Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

MALTE ZÖCKLER, DETLEV STALLING, HANS-CHRISTIAN HEGE

Fast and Intuitive Generation of Geometric Shape Transitions

Preprint SC 99-33 (September 1999)

Fast and Intuitive Generation of Geometric Shape Transitions

Malte Zöckler, Detlev Stalling, Hans-Christian Hege

September 1999

Abstract

We describe a novel method for continuously transforming two triangulated models of arbitrary topology into each other. Equal global topology for both objects is assumed, extensions for genus changes during metamorphosis are provided. The proposed method addresses the major challenge in 3D metamorphosis, namely specifying the morphing process intuitively with minimal user interaction and sufficient detail. Corresponding regions and point features are interactively identified. These regions are parametrized automatically and consistently, providing a basis for smooth interpolation. Utilizing suitable 3D interaction techniques a simple and intuitive control over the whole morphing process is offered.

Keywords: morphing, metamorphosis, surface parametrization

1 Introduction

Morphing or warping algorithms have received considerable attention in computer graphics and image processing. Morphing has become a standard technique in movie and entertainment industry. Although computer generated images which are rendered from true 3D models are common today, the majority of methods developed so far focuses on the problem of interpolating between 2D images.

For computer animation the interpolation of three dimensional models is an attractive alternative to 2D morphing. Using true 3D methods morphing sequences can be computed independently of e.g. light and camera positions. Also, in general more realistic results will be obtained, since shape information not visible in the start or end image can be taken into account.

Properties of a good morph. A morph defines the transition of an object A into another object B. This transformation should meet several criteria. First, it should be continuous and smooth, up to a discrete set of transition points where intended topology changes happen. Second, the intermediate objects should look "realistic" in some sense. Third, prominent features of object A should transform into features of B that correspond in some *semantic* sense. Especially to meet the latter criterium, *manual interaction* is required as a matter of principle.

Specific contributions. In this paper we present a method for transforming three-dimensional geometric models into each other. The models are supposed to be given in a polygonal boundary representation. Specifically, we assume that the surfaces are triangulated, but the vertex/edge/face structures do not need to be identical.

Our algorithm allows the user to identify *corresponding regions* as well as *corresponding points* in both models interactively. During this process both models are decomposed into a set of topologically equivalent patches, which afterwards are parametrized automatically. In this way a complete one-to-one correspondence between both models is achieved. In contrast to other approaches individual patches can be of disk-like as well as cylinder-like topology. The whole method is characterized by the following items:

- feature correspondences can be defined in an intuitive and very flexible way
- no restrictions of any type apply to the definition of corresponding point features
- time requirements for user interaction are low compared to other approaches
- the method is very fast; morph sequences are computed within a few seconds.

Although we primarily assume that the models to be morphed have the same topology, we also discuss methods for taking into account topology changes of various kinds.

The major design goal was to develop and combine algorithmic components in such a way that all steps, including the parametrization, are completely controlled by the animator's aims. Using suitable 3D interaction techniques manual input is greatly facilitated. The implementation proves that our method offers a simple and intuitive control over the whole morphing process and enables the animator to create morphing sequences in amazingly short time.

1.1 Previous work

The type of object representation has a strong impact on algorithms for object transformation. The major categories employed in computer graphics are volume and surface based representations. Correspondingly, existing morphing techniques for 3D objects can be divided into two major classes: *volume based methods* that interpolate two volumetric representations of the objects, often by using some kind of transformation function which is defined continously in $\mathbb{R}^3$, and *surface based methods* that first establish and then evaluate a correspondence function defined on the models' boundary representations.

Volume based methods [22, 12, 11, 3, 5] offer the advantage that topology does not matter. Furthermore, no surface models are needed whose generation, e.g. from segmented image volumes, might be costly. Drawbacks are that the morphing sequences are often expensive to compute, and that topological aspects like connectivity of intermediate models are hard to control.

Surface based methods usually consist of two steps [15]: first, establishing *correspondences* by assigning to each point of the source surface a point on the target surface, then *interpolating* between each pair of corresponding points.

Different approaches have been taken for establishing correspondences, see [19]. If the topology of both objects is the same, the correspondence problem can be solved using parametrizations, i.e. continuous bijective functions that map both objects to a standard domain. Points whose images coincide under these mappings are said to correspond.

A lot of work has been published in the field of surface parametrization, especially for polygonal or simplicial surfaces. In the computer graphics community parametrizations are used for instance for texture mapping [24, 23]. Kent et al. [16] introduced parametrizations for solving the correspondence problem in 3D morphing. They created parametrizations by projecting star-shaped objects to spheres and extended the method for some other classes of genus 0 objects [15]. An extension for objects that are star-shaped around an axis was invented by Lazarus and Verroust [17, 18]. Shapiro and Tal [25] describe how to map a general genus 0 object onto a convex polyhdron.

Kanai et al. [14] present a method for parametrizing objects homeomorphic to a disk or a sphere by cutting them into two sheets. Bao and Peng [2] propose a checkerboard-like decomposition into rectangular sub-regions to parametrize objects with arbitrary topology. Gregory et al. [10] apply a user-specified control mesh to decompose the surface into a large number of disk-like patches. However, this method requires heavy user interaction-times. Recently, a similar approach was presented by Kanai et al. [13]. Their approach offers more flexibility for parametrizing the individual disk-like patches.

Floater [7] discusses the use of graph theoretical methods to parametrize triangulations. Parametrization has also been used in the context of multiresolution representations of polygonal surfaces [6, 4, 21]. These methods handle objects with arbitrary topology and procedurally construct a mapping between the model and some polyhedral base domain. Unfortunately, a basic limitation impedes their use in shape morphing: the polyhedral domains are computed automatically and are not guaranteed to be the same for two different objects, even if these objects have the same topology. Therefore the correspondence problem between the original models is only replaced by a correspondence problem of the – possibly simpler – base domains. Lee et al. [20] try to achieve such a correspondence between coarse base domains by projecting points from one base domain onto the second one and applying a relaxation scheme. However, this relaxation scheme is not guaranteed to produce consistent, i.e. fold-over free, mappings. User interaction is required to manually fix these problems.

The interpolation issue has been adressed by only few authors up to now. Most algorithms use linear interpolation between corresponding points. Gregory et al. [10] transform points according to user-defined Bezier splines. Kent et al. [15] suggest to use Hermite splines, defined by the corresponding points and the surface normals normals.

More details about algorithmic variants developed up to now can be found in a survey article by Lazarus and Verroust that presents the current state in three-dimensional morphing [19].

1.2 Overview

The new surface based shape morphing method basically consists of the following steps (c.f. Figure 1):

1. **Definition of Correspondences.** The user defines corresponding regions (*patches*) on the boundaries of the two models. This is done interactively by drawing onto the surfaces using the mouse (2 to 14 patches in the examples shown in this paper). Within the patches an arbitrary set of optional feature points may be specified.
2. **Parametrization.** Then a global parametrization for each of the two models is computed by parametrizing corresponding regions individually. Extra care is taken to preserve continuity across the patch borders.
3. **Matching Features Points.** Additional feature points are matched in parameter space by modifying the parametrizations using a foldover-free warping technique.
4. **Interpolation.** The original meshes are merged in the parameter domain to construct a supergrid suitable for morphing. Corresponding points in both input models are computed for each vertex of this grid. These two positions are interpolated to produce the final morph.

The steps 1, 2, 3, 4 are described in sections 2, 3, 4, 5, respectively. Section 6 presents morphing sequences and discusses strengths and weaknesses of the

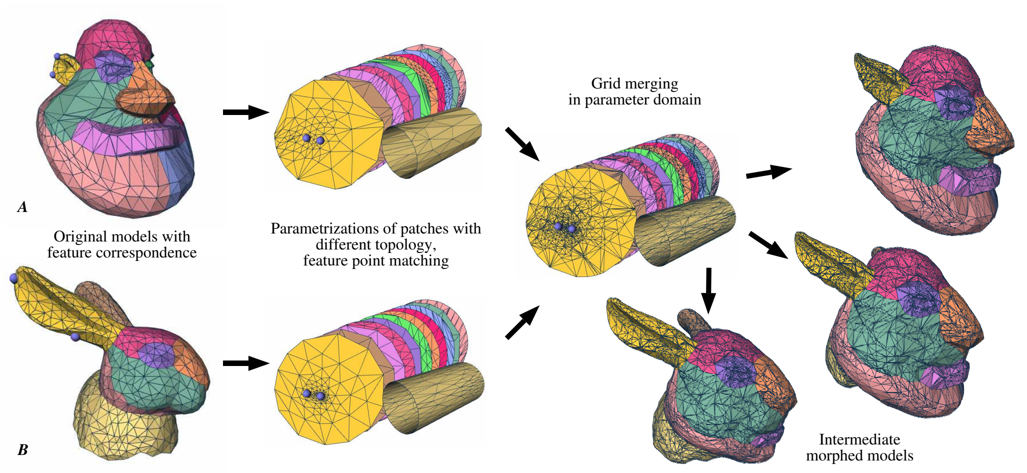

Figure 1: The different stages of our morphing method: First, the designer defines the correspondence between the models by drawing feature regions and feature points. The algorithm then computes a patch-wise parametrization, taking into account feature points on the border as well as within the regions. From the parametrized meshes a supergrid is constructed which then is used during interpolation.

presented method, as well as possible extensions.

Independently, similar approaches to morphing have been recently developed by Gregory et al. [10] and Kanai et al. [14]. They also decompose both models into topologically equivalent sets of patches and parametrize the patches individually. In contrast to their procedures, our algorithm can not only deal with disk-like patches, but also with patches of cylindrical topology. We also use a different kind of parametrization and can handle topology changes. The most important difference is the possibility to specify an arbitrary number of feature points within the individual patches. This way an accurate match can be achieved with only few patches.

1.3 Notation and Preliminaries

To facilitate understanding, let us shortly fix the notation and clarify some mathematical basics. The original triangulated geometrical objects are denoted by A and B. Patches of object A and B are denoted by P_i^A and P_i^B, where i is the patch index. $\mathcal{T}(A)$ marks the triangulation of object A, i.e. a set containing sets of vertices, edges and faces. Points contained in the triangulation are called vertices $\boldsymbol{V}$, in contrast to arbitrary surface points $\boldsymbol{X}$. Usually, points carry an object and point index, like $\boldsymbol{V}_i^A$ or $\boldsymbol{X}_j^B$.

A parametrization of a surface A is a continous bijective mapping $\mathcal{P}_A : A \longrightarrow$

$\mathbb{D}$ onto some base domain $\mathbb{D}$. We denote the points in the parameter domain by lowercase letters: $\boldsymbol{v}_i^A = \mathcal{P}_A(\boldsymbol{V}_i^A)$.

If two surface objects A and B are mapped onto the same domain, a one-to-one correspondence between these is defined implicitely: Let $\mathcal{P}_A : A \longrightarrow \mathbb{D}$ and $\mathcal{P}_B : B \longrightarrow \mathbb{D}$ be parametrizations of surfaces A and B with the same parameter domain $\mathbb{D}$. Then a point $\boldsymbol{X} \in A$ corresponds to $\boldsymbol{X}' = \mathcal{P}_B^{-1}(\mathcal{P}_A(\boldsymbol{X})),\ \boldsymbol{X}' \in B$.

The mapping $\mathcal{P}_B^{-1} \circ \mathcal{P}_A$, called *correspondence function*, is a one-to-one mapping between the objects A and B.

2 Feature Correspondence

Probably the most important step in a morphing framework is the definition of correspondences between the two models. Many existing morphing algorithms try to solve the correspondence problem automatically. However, when semantic information should be taken into account during the morph, fully automatic approaches inevitably must fail. Therefore, the user should have the ability to identify corresponding features interactively. This procedure should be as intuitive as possible, without restricting the designer by algorithmic needs.

In addition, the system should be tolerant about how much detail is specified manually. This means that constraints set by the user should be obeyed, but regions where no further correspondence has been specified should be transformed automatically.

2.1 Patches and Feature Points

In our system the designer defines correspondences interactively by drawing corresponding regions and points in both models. The number and arrangement of patches can be completely controlled by the user, as long as the topological structure of the patch sets is chosen to be the same. In particular, the number of neighbors of a patch is not limited. The topological equivalence is achieved in a natural way if like in our method the patches carry semantic information.

Currently, our system supports patches with the topology of a disk (1 boundary curve) and with the topology of a cylinder (2 boundary curves). Inside the patches an arbitrary number of additional feature points can be set. During the morph these points will be exactly mapped onto each other.

Correspondence information for each patch is computed automatically, taking into account the feature points. In this way, the system offers the opportunity to control feature correspondences with a high degree of detail, if necessary. On the other hand, if such detailed definition of correspondence is not required only few patches need to be defined.

2.2 User Interaction

In order to ease the definition of corresponding regions both models are displayed simultaneously on the screen using two different 3D viewers. Optionally, the viewers can be coupled such that always the same camera orientation is used in both windows. Region boundaries can be edited by clicking on surface points using the mouse. The points are connected in real-time by computing the shortest path along the edges of the triangulation. These paths represent the boundaries between patches. The patches themselves are found by a simple flood fill algorithm. Patches have to be named identically in both objects to define a correspondence.

Feature points inside a patch can also be edited using the mouse. Boundary vertices where multiple regions join are interpreted as feature points automatically. Whenever no feature points on the boundary occur, e.g. if a patch has no neighbors at all, the user is requested to select a pair of corresponding points on both contours in order to define the relative orientation. In a similar way the orientation of the boundaries of a cylindrical patch relative to each other must be fixed.

3 Parametrizing Patches

Once the models are decomposed into two topologically equivalent sets of patches, the problem of finding a global parametrization can be broken down to parametrizing the patches individually. In this section we describe how individual patches can be parametrized. We use so-called barycentric mapping, originally proposed in [26, 27], because it is a simple, fast, and robust technique.

More complex parametrization methods, e.g. harmonic maps [6] or shape preserving mapping [7] could also be used. However, the type of parametrization is considered not to be very important in morphing applications [19]. Note, that in our case the parametrization will be postprocessed anyway in order to match inner point features, compare Section 4.

In the following, we describe how border points are mapped into the parameter domain. Afterwards we discuss the parametrization of inner points using barycentric mapping. We then show how this technique can be extended to cylindrical patches.

3.1 Mapping the Borders

Barycentric mapping, as well as other methods like e.g. harmonic maps, allow to place border points with arbitrary spacings on the boundary of a convex polygon in the parameter domain. To achieve continuity and uniqueness across patch boundaries the border mapping has to fulfill certain requirements.

In general a border point is contained in at least two adjacent patches. If a patch is surrounded by more than one other patch, some of its border points are part of three or more patches. We call these points *branching points*. Additional pairs of corresponding border points defined by the user are treated in the same way as branching points. In Figure 2 the branching points of one particular patch are shown.

Since the patch configuration is the same in both models they both contain the same number of branching points. We want these points to be mapped onto each other. Therefore, the same parameter vectors have to be assigned to two branching points $\boldsymbol{T}_i^A$ and $\boldsymbol{T}_i^B$, i.e. $\boldsymbol{t}_i^A = \boldsymbol{t}_i^B$. It does not matter which value is actually chosen. For instance, one could place the branching points equidistantly on the border of the unit circle. To minimize distortions, we distribute them proportional to the averaged lengths of the border segments in the original 3D models. There is still a rotational degree of freedom which can be chosen arbitrarily.

One important problem one has to overcome when parametrizing patches individually is to assure continuity across patch boundaries. Since the parameter domains of different patches are completely independent, continuity of the parametrization across the boundary can not be defined. However, the correspondence function between the two models which is derived from their parametrizations can and must be continuous. Otherwise cracks or strong distortions at patch boundaries would occur during the shape transition.

A continuous correspondence function is obtained if a point $\boldsymbol{X}^A$ on the boundary between two patches P_1^A and P_2^A is mapped to the same point in B, irrespectively which patch parametrization is used, i.e. $\mathcal{P}_{B,1}^{-1}(\mathcal{P}_{A,1}(\boldsymbol{X}^A)) = \mathcal{P}_{B,2}^{-1}(\mathcal{P}_{A,2}(\boldsymbol{X}^A))$. Here for instance $\mathcal{P}_{A,1}$ denotes the map of object A's patch 1 to $\mathbb{D}$. While branching points are forced to be mapped uniquely, a unique correspondence for border points between adjacent branching points is achieved by mapping them proportional to the arc length in the particular border segment of the original model. This assures consistency because each border segment is shared by the two adjacent patches the points are contained in.

3.2 Barycentric Mapping

Let P be a patch of a triangulated model. We denote its vertices by $\boldsymbol{V}_i$, with $i = 1, ..., N_V$, and its set of edges by $\mathcal{E}$. A parametrization can be defined by assigning parameters $\boldsymbol{v}_i \in \mathbb{D}$ to each vertex and interpolating them linearly within the triangles. This defines a mapping of the triangulation to the domain $\mathbb{D} \subset \mathbb{R}^2$. To construct a homeomorphism the $\boldsymbol{v}$ have to be chosen in such a way that the mapped edges $E_j \in \mathcal{E}$ do not intersect each other in $\mathbb{D}$, otherwise the mapping would not be bijective. One technique to compute the $\boldsymbol{v}_i$ in such a way is called

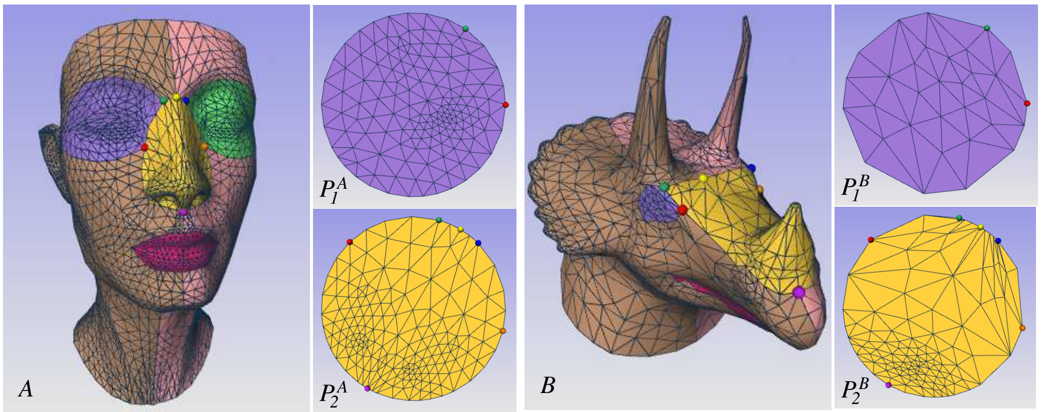

Figure 2: The figure shows the patch decomposition of two models and the *branching points* of one of the patches (the nose). Corresponding branching points in the two models have the same color. The parametrization of two of the six patches is shown for each of the models, namely the nose in yellow and the left eye in blue. Note, that corresponding border points of corresponding patches are forced to get the same parametric coordinates. It is not required that a branching point has the same coordinates in the parametrizations of all patches it belongs to. In the example the border length between the red and green point as well as their positions is different in P_1 and P_2. The blue and the yellow disk are two independent parts of the parameter domain. (See color plates in appendix.)

barycentric mapping.

Without loss of generality let $\boldsymbol{V}_1, ..., \boldsymbol{V}_{N_b}$ be the points on the boundary of patch P and $\boldsymbol{V}_{N_b+1}, ..., \boldsymbol{V}_{N_V}$ the inner points of the patch. We start by placing the parametric coordinates $\boldsymbol{v}_1, ..., \boldsymbol{v}_{N_b}$ of the border points $\boldsymbol{V}_1, ..., \boldsymbol{V}_{N_b}$ as described in the previous section.

Now we demand that each inner vertex in the parameter domain lies in the barycenter of its direct neighbors, i.e.

$$\boldsymbol{v}_i = \sum_{j=1}^{N_V} \lambda_{i,j} \boldsymbol{v}_j, \quad i = N_b + 1, \ldots, N_V \tag{1}$$

$$\lambda_{i,j} := \begin{cases} (\text{number of neighbors of vertex i})^{-1} & ; \quad (i,j) \in \mathcal{E} \\ 0 & ; \quad \text{else} \end{cases}$$

It is easy to show that the linear system (1) is not singular, so that it has a unique solution (see [7]). Moreover, Tutte [27] has shown that the mapping according to Equation (1) results in a consistent triangulation, i.e. no self intersections occur, provided the border points are mapped onto a convex polygon.

To actually compute the $\boldsymbol{v}_i$ any standard solver can be used. Since the number of neighbors of each point is small compared to the total number of inner points, the coefficient matrix $\{\lambda_{ij}\}$ is sparse. Therefore, iteration methods are well suited. We have used a simple SOR-scheme with $\omega = 0.9$ (see e.g. [9]):

```
let v = (0,0) for all inner points
define v_i for all border points
while not converged do
    for each inner point i do
        let v_i := (1-ω) v_i + ω Σ_{j=1}^{N_V} λ_{i,j} v_j
```

Floater [7] has shown that more sophisticated settings of the $\lambda_{i,j}$ can be used as well.

3.3 Cylindrical Patches

Although any object can be decomposed into patches of disk topology it is sometimes more convenient to use patches with other topologies as well. Therefore we have extended the barycentric mapping method to patches with cylindrical topology, using periodic boundary conditions.

A cylinder can be thought of as a rectangle with periodic boundary conditions in x-direction. Due to the periodic boundary conditions, equation (1) does not any longer represent a simple system of linear equations. However, it is still linear up to certain multiples of 2π. Therefore the iteration method proposed in Section 3.2 works if suitable initial conditions are chosen.

We start by artificially cutting the cylinder along an arbitrary line connecting its two borders. The resulting patch is then mapped onto a rectangle in the parameter domain defined by the points $(0,0)$, $(2\pi,0)$, $(0,1)$, and $(2\pi,1)$. Within this rectangle we again apply barycentric mapping. If the cylinder is merged again, a consistent parametrization is obtained. Due to the mapping of the cutting line artificial distortions are introduced which are undesired. Therefore the same iteration is used, this time taking into account periodic boundary conditions:

```
let v_i ≡ (u_i, v_i) = (0,0) for all inner points
define v_i ≡ (u_i, v_i) for all border points
while not converged do
    for each inner point i do
        for each neighbor j of i do
            if u_j - u_i > π then
                let u_j := u_j - 2π
            if u_j - u_i < -π then
                let u_j := u_j + 2π
        let v_i := (1-ω) v_i + ω Σ_{j=1}^{N_V} λ_{i,j} v_j
```

Obviously, this straight-forward implementation requires $|u_i - u_j| < \pi$ for all edges. Theorectically, this condition could be violated if the patch contains very few triangles. However, in practice we never observed such a case.

4 Matching Feature Points

After all surface patches have been parametrized using the methods described in the previous section a complete one-to-one correspondence between the input models A and B is achieved, namely by overlaying the two parametrizations $\mathcal{P}_A$ and $\mathcal{P}_B$ for each pair of patches. Details of how the correspondence function is evaluated are discussed in Section 5.

The approach described in Section 3.1 ensures that corresponding points on the boundary of a patch automatically receive the same parameter values $\boldsymbol{v}$. However, apart from the boundaries automatic parametrization methods like barycentric mapping allow no further control about which points inside a patch are identified with each other. For example, in case of two face models it is not clear that the tip of the nose is matched, i.e., that it receives the same $\boldsymbol{v}$ coordinates in the parametrization of both objects. One approach to this problem would be to subdivide the model into smaller patches, thus introducing more constraints. However, this can result in very complex patch configurations boosting interaction times. Instead, it is much more convenient for the designer to define feature points which shall be transformed onto each other within a single patch.

We meet this requirement by matching an arbitrary number of feature points inside a patch in a postprocessing step. In the following we describe how this can be done very easily for a single pair of feature points inside a patch. A more general approach, which handles an arbitrary number of feature point pairs is described in section 4.2.

4.1 Matching One Feature Point

Let P_A and P_B be two corresponding patches with disk topology and with vertices $\boldsymbol{V}_i^A$ and $\boldsymbol{V}_i^B$ and parametric coordinates $\boldsymbol{v}_i^A$ and $\boldsymbol{v}_i^B$. Let $\boldsymbol{F}_A$ and $\boldsymbol{F}_B$ be two inner points in these patches, that shall be transformed onto each other. Therefore the parametrizations have to be modified in such a way that the parameter values $\boldsymbol{f}_A$ and $\boldsymbol{f}_B$ become equal. To achieve this we assign the new parameter $\boldsymbol{f}'_A = \boldsymbol{f}'_B = (\boldsymbol{f}_A + \boldsymbol{f}_B)/2$ to both feature points. The parameter values of the other (non-feature) points of patch P_A are transformed according to

$$\boldsymbol{v}' = \boldsymbol{f}'_A + \alpha\,(\boldsymbol{q} - \boldsymbol{f}'_A), \quad \alpha = \frac{|\boldsymbol{v} - \boldsymbol{f}_A|}{|\boldsymbol{q} - \boldsymbol{f}_A|}, \tag{2}$$

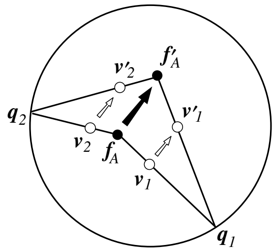

Figure 3: In order to move a single feature point $\boldsymbol{f}_A$ to a new position $\boldsymbol{f}'_A$ a simple geometric warping of the unit circle is applied.

where $\boldsymbol{q}$ denotes the intersection of a line through $\boldsymbol{f}_A$ and $\boldsymbol{v}$ with the unit circle, c.f. Figure 3. The parametrization of P_B is transformed in a similar way.

Note, that although Equation (2) defines a bijective mapping of the unit circle onto itself, our parametrization may become invalid after performing the 2D warp. This is because we only move the vertices of the parameter space triangulation, but still assume the edges of the triangles to be straight line segments afterwards. Therefore, theoretically foldovers may arise, especially if very large triangles are involved. Such triangles then must be subdivided until no more self-intersections occur. However, in practice we never detected foldovers in case of a single feature point.

4.2 Foldover-Free Warping in Parameter Space

If multiple feature points are to be matched no simple geometric transformation as in Equation (2) can be applied anymore. Instead, standard 2D image warping techniques could be applied. As mentioned above, a necessary requirement of such a warping is, that points on the border of the unit circle remain unchanged and that the mapping itself is bijective. One of the few 2D warping algorithms which guarantees bijectivity is *foldover-free image warping* introduced by Fujimura and Makarov [8]. We adapted a variant of this method to our purposes.

The main idea of the algorithm is to deform a coarse triangular warp mesh consisting of only feature points and some fixed points on the unit circle. While the feature points are moved towards the desired positions, the warp mesh is updated in order to prevent triangles from folding over. Every change in mesh connectivity defines a so-called *event*. In order to transform an arbitrary point $\boldsymbol{v}$ in the parameter domain, the triangle of the warp mesh containing $\boldsymbol{v}$ is determined and the barycentric coordinates of $\boldsymbol{v}$ with respect to this triangle are computed. Then the warp mesh is transformed until the first event occurs. An intermediate position of $\boldsymbol{v}$ is computed by applying the barycentric coordinates in the deformed warp mesh. After the event, $\boldsymbol{v}$ might belong to a different triangle. In this case

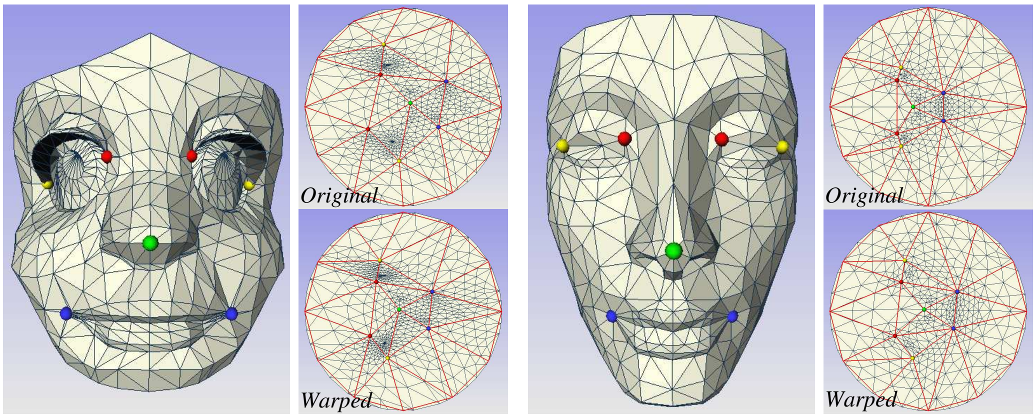

Figure 4: In order to match multiple feature points in a single patch two coarse triangular warp meshes are constructed. Together with the warp meshes both parametrizations are deformed until corresponding feature points have equal parameter values.

the barycentric coordinates with respect to the new triangle are computed. This process is repeated until the warp mesh has been fully transformed, i.e., until all feature points have reached their final positions. Boundary points as well as points falling in the tiny gap between the warp mesh and the circular border will remain fixed.

In order to minimize distortions again we move corresponding feature points linearly to the point halfway between them, i.e., $\boldsymbol{f}_A$ and $\boldsymbol{f}_B$ are mapped to $(\boldsymbol{f}_A + \boldsymbol{f}_B)/2$. The algorithm is illustrated in Figure 4. In this example seven feature points were set. No connectivity changes (events) had to be performed in order to match the features. Events are more likely to occur if more distant points are identified with each other. Like in [8] we use edge flips in order to prevent the warp mesh from folding over. An edge flip is performed whenever the maximum aspect ratio of the two adjacent triangles would be decreased by the flip. The aspect ratio of a triangle is defined by the radius of its circumcircle divided by the radius of its inner circle. To detect events points are translated according to a bisection strategy. The initial warp mesh is generated using a Delaunay algorithm.

As in the case of a single feature point, parameter space triangulations may become invalid although the warp itself defines a bijective mapping. This problem can occur especially if very distinct feature points are matched and thus the warp function becomes highly non-linear. Again, the solution is to subdivide the triangles until no more self-intersections occur.

5 Shape Transition

Once all patches of the two models A and B have been parametrized and all feature points have been matched the actual morph can be performed. In the following we describe how to compute a triangular model from A and B suitable for morphing and how to lookup the initial and final vertex positions for this model. We then discuss some interpolation issues, i.e., on which paths the vertices of the morph model are moved.

5.1 Grid Merging

After the parametrizations $\mathcal{P}_A$ and $\mathcal{P}_B$ have been computed, each vertex $\boldsymbol{V}_i^A$ of A can be moved to its corresponding position $\mathcal{P}_B^{-1}(\mathcal{P}_A(\boldsymbol{V}_i^A))$ on B. However, in general it is not a good idea to use one of the original meshes for morphing. Especially, if A contains (locally) less triangles than B, details of model B would be lost. As suggested in [15, 14] we therefore construct a new mesh M suitable for morphing by means of a grid intersection strategy. The triangulations of A and B are superposed in the parameter domain. Coincident vertices are identified and all inner edges that intersect are split by inserting a new vertex at the intersection point. Likewise, an edge is split if a vertex is lying on it. Boundary edges are treated in a special manner: In order to avoid that new boundary vertices occur due to intersecting boundary edges, the original boundary edges are replaced by new edges connecting the boundary vertices of A and B in consecutive order. To robustly detect coincident points and edge intersections a tolerance ϵ proportional to the smallest height in all triangles is used. The algorithm has been implemented efficiently using a quadtree data structure.

Finally, the whole domain is retriangulated by inserting additional edges where polygons with more than three vertices have been created. This produces a triangulation M suitable for performing the morph. The triangulation consists of a set of vertices $\boldsymbol{v}_i^M \subset \mathbb{D}$ in the parameter domain, as well as a set of triangles T_i^M connecting these vertices.

Theoretically, every edge of the first model could intersect every edge of the second one, resulting in a very large number of new vertices and triangles. In practice, though, there are much less intersections, as pointed out in [15]. Our experience shows that for two input meshes with approximately the same resolution the merged grid has about 3 to 5 five times as many triangles and vertices as the two models together.

5.2 Correspondence and Interpolation

For each vertex $\boldsymbol{v}^M$ we compute a corresponding position $\boldsymbol{X}_A = \mathcal{P}_A^{-1}(\boldsymbol{v}^M)$ in A as well as $\boldsymbol{X}_B = \mathcal{P}_B^{-1}(\boldsymbol{v}^M)$ in B. The inverse of the parametrization $\mathcal{P}_A^{-1}$ and $\mathcal{P}_B^{-1}$ is not given directly but has to be computed from the parameter values at the vertices. In order to compute $\boldsymbol{X}_A$ we have to find the triangle $\{\boldsymbol{v}_i^A, \boldsymbol{v}_j^A, \boldsymbol{v}_k^A\}$ in the parameter domain which contains $\boldsymbol{v}^M$. The corresponding point $\mathcal{P}_A^{-1}(\boldsymbol{v}^M)$ lies in the original triangle $\{\boldsymbol{V}_i^A, \boldsymbol{V}_j^A, \boldsymbol{V}_k^A\}$. Let (b_1^A, b_2^A, b_3^A) be the barycentric coordinates of $\boldsymbol{v}^M$ in $\{\boldsymbol{v}_i^A, \boldsymbol{v}_j^A, \boldsymbol{v}_k^A\}$. Then the exact position is

$$\mathcal{P}_A^{-1}(\boldsymbol{v}^M) = b_1^A \boldsymbol{V}_i^A + b_2^A \boldsymbol{V}_j^A + b_3^A \boldsymbol{V}_k^A. \tag{3}$$

The corresponding point in B is computed analogously. Note, that not only vertex positions can be interpolated according to Equation (3), but also other vertex attributes like colors or normals.

The morphing sequence can be generated most easily by linearly moving each vertex of the merged triangulation from its position in model A to the corresponding position in model B as a function of time t:

$$V_i^M(t) = (1-t)\,\mathcal{P}_A^{-1}(\boldsymbol{v}_i^M) + t\,\mathcal{P}_B^{-1}(\boldsymbol{v}_i^M). \tag{4}$$

In many cases this kind of linear interpolation produces satisfying results. In fact, all examples in this paper were generated using Equation (4). In some situations more sophisticated interpolations schemes are desirable, e.g., when the intermediate objects exhibit self-intersections or when complex physical deformations are to be imitated. As a simple alternative to linear interpolation other authors propose to move the vertices along cubic spline curves [10, 15]. Other interesting effects can be achieved by non-linearily varying the parameter t individually for each vertex [20].

5.3 Topology Changes

Up to now only the transition of topologically equivalent objects has been discussed. Nevertheless, our method can be extended to handle topology changes. To obtain convincing results user interaction is required to specify the way in which a topology change happens.

A first class of topology changes can be handled quite easily, namely the appearance of holes within the surface. For these problems, the user simply has to define a path where the model should rip. Artificial patch boundaries are then introduced, corresponding to the borders of the hole in the second surface. For example, Figure 6 (b), showing a morph between a cylinder-like model of Nefertiti and a disk-like model of a Triceratops, has been created in this way.

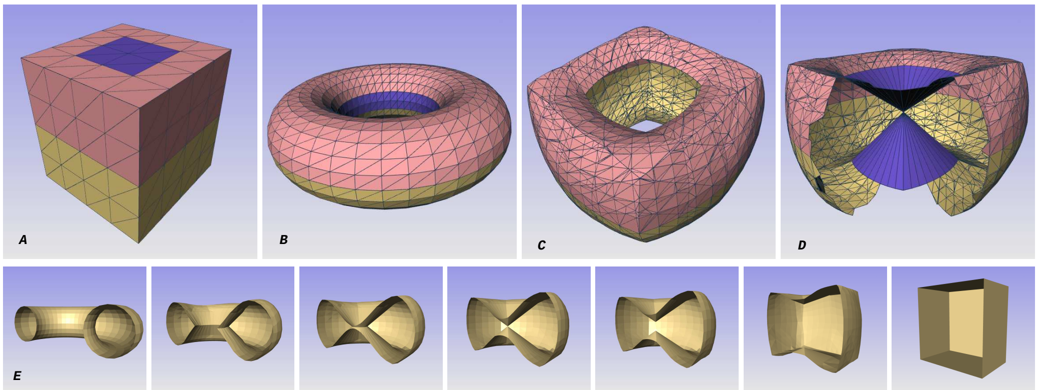

Figure 5: The figure shows the transition of a cube (A) into a torus (B). Two disk-like patches in the cube (shown in blue) transform into one cylindrical patch of the torus. The warp can be performed using an intermediate model (D). This is constructed by automatically editing a warped object constructed without the blue patches (C). In the lower row (E), the complete morphing sequence is shown.

An example for the more difficult class of topological transitions is the transformation of a genus 0 surface (sphere) into a genus 1 surface (torus). In this case two opposite disk-shaped patches have to join at one point, forming a double-cone. Once this happens, the two disk-shaped patches can also be interpreted as one cylindrical patch strangulated at the "equator". Therefore, the morphing can be performed as a two-step process. First, the initial surface is transformed into an intermediate model, the topology of which is ambiguous. Then this model is transformed into the second surface.

One way to obtain a suitable intermediate model is shown in Figure 5. The user selects the patches involved in the topology change, i.e. cylinder-like patch in one and two disk-like patches in the other model. The objects are warped neglecting these patches. Then an intermediate warped object is picked. In this model the boundaries of the missing patches are automatically retriangulated, forming a double-cone with user-defined center.

6 Results

The presented algorithm has been implemented within the visualization and modeling system Amira [1]. A large number of morphing sequences have been generated using the proposed method. The method has proven to be easy to use. The support of both, regional and point features, offers great flexibility for the designer. Some examples are depicted in Figure 6. Digital video clips of all morphs presented in the paper can be found on the internet at

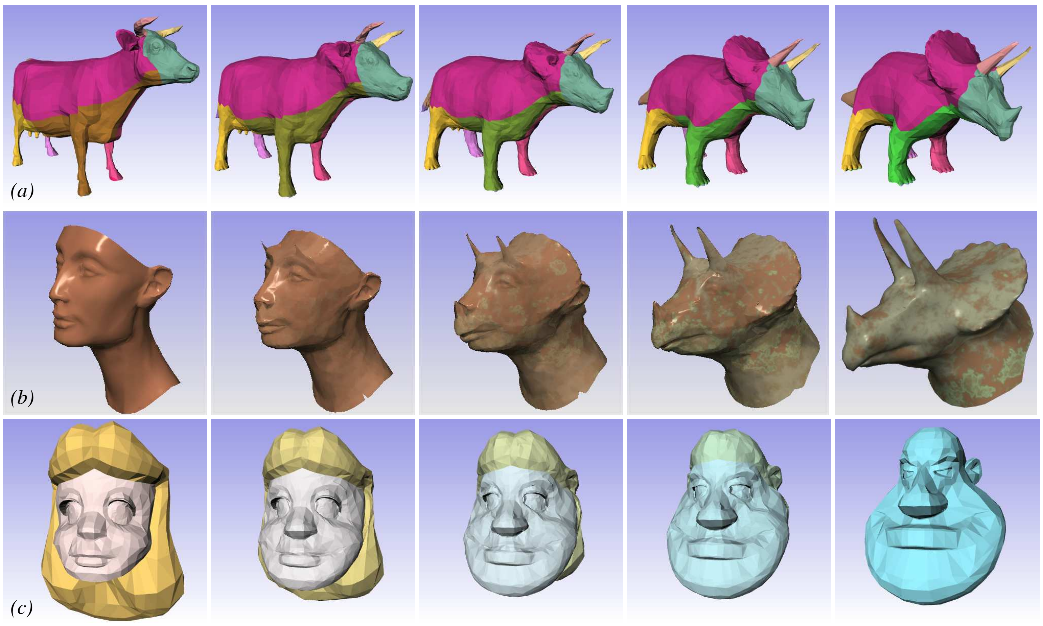

Figure 6: Different polygonal models morphed into each other. (See color plates in appendix.)

`http://www.zib.de/Visual/projects/morphing`. Note that quality and realism of a morph can best be judged in an animation.

Figure 6(a) shows the transition of a triceratops into a cow. 14 feature regions have been defined. For the face and the horns one additional feature point per patch has been used. Six regions have been defined for the morph between Nefertiti and the triceratops head shown in Figure 6(b). The patch decomposition is the same as in Figure 2. Finally, an example with only two corresponding regions is depicted in Figure 6(c). In this case prominent features of the face, like details of the eyes and the mouth, have been matched using 15 feature point pairs. If no natural decomposition into regions exists, feature points provide a powerful tool for fast and accurate definition of correspondence.

User interaction times ranged from 5 minutes for 6(c) up to 15 minutes for 6(a). The patch parametrizations, including feature point matching, can be computed very quickly (in less than 5 seconds for the most complex example with 11464 triangles in the original models). Grid merging as well as point location took less than 3 seconds each. The complete morphing process therefore can be performed in less than 10 seconds on an SGI O2 workstation with R10000 processor. Note that the intermediate models can be computed in real-time, once the other steps are done.

7 Conclusion and Future Work

We have presented a novel method for computing shape transitions between polygonal 3D objects. The method has proven to be a fast and feasible technique for 3D morphing. Compared to previous approaches user interaction times are significantly reduced by an intuitive scheme to define correspondences, including point features in addition to regional features or patches. The ability to define an arbitrary number of point features inside a single patch makes it possible to achieve high-quality morph sequences even with coarsely patchified input models. Interactive definition of sets of patches with equal topology becomes easy since individual patches have a semantic meaning. For example, they may correspond to the eyes, nose, or legs of a character.

There are a number of directions for future research. Like many other authors in this paper we concentrated on the correspondence problem of morphing. Definitely, more work is required in order to conceive better solutions for the interpolation problem, too. Also, the triangulation obtained by grid merging may contain too many triangles for certain real-time applications. Ideally, triangles should be added or removed dynamically so that the required degree of detail can be represented in each step of the morph sequence.

References

[1] Amira Visualization and Modeling System. `http://www.AmiraVis.com/`.

[2] BAO, H., AND PENG, Q. Interactive 3D morphing. *Computer Graphics Forum 17*, 3 (1998), C/23–C/30.

[3] BLOOMENTHAL, J., BAJAJ, C., BLINN, J., CANI-GASCUEL, M.-P., ROCKWOOD, A., WYVILL, B., AND WYVIL, G. *Introduction to Implicit Surfaces, chapter 8.* Morgan Kaufmann, 1997.

[4] CAMPAGNA, S., AND SEIDEL, H.-P. Parameterizing meshes with arbitrary topology. In *Image and Multidimensional Digital Signal Processing '98* (1998), B. G. H. Niemann, H.-P. Seidel, Ed., infix, pp. 287–290.

[5] COHEN-OR, D., SOLOMOVIC, A., AND LEVIN, D. Three-dimensional distance field metamorphosis. *ACM Transactions on Graphics 17*, 2 (Apr. 1998), 116–141.

[6] ECK, M., DEROSE, T., DUCHAMP, T., HOPPE, H., LOUNSBERY, M., AND STUETZLE, W. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH*

95 Conference Proceedings (Aug. 1995), R. Cook, Ed., Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 173–182.

[7] FLOATER, M. S. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design 14*, 3 (1997), 231–250.

[8] FUJIMURA, K., AND MAKAROV, M. Foldover-free image warping. *Graphical models and image processing: GMIP 60*, 2 (Mar. 1998), 100–111.

[9] GOLUB, G. H., AND LOAN, C. F. V. *Matrix Computations*. Johns Hopkins University Press, 1989.

[10] GREGORY, A., STATE, A., LIN, M., MANOCHA, D., AND LIVINGSTON, M. Feature-based surface decomposition for correspondence and morphing between polyhedra. In *Proc. Computer Animation '98, Philadelphia* (1998), IEEE CS Press, pp. 64–71.

[11] HE, T., WANG, S., AND KAUFMAN, A. Wavelet-based volume morphing. In *Proceedings of the Conference on Visualization '94* (Los Alamitos, CA, USA, Oct. 1994), R. D. Bergeron and A. E. Kaufman, Eds., IEEE Computer Society Press, pp. 85–92.

[12] HUGHES, J. F. Scheduled Fourier volume morphing. In *Computer Graphics (SIGGRAPH '92 Proceedings)* (July 1992), E. E. Catmull, Ed., vol. 26, pp. 43–46.

[13] KANAI, T., SUZUKI, H., AND KIMURA, F. Metamorphosis of arbitrary triangular meshes with user-specified correspondence. *IEEE Computer Graphics and Application (to appear)*.

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

[15] KENT, J. R., CARLSON, W. E., AND PARENT, R. E. Shape transformation for polyhedral objects. *Computer Graphics 26*, 2 (July 1992), 47–54.

[16] KENT, J. R., PARENT, R. E., AND CARLSON, W. E. Establishing correspondences by topological merging: a new approach to 3-D shape transformation. In *Graphics Interface '91* (1991), Canadian Information Processing Society, pp. 271–278.

[17] LAZARUS, F., AND VERROUST, A. Feature-based shape transformation for polyhedral objects. Eurographics association, pp. 1–14. The 5th Eurographics Workshop on Animation and Simulation.

[18] LAZARUS, F., AND VERROUST, A. Metamorphosis of cylinder-like objects. *The Journal of Visualization and Computer Animation 8*, 3 (July–Sept. 1997), 131–146.

[19] LAZARUS, F., AND VERROUST, A. Three-dimensional metamorphosis: a survey. *The Visual Computer 14*, 4 (1998), 373–389.

[20] LEE, A. W. F., DOBKIN, D., SWELDENS, W., AND SCHRÖDER, P. Multiresolution mesh morphing. *Computer Graphics 33*, Annual Conference Series (1999), 343–350.

[21] LEE, A. W. F., SWELDENS, W., SCHRÖDER, P., COWSAR, L., AND DOBKIN, D. MAPS: Multiresolution adaptive parameterization of surfaces. In *SIGGRAPH 98 Conference Proceedings* (July 1998), M. Cohen, Ed., Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 95–104.

[22] LERIOS, A., GARFINKLE, C. D., AND LEVOY, M. Feature-Based volume metamorphosis. In *SIGGRAPH 95 Conference Proceedings* (Aug. 1995), R. Cook, Ed., Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 449–456.

[23] LÉVY, B., AND MALLET, J. Non-distorted texture mapping for sheared triangulated meshes. In *SIGGRAPH 98 Conference Proceedings* (July 1998), M. Cohen, Ed., Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 343–352.

[24] MAILLOT, J., YAHIA, H., AND VERROUST, A. Interactive texture mapping. In *Computer Graphics (SIGGRAPH '93 Proceedings)* (Aug. 1993), J. T. Kajiya, Ed., vol. 27, pp. 27–34.

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

[26] TUTTE, W. T. Convex representations of graphs. *Proc. London Mathematical Society 10* (1960), 304–320.

[27] TUTTE, W. T. How to draw a graph. *Proc. London Mathematical Society 13* (1963), 743–768.