isometric-as-possible [16], since maintaining a near-isometry maintains the nearly-rigid parts of the mesh. We find that edge-length interpolation, being as-isometric-as-possible, indeed produces intermediates that are visually appropriate, without any of the well-known artifacts produced by interpolation

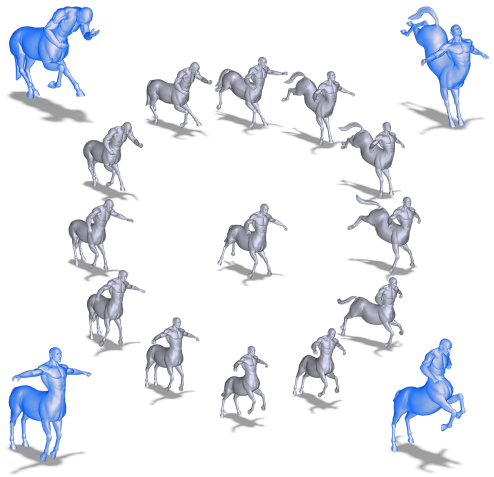


Fig. 4. Linearly interpolating the edge lengths of a triangle mesh between different poses produces interpolated meshes that are as-isometric-as-possible. Here, the input meshes in blue are different poses of an articulated model. The edge-length interpolated poses in gray inherently respect the structure of the articulated motion. Each edge length in a gray mesh is a weighted average of the corresponding four edge lengths in the blue input meshes. The difficult step in the process, that we treat in this paper, is producing the vertex positions from the averaged edge lengths.

of vertex positions. In Figure 4 we show bilinear interpolations along a circular path. Figure 5 gives an example of barycentric interpolation along a spline.

Animation of 3D objects is mostly done by interpolating keyframe poses. When the position of the mesh is controlled by a skeleton, the keyframes are specified by joint angles, which are interpolated in a joint-angle space. While this handles the basic pose, fixing skinning artifacts, specifying non-skeletal motion, and other details require additional solutions. One important tool here is blend shapes, meshes interpolated in the space of vertex positions, that are combined with joint-angle interpolation to adjust shapes. There have been many variants of this idea [19,25,17]. Interpolating in edge-length space could make this process simpler. For instance, it might be possible to do initial animation in joint space, edit the resulting meshes for cleanup and details, and then interpolate the edited meshes directly as keyframes in edge-length space. We show an example of this workflow for cleaning up a bent arm in Figure 3.

Finely detailed poses of real people are increasingly being captured as meshes or partial meshes [2,3,1]. Similarly, capturing human motion as surface meshes brings a huge amount of data to the problem of generating moving characters [33,30], and the advent of depth cameras like the Kinect should make this even more practical [14]. Fitting the raw data with a common template mesh, and then interpolating the corresponding meshes gives a way to interpolate captured poses or to connect together captured mesh motions, without the need to identify a skeleton. It is necessary to find the corresponding meshes, but this is often done as part of data clean-up anyway.

Since computing embeddings is computationally expensive and subject to occasional glitches, we produced the hand and cat video clips by embedding the mesh for every 20th or 25th frame as a sub-keyframe, and then interpolating between the sub-keyframe meshes by linearly interpolating edge lengths and dihedrals. This was faster and produced smoother motions.

4. Shape analysis

In this section we consider the shape space $\mathcal{E}$ associated with a combinatorial triangle mesh graph G. The points of $\mathcal{E}$ are vectors of edge lengths and the metric on $\mathcal{E}$ is the usual Euclidean distance. A drawback is that one point of $\mathcal{E}$ corresponds to a finite set of meshes, not just one. But the affordances are the fact that $\mathcal{E}$ is Euclidean and that it is inherently invariant under rotation and translation. The algorithm of Section 2 allows us to (usually) realize points of $\mathcal{E}$ computed by interpolation as meshes in $\mathbb{R}^3$.

By Euler's formula, a triangle mesh homeomorphic to the sphere has $3n-6$ edges, so this is the dimension of $\mathcal{E}$. Note that this is conveniently the $3n$ degrees of freedom of the possible vertex positions for G, minus the six degrees of freedom for rotation and translation. So $\mathcal{E}$ really is a $3n-6$-dimensional set in $\mathbb{R}^{3n-6}$. When one edge is removed from G, the remaining structure becomes flexible [22]; equivalently, one edge length can be changed independently of the others while remaining in the space $\mathcal{E}$.

Unfortunately, $\mathcal{E}$ has boundaries. A set of edge lengths has to obey the triangle inequalities, imposing $3t$ linear inequalities, where t is the number of triangles. Fortunately, if an edge-length vector $e \in \mathcal{E}$ is computed by linearly interpolating a set of input edge-length vectors that obey the triangle inequalities, so must e. But $\mathcal{E}$ has other boundaries as well. For instance, a valid triangle mesh cannot include a vertex where three triangles meet, one with angle $\pi/2$ and the other two with angles $\pi/5$; the angles at each vertex must satisfy a triangle inequality as well. These boundaries are unfortunately non-linear in the edge lengths, and can be violated when interpolating input meshes with vertices spanning very small solid angles. Nonetheless, in practice, we have ignored the boundaries of $\mathcal{E}$.

Let us compare $\mathcal{E}$ to the Generalized Procrustes shape-space $\mathbb{P}$ often used in practice for statistical shape analysis, and, for instance, in [2]. Given a collection of input shapes, represented as corresponding clouds of landmark points (in our case, the mesh vertices), the GPA process aligns them, under scale, translation and rotation, into a single global coordinate system so as to minimize the total RMS distance between all pairs of corresponding points. The $3n$-dimensional space formed by the point coordinates then is taken as $\mathbb{P}$, where the RMS distance between two input

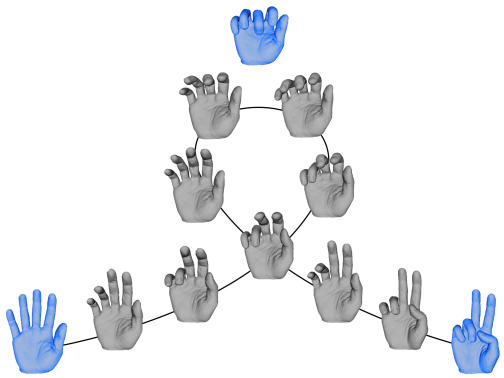


Fig. 5. The gray poses of the hands are interpolated from the three blue poses by interpolating their edge lengths. This sequence can be found in the video.

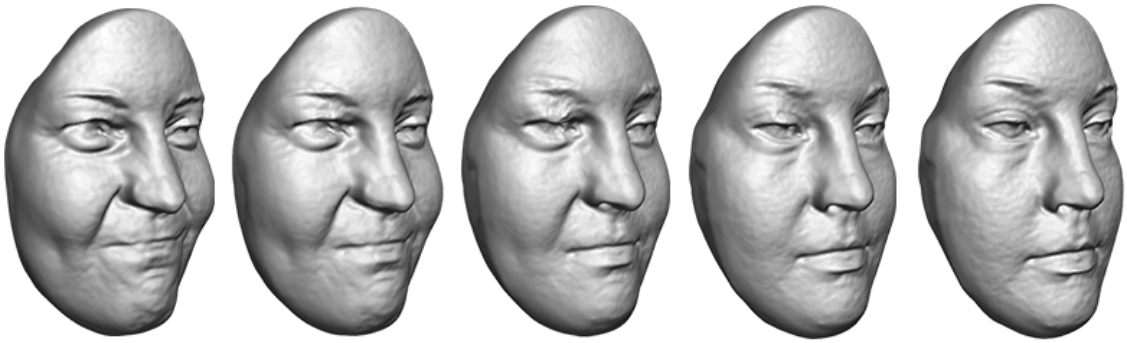


Fig. 6. Corresponding meshes are fitted to the scanned face meshes. Edge-length interpolation between the corresponding meshes produces the change of expression.

point sets is the same as the Euclidean distance between the two points in $\mathbb{P}$.

The alignment is intended to "remove" the part of the geometric difference due to scale, rotation and translation. But for even moderately complex shapes, global alignment cannot simultaneously align all the parts. This is illustrated in Figure 8. When this happens, the RMS distance ends up reflecting the misalignment between corresponding parts and misses similarities between the parts themselves.

Figure 7 shows an experiment comparing $\mathbb{P}$ and $\mathcal{E}$ on a synthetic example meant to evoke Figure 8. We varied a set of parameters to produce barbells with two kinds of ends, both with two different lengths for the bar. To make a fair comparison, we need to make $\mathcal{E}$ invariant under scale as well as rotation and translation. We scale the input edge length vectors e_i so that

$$\sum_i e_i = 1 \tag{12}$$

Since this is a linear equation, it defines a hyperplane in $\mathcal{E}$ containing all of the inputs, maintaining the linearity of the space. Normalizing for scale shrinks the barbells with the longer bar in both cases. When measured by the aligned RMS distance, the two long barbells are clearly more similar than the two short ones. When measured by the scaled edge-length distance, the distances become more comparable.

Figure 6 shows an example of the shape analysis workflow for scanned data. The meshes of the scanned faces are not corresponding. We manually place eight landmark points on the facial features, and then we use a method similar to that of Tsui et al. [29] to establish a smooth corresponding mesh that includes the landmark points on each surface. Interpolating the corresponding meshes for the faces produces the change in expression.

Alternatively, we can consider the set of faces in the space of edge length vectors and explore the space. In particular, we computed PCA vectors on a set of 600 scanned faces from the FRGC 2.0 dataset. The set of faces are made up of happy (green), angry (red), and neutral (black) examples, classified based on their respective facial expression. For any given happy/angry example of a particular individual, we have corresponding data of the same individual with a neutral expression. Projecting each of the faces onto the top three principal components gives us a rough clustering shown in Figure 9. While the variance explained by the three eigenvectors accounts for only 12% of the total variance in the dataset, there is a semblance of shape difference captured in the edge lengths, particularly between the neutral and happy expressions.

As a further example of exploring edge length space, we ask the question of how we can visualize new facial expressions that are representative of the individual expressions within our dataset (see Figure 10). As mentioned above, the individual faces with facial expressions are paired with a corresponding neutral expression. We take the difference of the edge length vectors of these two expressions. We then average the difference across all paired examples in the dataset to compute an average edge length difference vector. Finally, we add this difference vector to the average neutral face to produce a new edge length vector, which we use to reconstruct a previously unseen face. This demonstrates the ease of exploring the shape space.

5. Conformal mappings

Conformal mapping is an important mathematical tool in geometry processing, used mainly for parameterization and as a tool for establishing surface correspondences

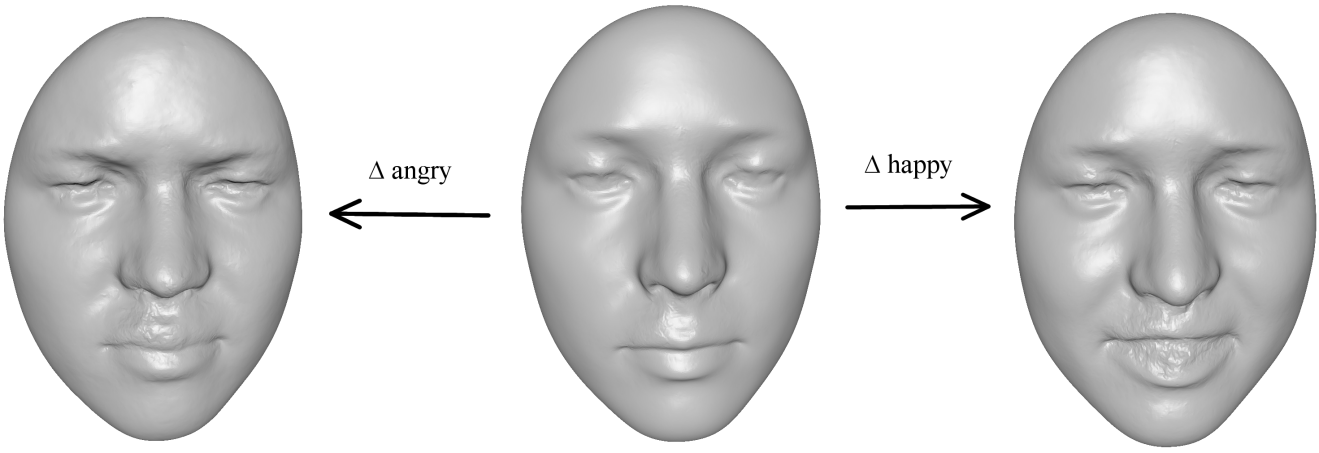


Fig. 10. The middle face is the average of all the neutral faces. The two outer faces represent the average neutral face plus the average difference between the neutral and happy/angry face of a particular person.

through parameterization. It is sometimes possible, however, to use conformal maps for interpolation. In the smooth case, when A and B are any two smooth surfaces homeomorphic to the sphere, then there is a conformal map $\mu()$ between A and B. Recent work in computer graphics [27,4] has led to discrete analogs of this idea. For instance, following [27], let $a_{i,j}, b_{i,j}$ be the lengths of edge i, j in two triangulated meshes A, B homeomorphic to the sphere. We associate a conformal factor u_i to each vertex v_i; we can think of u_i as a scaling factor that expands or contracts the neighborhood of v_i. We say that B is discretely conformally equivalent to A if there is a set of factors u such that

$$b_{i,j} = e^{(u_i+u_j)/2} a_{i,j} \tag{13}$$

Using the methods of [27,4], we can compute a set of conformal factors u that map the edge-lengths of any triangulated mesh A to a discretely conformally equivalent mesh B on the sphere. In Figure 11, we show interpolations visualizing this mapping from a face mesh A as a morph. At each time step, we linearly interpolate the factors u_i, beginning at zero and ending at the computed value controlling the edge lengths on the sphere, use the interpolated conformal factors to produce interpolated edge lengths, and finally embed the edge lengths using the algorithm of Section 2. We initialize the dihedrals as usual by linearly interpolating the dihedrals at the endpoints of the morph.

Two arbitrary embeddings T_1, T_2 of a mesh with combinatorial structure G are not, in general, discretely conformally equivalent; conformal equivalence is a more restrictive property. However, Crane et al. [11] recently gave a computational framework for mesh deformation using conformal maps, that preserve local details such as texture and mesh quality.

6. Limitations and discussion

This work certainly has limitations, some perhaps associated with our implementation and others inherent in the idea. The computation is very slow, since each embedding is computed by solving the high-dimensional non-linear system described in Section 2.1. This could probably be addressed by a more sophisticated approach, for instance using mesh multiresolution and, when computing sequences, leveraging redundancy.

In the videos, computing every frame as an independent embedding was slow and there were glitches due to poor embeddings. Embedding only sub-keyframe meshes and interpolating between them using the method described in Winkler [32] led to much nicer results.

It is not completely surprising that there are cases in which the embedding algorithm does not work for interpolation. For example, in the simple situation in Figure 1, the only nearby minima for intermediate frames we might want to generate using interpolated dihedrals would be the two input meshes. We believe that the conformally mapped face near the end of the sequence involves similar local changes, where a region shifts from convex to concave or visa versa. Other situations in which a crease appears or disappears seem to similarly cause jitter. Detecting and handling such jumps in the embedding is clearly an issue for further research.

Another simple situation in which there is an issue involved a quad mesh on a torus morphing to a tall barrel shape by undergoing a large anisotropic deformation. We

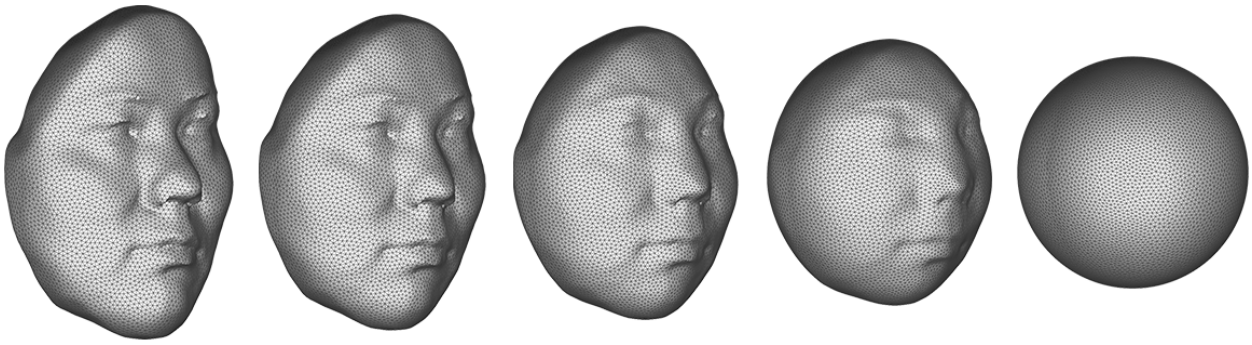


Fig. 11. Conformally mapping a scanned face (with the back of the head closed off) to the sphere. The discrete conformal factors are all zero on the left, and form a function $\gamma(v)$ on the vertices $v \in V$ on the spherical mesh on the right. The intermediate meshes are computed by setting the conformal factors to $k\gamma(v)$, for $k = 1/4, 1/2, 3/4$ as we go from left to right, computing the edge lengths from the conformal factors, and then embedding.

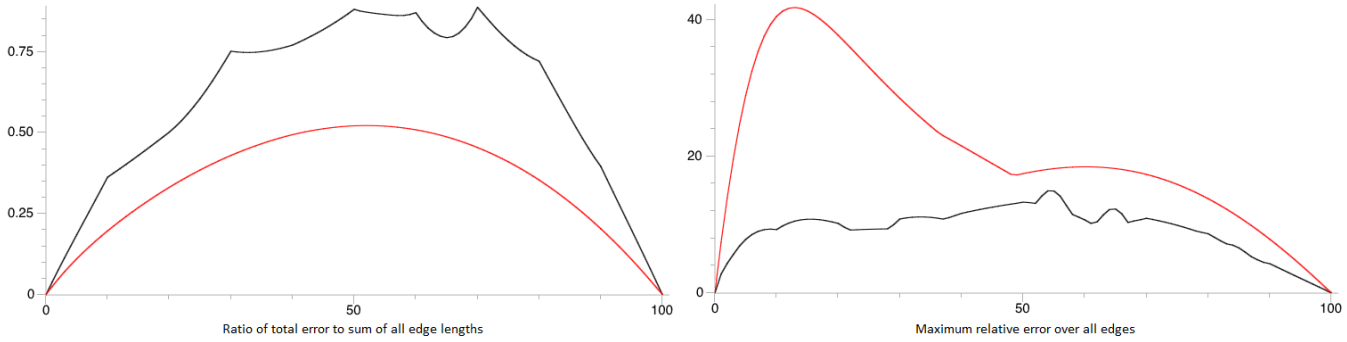


Fig. 12. The graph on the left depicts the ratio of total RMS error, between the linearly interpoliated edge lengths and the reconstructed meshes, to the sum of all linearly interpolated edge lengths. The red line represents the result from Winkler and the black line represents ours. The ratio is computed 100 times at even intervals while interpolating from the bent arm to the straight arm in Figure 3. The total error in Winkler's method peaks at .5% at around frame 50 while our error peaks at about .75%. The graph on the right shows the maximum relative error. Again Winkler is the red line and we are the black. We can see that in the worst case, around frame 15, the worst edge in Winkler's reconstruction is more than 40% incorrect. Our worst case occurs at frame 53, and is only 15% incorrect.

added a randomly chosen diagonal to each quad to produce a triangle mesh. Stretching the quads only in the vertical direction distorted the quads, since in a quad the diagonals do not scale linearly with the vertical edges. Linear edge-length interpolation created uneven surfaces, and, in the worst case, a completely incorrect embedding as a double-covering of the torus. Interpolating the edge length squared rather than edge length completely fixes this problem, but we do not prefer it as a general solution for interpolation because large and small edges grow at different rates. On the other hand, this example does indicate that the embedding method works on objects of higher genus.

Finally, our approach resembes the technique by Winkler [32]. The main difference is the non-lineaar step described in Section 2.1. In a comparison between the two methods we found that Winker [32] produced lower RMS difference between the reconstructed edge lengths and the interpolated edge lengths. However, our approach results in a lower maximum error, seen in Figure 12.

Acknowledgements

Acknowledgments of data sources and other assistance are omitted in this version.

References

[1] Brett Allen, Brian Curless, and Zoran Popović. Articulated body deformation from range scan data. In *ACM Transactions*

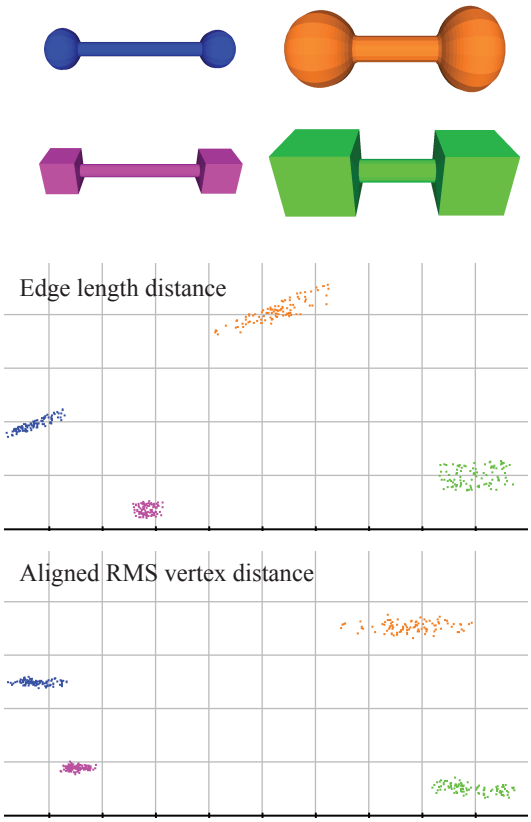


Fig. 7. Shape difference measured by edge-length distance reflects the similarities of parts better than aligned RMS distance. Using aligned RMS distance to compare the resulting shapes, the size difference dominates. The edge-length-distances, even with the scale normalization, are more sensitive to the similarities in the ends.

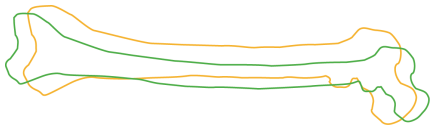


Fig. 8. Two primate femur sketches whose shafts have different lengths, but whose ends are nearly identical. The bones are aligned with respect to scale, rotation and translation. The RMS distance between the aligned bones captures the gross difference in shape, but fails to capture the fact that the ends are nearly identical.

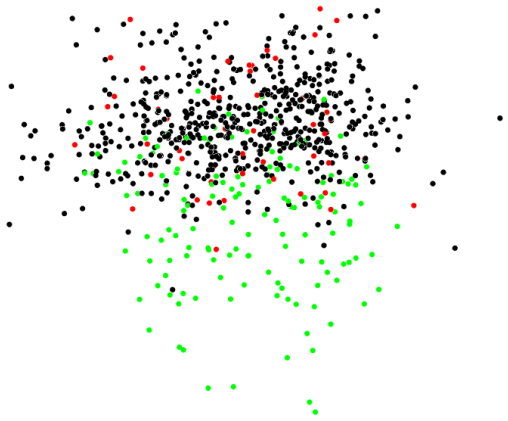


Fig. 9. Neutral (black), happy (green), and angry (red) faces, similar to those in Figure 6, are projected down onto two principal components from points in $\mathcal{E}$. The results show that angry and neutral faces are similar in $\mathcal{E}$, but happy faces have more diversity and move away from the central cluster.

on Graphics (TOG), volume 21, pages 612–619. ACM, 2002.

[2] Brett Allen, Brian Curless, and Zoran Popović. The space of human body shapes: reconstruction and parameterization from range scans. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 587–594. ACM, 2003.

[3] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 408–416. ACM, 2005.

[4] Mirela Ben-Chen, Craig Gotsman, and Guy Bunin. Conformal flattening by curvature prescription and metric scaling. In *Computer Graphics Forum*, volume 27, pages 449–458. Wiley Online Library, 2008.

[5] F. L. Bookstein. *Morphometric tools for landmark data: Geometry and Biology*. Cambridge Univ. Press, New York, 1991.

[6] Mario Botsch, David Bommes, and Leif Kobbelt. Efficient linear system solvers for mesh processing. In *Mathematics of Surfaces XI*, pages 62–83. Springer, 2005.

[7] A. H. Byrd and J. Nocedal. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific and Statistical Computing*, 16(5):1190–1208, 1995.

[8] A. Cauchy. Sur les polygones et polyedres. *Second Memoire, Journal École Polytechnique*, 9(87), 1813.

[9] Hung-Kuo Chu and Tong-Yee Lee. Multiresolution mean shift clustering algorithm for shape interpolation. *Visualization and Computer Graphics, IEEE Transactions on*, 15(5):853–866, 2009.

[10] R. Connelly. A counterexample to the rigidity conjecture for polyhedra. *Publications Mathmatiques de L'IHS*, 47(1):333–338, 1977.

[11] Keenan Crane, Ulrich Pinkall, and Peter Schröder. Spin transformations of discrete surfaces. In *ACM Transactions on Graphics (TOG)*, volume 30, page 104. ACM, 2011.

[12] Rhodri Davies, Carole Twining, and Chris Taylor. *Statistical Models of Shape*. Springer, 2008.

[13] H. Gluck. Almost all simply connected closed surfaces are rigid. In *Geometric Topology*, volume 438 of *Lecture Notes in Mathematics*, pages 225–239. Springer-Verlag, 1975.

[14] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, 2011.

[15] David G Kendall. Shape manifolds, procrustean metrics, and complex projective spaces. *Bulletin of the London Mathematical Society*, 16(2):81–121, 1984.

[16] Martin Kilian, Niloy J Mitra, and Helmut Pottmann. Geometric modeling in shape space. In *ACM Transactions on Graphics (TOG)*, volume 26, page 64. ACM, 2007.

[17] Paul G Kry, Doug L James, and Dinesh K Pai. Eigenskin: real time large deformation character skinning in hardware. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 153–159. ACM, 2002.

[18] Subhash R. Lele and Joan T. Richtmeier. *An invariant approach to the statistical analysis of shapes*. Chapman and Hall/ CRC, 2001.

[19] John P Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 165–172. ACM Press/Addison-Wesley Publishing Co., 2000.

[20] Yaron Lipman, Olga Sorkine, David Levin, and Daniel Cohen-Or. Linear rotation-invariant coordinates for meshes. *ACM Transactions on Graphics (TOG)*, 24(3):479–487, 2005.

[21] Jianqin Mao. Optimal orthonormalization of the strapdown matrix by using singular value decomposition. *Computers & mathematics with applications*, 12(3):353–362, 1986.

[22] Igor Pak. *Lectures on discrete and polyhedral geometry*. 2009.

[23] Frederick I Parke. Computer generated animation of faces. In *Proceedings of the ACM annual conference-Volume 1*, pages 451–457. ACM, 1972.

[24] Thomas W Sederberg, Peisheng Gao, Guojin Wang, and Hong Mu. 2-d shape blending: an intrinsic solution to the vertex path problem. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 15–18. ACM, 1993.

[25] Peter-Pike J Sloan, Charles F Rose III, and Michael F Cohen. Shape by example. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 135–143. ACM, 2001.

[26] Christopher G. Small. *The statistical theory of shape*. Springer, 1996.

[27] Boris Springborn, Peter Schröder, and Ulrich Pinkall. Conformal equivalence of triangle meshes. In *ACM Transactions on Graphics (TOG)*, volume 27, page 77. ACM, 2008.

[28] Richard E Strauss and Fred L Bookstein. The truss: body form reconstructions in morphometrics. *Systematic Biology*, 31(2):113–135, 1982.

[29] Alex Tsui, Devin Fenton, Phong Vuong, Joel Hass, Patrice Koehl, Nina Amenta, David Coeurjolly, Charles DeCarli, and Owen Carmichael. Globally optimal cortical surface matching with exact landmark correspondence. In *Information Processing in Medical Imaging*, pages 487–498. Springer, 2013.

[30] Daniel Vlasic, Pieter Peers, Ilya Baran, Paul Debevec, Jovan Popović, Szymon Rusinkiewicz, and Wojciech Matusik. Dynamic shape capture using multi-view photometric stereo. In *ACM Transactions on Graphics (TOG)*, volume 28, page 174. ACM, 2009.

[31] Yuanzhen Wang, Beibei Liu, and Y Tong. Linear surface reconstruction from discrete fundamental forms on triangle meshes. In *Computer Graphics Forum*, volume 31, pages 2277–2287. Wiley Online Library, 2012.

[32] Tim Winkler, Jens Drieseberg, Marc Alexa, and Kai Hormann. Multi-scale geometry interpolation. In *Computer graphics forum*, volume 29, pages 309–318. Wiley Online Library, 2010.

[33] Li Zhang, Noah Snavely, Brian Curless, and Steven M Seitz. Spacetime faces: High-resolution capture for˜ modeling and animation. In *Data-Driven 3D Facial Animation*, pages 248–276. Springer, 2007.