4.3.1 Laplacian representation

We have decided to use a rather simple scheme, which is not invariant under rotation, scaling, and shearing. Yet, it is very stable and proved sufficient for the application intended. Let the center of mass of the neighbors of vertex i be

$$\bar{\mathbf{v}}_\mathbf{i} = \frac{1}{|\mathcal{N}(i)|} \sum_i j \in \mathcal{N}(i) v_j \tag{4.16}$$

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

$$\tilde{\mathbf{v}}_\mathbf{i} = \mathbf{v}_\mathbf{i} - \bar{\mathbf{v}}_\mathbf{i} \tag{4.17}$$

For an illustration see Figure 4.6. If we write all vertices as a vector the forward

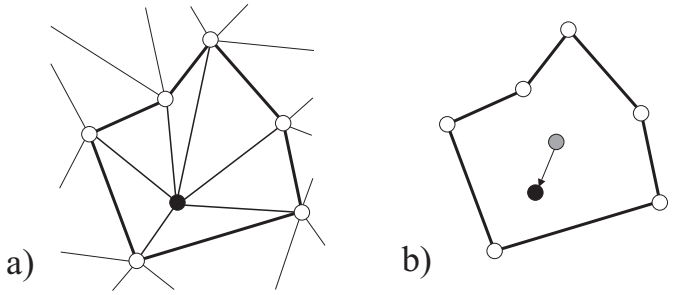

Figure 4.6: A vertex (black) and its neighborhood ring (white) in a). In Laplacian coordinates a vertex is represented by the difference to the centroid of its neighbors (b).

transformation (from absolute to relative coordinates) can be represented in matrix form. Let $A = \{a_{ij}\}$ be the adjacency matrix of the $\mathcal{M} = (K, V)$, i.e.

$$a_{ij} = \begin{cases} 1 & \{i, j\} \in K, \\ 0 & \text{else.} \end{cases}$$

and $D = \{d_{ij}\}$ 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 [Taubin 1995]. This is an important observation as it generalizes the approach to 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: Note that DA is a stochastic as well as a normal matrix. A stochastic matrix has an eigenvalue equal 1. In addition, the eigenvectors of normal matrices form a basis of full rank, meaning that all eigenvalues have multiplicity one. It follows that DA has exactly one eigenvalue equal 1 and, thus, L has exactly one eigenvalue equal 0.

The main idea of this work is to morph by linearly interpolating Laplacian coordinates rather than absolute coordinates. 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.

4.3.2 Representing transition states

Given one mesh topology (vertex-edge graph) and d vectors $V_i \in \mathbb{R}^m, i \in \{0, \ldots, d-1\}$ representing different shapes of the mesh we can compute their Laplacian representations $W_i = (I_D A)V_i$. A transition state is a matrix $T \in \mathbb{R}^{m \times d}$. The transition state defines how Laplacian coordinates are combined to form the Laplacian coordinate of the morphed mesh. Each row of T specifies weights for the linear combination of one vertex. If, for example, we want to morph between two meshes a row of T will consist of two entries, which, typically, sum up to one.

The Laplacian representation $\tilde{W}$ with respect to a particular transition state T is given by

$$\tilde{W}(i) = T(i) \begin{pmatrix} W_0^T(i) \\ \vdots \\ W_{d-1}^T(i) \end{pmatrix} \tag{4.18}$$

where $W(i)$ denotes the i-th row of W. The set of respective absolute coordinates is found by solving $(I - DA)\tilde{V} = \tilde{W}$ for $\tilde{V}$.

4.3.3 Computing absolute coordinates

Solving the equation $L\tilde{V} = \tilde{W}$ for $\tilde{V}$ is not possible in a naive way for two reasons. First, L is singular and second, typical meshes will induce matrix dimensions that make the use of explicit techniques prohibitive.

The first problem is easy to overcome. It was already shown in 4.3.1 that the solution $\tilde{V}$ is specified up to a translation. This means fixing one arbitrary vertex will lead to a linear system of equations with full rank.

A practical solution of the resulting linear system should account for the following conditions:

- L is very large and sparse, which prohibits explicit matrix techniques.
- The equation $L\tilde{V} = \tilde{W}$ has to be solved three times, i.e. for the x, y, and z vectors.
- In practice, good approximate solutions are known for $\tilde{V}$ as morphing changes the shape gradually and smoothly from one state to another. Knowing good approximate solutions calls for iterative matrix methods.

- The condition of L is 1. This causes traditional relaxation techniques to converge slowly.

We have decided to use an iterative method with pre-conditioning (see e.g. [Golub & Van Loan 1989]). Since pre-conditioning the matrix equation greatly influences convergence and has to be done only once we have decided to use incomplete LU decomposition as a pre-conditioner. The system is then solved using the conjugate gradient method.

4.3.4 Defining transitions and transition states

A transition state is represented by a matrix T as described in section 4.3.2. A morph, or transition, would be defined as a matrix T changing over time. In practice, one would define several key frames in terms of matrices. Interpolation of key frames is done in matrix space as interpolation in absolute coordinates has to be avoided.

Obviously, it is impossible to specify transition matrices by hand. We either need a user interface or automatic methods.

GUI for defining transition states

The basic requirement for a GUI is to make it easy to define a region of interest (ROI). A ROI is a part of the shape's boundary, for which the transition state will be modified. Modification is performed relative to the current overall transition state of the shape. Technically speaking, prior to the modification the shape is represented by a transition state T.

We have found it very comfortable to specify a ROI by two boundaries. An outer boundary separates the ROI from the rest of the shape. The inner boundary has to lie completely inside the region specified by the outer boundary. The inner boundary specifies the part of the region of interest, which is assigned the user defined values for this ROI. The region between inner and outer boundary is assigned values that range from the user's specification for the inner part to the old transition state. Specifically, we define a distance value d, which is zero inside the inner boundary, one outside the outer boundary, and represents the distance from the boundaries between them. A simple Dijkstra algorithm is used to compute the distances.

Assume the user specifies a new transition state $\tilde{T}$ for the ROI. The new overall transition state T' is defined for each vertex by

$$T' = (1 - d)\tilde{T} + dT \tag{4.19}$$

In a typical modification process, the user selects several ROIs and changes their respective transition states one after the other.

Spatially dependent transition states

Instead of defining the transition state of the shape explicitly using ROIs, the transition state could be defined implicitly. In particular, the transition state might be a function of absolute coordinates. This makes several interesting and important effects possible, e.g. a plane cutting the shape into parts with different transition states.

The definition of a transition state from absolute coordinates is given as a function that maps points in $\mathbb{R}^3$ to weights defining how to combine Laplacian coordinates in this point. It is assumed that the source shapes as well as the generated shape have their centers of mass at the origin of $\mathbb{R}^3$. Since, in the general case, the system of equations to be solved will be non-linear we use a simple heuristic to define a transition state. The absolute coordinates of vertices for each of the source shapes define a transition state. These transition states are averaged to yield the final transition state.

This definition has the advantage that smooth changes of the spatial distribution lead to smooth changes in the shape. For example, assume the user wants to move a plane through a shape so that the two parts correspond to two source shapes. The heuristic given above assures that moving the plane will grow one region and shrink the other, as desired.

Automatic transition sequences

The basic idea is to start the transition at one or several points of the shape and then to "grow" regions representing the target shape until the shape is "covered". In this process, the graph representing the mesh can be exploited.

Nice effects result from "flooding" the mesh, i.e. traversing the graph breadth-first and setting each visited vertex to its target coordinate.

Spatial effects can be achieved as was described in the previous section.

4.3.5 Results and applications

The main application intended for Laplacian coordinates and the presented user interface is, of course, spatially non-uniform mesh morphing. However, other modeling techniques, such as free-from modeling, could also benefit from this representation.

Spatially non-uniform morphing

We have produced several morphable meshes with the techniques described in the previous chapter.

The use of regions of interest for defining transition states and interpolating through these transition states to define an interesting morph sequence is presented in Figure 4.7. A morph sequence from the shape of an egg to the shape of a giraffe is produced. The idea was to let parts of the body pop out the egg one after the

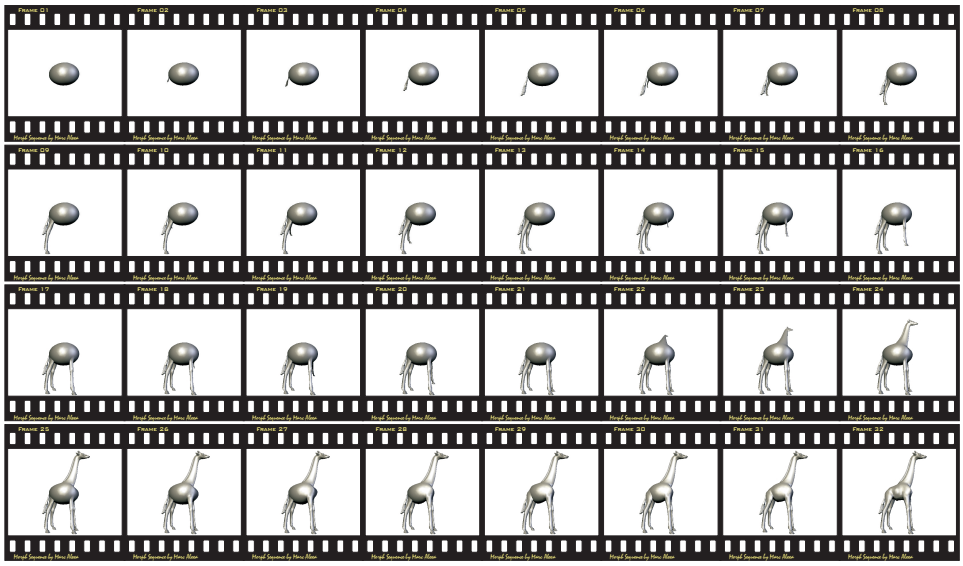

Figure 4.7: A morph sequence between an egg and a model of a giraffe genrated from several transition states. Transition states were defined using regions of interest (ROI). Each ROI corresponds to either the tail, one of the legs, the neck including the head, or the body. Eight transition states were defined, letting the different body parts of the giraffe pop out the egg one after the other.

other. This was achieved by defining ROIs for the different parts of the body and using them to specify transition states. A smooth transition sequence was produce by interpolating the key transition states.

Several examples of defining transition states from absolute coordinates are shown in Figure 4.8. A morphable model of a cow/pig is cut by a plane in two positions. Rather than a sequence several models are presented showing that spatial morph control could be used for modeling. One can imagine how many different creatures could be modeled with virtually no effort using this approach.

An explicit example of modeling using spatial morph control is depicted in Figure 4.9. The intent was to model a Pegasus, i.e. a horse with wings. One easily finds polyhedral models of a horse and an animal with wings. Using ROIs around the wings it is easy to define a transition state which yields the desired result.

In all of these examples morphing is performed between rather different shapes. An interesting application arises from locally morphing among different version of the same shape. In particular, think of different versions representing only parts of the spectrum of the mesh. The eigenvectors of the Laplacian L form a basis, which is the equivalent to a Fourier basis of a mesh [Karni & Gotsman 2000]. This basis could be exploited to define several band-limited versions of the shape. Locally morphing among these shapes has the effect of a local spectral filter.

4.3.6 Free-from modeling

Laplacian coordinates are not only useful for morphing. They can easily be used for free from modeling of meshes by exploiting that they describe the relative position of vertices. The general idea is this:

1. Compute Laplacian coordinates from the original model.
2. Specify the absolute coordinates of several vertices.
3. Compute a linear least squares solution to find the absolute coordinates of the free vertices.

In order to specify fixed vertices we found it convenient to use ROIs, again, defined by an inner and outer boundary. Vertices outside the outer boundary are fixed to their original position. Vertices inside the inner boundary can be moved by the user. The remaining vertices are free and will be computed using the linear least squares approach.

An examples is depicted in Figure 4.10. In the face of a young boy the nose tip is displaced. The remaining vertices are relaxed to approximately fit their Laplacian coordinates using varying sets of free vertices to achieve different effects.

4.3.7 Conclusions

We have presented a well defined method to allow mesh morphing in a spatially non-uniform way. The main idea is to use differential coordinates. Specifically, Laplacian coordinates are introduced, which seem to be well suited for the task of modeling. The advantages of Laplacian coordinates are that they are independent of transformations of the shape and rigorously defined.

However, Laplacian coordinates are sensitive to scaling and rotation of the shape. This can be a problem if corresponding regions of shapes have different size or orientation - despite the fact that shapes are overall reasonably aligned. A representation of coordinates as an affine sum of neighboring vertices would be insensitive to affine transforms. In the future we will try to explore this idea and overcome the problem that neighboring vertices might not be a base of $\mathbb{R}^3$.

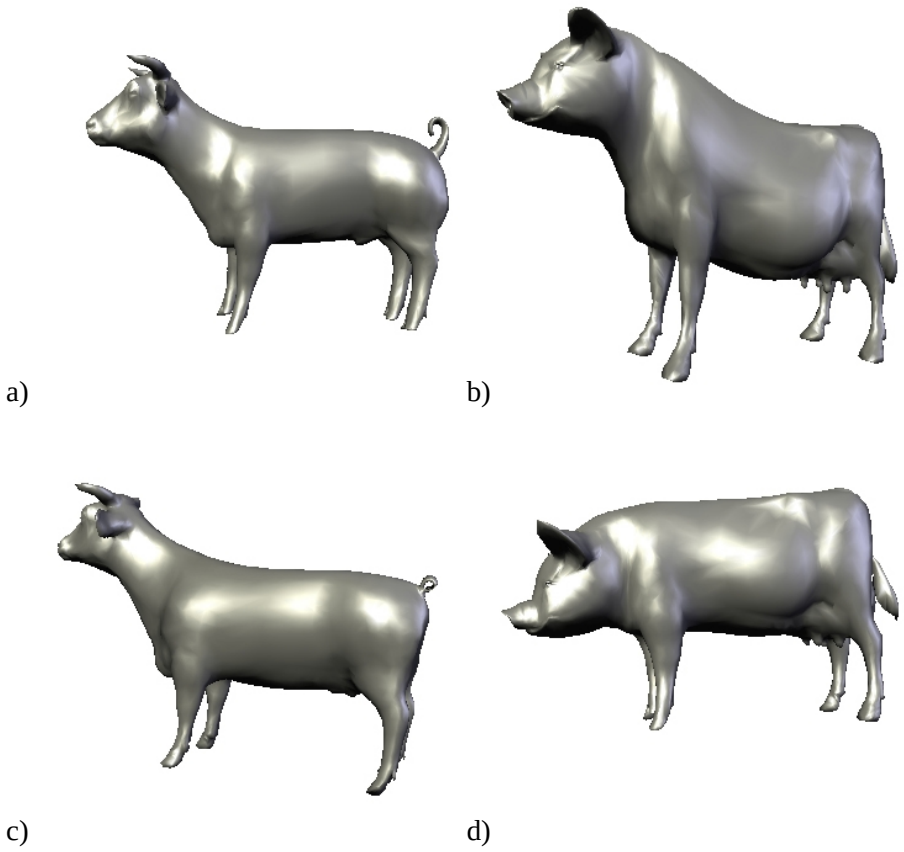

Figure 4.8: Defining the transition state using spatial constraints. Here, a plane is used to split the shape into two parts corresponding to one of two source shapes (representing the models of a cow and a pig). In a) and b) the plane separates head including neck from the body of the animals. In c) and d) the plane is approximately in the middle of the body.

Figure 4.9: Using spatial control for modeling. A Pegasus is modeled by morphing between models comprising its features. Local morph control is used to define a transition state so that only the wings of the duck appear on the horse.

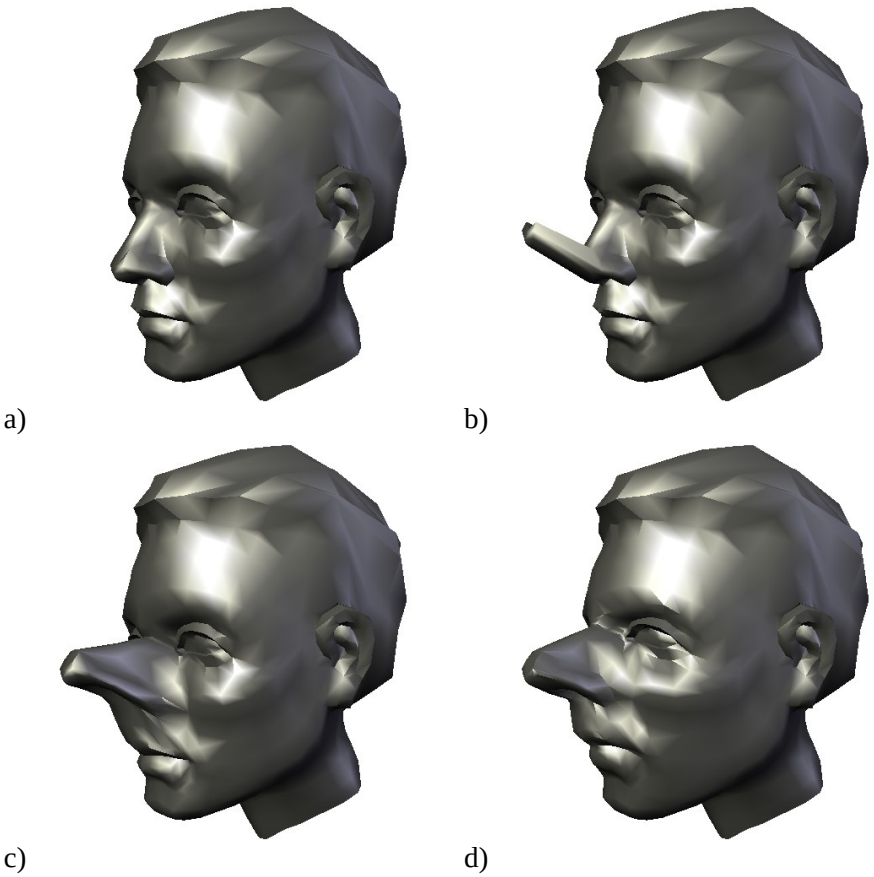

Figure 4.10: Using Laplacian coordinates for free-form modeling. The model of a boy?s face a) is deformed. The nose tip is displaced, b) is showing the displaced vertices. The shapes in c) and d) result from defining a set of free vertices, which are least-square fitted to their Laplacian coordinates.

4.4 Interpolation using isomorphic dissections

Sederberg et al. [1993] introduced techniques that minimize the deformation of the boundaries. Shapira & Rappoport [1995] suggested 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 represented the interior of the 2D shapes by compatible skeletons and applied the blend to the parametric description of the skeletons. The automatic creation of corresponding equivalent skeletons of two shapes is involved, and though theoretically possible for all shapes, it seems natural for similar shapes, but ambiguous for rather different shapes like the letters **U** and **T**.

Here, we present an object-space morphing technique that blends the interior of the shapes rather than their boundaries to achieve a sequence of in-between shapes which is locally least-distorting. Assuming that a boundary vertex correspondence of the source and target shapes is given, we apply an algorithm for dissecting the source and target shapes into isomorphic simplicial complexes, i.e. triangles or tetrahedra. Then, we develop a method for interpolating the locations of corresponding vertices, both boundary and interior, along their paths from the source to the target object.

Simplicial complexes allow the local deformation of the shapes to be analyzed and controlled. Floater and Gotsman have used barycentric coordinates to morph compatible triangulations with convex boundary so that no triangles flip on their way from the source to the target configuration [Floater & Gotsman 1999]. However, interpolation of barycentric coordinates is not motivated by or related to physical or esthetic principles.

We start by determining an optimal least-distorting morphing between a source simplex and a target simplex (triangles in the 2D case and tetrahedra in the 3D case). Then, the general idea is to find a transformation which is locally as similar as possible to the optimal transformation between each pair of corresponding simplices.

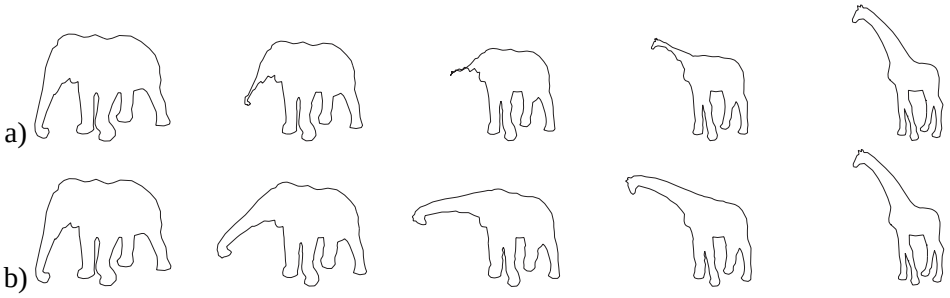

Figure 4.11: Contour blends of the elephant-giraffe example. Simple linear vertex interpolation in (a) vs. as-rigid-as-possible shape interpolation in (b).

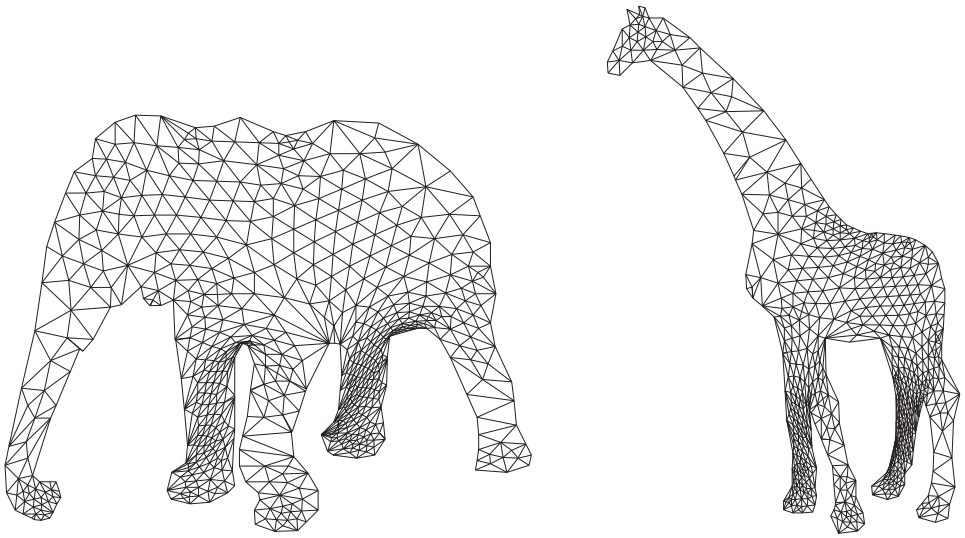

Figure 4.12: The homeomorphic dissections of the shapes in the elephant-giraffe example

4.4.1 Isomorphic dissections of shapes

In this section, we construct isomorphic dissections given two shapes in boundary representation. We assume that the correspondence of the boundaries has been established, i.e. a bijective map between boundary vertices is given. For polygons, reasonable correspondence can be found automatically [Sederberg & Greenwood 1992; Cohen et al. 1997]. In difficult cases, few correspondences could be specified manually and the remaining vertices are matched automatically. For polyhedral objects, several techniques exist, which are based on topological merging introduced by Kent et al. [1992]. Recent work [Gregory et al. 1998; Lee et al. 1999] also allows the specification of corresponding features which seems sufficient to produce acceptable results for a variety of polyhedral models.

Polygons

The problem of constructing a common triangulation for two given polygons is discussed in the literature as *compatible triangulation* [Aronov et al. 1993]. Triangulating a single polygon π is possible using only the vertices of the polygon (e.g. [Chazelle 1990]). However, this is usually not possible for two different polygons. Aronov et al. [1993] show how to triangulate two polygons in a compatible way if at most $O(n^2)$ additional vertices (so-called Steiner points) are allowed. The general scheme [Aronov et al. 1993] is to first triangulate each polygon independently. Then, both polygons are mapped to a regular n-gon so that corresponding boundary vertices coincide. The compatible triangulation is established by overlaying the two edge sets in the convex n-gon. The resulting new interior vertices are then

mapped back into the original polygons, yielding compatible triangulations of the source and target shapes.

We would like to stress that the quality of the blend, in terms of the quality of the in-between shapes, strongly depends on the shape of the simplices. In particular, skinny triangles (or tetrahedra in 3D) cause numerical problems. Thus, in the following, we describe how this scheme can be enhanced to yield compatible triangulations with a significantly better triangle shape.

First, we apply Delaunay triangulations (see any textbook on Computational Geometry, e.g. [de Berg et al. 1997]) as the initial triangulation since Delaunay triangulations maximize the minimum interior angle and, thus, avoid skinny triangles. Of course, any skinny triangle in the independent triangulations is inherited by the merged triangulation. Moreover, Delaunay triangulations are unique, and similar regions in the shapes will result in similar triangulations. Thus, skinny triangles resulting from the overlay process can be avoided.

Nevertheless, the merged triangulations still have skinny triangles, and further enhancement is required to avoid numerical problems. We optimize the triangulations by further maximizing the minimum interior angle, which is known to be a reasonable triangulation quality criterion (see e.g. [de Berg et al. 1997]). We use two independent operations:

1. Moving interior vertices. Freitag et al. [1999] show how to find vertex positions which maximize the minimum angle for a given triangulation.

2. Flipping interior edges simultaneously in both triangulations. This procedure follows the edge flip criteria used in Delaunay triangulation. Given that an edge flip is legal in both triangulations, it is performed if the operation increases the overall minimum angle.

The above two operations are applied in turn until no valid flips are necessary. Convergence is assured since each step can only increase the minimum angle. We call this procedure *compatible mesh smoothing*. The smoothing step optimizes the compatible triangulations without changing the vertex count.

However, we also consider changing the vertex count by means of splitting edges. The split operation is well-defined in terms of topology, if it is applied to both triangulations simultaneously, the isomorphy remains. The idea is to split long edges to avoid long skinny triangles. Splitting edges according to their lengths does not guarantee an increase in triangle quality. In practice, smaller triangles are more likely to be improved by the smoothing step. After each edge split, the triangulations are smoothed. This avoids the generation of edges in regions where the smoothing operation would produce nicely-shaped triangles. Figure 4.14 illustrates the results of splitting edges, as well as of the smoothing process.

4.4.2 Polyhedra

To the best of our knowledge, the three-dimensional analog to compatible triangulations has not been discussed in the literature. Work has been done to dissect

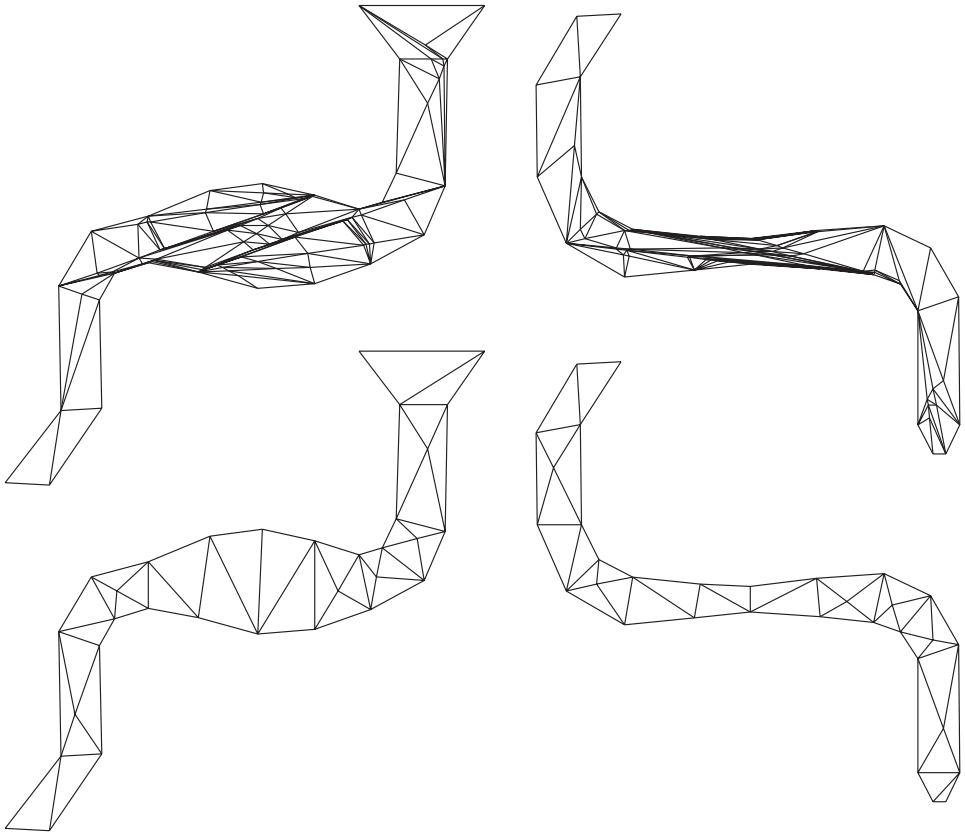

Figure 4.13: A comparison of compatible triangulations. The upper row shows triangulations generated from using ear-capping for the initial triangulation step. Initial triangulations are overlaid on a convex domain to produce compatible triangulations. The triangulations in the lower row were generated with the same general procedure, but using initial Delaunay triangulations. Far fewer triangles are induced, since Delaunay triangulations yield similar partitioning for similar regions.

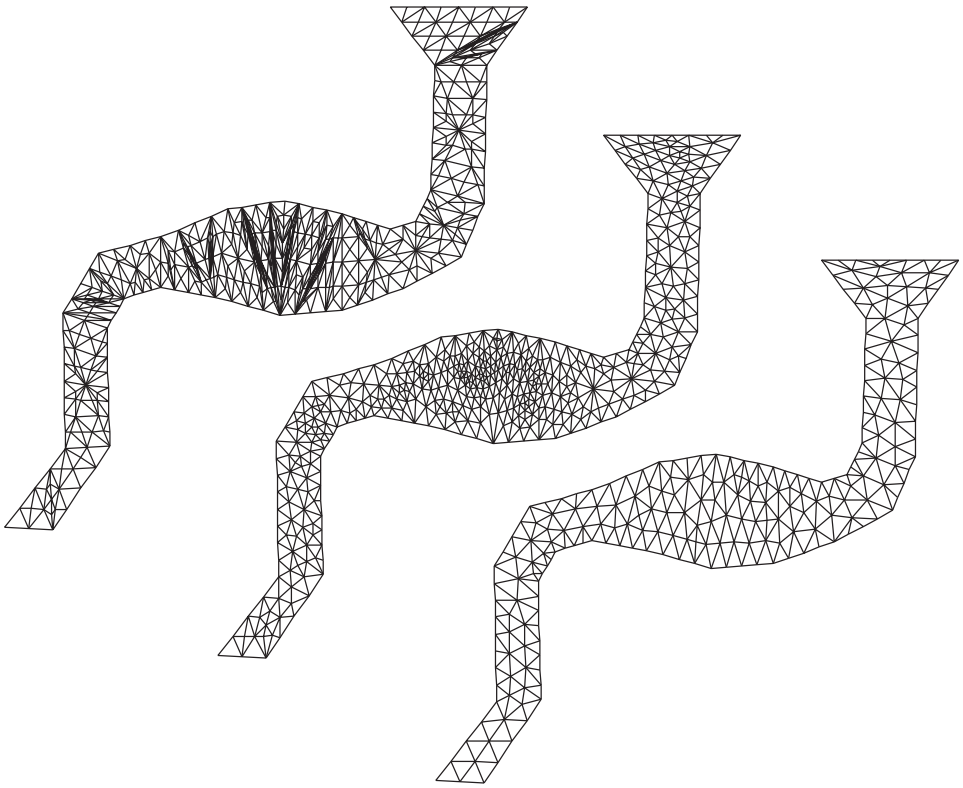

Figure 4.14: The mesh refinement process. In the first row, the merged Delaunay triangulations from Figure 4.13 are refined by edge splits until all edge lengths are bounded. The second row shows the result of compatible mesh smoothing on this triangulation. The third row shows the actual technique, where splitting and smoothing is performed concurrently. Note that the edge length bound is the same in the first and third row.

polyhedra into simplicial complexes, a process referred to as tetrahedralization. However, the work of Aronov et al. [1993] can be extended to genus 0 polyhedra.

First, the source and target polyhedra are tetrahedralized independently using common techniques, e.g., [Chazelle & Palios 1989]. Then, the tetrahedralizations are mapped to a corresponding convex shape. We as well as Shapiro & Tal [1998] describe methods to map an arbitrary genus 0 polyhedron to a convex shape. Since the source and target polyhedra are assumed to have the same vertex-edge topology and the convexification process is deterministic, the polyhedra are mapped to the same convex shape. The interior vertices of their tetrahedralizations are mapped using barycentric coordinates. The fact that vertices are mapped to a convex shape using barycentric coordinates for interior vertices assures that no tetrahedra will be flipped. Then, an overlay of the two tetrahedralizations is computed, where faces are cut against faces, resulting in new edges. Note that the intersection of two tetrahedra results in four-, five-, or six-sided convex shapes, which are easy to

tetrahedralize. The resulting structure is mapped back into original polyhedra. In case the source and target shapes are not genus 0, they have to be cut into genus 0 pieces which are independently treated as explained above.

4.4.3 Transforming shapes

Given two objects together with a set of point-to-point correspondences between user-defined control (anchor) points, one can define an elastic transformation between the objects that exactly satisfies the correspondences. However, to reduce the distortion of the in-between shapes, it is advisable to determine the *rigid* part of the transformation and interpolate it separately from the *elastic* part [Cohen-Or & Carmel 1998; Cohen-Or et al. 1998; Zhang 1996]. The rotational component of the rigid part should be interpolated so that the object is non-deforming, e.g. using quaternion interpolation [Shoemake & Duff 1992]. The rigid-elastic decomposition of the warp function and its particular interpolation are so chosen to minimize the distortion of the intermediate shapes. The rigid part performs the general positional changes, while the fine details are gradually changed by the elastic part.

In many applications, this decomposition does improve the morphing results, though it cannot prevent local distortions in cases of body movements which are more involved as may be found in articulated objects. The underlying assumption in [Cohen-Or & Carmel 1998; Cohen-Or et al. 1998; Zhang 1996] is that the movement can roughly be approximated by rotation, stretching and translation. If we consider objects such as animals' bodies or sophisticated mechanical objects, such as industrial robots, it is clear that even the simplest movements cannot be well approximated by a single rotation and translation. To reduce distortions in transformations of bodies comprising local rotations, the decomposition should be more elaborate. The idea is to determine local non-distorting motions rather than a global one. The composed shape morphing should behave locally as close as possible to the ideal local ones. Figure 4.11 shows a blend between an elephant and a giraffe. The two shapes are aligned and a single rotation cannot prevent the distortions of a linear interpolation, whereas the locally least-distorting interpolation yields a pleasing blend of such articulated objects.

Based on a compatible dissection of the interiors of the shapes (see Figure 4.12), we first define local affine transformations. Each of the local linear maps can be separately decomposed into a rotation and a stretch. Thus, locally, we know how to achieve a non-distorting morph. Then, these local transformations are composed into a global coherent non-distorting transformation, which minimizes the overall local deformation. It should be noted that our transformation is (globally) rigidly reducible; that is, if there is a single rigid transformation that aligns the objects, the morph follows such a path.

We only consider simplicial complexes as dissections of shapes. Specifically, a two-dimensional shape is a *polygon* and its dissection a *triangulation*, and a three-dimensional shape is a *polyhedron* and its dissection a *tetrahedralization*. In the following, we introduce an interpolation technique for determining vertex paths

in shape blending, given a source and a target shape represented by homeomorphic (compatible) triangulations. In Section 4.4.1, we show how to compute such homeomorphic dissections from boundary representations. Note that we describe the concept of determining the vertex paths in two dimensions for clarity; the extension to three or more dimensions is straightforward.

4.4.4 Least-distorting triangle-to-triangle morphing

Suppose the triangulation of the source and target shapes consists of only one triangle each. Let the source vertices be $P = (\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3})$ and the target vertices be $Q = (\mathbf{q_1}, \mathbf{q_2}, \mathbf{q_3})$, where vertices with the same index correspond. An affine mapping represented by matrix A transforms P into Q:

$$A\mathbf{p_i} + \mathbf{l} = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} \mathbf{p_i} + \begin{pmatrix} l_x \\ l_y \end{pmatrix} = \mathbf{q_i}, \quad i \in \{1, 2, 3\} \tag{4.20}$$

We do not take the translation $\mathbf{l}$ into account for shape interpolation since it does not describe any property of the shape itself except for its placing in the scene. Rather, we want to describe intermediate shapes by varying the rotational and scaling parts comprising A, over time. Note that the coefficients of A are linear in the coordinates of the target shape.

$$\begin{aligned}
a_1 &= \frac{q_{1_x}(p_{2_y}-p_{3_y})+q_{2_x}(p_{3_y}-p_{1_y})+q_{3_x}(p_{1_y}-p_{2_y})}{p_{1_x}p_{2_y}+p_{2_x}p_{3_y}+p_{3_x}p_{1_y}-p_{1_y}p_{2_x}-p_{2_y}p_{3_x}-p_{3_y}p_{1_x}} \\
a_2 &= -\frac{u_{1_x}(p_{2_x}-p_{3_x})+q_{2_x}(p_{3_x}-p_{1_x})+q_{3_x}(p_{1_x}-p_{2_x})}{p_{1_x}p_{2_y}+p_{2_x}p_{3_y}+p_{3_x}p_{1_y}-p_{1_y}p_{2_x}-p_{2_y}p_{3_x}-p_{3_y}p_{1_x}} \\
a_3 &= \frac{u_{1_y}(p_{2_y}-p_{3_y})+q_{2_y}(p_{3_y}-p_{1_y})+q_{3_y}(p_{1_y}-p_{2_y})}{p_{1_x}p_{2_y}+p_{2_x}p_{3_y}+p_{3_x}p_{1_y}-p_{1_y}p_{2_x}-p_{2_y}p_{3_x}-p_{3_y}p_{1_x}} \\
a_4 &= -\frac{u_{1_y}(p_{2_x}-p_{3_x})+q_{2_y}(p_{3_x}-p_{1_x})+q_{3_y}(p_{1_x}-p_{2_x})}{p_{1_x}p_{2_y}+p_{2_x}p_{3_y}+p_{3_x}p_{1_y}-p_{1_y}p_{2_x}-p_{2_y}p_{3_x}-p_{3_y}p_{1_x}}
\end{aligned} \tag{4.21}$$

Intermediate shapes $V(t) = (\mathbf{v_1}(t), \mathbf{v_2}(t), \mathbf{v_3}(t))$ are described as $V(t) = A(t)P$, where $A(t)$ is defined as explained in section 4.1.1.

Note that the rotation of the triangle does not contribute to its shape. However, this is no longer true for more than a single triangle, as we shall see in the next section, which discusses the generalization to more than one triangle.

4.4.5 Closed-form vertex paths for a triangulation

We now consider a triangulation $\mathcal{T} = \{T_{\{i,j,k\}}\}$ rather than a single triangle. Each of the source triangles $P_{\{i,j,k\}} = (\mathbf{p_i}, \mathbf{p_j}, \mathbf{p_k})$ corresponds to a target triangle $Q_{\{i,j,k\}} = (\mathbf{q_i}, \mathbf{q_j}, \mathbf{q_k})$. For each pair of triangles, we compute a mapping $A_{\{i,j,k\}}$, which can be factored by Eq. 4.5 to determine $A_{\{i,j,k\}}(t)$. Since most of the vertices correspond to more than one triangle, a mapping of all vertices could not (in general) be conforming with all the individual ideal transformations $A_{\{i,j,k\}}(t)$.
Let

$$V(t) = (\mathbf{v_1}(t), \ldots, \mathbf{v_n}(t)), t \in [0, 1] \tag{4.22}$$

be the desired paths of the vertices, satisfying

$$\begin{aligned} V(0) &= (\mathbf{p_1}, \ldots, \mathbf{p_n}) \\ V(1) &= (\mathbf{q_1}, \ldots, \mathbf{q_n}). \end{aligned}$$

We define $B_{\{i,j,k\}}(t)$ to be the matrix in the affine transformation from $P_{\{i,j,k\}}$ to $\mathbf{v_i}(t), \mathbf{v_j}(t), \mathbf{v_k}(t)$, i.e.

$$B_{\{i,j,k\}}(t)\mathbf{p_f} + \mathbf{l} = \mathbf{v_f}(t), \quad f \in \{i,j,k\} \tag{4.23}$$

Note that the coefficients of $B_{\{i,j,k\}}(t)$ are linear in $\mathbf{v_i}(t), \mathbf{v_j}(t), \mathbf{v_k}(t)$. We define an intermediate shape $V(t)$ as the vertex configuration which minimizes the quadratic error between the actual matrices $B_{\{i,j,k\}}(t)$ and the desired ones $A_{\{i,j,k\}}(t)$. This quadratic error functional is expressed as

$$E_{V(t)} = \sum_{\{i,j,k\} \in \mathcal{T}} \left\| A_{\{i,j,k\}}(t) - B_{\{i,j,k\}}(t) \right\|^2, \tag{4.24}$$

where $\| \cdot \|$ is the Frobenius norm. In this expression, the $A_{\{i,j,k\}}(t)$ are known for a fixed time t and each $B_{\{i,j,k\}}$ is linear in the $\mathbf{v_i}(t), \mathbf{v_j}(t), \mathbf{v_k}(t)$. Thus, $E_{V(t)}$ is a positive quadratic form in the elements of $V(t)$. In order to have a unique minimizer to $E_{V(t)}$, we should predetermine the location of one vertex, say $v_{1_x}(t), v_{1_y}(t)$, for example, by linear interpolation.
The functional $E_{V(t)}$ can be expressed in matrix form, setting

$$u^T = (1, v_{2_x}(t), v_{2_y}(t), \ldots, v_{n_x}(t), v_{n_y}(t))$$

as

$$E_{V(t)} = u^T \begin{pmatrix} c & G^T \\ G & H \end{pmatrix} u, \tag{4.25}$$

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:

$$H \begin{pmatrix} v_{2_x}(t) \\ v_{2_y}(t) \\ \vdots \end{pmatrix} = -G \tag{4.26}$$

Note that H is independent of t. This means we can invert H and find solutions for time t by computing the corresponding $G(t)$ and a single matrix multiplication:

$$V(t) = -H^{-1}G(t) \tag{4.27}$$

In practice, we compute the LU decomposition of H and find $V(t)$ by back substitution. Furthermore, the computations are separable and are performed independently for the two coordinates. Note that only G depends on the dimension,

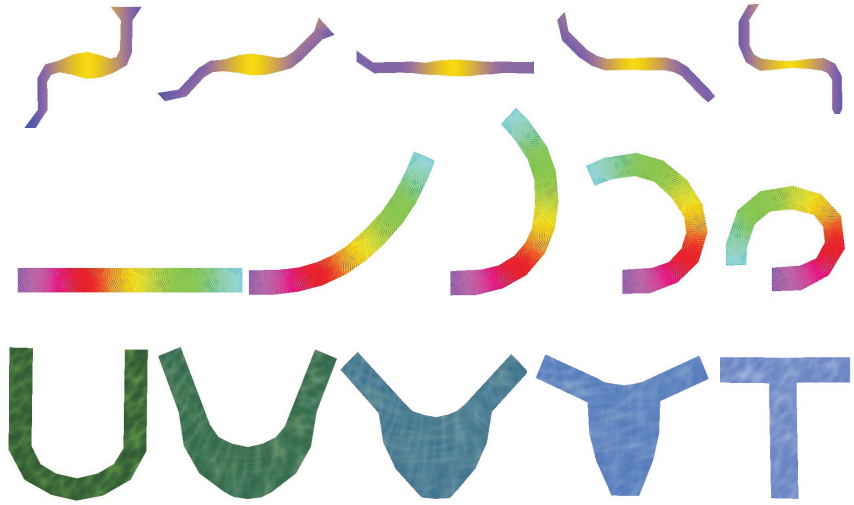

Figure 4.15: Transformations of different shapes representing solid objects. Note that parts of the shapes transform rigidly whenever possible. The lowest row shows an example where the shapes have no obvious common skeleton.

while H is the same for the x and y components. Thus, H is effectively of size $n-1 \times n-1$, which means the dominating factor of the computation is independent of the dimension.

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

4.4.6 Symmetric solutions

While we were satisfied with the degree of symmetry the previously explained approach exhibited in most of our test cases, a symmetric solution can be advantageous – in particular, if the corresponding triangles in the source and target shapes have largely differing area. We can make the solution symmetric by simply blending the optimization problems from both directions. Let $A_f^{\rightarrow}(t)$ be the affine transformation of triangle f from source to intermediate shape at time t, and $A_f^{\leftarrow}(t)$

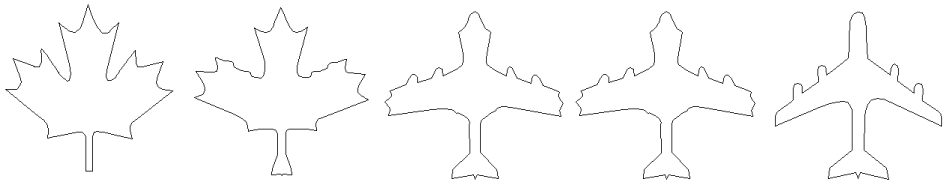

Figure 4.16: The contour of a maple leaf blended with a plane using as-rigid-as-possible shape interpolation. Note that the features of the plane grow out of contour according to the current direction of wings and not their final position.

the respective transformation coming from the target shape. Similarly, we define $B_f^{\rightarrow}(t)$ and $B_f^{\leftarrow}(t)$. We define intermediate $E_{V(t)}$, the vertex configuration at time t, by

$$E_{V(t)} = (1-t)E_{V(t)}^{\rightarrow} + tE_{V(t)}^{\leftarrow} \tag{4.28}$$

where

$$E_{V(t)}^{\rightarrow} = \sum_{f \in \text{ Tri}} \left\| A_f^{\rightarrow}(t) - B_f^{\rightarrow}(t) \right\|^2 \tag{4.29}$$

$$E_{V(t)}^{\leftarrow} = \sum_{f \in \text{ Tri}} \left\| A_f^{\leftarrow}(1-t) - B_f^{\leftarrow}(1-t) \right\|^2 \tag{4.30}$$

With this definition, we lose the advantage of only one matrix inversion for given source and target shapes. Instead, every time t requires the solution of a linear system of equations. Whether the computation times are acceptable depends on the shapes and the desired application.

4.4.7 Results and Conclusion

We have applied the techniques explained above to various inputs. The two-dimensional shapes are generated by extracting a contour out of an image. For the correspondence of contours, we defined manually several vertex-to-vertex correspondences, while the remaining vertices were automatically aligned. The resulting polygons were dissected as described above. In Figures 4.17, 4.18, 4.19, and 4.23, the triangulations were used to map a texture to the shape (as was suggested by Tal & Elber [1999]). Textures were extracted with the contours from the source images. More elaborate techniques for space-time control (e.g. [Lee et al. 1995]) could be easily integrated in our work to give the user more control as to what is transformed and when. Also note that the techniques are not restricted to simple polygons.

Since our technique interpolates shapes "naturally" in the sense that it preserves parts that just change relative position or orientation, it could be also used to extrapolate beyond the source and target shapes. Figure 4.20 demonstrates this with the example of Leonardo DaVinci's studies on proportions (see Figure 4.23

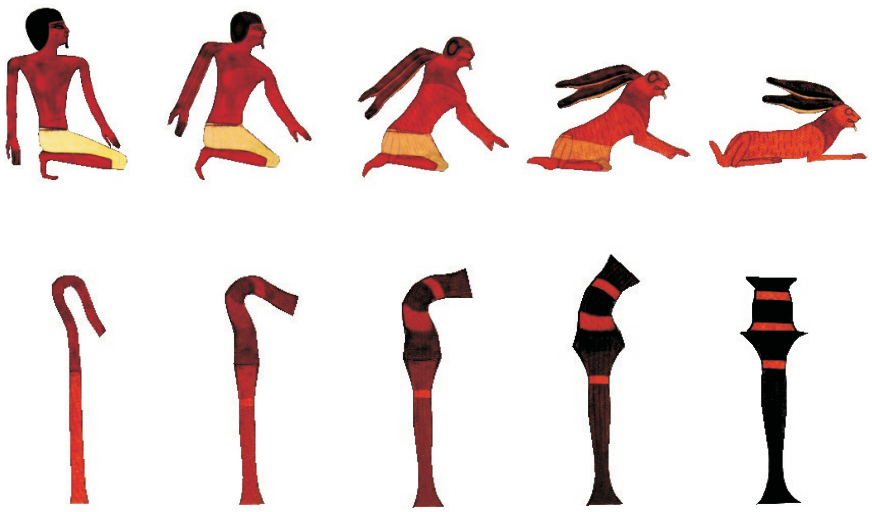

Figure 4.17: Morphs between Egyptian art pieces using textures from the original images. Contours are blended using as-rigid-as-possible shape interpolation and texture colors are linearly interpolated.

Figure 4.18: Contour blend of a penguin and a dolphin using only the texture of the penguin.

Figure 4.19: Morph between photographs of a tiger and a dinosaur.

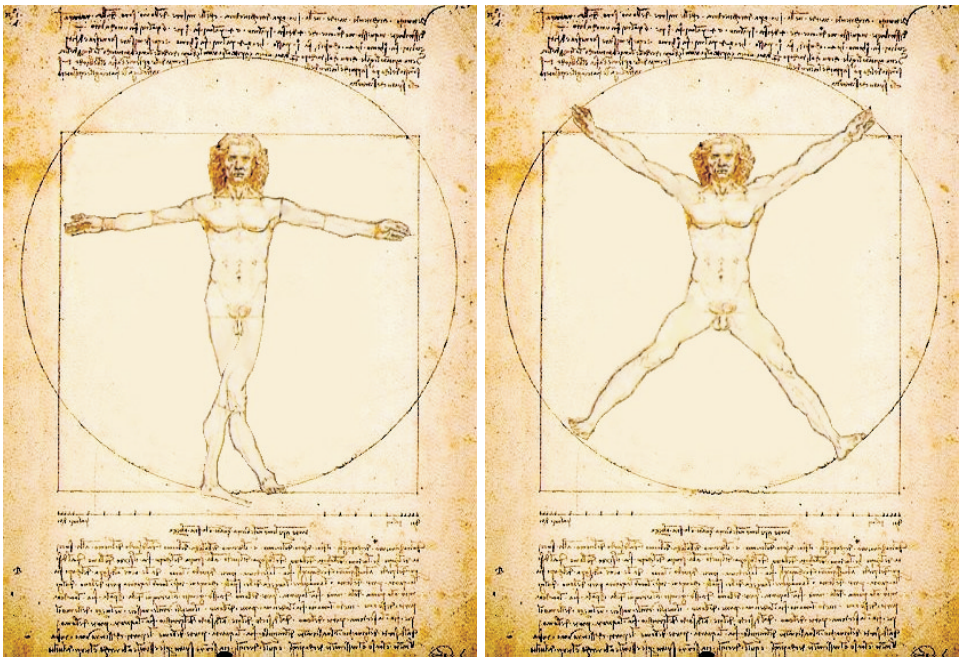

Figure 4.20: Shape extrapolation. Using as-rigid-as-possible shape interpolation, shapes can be naturally extrapolated beyond the source and target shapes. The images show the human figure of Leonardo DaVinci's proportions at time values -0.5 and 1.5.

for the interpolation). We can generate shapes for time values -0.5 and 1.5 while preserving the proportions of the human figure.

We have also applied the interpolation technique to three-dimensional models. The examples in Figure 4.21 were generated by using deformed versions of a polyhedral model. Note the difference between linear vertex interpolation (upper row) and as-rigid-as-possible interpolation (lower row). In Figure 4.22, morphable polyhedral models were generated using topological merging. As in the two-dimensional case, the vertex paths result from defining transformations for each pair of corresponding tetrahedra by factoring the affine transform into rotational and stretching components and, then, minimizing the deviation from these ideal transformations.

The current implementation seems to be robust and fast. The most time-consuming step is optimizing triangle shape. Without optimizing triangle shape numerical problems are likely to occur. In all our examples no simplex changed orientation (i.e. flipped), however, we have not been able to prove this to be a property of our approach.

The examples clearly demonstrate the superior quality of our approach compared to plain linear vertex interpolation. Additionally, it offers the possibility to texture the shapes, so that shape blending becomes applicable to images. In turn,

Figure 4.21: A simple example of three-dimensional objects. The difference of linear and as-rigid-as-possible vertex interpolation is demonstrated on a bent cigar-like shape.

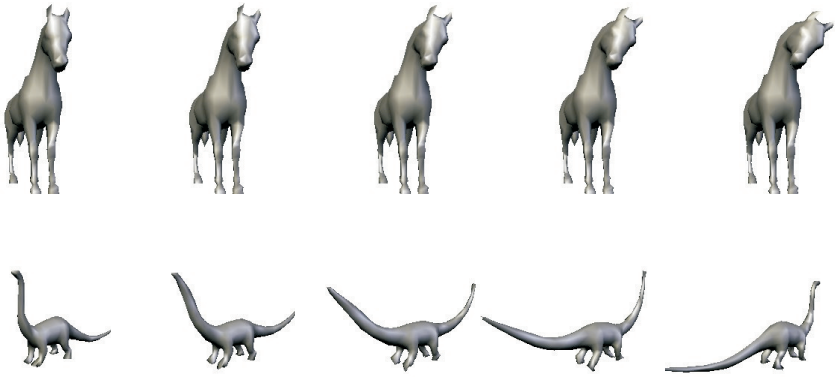

Figure 4.22: Our technique is also useful to mimic motions of articulated three-dimensional objects in case the underlying skeleton is missing, as demonstrated for a horse turning its head. The example in the lower row was produced using a polyhedral morphing technique (facilitating topological merging). Note that the lengths of the tails/necks are preserved.

traditional image morphing techniques could serve to generate the homeomorphic dissections of the shapes and, thus, make use of more advanced vertex/pixel interpolation technique(s). However, the quality of a morph lies in the eye of the beholder. Nevertheless, there is a clear consensus that - lacking other information - the geometry along the morph sequence should change monotonically with no superfluous distortions. The idea of as-rigid-as-possible shape interpolation is to avoid distortions as much as possible and let angles and scales change linearly. We believe that this captures the notion of the above-mentioned consensus.

Despite this, shape blending is always an esthetic problem and no automatic method will meet the needs that arise in different applications. Consequently, user interaction will always be necessary to produce the desired results. Nevertheless, we believe that more elaborate methods for shape blending simplify and minimize the involvement of the designer.

Finally, we want to mention that dissections of shapes seem to extend the concept of skeletons while fully capturing their information. Dissections are more powerful in representing the mechanics of shapes as they allow fine grained analysis of local behavior. In many cases, shapes naturally have no skeleton or their metamorphoses could not be described in terms of a skeleton. These benefits come along with easier and less ambiguous computation of dissections as compared to skeletons.