Computers & Graphics 59 (2016) 107–118

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Fast mapping and morphing for genus-zero meshes with cross spherical parameterization ☆

Chao Peng *, Sabin Timalsena

Department of Computer Science, University of Alabama in Huntsville, United States

ARTICLE INFO

Article history:
Received 11 January 2016
Received in revised form
31 May 2016
Accepted 1 June 2016
Available online 17 June 2016

Keywords:
Cross parameterization
Topology merging
Shape blending
Morphing

ABSTRACT

We introduce an efficient approach to establish surface correspondences between genus-zero triangle meshes and animate a morph between them. We employ the concept of mesh simplification and refinement to progressively parameterize meshes using a novel kernel sampling algorithm. Our two-step feature alignment method optimizes the distribution of vertices in a parametric domain. A new mesh is generated after merging mesh topologies, and it represents the shapes of all input meshes and their transitions in-between. Our approach merges topological structures rather than mapping surface properties. Complex meshes with tens of thousands of vertices can be parameterized and merged in a few minutes. The advantages of our approach are demonstrated through 3D morphing applications. Comparing to existing approaches, our solution advances in performance while maintaining high-quality parameterization and morphing.

Published by Elsevier Ltd.

1. Introduction

Merging topological structures of two or more 3D meshes into a single mesh is an important subject of computer graphics. It is also fundamentally important for achieving the effects such as morphing, retopologizing, mesh rigging transfer, etc. The general idea is parameterizing meshes into a suitable common domain. Then, those parametric representations are combined to form a new mesh whose topology fits geometric properties of all input meshes. Given genus-zero meshes, which are homeomorphic, we look for a new mesh representation whose topology supports smooth shape transitions between input meshes, regardless of differences of their original topologies or in correspondence of vertex indices. As introduced in [1,2], the process of finding parametric representations is known as *cross parameterization*.

A sphere is the natural domain for genus-zero meshes. Previous approaches for the construction of spherical parameterizations (e.g., [3–6]) do not achieve efficient performance, and issues like parametric distortion and poor feature alignment still exist: (1) high parametric distortion stretches triangles and causes overfolds on the surface of the parameterization, which prevent from obtaining valid topology merging and (2) user-specified feature constraints displace vertices in non-feature regions and cause triangle degeneration and overfolds.

Contribution. In this paper, we present an efficient approach to merge topological structures of homeomorphic genus-zero meshes with aligned surface correspondences. Our approach advances existing work in performance. Mesh simplification and refinement algorithms are adopted to progressively parameterize meshes using a novel metric that measures the distortion of parameterized triangles. Our parameterization algorithm is efficient and processes tens of thousands of triangles per minute. A novel two-step feature alignment procedure is developed to align user-specified feature constraints. We also develop a remeshing algorithm to generate a new mesh with generated spherical representations of the input meshes. Qualities of parameterization and remeshing are demonstrated through morphing applications. Our approach can produce visually pleasing morphing results for complex meshes that do not have trivial feature correspondences.

The rest of the paper is organized as follows. Section 2 reviews related work and describes parameterization issues. Section 3 gives a system overview. A mesh simplification method is presented in Section 4. Section 5 introduces a parameterization method based on mesh refinement. Section 6 presents a global optimization algorithm to optimize the distribution of vertices in the parametric domain. A feature alignment method is described in Section 7. Section 8 presents remeshing and morphing methods. We show our experimental results in Section 9. Section 10 concludes our work.

☆This article was recommended for publication by K. Hormann.

* Corresponding author. Tel.: +1 2568246433.
E-mail addresses: chao.peng@uah.edu (C. Peng), sabin.timalsena@uah.edu (S. Timalsena).

http://dx.doi.org/10.1016/j.cag.2016.06.001
0097-8493/Published by Elsevier Ltd.

2. Related work

Mesh parameterization is a technique to establish a map that transforms the points of a mesh into a parametric domain. The parametric domain is usually a simple surface, such as a 2D square, a sphere, a torus and a cube. A group of parameterized meshes can be applied with domain-specific operations without regard to their actual 3D shapes. In this section, we discuss previous work related to mesh parameterization.

A planar parameterization method cuts a mesh into flat pieces, just like creating texture mapping for 3D meshes. Sorkine et al. [7] parameterized the surface into a topological disk with strictly bounded distortion. Kraevoy and Sheffer [1] used triangular patch layouts of input meshes with a smoothing mechanism to reduce mapping distortions. Aigerman et al. [8] chose feature correspondent points and geodesic cuts. Their approach allows overlaps in the parametric domain, and bijectivity is recovered using a bijection lifting technique, but recovered mappings may cause artifacts if feature points are poorly chosen. Their follow-up work [9] encoded cut-invariance in the parameterization, but noticeable distortions exist if aligned feature pairs are not structurally similar. Planar parameterizations have been used for texture mapping [10] and generating geometry images [11], but they are not suitable for morphing applications. A 2D parametric domain has boundaries because of cutting. People have to spend many efforts to balance the number of continuous "pieces" and the degree of distortion; and in the 2D parametric domain, interpolation of mesh properties for morphing cannot be done effectively as such properties discontinue at the boundaries.

A spherical parameterization method does not require cutting. We review two categories of the method that projects a mesh to a sphere: *progressive method* and *relaxing method*.

2.1. Progressive method

A mesh is progressively simplified to a base mesh which is trivial for spherical projection. Quicken et al. [12] built a mesh hierarchy using a simplification procedure, and vertices were inserted back iteratively until reaching the full resolution of the mesh. Praun and Hoppe [3] used a progressive mesh. They first mapped the base mesh to a sphere; vertices were then split and projected to an optimal position in the ring of neighbor vertices in the spherical domain. Local angle and area distortions were minimized during vertex splitting and projection. Similarly, Wan et al. [13] developed a hierarchical optimization scheme to minimize both local and global distortion energy.

Progressive methods preserve features, but long and thin triangles may depress the surface of the sphere after the parameterization. In Fig. 1, the depression is very deep because the region has some large triangles with high curvature. When vertices are added back iteratively, the depression in that region will be further narrowed and deepen. Eventually, triangles in the depressed region will lead to flat appearance on the remeshed model.

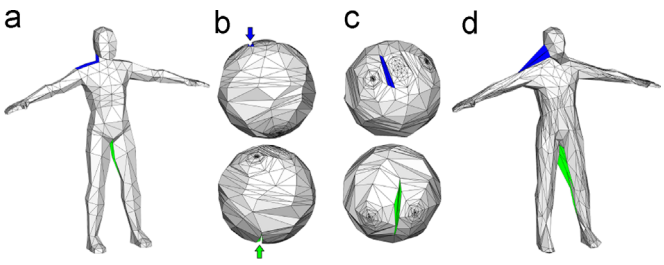


Fig. 1. *Depression issues in spherical representations.* (a) shows the original model with the highlights on low vertex density regions. (b) and (c) show the depressions on the sphere from different view angles. (d) is a remeshed model using (a) and another 3D model. Because of the depressions, the remeshed model has unexpected distortions in colored regions. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

2.2. Relaxing method

A relaxing method updates vertex positions repeatedly based on curvature. Gotsman et al. [4] cut out one triangle to create a boundary. Boundary vertices were projected to a convex boundary, and a barycentric formulation was used to obtain the position of each vertex as a function of its neighbors. Zhang [14] projected a 3D mesh onto a sphere in a brute-force manner, and then relaxed overlapped regions. Mocanu and Zaharia [15] flattened a mesh based on Gaussian curvature. Kazhdan et al. [16] employed a modified mean-curvature flow. Wang et al. [6] minimized ARAP energy, which is an extension of ARAP planar parameterization.

Those relaxing methods adopt global optimization strategies (e.g., minimizing flat energies [17]) and provide smooth transitions across the spherical surface. Edges are weighted with energy values to evaluate vertex distortion. The vertex with the most significant distortion changes its position by minimizing the energy of edges it connects with. This process could repeat until energy reaches zero. However, a desired parameterization result is usually obtained at a non-zero energy. Minimizing energy down to zero degenerates triangles; eventually, the sphere collapses to a point (see Fig. 2). Many relaxing methods, as mentioned by Floater and Hormann [18], have to use a threshold to limit the number of optimizing iterations. Another issue is that, relaxing methods push features into small regions on the sphere, which will be hard to restore after topology merging.

Our curvature reduction criteria for parameterization measures angle and area distortion and triangle aspect ratios. Comparing to previous work, our curvature reduction criterion alleviates the degree of depression that otherwise would occur with progressive methods. Our global optimization algorithm looses dense regions and compact sparse regions on the sphere to avoid triangle degeneration and foldover issues. Our approach is suitable for remeshing and morphing applications.

3. System overview

Fig. 3 illustrates processing stages of our approach:

- Simplification: this stage takes a genus-zero triangle mesh and simplifies it to a tetrahedron by collapsing edges. As a result, a progressive representation [19] is generated, which consists of the tetrahedron and a sequence of vertex split operations (inverse of the edge collapses) that are used to reconstruct the original mesh.
- Parameterization and optimization: With vertex split operations, vertices are inserted back one at a time. A spherical embedding is preserved by minimizing parametric distortion using our local distortion metric. A global distortion minimization is executed intermittently to ensure an appropriate vertex distribution on the sphere. Our parameterization and optimization methods are generic and can be used in conjunction with various parameterization strategies. As a result, we obtain a spherical representation with one-to-one vertex and triangle correspondences to the original mesh.

Fig. 2. *Degeneration problem while parameterizing based on relaxation or energy reduction.* As iterations of the minimization proceed, the minimizer fails to produce an acceptable configuration. Eventually the entire mesh will collapse to a point.

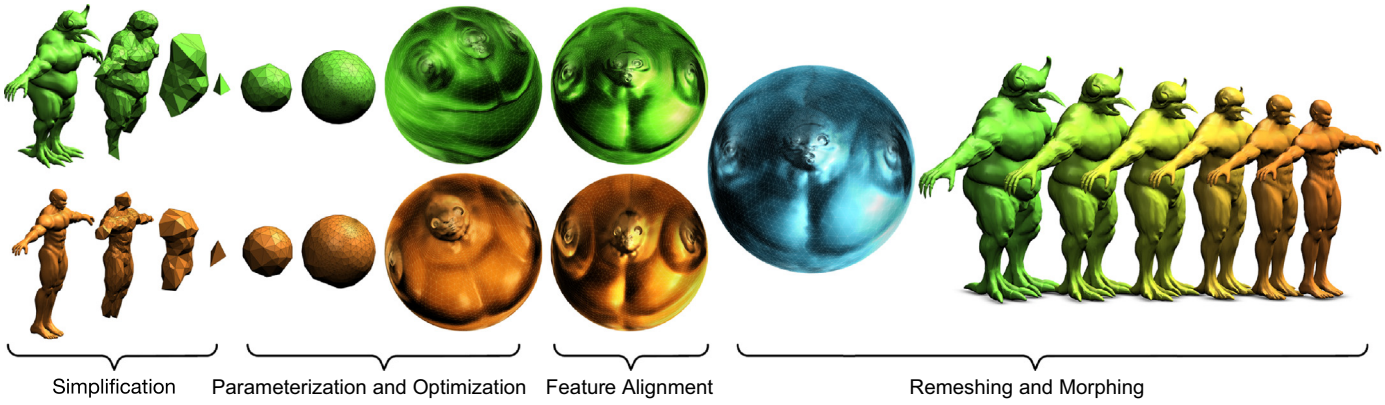


Fig. 3. *The system overview of our approach.*

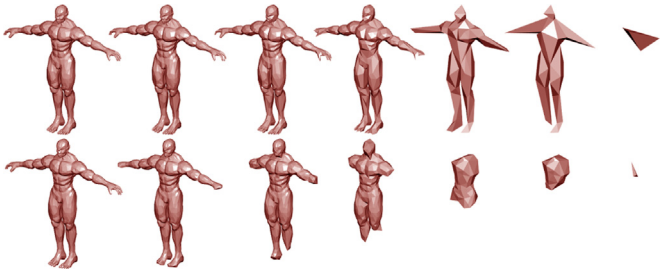


Fig. 4. *Mesh simplification examples.* The first row shows the sequence of simplified meshes using QEM [20] that preserves curvature. The second row shows the sequence of simplified meshes using our criteria that removes curvature and creates rounder and coarser versions of the mesh. The numbers of vertices from left to right are: 2053, 1500, 1000, 500, 100, 50, and 4.

- Feature alignment: this stage modifies the spherical representations of two meshes and aligns regions according to user-specified feature constraints.
- Remeshing and morphing: this stage generates a *supermesh* that represents appearance of both original meshes. A morph is achieved through a linear interpolation of corresponding vertex positions inherited from the original meshes.

4. Simplification

A mesh M is simplified to a tetrahedron M_0 (base mesh) by collapsing edges. When an edge is collapsed, one vertex is removed. M_0 is convex and vertices of M_0 are visible from the centroid, so it can be directly projected to the sphere. Many simplification criteria (e.g., mesh decimation [21] and QEM [20], appearance-preserving [22]) favor curvature preservation. As shown in the first row of Fig. 4, the meshes composed of less than 100 vertices contain arms and legs and are not convex. When parameterizing them, non-convex regions will cause distortion issues.

Our criteria favor removing curvature as early as possible. When simplifying the original mesh towards M_0, coarse versions tend to be rounded, as shown in the second row of Fig. 4. At each collapsing operation, we choose an edge $e(v_1, v_2)$, which is thereby collapsed by merging vertex v_1 to v_2. A cost function is defined in Eq. (1). All edges are evaluated using the cost function, and the edge with the lowest cost value is collapsed:

$$cost(v_1, v_2) = \underbrace{\left(\frac{1}{1+(2\pi-\sum_{\epsilon \in ring(v_1)}\alpha_\epsilon)/(\sum_{\tau \in neighbor(v_1)}A_\tau/3)}\right)}_{\text{Gaussian curvature}} \underbrace{\left(\sum_{\tau \in neighbor(v_2)}\frac{cr(\tau)}{2 \times ir(\tau)}\right)}_{\text{aspect ratio}} \quad (1)$$

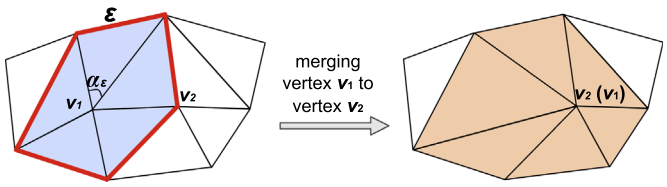


Fig. 5. *Neighbor triangles used in Eq. (1).* The red line indicates the 1-ring of vertex v_1. Neighbor triangles of vertex v_1 (blue) are used to calculate the Gaussian curvature of vertex v_1. Neighbor triangles of vertex v_2 (orange) are used to calculate the triangle aspect ratio when collapsing vertex v_1 to v_2. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Eq. (1) considers Gaussian curvature at v_1 and the triangle aspect ratio of the neighborhood. Fig. 5 illustrates neighbor triangles used in Eq. (1). α_ϵ is the angle at v_1 opposite to the boundary edge ϵ in the 1-ring of v_1. The 1-ring is formed by neighbor vertices. τ is a neighbor triangle and A_τ is the area of τ. The triangle aspect ratio is defined as circumradius of the triangle ($cr(\tau)$) divided by twice of inradius ($ir(\tau)$). This term has a minimum value of 1 for an equilateral triangle, and the value increases for sheared triangles.

5. Parameterization

The goal of parameterization is to form a mapping $f : M \rightarrow S$, which maps every vertex of M onto the sphere S. To achieve this, we first parameterize M_0 to S, which is an obvious process since M_0 is convex. In practice, rays are constructed by shooting from the centroid of M_0 to its vertices. S is set to be at the centroid of M_0, and the radius makes the surface area of S equal to the surface area of M_0. The parameterized vertex is the intersection between the ray and the surface of S.

After initial parameterization with M_0, remaining vertices of M (those removed during simplification) are inserted back iteratively. At each iteration, only one vertex is inserted. A later removed vertex is earlier split from where it was merged to [19], and is inserted back to the mesh and then parameterized on the sphere. Theoretically, the vertex could be placed at any position on the sphere; however, placing it at an inappropriate position would cause foldovers.

We introduce a *spherical kernel* for each inserted vertex. We show that placing the vertex inside this kernel guarantees a valid embedding (see Section 5.1). To minimize distortion of vertex placement, we present an efficient method that measures distortion for a set of candidate positions and places the vertex at the optimal position (see Section 5.2).

5.1. Kernel sampling

Inserting a vertex back to the mesh will connect the vertex to neighbor vertices, and will form a 1-ring domain (like a 3D umbrella). Inserting a vertex is equivalent to the reverse operation of edge-collapsing. Using Fig. 5 as an example, v_2 in the right image would be split into two vertices v_1 and v_2 in the left image. As a result, this vertex-splitting operation produces a 1-ring domain for v_2 (see red edges in Fig. 5). A spherical kernel is formed by the intersection of open hemispheres which are defined by the spherical polygon edges created by projecting the 1-ring edges with respect to the centroid of the sphere. As addressed by Praun and Hoppe [3], a spherical kernel is a spherical polygon on the surface of sphere. Parameterization will maintain a valid embedding as long as the vertex is projected inside the spherical kernel.

To fast find the optimal position, we adopt a barycentric interpolation sampling method. Direct sampling in the spherical kernel may cause interpolation problems and will be at a high computational cost. Instead, we generate sample points inside parameterized triangles of the 1-ring (see Fig. 6(a)). We use Eq. (2) to generate sample points in each triangle, where p is an interpolated point from three vertices of the triangle, v_1, v_2 and v_3; the equation has s, $t \in (0,1)$ and $s+t<1$, and their values vary based on a predefined stepping value:

$$p = s \cdot v_1 + t \cdot v_2 + (1-s-t) \cdot v_3 \tag{2}$$

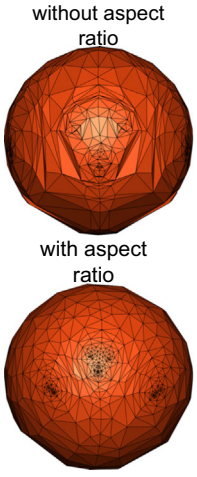
Some sample points will cause triangle foldovers because they are outside the spherical kernel so they must be removed. As shown in Fig. 6(b), we test sample points against the planes formed by the 1-ring edges and the centroid of the sphere. If a sample point is above all planes, it is valid; otherwise, it is outside the kernel and removed (see Fig. 6(c)). Rays shoot from the centroid to those valid sample points. and they intersect with the surface of the sphere. We call the intersections as *kernel points*, which are the possible positions to place the vertex.

5.2. Distortion measurement

We want to find the optimal kernel point that has the minimum distortion. We develop an error function (see Eq. (3)) to measure the degree of distortion of each kernel point. This function can be interpreted as Dirichlet energy per parameter area [23], which is a combination of angle distortion, area distortion and triangle aspect ratio in the 1-ring of the vertex. The optimal kernel point will be the one with the lowest error value.

$$error(p) = \sum_{\tau \in neighbor()} \underbrace{\left(\frac{\sum_{i\in[1,3]} \cot'\alpha_i \times \|\epsilon_i\|^2}{2 \times A_\tau}\right)}_{\text{angle distortion}} \times \underbrace{\left(\frac{A'_\tau}{A_\tau} + \frac{A_\tau}{A'_\tau}\right)^{\varrho}}_{\text{area distortion}} \times \underbrace{\left(\frac{cr(\tau)}{2 \times ir(\tau)}\right)^{\varphi}}_{\text{aspect ratio}} \tag{3}$$

In Eq. (3), τ represents a triangle in the 1-ring of the vertex. A_τ is the area of the triangle on the mesh. ϵ_i represents an edge of τ on the mesh. α_i is the angle opposite to ϵ_i on the sphere. A'_τ is the area of the triangle on the sphere. Notations of the aspect ratio term are the same as Eq. (1). ϱ and φ are positive values to control relative importance of angle, area and aspect ratio. The benefits of using angle and area terms were discussed in [18,24]: preserving angles reduces the shear of parameterized triangles, and preserving areas prevents features from squeezing into small regions. Nadeem et al. [25] proposed parameterization algorithms that balance angle and area distortion, but using only the angle and area distortion terms may not faithfully reproduce geometric features during remeshing if original meshes are not well tessellated or contain sliver triangles. To avoid occurrence of such problems, we add an aspect radio term. As shown in the inset on the right, without the aspect radio term, the parameterization result is harsh and rigid, and produces undesirable depression on the sphere. We also observe that the error value varies with the sphere size change, because the area distortion term in Eq. (3) is not invariant to scale. If the sphere size is significantly larger or smaller than the size of the original mesh, the area distortion term will have a major impact on distortion calculation. The lack of invariance can be alleviated by normalizing the area distortion with respect to the radius of the sphere. Therefore, we calculate a new radius for the sphere at each time a vertex is inserted. The new radius is calculated using Eq. (4), which ensures that the surface area of the sphere is equal to the surface area of M_i. In Eq. (4), M_i is obtained after inserting i vertices, and τ is a triangle of M_i:



$$radius = \sqrt{\left(\sum_{\tau \in M_i} A_\tau\right)/(4 \times \pi)} \tag{4}$$

6. Optimization

Using the optimal kernel point optimizes vertex placement locally in the 1-ring domain. Continuing to insert vertices will make certain regions of the sphere become dense. Eventually, small regions may contain many geometric features but large regions are sparse, and features cannot be preserved in those regions. We develop a global optimization method to appropriately loose dense regions and compact sparse regions.

Algorithm 1. Optimization.

Input: The original mesh M and its spherical parameterization S
Output: Optimized vertices
1: **procedure** Optimize-Distortion (M, S)
2: **do**
3: $maxDisp = 0$;
4: sort $S.V$ in decreasing order of $neiDisp$

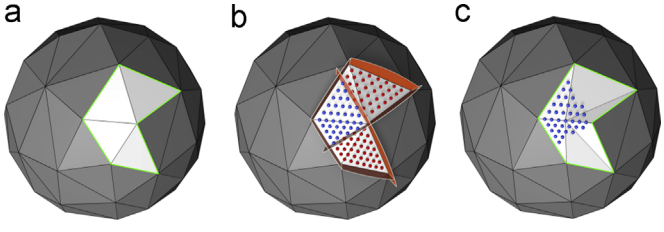


Fig. 6. *Kernel sampling and candidate clipping.* (a) highlights a parameterized 1-ring domain (the region by green lines); (b) demonstrates all sample points in the triangles of the 1-ring. They are tested against the planes (in orange). Blue ones are valid sample points while the red ones are invalid and removed; (c) shows only the valid sample points, which are projected to the spherical surface to create kernel points. The optimal position for placing the inserted vertex will be chosen from the kernel points. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

5: **for** each $v \in S.V$ **do**
6: generate sample points in v's 1-ring; ▷ Eq. (2)
7: calculate distortion errors of the kernel points; ▷ Eq. (3)
8: $pos =$ position of the kernel point with smallest error;
9: $v.disp = pos - v.pos$;
10: $v.pos = pos$;
11: **for** $x_i \in v.neiVerts$ **do**
12: $x_i.neiDisp = \sum y_i.disp$, where $y_i \in x_i.neiVerts$;
13: **end for**
14: **if** $v.disp > maxDisp$ **then**
15: $maxDisp = v.disp$;
16: **end if**
17: **end for**
18: **while** $maxDisp > disp\_threshold$ and not reaching the maximum number of iterations
19: **end procedure**

We reposition vertices after a certain amount of them are inserted. A *displacement* (*disp*) is defined for each vertex of the spherical representation *S*. It is the distance from the previous position of the vertex to the new position. We also employ a *neighbor displacement* (*neiDisp*) for each vertex, which is the sum of displacements of neighbor vertices due to repositioning. Algorithm 1 describes the detail. Vertices ($S.V$) are sorted based on the decreasing order of *neiDisp*. Kernel points are generated for each vertex in its spherical kernel, and the most favorable kernel point is chosen as the new position and used for computing *disp* (see lines 6–10). The displacement is propagated to *neiDisps* of neighbor vertices (see lines 11–13). During the optimizing process, we maintain a maximum displacement (*maxDisp*) of all vertices. The process is stopped when *maxDisp* is smaller than a predefined threshold *disp_threshold* or over a predefined maximum number of iterations. At the first iteration, *disp* and *neiDisp* are zero for every vertex, and we start the process with the last inserted vertex and its neighbor vertices.

disp_threshold is a convergence threshold. If the threshold value is too small, the optimizing process may take too many iterations or not converge; if the threshold value is too large, convergence is achieved but the number of iterations may not be enough to appropriately reposition the vertices in dense regions. Based on our experiments, a threshold value ranged in $(10^{-8}, 10^{-3})$ helps generate a descent optimization result. The maximum number of iterations (see line 18 in Algorithm 1) is used to stop the optimizing process when a failure of convergence occurs. Fig. 7 is an example illustrating the optimizing process.

7. Feature alignment

Given two meshes M_1 and M_2, we select the equal number of feature points and align them in their spherical representations S_1 and S_2. We first roughly align feature points, by rotating one sphere using a least-squares approach that brings the feature points close to those on the other sphere. We then exactly coincide the corresponding feature points using radial basis function (RBF) interpolation.

Rough alignment is crucial in order to achieve good quality of warping. Without the rough alignment prior to the exact alignment, parameterized triangles may be degenerated or folded over like the example shown in Fig. 8. We assume that S_1 and S_2 have been normalized to unit spheres and they are centered at (0, 0, 0). We use $\{\mathbf{v}_{i,1}\}_{i=1}^{n_1}$ to denote the set of vertices in S_1, where n_1 is the total number of vertices; similarly, we use $\{\mathbf{v}_{i,2}\}_{i=1}^{n_2}$ to denote vertices in S_2.

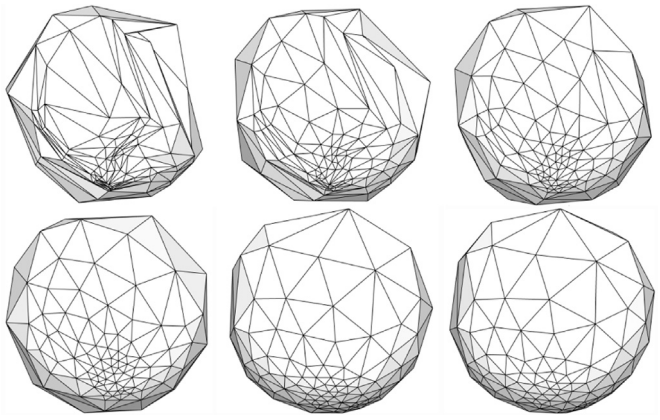


Fig. 7. *Parameterization improvement with the global optimization.* From top-left to bottom-right, the parameterization result is gradually improved by repositioning vertices. The spherical representation is for the Cow model and has 100 vertices inserted. The sequence of optimization results correspond to 1, 2, 4, 10, 20, and 30 iterations. The optimization converges after 30 iterations with *disp_threshold* set to 10^{-6}.

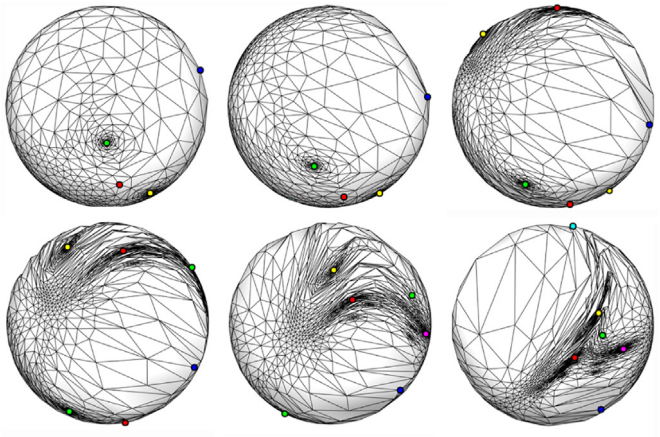


Fig. 8. *The triangle degeneration problem due to not having the rough alignment.* Colored dots are the feature points. From top-left to bottom-right, images are the results after 0, 2, 4, 6, 8, and 10 iterations of the RBF algorithm. Towards the end of the process, many triangles are very close to each other and become collinear. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

7.1. Rough alignment

The problem of rough alignment is a typical form of Procrustes problem [26]. It occurred in computer vision applications [27] and was solved with non-iterative algorithm involving singular value decomposition (SVD). We adopt the same idea and give a closed-form solution to align feature points in the spherical domain.

Let us pick a set of q feature points, denoted as $\{\mathbf{p}_{k,1}\}_{k=1}^{q}$ and $\{\mathbf{p}_{k,2}\}_{k=1}^{q}$, where $\mathbf{p}_{k,1}$ represents the position of the kth feature point in S_1. We seek a rotation matrix R for S_1, which minimizes the sum of squared distances to corresponding feature points in S_2, as shown in the following equation:

$$\Sigma^2 = \sum_{k=1}^{q} \| \mathbf{p}_{k,2} - \mathbf{R} \times \mathbf{p}_{k,1} \|^2 \tag{5}$$

To obtain R, we employ a SVD algorithm. A 3×3 matrix $H = \sum_{k=1}^{q} \mathbf{p}_{k,1}\mathbf{p}_{k,2}^T$ is calculated. H can be represented as the product of three matrices such that $H = USV^T$, where U and V are orthogonal matrices and S is a diagonal matrix. Finding these matrices needs eigenvalues and eigenvectors of HH^T and H^{TH}. Columns of U are

eigenvectors of HH^T and columns of V are eigenvectors of H^{TH}. Column vectors in U and V are ordered by the size of the corresponding eigenvalue (the eigenvector of the largest eigenvalue is the first column). S contains the square roots of eigenvalues from U or V in descending order. Then, rotation matrix R is derived as $R=VU^T$. If the determinant of R is equal to 1, R is valid. If it is equal to -1, the algorithm fails to find R. Failures rarely happen and we do not encounter any failure in our experiments. Theoretically, the determinant can be -1 when R is a reflection matrix rather than a rotation matrix. This could happen in the case that one or both of the feature point sets, $\{\mathbf{p}_{k,1}\}$ and $\{\mathbf{p}_{k,2}\}$, are coplanar (i.e., $q=3$). If this case happens, the desired rotation matrix can be recalculated as $R=V'U^T$, where V' is from V by inverting the sign of the third column.

7.2. Exact alignment

Moving feature points of one sphere to the other would introduce noticeable distortion (see Fig. 9). It is generally better to move feature points of both spheres to their midpoints on the great circle arcs. Let $\{\mathbf{m}_k\}_{k=1}^{q}$ be the set of midpoints, where $\mathbf{m}_i=(\mathbf{p}_{i,1}+\mathbf{p}_{i,2})/\mathbf{2}$. Since $\mathbf{m}_i$ is the Euclidean average of the points, it lies in the interior of the sphere rather than on the surface of the sphere. $\mathbf{m}_i$ is projected back onto the surface before performing the exact alignment algorithm. This is indeed just a simple way of computing the great-circle midpoint on the sphere.

We seek two sets of coefficient vectors $\{\mathbf{c}_{k,1}\}_{k=1}^{q}$ and $\{\mathbf{c}_{k,2}\}_{k=1}^{q}$ and two polynomials $\mathbf{P}^{(1)}$ and $\mathbf{P}^{(2)}$, such that for each vertex $\mathbf{v}_{i,1}$ in S_1 and $\mathbf{v}_{i,2}$ in S_2, the displacement functions $\mathbf{d}(\mathbf{v}_{i,1})$ and $\mathbf{d}(\mathbf{v}_{i,2})$ are given by:

$$\mathbf{d}(\mathbf{v}_{i,1})=\sum_{\mathbf{k=1}}^{\mathbf{q}}\phi(||\mathbf{v}_{i,1}-\mathbf{p}_{k,1}||)\mathbf{c}_{k,1}+\mathbf{P}^{(1)}(\mathbf{v}_{i,1}) \tag{6}$$

$$\mathbf{d}(\mathbf{v}_{i,2})=\sum_{\mathbf{k=1}}^{\mathbf{q}}\phi(||\mathbf{v}_{i,2}-\mathbf{p}_{k,2}||)\mathbf{c}_{k,2}+\mathbf{P}^{(2)}(\mathbf{v}_{i,2}) \tag{7}$$

where $\phi(\cdot)$ is a RBF equation. $\mathbf{P}^{(j)}(\mathbf{v})=\langle\mathbf{p}_1^j(\mathbf{v}),\mathbf{p}_2^j(\mathbf{v}),\mathbf{p}_3^j(\mathbf{v})\rangle$ is a vector valued polynomial, and p_1^j, p_2^j, p_3^j are linear polynomials. Based on the study presented by Deboer et al. [28], we use $\phi(r)=1-30r^2-10r^3+45r^4-6r^5-60r^3\log(r)$. After the exact alignment, the new position of a vertex is given by $\mathbf{v}_i'=\mathbf{v}_i+\mathbf{d}(\mathbf{v}_i)$.

In general, the number of feature points q is far less than the number of vertices ($<1\%$), so we are in favor of weighting vertices for displacement based on proximity to feature points. Specifically, nearby vertices are weighted more such that they obtain a larger amount of displacement to follow the feature points, and far-away vertices are less influenced. The displacement values of feature points are known (see Eq. (8)), and they are used to obtain coefficients $\mathbf{c}_k$ and polynomials p_1, p_2, p_3:

$$\mathbf{d}(\mathbf{p}_i)=\mathbf{m}_i-\mathbf{p}_i \tag{8}$$

Deboer et al. [28] presented a matrix formulation in order to solve such a system of equations. We adopt the matrix formulation to solve the 3D problem.

In Fig. 10, we demonstrate the steps of warping during the process of exact alignment. Comparing to Fig. 8, the exact alignment is accomplished with no foldover or degeneration issues. Feature points are guaranteed to be aligned exactly to midpoint positions, and they are used to interpolate vertex correspondences of the rest of the meshes. The input genus-zero meshes are homeomorphic to each other after initial parameterization, where each vertex on S_1 has a bijective projection on S_2. During warping of the spheres for feature alignment, vertices move along geodesic on the sphere, and homeomorphism still exists between S_1 and S_2, since the alignment algorithms only stretch and bend vertices on the surface of the sphere and do not change topological properties. After that, each vertex on S_1 has a new bijective projection on S_2 which matches the apparent correspondence of the 3D shape. Vertices of S_2 obey the same procedure.

8. Remeshing and morphing

After establishing feature-aligned maps $f_1: M_1\rightarrow S_1$ and $f_2: M_2\rightarrow S_2$, we construct a *supermesh* S_M, which is a semi-regular spherical mesh, from S_1 and S_2. We then obtain M_1' and M_2', which are isomorphic, using inverse maps $S_M\overset{f_1^{-1}}{\Longrightarrow}M_1'$ and $S_M\overset{f_2^{-1}}{\Longrightarrow}M_2'$. Isomorphic meshes are compatible to each other because they have a correspondence between their vertices, and the vertices will be interpolated to animate the effects of shape blending from one to the other.

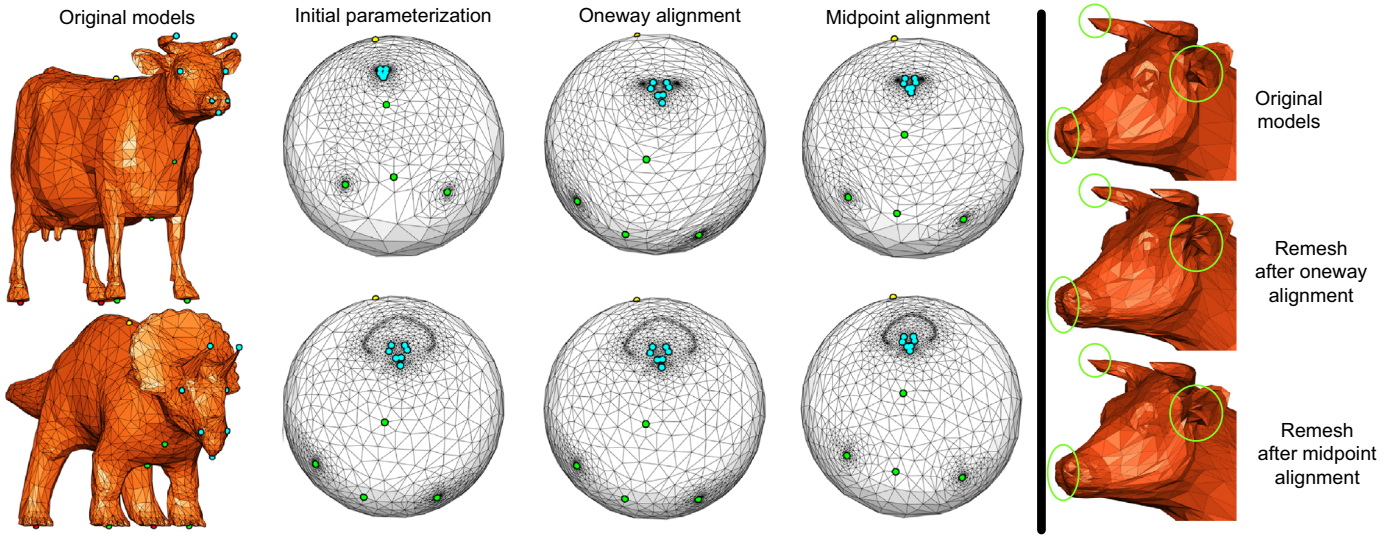


Fig. 9. *Comparison of two exact alignment methods.* In (a), the first column shows the original Cow and Triceratops with feature points highlighted; the second column shows the spherical parameterizations. The third column shows the results of oneway alignment by moving the feature points of the Cow to the corresponding ones of the Triceratops; the fourth column shows the result of moving the feature points of both parameterizations to their midpoints. (b) shows the appearance differences after remeshing, where both models are combined (see Section 8). Comparing to the appearance of the original Cow, the midpoint alignment method better preserves features. Oneway alignment causes artifacts (as circled) on the horns, some problems in the snout, and significant distortions in the ears.

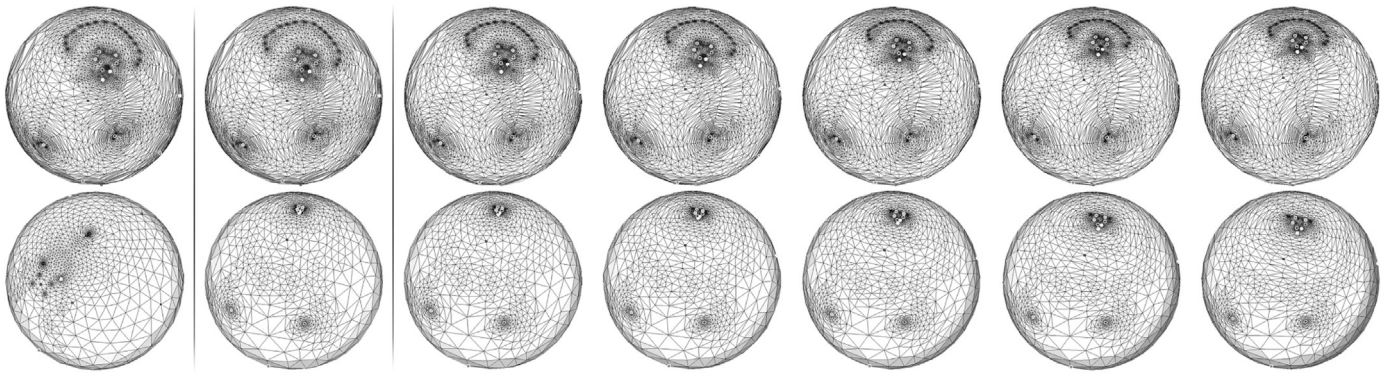


Fig. 10. *Steps of warping during the process of exact alignment.* First column shows the results of initial parameterization. Second column shows the result of midpoint rough alignment. The sequence of images in the subsequent columns shows the gradual change during the process of a 10-step exact alignment, recorded at 2-step intervals.

The process to generate the supermesh is *remeshing*. One possible remeshing method would be overlaying one sphere on the other and then calculating edge intersections. However, in addition to being complex in the implementation, this method would generate a very dense supermesh, in which the number of vertices may be an order of magnitude higher than the number of vertices in the input meshes [29]. Another possible method would be placing vertices of S_2 into triangles of S_1, and then casting a ray from the center of S_1 for each vertex of S_2 and intersecting the ray with the surface of S_1. The supermesh would be constructed by creating edges between each intersection and vertices of the triangle that it is placed in. Though the supermesh would represent M_1 very well (because original vertices and edges in M_1 are preserved), it would produce a poor approximation for M_2. We tried to use convex hull algorithms (e.g., [30]) to construct the supermesh. All vertices of S_1 and S_2 were placed on a spherical surface, and a convex hull was built in a divide-and-conquer manner. As shown in Fig. 11, the convex hull can produce good visual quality on the regions with low curvature, but noticeable artifacts occur in the regions with sharp geometric features such as ankles and heels.

We adopt a subdivision method (see Fig. 12). We start building S_M with an octahedron, which is embedded in a unit sphere and will be progressively refined. In each triangle of S_M, we identify the number of vertices of S_1 overlaid in the triangle by shooting rays from the center of the unit sphere to vertices and checking for intersections. If the triangle contains more than one intersections, we subdivide it into 4 equal area triangles. This process is repeated until each triangle in S_M contains at most one vertex of S_1. We overlay S_2 to S_M and apply the same subdividing strategy.

Appearance of M_2' is consistent to M_2 since those vertices originally from S_2 can be mapped back to where they were in M_2. However, a mapping is not established yet for those vertices to contribute to appearance of M_1'. They do not have corresponding position information from M_1. The vertices originally from S_1 will have the same issue when they are used in M_2'.

To solve this issue, for each vertex from S_2, we find the triangle in S_1 it intersects with. Barycentric coordinates at the intersection are used to interpolate from the three vertex positions of the triangle in M_1. The same process is applied to the vertices from S_1 to find interpolated positions using S_2 and M_2.

A morph can be animated by linearly interpolating vertex positions between M_1' and M_2'. The intermediate position of a vertex is obtained from Eq. (9), where $\mathbf{v}_i$ is the intermediate position at the time t for the ith vertex, $\mathbf{v}_i^1$ and $\mathbf{v}_i^2$ are the ith vertex in M_1' and M_2', respectively:

$$\mathbf{v}_i' = t\mathbf{v}_i^1 + (1-t)\mathbf{v}_i^2 \tag{9}$$

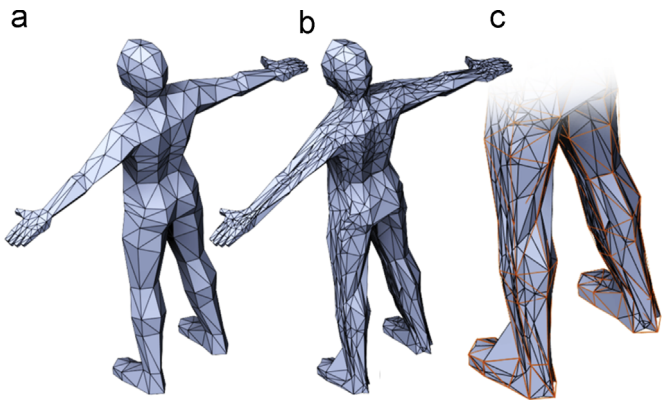


Fig. 11. *The problem of remeshing due to use of a convex hull.* (a) is the original model. (b) is the remeshed model of (a) after the cross parameterization with another model. (c) shows the original topology (orange lines) overlaid on the remeshed model. The original vertices are present, but the way they are connected causes the loss of the geometry features in many areas (e.g., ankles and heels). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

9. Experimental results

We have applied our approach to a number of triangular meshes to create spherical representations, supermeshes and morphs. The meshes have various geometric complexities. In this section, we focus on performance analysis and compare our approach to other state-of-art approaches. We also discuss the qualities of remeshing and morphing.

9.1. Performance

We have implemented our system on a Intel core i7 3.60 Hz machine. Our parameterization rate is 58K triangles/minute in average. Praun and Hoppe [3] presented a progressive method for parameterization and achieved a rate of 8K triangles/minute. Kraevoy and Sheffer [1] presented a cross parameterization and remeshing method and had about 5K triangles/minute. Saba et al. [31] provided an efficient numerical scheme to spherically parameterize meshes and achieved about 7.4K triangles/minute. Friedel et al. [17] employed an energy minimization to generate spherical representations and achieved about 14K triangles/minute. Athanasiadis et al. [32] presented a feature-based parameterization method and achieved about 1.1K triangles/minute. Aigerman [8] presented a parameterization method for surface property mapping and achieved about 3.5K triangles/minute. Fu et al. [33] parallelized the AMIP method using OpenMP in C++ and had about 150k triangles/minute, and they had 30k triangles/minute in the serialized implementation. Nadeem et al.

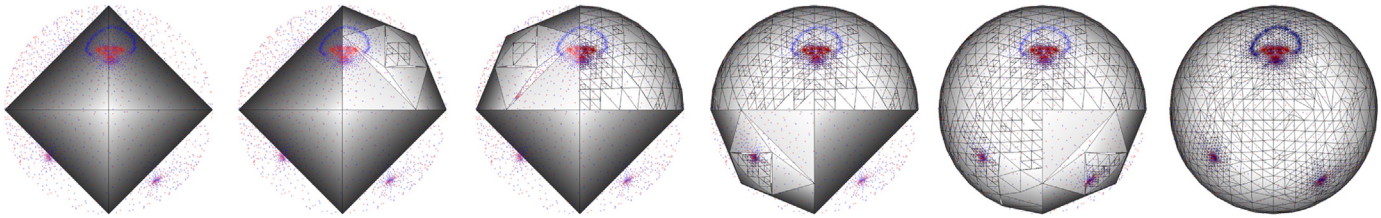


Fig. 12. *A subdivision example to construct the supermesh.* The remeshing process starts with an octahedron and refines it repeatedly. Red and blue dots represent vertices of the spherical representations S_1 (Cow) and S_2 (Triceratops), respectively. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Table 1
Time breakdowns for parameterization.

Models	# of vertices	# of triangles	Simplification time (s)	Parameterization		Optimization		Total
				Time (s)	Avg. # of kernel points	Time (s)	Avg. # of iterations	Time (s)
Dolphin	1867	3730	0.31 (8.0%)	0.24 (6.2%)	83.3	3.33 (85.8%)	19.9	3.88
Cow	2000	3996	0.32 (8.8%)	0.24 (6.6%)	86.5	3.06 (84.5%)	18.8	3.62
Horse	2000	3996	0.34 (6.3%)	0.36 (6.7%)	86.6	4.71(87.0%)	21.6	5.41
Eagle	2135	4266	0.39 (11.9%)	0.28 (8.5%)	87.1	2.61 (79.6%)	10.6	3.28
Duck	2497	4990	0.41 (10.2%)	0.37 (9.2%)	93.4	3.24 (80.6%)	10.4	4.02
Triceratops	2500	4996	0.39 (9.5%)	0.33 (8.1%)	89.7	3.37 (82.4%)	17.5	4.09
David	5197	10,390	0.66 (7.7%)	0.78 (9.1%)	88.0	7.13 (83.2%)	12.4	8.57
Woman (Run)	5676	11,348	0.76 (7.3%)	0.69 (6.6%)	92.4	8.95 (86.1%)	21.0	10.40
Woman (Dance)	5691	11,378	0.77 (7.2%)	0.70 (6.6%)	92.3	9.21 (86.2%)	23.6	10.68
Manhead	10,331	20,658	1.27 (6.1%)	2.09 (10.1%)	92.1	17.42 (83.8%)	15.1	20.78
Monster	24,454	48,904	4.27 (6.6%)	8.09 (12.4%)	94.6	52.83 (81.0%)	21.1	65.19
Elf	34,194	68,384	6.87 (6.4%)	14.65 (13.7%)	97.2	85.63 (79.9%)	22.9	107.15
Merman	43,105	86,206	7.35 (4.4%)	23.08 (14.0%)	96.5	134.93 (81.6%)	23.4	165.36

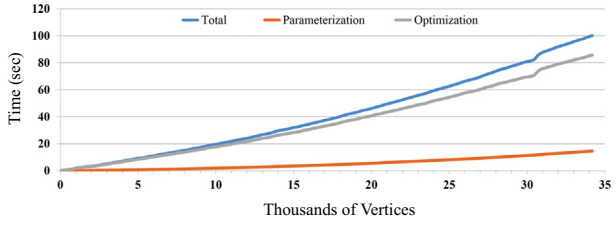


Fig. 13. The performance graph of parameterization and optimization for the Elf model while vertices are progressively inserted.

[25] proposed a divide-and-conquer method to maintain a good balance between angle and area distortion, and they achieved about 2.5K triangles/minute for balanced mapping.

GPU-accelerated methods were also proposed. Athanasiadis and Fudos [34] presented a parallel approach that can parameterize up to 400K triangles in less than 25 s. The authors adopted a local optimization presented by Sander et al. [35] which reduced their GPU computational time up to 50%. The local optimization is parallelizable since it avoids data dependency which otherwise would be introduced in a global optimization. In our approach, using only a local optimization is not sufficient. We rely on the global optimization to reposition locally parameterized vertices so as to achieve a minimum distortion and a global bijectivity (see Section 6). Due to the inter-vertex dependency at the iterative computation of displacement values, our global optimization algorithm is not parallelizable.

Spherical parameterization. The generation of a spherical representation consists of three stages: *simplification*, *parameterization* and *optimization*. Table 1 shows the breakdowns of timing results. Simplification stage linearly scales with the number of edges; it never becomes a performance bottleneck. For every tested model, it takes less than 10% of the total time.

Table 2
Experimental result for feature alignment and remeshing.

Pair of models	Alignment time (s)	Remeshing time (s)	Supermesh		# of Ctrl. points
			# of Vert.	# of Tri.	
Cow + Triceratops	0.03	1.11	3905	7696	17
Cow + Horse	0.03	0.83	3333	6540	17
Eagle + Duck	0.02	1.32	4503	8886	12
David + Manhead	0.09	12.88	13,078	25,874	20
Woman (Run + Dance)	0.73	7.47	9021	18,038	57
Monster + Elf	0.29	187.52	51,724	103,022	18
Dolphin + Merman	0.11	173.86	57,568	114,728	8

Parameterization stage is fast; it takes 6.2–13.8% of the total time. Time spent on placing a vertex on the sphere is determined by the number of kernel points generated in the spherical kernel. The number of kernel points is controlled by a stepping value (see Section 5.1). In our experiment, we set the stepping value to 0.11.

Optimization stage takes 3.33–134.93 s for the models used in our experiment. It dominates the total time (over 80%) due to the ping-pong effect during the iterations in Algorithm 1 (see Section 6). The displacement threshold is set to 10^{-6} for every tested model. For complex models, optimization is first executed after 25 vertices are inserted, and then this number is slowly increased by a factor of 1.25 after each execution. The more vertices a mesh has, the longer time is needed to spread displacement values across the surface of the sphere. This is why the Monster and Elf take longer time than others. Fig. 13 illustrates the performance graph of parameterization and optimization stages for the Elf model. The number of iterations is also

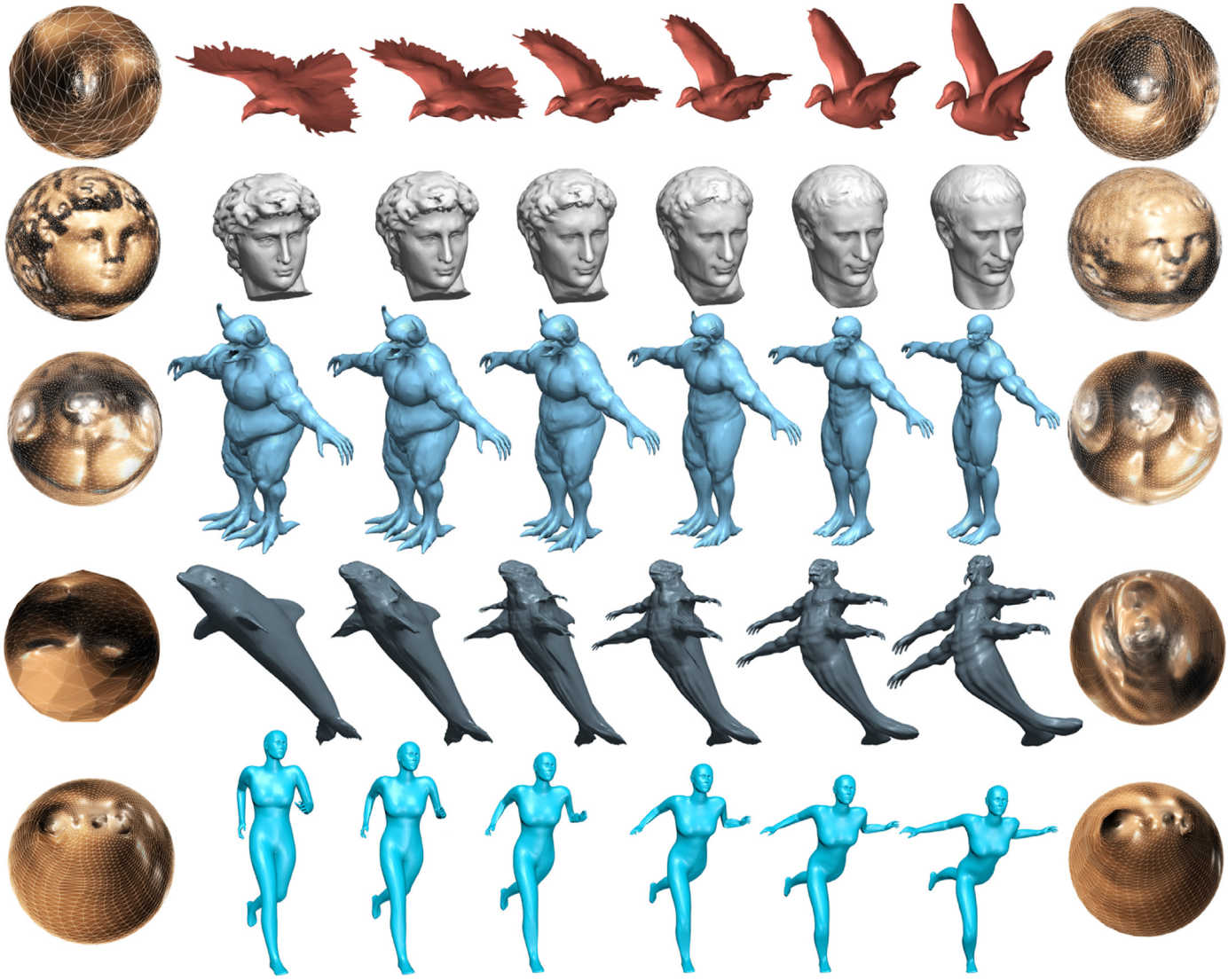


Fig. 14. *Morphing examples.* A normal mapping technique is applied on the spherical representations to display hints of shape casting.

Fig. 15. Morphing examples among Cow, Horse and Triceratops.

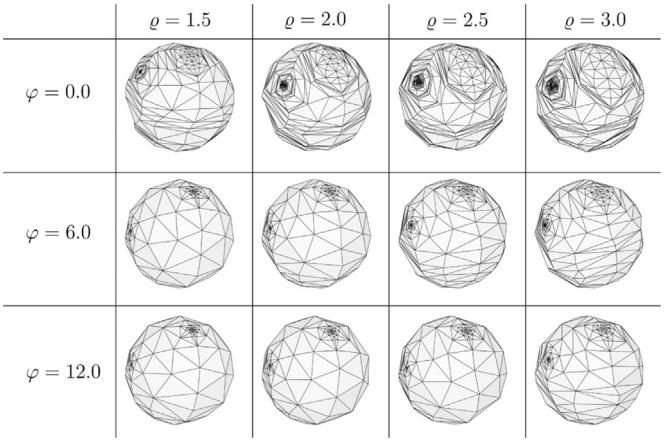


Fig. 16. Effects of area preservation and aspect ratio terms.

influenced by the degree of geometric deformation. For the Cow, Horse and Eagle, they all have about 2K vertices, but the Cow and Horse require more iterations than the Eagle because of the costly optimizing process on the appendages like legs, the tail, horns, etc.

Feature alignment and remeshing. Table 2 shows the experimental results of feature alignment and remeshing. The time of feature alignment is less than 1 s. Alignment performance is influenced by the geometric complexity of input meshes and the number of feature points. That is why the Woman (Run+Dance)

and Monster+Elf take more time for alignment than other pairs. For the Woman (Run+Dance), 57 feature points are set around micro-features like knees, fingertips, knuckles, etc., which slow down the execution of the matrix solver. Table 2 suggests that remeshing is time-consuming. This is because every vertex has to be tested for an intersection with triangles on the supermesh.

9.2. Quality

Figs. 14 and 15 illustrate morphing results. Comparing to blendshape techniques (e.g., [36,37]), our approach accepts the meshes with different topologies and different numbers of geometric primitives.

Eq. (3) in Section 5.2 describes the exponent ϱ of the area distortion term and the exponent φ of the aspect ratio term. A high value of ϱ may create sheared triangles and narrow the kernel. Increasing the value of φ can reduce the influence of area distortion if they are not balanced. Fig. 16 shows the effect of varying the values of ϱ and φ. We use $\varrho = 3.0$ and $\varphi = 12.0$ in our experiment, which maintains a good balance of the terms and avoids the problem of surface depressions.

Fig. 17 shows the spherical representation of the Elf model, and Fig. 18 illustrates histograms of the parameterization. During simplification, for a complex model like the Elf which contains extremely dense vertices, we combine the QEM and curvature reduction criteria: we first bring the number of vertices from tens of thousands down to a few thousands with the OEM, as long as the number of vertices is still sufficient to represent silhouette detail, and then we use curvature reduction criteria to create a tetrahedron for the initial parameterization.

We use the subdivision method to create the supermesh. Comparing to the multi-resolution method by Lee et al. [29] which created a metamesh, our subdivision method is memory efficient. The number of triangles required by the metatmesh is 3.8 times larger than our supermesh.

Comparison. We compare our work with the approach proposed by Mocanu and Zaharia [15], which used spherical parameterization methods to align and combine input meshes. Different from our work, Mocanu and Zaharia employed a local flattening algorithm to parameterize vertices and align feature correspondences, by repeatedly reducing the stretch of Gaussian curvature. We measure the visual difference by subtracting the rendered image of the remeshed model

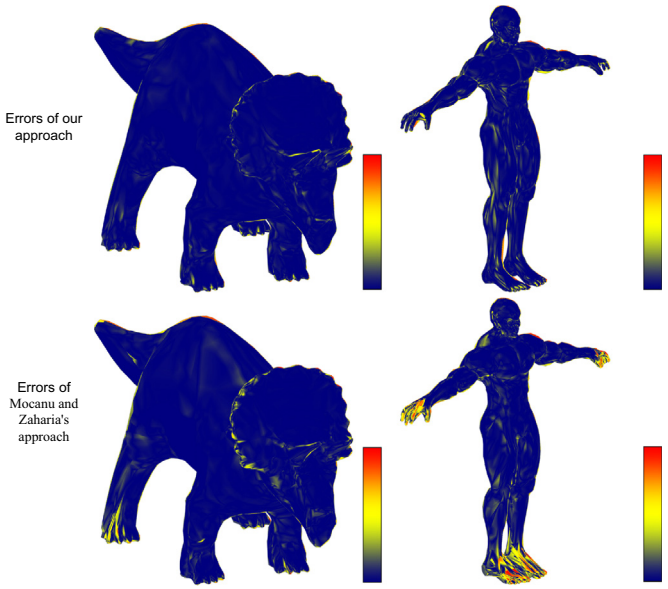


Fig. 19. *Visual quality comparison.* We use a color scheme to illustrate visual errors. Dark blue means the remeshed model appears closer to the original model; bright red means high visual errors. The result of our approach has less errors than Mocanu and Zaharia [15], especially on the surface regions such as fingers and toes. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

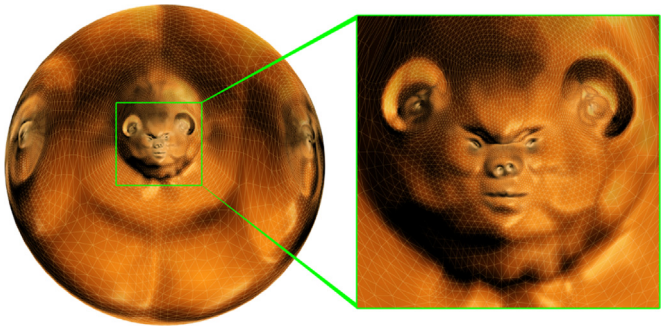


Fig. 17. *The spherical representation of Elf model with 34K vertices.* The zoomed-in window shows the parameterized facial details.

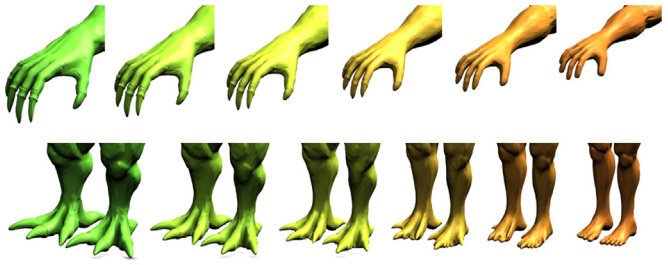


Fig. 20. *Minor feature preservation during morphing.* Minor features like the fingers with a high similarity are aligned correspondingly while morphing. Though the feet shown in the second row do not have similar shapes (the number of toes is different), they smoothly morph from one shape to the other.

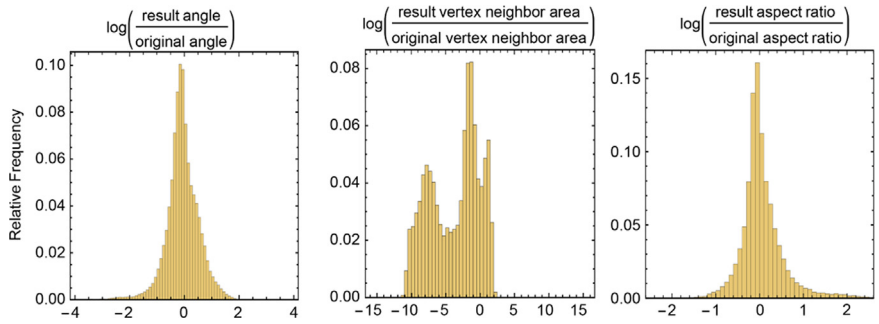


Fig. 18. Histograms of angle distortion, area distortion and aspect ratio for parameterization of the Elf model in Fig. 17.

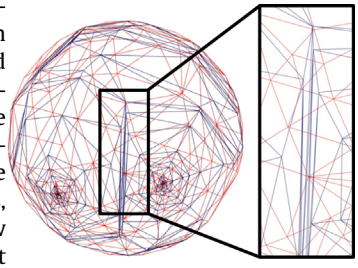
from the rendered image of the original model. Fig. 19 illustrates the visual differences, where both approaches are with the same rendering settings. Our approach better preserves visual appearance, especially on those sharp or highly deformed regions. The morphing results generated based on Mocanu and Zaharia's approach have noticeable artifacts on the regions of face, hand and foot due to high stretchiness of parameterized triangles. Stretched triangles in these regions cause triangle degeneration and foldover problems.

Minor feature preservation. In order to align minor features like fingers and toes, we run two passes of the exact alignment. We first add the feature points to align major features such as elbows, knees, shoulders, etc. We then add the feature points for minor features. An example of the alignments for fingers and toes is shown in Fig. 20. Our approach preserves the features that have low similarities. Fig. 20 shows the feet with the different number of toes smoothly blended without over-tessellation or sparseness.

10. Conclusion

We have presented an efficient approach for spherical parameterization, remeshing, and morphing of genus-zero meshes. We use a curvature reduction criteria for mesh simplification. The methods of the kernel sampling, global optimization and feature alignment allow us to parameterize a set of complex meshes in just a few minutes. The remeshing method generates a supermesh that correctly represents the appearance of original input meshes. Comparing to existing approaches, the approach presented in this paper advances in performance while maintaining high quality parameterization and morphing.

Limitations. Same as many existing spherical parameterization approaches (e.g., [4,3,15]), our approach has an inherent limitation that, for a highly deformed shape, we cannot simultaneously have both low stretched expanse and high conformality. Artifacts exist in our results when a long and thin triangle of one spherical representation overlaps multiple triangles of the other spherical representation. This usually happens when the geometric primitive densities of input meshes are significantly different (e.g., 40K triangles of Merman merges with 1K triangles of Dolphin). As shown in the inset on the right, long and thin triangles (blue) cross multiple triangles (red) after feature alignment. The remeshing algorithm cannot adequately sample across those blue triangles, because it does not add new vertices in the supermesh; it only creates a new topology based on the distribution layout of the vertices inherited from the input spherical parameterizations. Though the aspect ratio term used for parameterization greatly alleviate the problem, artifacts such as the lack of feature detail can be observed in some regions. *Future work.* There are a few things we would like to do in the future to strengthen our work. It would be beneficial to introduce varied radii to support different sizes of features (e.g., small radii support parameterization on fingers). It is worth to explore multi-phase alignment: aligning coarse features first, then align finer features. The kernel sampling process could be implemented in a parallel fashion, we would like to research on a GPU implementation for it. The proposed global optimization is essential but time-consuming. We would like to explore possible ways to remove the data dependency introduced between iterations of the global optimization. Spherical parameterization is suitable for genus-zero meshes; for high genus meshes, multi-layered spherical parameterization approaches were proposed [38,39]. We would like to find out how to extend our approach to the multi-layer spherical domain.



Acknowledgment

This work was partially supported by the NSF grant, CNS 1464323. We gratefully thank anonymous reviewers for their suggestions and comments. We would like to thank Timothy S. Newman for his valuable feedback and Vinny Argentina for helping us evaluate applicability of this work.

Appendix A. Supplementary data

Supplementary data associated with this paper can be found in the online version at http://dx.doi.org/10.1016/j.cag.2016.06.001.

References

[1] Kraevoy V, Sheffer A. Cross-parameterization and compatible remeshing of 3d models. in: ACM SIGGRAPH 2004 papers. SIGGRAPH'04. New York, NY, USA: ACM; 2004. p. 861–9. http://doi.acm.org/10.1145/1186562.1015811.

[2] Schreiner J, Asirvatham A, Praun E, Hoppe H. Inter-surface mapping. in: ACM SIGGRAPH 2004 papers. SIGGRAPH'04. New York, NY, USA: ACM; 2004. p. 870–7. http://doi.acm.org/10.1145/1186562.1015812.

[3] Praun E, Hoppe H. Spherical parametrization and remeshing. In: ACM SIGGRAPH 2003 Papers. SIGGRAPH'03. New York, NY, USA: ACM; 2003. p. 340–9. http://doi.acm.org/10.1145/1201775.882274 [ISBN 1-58113-709-5].

[4] Gotsman C, Gu X, Sheffer A. Fundamentals of spherical parameterization for 3d meshes. In: ACM SIGGRAPH 2003 papers. SIGGRAPH'03. New York, NY, USA: ACM. 2003. p. 358–63. http://doi.acm.org/10.1145/1201775.882276 [ISBN 1-58113-709-5].

[5] Mocanu B, Zaharia T. Direct spherical parameterization of 3d triangular meshes using local flattening operations. In: Proceedings of the 7th international conference on advances in visual computing – volume part I. ISVC'11. Berlin, Heidelberg: Springer-Verlag; 2011. p. 607–18. ⟨http://dl.acm.org/citation.cfm?id=2045110.2045177⟩ [ISBN 978-3-642-24027-0].

[6] Wang C, Liu Z, Liu L. As-rigid-as-possible spherical parametrization. Graph Models 2014;76(5):457–67. http://dx.doi.org/10.1016/j.gmod.2014.03.016.

[7] Sorkine O, Cohen-Or D, Goldenthal R, Lischinski D. Bounded-distortion piecewise mesh parameterization. In: Proceedings of the conference on visualization'02. VIS'02. Washington, DC, USA: IEEE Computer Society; 2002. p. 355-362. URL ⟨http://dl.acm.org/citation.cfm?id=602099.602154⟩ [ISBN 0-7803-7498-3].

[8] Aigerman N, Poranne R, Lipman Y. Lifted bijections for low distortion surface mappings. ACM Trans Graph 2014;33(4):69:1–12. http://dx.doi.org/10.1145/2601097.2601158.

[9] Aigerman N, Poranne R, Lipman Y. Seamless surface mappings. ACM Trans Graph 2015;34(4):72:1–13. http://dx.doi.org/10.1145/2766921.

[10] Zhou K, Synder J, Guo B, Shum HY. Iso-charts: stretch-driven mesh parameterization using spectral analysis. In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on geometry processing, SGP'04. New York, NY, USA: ACM; 2004. p. 45–54. http://doi.acm.org/10.1145/1057432.1057439 [ISBN 3-905673-13-4].

[11] Gu X, Gortler SJ, Hoppe H. Geometry images. In: Proceedings of the 29th annual conference on computer graphics and interactive techniques. SIGGRAPH'02. New York, NY, USA: ACM; 2002. p. 355–61. http://doi.acm.org/10.1145/566570.566589 [ISBN 1-58113-521-1].

[12] Quicken M, Brechbuhler C, Hug J, Blattmann H, Szekely G. Parameterization of closed surfaces for parametric surface description. In: Proceedings of the IEEE conference on computer vision and pattern recognition, vol. 1; 2000. p. 354–60. http://dx.doi.org./10.1109/CVPR.2000.855840.

[13] Wan S, Ye T, Li M, Zhang H, Li X. Efficient spherical parametrization using progressive optimization. In: Hu SM, Martin R, editors. Computational visual media. Lecture notes in computer science, vol. 7633. Berlin, Heidelberg: Springer; 2012. p. 170–7 [ISBN 978-3-642-34262-2].

[14] Zhang R. 3d mesh metamorphosis from spherical parameterization for conceptual design [Ph.D. dissertation]; 2011.

[15] Mocanu B, Zaharia T. A complete framework for 3d mesh morphing. In: Proceedings of the 11th ACM SIGGRAPH international conference on virtual-reality continuum and its applications in industry. VRCAI'12. New York, NY, USA: ACM; 2012. p. 161–70. http://doi.acm.org/10.1145/2407516.2407558 [ISBN 978-1-4503-1825-9].

[16] Kazhdan M, Solomon J, Ben-Chen M. Can mean-curvature flow be made non-singular? ArXiv e-prints arXiv:1203.6819; 2012.

[17] Friedel I, Schröder P, Desbrun M. Unconstrained spherical parameterization. J Graph GPU Game Tools 2007;12(1):17–26. http://dx.doi.org/10.1080/2151237X.2007.10129230.

[18] Floater M, Hormann M. Surface parameterization: a tutorial and survey. In: Dodgson N, Floater M, Sabin M, editors. Advances in multiresolution for geometric modelling. Mathematics and visualization. Berlin, Heidelberg: Springer; 2005. p. 157–86 [ISBN 978-3-540-21462-5].

[19] Hoppe H. Progressive meshes. In: Proceedings of the 23rd annual conference on computer graphics and interactive techniques. SIGGRAPH'96. New York, NY, USA: ACM; 1996. p. 99–108. http://doi.acm.org/10.1145/237170.237216 [ISBN 0-89791-746-4].

[20] Garland M, Heckbert PS. Surface simplification using quadric error metrics. In: Proceedings of the 24th annual conference on computer graphics and interactive techniques. SIGGRAPH'97. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.; 1997. p. 209–16. http://dx.doi.org/10.1145/258734.258849 [ISBN 0-89791-896-7].

[21] Schroeder WJ, Zarge JA, Lorensen WE. Decimation of triangle meshes. In: Proceedings of the 19th annual conference on computer graphics and interactive techniques. SIGGRAPH'92; New York, NY, USA: ACM; 1992. p. 65–70. http://doi.acm.org/10.1145/133994.134010 [ISBN 0-89791-479-1].

[22] Cohen J, Olano M, Manocha D. Appearance-preserving simplification. In: Proceedings of the 25th annual conference on computer graphics and interactive techniques. SIGGRAPH'98. New York, NY, USA: ACM; 1998. p. 115–22. http://doi.acm.org/10.1145/280814.280832 [ISBN 0-89791-999-8].

[23] Hormann K, Greiner G. MIPS: an efficient global parametrization method. Defense Technical Information Center; 2000.

[24] Gu X, Wang Y, Chan T, Thompson P, Yau ST. Genus zero surface conformal mapping and its application to brain surface mapping. IEEE Trans Med Imaging 2004;23(8):949–58.

[25] Nadeem S, Su Z, Zeng W, Kaufman A, Gu X. Spherical parameterization balancing angle and area distortions. IEEE Trans Vis Comput Graph 2016;99:1. http://dx.doi.org/10.1109/TVCG.2016.2542073.

[26] Schönemann PH. A generalized solution of the orthogonal procrustes problem. Psychometrika 1966;31(1):1–10. http://dx.doi.org/10.1007/BF02289451.

[27] Arun KS, Huang TS, Blostein SD. Least-squares fitting of two 3-d point sets. IEEE Trans Pattern Anal Mach Intell 1987;9(5):698–700. http://dx.doi.org/10.1109/TPAMI.1987.4767965.

[28] De Boer A, Van der Schoot MS, Bijl H. New method for mesh moving based on radial basis function interpolation. In: ECCOMAS CFD 2006: Proceedings of the European conference on computational fluid dynamics. Egmond aan Zee, The Netherlands, September 5–8, 2006. Delft University of Technology, European Community on Computational Methods in Applied Sciences (ECCOMAS); 2006.

[29] Lee AWF, Dobkin D, Sweldens W, Schröder P. Multiresolution mesh morphing. In: Proceedings of the 26th annual conference on computer graphics and interactive techniques. SIGGRAPH'99. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.; 1999. p. 343–50. http://dx.doi.org/10.1145/311535.311586 [ISBN 0-201-48560-5].

[30] Barber CB, Dobkin DP, Huhdanpaa H. The quickhull algorithm for convex hulls. ACM Trans Math Softw 1996;22(4):469–83. http://dx.doi.org/10.1145/235815.235821.

[31] Saba S, Yavneh I, Gotsman C, Sheffer A. Practical spherical embedding of manifold triangle meshes. In: 2005 international conference on shape modeling and applications; 2005. p. 256–65. http://dx.doi.org/10.1109/SMI.2005.32.

[32] Athanasiadis T, Fudos I, Nikou C, Stamati V. Feature-based 3d morphing based on geometrically constrained spherical parameterization. Comput Aided Geom Des 2012;29(1):2–17. http://dx.doi.org/10.1016/j.cagd.2011.09.004.

[33] Fu XM, Liu Y, Guo B. Computing locally injective mappings by advanced MIPS. ACM Trans Graph 2015;34(4):71:1–12. http://dx.doi.org/10.1145/2766938.

[34] Athanasiadis T, Fudos I. Parallel computation of spherical parameterizations for mesh analysis. Comput Graph 2011;35(3):569–79. http://dx.doi.org/10.1016/j.cag.2011.03.022 [Shape modeling international (SMI) conference 2011 URL: ⟨http://www.sciencedirect.com/science/article/pii/S0097849311000616⟩].

[35] Sander PV, Nehab D, Barczak J. Fast triangle reordering for vertex locality and reduced overdraw. In: ACM SIGGRAPH 2007 papers. SIGGRAPH'07; New York, NY, USA: ACM; 2007. http://doi.acm.org/10.1145/1275808.1276489.

[36] Lewis JP, Anjyo K, Rhee T, Zhang M, Pighin F, Deng Z. Practice and theory of blendshape facial models. In: Lefebvre S, Spagnuolo M, editors. Eurographics 2014 – state of the art reports. Goslar, Germany: The Eurographics Association; 2014, http://dx.doi.org/10.2312/egst.20141042.

[37] Lorach T. Gpu blend shapes. NVidia Whitepaper 2007:69.

[38] Lee TY, Yao CY, Chu HK, Tai MJ, Chen CC. Generating genus-n-to-m mesh morphing using spherical parameterization. Comput Anim Virtual Worlds 2006;17(4):433–43. http://dx.doi.org/10.1002/cav.146.

[39] Zeng W, Li X, Yau ST, Gu X. Conformal spherical parametrization for high genus surfaces. Commun Inf Syst 2007;7(3):273–86 [URL ⟨http://projecteuclid.org/euclid.cis/1201531975⟩].