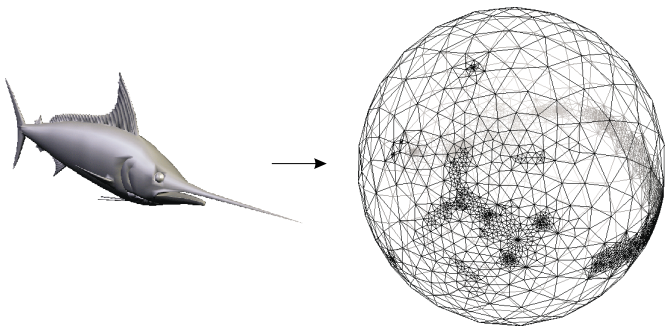
Chapter 2

Correspondence of shapes



In this chapter we aim at finding corresponding vertex positions on two or more shapes. We assume the boundaries of the shapes to be manifolds and homeomorphic (as long as not noted otherwise). Given two manifold 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 automatically generated feature correspondences (i.e. vertex-vertex correspondences). Depending on the approach chosen, this is done by re-parameterization 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.

$$\begin{array}{ccc} \{i\} \in K_0 & \xrightarrow{W_0} & \phi_{V_1}(B_0) \\ {\scriptstyle \phi_{S_0}}\downarrow & & \uparrow{\scriptstyle \phi_{S_1}^{-1}} \\ \mathbb{S}^2 & \xrightarrow[f]{} & \mathbb{S}^2 \end{array}$$

The main problems in this approach are to compute the vertex coordinates S_0, S_1 on the sphere and the re-parameterization 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 parameter 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.

$$\begin{array}{ccc} \{i\} \in K_0 & \xrightarrow{W_0} & \phi_{V_1}(B_0) \\ {\scriptstyle \phi_{L_0}}\downarrow & & \uparrow{\scriptstyle \phi_{L_1}^{-1}} \\ |L| & \xrightarrow[f]{} & |L| \end{array}$$

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 re-parameterization for feature alignment is introduced. Finally, some comments on rarely mentioned details in the correspondence problem are given.

2.1 Parameterizing 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 parameterization 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 [Hormann & Greiner 2000; Lévy & Mallet 1998; Zigelmann et al. 2002].

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 continuous 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 [Hormann & Greiner 2000; Lévy & Mallet 1998] or are of algorithmic nature (e.g. Gregory et al. [1999], 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 $0 \leq i < n$ and the fixed boundary vertices have indices $n \leq i < N$. We aim at finding positions $\mathbf{w_i}$ in the plane with $|\mathbf{w_i}| = 1, n \leq i < N$. The mapping is bijective if and only if no edges cross or, alternatively,

$$\forall (i,j,k) \in K. \det \begin{pmatrix} \mathbf{w_{i_x}} & \mathbf{w_{i_y}} & 1 \\ \mathbf{w_{j_x}} & \mathbf{w_{j_y}} & 1 \\ \mathbf{w_{k_x}} & \mathbf{w_{k_y}} & 1 \end{pmatrix} > 0. \tag{2.1}$$

However, this quadratic expression is awkward to use as a criterion to guarantee that the planar embedding is valid, which is why most approaches resort to the more restrictive 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 approach of Gregory et al. [1998, 1999] is explained.

Barycentric mapping

Tutte [1963] 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:

$$\mathbf{w_i} = \sum_{j \in \mathcal{N}(i)} \frac{1}{|\mathcal{N}(i)|} \mathbf{w_j} \tag{2.2}$$

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

$$\lambda_{i,j} = \begin{cases} d_i^{-1} & \{i,j\} \in K \\ 0 & \{i,j\} \notin K \end{cases} \tag{2.3}$$

this can be written as the mentioned system of linear equations

$$(I - \Lambda) \begin{pmatrix} \mathbf{w_0} \\ \mathbf{w_1} \\ \cdots \\ \mathbf{w_{n-1}} \end{pmatrix} = \begin{pmatrix} \sum_{i=n}^{N-1} \lambda_{0,i} \mathbf{w_i} \\ \sum_{i=n}^{N-1} \lambda_{1,i} \mathbf{w_i} \\ \cdots \\ \sum_{i=n}^{N-1} \lambda_{n-1,i} \mathbf{w_i} \end{pmatrix} \tag{2.4}$$

The matrix $(I - \Lambda)$ has full rank and, thus, there is exactly one solution. Tutte [1963] 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.

Shape preserving parameterization

In the barycentric mapping the weights λ contain only topological information. Floater [1997] 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

$$\mathbf{w_i} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \sum_{j \in \mathcal{N}(i)} \lambda_{i,j} \mathbf{w'_j} \tag{2.5}$$

and

$$1 = \sum_{j \in \mathcal{N}(i)} \lambda_{i,j}. \tag{2.6}$$

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 possible 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

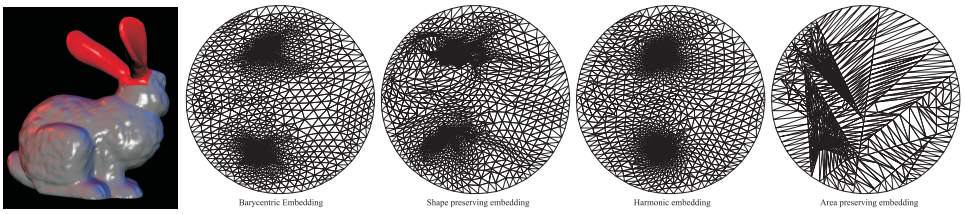


Figure 2.1: A part of a mesh parameterized on the unit disk using different mapping techniques. The original geometry is highlighted in red. A barycentric mapping (see Section 2.1) does not reflect the geometry of the mesh. The shape preserving embedding tries to capture the local shape of the mesh by locally approximating conformal maps (see Section 2.1). Discretized harmonic embeddings minimize metric distortion (see Section 2.1). The area preserving embedding is a recursive process, which aims at approximating the original area of triangles (see Section 2.1).

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:

$$\lambda_{i,j} = \frac{1}{|\mathcal{N}(i)|} \sum_k \lambda_{i,j}(k) \tag{2.7}$$

The positions w_i are computed by solving (2.4). Recently, Floater [2002, 2001] has proven a generalization of Tutte's theorem, which shows that it is sufficient that each vertex is a convex combination of its neighbors.

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 Dirichlet energy

$$E_D(u) = \frac{1}{2} \int_\Omega |\nabla u|^2. \tag{2.8}$$

Pinkall and Polthier [1993] 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. (2.4). A somewhat clearer derivation can be found in a more recent work of Polthier [2000]. There, it is shown that the discrete Dirichlet energy is

$$E_D(u) = \frac{1}{4} \sum_{i,j|\{i,j\}\in K} (\cot \alpha_{i,j} + \cot \beta_{i,j}) |\mathbf{v_i} - \mathbf{v_j}|^2,$$
$$\alpha_{i,j} = \angle(i, k_0, j), \beta_{i,j} = \angle(i, k_1, j), \{i, j, k_c\} \in K \tag{2.9}$$

and that the minimizer $\nabla E = 0$ solves

$$0 = \frac{1}{2} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{i,j} + \cot \beta_{i,j})(\mathbf{v_i} - \mathbf{v_j}) \tag{2.10}$$

at each vertex i. This leads to weights

$$\lambda_{i,j} = \begin{cases} \frac{\cot \alpha_{i,j} + \cot \beta_{i,j}}{\sum_{j \in \mathcal{N}(i)} (\cot \alpha_{i,j} + \cot \beta_{i,j})} & \{i,j\} \in K \\ 0 & \{i,j\} \notin K \end{cases} \tag{2.11}$$

which are used to obtain an embedding by solving Eq. (2.4).

Another formulation, which is probably better known in the graphics community, is given by Eck et al. [1995].

Area preserving Divide&Conquer mapping

Gregory et al. [1998, 1999] 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 diametrical 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.

Comparison and Conclusion

We have embedded parts of a mesh using the four approaches presented above. Note that the solution of matrix equation (2.4) could be performed using hierarchical techniques [Lee et al. 1998; Hormann et al. 1999], which is equivalent to using multi-grid 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 2.1. It is apparent that the general structure of larger and smaller triangles is very similar in all embeddings generated using linear optimization techniques. 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 to the high ratio of either areas or angles 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 parameterization 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 re-parameterization (see Section 2.4).

2.2 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.

2.2.1 Star shapes

Kent et al. [1991, 1992] 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 form 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 2.2

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 half-spaces induced by the face elements of the mesh. The intersection of all half-spaces 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 [Preparata & Shamos 1985].

2.2.2 Star shapes around an axis

However, genus 0 objects can be parameterized not only on the sphere. Lazarus and Verroust [1997] parameterize the polyhedra over an axis and two spherical caps at the ends of the axis. The objects which are suitable for this approach can be considered as star-shaped around a 3d curve. The curves have to be specified by the user for each input object. The curves are used as axes to parameterize the object

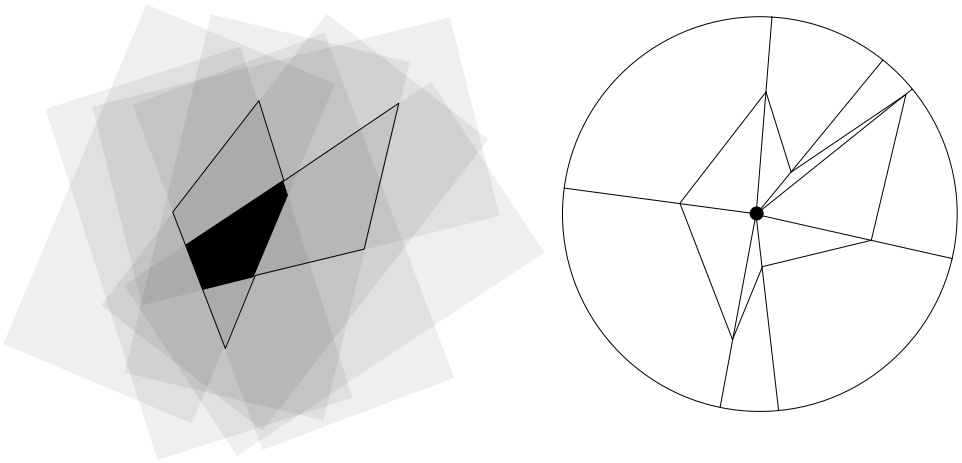


Figure 2.2: A polygonal star shape and its projection to a circle. The kernel of a star shape is the intersection of all open half spaces over the edges (faces in case of a polyhedron). Every point in the kernel induces a bijective mapping to the circle by projection.

onto three sheets. One domain results from cross-sections which are orthogonal to the axis, and two result from hemispherical regions around the endpoints of the axis.

2.2.3 Curve evolution

Carmel and Cohen-Or [1998] use curve evolution to find a parameterization of a mesh. The curve evolution process iteratively deforms the curve over time. The deformation of the curve from one time instance to the next is defined by curvature and normal field of the curve. Points of the curve are moved along normal directions with a step width depending on the local curvature. Using the right parameters, curve evolution will transform a simple curve into a convex curve so that the curve is simple in all time steps. Carmel and Cohen-Or adapt curve evolution for polygons and polyhedra since the theory for the continuous case does not hold in the discrete one. See Figure 2.3 for an illustration.

2.2.4 Simplification

Shapiro and Tal [1998] 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

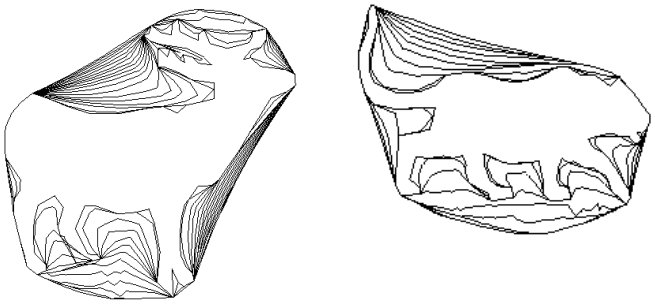


Figure 2.3: Two examples for curve evolution adapted to polygons.

strictly less than 6 has to exist.

Once the shape is simplified to a tetrahedron, vertices are re-attach 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.

2.2.5 Spring embedding

We have introduced a variation of spring embedding to embed polyhedra on the unit sphere [Alexa 1999; Alexa 2000]. In spring embedding algorithms, one tries to minimize a potential defined as

$$W = \sum_{\{i,j\}\in E} \kappa_{i,j} \left\| \mathbf{w_i} - \mathbf{w_j} \right\|^2 . \tag{2.12}$$

Algorithms based on this paradigm produce nice results in the plane (see the previous section). In the planar case, the embedding is supported by a fixed peripheral cycle of the graph. Obviously, a peripheral cycle does not make sense on a closed manifold. But, if none of the vertices is fixed, the minimum energy state is reached when all vertices coincide.

Our solution to this problem is as follows: Start with a reasonably equal distribution of the vertices on the sphere. During the relaxation towards a minimum energy state longer edges are penalized. Because the collapse of the vertices into one point has to pull at least one triangle over the equator, penalizing long edges effectively prevents the vertices from collapsing.

More precisely, in each step of the relaxation process, vertex i is moved

$$\mathbf{p_i} = c \frac{\sum_{\{i,j\}\in E} (\mathbf{w_i} - \mathbf{w_j}) \left\| \mathbf{w_i} - \mathbf{w_j} \right\|}{|\mathcal{N}(i)|} \tag{2.13}$$

Multiplying $(\mathbf{w_i} - \mathbf{w_j})$ by its length results in a quadratic weight for the edge lengths, such that longer edges are shortened. The constant c is used to control

the overall move length. With $c = 1$, the relaxation runs robustly but not very efficiently. For very short edges, the quadratic weight makes moves very short and, thus, convergence very slow. Therefore, it is advisable to scale up the move length proportionally to the inverse of the longest edge incident upon one vertex. The result is a much faster convergence.

Because $(\mathbf{w_i} + \mathbf{p_i})$ is not necessarily on the unit-sphere, we normalize the term.

Gumhold [2000] has reported another solution to the same problem, which is simpler and more elegant: The sphere is re-centered after each relaxation round, i.e.

$$\mathbf{w_i} = \frac{1}{n}\left(\sum_{j \in K} \mathbf{w_j}\right) - \mathbf{w_i} \tag{2.14}$$

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, i.e. the side that was on the outside of the model is on the outside of the sphere (obviously, the surface cannot fold back upon itself without at least one triangle being upside down).

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 $sgn\left((\mathbf{w_0} \times \mathbf{w_1}) \cdot \mathbf{w_2}\right)$. So, the relaxation is not terminated until the orientation of each face is the same as in the original model. Because this test is rather expensive, it should be done only every R iterations. We use $R = 10000$. After the embedding is valid a conventional epsilon bound is used as the final termination criterion. Actually, we use $\max_i\left(\|\mathbf{p_i}\|\right) < \epsilon$.

A relaxation process for the polyhedral model of a horse is depicted in Figure 2.4 and resulting embeddings for several models are shown in Figure 2.5.

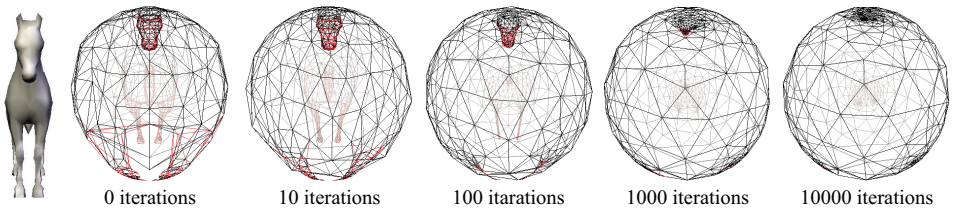


Figure 2.4: Embedding a polyhedral object on a sphere using relaxation. Initially, the vertices are projected through an interior point of the model onto a unit sphere. The relaxation is finished when all faces are oriented correctly. Incorrectly oriented faces are surrounded by red edges.

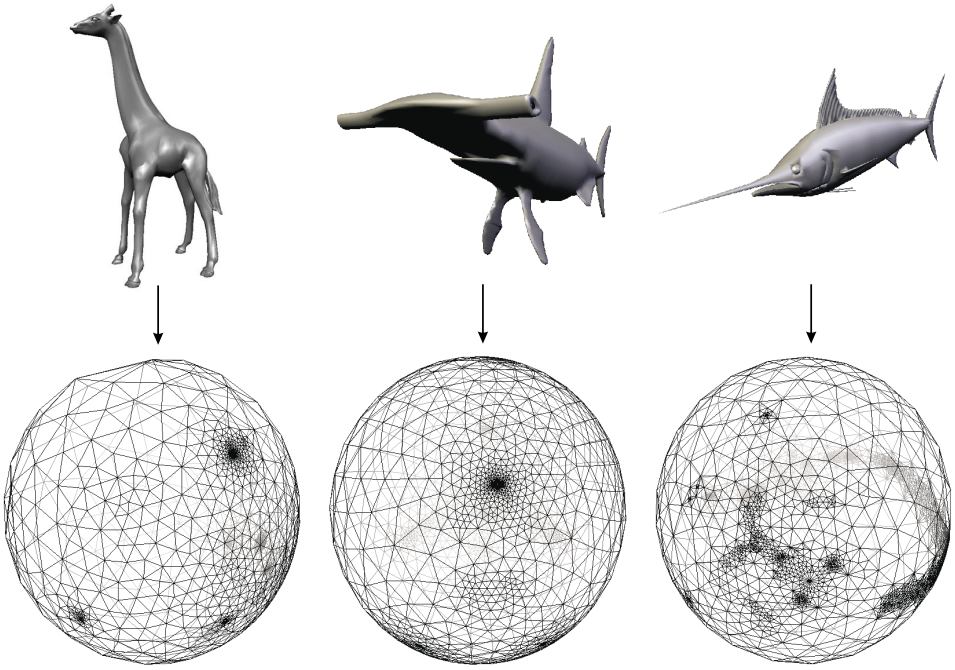


Figure 2.5: Sphere embeddings of the models of a giraffe, a hammerhead shark, and a swordfish.

2.2.6 Embedding in the plane

It is long known that the topology of the sphere can be embedded into the 2d plane. Kanai et al. [1997, 1998] use harmonic mappings to embed genus zero meshes in the plane. To find the parameterization in the plane they use the discrete harmonic mappings (see Section 2.1).

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 2.4) are necessary to make these embeddings useful. 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 topological defects in the mesh) while the relaxation generates smooth embeddings. Both techniques suffer from the area compression problem mentioned earlier.

2.3 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 a to the plane using one of the techniques discussed in Section 2.1. 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.

2.3.1 Automatic dissection of shapes

Ideally, the dissection process would not require the user to assist. However, the fully automatic dissection of two meshes into isomorphic structures seems to be a hard problem. The approach of Kanai et al. [1997, 1998] 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 known as mesh partitioning and naturally related to graph theory. Some algorithms try to balance the size of patches (e.g., Eck et al. [1995], Karypis & Kumar [1998]).

In many multi resolution methods, however, the base domain (the structure of large patches) is found by simplifying the mesh using vertex removal [Schroeder et al. 1992; Schroeder 1997; Klein 1998; Kobbelt et al. 1998] or edge collapse [Hoppe 1996; Garland & Heckbert 1997; Lindstrom & Turk 1998].

These techniques might help in deriving a single base domain for two meshes. Lee et al. [1999] use two independently established base domain to generate one base domain for both meshes. They employ their MAPS scheme [Lee et al. 1998] 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).

2.3.2 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 [1996] 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. [1998, 1999] assist the user in defining the edges (see Figure 2.6). 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.

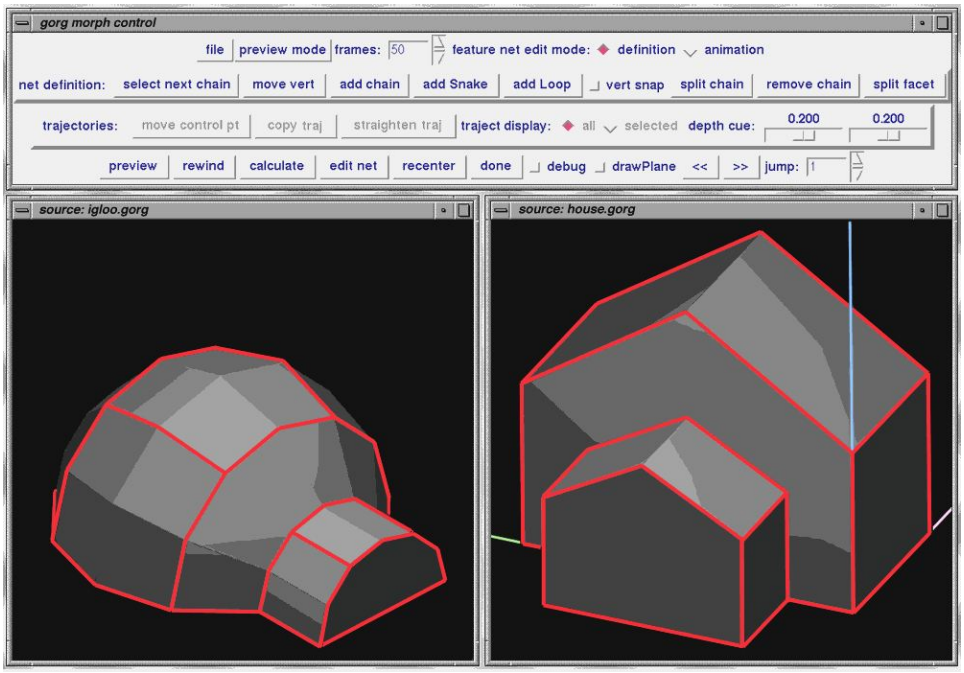


Figure 2.6: User guided decomposition of meshes. Here, the user constructs closed loops and segments to dissect two meshes into isomorphic patch networks. Reprinted from Gregory et al. [1998].

The works of Bao & Peng [1998] and Zöckler et al. [2000] 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 the work of Gregory et al. [1998, 1999]

In the approach of Kanai et al. [2000] 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 2.7 shows the resulting dissection 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 [Kanai & Suzuki 2000].

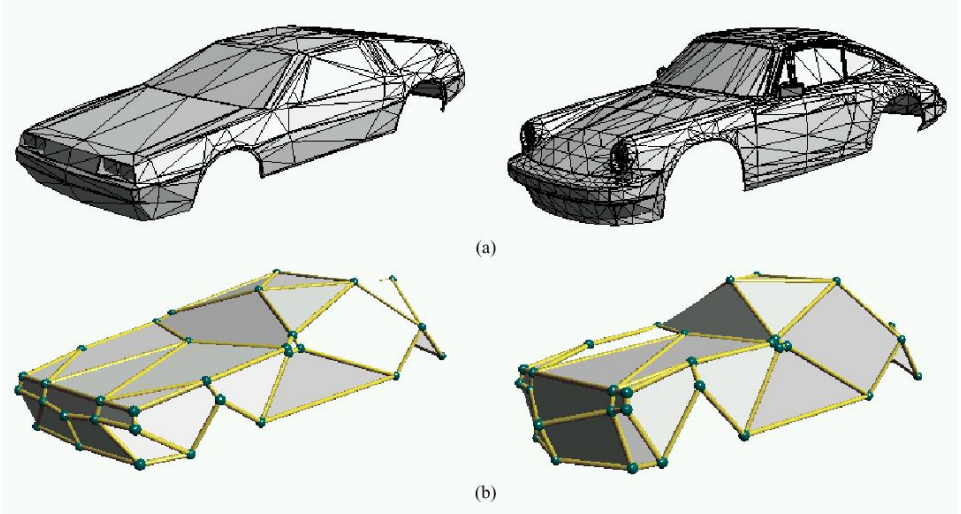


Figure 2.7: In the approach of Kanai et al. [2000] the user selects only corresponding vertices and how they are to be connected. The mesh is the dissected using shortest path connecting the vertices. Reprinted from Kanai et al. [2000].

However, even using the exact shortest path can lead to problems. Praun et al. [2001] 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 two 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.

2.4 Feature alignment

The necessity for aligning prominent features becomes evident even in very simple examples. Figure 2.8 shows two morphs between models of a young pig and a grown-up pig. In the upper sequence, no features were aligned and the resulting morph is unacceptable. The lower sequence of Figure 2.8 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 topology of the models (in this example, the different mesh topologies are obvious from the different vertex counts of the models).

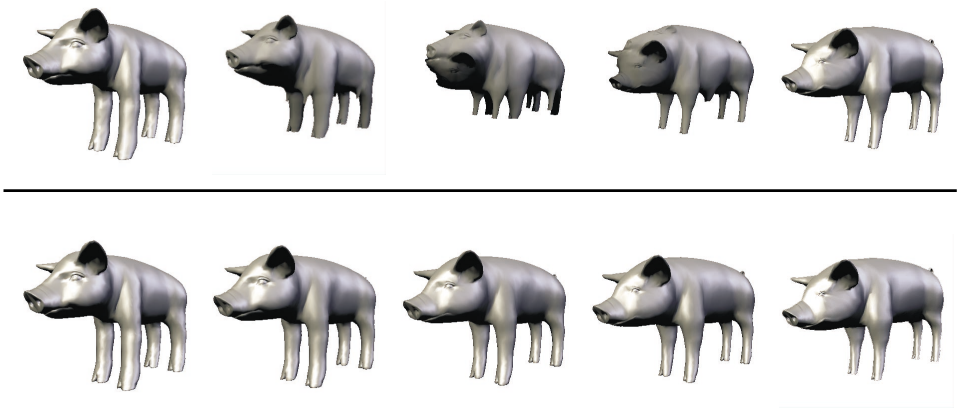


Figure 2.8: Morphs between the models of a young pig and a grown-up pig. In the upper row, no feature alignment is used, which leads to unpleasant effects (e.g., eight legs in the intermediate models). In the lower row, the eyes, ears, hoofs, and the tail are aligned (a total of 17 vertex-vertex correspondences), yielding a smooth transformation.

2.4.1 User-selected vs. shape features

A difficult task is to identify common features of 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, corre-

sponding edges and vertices) they can specify which parts of the meshes correspond. However, except for the techniques presented by Kanai et al. the user is also involved in other tasks, which make the process complicated and lengthy.

However, the shapes' geometries also contain information useful to exploit. One could extract prominent geometric features of the shapes (e.g. Hubeli & Gross [2001]) and try to match them. Several functions over the parameter domain of the function seem to be worth looking at. It is important that these functions are independent of the parameterization, 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, without a distant view. The most prominent assets for describing 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 independent 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 $n : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ or the Gaussian curvature $c : \mathbb{R}^2 \rightarrow \mathbb{R}$. More generally, these descriptions are part of the fundamental forms of the approximated smooth surface.

It is clear that this information about the shapes does not lead to point to point correspondences 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, Surahzky and Elber [2001] use the Integral over the inner products of normals:

$$D = \int_S \langle n_1, n_2 \rangle \, dS \tag{2.15}$$

Here, the inner product between normals and the integration over the surface represent particular choices. One might 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 parameterization is changed so that the functional is minimized. Note that no point has an a-priori optimum placement making this problem much harder to solve than aligning specified point to point correspondences.

2.4.2 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 features. Note that this is not possible for parameterizations resulting from dissection as the orientation of each patch is determined by neighboring patches.

In [Alexa 1999; Alexa 2000] we align a set of point to point correspondences by rotating the spherical embeddings of the mesh. The objective function to be minimized is the squared distance of corresponding points, however, we could have also used the inner products of points on the sphere. The minimization problem can be solved using multidimensional numerical minimization (three values have to be found: an axis of rotation and an angle).

2.4.3 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.

Zöckler et al. [2000] and our approaches [Alexa 1999; Alexa 2000] allow the user to select a set of point to point correspondences. Warping techniques similar to those used in image morphing (e.g., see [Ruprecht & Muller 1995; Wolberg 1998]) are used deform the parameterization so that cooresponding points coincide. Whether the parameter domain is a disk as in [Zöckler et al. 2000] or a sphere as in [Alexa 1999; Alexa 2000] 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 would be warped. But since the algorithm later requires 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 fold-over.

The solution is to make several local and small deformations instead of searching for one warping function that is applied once to all vertices. This way, we can control the effect of the deformation (does it introduce fold-over, or not). The mapping function proposed in [Alexa 1999; Alexa 2000] to move a vertex from $\mathbf{w}$ to $\hat{\mathbf{w}}$ is

$$f(\mathbf{x}) = \begin{cases} \mathbf{x} + c\,(r - \|\mathbf{x} - \mathbf{w}\|)\,(\hat{\mathbf{w}} - \mathbf{w}) & \|\mathbf{x} - \mathbf{w}\| < r \\ \mathbf{x} & \|\mathbf{x} - \mathbf{w}\| \geq r \end{cases} \tag{2.16}$$

where r is the radius of influence for the map. However, other functions could be used as well.

We have proposed to warp only as much as is possible with the given triangulation. If the mapping starts to introduce fold-over in the triangulation we first shorten the move length by adjusting the constant c and then make the map more local by adjusting the radius of influence r. However, the features are not guaranteed to coincide after this process.

The map is applied to both models and the necessary move is derived from the current positions. That is, we do not define an intermediate position for the feature vertices, as is common in image morphing. So, in case the deformation in one

model would introduce fold-overs, coincidence can still be achieved by deforming the other model.

Zöckler et al use the fold-over free warping scheme of Fujimura and Makarov [1998]. They also warp in small steps. However, if fold-over occurs they change the mesh topology 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 [Eckstein et al. 2001; Lévy 2001]. These techniques could be applied for the problem here.

Lévy [2001] formulates the problem of satisfying given point constraints by incorporating the squared error of the point correspondences into the energy functional used to generate the parameterization. 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 embedding and then increasing importance of the constraints as long as the embedding stays valid.

Eckstein et al. [2001] 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.

2.5 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 reliable 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.