Feature-based 3D Morphing based on Geometrically Constrained Spherical Parameterization

Theodoros Athanasiadis, Ioannis Fudos, Christophoros Nikou and Vasiliki Stamati

Department of Computer Science, University of Ioannina, GR45110 Ioannina, Greece

Abstract

Current trends in free form editing motivate the development of a novel editing paradigm for CAD models beyond traditional CAD editing of mechanical parts. To this end, we need robust and efficient 3D mesh deformation techniques such as 3D structural morphing.

In this paper, we present a feature-based approach to 3D morphing of arbitrary genus-0 polyhedral objects that is appropriate for CAD editing. The technique is based on a sphere parameterization process built on an optimization technique that uses a target function to maintain the correspondence between the initial polygons and the mapped ones, while preserving topology and connectivity through a system of geometric constraints. Finally, we introduce a fully automated feature-based technique that matches surface areas (feature regions) with similar topological characteristics between the two morphed objects and performs morphing according to this feature correspondence list. Alignment is obtained without user intervention based on pattern matching between the feature graphs of the two morphed objects.

Keywords: mesh parameterization, morphing, feature-based models, geometric constraints, animation

1. Introduction

Feature-based computer-aided design has enabled efficient and robust editing of complex CAD models through effectively capturing designer intent

Email address: {thathana,fudos,cnikou,vicky}@cs.uoi.gr (Theodoros Athanasiadis, Ioannis Fudos, Christophoros Nikou and Vasiliki Stamati)

[1]. There is an increasing trend to make the CAD design process accessible to users with no previous CAD/CAM software experience. To this end, researchers and manufacturing companies have proposed to mimic the way an artist shapes a sculpture: start from a volume or object that is close to the intended target and iteratively shape (morph) its parts to finally render what the artist had in mind.

Our ultimate goal is to offer a novel editing paradigm for CAD models that goes beyond traditional CAD editing of mechanical parts. Towards this goal, we present an accurate and robust feature-based morphing technique that can be applied between any pair of genus-0 objects.

Although there are quite versatile and accurate methods for 2D image morphing, the 3D case remains an open problem both in terms of feasibility and accuracy.

Existing methods for 3D morphing can be categorized into two broad classes: *volume-based* or *voxel-based* [2] and *mesh-based* or *structural* [3] approaches. In this paper, we follow a mesh-based approach. The volume-based approach represents a 3D object as a set of voxels usually leading to computationally intensive computations. The volume-based approach exhibits better results in terms of boundary smoothness and rendering, since the intermediate morphs are represented as volumes. Techniques such as marching cubes [4] are employed to acquire the final polygonal representation used for rendering. Furthermore, most applications in graphics use mesh-based representations, making mesh-based modeling more broadly applicable.

Although mesh morphing is more efficient as compared to volume-based morphing, it requires a considerable preprocessing of both the source and the target object. Mesh morphing involves two steps. The first step establishes a mapping between the source and the target object (correspondence problem), which requires that both models are meshed isomorphically with a one-to-one correspondence. The second step involves finding suitable paths for each vertex connecting the initial position to the final position in the merged mesh topology (interpolation problem). due to the increased space complexity of the representation.

In this paper, we introduce a sound and complete approach to morphing between any two genus-0 objects. Recall that genus-0 objects are by definition homeomorphic to the sphere. Our mapping works in two phases. In the first phase, we calculate an initial bijective mapping. In the second phase, we optimize the mapping to achieve a better placement under specific geometric criteria and topological constraints. We also present an improvement of this

approach that takes into consideration 3D features and derives a feature correspondence set to improve the final visual effect. This is a very important characteristic for similar objects, as in the case of morphing between two articulated human representations. Object alignment, feature detection and feature point matching is performed automatically without user intervention.

In a nutshell, this paper makes the following technical contributions:

- Presents a feature preserving spherical parameterization process based on geometrically constrained optimization.
- Introduces an algorithm that captures high level geometric structure by building a feature region adjacency graph.
- Describes a novel feature matching technique that automatically aligns two solid objects and identifies a set of feature correspondence points.
- Introduces a feature guided optimized parameterization that is used to achieve smooth visual results in morphing between objects with structural similarities.

The rest of this paper is structured as follows. Section 2 presents related work on 3D morphing. Section 3 presents the spherical parameterization step of our approach. Section 4 briefly describes the efficient computation of the intersections among the polygons on the sphere and the calculation of the interpolation trajectory. Section 5 presents an alternative mapping method that can be applied to one of the morphed objects based on the mapping of the other object and a feature point correspondence list of the two meshes. Section 6 presents an experimental evaluation of our method and some visual morphing results. Finally, Section 7 offers conclusions.

2. Related Work

Most surface-based mesh morphing techniques employ a merging strategy to obtain the correspondence between the vertices of the input model. The merging strategy may be either automatic or user specified. Kent et al. [3] proposes an algorithm for the morphing of two objects topologically equivalent to the sphere. The mapping presented is accomplished by a mere projection to the sphere and thus is applicable solely to star shaped objects.

Kanai et al. [5] use a spring system to model the mesh and gradually force it to expand or shrink on the unit sphere by applying a force field. Methods

using springs do not always produce acceptable mappings especially when handling complex non convex objects. We overcome this problem successfully in our approach.

In [6, 7], a spring-like relaxation process is used. The relaxation solution may collapse to a point, or experience foldovers, depending on the initial state. Several heuristics achieving convergence to a valid solution are used.

[8, 9, 10] describe methods to generate a provable bijective parameterization of a closed genus-0 mesh to the unit sphere. The projection involves the solution of a large system of non-linear equations. A set of constraints on the spherical angles is maintained to achieve a valid spherical triangulation. We have adapted some of these ideas in our work.

Schreiner et al. [11] present a method that directly creates and optimizes a continuous map between the meshes instead of using a simpler intermediate domain to compose parameterizations. Progressive refinement is used to robustly create and optimize the inter-surface map. The refinement minimizes a distortion metric on both meshes. Kraevoy and Sheffer [12] present a method that relies on mesh refinement to establish a mapping between the models. First a mapping between patches over base mesh domains is computed and then mesh refinement is used to find a bijective parameterization. An advantage of this approach is that it naturally supports feature correspondence, since feature vertices are required as user input for the initial patch mapping. However, it requires user supervision and interaction whereas our method is fully automated.

In [13], reeb-graphs and boolean operations are used to extend spherical parameterization for handling models of arbitrary genus. Existing methods for producing valid spherical embeddings of genus-0 models can be integrated into their framework. In that respect, this work is orthogonal to our approach. Another method that uses reeb-graphs for morphing topologically different objects of arbitrary genus is [14]. The method specifies the correspondence between the input models by using graph isomorphic theory. The super Reeb graph, which has the equivalent topological information to the Reeb graphs of the two input objects, is constructed and used to conduct the morphing sequence.This method is very interesting from a theoretical point of view, but in practice the resulting matching may be unintuitive. Our method obtains intuitive matching results for similar objects and produces visually smooth morphing sequences.

Finally, Lin and Lee [15] provide efficient techniques for morphing 3D polyhedral objects of genus-0. The emphasis of the method is on efficiency

and requires the definition of feature patches to perform 2D mapping and subsequent merging. Their method does not avoid self intersection and requires embedding merging and user intervention for mapping. Our method overcomes these shortcomings in the expense of considerable increase in preprocessing time for mapping.

The method presented in this paper overcomes the limitations of prior methods and allows for a totally automated and appropriate for morphing mapping of an object of genus-0 surface into a 2D space with spherical topology. An initial mapping over the unit sphere is computed and used as initial state and is then improved by employing nonlinear optimization. For smoother morphing that exploits the high level geometric structure we have introduced a feature-based approach. Feature correspondence is performed automatically without any user intervention.

3. Topology Preserving Spherical Parameterization

3.1. Preliminaries

A *planar triangulation* is a simple triangulated plane graph whose edges are represented by straight lines. The triangulation is called *valid* when the only intersections between its edges are at the common endpoints. It has been shown by Fary [16] that every planar graph G has a valid straight line representation. Therefore, for any planar graph G there exists a set of points p such that the induced triangulated graph $T(G, p)$ is valid. A way to construct such a valid triangulated graph is described in [17]. The boundary vertices of G are mapped to a convex polygon with the same number of vertices and in the same order. Then, the interior vertices are placed such that each vertex is the centroid of its neighboring vertices. This was extended by Floater [18] who has proven that each vertex v_i=(x_i,y_i) can be any convex combination of its N_i neighboring vertices (1).

Consequently, for finding a one-to-one bijective mapping for a mesh with an open boundary B to a convex parametric domain $P \in R^2$ (e.g. a unit disk), a sufficient condition is to find a set of positive weights that satisfy (1)

and solve the corresponding linear system for those weights.

$$\begin{aligned} v_i &= \sum_{v_j \in N_i} w_{ij} v_j \\ \sum_{v_j \in N_i} w_{ij} &= 1 \\ w_{ij} &> 0 \end{aligned} \tag{1}$$

Theorem 1. *The linear system (1), which expresses the position of each node as a convex combination of its neighbors, has a unique solution if at least one of its nodes is fixed.*

Proof. See Appendix A. □

Thus, the resulting system has always a unique solution provided that the boundary vertices are fixed. A straightforward choice is to choose equal weights resulting in each vertex representing the centroid of its neighbors. This is also called barycentric mapping. For a mesh $M(V, E)$, barycentric mapping minimizes the sum of the squares of edges lengths, with respect to a fixed boundary. This is due to:

$$\begin{aligned} f(v_1, v_2, ..., v_n) &= \sum_{(v_i, v_j) \in E} ||v_i - v_j||^2 \\ ||v_i - v_j||^2 &= (x_i - x_j)^2 + (y_i - y_j)^2 \end{aligned} \tag{2}$$

Since f is convex, it has a global minimum when $\partial f / \partial x_i = \partial f / \partial y_i = 0$ for $i = 1, ..., n$:

$$\frac{\partial f}{\partial x_i} = 2 \sum_{v_j \in N_i} (x_i - x_j), \quad \frac{\partial f}{\partial y_i} = 2 \sum_{v_j \in N_i} (y_i - y_j) \tag{3}$$

This is equivalent to solving the linear system (1) with equal weights:

$$\begin{aligned} \sum_{v_j \in N_i} w_i x_j &= x_i \\ \sum_{v_j \in N_i} w_i y_j &= y_i \\ w_i &= \frac{1}{|N_i|} \end{aligned} \tag{4}$$

The theory for planar parameterizations can be directly extended to a spherical domain by reducing the problem to the planar case. The parameterization is then computed in polar coordinates. One approach to alleviate the problem is to select two vertices as the poles (north and south) of the parameterization. Subsequently, a geodesic path must be established between the poles over the mesh surface. The path connecting the two poles defines the boundaries of the parameterization and thus the spherical surface can be converted to a unit disk. Therefore, we can directly use the previous analysis to compute one-to-one bijective mapping. If equal weights are chosen and the poles are selected based on the largest distance along the z direction in object space, the resulting system is the linear system proposed by Brechbuhler et al. [19]. This way a valid spherical parameterization can always be produced for every mesh. The quality of parameterization depends on the choices for the poles and the connecting path. It turns out that selecting a good path is a difficult problem that affects the distortion in the final parameterization.

Another approach is to cut out a triangle from the mesh, leaving an open boundary, and make the mesh homeomorphic to the unit disk. This approach, also referred to in literature as stereo mapping, usually results in very distorted parameterizations since using the corresponding unit triangle as a boundary tends to cluster the remaining vertices in the center of the triangle.

As explained, the main drawback of the previously described techniques is the unnecessary distortion introduced by the parameterization. Unfortunately, generalizing the barycentric coordinates and the planar parameterization theory to a spherical domain is not straightforward. Since the domain is non-planar, expressing a vertex on the sphere as a convex combination of its neighbors is not feasible in general. This would imply for example that if the neighbors of a vertex are co-planar, then the vertex should also lie on the same plane. Nevertheless, it turns out that the following holds:

Theorem 2. *If each vertex position is expressed as some convex combination of the positions of its neighbors projected on the sphere (5), then the formed spherical triangulation is valid.*

$$
\begin{aligned}
v_i &= \frac{\sum_{v_j \in N_i} w_{ij} v_j}{|| \sum_{v_j \in N_i} w_{ij} v_j ||} \\
\sum_{v_j \in N_i} w_{ij} &= 1 \\
w_{ij} &= w_{ji} \\
w_{ij} &> 0
\end{aligned}
\tag{5}
$$

Proof. This is an immediate result of Theorem 2 in Gotsman et al. [20]. □

Problem (5) can be expressed as a set of $3n-3$ non-linear equations for the nodes $i = 1, ..., n-1$ of the mesh, where $n = |V|$ is the number or vertices. The equation for the last vertex is redundant in the case of a connected triangular mesh. We should also introduce n equations that constrain the vertices to lie on the unit sphere. We then seek the positions of the vertices $v_i(x_i, y_i, z_i)$ and the n auxiliary variables a_i:

$$
\begin{aligned}
a_i x_i - \sum_{v_j \in N_i} w_{ij} x_j &= 0, i = 1, \ldots, n-1 \\
a_i y_i - \sum_{v_j \in N_i} w_{ij} y_j &= 0, i = 1, \ldots, n-1 \\
a_i z_i - \sum_{v_j \in N_i} w_{ij} z_j &= 0, i = 1, \ldots, n-1 \\
x_i^2 + y_i^2 + z_i^2 &= 1, i = 1, \ldots, n
\end{aligned}
\tag{6}
$$

In general, a solution to this system is not unique. Without removing some degrees of freedom there are infinite solutions due to the possible rotations over the sphere (three degrees of freedom). More importantly, there are degenerate solutions that satisfy (6). The most obvious one is observed when $a_i = 0$ where all vertices of the parameterization collapse to one point on the sphere. Another possible degenerate solution can occur when the mesh contains a Hamiltonian cycle and the vertices are mapped to the equator of the sphere. Other degenerate solutions also exist, see e.g. [20]. We can eliminate the infinite solutions if we fix three degrees of freedom (for example vertex v_n and an angle that will determine a unit circle on which a second vertex lies).

However, even if we manage to avoid degenerate solutions we may still have a finite but exponentially large number of solutions (see e.g. [21, 22]) that we may have to eliminate by overconstraining or by introducing inequalities.

Even a robust and stable non-linear solver may converge to degenerate solutions for the system of equations (6). A key observation is that, as the solver iterations proceed, some triangles start growing and eventually pass through the equator of the sphere. The fundamental problem is that the spherical energy minimum occurs at a collapsed configuration. This situation occurs because the continuous spherical energy is approximated by a quadratic energy function calculated over the mesh triangles. Therefore, since the area of a planar triangle is always smaller than the area of the corresponding spherical triangle, an estimation error is introduced in the calculation of the energy over the surface. This error increases disproportionately with the size of the triangles. Therefore, the non-linear minimizer may minimize the corresponding distortion metric (energy function) over the sphere surface by increasing the size of the triangles with the largest error.

One common fix to avoid these degenerate solutions is to fix three or more vertices, thus constraining the solver. In practice however there are two problems, the extra constraints introduce additional distortion in the parameterization, and it is difficult in general to determine a proper set of fixed vertices. In addition, without paying special attention to the set of the constrained vertices, the non-linear problem may become infeasible.

Another important issue with the system of equations in (6) is that there is no guarantee that the solution is bijective in the case of non-symmetric or negative weights. Therefore, adapting the weights for morphing is difficult because weights for conformal mappings can be negative and weights for authalic mappings are non-symmetric [23].

Summarizing the above observations, if we try directly to solve (6), the following problems occur:

- *Non convexity.* The constraints $x_i^2 + y_i^2 + z_i^2 = 1$ are not convex. Therefore classical convex minimization cannot be used directly.

- *Non regularity.* Due to non-convex constraints, uniqueness and higher regularity of solutions cannot be expected [24].

- *Non uniqueness.* The energy does not have a unique minimum and degenerate solutions always exist.

An approach to tackle these difficulties was proposed in [10]. Here a penalty term d_{min}^{-2}, where d_{min} is the minimum distance of each triangle from the sphere center, was added in the corresponding planar quadratic energy and the constraints were removed. The motivation of this approach is to provide an upper bound of the spherical energy by scaling the corresponding planar energies of the triangles. Therefore, the corresponding problem (6) is transformed to an unconstrained one, that can be solved with standard methods. However, this approach is possible to restrict the minimize process and the convergence properties are unclear since no theoretical guarantee is provided.

In the method presented in this paper, we overcome the above difficulties by employing a two step approach. An initial bijective parameterization over the unit sphere is computed and is used as an initial guess for a nonlinear optimization process. The optimized parameterization is guaranteed to be bijective by enforcing a proper set of constraints.

3.2. Initial Spherical Parameterization

A possible technique that one can use to obtain an initial parameterization is an iterative process that attempts to converge to a valid parameterization by applying local improvement (relaxation) [6]. The principle for this improvement is to reduce the spring energy of the points with Laplacian smoothing ignoring the sphere constraint and renormalise the solution to obtain valid spherical points. In practice however, the iterative process may converge to a degenerate solution and will then require a restart. Since Laplacian smoothing does not perform any triangle area balancing, certain elements may collapse leading to a degenerate solution. The following key observation motivates our approach,

Observation 1. Iterative projected Laplacian smoothing collapses after one or more elements overgrow.

The above observation motivates an area balancing procedure where the new position of the vertices is determined based on an area weighted sum. More specifically, the new position of each vertex is determined by the weighted sum of the centroids of the surrounding triangles, where the weights are determined by the area of each neighboring triangle. This approach yields a smoother mesh with more balanced element area since larger polygons tend to attract vertices, while smaller polygons tend to repulse them. Since the

set of weights is positive, each individual folding is not stable and is forced to unfold according to the area weighted centroid attraction rule.

The above procedure is expressed concisely by the following steps:

1. Let $(v_1, v_2, ..., v_n)^0$ be an initial guess for the solution
2. For $j = 0$; until no folded elements exist; j++
   (a) Set $v_i^{j+1} = \sum_{v_k \in N_i} a_k c_k$ for $i = 1, ..., n$
   (b) Set $v_i^{j+1} = \frac{v_i^{j+1}}{||v_i^{j+1}||}$ for $i = 1, ..., n$

where a_k is the area of the corresponding k-th triangle that is adjacent to vertex v_i and c_k is the centroid of this triangle. The initial solution is obtained by normalizing the original vertex coordinates (assuming the object is centered at the origin),

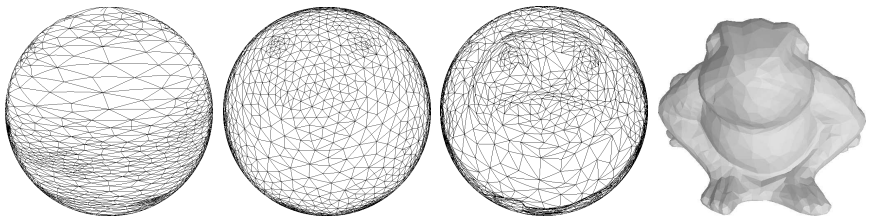
$$v_i^0 = \frac{v_i}{||v_i||} \text{for } i = 1, ..., n \tag{7}$$

The normalizing denominator maintains vertices on the unit sphere.

We have used two alternative methods for obtaining an initial mapping: barycentric mapping and area weighted Laplacian smoothing. In barycentic mapping, two polar coordinates are determined for all vertices in two steps. Two vertices are selected as the poles (north and south) for this process. The poles must not be too close as this will result in a poor initial parameterization. Therefore, we have implemented this initial mapping by selecting as poles the vertex pair with the largest distance between them (diameter of the solid). In Laplacian smoothing, we use the area weighted variation. Figure 1 shows the results of the initial mapping when applying the planar barycentric mapping method and Laplacian smoothing on the frog from [25]. In general, Laplacian smoothing is faster and provides a robust unfolded initial mapping while preserving similarities with the initial mesh.

Objective Function: We use as the objective function f to be minimized the sum of all dot products of every mapped vertex v_i^s with their corresponding initial position v_i^0 on the mesh.

$$f(v_1, v_2, ..., v_n) = \sum_{v_i \in V} v_i^0 \cdot v_i^s \tag{8}$$



(a) (b) (c) (d)

Figure 1: (a) The result of the initial mapping using the planar barycentric method, (b) the Laplacian smoothing technique, (c) the result after optimization and (d) the original frog model.

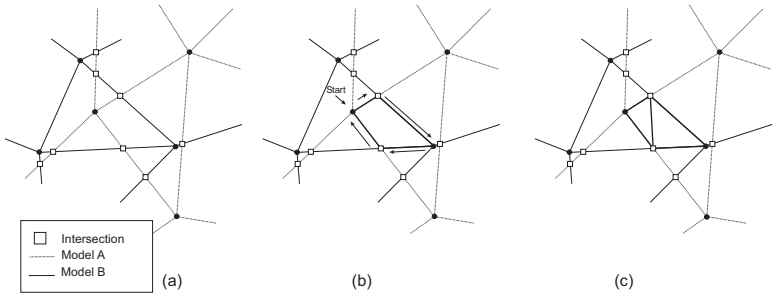


Figure 2: (a) Finding intersections in merged topology, (b) curve faces visited in clockwise manner and (c) triangulation

3.3. Optimized Parameterization for Morphing

For the optimized mapping we use the following objective function and set of constraints that are appropriate for morphing:
Geometric Constraints: For each vertex v_i we use the spherical constraints from (6) to keep the vertices on the unit sphere surface.

$$x_i^2 + y_i^2 + z_i^2 = 1 \tag{9}$$

Topological Constraints: For each face f_i of the mesh with vertices v_{i0}, v_{i1}, v_{i2} and for each vertex of this face, each vertex should stay on the same side of the plane defined by the other two vertices and the center of the sphere:

$$\begin{aligned} (v_{i1} \times v_{i2}) \cdot v_{i0} > 0 \\ (v_{i2} \times v_{i0}) \cdot v_{i1} > 0 \\ (v_{i0} \times v_{i1}) \cdot v_{i2} > 0 \end{aligned} \tag{10}$$

The goal of our optimization approach is to find a valid spherical parameterization suitable for morphing purposes. Since the minimization of the above objective function does not guarantee the validity of the final spherical parameterization, we must enforce this requirement with additional topological constraints (10). These constraints guarantee that the final parameterization is valid provided that we initiate the solver with a valid solution. In addition, by using a suitable tolerance $\epsilon > 0$ instead of 0, these constraints offer control over the area of each face in the final parameterization and thus degenerated elements are avoided.

The choice of the objective function is motivated by the observation that a mapping suitable for morphing should introduce more distortion in the concave areas. Therefore, the structurally important vertices of the mesh (for example those over the convex hull) are kept in their original projected positions on the sphere, whereas the distortion is concentrated on the less significant concave areas. Figure 1 illustrates the optimized mapping for the frog, while Figure 14(a) illustrates the final optimized mapping on the sphere for the Blender Suzanne model [26] and the head model [25]. The preservation of the initial characteristics is apparent.

To solve the problem (8) under the constraints (9) and (10) for large meshes it is important to have a stable and efficient numerical procedure. We use the Ipopt software [27], an implementation of the primal-dual interior point approach for nonlinear programming. This approach provides an efficient method for handling problems with large numbers of inequality constraints. Furthermore, interior-point methods allow convergence from poor starting points that may appear when computing an initial solution from methods such as the barycentric mapping (see Figure 1(a)). Under mild assumptions, it can be shown that such a procedure converges to a local solution of the original problem [28]. Moreover, since this is a convex optimization problem if a local minimun exists, then it is a global minimum. For all the examples the optimization process successfully terminated, satisfying the convergence tolerance error we have used (10^{-6}). As far as efficiency is concerned the experimental evaluation of Section 6 indicates an $O(|V|^2)$ behavior for the NL optimization step.

4. Surface Correspondence and Interpolation

Following the successful mapping of two meshes M_A and M_B on the sphere, a merging process of the two topologies is performed. The purpose of

this step is to create a final merged topology that is suitable for navigating back and forth to the original models.

This process requires each projected edge of one model to be intersected with each projected edge of the other. The algorithm to compute this step efficiently is based on the observation that starting from an intersection over an edge we can traverse all the remaining intersections by exploiting the topological information contained in the models. The complexity of this step is $O(E_A + K)$ where K is the total number of intersections.

From the intersections found, along with the vertices of the two models, a set of spherical regions bounded by circular arcs is determined. These regions are always convex, therefore it is straightforward to triangulate them. First for each edge, the list of intersections that belong to that edge is sorted by the distance from each vertex of the edge. Additionally, for each vertex, a list of the edges incident to it in clockwise order is calculated. Based on the aforementioned geometrical data we traverse each closed bounded region in a clockwise order and compute the triangulated merged topology in $O(K \log K)$ time complexity. Figure 2 illustrates this process.

The final step of the algorithm involves the projection of the merged topology back to the original models. For each model A the vertices of model B along with the intersection points are mapped back to A.

Following the successful establishment of a correspondence between the source and target vertices, the vertex positions are interpolated to acquire the final morphing sequence. To this end, we use simple linear interpolation. The advantage of linear interpolation, besides its simplicity, is that it can be efficiently realized on GPUs using a simple morphing shader for interpolating vertices and attributes (lighting, textures) in real-time. Nevertheless, linear interpolation may not always be desirable, especially in very complex meshes where self-penetrations may appear during the morphing sequence of the models. More advanced interpolation techniques are applied in such cases. Some of them are also implemented in shaders but their performance may vary depending on the limits set by the GPU.

5. Feature-based Morphing

The overall process of feature based morphing is presented in Figure 3. First, the optimized spherical parameterizations are computed for both models (this step can be carried out as preprocessing and the mapping can be

stored along with the mesh representation). Then feature regions are detected on both models using region growing and matched between the two models. Subsequently, feature point pairs are extracted and an optimized spherical parameterization is computed for the second model with respect to the feature point pairs. Finally, the actual morphing is carried out on the GPU.

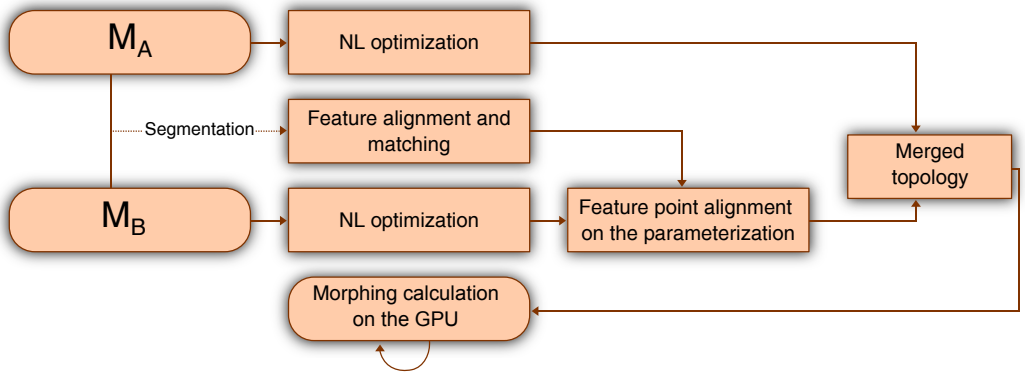


Figure 3: Overview of the feature-based algorithm.

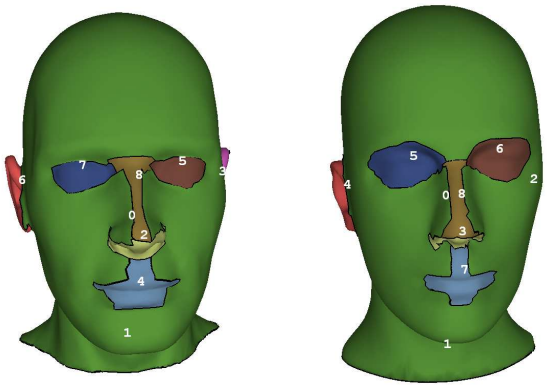
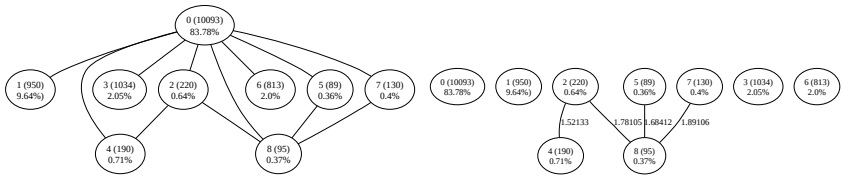


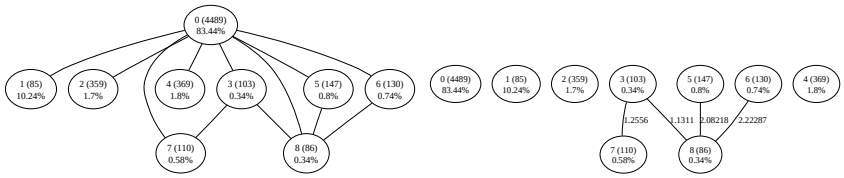
Figure 4: Detecting feature regions in two head meshes: (left) mesh M_A and (right) mesh M_B. Numbers correspond to identifiers for feature regions.

To detect feature regions on meshes, we built on a method developed earlier in [29] for reverse engineering based on discovering features on the point cloud by detecting local changes in the structure of the point cloud. This method works even better on meshes, since in meshes vertex adjacency information is provided a priori.



(a) Original adjacency graph of M_A (b) Reduced graph of M_A, all edges with large geodesic distances are eliminated.

(see Figure 4), the number of nodes and the area covered in the original model.



(c) Original graph of M_B (d) Reduced graph of M_B

Figure 5: Graph reduction of the head meshes.

We use region growing, detection of rapid variations of the surface normal and the concavity intensity and saddle points of the concavity intensity (the concavity intensity is the distance from the convex hull). This results in a number of regions that represent object feature regions (Figure 4). In the context of this paper, we employ this method to detect features in models for the purposes of matching and alignment of the two meshes that are to be morphed.

More specifically several softer features on the meshes are detected using a characteristic called *concavity intensity* of a point which represents the smallest distance of a point from its convex hull.

Definition 1. The *concavity intensity* of a vertex v_i of a mesh denoted by $I(v_i)$ is the distance of v_i from the convex hull of the mesh.

This characteristic is used to detect soft convex or concave features on the mesh. The surface normal and the concavity intensity are used in conjunction with a region growing method that results in detecting sets of faces, called *feature regions* or simply *features* that correspond to areas with distinctive characteristics.(Figures 4 and 14(b)). Finally, we merge adjacent feature regions with similarity criteria, starting from features of small area. For a

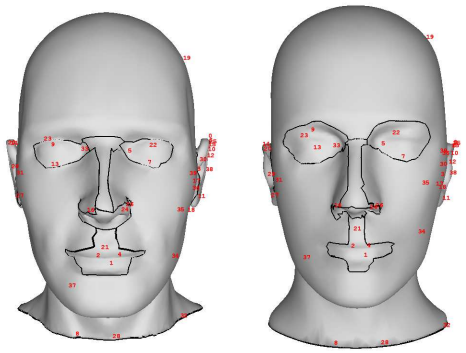


Figure 6: Detecting feature points inside feature regions.

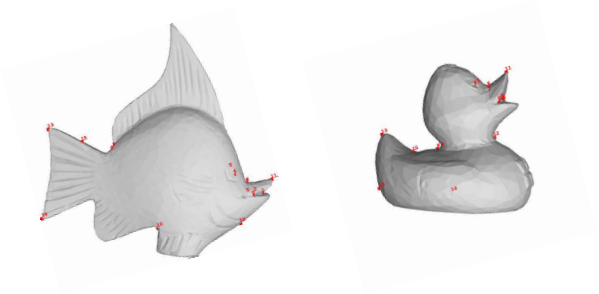


Figure 7: Feature point matching for the fish and the duck model.

mesh with a set of vertices V, this process takes time $O(|V|)$ due to the planarity of the original graph and the almost linear behavior of the smaller-regions-merge-first practice.

After obtaining the features of the object, we create a connectivity graph that captures adjacency information as illustrated in Figure 5. For each edge, we calculate the geodesic distances between the centroids of the corresponding feature regions. The graphs are then simplified by reducing the edges that correspond to large geodesic distances to facilitate region matching (Figure 5). In addition, small regions that can introduce noise and are insignificant are merged to form larger regions and when this is not feasible they are eliminated. Elimination of large distances is motivated by the observation that usually meshes do not exhibit a general structural similarity but rather a local feature one. This process takes at most time $O(|R|^2)$ where R is the original set of feature regions.

In simple cases (Figure 4), where meshes have an almost identical structure, matching of the corresponding graphs is trivial. For more complex cases (see Figure 14), meshes possess only local structural feature similarities. Therefore, by eliminating the edges with large geodesic distances we match only local neighborhoods in the graph. These local neighborhoods still capture higher level information about the structure of the features, for example they detect eyes, nose and mouth similarities between completely different character models. Alternatively, for larger graphs subgraph matching algorithms were tested for detecting similar subgraphs in the two reduced adjacency graphs using randomized algorithms (see e.g. [30]). It remains to be determined whether such techniques are meritorious in terms of efficiency and scope. Finally, if one of the two graphs is of small size and it can be considered as a fixed pattern then a deterministic subgraph isomorphism algorithm may be used [31] which derives an $O(|R|)$ algorithm.

The reduced adjacency graphs are used to perform a 3D alignment of the two models and establish a correspondence between the regions. This is achieved by first matching the two highest degree nodes in the two graphs and then performing a 3D alignment of the two models. The remaining regions are paired according to their degree and the distance between them. Furthermore, we also take into consideration the area covered by each region by favoring the matching of regions covering similar areas. We have used the

following heuristic similarity measure for matching,

$$s_{ij} = ||c_i - c_j|| \frac{\max\{a_i, a_j\}}{\min\{a_i, a_j\}} \frac{\max\{d_i, d_j\}}{\min\{d_i, d_j\}} \tag{11}$$

where c_i and c_j are the centroids of regions i and j, a_i and a_j are the corresponding areas and d_i and d_j are the degrees of the nodes in the reduced region adjacency graphs. This entire step for feature region alignment and matching takes time $O(|R_1||R_2|)$, where R_1, R_2 are the sets of nodes of the reduced graphs.

Moreover, for each feature region we detect points with certain properties that capture specific structural characteristics of the meshes. The resulting point set, called a *feature point set*, provides a summary of concave and convex regions of the object.

Definition 2. A vertex v_i is called a *feature point*, if and only if, $I(v_i)$ exhibits a local extremum at v_i.

Following the establishment of a correspondence between the region patches of the two models, the feature points of the corresponding patches are associated according to their distance. Since the patches may be in different locations in each model, the two regions are translated so that their corresponding centroids coincide. This step has worst case time complexity $O(|V| + |FP|^2)$, where FP is the set of feature point pairs. Figures 6, 7, and 14(c) illustrate the final feature point matching for different models.

So the overall time complexity for the feature matching is $O(|V| + |R|^2 + |FP|^2)$ and the space complexity is $O(|V| + |R| + |FP|)$.

For the feature based mapping of the second model we use the following objective function and set of constraints to obtain a more appropriate mapping based on the feature point correspondence of the models:
Objective Function: We use as the objective function to be minimized the sum of all dot products of every pair $p_i = (v_{i1}, v_{i2})$ of feature vertices $v_{i1} \in M_A$, $v_{i2} \in M_B$. Let FP be the set of pairs of feature vertices p_i.

$$\sum_{p_i \in FP} v_{i1} \cdot v_{i2} \tag{12}$$

Geometric Constraints: For each vertex v_i we use constraint (9).

Input: Two triangular meshes M_A and M_B
for *each vertex* v_i *of* M_A **do**
    calculate $I(v_i)$
end
for *each vertex* v_j *of* M_B **do**
    calculate $I(v_j)$
end
for M_A *and* M_B **do**
    compute the corresponding feature region sets F_A and F_B
end
for F_A *and* F_B **do**
    compute the corresponding connectivity graphs and perform graph reduction on them
end
Establish a correspondence of the two nodes with the highest degree in the two graphs and perform a 3D alignment of F_1 and F_2 up to rotation based on that correspondence;
for *each feature region in* F_B **do**
    find a feature region in F_A using the similarity measure (11) and match the corresponding feature point sets
end
Calculate the spherical parameterization for M_A and M_B;
Optimize the spherical parameterization of M_B in order to match the paired feature points of the parameterizations;

Figure 8: The algorithm for feature based morphing.

Topological Constraints: In addition to equation (9), the length of each edge e_i (circular arc over the sphere) that connects the vertices v_{i1},v_{i2} must remain the same during optimization:

$$v_{i1} \cdot v_{i2} = v_{i1}^s \cdot v_{i2}^s \tag{13}$$

recall that v_i^s is the position of vertex v_i after the sphere optimization process. By doing so, we preserve the topology of the second object during the optimization process. In addition, this avoids very long stretches of triangles to satisfy a certain feature point pair matching. The same algorithm as in Section 3.3 is used for the optimization. This final nonlinear optimization step is applied on the optimized spherical mapping of the second model. This is the

only optimization step that cannot be performed as preprocessing and thus cannot be stored along with the model representation. Fortunately, however the convergence speed of this optimization step depends on the number of feature points and the number of faces of the second model, as it is indicated by the experiments in Section 6.

The method described above offers a way of automatically detecting a set of feature point pairs in structurally similar meshes to guide the morphing sequence. Generally, it is a difficult task to automatically find common features in every pair of shapes. The fundamental issue is that common features are usually defined through a semantic description and not through a geometric one. In addition, meshes may exhibit a global structural similarity instead of a local feature similarity.

Our underlying assumption for the graph matching step is that the relative placement of the features we want to detect, bears a resemblance between the two models as far as local features are concerned. Thus, our approach is sensitive in the level of detail that is used for the segmentation step. Although we try to exploit the high level representations of the objects in conjunction with low level geometric metrics (area and distances), in cases of global similarity we must tune the segmentation parameters to limit the number of the regions detected. Moreover, we assume that the segmentation step results in a set of convex or almost convex regions and therefore the distances of the centroids are a reasonable approximation. Another problem may occur when there exists inherent symmetry in each mesh and as a result of this symmetry the matching process yields erroneous results.

To tackle these deficiencies a more general subgraph matching algorithm was examined, such as the randomized subgraph search algorithm [30], on a more detailed region graph. Nevertheless, our extensive experiments indicate that usually a rough matching of the feature regions gives satisfactory results. Therefore, whether such an extension would be beneficial in terms of accuracy and performance it remains to be determined. To summarize our current approach to feature-based morphing has the following limitations :

- Sensitivity to the segmentation step, since the high level representation may affect the matching of similar features.
- Intra object symmetry may yield non optimal feature point pairs.
- Semantically different features with similar connectivity and geometry can be matched.

- Very soft features that are not clearly identified by the segmentation algorithm may be left out.

The algorithm for feature-based morphing is presented in Figure 8. Figures 15 and 16 illustrate the visual improvement offered by this method.

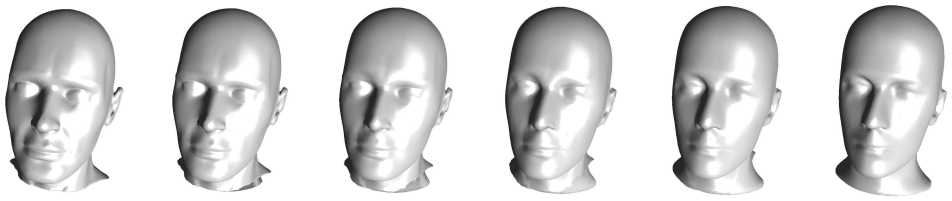


Figure 9: Morphing with alignment and feature point matching. Morphing is visually smooth through the entire sequence.

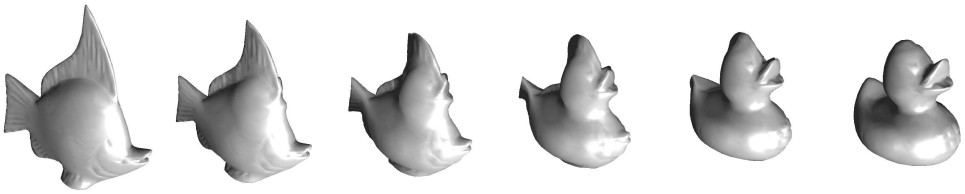


Figure 10: Morphing with alignment but no feature point matching: fish (4994 faces) to duck (1926 faces), merged topology has 28526 faces.

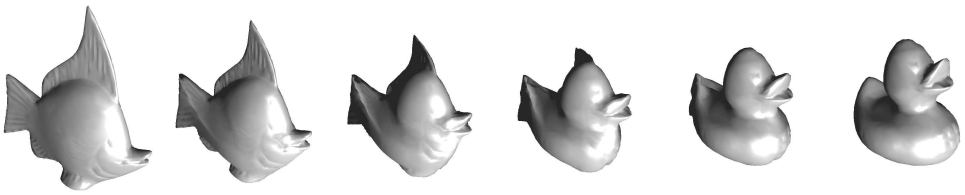


Figure 11: Morphing with alignment and feature point matching: fish (4994 faces) to duck (1926 faces), merged topology has 33038 faces.

6. Experiments and Performance Evaluation

We have developed software for implementing mapping, merging and interpolation as described in the previous sections. The software is available

Figure 12: Morphing with alignment but no feature point matching of the Charioteer model (11098 faces) to a Cycladic idol model (16798 faces), merged topology has 142422 faces.

Figure 13: Morphing with alignment and feature point matching, merged topology has 142512 faces

at `http://www.cs.uoi.gr/~fudos/cagd2011.html`. The platform used for development consists of a Windows XP Professional based system running on a Intel Pentium Q6600 Core 2 at 2.4GHz, 2GByte of RAM, with NVIDIA GeForce 8600GT. We have developed the system on Visual Studio 2005, using OpenGL 2.0 (Shader Model 3.0) and GLUT.

Table 1 summarizes the results of some of our experiments on mapping for different models using both the barycentric method and the Laplacian smoothing initialization. $|V|$ is the number of vertices of the mesh, $|F|$ is the number of faces, and $|C|$ is the number of constraints for the optimization procedure. The number of iterations refers to the optimization phase, while time refers to the total time for both deriving the initial mapping and for performing optimization.

We observe that the Laplacian smoothing initialization yields a much faster convergence in the optimization phase (half the number of iterations and 50% faster). Our extensive experiments indicate that the number of iterations increases quadratically over the number of vertices of the polyhedral representation for triangular models. This is a considerable overhead but it can be calculated offline during a preprocessing phase and stored along with

the polyhedral representation. Table 2 shows the results for the same set of experiments for the same model with different LODs ranging from 854 faces up to 5610 for the Suzanne model. This set of experiments confirms the above observations.

Finally, Table 3 shows the results of the feature guided optimization step. This step is significantly faster compared to the original parameterization and it depends on the number of feature point $|FP|$ and the number of faces of the second model. Another factor that affects the speed of convergence is the similarity of the two models. For example in Table 3 we observe a rapid convergence, in terms of iterations, for the case of the two similar head meshes (Figure 6).

Table 1: Experimental results of mapping with different models of various level of detail

model	method	$\|V\|$	$\|F\|$	$\|C\|$	iterations	time (secs)
Suzanne	Laplace	429	854	2991	36	10.9
Suzanne	Barycentric	429	854	2991	78	22.6
Bunny(Lod1)	Laplace	440	876	3068	94	24.3
Bunny(Lod1)	Barycentric	440	876	3068	165	52.0
Frog(Lod1)	Laplace	1964	3924	13736	70	422.2
Frog(Lod1)	Barycentric	1964	3924	13736	152	895.8

Table 2: Experimental results with the same model with different levels of detail

model	method	$\|V\|$	$\|F\|$	$\|C\|$	iterations	time (secs)
Suzanne(Lod1)	Laplace	429	854	2991	36	10.9
Suzanne(Lod1)	Barycentric	429	854	2991	78	22.6
Suzanne(Lod2)	Laplace	703	1402	4909	26	21.3
Suzanne(Lod2)	Barycentric	703	1402	4909	70	54.8
Suzanne(Lod3)	Laplace	1404	2804	9816	49	151.3
Suzanne(Lod3)	Barycentric	1404	2804	9816	91	271.3
Suzanne(Lod4)	Laplace	2807	5610	19637	79	934.0
Suzanne(Lod4)	Barycentric	2807	5610	19637	136	1578.6

As mentioned in Section 4, merging takes in average $O(K \log K)$ time, where K is the number of intersections. For all cases in Tables 1 and 2, this step took less than 2.5 sec. Finally, the interpolation step is implemented

Table 3: Feature alignment optimization

model1	model2	$\|FP\|$	$\|F\|$	$\|C\|$	iterations	time (secs)
Fish	Duck	23	1926	9632	123	14.3
Head1	Head2	26	11040	55202	34	39.9
Head2	Suzanne	36	5610	25245	171	78.7
Head2	Caesar	60	13530	67652	182	321.1

in GPU so it is very fast and can accommodate almost unlimited number of frames.

Figures 9, 10 and 11 illustrate 3 different cases of morphing whereas Figure 14 shows the different steps to obtain the final morphing sequence. We have performed the experiments on well-known models such as the Stanford bunny [32], the Blender Suzanne [26] and the Aim@shape frog [25]. Finally, we have applied the algorithm to generate the morph sequence for two models obtained from a 3D scanner The results are illustrated in figures 12, 13 and 16. In addition to the geometry, the textures of the model were interpolated to produce the final morphing sequence.

7. Conclusions

We have presented a method that performs morphing between arbitrary genus-0 objects without any user intervention. The sphere mapping can be considered as a preprocessing step and stored along with the representation of the solid. The merging is very fast in the average case, and the interpolation is implemented with GPU GLSL shaders. Finally, we have presented a fully automated technique for feature matching and alignment that greatly improves the visual effect and allows for applying controlled morphing to CAD model editing. We have used our method successfully on object pairs of similar topology (for examples busts) and of a quite different one (fish and duck). We are currently exploring the feasibility of parallelization through GPUs of the optimization phase and the use of user defined constraints for feature matching and morphing-based editing.

Furthermore, recent results on parallel computation of spherical parameterizations for mesh analysis [33] may adapted for fast accurate feature region detection. Finally, the benefits of using randomized subgraph matching algorithms for detecting common feature patterns [30] in objects with large

number of features should be investigated further.

References

[1] C. Hoffmann, R. Joan-Arinyo, On user-defined features, Computer Aided Design 30 (1998) 321–332.

[2] A. Lerios, C. D. Garfinkle, M. Levoy, Feature-based volume metamorhosis, in: Proceedings of SIGGRAPH 1995, ACM SIGGRAPH, pp. 449–456.

[3] J. R. Kent, W. E. Carlson, R. E. Parent, Shape transformation for polyhedral objects, in: Proceedings of SIGGRAPH 1992, volume 26(2), New York, Published as Computer Graphics, 1992, pp. 47–54.

[4] W. E. Lorensen, H. E. Cline, Marching cubes: A high resolution 3d surface construction algorithm, in: Proceedings of SIGGRAPH 87, published as Computer Graphics, ACM SIGGRAPH, pp. 163–169.

[5] T. Kanai, H. Suzuki, F. Kimura, 3d geometric metamorphosis based on harmonic map, in: PG '97: Proceedings of the 5th Pacific Conference on Computer Graphics and Applications, IEEE Computer Society, Washington, DC, USA, 1997, p. 97.

[6] M. Alexa, Merging polyhedral shapes with scattered features, The Visual Computer 16 (2000) 26–37.

[7] M. Zwicker, C. Gotsman, Meshing point clouds using spherical parameterization, in: Proceedings of the Eurographics Symposium on Point-Based Graphics, Zurich.

[8] A. Sheffer, C. Gotsman, N. Dyn, Robust spherical parameterization of triangular meshes, Computing 72 (2004) 185–193.

[9] S. Saba, I. Yavneh, C. Gotsman, A. Sheffer, Practical spherical embedding of manifold triangle meshes, in: Proceedings of the International Conference on Shape Modeling and Applications 2005, pp. 258–267.

[10] I. Friedel, P. Schröder, M. Desbrun, Unconstrained spherical parameterization, Journal of Graphics, GPU, and Game Tools 12 (2007) 17–26.

[11] J. Schreiner, A. Asirvatham, E. Praun, H. Hoppe, Inter-surface mapping, ACM Trans. Graph. 23 (2004) 870–877.

[12] V. Kraevoy, A. Sheffer, Cross-parameterization and compatible remeshing of 3d models, ACM Trans. Graph. 23 (2004) 861–869.

[13] T.-Y. Lee, C.-Y. Yao, H.-K. Chu, M.-J. Tai, C.-C. Chen, Generating genus-n-to-m mesh morphing using spherical parameterization: Research articles, Comput. Animat. Virtual Worlds 17 (2006) 433–443.

[14] P. Kanonchayos, T. Nishita, S. Yoshihisa, T. L. Kunii, Topological morphing using reeb graphs, in: CW '02: Proceedings of the First International Symposium on Cyber Worlds (CW'02), IEEE Computer Society, Washington, DC, USA, 2002, p. 0465.

[15] C.-H. Lin, T.-Y. Lee, Metamorphosis of 3d polyhedral models using progressive connectivity transformations, IEEE Transactions on Visualization and Computer Graphics 11 (2005) 2–12.

[16] I. Fary, On straight line representation of planar graphs, Acta Univ. Szeged Sect. Sci. Math. 11 (1948) 229–233.

[17] W.T.Tutte, How to draw a graph, Proc. London Math. Soc 13 (1963) 743–768.

[18] M. S. Floater, Parametrization and smooth approximation of surface triangulations, Computer Aided Geometric Design 14 (1997) 231–250.

[19] C. Brechbuhler, G. Gierig, O. Kubler, Parametrization of closed surfaces for 3d shape description, Computer Vision and Image Understanding 61 (1995) 154–170.

[20] C. Gotsman, X. Gu, A. Sheffer, Fundamentals of spherical parameterization for 3d meshes, in: ACM Transactions on Graphics 22, pp. 358–363.

[21] I. Fudos, C. M. Hoffmann, A graph-constructive approach to solving systems of geometric constraints, ACM Trans. Graph. 16 (1997) 179–216.

[22] M. Sitharam, A. Arbree, Y. Zhou, N. Kohareswaran, Solution space navigation for geometric constraint systems, ACM Trans. Graph. 25 (2006) 194–213.

[23] M. Desbrum, M. Meyer, P. Alliez, Intristic parameterization of surface meshes, in: Eurographics Proceedings.

[24] S. Hildebrandt, H. Kaul, K.-O. Widman, Dirichlet's boundary value problem for harmonic mappings of riemannian manifolds, Mathematische Zeitschrift 147 (1976) 225–236.

[25] Aim@shape, AIM@SHAPE Shape Repository v4.0, Department of Genova, Institute for Applied Mathematics and Information Technologies, CNR, http://shapes.aimatshape.net, AIM@SHAPE Project.

[26] Blender, Blender Suite, Open Source Suite, http://www.blender.org, Blender Foundation.

[27] A. Wachter, L. T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, Mathematical Programming 106 (2006) 25–57.

[28] A. Forsgren, P. E. Gill, M. H. Wright, Interior methods for nonlinear optimization, SIAM Review 44 (2002) 525–597.

[29] V. Stamati, I. Fudos, A feature based approach to re-engineering objects of freeform design by exploiting point cloud morphology, in: SPM '07: Proceedings of the 2007 ACM symposium on Solid and physical modeling, ACM, New York, NY, USA, 2007, pp. 347–353.

[30] A. Berner, M. Bokeloh, M. Wand, A. Schilling, H.-P. Seidel, A graph-based approach to symmetry detection, in: Symposium on Volume and Point-Based Graphics, Eurographics Association, Los Angeles, CA, 2008, pp. 1–8.

[31] D. Eppstein, Subgraph isomorphism in planar graphs and related problems, J. Graph Algorithms Appl. 3 (1999).

[32] Stanford, The Stanford 3D Scanning Repository, Stanford University, http://graphics.stanford.edu/data/3Dscanrep, Stanford Computer Graphics Laboratory.

[33] T. Athanasiadis, I. Fudos, Parallel computation of spherical parameterizations for mesh analysis, Computers & Graphics 35 (2011). Shape Modeling International 2011.

[34] G. H. Golub, C. V. Loan, Matrix Computations, Johns Hopkins Univ. Press, 1996.

Appendix A. Solving Linear Systems with Laplacian Smoothing

Let (x_i, y_i) denote the coordinates of the ith node of a mesh. In addition, let the coordinates of its adjacent vertices be $(x_j, y_j) : v_j \in N_i$, where N_i denotes the set of neighbors of node v_i. If we assign a set of positive weights w_{ij}, where w_{ij} is the weight of a neighbor node j when determining node i then,

We will prove *Theorem 1*, i.e. that the linear system (1), which expresses the position of each node as a convex combination of its neighbors, has a unique solution if at least one node is fixed.

Proof. Let b and m represent the numbers of boundary (or fixed) and interior (or free) nodes, respectively. Next, define x_B and y_B to be vectors of length b that contain the initial x and y coordinates of the boundary nodes. Similarly, define x_I and y_I to be the vectors of length m that contain the initial x and y coordinates of the interior nodes. Thus, $[x_B|y_B]$ and $[x_I|y_I]$ contain the original positions of the boundary and interior nodes respectively. The weighted matrix L, for the graph $G(V; E; w)$ is:

$$L(i,j) = \begin{cases} -w_{ij}, & i \neq j \\ \sum_{k \in V} w_{ik}, & i = j \end{cases} \qquad w_{ij} = 0, \quad (i,j) \notin E \tag{A.1}$$

where the boundary nodes are placed in last b rows and columns $m + 1, \ldots, m + b$, i.e. after the interior nodes which are placed in the first m rows and columns $1, \ldots, m$. Let $A = [A^I|A^B]$ be the matrix that is derived from the weighted matrix L by deleting its last b rows. Then, the linear system (1) is expressed as:

$$A^I[x_I|y_I] = -A^B[x_B|y_B] \tag{A.2}$$

where A^I is an $m \times m$ matrix that contains all the weights corresponding to the interior neighbors. In addition, A^B is an $m \times b$ matrix contains all of the weights corresponding to the boundary neighbors. Because the mesh is connected and a positive weight is associated with each edge, A^I is irreducible. In addition, $|a^I_{ii}| \geq \sum_{j=1, j\neq i}^{m} |a^I_{ij}|$ for each row because the diagonal elements are 1, and the off-diagonal elements are negative summing to a value in $[-1, 0]$. Equality is true if the corresponding vertex is connected only with interior nodes. Therefore, if $b > 0$, there exist i such that $|a^I_{ii}| > \sum_{j=1, j\neq i}^{m} |a^I_{ij}|$, and

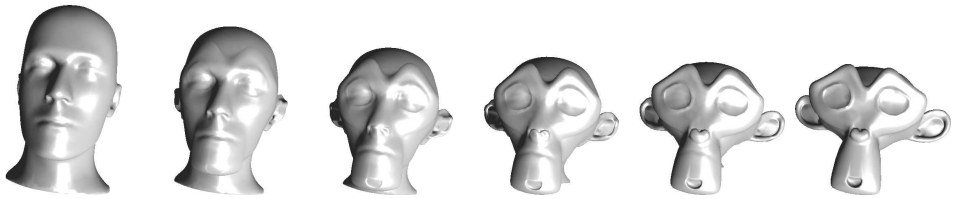
A^I is weakly dominant. Thus, A^I is invertible and has a unique solution [34], if there is at least one boundary node.

Moreover, it can be shown that the *Jacobi* iteration for each row i represents a step of the *simultaneous version* of Laplacian smoothing, where all the positions are modified simultaneously,

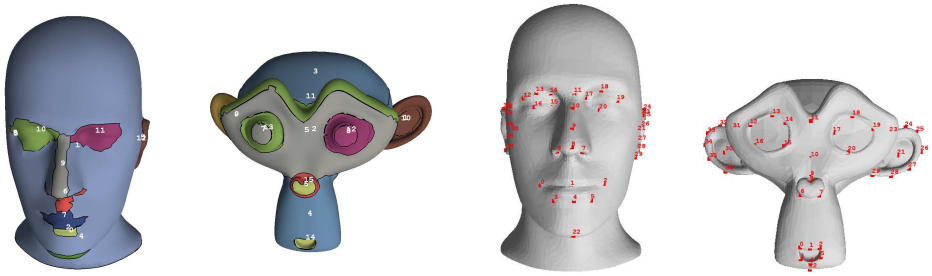
$$[x_I|y_I]_i^{k+1} = \sum_{\substack{j=1 \\ j\neq i}}^{m} w_{ij}[x_I|y_I]_j^k + \sum_{j=m+1}^{m+b} w_{ij}[x_B|y_B]_j \tag{A.3}$$

Similarly, it can be shown that the iterations produced by the *Gauss-Seidel* method are the same as the *sequential version* of Laplacian smoothing, where the positions are modified sequentially and depend on the order in which the vertices are considered. Since A^I is irreducible and weakly dominant and thus invertible, both Jacobi and Gauss-Seidel methods converge [34], and thus Laplacian smoothing converges to the same point which is the solution of the linear system (A.2). □

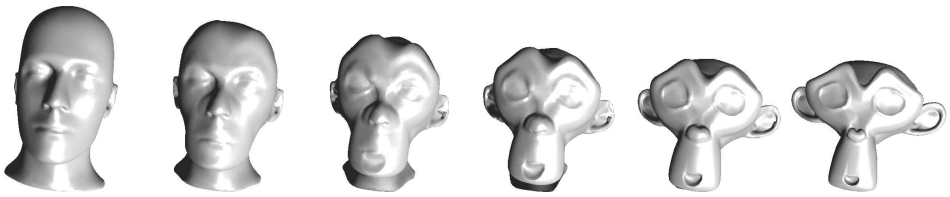
(a)



(b)



(c) (d)



(e)

Figure 14: An example for the entire feature-based morphing process: (a) (from left to right): The head model (11042 faces) and the optimized spherical parameterization, Suzanne (5600 faces) and the corresponding optimized spherical parameterization, and finally the optimized spherical parameterization of Suzanne with respect to the feature point pairs detected. (b) Morphing without feature point matching (the rightmost spherical parameterization of (a) is not used). (c) The feature regions are detected and paired. (d) Establishing feature point correspondence between the two meshes. (e) Finally, after optimizing one of the spherical parameterizations with the use of the feature point pairs, a visually smooth morphing sequence is obtained by linearly interpolating the vertices.

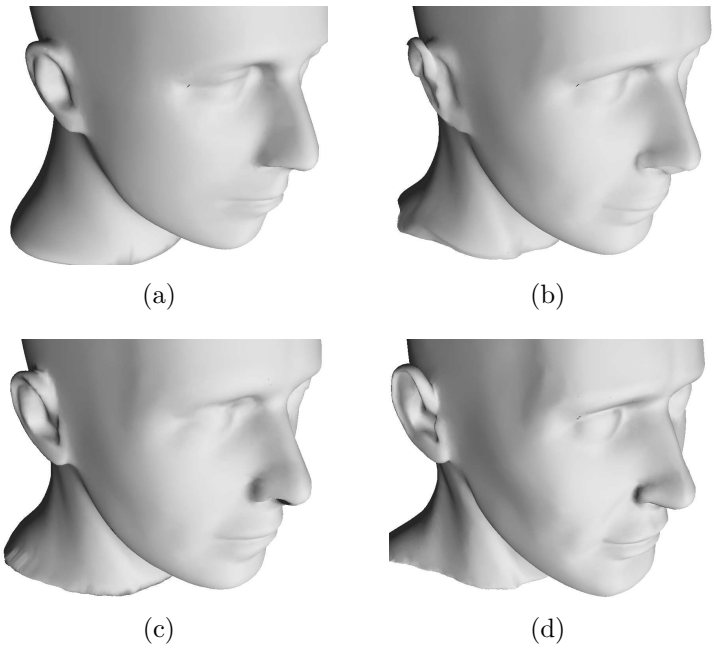


Figure 15: Close-up of the morphing sequence: (a) Model M_1, (b) 50% morph without feature point matching, (c) 50% morph with feature point matching and (d) target model M_2. The improvement around the ear area with feature point matching in (c) as compared to morphing without feature point matching in (b) is evident.

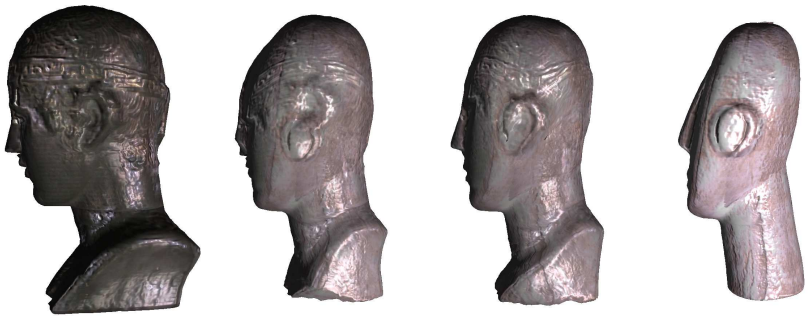


Figure 16: Comparison of morphing results: (a) Model M_1, (b) 50% morph without feature point matching, (c) 50% morph with feature point matching and (d) target model M_2. The improvement around the ear area and the outline of the model with feature point matching in (c) as compared to morphing without feature point matching in (b) is apparent.