Feature-based 3D Morphing based on Geometrically Constrained Sphere Mapping Optimization

Theodoris Athanasiadis
Dept. of Computer Science
University of Ioannina
GR45110 Ioannina, Greece
thathana@cs.uoi.gr

Ioannis Fudos
Dept. of Computer Science
University of Ioannina
GR45110 Ioannina, Greece
fudos@cs.uoi.gr

Christophoros Nikou
Dept. of Computer Science
University of Ioannina
GR45110 Ioannina, Greece
cnikou@cs.uoi.gr

Vasiliki Stamati
Dept. of Computer Science
University of Ioannina
GR45110 Ioannina, Greece
vicky@cs.uoi.gr

ABSTRACT

Current trends in free form editing suggest the development of a new novel editing paradigm for CAD models beyond traditional CAD editing of mechanical parts. To this end we wish to develop accurate, 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 mapping process built on an optimization technique that uses a target function to maintain the correspondence among 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 morphological characteristics between the two morphed objects and performs morphing according to this feature correspondence list. Alignment is obtained without user intervention and is based on pattern matching between the feature graphs of the two morphed objects.

Categories and Subject Descriptors

J.6 [**Computer-Aided Engineering**]: Computer-Aided Design (CAD); I.3.7 [**Three-Dimensional Graphics and Realism**]: Animation

General Terms

feature-based morphing

Keywords

Laplacian smoothing, features, CAD models, solid modeling

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SAC'10 March 22-26, 2010, Sierre, Switzerland.
Copyright 2010 ACM 978-1-60558-638-0/10/03 ...$10.00.

1. INTRODUCTION

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

Although we have some 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* [11] and *mesh-based* or *structural* [8] approaches. The volume-based approach represents a 3D object as a set of voxels usually leading to computationally intensive computations. The mesh-based approach exhibits better results in terms of boundary smoothness and rendering, since the intermediate morphs are represented as volumes. Techniques such as marching cube [13] 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 the two considered objects. 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 (interpolation problem). For performing structural morphing, we can use *boundary representation (Brep)* or *surface representation* in which we represent each object by its surface description, or *volumetric or solid meshes*, for instance tetrahedral representations. In volumetric mesh morphing, it is much easier to maintain robustness and avoid the

folding phenomenon. However, it is computationally expensive compared to surface mesh morphing since the number of elements in the former case is much larger in comparison to the latter.

In this paper we propose an efficient surface mesh morphing that maintains robustness. It introduces 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 approach builds on a spherical mapping approach presented in [4] for the purpose of parameterization of closed surfaces. Our mapping works in two phases. In the first phase, we perform an initial mapping. In the second phase, we optimize the mapping to achieve a better placement under specific geometric criteria and under a set of topological constraints. For the first phase, we present a much faster alternative to [4] based on Laplacian smoothing and adapt the second phase accordingly to capture morphing related requirements. 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 an easy to implement feature-compatible method for mapping genus-0 3D objects on the sphere using an optimization technique that achieves a mapping geometrically similar to the original object while preserving connectivity and topology with the use of geometric constraints.
- Introduces a feature-based fully automated method that achieves smooth visual results in morphing between objects with structural similarities.

Section 2 presents related work on 3D morphing. Section 3 presents the sphere mapping 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 correspondence list of the two solid representations. Section 6 presents an experimental evaluation of our method and some visual morphing results. Finally, section 7 provides 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. [8] 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.

In [6] the authors use a spring system to model the mesh and gradually force the mesh to expand or shrink on the unit sphere by applying a force field. Our method uses a similar technique for determining an initial mapping over the unit sphere. Methods using springs do not always produce acceptable mappings especially when handling complex non convex objects. We overcome this problem successfully in our approach.

[2, 19] use a spring-like relaxation process. 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. Our approach provably achieves convergence with no foldovers and collapsing triangles.

[16, 14, 5] 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.

[15] presents 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. [9] also presents 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.

[10] uses reeb-graphs and boolean operations 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 [7]. 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.

[12] provides 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 expense of a considerable preprocessing time for mapping.

The method presented in this paper overcomes these limitations 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 an initial state and then improved by nonlinear optimization. For smoother morphing that takes advantage of object morphology we introduce a feature-based approach. Feature correspondence

is performed automatically without any user intervention.

3. TOPOLOGY PRESERVING MAPPING TO THE 3D SPHERE

3.1 Initial Mapping

We have used two alternative methods for obtaining an initial mapping. *Thermal conduction* adapted from [4], is actually an initial mapping of bounding voxels and is performed in polar coordinates. Two polar coordinates θ and ϕ 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. [4] suggests selecting the poles based on the z coordinate in object space and this is reasonable due to the fact that they use it for voxel objects. Instead, we have implemented this initial mapping by selecting as poles the vertex pair with the largest distance between them (diameter of the solid).

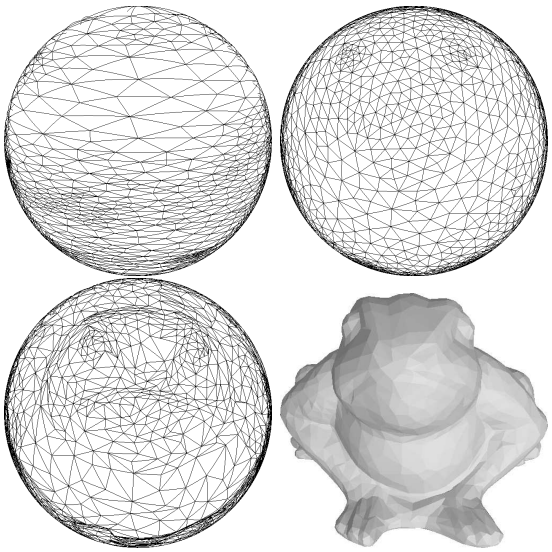


Figure 1: The result of initial mapping with the thermal conduction method (top left), with the Laplacian smoothing technique (top right), after optimization (bottom left) and finally the original frog model (bottom right).

Laplacian smoothing is a simple and efficient method for mesh smoothing. Every node is progressively shifted towards the centroid of its adjacent nodes. This is a local operation since at each step the movement of a vertex is determined only by its neighboring vertices.

A mesh can be thought of as a spring system by considering each edge connecting two nodes as a linear spring. Laplacian smoothing is then considered for minimizing the spring forces that are active on each node. Since a balanced spring system over the sphere can not contain folded elements, it turns out that if Laplacian smoothing is applied to every vertex and the vertex is projected on the sphere, all folded elements tend to unfold after a sufficient number of iterations. Since Laplacian smoothing does not perform any triangle area balancing, certain elements may become degenerate. Besides, the relaxation process may collapse if one or more elements overgrow [2]. We use a variation where we determine the position based on the weighted sum of the centroids of the surrounding triangles of each vertex. We use as weights the areas of each such triangle. This simple approach yields a smoother mesh with more balanced element area since larger polygons tend to attract vertices while smaller polygons tend to repulse them. It is easy to prove convergence to robust configurations (avoidance of foldovers and degenerate edges and triangles) by arguing that complicated mesh folding consists of a sequence of simple triangle foldings. However, each individual folding is not stable and is forced to unfold according to the area weighted centroid attraction rule. Figure 1 shows the results of the initial mapping when applying the thermal conduction method and Laplacian smoothing on the frog from [1]. Laplacian smoothing is faster, is guarantied to provide a robust unfolded initial mapping and preserves similarities with the initial mesh.

The above procedure is expressed concisely by the following two steps: We first project each vertex on the unit sphere by:

$$S(V_o) = \frac{V_o}{||V_o||}, \forall V$$

where V is the original mesh vertex and $S(V)$ is its image on the unit sphere surface. Then, while folded elements still exist, we transform each vertex V_s (initially $V_s = S(V)$) on the unit sphere surface as follows:

$$V_s = \frac{\sum_i^m area_i Centroid_i}{||\sum_i^m area_i Centroid_i||}$$

where $area_i$ is the area of the corresponding i-th triangle that is adjacent to vertex V_s, $Centroid_i$ is the centroid of the same triangle. Note that vertex V_s is adjacent to m triangles. The normalizing denominator maintains V_s on the unit sphere.

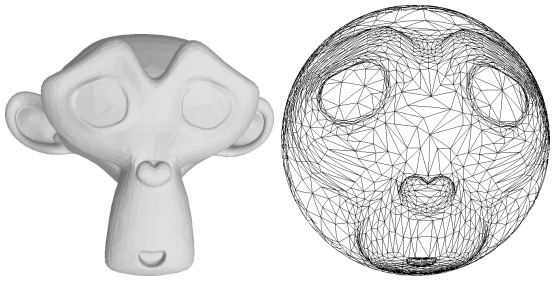


Figure 2: The final optimized result of mapping (right) applied to the Blender monkey with 5600 faces (left).

3.2 Optimizing Mapping for Morphing

In *spherical parameterization* it is common to maintain uniform triangle area and avoid long edges. We present this approach and adapt it by introducing a more appropriate for morphing set of constraints.

For each vertex $V_s(v_x, v_y, v_z)$:

$$v_x^2 + v_y^2 + v_z^2 = 1, \forall V_s(v_x, v_y, v_z) \quad (1)$$

To avoid unequal faces, for each face f on the sphere, the

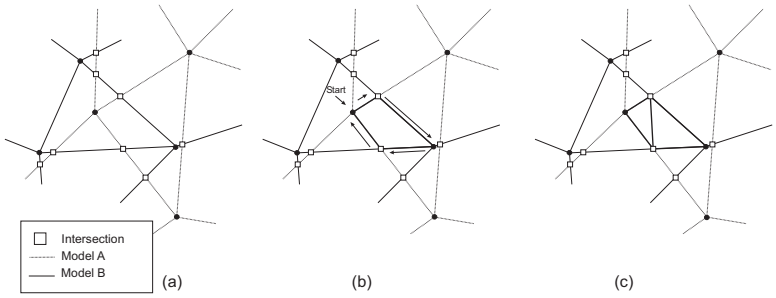


Figure 3: (a) Finding intersections in merged topology (b) Curve faces visited in clockwise manner (c) Triangulation

area is constrained to be exactly

$$area(f_s) = \frac{4\pi}{n}, \forall f_s \tag{2}$$

where n is the number of faces. Finally the angles of each face are constrained to be in $[0, \pi]$ which is compiled to six inequalities per triangular face. The objective function that favors short lengths on face edges is:

$$\sum_{\forall f_s} \sum_{i=1}^{3} cos(s_i^{f_s}) \tag{3}$$

where $s_i^{f_s}$ is the angle (length of the arc) formed by the i-th edge of face f_s and the center of the unit sphere. We implemented this method for morphing and observed that the results were not appropriate for our purposes. The method is very slow in converging and may place some faces very far away from their original position or place neighboring faces in distant spots on the sphere.

For this reason we use the following set of constraints and objective function that are more appropriate for morphing: *Geometric Constraints:* For each vertex $V(v_x, v_y, v_z)$ we use equation (1) to keep the vertices on the unit sphere surface. *Topological Constraints:* For each face with vertices V_0, V_1, V_2 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:

$$(V_1 \times V_2) \cdot V_0 > 0 \tag{4}$$

Objective Function: We use as the objective function to be minimized the sum of all inner products of every mapped vertex V_s with their corresponding original position $V_o = M(V_s)$ on the mesh.

$$\sum_{\forall V_s} V_s \cdot M(V_s) \tag{5}$$

For optimization we have used several local optimization techniques. The best results were attained with a gradient descent method. Figure 1 illustrates the optimized mapping for the frog, while Figure 2 illustrates the final optimized mapping on the sphere for the Blender monkey [3]. The preservation of the initial characteristics is apparent.

4. SURFACE CORRESPONDENCE AND INTERPOLATION

Following the successful mapping of two objects A and 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 3 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 the other model B along with the intersection points are projected on 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 using GPUs using a simple morphing shader for interpolating vertices and features (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

To detect feature regions in a point cloud we built on a method [17] developed earlier for reverse engineering based on discovering features on the point cloud by detecting local changes in the morphology of the point cloud. We use region

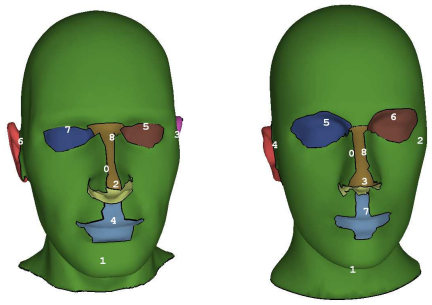


Figure 4: Detecting feature regions in two head meshes.

growing, detection of rapid variations of the surface normal and the concavity intensity, i.e. the distance from the convex hull. This results in a number of regions that represent object feature areas (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 morphed solids.

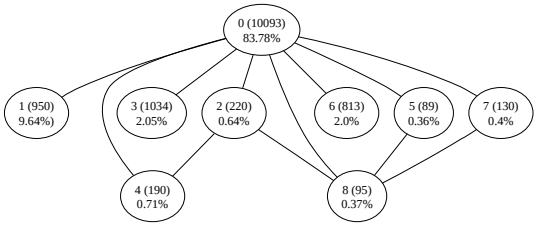
More specifically, morphological features in the point cloud are detected using a characteristic called *concavity intensity* of a point which represents the smallest distance of a point from its convex hull.

Definition 1: *Concavity intensity* of a vertex V_o of a mesh denoted by $I(V_o)$ is the distance of V_s from the convex hull of the mesh.

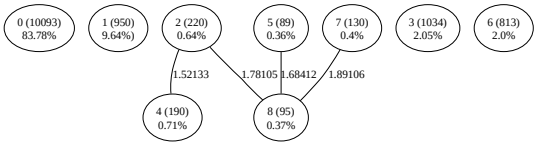
This characteristic is used to detect concave features in the point cloud. Feature regions are detected by rapid variations of the surface normal and the concavity intensity. These two characteristics are combined in a region growing method that results in sets of points corresponding to individual features (Figure 4). After obtaining the features of the object we create a connectivity graph that captures adjacency information as illustrated in Figure 5. For every edge we calculate the geodesic distances between the centroids of the corresponding feature regions. The graphs are then simplified by reducing edges that correspond to large geodesic distances (Figure 5). In addition, small regions that can introduce noise and are not significant are merged. The reduced adjacency graphs are used to perform a 3D alignment of the two models and establish a correspondence between the region patches. This is achieved by first matching the three 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 but we also take into account the distance between them. Furthermore, we also take into consideration the area covered by each region favoring the matching of regions covering similar areas. Equation (6) summarizes the distance between two feature regions i and j, where C_i and C_j are the centroids of regions i and j.

$$Distance(i,j) = ||C_i - C_j|| \frac{\max\limits_{i,j} Area}{\min\limits_{i,j} Area} \tag{6}$$

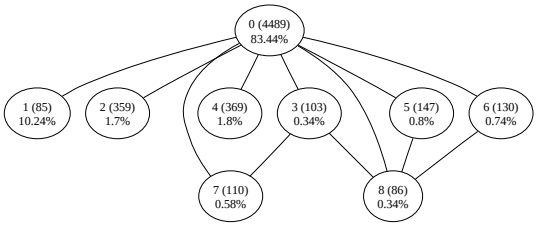
For each feature region we detect points with certain properties that provide a high level description of specific structural characteristics of the solids. The resulting point set, called a *feature point set*, provides a high-level description of concave and convex extrema the object. For each object we have:



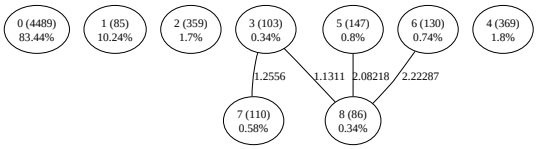
(a) Original adjacency graph of M_1 showing the region number (see Figure 4), the number of nodes and the area covered in the original model.



(b) Reduced graph of M_1, all edges with large geodesic distances are eliminated.



(c) Original graph of M_2



(d) Reduced graph of M_2

Figure 5: Graph reduction of the head meshes.

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

Following the establishment of a correspondence between the region patches of the two models the feature points of the corresponding patches are paired according to the distance between them.

For the feature based mapping of the second model we use the following set of constraints and objective functions to obtain a more appropriate mapping based on the feature point correspondence of the models:

Geometric Constraints: For each vertex $V_s(v_x, v_y, v_z)$ we use equation (1).

Topological Constraints: In addition to equation (4), the length of each edge (circular arc over the sphere) must remain the same during the optimization:

$$V_{i1} \cdot V_{i2} = ||M(V_{i1}) - M(V_{i2})|| \tag{7}$$

recall that $M(V)$ is the initial position of vertex V on the original mesh. By doing so, we preserve the morphology of the second object during the optimization process. This avoids very long stretches of the triangles to satisfy a certain feature point pair matching.

Objective Function: We use as the objective function to

Input: Two polyhedral representations for objects S_1 and S_2
for *each vertex* V *of* S_1 **do**
calculate $I(V)$
end
for *each vertex* U *of* S_2 **do**
calculate $I(U)$
end
for S_1 *and* S_2 **do**
compute the corresponding feature region sets F_1 and F_2
end
for F_1 *and* F_2 **do**
compute the corresponding connectivity graphs and perform graph reduction on them
end
Establish a correspondence of the three nodes with the highest degree in the two graphs and perform a 3D alignment of F_1 and F_2 up to scaling,rotation and translation based on that correspondence;
for *each feature region in* F_2 **do**
find the closest feature region in F_1 that covers similar area and match the corresponding feature point sets
end
Perform the sphere mapping on S_1;
Perform the sphere mapping of S_2 under the additional constraint that each mapped point has to be close to the corresponding point of the first object;

Figure 6: The algorithm for feature based morphing.

be minimized the sum of all inner products of every mapped feature vertex V_B of the second model with their corresponding mapped feature vertex of the first model V_A

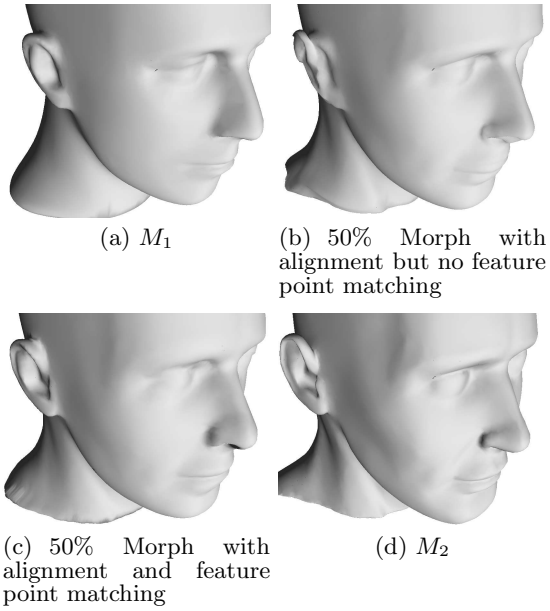
$$\sum_{\forall V_A} V_A \cdot V_B \tag{8}$$

The algorithm for feature-based morphing is illustrated in Figure 6, whereas Figure 11 illustrates the visual improvement offered by this method.

6. EXPERIMENTS AND PERFORMANCE EVALUATION

We have developed software for implementing mapping, merging and interpolation as described in the previous sections. The platform used for development was 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 Thermal and the Laplacian smoothing initialization. The number of iterations refers to the optimization phase, while the 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



(a) M_1 (b) 50% Morph with alignment but no feature point matching

(c) 50% Morph with alignment and feature point matching (d) M_2

Figure 11: Close-up of the morphing sequence.The improvement around the ear area is noticeable.

number of iterations increases quadratically over the number of faces 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 show the results for the same set of experiments for the same model with different LODs ranging from 854 faces up to 5610 for the monkey model. This set of experiments confirms the above observations.

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

Figures 7, 8, 9 and 10 illustrate 4 different cases of morphing. We have performed the experiments on well-known models such as the Stanford bunny [18], the Blender monkey [3] and the Aim@shape frog [1].

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 preprocessing 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 quite different topology (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.

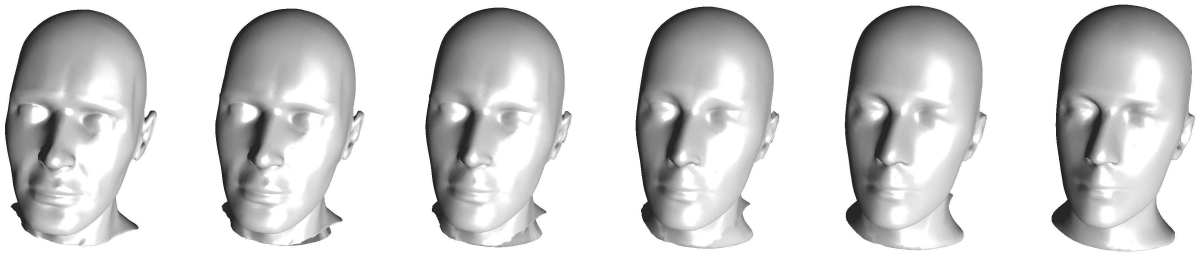


Figure 7: Morphing with alignment and feature point matching. Morphing is visuallly smooth through the entire sequence.

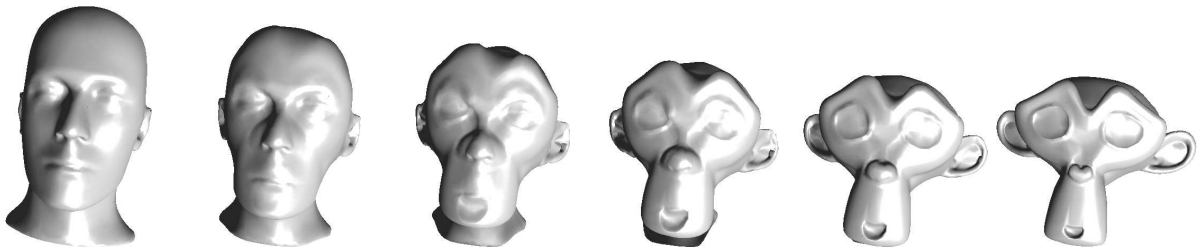


Figure 8: Morphing with alignment and feature point matching. Morphing is visuallly smooth through the entire sequence.

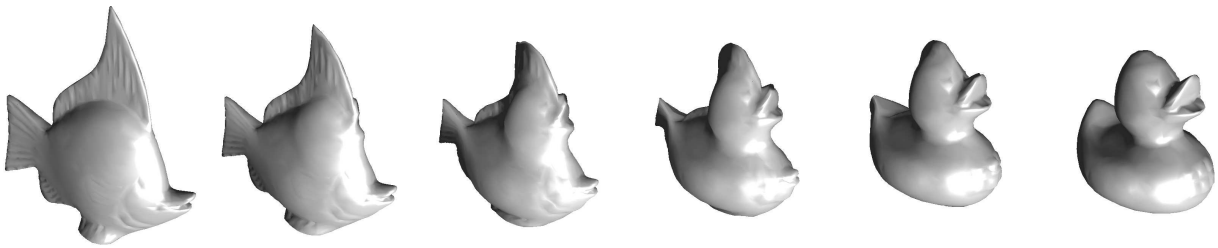


Figure 9: Morphing with alignment but no feature point matching: fish (4994 faces) to duck (1926 faces), merged topology has 28526 faces.

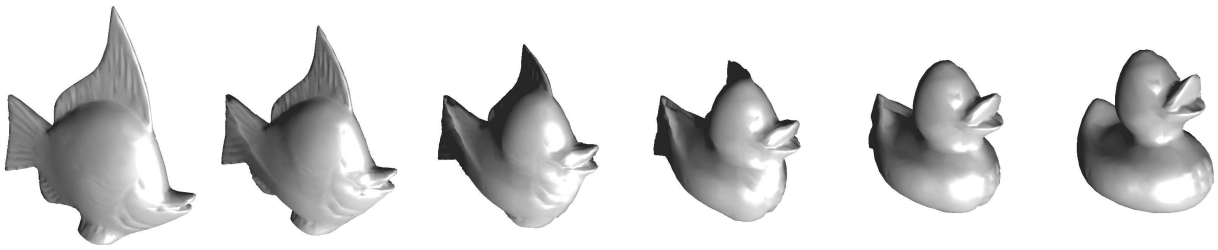


Figure 10: Morphing with alignment and feature point matching: fish (4994 faces) to duck (1926 faces), merged topology has 33038 faces.

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

model	method	# vertices	# faces	# constraints	# iterations	time (secs)
Monkey	Laplace	429	854	2991	36	10.9
Monkey	Thermal	429	854	2991	78	22.6
Bunny(Lod1)	Laplace	440	876	3068	94	24.3
Bunny(Lod1)	Thermal	440	876	3068	165	52.0
Frog(Lod1)	Laplace	1964	3924	13736	70	422.2
Frog(Lod1)	Thermal	1964	3924	13736	152	895.8

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

model	method	# vertices	# faces	# constraints	# iterations	time (secs)
Monkey(Lod1)	Laplace	429	854	2991	36	10.9
Monkey(Lod1)	Thermal	429	854	2991	78	22.6
Monkey(Lod2)	Laplace	703	1402	4909	26	21.3
Monkey(Lod2)	Thermal	703	1402	4909	70	54.8
Monkey(Lod3)	Laplace	1404	2804	9816	49	151.3
Monkey(Lod3)	Thermal	1404	2804	9816	91	271.3
Monkey(Lod4)	Laplace	2807	5610	19637	79	934.0
Monkey(Lod4)	Thermal	2807	5610	19637	136	1578.6

8. REFERENCES

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

[2] M. Alexa. Merging polyhedral shapes with scattered features. *The Visual Computer*, 16(1):26–37, 2000.

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

[4] C. Brechbuhler, G. Gierig, and O. Kubler. Parametrization of closed surfaces for 3d shape description. *Computer Vision and Image Understanding*, 61(2):154–170, 1995.

[5] I. Friedel and P. Schröder and M. Desbrun. Unconstrained spherical parameterization. In *ACM SIGGRAPH 2005 Sketches*, page 134. ACM, 2005.

[6] T. Kanai, H. Suzuki, and F. Kimura. 3d geometric metamorhosis based on harmonic map. In *Proc. of the 5th Pacific Computer Graphics and Applications*. IEEE, 1997.

[7] P. Kanonchayos, T. Nishita, S. Yoshihisa, and T. L. Kunii. Topological morphing using reeb graphs. In *Proc. of the 1st International Symposium on Cyber Worlds (CW'02)*, page 0465, Washington, DC, USA, 2002. IEEE.

[8] J. R. Kent, W. E. Carlson, and R. E. Parent. Shape transformation for polyhedral objects. In *Proc. of SIGGRAPH 1992, publ. as Computer Graphics*, volume 26(2), pages 47–54, 1992.

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

[10] T. Y. Lee, C. Y. Yao, H. K. Chu, M. J. Tai, and C. C. Chen. Generating genus-n-to-m mesh morphing using spherical parameterization: Research articles. *Comput. Animat. Virtual Worlds*, 17(3-4):433–443, 2006.

[11] A. Lerios, C. D. Garfinkle, and M. Levoy. Feature-based volume metamorhosis. In *Proc. of SIGGRAPH 1995*, pages 449–456. ACM, 1995.

[12] C. H. Lin and T. Y. Lee. Metamorphosis of 3d polyhedral models using progressive connectivity transformations. *IEEE Trans. of Visualization and Computer Graphics*, 11(1):2–12, 2005.

[13] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proc.of SIGGRAPH 87, publ. as Computer Graphics*, pages 163–169. ACM, 1987.

[14] S. Saba, I. Yavneh, C. Gotsman, and A. Sheffer. Practical spherical embedding of manifold triangle meshes. In *Proc. of the Int. Conf. on Shape Modeling and Applications '05*, pages 258–267, 2005.

[15] J. Schreiner, A. Asirvatham, E. Praun, and H. Hoppe. Inter-surface mapping. *ACM Trans. Graph.*, 23(3):870–877, 2004.

[16] A. Sheffer, C. Gotsman, and N. Dyn. Robust spherical parameterization of triangular meshes. *Computing*, 72(1-2):185–193, 2004.

[17] V. Stamati and I. Fudos. A feature-based approach to re-engineering objects of freeform design by exploiting point cloud morphology. In *Proc. SPM of '07*. ACM, Beijing, China 2007.

[18] Stanford. The Stanford 3D Scanning Repository, Stanford University, http://graphics.stanford.edu/data/3Dscanrep.

[19] M. Zwicker and C. Gotsman. Meshing point clouds using spherical parameterization. In *Proc. of the Eurographics Symposium on Point-Based Graphics*, Zurich, June 2004.