submitted to Eurographics Symposium on Geometry Processing (2004)

3D Morphing without User Interaction

Raquel Urtasun, Mathieu Salzmann, Pascal Fua

EPFL - CVLab
1015 Lausanne, Switzerland

Abstract

Nowadays, 3D morphing, defined as smoothly transforming a 3D object into another one, remains a challenging topic in the graphics community. The usual way to deal with such process consists first in establishing a point-to-point correspondence and then in computing the intermediate shapes. In the past, much effort has been invested to automate the correspondence problem, but all the proposed methods need some user interaction, consisting in a set of clicks, to generate visually pleasant results.
We present a fully automatic morphing algorithm for arbitrary genus 0 meshes, that combines improved registration techniques with feature points detection in order to automate the point correspondences and that directly defines a more realistic interpolation technique than the classical linear one. To our knowledge this is the first successful attempt to completely automate the process while resulting in meaningful intermediate shapes.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction

Object metamorphosis, commonly known as morphing, aims at smoothly transforming a 2D or 3D object into another one. While the problem of interpolating 2D objects seems almost solved, 3D morphing remains an open issue.

The usual way to transform an object into another is to split the problem into two sub-problems. The first one involves establishing point-to-point correspondences. The second step consists of computing the intermediate shapes given those correspondences. Whereas this second step is automatic, the first one typically requires extensive user interaction to yield pleasing intermediate shapes. This is usually achieved by manually specifying a number of correspondences, as the ones shown in Figure 4 for two face meshes.

By contrast, as shown in Figure 1, we propose a fully automatic algorithm that produces a smooth transition from one shape to the other, without any disturbing artifact in the intermediate shapes. It is applicable to any genus 0 meshes and, to the best of our knowledge, is the first successful attempt to completely automate the process while resulting in meaningful intermediate shapes.

We started from Kanai's well known algorithm [KSK98] and extended it in the following way:

1. Automate the process of creating harmonic maps.
2. Use of improved registration techniques not to align manually source and target meshes
3. Remove disturbing artifacts by using local curvature matching.
4. Use registration results to define a more realistic interpolation technique than the classical linear one.

This results in a fully automatic morphing algorithm for arbitrary genus 0 meshes with intermediate shapes preserving the geometrical properties of the source and target objects. Moreover the meshes do not need to be manually aligned as in most of the state of the art methods, as shown in Figure 1.

Even if the methods described in this paper to automate the morphing process are applied to Kanai's [KSK98] algorithm, they are general enough to be applied to other well know morphing techniques [ST98, Ale00, Ale02].

Section 3 describes the basic correspondence algorithm. The improvements presented in this paper to automate such algorithm are explained in Section 4 followed by the inter-

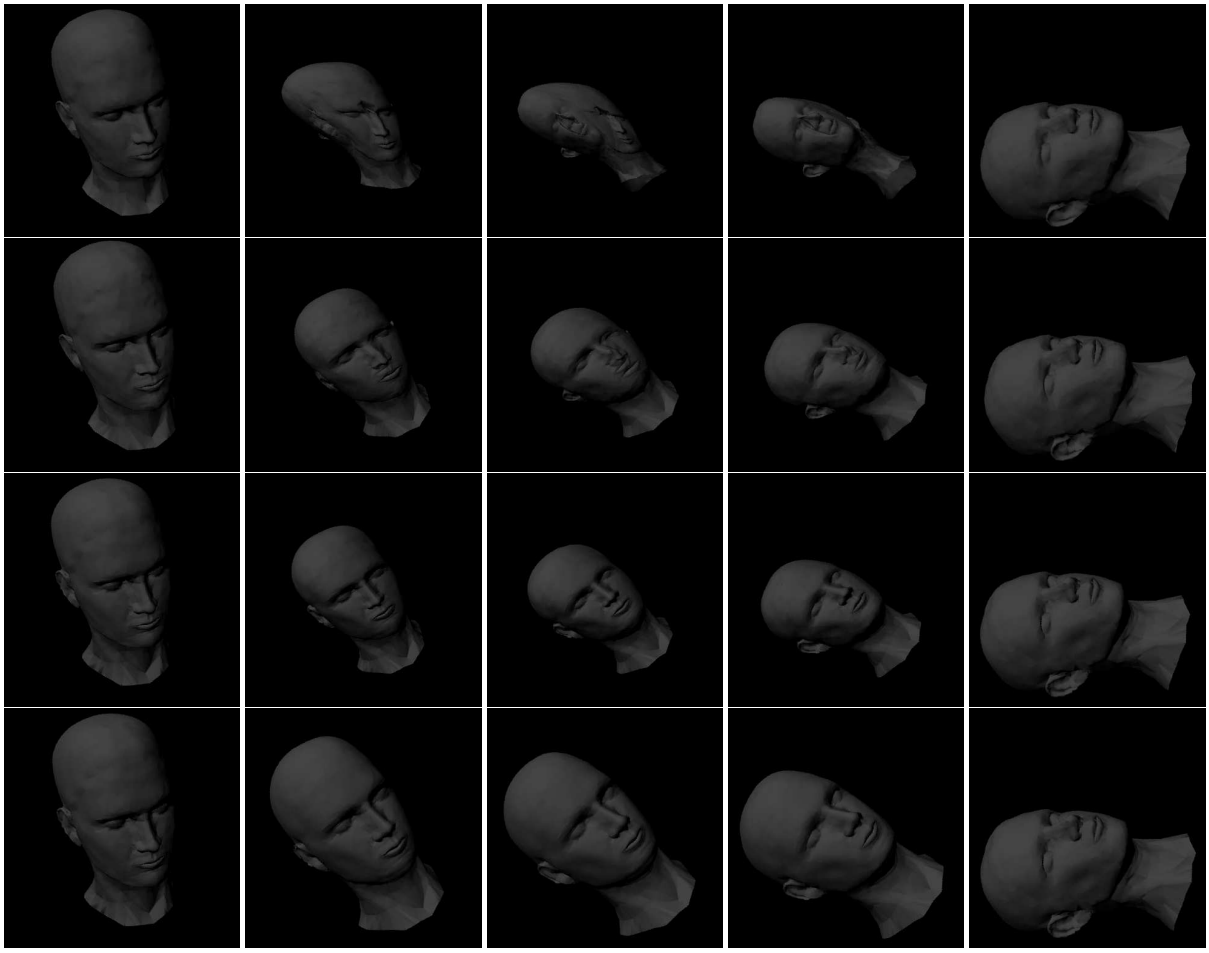

Figure 1: *Heads interpolation.* ***Top row****: Kanai's method of section 3 without manual interaction.* ***Second row****: Results using registration, linear interpolation, but no curvature alignment. This result is equivalent to Kanai's method by clicking in a subset of points of the source and target borders. The remaining results could not be obtained by such algorithm.* ***Third row****: ICP and curvature alignment with linear interpolation.* ***Bottom row****: ICP and curvature alignment with Slerp quaternion interpolation*

polation method. We end up with results, conclusion and a brief description of future work.

2. Related Work

Blanz et al. [BBPV03] proposed an automatic method for merging 3D objects of the same topology. These objects were deformed based on a PCA mesh and texture database. While showing impressive results, the applicability of the method is small since the meshes need to have the same topology, and a large database is needed. Surazhsky and Elber [SE01] introduced a way to automate parametric surface morphing based on the curvature.

The classical mesh morphing techniques are non-automatic and start by establishing the correspondences between the source and target meshes. The correspondence problem is solved in the past by building a parameterization of both meshes and by merging them to obtain meshes of the same topology. Two different types of parameterization have been studied in the past: projection to a disk and projection to a sphere.

In the case of the projection to a disk, Eck et al. [EDD*95] showed how to create a projection using discrete harmonic mapping. Such maps minimize the energy of the springs network where each edge is considered as a spring. A harmonic map tends to preserve the aspect ratio of the triangles while minimizing metric distortion.

Floater [Flo97] defined a shape preserving parameterization by computing weights that reflect the local shape of the mesh for barycentric mapping. Gregory et al. [GSL*98, GSL*99] introduced a recursive algorithm that

also aims at preserving the areas of the triangles. The disk is recursively divided into two pieces that are mapped independently. The separation is chosen in such a way to respect the ratio of the original areas.

In the field of spherical projections, Kent et al. [KCP92] were the first to show how to parameterize a star-shaped mesh onto a unit sphere. However, their algorithm only applies to a certain class of genus 0 meshes.

Shapiro et al. [ST98] presented one of the first reliable algorithms to convert an arbitrary genus 0 mesh into a convex polyhedron. They first remove vertices in order to simplify the shape to a tetrahedron. Then the removed vertices are reattached while keeping the shape convex.

Alexa [Ale00] introduced an algorithm based on barycentric mapping and adapted to the spherical case. He fixes four anchors and iteratively places the other vertices at the centroid of their neighbors. This method works for any genus 0 polyhedra, but depends on the initial state. The convergence to a valid embedding is not ensured, and in case of failure the initial state must be changed.

The interpolation problem that consists in computing the intermediate shapes has also been well studied in the past. The most straightforward method is linear interpolation. Such an interpolation gives satisfying results when objects have the same orientation, but different orientations can lead to unwanted deformations.

A better approach, introduced by Alexa et al. [ACOL00], consists in using the transformation $(\mathcal{R}t)$ between the two objects. The new positions of the vertices can then be computed as a combination of the Euclidean intermediate transform and mesh deformation.

We use planar projections since they are well known and easy to implement. We extend such technique to be fully automatic for arbitrary genus 0 meshes, combining improved registration techniques with feature points detection, that directly define a more realistic interpolation technique than the classical linear one. However, our work can be adapted to other morphing algorithms.

3. Basic method

The first step in morphing is to establish correspondences between the different points of the two given arbitrary meshes. In this section we briefly review the method we use to solve the correspondence problem. The approach was designed by Kanai [KSK98] based on Eck et al. harmonic mapping [EDD*95].

The main idea of using harmonic maps is, given a model $\mathcal{M} \subset \mathbf{R}^3$, to obtain a projection of such a model on a unit disk $\mathcal{H} \subset \mathbf{R}^2$, in order to be able to merge the source and target maps later. A harmonic map $h : \mathcal{M} \rightarrow \mathcal{H}$ can be visualized as a map that minimizes the total energy $E_{harm}(h)$ of a springs network, as defined in Equation 1. It tends to minimize metric distortion while attempting to preserve the aspect ratio of the triangles.

Rather than computing a harmonic map, Eck et al. [EDD*95] used a piecewise linear approximation for computational cost reasons. First, n vertices composing the boundary of the model are positioned on the boundary circle of the 2D unit disk, so that the angle between two connected vertices is proportional to the length of the edge between these two vertices.

The positions of the remaining vertices are then calculated by minimizing the total energy E_{harm}:

$$E_{harm}(h) = \frac{1}{2} \sum_{e_{ij} \in Edges(\mathcal{H})} \kappa_{i,j} \|h_i - h_j\|^2 \qquad (1)$$

where i, j denote indices of vertices v_i and v_j, and h_i, h_j are their position in $\mathcal{H}$. *Edges(H)* denotes the set of all edges in $\mathcal{H}$ and e_{ij} is an edge connecting vertex v_i with vertex v_j. $\kappa_{i,j}$ denotes the spring constant for edge e_{ij} and is given by:

$$\kappa_{i,j} = \frac{l_{i,k_1}^2 + l_{j,k_1}^2 - l_{i,j}^2}{A_{i,j,k_1}} + \frac{l_{i,k_2}^2 + l_{j,k_2}^2 - l_{i,j}^2}{A_{i,j,k_2}} \qquad (2)$$

where $l_{i,j}$ denotes the length of edge e_{ij} in $\mathcal{M}$ and $A_{i,j,k}$ is the area of facet *{i,j,k}* in $\mathcal{M}$.

Once harmonic maps for both source and target are computed, we need to merge them in order to have a point-to-point correspondence ready for interpolation. Kent et al's merging algorithm [KCP92], used by Kanai [KSK98], produces two topologically identical new meshes for the source and target objects. Each new mesh has a total number of vertices equal to $N_a + N_b + I_{tot}$, where N_a and N_b are the number of vertices in the source and the target respectively, and I_{tot} is the number of edge intersections. Then, a new triangulation must be defined.

Kanai et al. [KSK98] described a solution to the problem of building a new triangulation given vertices and edges. The basic idea is to go through each vertex, to sort its incident edges and to create a facet between any two continuous edges until every edge has a facet on both sides.

Let $\mathcal{H}$ be the new harmonic map resulting from the merging algorithm. Let h_i be a vertex of $\mathcal{H}$. The first step consists of sorting the edges incident to h_i in a counter-clockwise order. Once this is done, let e_{ij} and e_{ik} be two continuous edges. A new facet $f = \{i, j, k\}$ is generated by using e_{ij} and e_{ik}. Finally, if the edge e_{jk} did not exist, it is created and added to the list of edges. This operation continues for every vertex until each edge has a facet on both sides. The computed triangulation can be used for both new models that are now ready for interpolation.

Figure 2 shows the harmonic map computed for a human face. Even though harmonic mapping tends to preserve the aspect ratio of the triangles, we can notice compressed regions on the map corresponding to regions of the model hav-

ing a large area relative to their circumference such as the nose.

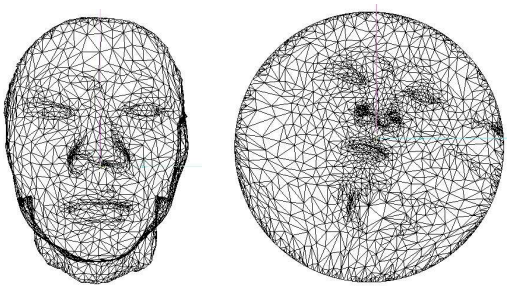

Figure 2: *Human face and its corresponding harmonic map.*

The main drawback of Kanai's method is that, without any user interaction, the geometry of the meshes is not taken into account. This results in losing the properties of the source and target meshes in the intermediate interpolation steps, as shown in the top row of Figure 1 for face meshes.

To solve this problem, Kanai et al. allow the user to click on border vertices of both meshes. One selected vertex per mesh, called the *boundary control vertex*, is used as a starting point to build the harmonic map. The remaining vertices, *boundary loop*, must be selected in the same direction for the source and target objects in order to avoid collapsing the resulting mesh to a single point during interpolation. The second row of Figure 1 depicts the face morphing when using Kanai's manual interaction. Note the local mismatches around the eyes, mouth, etc.

As will be described in the following section, our method avoids this manual step by combining improved registration techniques with feature points detection. As a first step, we find the global transformation between the two meshes. Then, using this transformation, we build both harmonic maps respecting their orientation and merge them. Finally, in order to avoid local deformations, we apply a feature alignment method.

4. Establishing Correspondences Automatically

To solve the correspondence problem, a general registration is first established in order to globally align the two meshes by determining the $\mathcal{R}t$-transform between them. Then a correlation based method is used to build the harmonic maps automatically. At this point, our results are equivalent to those obtained with the method presented in Section 3, but without the need of any click. Moreover, we improve these results by eliminating the unpleasant artifacts with a feature alignment approach based on curvature.

4.1. ICP Alignment

The *Iterative Closest Point* algorithm is an iterative approach to find a point-to-point correspondence as well as a transformation matrix between two sets of points in $\mathbf{R}^3$. While originally developed by Zhang [Zha94], we slightly modified Dewaele et al's improved method [DDH04] to decrease the chances of being trapped into local minima.

Let $S = \{x_i\}_{i\in[1,N]}$ and $S' = \{x'_j\}_{j\in[1,N']}$ be two sets of points in $\mathbf{R}^3$. Let $\mathcal{T}$ be the transformation between S and S', we say that S and S' are *registered* by transformation $\mathcal{T}$ if the sum of the distances between $\mathcal{T}(S)$ and S' is equal to zero. In a similar way, the sum of the distances between $\mathcal{T}^{-1}(S')$ and S must also be equal to zero.

The goal of registration is to find the motion between the two clouds of points S and S'. Such a motion actually corresponds to a rotation $\mathcal{R}$ and a translation t, that can be found by minimizing the following equation with respect to $\mathcal{R}$ and t:

$$\mathcal{F}(\mathcal{R},t) = \frac{1}{\sum_{i=1}^{n} p_i} \sum_{i=1}^{n} p_i d^2(\mathcal{R}x_i + t, S') \tag{3}$$

where $d(x,S')$ denotes the Euclidean distance of point x to S', as defined in Equation 4, and p_i takes the value 1 if the point x_i can be matched to a point on S' or 0 otherwise, in order to suppress the influence of outliers.

$$d(x_i,S') = \min_{x'\in S'} d(x_i,x') \tag{4}$$

This problem has first been solved by Zhang [Zha94] in an iterative 3 steps procedure:

1. For each point $x_i \in S$ find its correspondent as the closest point in S'.
2. Given the correspondences compute $\mathcal{T}$ which minimizes Equation 3.
3. Go to step 1 if the displacement is bigger than a threshold.

The algorithm described above uses one-to-one correspondences that can result in convergence instabilities. Granger et al [GP02] and Rangarajan et al [RCB97] proposed a solution for such problem where they take into account not a one-to-one correspondence but a one-to-n, where $n \ll N, N'$, with $N = |S|$ and $N' = |S'|$:

1. For each pair of points $(x_i, x'_j) \in [1,N] \times [1,N']$, compute the probability p_{ij} that x_i matches x_j using the distance $d_{ij} = \|\mathcal{R}x_i + t - x'_j\|$ in Equation 6.
2. Compute the transformation $\mathcal{T}$ that minimizes Equation 5.

$$\mathcal{F}(\mathcal{R},t) = \sum_{(i,j)\in[1,N]\times[1,N']} p_{ij} d_{ij}^2 \tag{5}$$

$$p_{ij} = \frac{1}{P_i} e^{\frac{-d_{ij}^2}{\sigma_p^2}} \quad with\ P_i = \sum_{k=1}^{N'} e^{\frac{-d_{ik}^2}{\sigma_p^2}} \tag{6}$$

Dewaele et al. [DDH04] introduced the concept of *modeling error* σ_{model}, that consists of the difference between both meshes S and S' if they are perfectly registered. They impose a monotonically decreasing criterion for the value of σ_p, from 2 or 3 times σ_{model} to a final value of σ_{model}. Decreasing σ_p within the first few iterations of the algorithm and never let it increase again may lead to be trapped easily into local minima As a matter of fact, we let σ_p increase, but storing the best transformation matrix $\mathcal{T}$ and the corresponding probabilities $\{p_{ij}\}$. At each step, the value of σ_p is computed as follows:

$$\sigma_p = \frac{1}{N}\sum_{i=1}^{N} \min_{j\in[1,N']} d_{ij} \tag{7}$$

Thus σ_p is a direct measure of the precision of the match between the two sets of points.

4.2. Automate the Harmonic Maps Construction

We improved the method described in Section 3 by avoiding user intervention for selecting boundary control vertices to orient both maps in the same way. Such vertices are automatically found by using the *Iterative Closest Point* algorithm (ICP) presented in the previous section.

The ICP algorithm gives us the rotation and translation between the original meshes as well as an estimation of the correspondence between the vertices given by the probability of Equation 6. The source map is built as presented in Section 3, and the target map is computed in order to make these boundary vertices coincident.

We also need to define the path following the same direction for both borders, i.e. the second boundary vertex placed on both maps must also correspond, otherwise the morphing process will result in shrinking the object to a line and growing it again in the opposite direction,as shown in the top row of Figures 1, 5, 7 and 8.

Since only two such paths exist for the target border (each boundary vertex forms indeed an edge with only two other boundary vertices), we can check both of them and compute a convolution between each possible target border and the source one. The best match will result in the greatest convolution value.

Let n_{border_a} and n_{border_b} be the number of boundary vertices in the source and target respectively. Let v_i^a be a boundary vertex of the source and let h_i^a be its corresponding position on the harmonic map. For each h_i^a, we need to add a corresponding vertex h_i^b on the target harmonic map, unless an existing vertex of the target map already is at the same position. Such a vertex is found by intersecting the segment s_i between the origin and h_i^a with each edge of the target harmonic map. The new vertex h_i^b is the intersection point and its 3D position v_i^b can be computed using the corresponding edge in the 3D object. The same idea is applied to add vertices on the source harmonic map corresponding to boundary vertices of the target map. This method is presented in Figure 3.

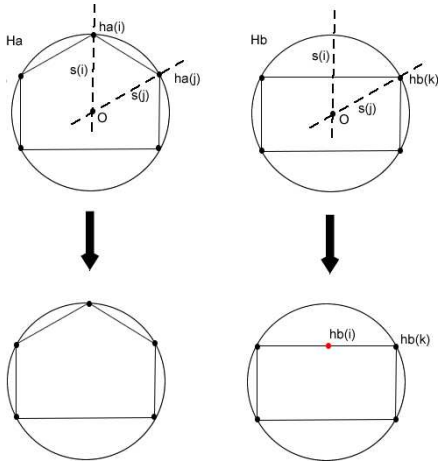

Figure 3: *In map $\mathcal{H}^a$, segment s_i between the origin and h_i^a intersects an edge of $\mathcal{H}^b$. Vertex h^b is then added on this edge. On the other hand, the intersection of segment s_j with an edge of $\mathcal{H}^b$ coincides with an existing vertex h_k^b, no vertex is added to $\mathcal{H}^b$.*

We now have the same number n_{border} of boundary vertices in each map, where

$$\max(n_{border_a}, n_{border_b}) \leq n_{border} \leq n_{border_a} + n_{border_b}$$

Let v_i^a be a boundary vertex of the source and let v_i^b be its corresponding vertex on the target border. We use the term *corresponding* to refer to the position of the vertex on the border and not to the correspondence given by the ICP algorithm. The 3D convolution is then computed as follows:

$$C = \frac{1}{n_{border}} \sum_{i=1}^{n_{border}} v_i^a . v_i^b \tag{8}$$

The direction that gives the maximum convolution is the one chosen. The interior points of the maps are then computed using the basic method of Section 3.

4.3. Curvature Adjustment

The ICP algorithm described in Section 4.1 gives a rough correspondence of the vertices, dealing with a poor interpolation in mismatched places such as the nose, ears, eyes and mouth as shown in the second row of Figures 1 and 5 for two different faces. A local refinement criterion has to be defined for such regions.

Alexa [Ale00] has shown that, given a small set of fixed correspondence points selected by the user in an interactive tool, the parameterization could be deformed in order to respect such correspondence. Here we show how the curvature

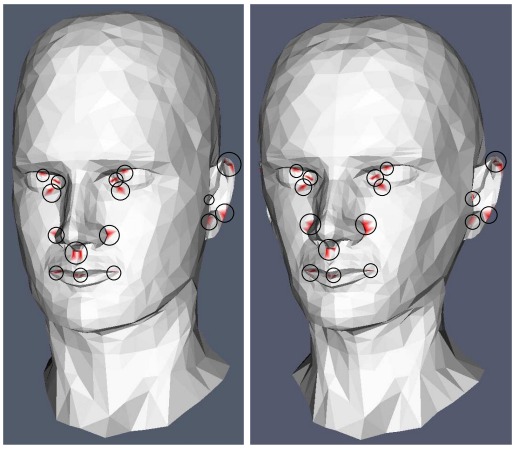

Figure 4: *Source and target feature selected points by the minima of the Gaussian curvature.*

can be used in order to automate such selection if a good initial correspondence is given (by the ICP algorithm).

Curvature is the chosen feature since it is invariant to position and orientation, and is scale independent. This is important since for example two noses of different sizes may be wanted to be merged. First the *Gaussian* and *Mean* curvatures are computed by using the Gauss-Bonnet Scheme [SMS*03]. It can be seen from the geometrical properties of the meshes, as depicted in Figure 4, that by selecting the negative minima of the Gaussian curvature, at least one wanted feature point is detected per principal patch (nose, ears, mouth, etc). Given two embeddings of vertices V^a, V^b on the disk as well as two ordered sets of features F^a, F^b, where $F^a \subseteq V^a$, $F^b \subseteq V^b$ and $|F^a| = |F^b| = n_f$. We denote the i-th element of V^K as v_i^K and the i-th element of F^K as f_i^K, with $K = \{a, b\}$. The following algorithm defines how to get the automatic match:

1. Compute the Gaussian C_g and Mean C_m curvatures in the source and target meshes
2. Obtain the local minima of the negative Gaussian curvatures of the source mesh that are smaller than a given threshold c
$$A = \{v_i^a;\ C_g(v_i^a) < c\}$$
3. For each vertex in the source object selected by step 2, find the best match in a region of the target, defined by a small disk $S^2(r)$ of radius r in the harmonic map space proportional to the inverse of the density of the points in such region, with respect to the following criteria:
$$\forall v_i^a \in A; \quad \text{if } E < m: \ f_k^a = v_i^a, \ f_k^b = v_j^b$$
$$\mathrm{E} = \min_{h_j^b \in S^2(r)} \left\{ \|C_g(v_i^a) - C_g(v_j^b)\| + \|C_m(v_i^a) - C_m(v_j^b)\| \right\}$$

where m is a constant value, and f_k^a, f_k^b are the k-th matching features. The radius is taken to be a function of the density of the points in order to take into account the influence of the deformation of the polygons in the harmonic map.

Once these correspondences have been established, the parameterization is deformed to align the features by an adapted version of Alexa's algorithm [Ale00] to planar projections. Since Alexa projected his meshes on spheres, he had to ensure that the vertices remained on the sphere. In our case, we just need to ensure that the vertices won't go further than the border of the unit disk. The algorithm is then adapted in the following way:

1. Rotate the first harmonic map using the transformation given by ICP
2. $\forall i,\ 1 \le i \le n_f$, compute the desired position w_i as the average of corresponding vertices:
$$w_i = \frac{f_i^a + f_i^b}{2}$$
3. $\forall i,\ 1 \le i \le n_f$, define the displacement $t_i = w_i - f_i^K$ necessary to move the i-th feature. Then apply the following map to all vertices:
$$\forall j \ \ v_j^K = \begin{cases} v_j^K + t_i(d - \|v_j^K - f_i^K\|) & \text{if } \|v_j^K - f_i^K\| < d \\ v_j^K & \text{otherwise} \end{cases}$$
Check after each mapping if the embedding is still valid. If not, decrease the length of t_i
4. Repeat step 3 until the positions of the vertices remain unchanged after a complete loop
5. If the features vertices are not in their desired positions and $d > 0$, decrease d and go back to step 3

4.4. The Merging Algorithm

The main drawback of Kent et al's method [KCP92], presented in section 3, is that it introduces noise in the shape of the meshes. The new vertices are placed on the facets and therefore force them to be flat, while their real position should be some distance above or below the facet.

Krabacher et al. [KSH01] presented a method to interpolate the position of a vertex v given its barycentric coordinates $\mathbf{b} = (b_1, b_2, b_3)$. The 3D coordinates are computed as follows:

$$v = \sum_{i=1}^{3} b_i (v_i + \mathbf{n}\delta_i) \tag{9}$$

where v_i is a vertex of the facet containing the new vertex, $\mathbf{n}$ is the vertex normal of v and δ_i is given by:

$$\delta_i = d_i \cos\beta_i + d_i \sqrt{\frac{(1 - \cos^2\beta_i)(1 - \cos\phi_i)}{1 + \cos\phi_i}} \tag{10}$$

$$\cos\phi_i \simeq \mathbf{n} \cdot \mathbf{n}_i \qquad \cos\beta_i \simeq \frac{\mathbf{n} \cdot \mathbf{d}_i}{d_i} \tag{11}$$

where $\mathbf{n}_i$ is the vertex normal of v_i, $\mathbf{d}_i$ is the vector between

v_i and the projection of the new vertex on the facet and d_i is the norm of $\mathbf{d}_i$.

Merging the two harmonic maps allows us to obtain the same number of vertices in the new source model and the new target model, i.e. $N_a + N_b + I_{tot}$. Then a new triangulation for these models is built with the method described in Section 3, and the system is ready to compute the intermediate shapes.

5. Interpolation

Once the correspondences of the source and target points have been established, the intermediate shapes at various time steps of the transformation going from the source object to the target object can be calculated.

The basic algorithm is to establish a linear interpolation between the position of the correspondent vertices:

$$V(\alpha) = (1-\alpha)V^a + \alpha V^b, \qquad \alpha \in [0,1] \tag{12}$$

where V^a, V^b are correspondent vertices in source and target meshes. Even if the correspondences are well established the morphing is not realistic since, by linearizing a rotation and a translation, the object loses its proportions, as is shown in the case of the faces in the third row of Figure 1.

Alexa [ACOL00] has proposed a linear interpolation of the $\mathcal{R}t$ transformation between the source and the target meshes.

$$V(\alpha) = (1-\alpha)\mathcal{R}tV^a + \alpha V^b, \qquad \alpha \in [0,1] \tag{13}$$

It is well known in the graphics community, that the smoothest interpolation is the Slerp quaternion interpolation. Since the $\mathcal{R}t$ transformation is recovered by the ICP algorithm of Section 4.1, an intermediate mesh can be computed by applying a linear interpolation for the translation, a Slerp in quaternion space for the rotation and a linear interpolation of the remaining deformation between the source vertex after alignment and the target vertex.

$$V(\alpha) = \mathcal{R}_s(1-\alpha)tV^a + \alpha V^b, \qquad \alpha \in [0,1] \tag{14}$$

where $\mathcal{R}_s$ is the rotation matrix corresponding to the quaternion defined as:

$$q_s = Slerp(q_n, q, (1-\alpha)) \tag{15}$$

where q_n is the quaternion of null rotation, and q is the quaternion representing the rotation obtained by the ICP algorithm.

In the following section, morphing results for several meshes are discussed.

6. Results

Figure 5 depicts the results of interpolating two faces with different topologies. The first row shows state of the art techniques without using interaction. The second row depicts the results using the registration techniques of Section 4.1. Local deformation due to small mismatches produces unpleasant artifacts around the eyes, mouth, nose, etc. By applying the curvature alignment of Section 4.3 such artifacts disappeared, and the intermediate shapes still look like faces, as shown in the bottom row.

Existing methods need the source and target meshes to be registered, that means to be oriented in the same manner and centered at the same position, as those shown in Figure 5. The method proposed in this paper uses an improved registration algorithm to compute the transformation between the source and the target. The top row of Figure 1 shows the results using Kanai's method without user interaction. By using the result of the ICP algorithm, some artifacts remain visible in the intermediate shapes, as depicted in the second row. By using the curvature adjustment of Section 4.3 such artifacts disappear. Using linear interpolation results in an object that loses its proportions while morphing, as depicted in the third row. The quaternion Slerp method of Section 5 solves such problems, as depicted in the bottom row of Figure 1.

Results of morphing the asian face of Figure 2 and a caucasian face are shown in Figure 6. Notice that the results obtained by using the registration, curvature alignment, and quaternion slerp interpolation of Sections 4 and 5 do not present any unpleasant local artifact.

While made in theory for open 3D objects, the method of Section 3 can be extended to closed objects, as shown in Figure 7 for an ellipse and a star. Both meshes are cut automatically by first registering them and cutting them by a plane passing through the best match and the center of mass. If the connectivity of at least one half is not preserved, the cutting has to be done again with a new plane passing through the following best match and the center of mass. The border is then smoothed by adding vertices at the intersections between the plane and the edges. The algorithm of Section 4 is then applied to each half separately, merging the results at the end. The right shape of the objects is maintained since both halves of the same object shared the same border. The top row of Figure 7 depicts the results using the method of Section 3 without any manual interaction, while the bottom part shows the results automatically generated by the algorithm of Section 4, using ICP and linear interpolation. To obtain similar results with Kanai's method, one would have to manually click on border points in the source and target meshes, as well as manually define the cutting plane.

Interpolation results between an ellipse and a face are shown in Figure 8. The first row shows the existing techniques used without user interaction while the bottom depicts the results of the automatic method using registration and a linear interpolation.

A video is given as additional material for each of the figures shown in the paper.

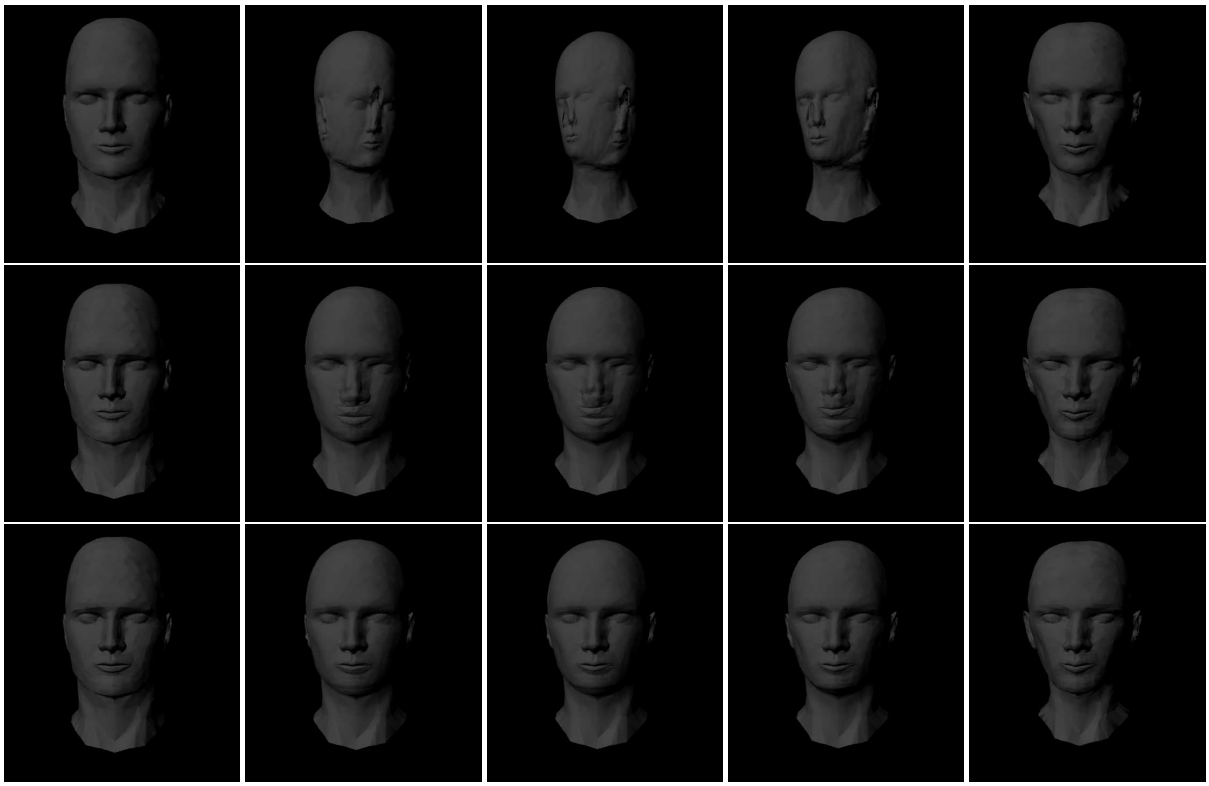

Figure 5: *Interpolating the two face meshes of Figure 4.* ***Top row****: Basic method of Section 3 without manual interaction.* ***Second row****: Results of the registration followed by a linear interpolation. Some clicking would be needed by Kanai's method to generate similar results.* ***Bottom row****: Results using registration, curvature alignment and linear interpolation.*

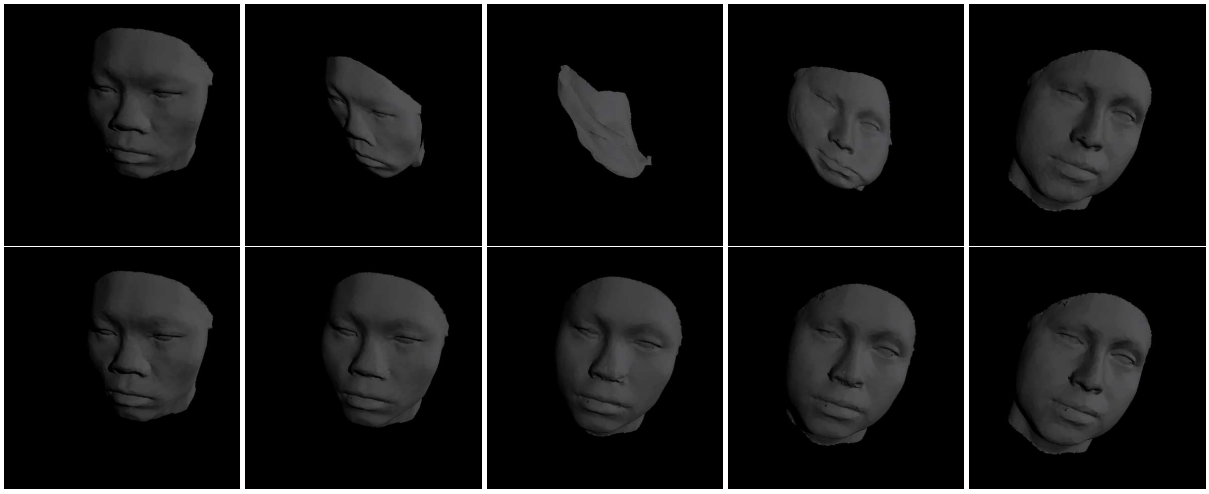

Figure 6: *Interpolating an asian and a caucasian face.* ***Top row****: Basic method of Section 3 without user interaction.* ***Bottom row****: Result using registration, curvature alignment and quaternion Slerp interpolation.*

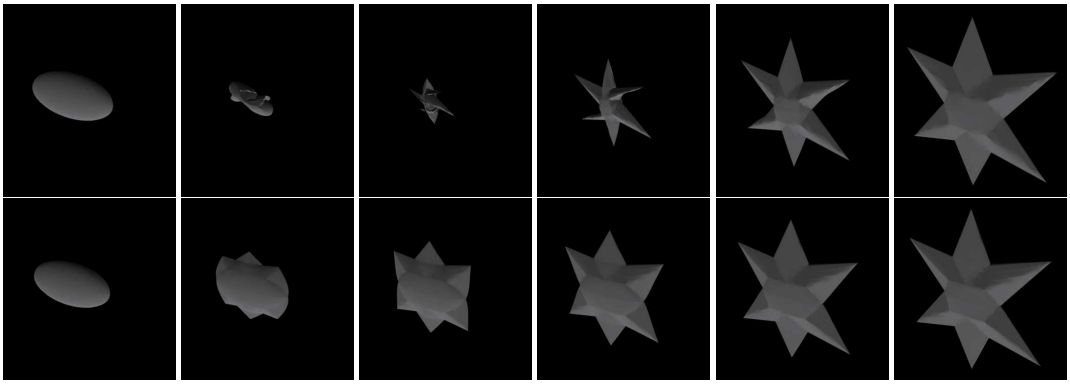

Figure 7: *Interpolating 3D closed objects: ellipse and star.* ***Top row****: Method of Section 3 without manual interaction.* ***Bottom row****: Results of the automatic interpolation method described in Section 4. The method of Section 3 would need clicking in a subset of points of the source and target borders, and manually specifying the cutting plane in order to obtain similar results.*

Figure 8: *Interpolating an ellipse and a face.* ***Top row****: Method of Section 3 without manual interaction.* ***Bottom row****: Results of the automatic interpolation method described in Section 4.*

7. Conclusion

We have presented a fully automated algorithm that combines improved registration techniques with feature point detection to solve the correspondence problem, eliminating unpleasant local artifacts. The method proposed is general in the sense that, even if it is applied in this paper to planar projection, it can be used with other correspondence techniques, and will be a line of future research. Moreover, the registration result can be used to directly define a more realistic interpolation technique than the linear one, where the object does not lose its proportions while interpolating.

The proposed algorithm has been tested with a wide range of topologically different meshes, representing open and closed objects, with very satisfactory results. The results remained visually pleasant even when using meshes where any local artifacts were allowed, as the case of morphing different faces.

References

[ACOL00] ALEXA M., COHEN-OR D., LEVIN D.: As-rigid-as-possible shape interpolation. In *Siggraph 2000, Computer Graphics Proceedings* (2000), Akeley K., (Ed.), ACM Press / ACM SIGGRAPH / Addison Wesley Longman, pp. 157–164. 3, 7

[Ale00] ALEXA M.: Merging polyhedral shapes with scattered features. *The Visual Computer 16*, 1 (2000), 26–37. 1, 3, 5, 6

[Ale02] ALEXA M.: Recent advances in mesh morph-

ing. *Computer Graphics Forum 21*, 2 (2002), 173–196. 1

[BBPV03] BLANZ V., BASSO C., POGGIO T., VETTER T.: Reanimating Faces in Images and Video. In *Eurographics* (Granada, Spain, September 2003). 2

[DDH04] DEWAELE G., DEVERNAY F., HORAUD R.: Hand motion from 3d point trajectories and a smooth surface model. In *European Conference on Computer Vision* (2004). 4, 5

[EDD*95] ECK M., DEROSE T., DUCHAMP T., HOPPE H., LOUNSBERY M., STUETZLE W.: Multiresolution analysis of arbitrary meshes. *Computer Graphics 29*, Annual Conference Series (1995), 173–182. 2, 3

[Flo97] FLOATER M. S.: Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design 14*, 4 (1997), 231–250. 2

[GP02] GRANGER S., PENNEC X.: *Multi-scale EM-ICP: A Fast and Robust Approach for Surface Registration*. Tech. rep., INRIA Sophia Antipolis, 2002. 4

[GSL*98] GREGORY A., STATE A., LIN M., MANOCHA D., LIVINGSTON M.: Feature-based surface decomposition for correspondence and morphing between polyhedra. In *Proceedings of the Computer Animation* (1998), IEEE Computer Society, p. 64. 2

[GSL*99] GREGORY A., STATE A., LIN M. C., MANOCHA D., LIVINGSTON M. A.: Interactive surface decomposition for polyhedral morphing. *The Visual Computer 15*, 9 (1999), 453–470. 2

[KCP92] KENT J. R., CARLSON W. E., PARENT R. E.: Shape transformation for polyhedral objects. *Computer Graphics, SIGGRAPH Proceedings 26*, 2 (1992), 47–54. 3, 6

[KSH01] KRABACHER S., SEEGER S., HÄUSLER G.: Refining triangle meshes by non-linear subdivision. In *3-D Digital Imaging and Modeling* (2001), pp. 270–277. 6

[KSK98] KANAI T., SUZUKI H., KIMURA F.: Three-dimensional geometric metamorphosis based on harmonic maps. *The Visual Computer 14*, 4 (1998), 166–176. 1, 3

[RCB97] RANGARAJAN A., CHUI H., BOOKSTEIN F. L.: The softassign procrustes matching algorithm. In *Information Processing in Medical Imaging* (1997), pp. 29–42. 4

[SE01] SURAZHSKY T., ELBER G.: Matching free form surfaces. *Computers & Graphics 25* (2001), 3–12. 2

[SMS*03] SURAZHSKY T., MAGID E., SOLDEA O., ELBER G., RIVLIN E.: A comparison of gaussian and mean curvatures estimation methods on triangular meshes. *IEEE International Conference on Robotics and Automation* (September 2003), 1021–1026. 6

[ST98] SHAPIRO A., TAL A.: Polyhedron realization for shape transformation. *The Visual Computer 14*, 8/9 (1998), 429–444. 1, 3

[Zha94] ZHANG Z.: Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision 13*, 2 (1994), 119–152. 4