Volume xx (200y), Number 3 pp. 1–23

Recent Advances in Mesh Morphing

Marc Alexa

Interactive Graphics Systems Group, Department of Computer Science, Technische Universität Darmstadt
Rundeturmstr. 6, 64283 Darmstadt, Germany
phone: ++49 6151 155 674, fax: ++49 6151 155 669
email: alexa@gris.informatik.tu-darmstadt.de

Abstract

Meshes have become a widespread and popular representation of models in computer graphics. Morphing techniques aim at transforming a given source shape into a target shape. Morphing techniques have various applications ranging from special effects in television and movies to medical imaging and scientific visualization. Not surprisingly, morphing techniques for meshes have received a lot of interest lately.
Thiswork sums up recent developments in the area of mesh morphing. It presents a consistent framework to classify and compare various techniques approaching the same underlying problems from different angles.

1. Introduction

Morphing techniques transform one shape into another[23]. With the introduction in TV and movies, morphing is nowadays known to an audience beyond the computer graphics community. At the same time, morphing has estbalished itself as interesting research area. Recently, the focus is shifting from handling representations of space (images, volumes) to using explicit boundary representations, interpolating or blending the shape of the objects. This work concentrates on the popular piecewise linear boundary representations, namely meshes.

Blending shapes rather than the space they are embedded in can lead to better results but is also more involved, because a proper mapping between the shapes is needed. Defining such a mapping is not trivial for two main reasons. First, it requires a parametrization of the boundary representation, and second, the mapping might involve shapes with different topology.

Besides the parameterization problem, which is fundamental in many areas dealing with meshes, morphing also requires to find suitable paths for the elements of a mesh. This part has an aesthetic component, however, several reasonable conditions should be observed, i.e. the shape should not self intersect or collaps as it varies from source to target configuration.

Traditionally, morphing is applied to two shapes: a source and a target shape. Morphing among more than two shapes can be seen as generating elements in a space of shapes. This has interesting applications for modeling, animation, and analysis. Analysis using well-established methods such as the principal component analysis has gained interest lately.

This report exlains mainly techniques for morphing between two meshes. This avoids some clutter in the formalism. Once all methods are explained for two meshes, possible extensions to more than two meshes and their applications are discussed.

2. Terminology & Framework

Mesh morphing techniques involve computations on the geometry as well as the connectivity of meshes. For simplicity this report concentrates on triangle meshes. In the context of morphing it seems to be acceptable to triangulate polygonal meshes prior to the application of a morphing technique. To classify and understand mesh morphing techniques it is helpful to use the now widespread terminology from Spanier[64]. A mesh $\mathcal{M}$ is described by a pair (K, V), where K is a simplicial complex representing the connectivity of vertices, edges, and faces and $V = (\mathbf{v_1}, \ldots, \mathbf{v_n})$ describes the geometric positions of the vertices in $\mathbb{R}^d$, where typically $d = 3$.

The abstract complex K describes vertices, edges, and faces as $\{0, 1, 2\}$-simplices, that is, edges are pairs $\{i, j\}$, and faces are triples $\{i, j, k\}$ of vertices. The *topological realization* maps K to a simplicial complex $|K|$ in $\mathbb{R}^n$: The

vertices are identified with the canonical basis of $\mathbb{R}^n$ and each simplex $\mathbf{s} \in K$ is represented as the convex hull of the points $\{\mathbf{e_i}\} \in \mathbb{R}^n, i \in \mathbf{s}$ Thus, each 0-simplex is a point, each 1-simplex is a line segment, and each 2-simplex is a triangle in $\mathbb{R}^n$.

The *geometric realization* $\phi_V(|K|)$ is a linear map of the simplicial complex $|K|$ to $\mathbb{R}^d$, which is defined by associating the basis vectors $\mathbf{e_i} \in \mathbb{R}^n$ with the vertex positions $\mathbf{v_i} \in V$. The map ϕ_V is an *embedding* if ϕ_V is bijective. The importance of an embedding is that every point $\mathbf{p}$ on the mesh can be uniquely represented with a barycentric coordinate $\mathbf{b}$, i.e. $\mathbf{p} = \phi_V(\mathbf{b})$. Such barycentric coordinates have at most three non-zero components and specify the position of a point relative to a simplex. If the point is coincident with a vertex it is a canonical basis vector, if the point lies on an edge it has two non-zero components, otherwise it has three and lies on a face.

The neighborhood ring of a vertex $\{i\}$ is the set of adjacent vertices $\mathcal{N}(i) = \{j | i, j \in K\}$ and its star is the set of incident simplices $\mathcal{S}(i) = \bigcup_{i \in s, s \in K} s$.

In the classical setting of mesh morphing two meshes $\mathcal{M}_0 = (K_0, V_0)$ and $\mathcal{M}_1 = (K_1, V_1)$ are given. The goal is to generate a family of meshes $\mathcal{M}(t) = (K, V(t)), t \in [0, 1]$ so that the shape represented by the new connectivity together with the geometries $V(0)$ and $V(1)$ is identical with the original shapes, i.e. $\phi_{V(0)}(|K|) = \phi_{V_0}(|K_0|)$ and $\phi_{V(1)}(|K|) = \phi_{V_1}(|K_1|)$. Most of the time the paths $V(t)$ are required to be smooth. The generation of this family of shapes is typically done in three subsequent steps:

1. Finding a correspondence between the meshes. More specifically, computing coordinates W_0, W_1 that lie on the other mesh, i.e. $W_0 \in \phi_{V_1}(|K_1|)$ and $W_1 \in \phi_{V_0}(|K_0|)$. Each coordinate in W_0, W_1 is represented as a barycentric coordinate with respect to a simplex in the other mesh. Note that ϕ_{W_0} will not map $|K_0|$ to $\phi_{V_1}(|K_1|)$ (and vice versa), as only the vertices are mapped to the other mesh but not the edges and faces. Particularly important is the alignment of automatically detected or user specified features of the meshes.
2. Generating a new, consistent mesh connectivity K together with two geometric positions $V(0), V(1)$ for each vertex so that the shapes of the original meshes are reproduced. The traditional morphing approach to this problem is to create a superset of the simplicial complexes K_0 and K_1. However, remeshing techniques as used in multiresolution techniques are also attractive.
3. Creating paths $V(t), t \in]0, 1[$ for the vertices. While in general this is an aesthetic problem, several constraints seem reasonable to help in the design process. For example, in most applications the shape is not expected to collapse or self intersect and, generally, the paths are expected to be smooth.

In the following, recent work will be explained in terms of the above mentioned problem areas. This state of the art report focuses on mesh morphing, however, if believed to be instructive also techniques dealing with polygons are discussed.

3. Correspondence of shapes

In this section we aim at finding corresponding vertex positions on two or more shapes. Given two meshes $\mathcal{M}_0$ and $\mathcal{M}_1$, the result of this procedure is a set of barycentric coordinates B_0 so that the geometry $W_0 = \phi_{V_1}(B_0)$ of the barycentric coordinates on $\mathcal{M}_1$ is an embedding ϕ_{W_0} of $\mathcal{M}_0$ on the surface of $\mathcal{M}_1$, and vice versa. The idea is that this mapping of vertices from one mesh to the other accomplishes the main part of a bijective mapping between the surfaces of $\mathcal{M}_0$ and $\mathcal{M}_1$. After this step only the edges and faces have to be adjusted accordingly.

The process is typically done by finding a common parameter domain D for the surfaces. By mapping each surface bijectively to that parameter domain, the mapping between the shapes is established. The typical parameter domains for meshes in the context of morphing are the sphere $\mathbb{S}^2$ (in case the meshes are topological spheres) or a collection of topological disks represented as a piecewise linear parameter domain L. In case of the disks, the meshes have to be decomposed into isomorphic structures of disks (which requires them to be homeomorphic). A major constraint is to take into account user specified or autmatically generated feature correspondences (i.e. vertex-vertex correspondences). Dependending on the approach chosen, this is done by reparameterization or by decomposing the meshes according to the feature correspondence.

In case of mapping to a sphere, an embedding ϕ_S with $S = \{s_0, s_1, \ldots\}, s_i \in \mathbb{R}^3, |s_i| = 1$ is computed. The embeddings on the sphere are aligned according to the feature correspondence using a bijective map f that maps spheres into spheres.

$$\begin{tikzcd} \{i\} \in K_0 \arrow[r, "W_0"] \arrow[d, "\phi_{S_0}"'] & \phi_{V_1}(B_0) \\ \mathbb{S}^2 \arrow[r, "f"'] & \mathbb{S}^2 \arrow[u, "\phi^{-1}_{S_1}"'] \end{tikzcd}$$

The main problems in this approach are to compute the vertex coordinates S_0, S_1 on the sphere and the reparameterization f.

The decomposition approach is more general and more difficult. In addition to generate embeddings of the topological disks one has to decompose the meshes in an isomorphic way, taking possible feature correspondences into account. Formally, an abstract simplicial complex L consisting of a subset of the vertices in K_0, K_1 is used as coarse approximation of both meshes:

$$\phi_{V_0}(|L|) \approx \phi_{V_0}(|K_0|), \phi_{V_1}(|L|) \approx \phi_{V_1}(|K_1|)$$

Typically, L is topological minor of K_0 as well as K_1, i.e.

it is a partition of the meshes. Vertices in K_0, K_1 are identified with a face in L and all vertices belonging to a particular face are embedded in its planar shape. Thus, the common paramter domain is the topological realization $|L|$, where each vertex is represented with a barycentric coordinate with respect to a particular face in L. This requires to embed pieces of the mesh in the plane.

$$\begin{tikzcd}
\{i\} \in K_0 \arrow[r, "W_0"] \arrow[d, "\phi_{L_0}"'] & \phi_{V_1}(B_0) \\
|L| \arrow[r, "f"'] & |L| \arrow[u, "\phi_{L_1}^{-1}"']
\end{tikzcd}$$

Following, techniques to embed simply-connected bounded and unbounded meshes in the plane and on the sphere are explained. Then, approaches to dissect the meshes into isomorphic patch-networks (or, equivalently, inducing base-domains $|L|$ on $\mathcal{M}_0, \mathcal{M}_1$) are discussed. After these basic embedding steps reparameterization for feature alignemnt is introduced. Finally, some comments on rarely mentioned details in the correspondence problem are given.

3.1. Mapping topological disks

Simply-connected parts of the boundary of three dimensional shapes are homeomorphic to a disk and, therefore, called topological disks. In order to find a paramterization of such pieces we need a bijective map of a bounded, simply connected mesh to the plane.

In our application we need to find a bijective map between patches. Thus, it is necessary to constrain the boundary of the patches to a particular shape. Here, we concentrate on mapping an arbitrary bounded and simply connected mesh to a unit disk so that boundary vertices of the mesh lie on the unit circle. This limits the applicability of several parameterization approaches, which allow the boundary of a triangulated surface to be non-convex or not to be fixed a priori to achieve smoother or less distorted mappings[30, 46, 72].

In a first step the boundary vertices are fixed on the unit circle. First, the three vertices from the base domain L are fixed in an equiangular way. This is necessary to make sure that adjacent faces in the base domain have a continous parameterization across base domain edges. The remaining boundary vertices are fixed so that the arc lengths between neighboring vertices are proportional to the original edge lengths. The remaining (interior) vertices are free and their position is determined by a relation to neighboring vertices.

Most of the publicized approaches to solve this task minimize a quadratic error functional expressed as the vertex position relative to its neighbors. This boils down to solving a system of linear equations. Non-linear approaches either use higher order functionals to be minimized[30, 46] or are of algorithmic nature (e.g. Gregory et al.[25, 26], which is discussed after the linear techniques).

More specifically, let $\{\mathbf{v_i}\}$ be the vertices to be mapped to the disk so that the free interior vertices have indices $0 \le i < n$ and the fixed boundary vertices have indices $n \le i < N$. We aim at finding positions $\mathbf{w_i}$ in the plane with $|\mathbf{w_i}| = 1, n \le i < N$. The mapping is bijective if and only if no edges cross or, alternatively,

$$\forall (i,j,k) \in K. \det \begin{pmatrix} \mathbf{w_{i_x}} & \mathbf{w_{i_y}} & 1 \\ \mathbf{w_{j_x}} & \mathbf{w_{j_y}} & 1 \\ \mathbf{w_{k_x}} & \mathbf{w_{k_y}} & 1 \end{pmatrix} > 0. \tag{1}$$

However, this quadratic expression is akward to use as a criterion to guarantee that the planar embedding is valid, which is why most approaches resort to the more restricitve but sufficient linear conditions.

In the following we discuss three ways to define a linear system, whose solution yields positions for the vertices. In addition, the Divide&Conquer appraoch of Gregory et al.[25, 26] is explained.

3.1.1. Barycentric mapping

Tutte[70] has shown how to embed planar graphs in the plane using a barycentric mapping. In our restricted setting, the idea is simply to place every interior vertex at the centroid of its neighbors:

$$\mathbf{w_i} = \sum_{j \in \mathcal{N}(i)} \frac{1}{d_i} \mathbf{w_j} \tag{2}$$

Setting $\Lambda = \{\lambda_{i,j}\}$ with

$$\lambda_{i,j} = \begin{cases} d_i^{-1} & \{i,j\} \in K \\ 0 & \{i,j\} \notin K \end{cases} \tag{3}$$

this can be written as the mentioned system of linear equations

$$(I - \Lambda) \begin{pmatrix} \mathbf{w_0} \\ \mathbf{w_1} \\ \dots \\ \mathbf{w_{n-1}} \end{pmatrix} = \begin{pmatrix} \sum_{i=n}^{N-1} \lambda_{0,i} \mathbf{w_i} \\ \sum_{i=n}^{N-1} \lambda_{1,i} \mathbf{w_i} \\ \dots \\ \sum_{i=n}^{N-1} \lambda_{n-1,i} \mathbf{w_i} \end{pmatrix} \tag{4}$$

The matrix $(I - \Lambda)$ has full rank and, thus, there is exactly one solution. Tutte[70] has shown that this solution is a valid embedding of the mesh.

Note, that the shape of the mesh has no effect on the placement of vertices in the plane. All information for the embedding comes from K and it is clear that the embedding cannot reflect geometric properties contained in V of the mesh. In the following we try to incorporate information about the original shape.

3.1.2. Shape preserving parameterization

In the barycentric mapping the weights λ contain only topological information. Floater[16] determines weights that reflect

the local shape of the mesh. More precisely, the λ are so chosen that the angles and lengths of edges around a vertex are taken into account.

To compute the weights for a particular vertex $\mathbf{v_i}$ this vertex is placed in the origin and incident edges are laid out in the plane using the original edge lengths and angles proportional to the original angles. This is assumed to be the ideal parameterization $\mathbf{w_i'}$ of the mesh with respect to $\mathbf{v_i}$.

The weights are computed in way that would result in placing $\mathbf{w_i}$ in the origin if the neighbors $\mathbf{w_j'}$ were fixed and the system of equations had to be solved. Thus, we have

$$\mathbf{w_i} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \sum_{j \in \mathcal{N}(i)} \lambda_{i,j} \mathbf{w_j'} \tag{5}$$

and

$$1 = \sum_{j \in \mathcal{N}(i)} \lambda_{i,j}. \tag{6}$$

If $\mathbf{v_i}$ has only three neighbors this exactly determines the positive weights, for more than three neighbors a positive solution has to be chosen from the space of possibles solutions. Note that positivity results in convex combinations, which are necessary to assure a valid embedding. Floater presents a method to compute reasonable weights, which are guaranteed to be positive: Take the cyclically ordered set of neighbors $j_k \in \mathcal{N}(i), k \in \mathbb{Z}_{|\mathcal{N}(i)|}$. Determine sets of weights $\lambda_{i,j}(k)$ with respect to three subsequent neighbors $j_k, j_{k+1}, j_k{+}2$. This yields non-negative $\lambda_{i,j}(k)$ for each k. These weights are averaged to yield the final weights:

$$\lambda_{i,j} = \frac{1}{|\mathcal{N}(i)|} \sum_k \lambda_{i,j}(k) \tag{7}$$

The positions w_i are computed by solving (4). Recently, Floater[18, 17] has proven a generalization of Tutte's theorem, which shows that it is sufficient that each vertex is a convex combination of its neighbors.

3.1.3. Discrete harmonic mappings

Harmonic mappings are a concept found in several fields in mathematics using differentials. Harmonic maps are often described as the function u among all functions mapping to a given domain Ω that minimize the Direchlet energy

$$E_D(u) = \frac{1}{2} \int_\Omega |\nabla u|^2. \tag{8}$$

Pinkall and Polthier[53] show how to discretize this problem for triangles, so that weights are derived per vertex and neighbor leading to a system of linear equations of the form of Eq. (4). A somewhat clearer derivation can be found in a more recent work of Polthier[54]. There, it is shown that the discrete Dirchlet energy is

$$E_D(u) = \frac{1}{4} \sum_{i,j|\{i,j\} \in K} (\cot \alpha_{i,j} + \cot \beta_{i,j}) |v_i - v_j|^2,$$

$$\alpha_{i,j} = \angle(i, k_0, j), \beta_{i,j} = \angle(i, k_1, j), \{i, j, k_c\} \in K \tag{9}$$

and that the minimizer solves

$$0 = \frac{1}{2} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{i,j} + \cot \beta_{i,j})(\mathbf{v_i} - \mathbf{v_j}) \tag{10}$$

at each vertex i. This leads to weights

$$\lambda_{i,j} = \begin{cases} \frac{\cot \alpha_{i,j} + \cot \beta_{i,j}}{\sum_{j \in \mathcal{N}(i)} (\cot \alpha_{i,j} + \cot \beta_{i,j})} & \{i,j\} \in K \\ 0 & \{i,j\} \notin K \end{cases} \tag{11}$$

which are used to obtain an embedding by solving Eq. (4).

Another formulation, which is probably better known in the graphics community, is given by Eck et al.[13].

3.1.4. Area preserving Divide&Conquer mapping

Gregory et al.[25, 26] describe a recursive algorithm that aims at preserving the area of triangles in the mapping. The idea is to induce the mapping be recursively dividing the patch into two pieces, which are then mapped independently. The dissections are so chosen that the ratio of areas in the original mesh and the embedding are approximately the same.

In particular, two diamterical vertices on the boundary of a patch are chosen. A shortest path is computed using Dijkstra's algorithm. This path is mapped to the segment connecting the two vertices. Then the path is altered to minimize the difference of the ratio of areas in the embedding and on the triangulated surface. The segment divides the patch into two halves, which are treated in the same way, until all vertices are mapped.

3.1.5. Comparison and Conclusion

We have embedded parts of a mesh using the four approaches presented above. Note that the solution of matrix equation (4) could be performed using hierarchical techniques[44, 31], which is equivalent to using multigrid methods. However, the matrix has sparse structure and we have found it sufficient to use iterative solvers exploiting the sparseness.

Some of the results of the comparison are shown in Figure 1. It is apparent that the general structure of larger and smaller triangles is very similar in all embeddings generated using linear optimization techiques. This suggests that connectivity is the major factor in these type of embeddings. Changing the weights used to compute the embedding only changes the local behavior of the embedding. They share the problem of area compression: Inner triangles have much less area than outer triangles. The area preserving scheme eliminates this problem at the cost of distorted triangle shapes.

In fact, all these parameterizations might be unusable due

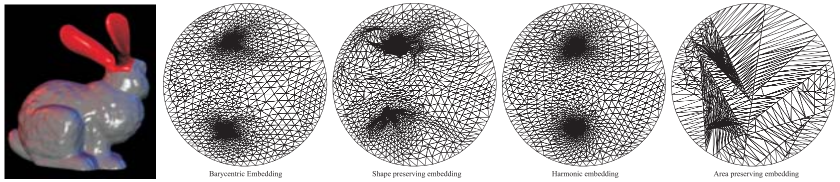

Figure 1: *A part of a mesh parameterized on the unit disk using different mapping techniques. The original geometry is highlighted in red. A barycentric mapping (see Section 3.1.1) does not reflect the geometry of the mesh. The shape preserving embedding tries to capture the local shape of the mesh by locally approximating conformal maps (see Section 3.1.2). Discretized harmonic embeddings minimize metric distortion (see Section 3.1.3). The area preserving embedding is a recursive process, which aims at approximating the original area of triangles (see Section 3.1.4).*

to the high ratio of either areas or anlges and the limited precision of floating point numbers. It has been observed that the base domain should have enough “skin” to allow for a reasonable parametrization of the mesh.

The small differences in local shape do not seem to have much influence on the resulting correspondence of the shapes. This is even more true when local features of the shapes are aligned by reparameterization (see Section 3.4).

3.2. Parameterizing topological spheres

Unbounded simply-connected 2-manifolds are called topological spheres because they are homeomorphic to spheres. A natural parameter domain for such shapes is, therefore, a unit sphere.

3.2.1. Star shapes

Kent et al.[37, 38] were the first to present techniques to map certain classes of genus 0 meshes to a sphere. A particularly simple class of objects are convex shapes. A convex shape has the property that a straight line connecting any two boundary points of the shape lies completely inside the model. Thus, all points are visible from any interior point of the shape and a projection through an interior point onto an enclosing sphere is necessarily bijective.

A generalization of this idea extends the class of shapes to star shapes. Such shapes have at least one interior point so that straight lines connecting this interior point with boundary points lie completely inside the shape. Interior points with this property are called star points. Obviously, projecting the boundary points of a shape through a star point onto an enclosing sphere is a bijective mapping. Specifically, if point O is visible from all vertices of the mesh then translate all points so that O coincides with the origin. Then normalize all vertex coordinates. These vertex coordinates are the parameterization of the mesh vertices on a unit sphere. An illustration is given in Figure 2.

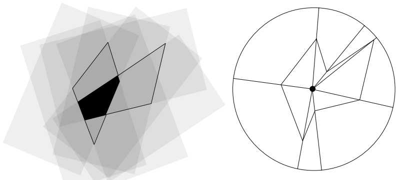

Figure 2: *A polygonal star shape and its projection to a circle. The kernel of a star shape is the intersection of all open half spaces over the edges (faces in case of a polyhedron). Every point in the kernel induces a bijective mapping to the circle by projection.*

The only problem is to determine whether a shape is star shaped and if so to find a star point. For piecewise linear shapes (meshes) this can be done by intersecting halfspaces induced by the face elements of the mesh. The intersection of all halfspaces is called kernel. If the kernel is non-empty the mesh is star shaped and every point inside the (convex) kernel is a suitable star point. The kernel of a mesh in 3D can be computed in $O(n \log n)$ using standard techniques [56].

3.2.2. Simplification

Shapiro and Tal[62] seem to be the first to present a reliable scheme that turns arbitrary genus 0 polyhedra into convex shapes. They first simplify the shape using vertex removal until the simplified shape is a tetrahedron. Only vertices with valence 3,4, and 5 are removed. Since the mesh is triangular such vertices always exist: It follows easily from the Euler-Poincare formulas that the average degree in any triangular (surface) mesh is less than 6. Thus, at least one vertex with degree strictly less than 6 has to exist.

Once the shape is simplified to a tetrahedron, vertices

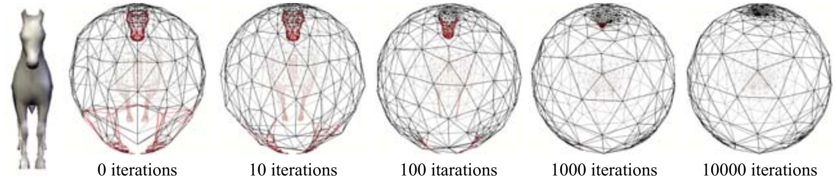

Figure 3: *Embeding a polyhedral object on a sphere using relaxation. Initially, the vertices are projected through an interior point of the model onto a unit sphere. The relaxation is finished when all faces are oriented correctly. Incorrectly oriented faces are surrounded by red edges.*

are reattach making sure that the shape stays convex. More specifically, it is shown how to attach vertices with degree 3, 4 and 5 to a convex shape so that the shape stays convex. More specifically, if a vertex $\{i\}$ has to be added to a face f, its position has to be outside the convex hull of the current mesh but inside the kernel of faces adjacent to f.

3.2.3. Spring embedding

Alexa[1] introduced a variation of the methods presented for planar embeddings to embed polyhedra on the unit sphere. The basic idea is the same as in barycentric mappings: Place each vertex in the centroid of its neighbors. On the sphere, however, two conditions of the planar case are violated. First, convex combinations of the neighbors' positions are not part of the domain (the sphere) and, second, no peripheral cycle is given to support the embedding.

The approach is to use a relaxation algorithm to compute the solution to the barycentric constraints. The starting configuration is generated by computing an interior point of the solid model represented by the mesh and then projecting all vertices to a sphere, which is centered at the interior point. The relaxation algorithm repeatedly places each vertex at the centroid of its neighbors. Since the centroid is not on the sphere the coordinate is normalized:

$$\mathbf{w}_\mathbf{i}^{\mathbf{l+1}} = \frac{\sum_{j\in\mathcal{N}(i)} \mathbf{w}_\mathbf{j}^\mathbf{l}}{\left\|\sum_{j\in\mathcal{N}(i)} \mathbf{w}_\mathbf{j}^\mathbf{l}\right\|} \tag{12}$$

The main problem of this approach is the missing of fixed vertices. In the planar case the fixed vertices avoid that all vertices collapse to one point, which is the trivial solution to (12). On the sphere, a local minimum exists such that the points are distributed over the sphere, however, the naive relaxation algorithm tends to find the global minimum, i.e. all vertices coincide.

Gumhold[27] has reported an elegant and simple solution to this problem: The sphere is recentered after each relaxation

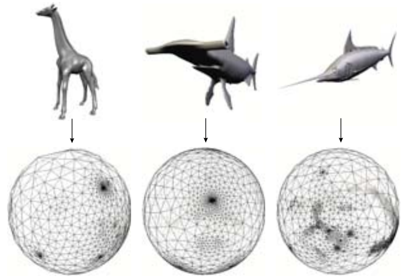

Figure 4: *Sphere embeddings of the models of a giraffe, a hammerhead shark, and a swordfish.*

round, i.e.

$$\mathbf{w}_\mathbf{i}^{\mathbf{l+1}} = \frac{1}{n}\left(\sum_{j\in K} \mathbf{w}_\mathbf{j}^\mathbf{l}\right) - \mathbf{w}_\mathbf{i}^\mathbf{l} \tag{13}$$

If we want to guarantee the topological correctness of the embedding, an epsilon bound of any kind is inadequate as the only termination criterion. Instead, the process is finished only when a valid embedding is found. The embedding is valid, if and only if all faces are oriented the same.

We can check this condition by testing the orientation of three consecutive vertices along the boundary of each face. Here, orientation refers to whether the three vertices make a clockwise turn on the surface of the sphere. This can be computed by evaluating $sgn\left((\mathbf{v_0} \times \mathbf{v_1}) \cdot \mathbf{v_2}\right)$.

A relaxation process for the polyhedral model of a horse is depicted in Figure 3 and resulting embeddings for several models are shown in Figure 4.

3.2.4. Comparison and Conclusion

None of the techniques discussed above makes a particular attempt to preserve the properties of the original mesh. Additional constraints (as discussed in Section 3.4) are necessary to make these embeddings usefull. The central projection is obviously limited to a small class of objects. We find that the two techniques for general genus 0 meshes have somewhat complementary features/problems: The simplification approach is more robust (in terms of geometric computations and sensitivity to topolgical defects in the mesh) while the relaxation generates smooth embeddings. Both techniques suffer from the area compression problem mentioned earlier.

3.3. Isomorphic dissection

The more general approach to establish correspondence between meshes is to dissect them into pieces. Each piece is a topological disk and can be mapped to the plane using one of the techniques discussed in Section 3.1. Of course, the shapes have to be split in such a way that the graphs representing the dissections have equivalent topologies.

This approach is not limited to a particular topology of the shapes, since the dissection results in a set of topological disks. However, the shapes need to be homeomorphic so that their dissections could be topologically equivalent. With extra conditions it is possible to deal also with topologically different shapes.

3.3.1. Automatic dissection of shapes

Ideally, the dissection process would not require the user to assist. However, the fully automatic dissection of two meshes into isomrphic structures seems to be a hard problem. The approach of Kanai et al.[33, 34] uses a single patch and, thus, automatically decomposes into isomorphic structures. However, the approach is limited to genus 0 meshes and suffers from the already mentioned area compression problems in the embedding.

Several techniques exist for the dissection of a single mesh. In the context of multi resolution models several approaches require the mesh to be broken into patches. This problem is know as mesh partioning and naturally related to graph theory. Some algorithms try to balance the size of patches (e.g., Eck et al.[13], Karypis & Kumar[36]).

In many multi resolution methods, however, the base domain (the structure of large patches) is found by simplifying the mesh using vertex removal[59, 58, 39, 41] or edge collapse[29, 21, 47].

These techniques might help in deriving a single base domain for two meshes. Lee et al.[43] use two independently established base domain to generate one base domain for both meshes. They employ their MAPS scheme[44] to build independent parameterizations over different base domains. These base domains are merged (see Section 4) so that the resulting merged base domain contains both independent base domains as subgraphs. Note, that in general the correspondence problem had to be solved for the geometry of the base domains. Lee et al. assume that the geometry of the base domains is so similar that this problem could be solved with simple heuristics (e.g. projecting in normal direction).

3.3.2. User specification of isomorphic dissections

The underlying idea of all works in this section is that the user specifies the topology/connectivity of the base domain and the location of the base domain vertices on the original meshes. Tracing the edges of the base domain on the mesh is more or less done automatically.

DeCarlo and Gallier[12] do not assist the user specifying the edges. While this way of defining the dissection gives a lot of freedom to the user it is very time consuming.

Gregory et al.[25, 26] assist the user in defining the edges (see Figure 5). The base domain is developed while intersecting the surfaces. The user defines a pair of vertices on a mesh and the system finds a shortest path of mesh vertices connecting the defined vertices. Subsequently, feature vertices can be connected to existing feature vertices using shortest paths along the mesh. By picking corresponding vertices in the input meshes the system will construct the same graph in the input meshes. A problem could arise from the fact that only mesh vertices are used to find shortest path.

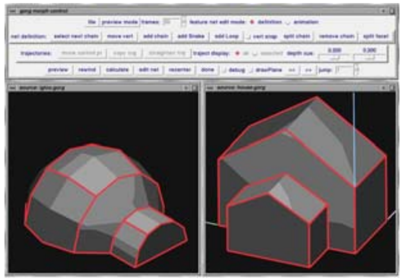

Figure 5: *User guided decompsition of meshes. Here, the user constructs closed loops and segments to dissect two meshes into isomorphic patch networks. Reprinted from Gregory et al.*[25, 26].

The works of Bao and Peng[6] and Zöckler et al.[73] are similar in spirit. However, it seems that they allow to use more points to define the boundary of a patch. Points are connected with the shortest paths in the vertex-edge graph as in in the work of Gregory et al.[25, 26]

In the approach of Kanai et al.[35] the user first defines a

set of corresponding feature vertices. Aware of the problems resulting from using a shortest path consisting of mesh vertices the authors compute the shortest path on the piecewise linear surface connecting the feature vertices. This path may or may not coincide with vertices and edges. Figure 6 shows the resulting dissectin for two cars. Since computing exact shortest path on polyhedral surfaces is difficult and time consuming they employ an approximate method that refines the original mesh and uses Dijkstra's algorithm[32].

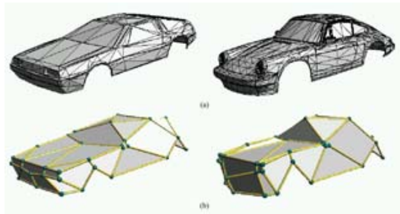

Figure 6: *In the approach of Kanai et al.*[35] *the user selects only corresponding vertices and which have to be connected. The mesh is the dissected using shortest path connecting the vertices. Reprinted from Kanai et al.*[35].

However, even using the exact shortest path can lead to problems. Praun et al.[55] illustrate the problem and propose better solutions: If a shortest path would cross an already established edge of the base domain, the shortest possible connection avoiding the intersection is computed using a wavefront algorithm. However, also the order of vertices being connected is important, because several edges might enclose an unconnected vertex. This problem can be avoided by traversing the vertices along a spanning tree.

In our view, the underlying problem is that on non-convex and unbounded shapes more than one geodesic between to points exists on the surface. We believe that a set of these geodesics is sufficient to trace out the given connectivity of the base domain. To implement this, first all geodesics between connected vertices of the base domain would be computed. Then, these edges would be inspected for possible intersection. The intersection-free subset yields the decomposition of the original mesh.

3.4. Feature alignment

The necessity for aligning prominent features becomes evident even in very simple examples. Figure 7 shows two morphs between models of a young pig and an adult pig. In the upper sequence, no features were aligned and the resulting morph is unacceptable. The lower sequence of Figure 7 shows a morph produced with some features (ears, eyes, hoofs, and the tail) aligned. The result is obviously more pleasing. Surprisingly, the need of user guidance becomes more obvious when the shapes are similar. This is because we can envision a transformation, i.e. we expect common features of the models (head, legs, etc.) to be preserved. But this does not happen, of course, due to the different mesh connectivity of the models (in this example, the different mesh connectivities are obvious from the different vertex counts of the models).

3.4.1. User-selected vs. shape features

A difficult task is to identify common features in several shapes. It seems impossible to automatically find such common features as they are mostly defined in a semantic and not necessarily in a geometric way. The user can identify these features and provide information about their location and correspondence (for instance as vertex-vertex correspondence of a few vertices). The algorithm should exploit this information as much as possible.

All dissection type methods explained above offer this way of user-control. Since the user explicitly chooses corresponding patches (and, therefore, corresponding edges and vertices) they can specify which parts of the meshes correspond. However, the user is also involved in other tasks, which can make the process complicated and lengthy.

The shapes' geometry also contain information useful to exploit. Several functions over the parameter domain of the meshes seem to be worth looking at. It is important that these functions are independent of the paramterization, i.e. are intrinsic to the shape and do not change if the description of the shape is changed. Such functions are especially considered in *differential geometry*, which could be seen as exploring a shape on the shape, i.e. without a distant view. The most prominent assets for describng shapes in differential geometry are

- normals, which are independent of translation and scaling but sensitive to rotation and
- curvature (principal curvatures, mean or gaussian curvature), which is independet of translation and rotation but sensitive to scaling.

The parameterization of the shape's boundary allows to represent these quantities as a function in two variables, i.e. the normal $n : \mathbb{R}^2 \to \mathcal{S}^2$ or the gaussian curvature $c : \mathbb{R}^2 \to \mathbb{R}$.

It is clear that this information about the meshes does not lead to point to point correspondeces such as user selected features. Instead the quality of the match of two shapes is quantified as a function of the distance of the shape descriptors. For example, Surazhsky and Elber[66] use the integral over the inner products of normals:

$$R = \int_D \langle n_1, n_2 \rangle \, dD \qquad (14)$$

Here, the inner product between normals and the integration over the surface represent particular choices. One might

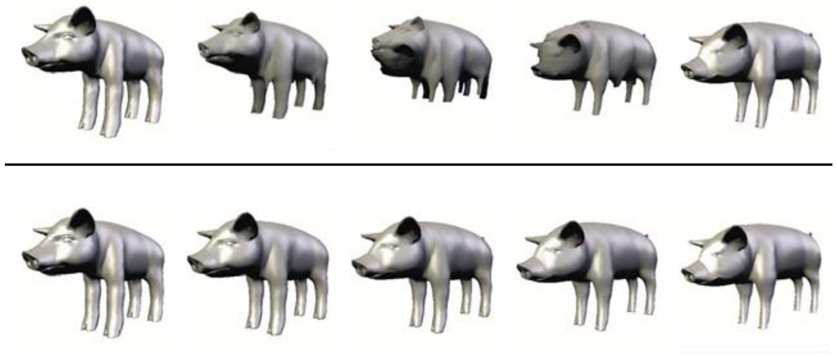

Figure 7: *Morphs between the models of a young pig and a grown-up pig. In the upper row, no feature alignment is used, which leads to unpleasant effects (e.g., eight legs in the intermediate models). In the lower row, the eyes, ears, hoofs, and the tail are aligned (a total of 17 vertex-vertex correspondences), yielding a smooth transformation.*

choose another metric for the difference of normals as well as another method to take into account the set of differences (e.g. the maximum of the angles between normals). In order to match shapes based on such criteria the paramterization is changed so that the functional is mimized. Note that no point has an apriori optimum placement making this problem much harder to solve than aligning specified point to point correspondences.

3.4.2. Transforming to align features

As a first step in an alignment procedure the parameter domains should be transformed using affine transform to roughly align the fatures. Note that this is not possible for paramterizations resulting from dissection as the orientation of each patch is determined by neighboring patches.

Alexa[1] aligns a set of point to point correspondences by rotating the spherical embeddings of the mesh. The objective function to be minized is the squared distance of corresponding points. The minimization problem can be solved using the techniques explained in Section 5.2.

3.4.3. Warping parameterizations to align features

In general, one could generate any parameterization of the meshes as a first step to establish correspondence. After this, the parameterization domain can be used to align user selected features or automatically generated features in terms of a re-parameterization of one or more of the initial parameterizations.

Alexa[1] and Zöckler et al[73] explicitly allow the user to select a set of point to point correspondences. Warping techniques similar to those used in image morphing (e.g., see the overview works of Ruprecht[57] or Wolberg[71]) are used to deform the parameterization so that corresponding points coincide. Whether the parameter domain is a disk[73] or a sphere[1] does not make a difference for the general approach.

In contrast to image warping, it is absolutely necessary that the warp does not introduce incorrectly oriented faces. This would be less of a problem if vertices as well as edges were warped. But since the algorithm later might require edge-edge intersection tests, warping the edges is impractical. Instead, edges should be (still) defined as the shortest path between vertices. That is, we warp the vertices only. Thus, even injective warping functions might introduce foldover.

Two solutions have been proposed in the context of morphing: Alexa warps only as much as is possible with the given triangulation. If the mapping starts to introduce foldvover in the triangulation the warp is made more local by adjusting the radius of incluence. However, the features are not guaranteed to coincide after this process.

Zöckler et al use the foldover free warping scheme of Fujimura and Makarov[20]. They also warp in small steps. However, if foldover occurs they change the mesh connectivity to assure that the embedding stays valid. In particular, they use edge flips for this task. This changes the original triangulation of the meshes.

Recent work in texture mapping allows to incorporate point constraints[14, 45]. These techniques could be applied for the problem here.

L‘evy[45] formulates the problem of satisfying given point

constraints by incporating the squared error of the point correspondences into the energy functional used to generate the parametrization. Using a scalar to weigh the importance of the point correspondence allows to trade between the regularization term for the smoothness of the parameterization and the accuracy of satisfying the constraints. On the other hand this mixed energy functional does not guarantee a valid embedding. A possible way would be to start with a valid emebdding and then increasing importance of the constraints as long as the embedding stays valid.

Eckstein et al.[14] propose a scheme that allows to exactly satisfy constraints whenever possible. It might be necessary to introduce additional vertices in the triangulation for this. The triangulation is first simplified so that it contains only the constrained vertices. These are placed accordingly and the mesh is then refined again. During the refinement process it might be necessary to insert additional vertices because straight edges connecting vertices could intersect.

3.5. Conclusions

The ideal algorithm for finding a parameterization of a mesh has not been found. In general, coarse simplifications of the original meshes are accepted as useful parameter domains. In the context of morphing they are not ideal for two reasons:

- For seemingly different shapes a common base domain might be hard to find and the decomposition of the original mesh *forces* the user to interact.
- The alignment of features (e.g. shape features) is restricted to corresponding patches of the base domain.

In view of these limitations the simple solution to embed topological spheres on a unit sphere has some appeal. However, embedding complex shapes on a sphere might result in a distorted parameterization because the local ratio of surface area between sphere and original shape differs.

It seems that finding a common base domain is the method of choice. For applications, in which one base domain is needed for more than one shape, techniques should be developed that include geometric features in the decomposition process.

We still search for a realiable method that works on arbitrary input, takes any number of user-constraints into account, optimizes a reasonable resemblance of the shapes, and is sufficiently fast.

4. Representation mesh

Given two embeddings W_0, W_1 of meshes $(V_0, K_0), (V_1, K_1)$ on a common domain D we aim at generating one mesh connectivity K with vertex positions $V(0), V(1)$ so that the original shapes are reproduced, i.e.

$$\phi_V(0)(|K|) = \phi_{V_0}(|K_0|), \phi_V(1)(|K|) = \phi_{V_1}(|K_1|). \tag{15}$$

Note that the vertex positions $V(0), V(1)$ are already available using the barycentric coordinates of each vertex w.r.t. the base domain. These barycentric coordinates allow to map each vertex from one mesh to the other. However, the exact mapping of vertices onto the piecewise linear surface might lead to bad results. The next subsection discusses better alternatives for the absolute position of vertices.

The main point of this section is to establish the common connectivity K. The typical approach found in the morphing literature is to generate a supergraph of the connectivities K_0, K_1, i.e. one that contains the simplices of both plus additional vertices if edges cross. This graph is found by *map overlay*. Here, we distinguish two cases:

1. Bounded meshes embedded in a disk.
2. Unbounded meshes, assuming the geometries of several meshes are sufficiently close.

These cases stem from the parameterization methods presented in the previous chapter.

Looking at multiresolution techniques for meshes is an alternative way of generating a common connectivity is remeshing. In particular, the parameterization is exploited to map planar coordinates of refinement operators to coordinates on the surface of the shapes. Provided the base domain accurately represents sharp features of the meshes this approach has the advantage that it is much easier to scale. The size can be easily adapted to the desired precision. For the same reason this approach is easier to extend to more than two meshes.

4.1. Mapping parameter values to the surface

After the meshes have been paramterized it is easy to find the posistion of a particular vertex on the surface of a mesh. Assume we want to find the position of vertex $\mathbf{v_{1_i}}$ of the first mesh on the second mesh. We determine the vertices $\{\mathbf{w_{2_j}}\}$ comprising the face in the parameterization in which the parameter domain position $\mathbf{w_{1_i}}$ lies. Then, $\mathbf{w_{1_i}}$ is represented in barycentric coordinates with respect to $\{\mathbf{w_{2_j}}\}$:

$$\mathbf{w_{1_i}} = \sum b_k \mathbf{w_{2_{j(k)}}} \tag{16}$$

The position of $\mathbf{v_{1_i}}$ in the other mesh is found as

$$\mathbf{w_{1_i}}' = \sum b_k \mathbf{v_{2_{j(k)}}} \tag{17}$$

This is the exact position on the piecewise linear shape and the way used in most of the morphing literature.

However, this does not take into account the idea that piecewise linear shapes are (in most cases) just approximations of smooth shapes. Particular practical problems occur when normals have to be rebuild from these new geometric positions: Vertices inside a face get the face's normal. If standard rendering methods are used (vertex normals and Gouraud shading) this results in detoriated shading.

It would be advantagous to find positions which result in

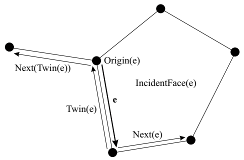

Figure 8: *The doubly connected edge list.*

a smooth surface. More specifically, we would like to use the barycentric coordinates to find positions *over* a triangular face and not necessarily on the face. This calls for methods defining a smooth surface from a coarse mesh. An obvious choice for such a method would be subdivision (e.g., Loop subdivision[48] or Kobbelts $\sqrt{3}$-scheme[40]).

4.2. Map overlay data structure

We need a data structure to store the meshes, which allows to add and remove edges, gives quick access to topological information (e.g., the ordering of edges around a vertex), and is not to heavy in terms of storage. We choose the doubly connected edge list[51] (sometimes called twin-edge data structure). The basic data type of this data structure is the edge. Edges are stored as two directed half edges. More specifically, the following information is stored:

Face The face record contains a poiner to an arbitrary half edge on its boundary.

Edge Each edge record contains pointers to

- its originating vertex,
- the face it bounds,
- the half edge connecting the same vertices but in the opposite direction (its *twin*),
- the next half edge along the boundary of the bounded face.

Vertex The vertex record contains a pointer to an arbitrary half edge originating from this vertex as well as location in space and other attributes (e.g., normal, color, texture coordinate).

Figure8 illustrates the data structure. Note that it is particular easy to iterate along the boundaries of faces (next pointers) or through all edges incident upon a vertex in their circular order (twin → next). A good description of the doubly connected edge list can be found in Berg et al[11].

4.3. Open meshes embedded in a disk

Several algorithms were proposed for the problem of overlaying planar graphs - see a textbook[11]. In general, the planar map overlay has the complexity $O(n \log n + k)$, where n is the number of edges and k is the number of intersections. If the two subdivisions are connected (as in our case) the planar overlay can be computed in $O(n+k)$[15].

The general paradigm for planar overlay is *plane sweep*. Sweep algorithms process the input with a virtual line moving along its normal direction. Whenever a vertex intersects the sweep line the corresponding edge is added (the vertex is the starting point of this edge) or removed (the vertex is the endpoint) from the list of active edges. The list of active edges is tested for intersection with added edges. To further reduce the number of necessary intersection tests the active edges are stored in their order along the sweep line. This is done by inserting edges in the correct position. In addition, the order has to be updated at intersection points. Using the ordering, only neighboring edges have to be tested for intersection. This processing leads to an algorithm with complexity $O(n \log n + k)$. By exploiting that two connected graphs are intersected the complexity can be reduced to $O(n+k)$.

In the case that meshes are embedded on the disk special care has to be taken for the boundaries of the meshes. While we assume that the embedding is surjective (i.e. fills the disk), the boundary in fact is a polygon leaving small empty regions between the disk and the polygon. However, it is clear that the boundaries of the meshes to overlay should be mapped onto each other. So in order to avoid that the boundary polgons intersect with inner edges of the other mesh the boundaries have to be merged first. This is done by simply connecting the vertices of all meshes on the disk along the linear order given by the disks boundary. After this boundary polygon has been established the planar mesh overlay procedure can be computed.

4.4. Closed meshes in arbitrary position

There seem to be only a few publications about the overlay of meshes in general position (i.e. the triangulated surfaces are close to each other but not e.g. planar). Note that plane sweep solutions are not applicable in this case. Few publications deal with overlaying two subdivision of the sphere. Kent et al.[38] give an algorithm for the sphere overlay problem, which needs $O(n + k \log k)$ time. Alexa[1] has presented a solution to this particular problem, which *reports* the intersection of two spherical subdivisions in the optimum time of $O(n+k)$. Also, both algorithms exploit the topological properties of both subdivisions, which are used to guarantee the correct order of intersections. Here, we generalize these algorithms to work on two arbitrary shaped meshes, which are assumed to be suffiently close to each other. We also alleviate the problem that the published version[1] had a worst case complexity of $O(n + k \log n)$ for the *construction* of the merged mesh using the already reported intersections.

The algorithm consists of two main parts: First, finding all intersections, and second, constructing a representation for the merged model.

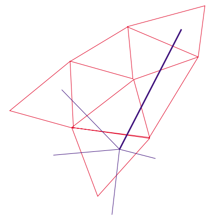

Figure 9: *Edge-edge intersections are determined by following an edge (blue in this illustration) over the faces of the other triangulation (red). After finding an intersection the face-to-face coherence exploited and only the edges of the next face are tested.*

4.4.1. Finding the intersections

In the algorithm two geometric functions are needed: One to decide if and where two edges intersect on the sphere, and a second to decide whether a point lies inside a face. Both geometric properties can be checked in a projection to the tangent plane of the surfaces. Since the meshes are supposed to be close in space their tangent planes should not differ to much. A suitable way of finding a common tangent plane is to take the cross product of two edges (i.e. the two edges to intersect, or two edges of the face to check).

The basic idea is to traverse the graphs breadth first, keeping information about the face that contains the current working edge and exploiting face-to-face neighbor information. Choose an arbitrary vertex $\{i\} \in K_0$ and search the 2-simplex $f = \{f_1, f_2, f_3\} \in K_1$ that contains it in under the bijective mapping. Start with an edge $e \in \mathcal{S}(i)$. Store e together with f on a stack. In general, the stack will always contain a directed edge together with the face in the other mesh containig the origin of this edge. The basic idea of the traversal is to walk over the faces following an edge (see Figure 9). Each edge $e = \{e_1, e_2\}$ is intersected first with the three edges $\{f_1, f_2\}, \{f_2, f_3\}, \{f_3, f_1\}$ bounding f, which contains $\phi_{W_0}(e_1)$. When an intersection is found the working edge e is emanating to the next face, i.e. the one that shares the intersected edge. This face is set to be f and is inspected in turn.

The same process is repeated with edges in K_1. This is necessary to find the topological orders of edges in K_0 cutting edges in K_1. Each edge is tested against three edges plus two additional intersection tests for each intersection being found. Thus, the algorithm has constant costs per edge and per intersection and the complexity is $O(n+k)$.

4.4.2. Generating the data structures

An appropriate data structure for storing the intersections is needed. Information about an intersection should be accessible from both intersecting edges at constant costs. We use a hashtable with edge indices as key values. When edge-edge intersections are found and stored in the intersection lists a pointer to the entry in the hashtable is stored. This means, both edges point to the same data structure containing information about the intersection (the intersecting edges in the beginning). The hashtable is only needed to access the entry when the intersections have already been computed by processing K_0 and need to be found when intersections from K_1 are generated. After reporting all intersections the hashtable is discarded.

The following two step algorithm contructs the merged mesh: First, edges in K_0 are cutted. We iterate through the intersection list of an edge and cut the edge at each intersection point. Thus, a new edge (two half edges) are generated for each intersection. The new edge represents the part of the edge that has to be processed. At each intersection the data structure containing the respective information is updated to now contain the two parts of the edge incident upon the intersection point. At this point only the twin pointers of the half edges are updated. The next pointers are left empty.

Second, edges in K_1 are processed. As in the first step edges are cut into two pieces at each intersection point. However, this time also the next pointers are updated. This is done by using the information stored in the intersection data structure, which now contains both edges of the already cut edge in K_1.

After all intersections are processed in this way we have a valid vertex and edge lists of the embedding. It remains to compute the records for the faces. Note that faces created from intersecting triangles are convex polygons with 3 to 6 sides, which should be triangulated. This is another subtelty, which is more involved as it may seem: While the polygon resulting from the intersection is convex it is not clear what shape it has in other geometric configurations, e.g. those of the source meshes. In principle one should find a triangulation that is admissible in all source geometries. This might be difficult and could lead to the need for additional vertices. The problem is known as *compatible triangulation* and discussed in detail in another context in Section 5.5.1.

Note that this appraoch could be extended to bounded meshes, however, boundaries require special treatment beyond the scope of this report.

4.5. Remeshing

A mesh is typically just an approximation of a shape. We have already seen that the mesh overlay process together with using coordinates lying exactly on the mesh might introduce artifacts into the source meshes (see Section 4.1). Thus, even if the original mesh connectivities are available as subsets of K the reproduction of the original shapes though exact is not ideal. It seems that the perfect reconstruction of the source shapes is impossible and we could as

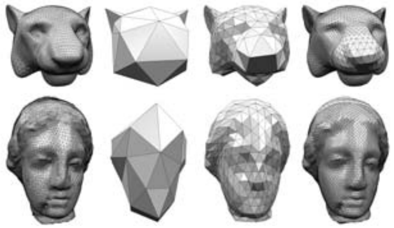

Figure 10: *A multiresolution mesh representation build over the same base domain to represent two geometries. Reprinted from Michikawa et al.*[50].

well use any mesh connectivity to approximate both given shapes.

Remeshing techniques have been used to construct semi-regular meshes from irregular input[44]. The irregular mesh is reduced to an irregular base domain. The base domain is refined inserting only regular vertices. The idea is to use refinement operators as known from subdivision surfaces, however, without using the geometric rules attached to the refinement. Instead, geometric positions are found by exploiting the bijection between original surface geometry and the parameterization. For example, using the 1-4 split the parameter domain positions of inserted vertices are given as edge bisectors. This parameter leads to the coordinate on the surface of the mesh.

This idea has recently been used to construct morphable meshes by Michikawa et al.[50] (see Figure 10). In this context, each parameter value leads to two coordinates. After several refinement steps a semi-regular mesh connectivity K is constructed together with coordinates $V(0), V(1)$ as desired. Because the refined connectivity is defined by the rules of the refinement used, only the base domain connectivity has to be stored explicitly.

To achieve a desired approximation accuracy, the number of refinement steps should be adapted to the geometric complexity of the meshes. Note that refinement could be done adaptively depending on the viewing conditions without necessarily computing and storing all coordinates of the refinement levels.

4.6. Comments

The remeshing approach is appealing because it allows to scale the size of the representation mesh. Its only limitation is the accurate representation of sharp features in the original shapes. In conventional multiresolution models this problem is alleviated by fitting the base domain to these features. In the context of morphing the base domain has to represent the features of several meshes, which do not necessarily coincide. This, again, incurs extra burden on the user, because a more complex base domain has to be induced on the input meshes. In addition, a more complex base domain limits the possibilities of automatic feature alignment methods. However, the flexible and lean representation mesh seems worth it.

5. Vertex paths

After the computation of one mesh connectivity K and two mesh geometries represented by vertex coordinates $V(0)$ and $V(1)$ it remains to compute vertex coordinates for the blended shapes. For a typical morphing animation, a set of vertex coordinates $V(t), t \in]0,1[$ has to be generated.

A simple choice is linear interpolation[35, 38, 25, 73]. A rigid[9, 10] or affine[1] transform prior to linear vertex interpolation yields better results. More complex behavior during the transform calls for more elaborate mathods. Such methods decompose the shape in to linear pieces and treat these pieces separately.

5.1. Linear interpolation of vertices

The easiest way to produce blends of corresponding shapes is to interpolate the coordinates of vertices. Given a transition parameter t the coordinates of an interpolated shape are computed by

$$V(t) = (1-t)V(0) + tV(1) \tag{18}$$

This type of interpolation produces good results if the shapes have the same orientation and are somewhat similar. Figures 11 and 12 show morph sequences obtained by linear interpolation.

Different orientation could lead to displeasing results. Imagine two squares that are rotated by 180 degrees against each other. If simple vertex interpolation is applied in this configuration, the interpolated shapes will shrink until the shape is collapsed to one point and then grow again. This is not the desired result in most applications. It is advisable to interpolate the orientation separately from the verex coordinates.

5.2. Interpolation of Orientation

Several ways exist to compute a relative orientation of two shapes. Note that it is difficult to interpolate the orienation of more than two shapes in 3D so the following discussion will be restricted to two shapes.

As a first step, the shapes are usually translated so that their centers of mass coincide with the origin. Then, a rotation[9, 10] or an affine transform[1] is computed seprating the rigid/affine part from the elastic part of the morph. A way of defining the rigid/affine part is to minimize the squared

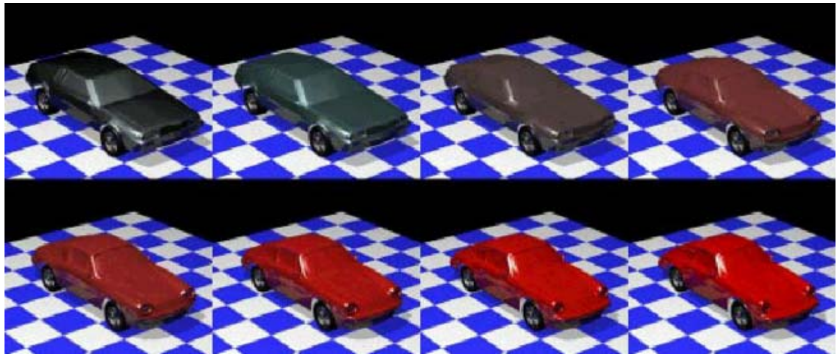

Figure 11: *A morph sequence obtained by linear interpolation using the base domains depcited in Figure 12. Reprinted from Kanai et al.*[35].

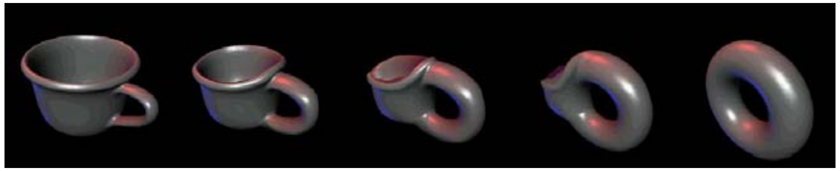

Figure 12: *A morph of objects with genus higher than zero. Reprinted from Lee et al.*[43].

distances of corresponding vertices using the corresponding transform. The minimization problem of finding an affine transform can be solved using the pseudo inverse of the coordinate vector. Let the vertex vectors be arranged as a $n \times 3$ matrix

$$V = \begin{pmatrix} v_{1_x} & v_{1_y} & v_{1_z} \\ v_{2_x} & v_{2_y} & v_{2_z} \\ v_{3_x} & v_{3_y} & v_{3_z} \\ \dots \end{pmatrix}.$$

Then the squared distance of coordinates under an affine transform A is

$$(V(0)A - V(1))^2 \tag{19}$$

and has to be minimized. This leads to linear system of equations, which can be solved using pseudo inverse $V(0)^+$:

$$A = V(0)^+ V(1) = \left(V(0)^T V(0)\right)^{-1} V(0)^T V(1) \tag{20}$$

Alternatively, the least squares solution (or, the pseudo inverse) could be computed using the SVD, which allows explicit control over the sensitivity to near rank deficiencies[22].

Intermediate shapes $V(t) = \{\mathbf{v_1}(t), \mathbf{v_2}(t), \dots\}$ are described as $V(t) = A(t)V(0)$. The question is how to define $A(t)$ reasonably? The simplest solution would be: $A(t) = (1-t)I + tA$. However, some properties of $A(t)$ seem to be desirable, calling for a more elaborate approach:

- The transformation should be symmetric.
- The rotational angle(s) and scale should change monotonic.
- The transoform should not reflect.
- The resulting paths should be simple.

The basic idea is to factor A into rotations (orthogonal matrices) and scale-shear parts with positive scaling components. Alexa et al.[3] have examined several decompositions. Through experimentation, they have found a decomposition into a single rotation and a symmetric matrix (i.e. the polar decomposition), to yield the visually-best transformations. This result is supported by Shoemake[63] for mathematical, as well as psychological, reasons. The decomposition can be

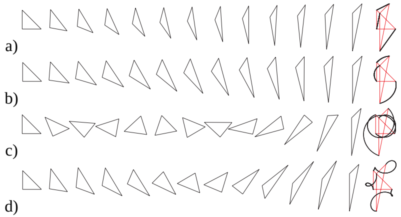

Figure 13: *Transformations of a single triangle. (a) Linear vertex interpolation. (b-d) An affine map from the source to the target triangle is computed and factored into rotational and scale-shear parts. Intermediate triangles are constructed by linearly interpolating the angle(s) of rotation, the scaling factors, and the shear parameter. (b) is generated using the SVD; (c) shows the results of reducing the overall angle of (b) by subtracting 2π from one of the angles; (d) corresponds to Equation 22 and represents the method of our choice. The last column in all rows shows plots of the vertex paths.*

deduced from the SVD as follows

$$A = R_\alpha D R_\beta = R_\alpha (R_\beta R_\beta^T) D R_\beta = (R_\alpha R_\beta)(R_\beta^T D R_\beta) = R_\gamma S \quad (21)$$

however, there are computationally cheaper alternatives[63]. Based on the decomposition, $A(t)$ is computed by linearly interpolating the free parameters in the factorizations in (21) , i.e.

$$A_\gamma(t) = R_{t\gamma}((1-t)I + tS). \quad (22)$$

Figure 13 illustrates the resulting transformations for a triangle. For comparison, 13(a) shows linear interpolation of vertex coordinates. The transformation resulting from a singular value decomposition and linear interpolation $A_{\alpha,\beta}(t)$ is depicted in 13(b). Note that the result is symmetric and linear in the rotation angle but still unsatisfactory, since a rotation of more than π is unnecessary. However, if we subtract 2π from one of the angles (depicted in 13(c)) the result is even more displeasing. We have found that decomposing A into one rotation and a symmetric matrix and using $A_\gamma(t)$ yields the best results (Figure 13(d)). It avoids unnecessary rotation or shear compared to the SVD and is usually more symmetric than a QR decomposition-based approach.

5.3. Interpolation of intrinsic boundary representation

Linear interpolation of vertices can lead to undesirable effects such as shortening of parts of the boundary during the transition. To avoid such problems, Sederberg et al.[60] propose to interpolate an intrinsic representation of the boundary. For polygons, such an intrinsic representation representation are edge length an interior angles. Unfortunately, there is no simple analog in 3D. An attempt was made to extend the ideas of to polyhedra[65] but the methods are computationally expensive and unreliable.

5.4. Smoothing paths among neighboring vertices

If every vertex is interpolated independently the paths of neighboring vertices are independent. However, one might expect a coherence among those paths. Ohbuchi et al.[52] construct a subdivision surface connecting the two boundary representations. This is comparable to the variational approach of Turk and O'Brian[69] for implicit shapes. Intermediate shapes are defined as cross sections of the subdivision surface. Since the subdivision surface is smooth in all directions not only the paths of sinlge vertices are smooth but they are also smooth among local neighborhoods. By introducing trans-finite constraints on the subdivision surface it is possible to preserve sharp features of the original meshes.

5.5. Interpolation of the interior of shapes

Shapira and Rappaport[61] suggest that a proper morph cannot be expressed merely as a boundary interpolation, but as a smooth blend of the interior of the objects. To achieve such an interior interpolation, they represent the interior of the 2D shapes by compatible skeletons and apply the blend to the parametric description of the skeletons. An extension of this approach to meshes - though theoretically possible - has not been presented so far. The extension of this idea to 3D has been investigated by Blanding et al.[7].

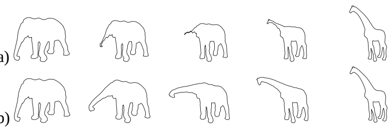

Figure 14: *Contour blends of an elephant and a giraffe. Interpolation of the boundary (a) vs. a decomposition based approach (b).*

Another way to represent the interior of the shapes is to decompose the shape into linear pieces or, more specifically, into simplices. The works of Floater, Gotsman, and Surazhsky[19, 24] and Alexa et al.[3] use this type of decomposition mainly for polygons, however, the extension to meshes has been demonstrated. The main difficulty in extending these approaches lies in the reliable computation of isomorphic dissections of meshes into simplicial complexes.

We will first discuss ways of generating such isomorphic complexes and then explain the possibilities they open for computing vertex paths.

5.5.1. Isomorphic Simplicial Complexes of Shapes

Simplicial complexes allow the local deformation of the shapes to be analyzed and controled. Here, we explain how to construct isomorphic dissections given two shapes with identical boundary connectivity.

The problem was first discussed by Aronov et al.[5] for polygons. They offer two general approaches: The first approach is to triangulate the polygons independently and then use a piecewise linear bijective map to compute a planar overlay of the triangulations. This is somewhat similar to the planar embeddings explained in Section 3.1 together with the overlay procedures in Section 4.3. The second approach of Aronov et al. is a universal triangulation that fits every n-sided polygon. This approach is extended by Gotsman and Surazhsky[24] to generate triangulations with few interior vertices. However, it is unclear how to extend the general triangulation to meshes because of the more complex boundary connectivity. For this reason, we concentrate on the first approach.

It seems that the mesh-version of compatible triangulations has not been discussed in the literature. However, the procedure is conceptually the same. The meshes are tetrahedralized independently using common techniques[8]. Then, a piecewise linear bijective map is computed between the shapes, typically using a common parameter domain. This parameter domain is used to compute an overlay of the simplical complexes.

In case the common parameter domain for the meshes is a sphere, the interior of the sphere could be used as the parameter domain for the tetrahedra. If a piecewise linear parameter domain is used it seems more difficult to find a mapping of the interiors. If the parameter domain is given and then induced onto the meshes one could as well prescribe a simplicial base domain and induce this simplical complex onto the (independent) simplicial complexes of the original meshes. The resulting structures are merged using a plane sweep algorithm similar to the line sweep algorithms discussed in Section 4.3.

The resulting simplicial complex might contain many, ill-shaped simplices, which cause the following determination of the vertex paths to detoriate. For that reason, both, Gotsman and Surhazky as well as Alexa et al. try to improve the simplicial complex while preserving the isomorphism. Gotsman and Surazhsky try to minimize the number of resulting simplices. Alexa et al. employ an approach motivated by meshing techniques, however, adapted to the situation that one connectivity has to work for two shapes.

It seems advantagous to start with Delaunay triangulations because they avoid the generation of skinny simplices, which would be inherited in the merged complex, and because the same connectivity is produced for similar regions, which reduces the number of extra simplices generated by the overlay. The following smoothing strategy tries to maximize the minimum angle (as the Delaunay triangulation does) by independently moving interior vertices and concurrently flipping edges. This procedure is called *compatible mesh smoothing*. If the result needs to be improved further, the vertex count is increased by means of splitting edges. The split operation is well-defined in terms of topology, if it is applied to both triangulations simultaneously, the isomorphism remains. The idea is to split long edges to avoid long skinny triangles. Figure 15 shows a result achieved with this approach.

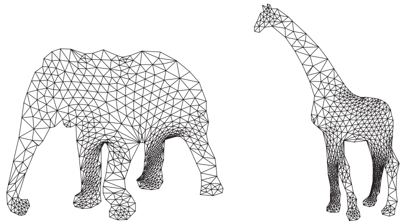

Figure 15: *The homeomorphic dissections of the shapes in the elephant-giraffe example*

5.5.2. Morphing barycentric coordinates

The approach of Gotsman et al.[19, 24, 67] requires not only the interior of the shapes to be decomposed but also the exterior. The exterior is bounded by a common fixed convex shape. This fixed convex shape allows to represent each interior vertex with barycentric coordinates:

$$\mathbf{v_i} = \sum_{j \in \mathcal{N}(i)} \lambda_{i,j} \mathbf{v_j}, \quad \sum_{j \in \mathcal{N}(i)} \lambda_{i,j} = 1, \lambda_{i,j} > 0 \quad (23)$$

Thus, each shape is fully described by the matrix of weights $\Lambda = \{\lambda_{i,j}\}$. The idea of the approach is to linearly interpolate these barycentric coordinates, i.e. $\Lambda(t) = (1-t)\Lambda(0) + t\Lambda(1)$. As was shown in Section 3.1.1 a convex boundary together with barycentric coordinates for the interior vertices results in a valid embedding of the complex (this is obiously not restricted to the two-dimensional case). A convex combination of the barycentric coordinate results in another barycentric coordinate so that the resulting vertex coordinates along the paths describe only valid (i.e. non-intersecting) shapes. This is a unique feature of this approach.

In this general setting several choices allow to influence the result of the morph. The first choice is with respect to the barycentric coordinates Λ. If a vertex has more than four neighbors (three in the planar case) its barycentric coordinate is not unique. A possible way of choosing the coordinate is to minimze the squares if the weights by using the pseudo inverse (see Section 5.2). Next, the paths do not have to be linear in the space of barycentric coordinates. Since the linear paths of vertices has some desriable features (e.g. it is short) one could modify the original barycentric coordinates Λ towards the original vertex representation (represented by I) and, thus, the linear morph. Surazhsky and Gotsman[67] show that Λ^m gets closer to the linear morph with increasing m and that if the linear morph is invalide their is a m' deviding the family Λ^m into sets of valid ($m \leq m'$) and invalid ($m > m'$) morphs.

5.6. Composing local ideal transforms

The general idea of Alexa et al.[3] is to find a transformation which is locally as similar as possible to the optimal transformation between each pair of corresponding simplices. The optimal simplex transform is found by factoring the affine transform defined by the pair of corresponding simplices as explained in Section 5.2. This defines ideal trajectories for each simplex.

Now consider the simplicial complex rather than a single triangle. We define an error functional for a candidate vertex configuration $V(t) = (\mathbf{v_1}, \mathbf{v_2}, \ldots)$

$$E_{V(t)} = \sum_{s \in K} \|A_s(t) - B_s(t)\|^2 , \qquad (24)$$

where $A_s(t)$ is the desired ideal transform computed for simplex s, $B_s(t)$ is the affine transform induced from $V(0)$ to $V(t)$, and $\| \cdot \|$ is the Frobenius norm.

We define an intermediate shape $V(t)$ as the vertex configuration which minimizes this error between the desired coordinates for each individual simplex and the space of admissible coordinates.

Note that the coefficients of $B_s(t)$ are linear in $V(t)$ and that the $A_s(t)$ are known for a fixed time t Thus, $E_{V(t)}$ is a positive quadratic form in the elements of $V(t)$. The functional $E_{V(t)}$ can be expressed in matrix form as

$$E_{V(t)} = \mathbf{v}^T \begin{pmatrix} c & G^T \\ G & H \end{pmatrix} \mathbf{v}, \qquad (25)$$

where $c \in \mathbb{R}$ represents the constant, $G \in \mathbb{R}^{2n \times 1}$ the linear, and $H \in \mathbb{R}^{2n \times 2n}$ the mixed and pure quadratic coefficients of the quadratic form $E_{V(t)}$. The minimization problem is solved by setting the gradient $\nabla E_{V(t)}$ over the free variables to zero.

The above definition has the following notable properties:

- For a given t, the solution is unique.
- The solution requires only one matrix inversion for a specific source and target shape. Every intermediate shape is found by multiplying the inverted matrix by a vector.
- The vertex path is infinitely smooth, starts exactly in the source shape, and ends exactly in the target shape. These are properties typically difficult to achieve in physically-based simulations.

Figure 16 shows transformations of some simple shapes produced with the described method.

5.7. Non-uniform interpolation

So far we have always morphed the whole mesh, i.e the transition has been desribed by a scalar transition parameter t. Now, we want to locally morph certain features or regions of interest, i.e. the transition parameters are different for different vertices. We will call the set of transition paramters for vertices the *transition state*.

An idea to specify non-uniform morphs is to explicitly define the path of several vertices. Gregory et al.[25, 26] allow the user to define the paths of base mesh vertices, which are used to modify the linear interpolation of the remaining vertices (see Figure 17). If a multiresolution hierachy is used paths could be defined on each of the levels and propagated to higher levels. This is a concept introduced in multiresolution mesh modeling[13, 49, 74, 42, 28], which has been used for morphing by Michikawa et al.[50].

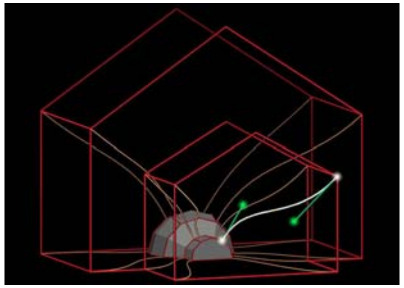

Figure 17: *Specifying non-uniform morph paths. Gregory et al.[25, 26] allow to specify the paths in base domain vertices explcitely (e.g. using splines).*

A major problem when morphing only locally arises from the fact that corresponding features might not have the same position in space and, thus, interpolation of absolute coordinates could lead to undesirable effects. This problem is illustrated in Figure 18. The shapes in a) and b) are source and target geometry of one mesh. The idea is to locally change the geometry of the baby's face so that the nose takes the shape of the boy's. Locally interpolating vertex coordinates leads to the shape depicted in c), which is clearly not usable. Note that the faces are overall aligned in space and that the misalignment of the noses results from different relative positions in the faces.

We could ease the problem of misalignment by assigning an affine transform to a local morph. This can be done in case

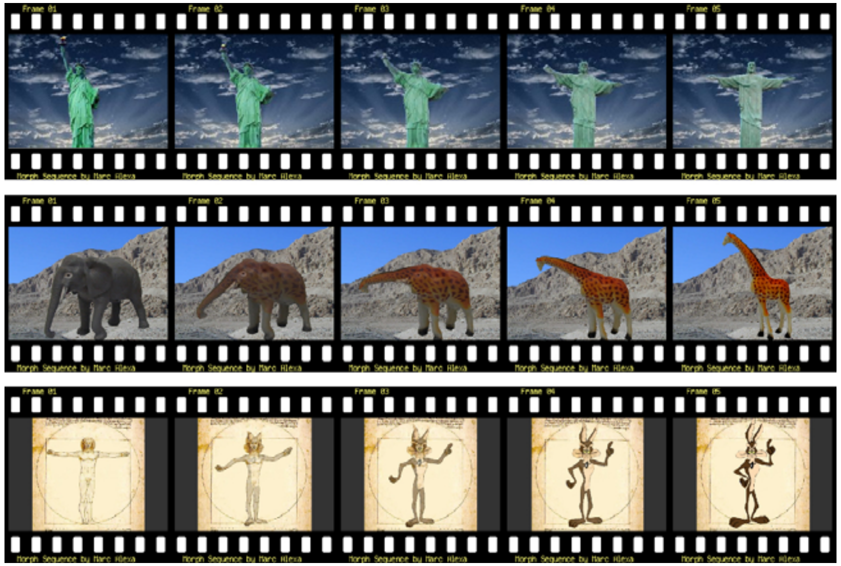

Figure 16: *Transformations of different shapes representing solid objects. Note that parts of the shapes transform rigidly whenever possible.*

the mesh is represented over a base domain. The coordinates could be represented relative to the base domain. However, the fixed base domain limits the flexibility and might introduce continuity problems at the base domain edges. More generally, a shape should be defined by the transition state of its vertices. In that way, the transition states is representative for the shape of a morphable object. This could be a very compact way of representing deforming or animated objects.

The main idea to overcome the limitations is to represent vertex coordinates with respect to their neighbors in the mesh (and not with respect to some larger structure). Given a vertex and its one-neighborhood ring (see Figure 19 a), the position should be described relative to the positions of vertices in the neighborhood. Further, the representation of a vertex should be linear in the absolute coordinates. Nonlinear functions tend to be numerically difficult to handle and many morphable meshes have sliver triangles, which, together, leads to unpredictable results.

The relative representation aims at making the shape of the mesh invariant to translation or, ideally, invariant under affine transforms. If a vertex were represented in the affine space of its neighbors invariance under affine transforms would trivially follow.

5.7.1. Laplacian Representation

Alexa[2] uses a rather simple scheme, which is not invariant under rotation, scaling, and shearing. Assume we want to represent the position of vertex $\{i\}$. Compute the center of mass of the neighbors

$$\bar{v}_i = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{v_j} \qquad (26)$$

and let the new representation be the difference of this center of mass to the original position:

$$\tilde{\mathbf{v}}_\mathbf{i} = \mathbf{v_i} - \bar{\mathbf{v}} \qquad (27)$$

For an illustration see Figure 19. If we write all vertices as a vector the forward transformation (from absolute to relative coordinates) can be represented in matrix form. Let A be the adjacency matrix of the mesh and D be a diagonal matrix with $d_{ii} = 1/|\mathcal{N}(i)|$. The transform is represented by $L = I - DA$. Note that L is a Laplacian of the mesh[68]. This is an important observation as it generalizes the approach to

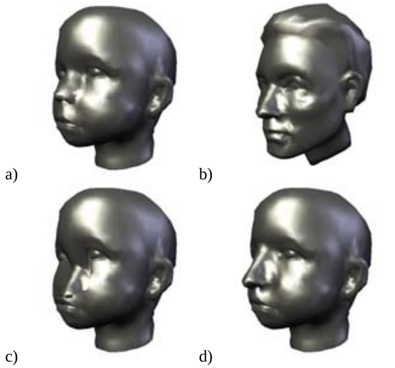

Figure 18: *Given a mesh with two geometries a) and b) so that corresponding features (eyes, ears, nose, mouth, etc.) are represented by the same vertices in both geometries. If one feature (in this example the nose) is morphed towards the target geometry in absolute coordinates, different positions in space lead to undiserable effects shown in c). The shape in d) shows a more pleasing result achieved by interpolating a differential encoding of the vertices.*

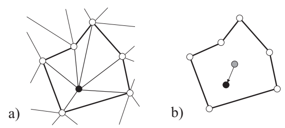

Figure 19: *A vertex (black) and its neighborhood ring (white) in a). In Laplacian coordinates a vertex is represented by the difference to the centroid of its neighbors (b).*

shape representations other than meshes, e.g. parametric or implicit functions.

The backward transformation (from relative to absolute coordinates) is, by construction, not unique. It should be uniquely determined up to a translation. This means, $L \in \mathbb{R}^{m \times m}$ should have rank $m-1$, which is indeed so[2].

The main idea of this approach is to morph by linearly interpolating Laplacian coordinates rather than absolute coordinates. This is somewhat similar to interpolating barycentric coordinates (see Section 5.4.2), however, here coordinates cannot be convex sums of neighbors as neighbor rings are not necessarily a base of $\mathbb{R}^3$. As such, the foldover-free property of convex combination morphs cannot be established here. Since Laplacian coordinates are linear in absolute coordinates morphing the whole shape (i.e. all vertices have the same transition state) will be the same in absolute and Laplacian coordinates. Yet, if the desired transitions are different for subsets of vertices interpolating Laplacian coordinates yields more reasonable results.

Alexa[2] gives details on how to generate and represent transition states and how to solve the resulting linear system.

6. More than two meshes

The conceptual extension of the framework to more shapes is straightforward. Given meshes $\mathcal{M}_i = (V_i, K_i)$ a common connectivity K together with vertex sets $V(\mathbf{e_i})$ is established. The vertex sets form a base of a space, which is reflected by using canonical base vectors $\mathbf{e_i}$ as indices. A morphed shape $(V(\mathbf{s}), K)$ is represented by a vector $\mathbf{s} = (s_0, s_1, \ldots)$ reflecting the shares of the meshes $\mathcal{M}_0, M_1, \ldots$.

Not all techniques presented in this framework are equally suited to be extended to more meshes. The correspondence problem discussed in Section 3 seems to be relatively easy to extend. All meshes are embedded in the given parameter domain, which leads to barycentric representation of the original vertices. If each set of original vertices V_i needs to be mapped to all other meshes $\mathcal{M}_j, i \neq j$ the complexity would grow quadratically with the number of meshes. However, this is not necessary if a remeshing strategy is used to generate a consistent mesh connectivity (see Section 4.5). This procedure generates the same set of vertices over all shapes, thus, the complexity is linear in the number of meshes times the number of vertices used in the remesh, which is the best we can expect. Concluding, the best way to generate the set $\{(V(\mathbf{e_i}), K\}$ is to embed all meshes in a common parameter domain (spherical or piecewise linear) and then remesh to the desired accuracy. This has been demonstrated by Michikawa et al.[50] (see Figure 20).

The vertex path problem now extends to compute combinations of several vertex vectors. Linear vertex combination is easily extended:

$$V(\mathbf{s}) = \sum_i s_i V(\mathbf{e_i}) \tag{28}$$

Surprisingly, any technique involving rotations such as the ones explained in Sections 5.2 and 5.4 seem to be difficult to extend. Instead of interpolating the orientation one could compute the principal components (moments) of the shapes and align them with the canonical axes of the coordinate system. To extend the local morph approach explained in Section 5.5 the linear combination has to be applied to the Laplacian coordinates.

Applications of such spaces of meshes range from modeling and analysis of shapes to animation. Praun et al. have termed the synthesis-analysis part digital geometry processing (DGP)[55]. Modeling could be achieved be combining several shape (features) to yield the desired result. Using techniques such as the principal component analysis, spectral properties of the mesh family can be explored.

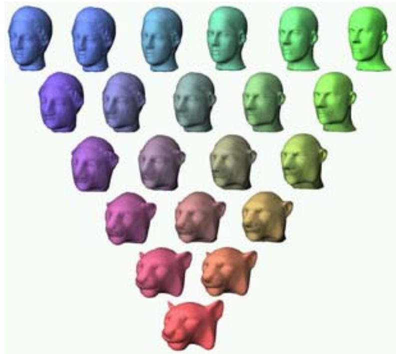

Figure 20: *A space of shapes generate from three input shapes and linear interpolating their geometry vectors. Correspondence has been established using a coarse base domain and a multiresolution mesh. Reprinted from Michikawa et al.*[50].

The space of meshes $(V(\mathbf{e_i}), K)$ allows to represent animations as a curve $\mathbf{s}(t)$. A classical key frame animation with k key frames could be simply models as a k-dimensional space, where the curve linearly interpolates subsequent key frames. Alexa and Müller[4] use the PCA together with rigid motion detection to find a more compact space (see an illustration in Figure 21). In this space the main part of the animation is stored in the rigid motion of the first base vector. The additional base vectors are sorted according to their energy in the spectrum of the animation. This allows to progressively store and stream mesh animations, where the progressiveness is with respect to movements and not model fidelity. The understanding of certain features of the animtion as bases of a linear space gives this representation semantics. It is possible to identify e.g. the smile in a facial animation with a particular basis and, thus, to modify only the smile without the need to work on all key frames.

7. Conclusions

Mesh morphing has reached a state where basic problems are solved, yet, a practical working system is not available. The correspondence and representation problems can be seen as the application of several techniques now common in multiresolution representations and modeling. The vertex path problem is specific to morphing applications and leaves room for improvement.

What is still missing is a robust implementation of current techniques. As with other geometric techniques, many of the approaches suffer from numerical problems. Many "detail" problems such as normal and texture coordinate interpolation can cause trouble in practice.

With the extension of mesh morphing to linear spaces of meshes several interesting avenues for future research arise. However, one might ask the question whether meshes (with fixed connectivity) are the right representation for deforming shapes, anyway.

Additional material

The web site http://www.sfc.keio.ac.jp/~kanai/GeomMorph/ maintained by Takashi Kanai has links to many other project web pages on mesh morphing and is a great source of information and material.

Acknowledgements

Encouraging and insightful discussions with many knowledgeable people, namely Daniel Cohen-Or, Jose Encarnação, Craig Gotsman, David Levin, Wolfgang Müller, are gratefully acknowledged. Many thanks to Arthur Lee, Ming Ling, Takashi Kanai, Peter Schröder for allowing me to re-use their original illustrations and morph sequences. Thanks to the anonymous referees for their helpful and detailed reports. This work contains (rephrased) parts of the author's prior works.

References

1. M. Alexa. Merging polyhedral shapes with scattered features. *The Visual Computer*, 16(1):26–37, 2000. ISSN 0178-2789.
2. M. Alexa. Local control for mesh morphing. In B. Werner, editor, *Proceedings of the International Conference on Shape Modeling and Applications (SMI-01)*, pages 209–215, Los Alamitos, CA, May 7–11 2001. IEEE Computer Society.
3. M. Alexa, D. Cohen-Or, and D. Levin. As-rigid-as-possible shape interpolation. *Proceedings of SIGGRAPH 2000*, pages 157–164, July 2000. ISBN 1-58113-208-5.
4. M. Alexa and W. Müller. Representing animations by principal components. *Computer Graphics Forum*, 19(3):411–418, August 2000. ISSN 1067-7055.
5. B. Aronov, R. Seidel, and D. Souvaine. On compatible triangulations of simple polygons. *Computational Geometry: Theory and Applications*, 3:27–35, 1993.
6. H. Bao and Q. Peng. Interactive 3d morphing. *Computer Graphics Forum*, 17(3):23–30, 1998. ISSN 1067-7055.

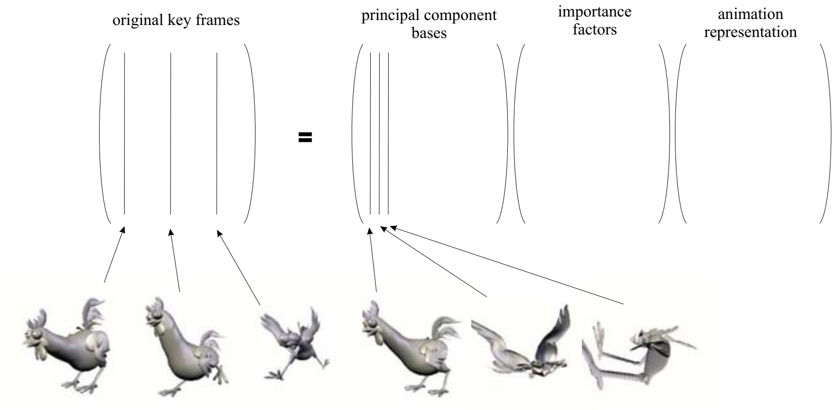

Figure 21: *The SVD applied to a space of meshes. Here the original space represents a key frame animation, the result allows to represent the same animation with less geometries containing most of the original animation's energy.*

7. Blanding, Turkiyyah, Storti, and Ganter. Skeleton-based three-dimensional geometric morphing. *CGTA: Computational Geometry: Theory and Applications*, 15, 2000.

8. B. Chazelle and L. Palios. Triangulating a non-convex polytope. In K. Mehlhorn, editor, *Proceedings of the 5th Annual Symposium on Computational Geometry (SCG '89)*, page 393, Saarbrücken, FRG, June 1989. ACM Press.

9. D. Cohen-Or and E. Carmel. Warp-guided object-space morphing. *The Visual Computer*, 13(9-10):465–478, 1998. ISSN 0178-2789.

10. D. Cohen-Or, A. Solomovici, and D. Levin. Three-dimensional distance field metamorphosis. *ACM Transactions on Graphics*, 17(2):116–141, April 1998. ISSN 0730-0301.

11. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry – Algorithms and Applications*. Springer-Verlag, Berlin Heidelberg, 1997.

12. D. DeCarlo and J. Gallier. Topological evolution of surfaces. *Graphics Interface '96*, pages 194–203, May 1996. ISBN 0-9695338-5-3.

13. M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. *Proceedings of SIGGRAPH 95*, pages 173–182, August 1995. ISBN 0-201-84776-0. Held in Los Angeles, California.

14. I. Eckstein, V. Surazhsky, and C. Gotsman. Texture mapping with hard constraints. *Computer Graphics Forum*, 20(3):95–104, 2001. ISSN 1067-7055.

15. U. Finke and A. Hinrichs. Overlaying simply connected planar subdivisions in linear time. In *Proceedings of the 11th Annual Symposium on Computational Geometry*, pages 119–126, New York, NY, USA, June 1995. ACM Press.

16. M. S. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250, 1997. ISSN 0167-8396.

17. M. S. Floater. Convex combination maps. *Algorithms for Approximation IV*, 2001.

18. M. S. Floater. One-to-one piecewise linear mappings over triangulations. *Math. Comp.*, to appear, 2002.

19. M. S. Floater and C. Gotsman. How to morph tilings injectively. *Journal of Computational and Applied Mathematics*, 101:117–129, 1999.

20. K. Fujimura and M. Makarov. Folder-free image warping. *Graphical Models and Image Processing*, 60(2):100–111, March 1998.

21. M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. *Proceedings of SIGGRAPH 97*, pages 209–216, August 1997. ISBN 0-89791-896-7. Held in Los Angeles, California.

22. G. H. Golub and C. F. Van Loan. *Matrix Computations*, volume 3 of *Johns Hopkins Series in the Mathematical Sciences*. The Johns Hopkins University Press, Baltimore, MD, USA, second edition, 1989. Second edition.

23. J. Gomes, L. Darsa, B. Costa, and L. Velho. *Warping and Morphing of Graphical Objects*. Morgan Kaufman Publishers, 1999.

24. C. Gotsman and V. Surazhsky. Guaranteed intersection-free polygon morphing. *Computers & Graphics*, 25(1):67–75, February 2001. ISSN 0097-8493.

25. A. Gregory, A. State, M. Lin, D. Manocha, and M. Livingston. Feature-based surface decomposition for correspondence and morphing between polyhedra. *Computer Animation '98*, June 1998. Held in Philadelphia, Pennsylvania, USA.

26. A. Gregory, A. State, M. C. Lin, D. Manocha, and M. A. Livingston. Interactive surface decomposition for polyhedral morphing. *The Visual Computer*, 15(9):453–470, 1999. ISSN 0178-2789.

27. S. Gumhold. Personal communication on embedding meshes, 2000.

28. I. Guskov, W. Sweldens, and P. Schröder. Multiresolution signal processing for meshes. *Proceedings of SIGGRAPH 99*, pages 325–334, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California.

29. H. Hoppe. Progressive meshes. *Proceedings of SIGGRAPH 96*, pages 99–108, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

30. K. Hormann and G. Greiner. Mips: an efficient global parametrization method. In P. S. P.-J. Laurent and L. L. Schumaker, editors, *Curve and Surface Design: Saint-Malo 1999*, pages 153–162. Vanderbilt University Press, 2000.

31. K. Hormann, G. Greiner, and S. Campagna. Hierarchical parametrization of triangulated surfaces. In B. Girod, H. Niemann, and H.-P. Seidel, editors, *Vision, Modeling and Visualization '99*, pages 219–226. infix, 1999.

32. T. Kanai and H. Suzuki. Approximate shortest path on a polyhedral surface based on selective refinement of the discrete graph and its applications. *Proc. Geometric Modeling and Processing 2000*, pages 241–250, 2000.

33. T. Kanai, H. Suzuki, and F. Kimura. 3d geometric metamorphosis based on harmonic map. *Pacific Graphics '97*, October 1997. Held in Seoul, Korea.

34. T. Kanai, H. Suzuki, and F. Kimura. Three-dimensional geometric metamorphosis based on harmonic maps. *The Visual Computer*, 14(4):166–176, 1998. ISSN 0178-2789.

35. T. Kanai, H. Suzuki, and F. Kimura. Metamorphosis of arbitrary triangular meshes. *IEEE Computer Graphics & Applications*, 20(2):62–75, March/April 2000. ISSN 0272-1716.

36. G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. Technical Report 98–036, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455, 1998.

37. J. Kent, R. Parent, and W. E. Carlson. Establishing correspondences by topological merging: A new approach to 3-d shape transformation. *Graphics Interface '91*, pages 271–278, June 1991.

38. J. R. Kent, W. E. Carlson, and R. E. Parent. Shape transformation for polyhedral objects. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):47–54, July 1992. ISBN 0-201-51585-7. Held in Chicago, Illinois.

39. R. Klein. Multiresolution representations for surfaces meshes based on the vertex decimation method. *Computers & Graphics*, 22(1):13–26, February 1998. ISSN 0097-8493.

40. L. Kobbelt. sqrt(3) subdivision. *Proceedings of SIGGRAPH 2000*, pages 103–112, July 2000. ISBN 1-58113-208-5.

41. L. Kobbelt, S. Campagna, and H.-P. Seidel. A general framework for mesh decimation. *Graphics Interface '98*, pages 43–50, June 1998. ISBN 0-9695338-6-1.

42. L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. *Proceedings of SIGGRAPH 98*, pages 105–114, July 1998. ISBN 0-89791-999-8. Held in Orlando, Florida.

43. A. Lee, D. Dobkin, W. Sweldens, and P. Schröder. Multiresolution mesh morphing. *Proceedings of SIGGRAPH 99*, pages 343–350, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California.

44. A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. Maps: Multiresolution adaptive parameterization of surfaces. *Proceedings of SIGGRAPH 98*, pages 95–104, July 1998. ISBN 0-89791-999-8. Held in Orlando, Florida.

45. B. Lévy. Constrained texture mapping for polygonal meshes. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 417–424. ACM Press / ACM SIGGRAPH, August 2001. ISBN 1-58113-292-1.

46. B. Lévy and J.-L. Mallet. Non-distorted texture mapping for sheared triangulated meshes. *Proceedings of SIGGRAPH 98*, pages 343–352, July 1998. ISBN 0-89791-999-8. Held in Orlando, Florida.

47. P. Lindstrom and G. Turk. Fast and memory efficient polygonal simplification. *IEEE Visualization '98*, pages 279–286, October 1998. ISBN 0-8186-9176-X.

48. C. Loop and T. DeRose. Generalized b-spline surfaces of arbitrary topology. *Computer Graphics (Proceedings of SIGGRAPH 90)*, 24(4):347–356, August 1990. ISBN 0-201-50933-4. Held in Dallas, Texas.

49. M. Lounsbery, T. D. DeRose, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Transactions on Graphics*, 16(1):34–73, January 1997. ISSN 0730-0301.

50. T. Michikawa, T. Kanai, M. Fujita, and H. Chiyokura. Multiresolution interpolation meshes. In *9th Pacific Conference on Computer Graphics and Applications*, pages 60–69. IEEE, October 2001. ISBN 0-7695-1227-5.

51. D. E. Muller and F. P. Preparata. Finding the intersection of two convex polyhedra. *Theoretical Computer Science*, 7(2):217–236, 1978.

52. R. Ohbuchi, Y. Kokojima, and S. Takahashi. Blending shapes by using subdivision surfaces. *Computers & Graphics*, 25(1):41–58, February 2001. ISSN 0097-8493.

53. U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimanetal Mathematics*, 2(1):15–36, 1993.

54. K. Polthier. Conjugate harmonic maps and minimal surfaces. Technical Report Preprint No. 446, TU Berlin, SFB 288, 2000.

55. E. Praun, W. Sweldens, and P. Schröder. Consistent mesh parameterizations. *Proceedings of SIGGRAPH 2001*, pages 179–184, August 2001. ISBN 1-58113-292-1.

56. F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Texts and Monographs in Computer Science. Springer-Verlag, Berlin, Germany, 1985.

57. D. Ruprecht and H. Muller. Image warping with scattered data interpolation. *IEEE Computer Graphics & Applications*, 15(2):37–43, March 1995.

58. W. J. Schroeder. A topology modifying progressive decimation algorithm. *IEEE Visualization '97*, pages 205–212, November 1997. ISBN 0-58113-011-2.

59. W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. Decimation of triangle meshes. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):65–70, July 1992. ISBN 0-201-51585-7. Held in Chicago, Illinois.

60. T. W. Sederberg, P. Gao, G. Wang, and H. Mu. 2d shape blending: An intrinsic solution to the vertex path problem. *Proceedings of SIGGRAPH 93*, pages 15–18, August 1993. ISBN 0-201-58889-7. Held in Anaheim, California.

61. M. Shapira and A. Rappoport. Shape blending using the star-skeleton representation. *IEEE Computer Graphics & Applications*, 15(2):44–50, March 1995.

62. A. Shapiro and A. Tal. Polyhedron realization for shape transformation. *The Visual Computer*, 14(8-9):429–444, 1998. ISSN 0178-2789.

63. K. Shoemake and T. Duff. Matrix animation and polar decomposition. *Graphics Interface '92*, pages 258–264, May 1992.

64. E. H. Spanier. *Algebraic Topology*. McGraw-Hill, New York, 1966.

65. Y. M. Sun, W. Wang, and F. Y. L. Chin. Interpolating polyhedral models using intrinsic shape parameters. *The Journal of Visualization and Computer Animation*, 8(2):81–96, April-June 1997. ISSN 1049-8907.

66. T. Surazhsky and G. Elber. Matching free form surfaces. *Computers & Graphics*, 26(1):??–??, 2001. ISSN 0097-8493.

67. V. Surazhsky and C. Gotsman. Controllable morphing of planar triangulations. *ACM Transactions on Graphics*, 2(3), August 2001.

68. G. Taubin. A signal processing approach to fair surface design. *Proceedings of SIGGRAPH 95*, pages 351–358, August 1995. ISBN 0-201-84776-0. Held in Los Angeles, California.

69. G. Turk and J. O'Brien. Shape transformation using variational implicit functions. *Proceedings of SIGGRAPH 99*, pages 335–342, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California.

70. W. T. Tutte. How to draw a graph. *Proc. London Mathematical Society*, 13:743–768, 1963.

71. G. Wolberg. Image morphing: a survey. *The Visual Computer*, 14(8-9):360–372, 1998. ISSN 0178-2789.

72. G. Zigelmann, R. Kimmel, and N. Kiryati. Texture mapping using surface flattening vie multi-dimensional scaling. *IEEE Transactions on Visualization and Computer Graphics*, to appear, 2002.

73. M. Zöckler, D. Stalling, and H.-C. Hege. Fast and intuitive generation of geometric shape transitions. *The Visual Computer*, 16(5):241–253, 2000. ISSN 0178-2789.

74. D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. *Proceedings of SIGGRAPH 97*, pages 259–268, August 1997. ISBN 0-89791-896-7. Held in Los Angeles, California.

Marc Alexa / Recent Advances in Mesh Morphing 23

submitted to COMPUTER GRAPHICS *Forum* (9/2002).