

# Inter-Surface Mapping

John Schreiner  
University of Utah

Arul Asirvatham  
University of Utah

Emil Praun  
University of Utah

Hugues Hoppe  
Microsoft Research

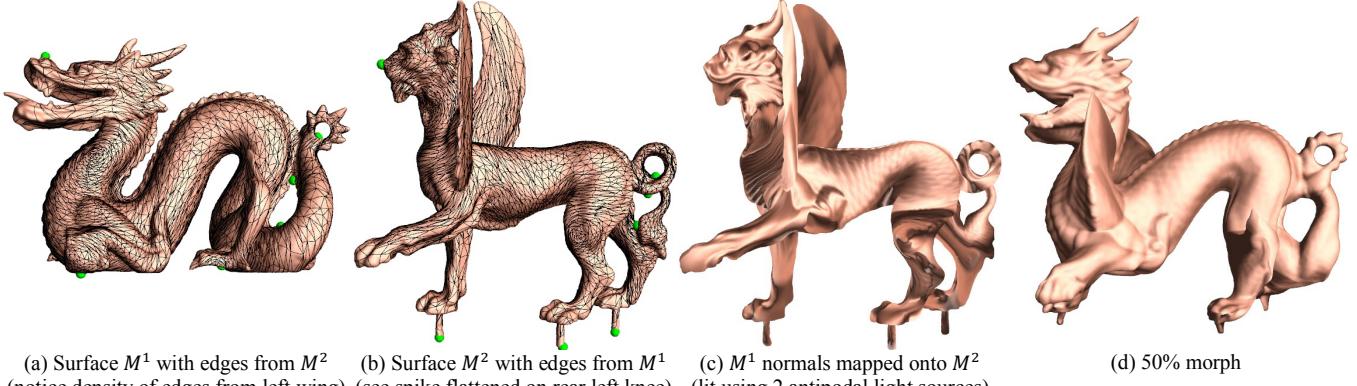


Figure 1: Inter-surface map for two objects of genus 2, initialized with 8 user-specified feature points. (Symmetric stretch efficiency 0.311).

## Abstract

We consider the problem of creating a map between two arbitrary triangle meshes. Whereas previous approaches compose parametrizations over a simpler intermediate domain, we directly create and optimize a continuous map between the meshes. Map distortion is measured with a new symmetric metric, and is minimized during interleaved coarse-to-fine refinement of both meshes. By explicitly favoring low inter-surface distortion, we obtain maps that naturally align corresponding shape elements. Typically, the user need only specify a handful of feature correspondences for initial registration, and even these constraints can be removed during optimization. Our method robustly satisfies hard constraints if desired. Inter-surface mapping is shown using geometric and attribute morphs. Our general framework can also be applied to parametrize surfaces onto simplicial domains, such as coarse meshes (for semi-regular remeshing), and octahedron and toroidal domains (for geometry image remeshing). In these settings, we obtain better parametrizations than with previous specialized techniques, thanks to our fine-grain optimization.

**Keywords:** surface parametrization, shape morphing, remeshing.

## 1. Introduction

Surface parametrization refers to mapping a triangle mesh onto a simpler domain such as the plane, the sphere, or a coarse simplicial domain. The parametrization is represented by a map  $\phi_{D \rightarrow M}$  where  $M$  is the mesh and  $D$  is the simpler domain. In computer graphics, parametrization is central to texture mapping, whereby images placed in the domain are sampled on rendered surfaces to provide texture detail, place decals, encode shadows, record radiance transfer coefficients, etc. Surface parametrizations also appear in numerous applications, including digital geometry processing, morphing, surface editing, object recognition, and geometry remeshing.

We address the more general problem of directly constructing a continuous bijective map  $\phi_{M^1 \rightarrow M^2}$  between two triangle meshes

$M^1$  and  $M^2$  of the same topology. (Continuity precludes maps between surfaces with different genus or number of boundaries.) Unlike previous approaches which compose parametrizations of  $M^1$  and  $M^2$  over some intermediate domain (as reviewed in Section 2), we directly optimize the quality of the *overall* map  $\phi_{M^1 \rightarrow M^2}$ . Our method works for arbitrary genus and does not require the user to provide a simplicial complex (e.g. [Praun et al 2001]). The user may optionally specify corresponding feature points on  $M^1$  and  $M^2$ , and our construction guarantees that the map satisfies these constraints.

Some parametrization schemes may require a large set of manually specified features to guide the parametrization process to a good (or even valid) solution. As we shall show, our mapping method is robust even with few feature constraints. Moreover, directly minimizing the distortion of the inter-surface map tends to naturally align corresponding shape elements. Of course, a few user-specified constraints are helpful for overall registration and for linking semantically related regions.

Our approach adds a new fundamental tool to the Digital Geometry Processing toolbox. Its main **contributions** are:

- Inter-surface mapping without any intermediate domain, to directly measure the distortion of the overall map.
- Symmetric distortion metric, i.e. invariant to the interchange of  $M^1$  and  $M^2$ .
- Symmetric coarse-to-fine optimization algorithm to provide robustness and convergence to a good solution.
- Initialization of map to robustly satisfy any user-specified feature correspondences.

**Additional scenarios.** While our motivating application is the creation of maps between surfaces of comparable complexity, our framework can also be used in cases where  $M^1$  is a simpler mesh, possibly inferred from  $M^2$ :

- Simplicial parametrization (for semi-regular remeshing): given a surface  $M^2$  and desired domain vertices on  $M^2$ , we automatically create domain  $M^1$  and a parametrization.
- Octahedral parametrization (for geometry-image remeshing):  $M^1$  is a regular octahedron, and feature points are unnecessary.
- Toroidal parametrization (for remeshing of genus-1 shapes).

Our more general optimization framework actually obtains better results than the previous techniques specialized to these scenarios.

## 2. Related work

**Planar parametrization.** The traditional surface parametrization problem considers the case where the domain  $D$  is a planar region  $P \subset \mathbb{R}^2$  (see survey in [Floater and Hormann 2003]). The map  $\phi_{D \rightarrow M}$  is represented by the parametric locations of vertices of  $M$  within the plane. Optimization can freely move the vertices within the domain as long as bijectivity is maintained.

Kraevoy et al [2003] present the *Matchmaker* scheme for satisfying corresponding feature point constraints in  $D$  and  $M$ . We extend their scheme to form a corresponding graph of paths on two surfaces  $M^1, M^2$  of arbitrary genus  $g$ , possibly with boundaries. To guarantee the successful termination of the path insertion process, we impose ordering constraints on the neighbors of a feature vertex, and we trace a spanning tree and  $2g$  non-separating cycles before completing the full graph. Consistent neighbor ordering is necessary to avoid partial graphs that are impossible to complete, as shown on the right (if  $D$  and  $E$  link to the same base vertex  $B$  or  $C$ , this will result in flipped triangles; if they link to different ones, edges will cross.)

An important limitation of planar parametrization is that representing an entire surface requires that it be cut into one or more disk-like charts, where each chart is parametrized independently. Some techniques cut the surface into a single chart [e.g. Gu et al 2002; Sheffer et al 2002; Sorkine et al 2002], while others cut it into an atlas of charts [e.g. Maillot et al 1993; Sander et al 2001; Levy et al 2002; Gu and Yau 2003]. In either case, the cuts break the continuity of the parametrization, making it difficult to use a planar parametrization approach to construct a continuous map between two *different* surfaces, since their cut structures differ.

**Spherical parametrization.** By letting the surface domain  $D$  be the unit sphere  $S$ , one can directly parametrize a closed genus-zero surface without any cuts. Examples of spherical parametrization methods include [Haker et al 2000; Alexa 2002; Gotsman et al 2003; Praun and Hoppe 2003].

**Simplicial parametrization.** Another approach lets the domain  $D$  be a coarse base mesh. The surface is partitioned into triangular regions that are mapped respectively to faces of  $D$  [e.g. Eck et al 1995; Lee et al 1998; Guskov et al 2000; Praun et al 2001].

The challenge in simplicial parametrization is that it is difficult to globally optimize the parametrization. Whereas planar and spherical domains are smooth everywhere, simplicial domains have sharp edges and vertices. Since the whole domain cannot be simultaneously “unfolded”, most methods iteratively apply a linear relaxation to a small group of adjacent faces. For example, Eck et al [1995] iteratively unfold a pair of adjacent domain faces and reparametrize the surface neighborhood over the resulting quadrilateral. Guskov et al [2000] perform local reparametrizations over 1-ring vertex neighborhoods, with the advantage that the images of domain vertices can shift over the surface.

Rather than iteratively optimizing local neighborhoods, Khodakovsky et al [2003] set up a global system where the mesh edges spanning adjacent domain faces are treated as if the two faces were locally unfolded into a plane. Solving the global system provides much faster convergence. Unfortunately, the domain vertices are fixed during the global system, and must be relaxed separately using traditional 1-ring relaxation.

The inter-surface mapping problem could be viewed as an instance of simplicial parametrization where the domain  $D = M^1$  is an unusually complicated simplicial domain. However, existing simplicial parametrization techniques are not applicable, because:

- (1) They require an initial correspondence from all vertices of  $M^1$  to surface  $M^2$ , obtained by the *construction* of  $M^1$  from  $M^2$ .

(2) Even with this initial correspondence, the techniques would converge too slowly due to the high complexity of  $M^1$ .

(3) Simplicial parametrization techniques ignore the geometry of surface  $M^1$ , since they assume it to be an abstract domain.

**Inter-surface mapping.** Lee et al [1999] create an inter-surface map between two surfaces  $M^1, M^2$  by first constructing simplicial parametrizations  $\phi_{D^1 \rightarrow M^1}, \phi_{D^2 \rightarrow M^2}$ . Because the domain meshes  $D^1, D^2$  are different, user assistance is required to form a good map between them, and this map construction is not robust.

To overcome this drawback, Praun et al [2001] develop a simplicial parametrization method in which the connectivity of the simplicial complex  $D$  can be specified *a priori*. Given a genus-0 simplicial complex and desired images of each domain vertex on multiple surfaces, they construct *consistent* parametrizations  $\phi_{D \rightarrow M^1}, \phi_{D \rightarrow M^2}$  over the shared simplicial domain  $D$ .

Both spherical parametrization and consistent simplicial parametrization can be used to create a continuous map between two surfaces  $M^1$  and  $M^2$  by forming the composition  $\phi_{D \rightarrow M^2} \circ \phi_{D \rightarrow M^1}^{-1}$  (where  $D$  is the sphere or simplicial domain, respectively). However, using an intermediary domain may result in a poor inter-surface map, since each sub-map *ignores* the non-uniform distortion present in the other. For example, when creating a map between a cow and a horse, the cow legs would not be encouraged to match up with the horse legs. While it is possible to manually force correspondences of constraints on a dense set of domain vertices, a more elegant and flexible solution is to automatically *favor* this correspondence within the distortion metric itself.

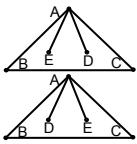
Kraevoy and Sheffer [2004] use the composition  $\phi_{D \rightarrow M^2} \circ \phi_{D \rightarrow M^1}^{-1}$  to remesh  $M^2$  using the connectivity of  $M^1$  (together with additional vertices). They smooth the map using a spring relaxation with edge weights related to local remesh error. Their approach is much faster than ours, and produces simpler meshes than our method run to full resolution. However, our coarse-to-fine computation can be stopped at any lower resolution, or the final map could be post-simplified considering both mesh geometries. Since their map uses a conformal metric (as in Figures 5, 9, 10), the user must associate all interesting model features. Their scheme is robust for genus-zero models; although they demonstrate maps between models of nonzero genus, their algorithm may abort and ask the user for additional feature constraints.

In this paper, we directly optimize the distortion of the inter-surface map, and robustly support surfaces of arbitrary genus.

## 3. Approach overview

**Map representation.** Our goal is to produce a piecewise-linear map between two triangulated surfaces. Unlike in planar parametrization, the linear pieces of the map are finer than the original mesh faces, as they correspond to triangles of a *mutual tessellation* [Turk 1992] (a.k.a. *meta-mesh* [Lee et al 1999]) of the two surfaces. Vertices of this meta-mesh include the vertices of both initial meshes as well as vertices formed by edges of  $M^1$  intersecting those of  $M^2$ . To fully specify the map, for each mesh vertex we record the face of the other mesh to which it maps, along with barycentric coordinates within that face, and for each edge-edge intersection, we record the two ratios formed by the split point on each edge. Together, the vertex and edge-edge barycentric coordinates define a set of polygonal sub-regions on faces of  $M^1$  and  $M^2$ . We define a unique piecewise-linear map by further triangulating these polygonal regions.

On rare occasions, it is necessary to “bend” the image of an edge of  $M^1$  inside a triangle of  $M^2$  and vice-versa. We achieve this by introducing special *kink vertices* of valence 2 in mesh  $M^i$  (Section 5). These vertices have a corresponding face and barycentric coordinates in the other mesh just like regular vertices.



**Algorithm overview.** Our strategy is to use progressive refinement to robustly create and optimize the inter-surface map. Even for planar and spherical parametrizations, which involve smooth domains, coarse-to-fine approaches help parametrizations converge to good solutions [Hormann et al 1999b; Sander et al 2002; Aksoylu et al 2003]. For inter-surface maps, the lack of domain smoothness exacerbates the problem of local minima, further motivating progressive refinement.

Our method first constructs progressive mesh (PM) representations of both  $M^1$  and  $M^2$  [Hoppe 1996]. To simplify the task of initializing the inter-surface map (and in fact make this task trivial), we constrain the two progressive meshes to have base meshes with identical connectivities. And, to satisfy user-specified correspondences, feature points are retained as vertices in the base meshes. Consequently the algorithm becomes provably robust. A trivial valid map is created initially, and the refinement operations always succeed, so that by induction we are guaranteed a valid map between the fully refined surfaces.

The **basic steps** of our algorithm are:

- (1) Partition the surfaces  $M^1$  and  $M^2$  into a corresponding set of triangular patches, by tracing a set of corresponding paths. If user-specified features are provided, these are chosen as path endpoints. (Section 4)
- (2) Create progressive mesh representations of both  $M^1$  and  $M^2$ , using the path networks to constrain the simplifications, resulting in two base meshes with identical connectivities.
- (3) Establish a trivial map between the two base meshes: a 1-to-1 map on vertices, with no edge-to-edge intersections.
- (4) Iteratively refine the two progressive meshes. After each vertex split, update the inter-surface map and optimize it on the local neighborhood. When both meshes are fully refined, we obtain the inter-surface map. (Section 5)

Steps 1 and 4 are the most challenging, and are presented in more detail in the next two sections. To create the progressive meshes in Step 2, we constrain the edge collapse sequence to preserve the topology of the paths, as described by Sander et al [2001]. We thus obtain base domains whose edges correspond to original paths and whose triangles correspond to original patches (see Figure 2). Since the two base domains have the same connectivity, the construction of the initial map between them in Step 3 is trivial.

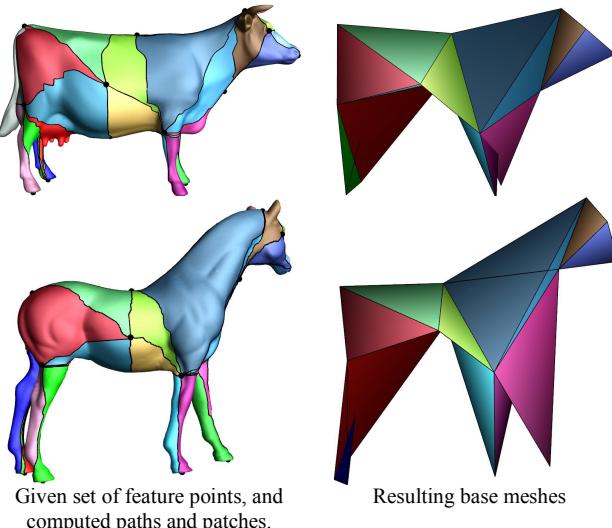


Figure 2: Example of consistent partitioning process.

#### 4. Initialization of coarse map

Our goal is to form a consistent partitioning of meshes  $M^1$  and  $M^2$  into corresponding triangular patches. The patch boundaries are defined by path networks linking together feature vertices. These feature vertices are optionally specified by the user. If their number is insufficient for the given surface genus (e.g. at least 4 features for genus 0, more for higher genus), our algorithm automatically inserts additional pairs. We allow the path connectivity to be either specified (as in [Praun et al 2001]) or arbitrary (as in [Kraevoy et al 2003]), depending on the application scenario.

Our approach is to link together corresponding feature pairs on both meshes using constrained shortest paths, similarly in spirit to the methods of Praun et al [2001] and Kraevoy et al [2003]. We add paths in a greedy fashion, subject to constraints that ensure consistent topology, and using heuristics that avoid swirls. When a maximal graph of non-crossing paths has been created, the two surfaces have been partitioned into triangular patches.

**Path tracing.** We trace the shortest path between a pair of feature vertices using a Dijkstra search. The search is constrained to not intersect with paths already in the network. To obtain path networks with consistent topologies between the two meshes, we must maintain a consistent ordering of the neighbors of each vertex. Therefore an additional constraint on the paths is to start and end in corresponding sectors on the two meshes. (We assume that the meshes are orientable.) When the shortest paths on each mesh are not consistent, we trace two candidate pairs of paths, by imposing the sectors from  $M^1$  on the path on  $M^2$ , and vice-versa, and then pick the best pair.

To allow the creation of a valid path between any pair of features, we lazily add extra Steiner vertices in the meshes, as suggested by Kraevoy et al [2003]. Our scheme performs Dijkstra searches on both the mesh vertices and the edge midpoints. Since using edge midpoints in a path corresponds to adding Steiner vertices, we give preference to paths that do not use them. This may lead to slightly more jagged paths, but the precise geometry of the paths is not critical to the final map, since the paths are not constraints — they only guide the construction of compatible PM sequences.

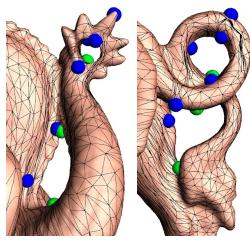
Our greedy path-insertion algorithm selects the best pair of corresponding paths from a priority queue sorted by the sum of path lengths on  $M^1$  and  $M^2$ . The queue is initially populated by tracing paths from each vertex to its 10 closest neighbors. When the best candidate is selected, we check whether it is still valid, and if not we recompute it and insert it back in the queue.

To guarantee the success of the algorithm, we must avoid enclosing any vertex within a path cycle not connected to it. Praun et al [2001] observe that for genus-0 surfaces it is sufficient to first build a spanning tree of the feature vertices (before forming any cycles). We generalize this approach to arbitrary genus. To this end, we must distinguish between separating and non-separating cycles formed by the paths. (A separating cycle is one that breaks the surface into two disjoint components.) Our strategy is to first build a maximal path network without separating cycles, before adding any paths forming separating cycles.

For a surface of genus  $g$  with  $k$  feature vertices, the maximal non-separating graph is the union of a tree spanning all feature points and  $2g$  non-separating cycles, and thus has exactly  $k - 1 + 2g$  paths. This maximal non-separating graph topologically cuts the surface into a disc [Gu et al 2002], with all the sectors around feature vertices as vertices on the boundary of the disc. The neighbor ordering constraint ensures that the ordering of the disc vertices is the same for both  $M^1$  and  $M^2$ . In such a configuration, there always exists a unique way to link any two vertices (sectors adjacent to a feature). Once such a path is added, each of the two

topological discs representing  $M^1$  and  $M^2$  is further split into two discs, which can be then consistently decomposed.

The new path that needs to be added to split the discs may sometimes link two features that are already connected (by a path in different sectors, going across a handle of the objects). In such cases we automatically introduce additional feature points to support the new path. As an example, for the pair of genus-2 surfaces in Figure 1, 8 features are specified by the user, and 7 additional ones are automatically introduced (blue dots in the inset close-ups).



There are two issues related to building the maximal non-separating graph: avoiding separating cycles, and avoiding swirls.

**Avoiding separating cycles.** If a newly introduced path between vertices  $A$  and  $B$  forms a cycle, we test whether it is separating, and if so, we replace the path with one forming a non-separating cycle using an algorithm similar to that of Lazarus et al [2001]. Specifically, we perform two simultaneous breadth-first searches starting from the vertices incident to the path  $AB$ , on its two sides. The searches are constrained by the existing path network and by the candidate path  $AB$ . Each visited vertex is tagged with its parent (the vertex visited previously to get to it) and with the left/right side of  $AB$  it connects to. If we ever reach a “left” vertex from a “right” tagged one, then the cycle is non-separating. Otherwise, we form a new non-separating cycle as follows. The boundary of the region visited in the search at a certain time is in general composed of several contours that can subsequently split, merge, or contract to a point. When contours merge (say at a point  $O$ ), we trace back two paths to the previous split event  $P$ , using the “parent” fields. From this non-separating cycle between  $P$  and  $O$  we select the vertex  $X$  closest to  $A$  and  $B$ . We measure distance by tracing paths  $XA, XB$  that (1) do not cross the cycle at points other than  $X$ , (2) meet the cycle from opposite sides, and (3) end at  $A$  and  $B$  on the same side of the temporary  $AB$  path. The path  $AX-XB$  forms the final path.

If there are not enough user-provided features to resolve the genus of the object, we trace non-separating cycles connecting to one of the existing features using a procedure similar to the one above (with  $A = B$  and no “left”/“right” tags), and create two new feature constraints to support the cycle.

**Avoiding swirls.** A *swirl* is an awkward geometric configuration in which paths between feature vertices take unnecessarily long routes around other existing paths. More precisely, the presence of corresponding feature constraints establishes homotopy classes on the set of inter-surface maps. Two maps belong to the same class if there exists a continuous deformation between them that maintains the constraints. Since swirls correspond to “poor” homotopy classes, they cannot be fixed using local continuous relaxation [Praun et al 2001]. We have found two heuristics to be effective at avoiding swirls.

The first heuristic is to prefer early connection of feature points at mesh extremities. To identify mesh extremities, we compute for each feature vertex an average distance to the closest set of neighboring features (8 in our implementation). Vertices with a high distance (top 25%) are considered extrema.

The second heuristic is to delay paths that pass on the “wrong side” of neighboring features [Praun et al 2001], and when forced to choose such a path, to re-route it on the correct side. For each candidate path, we gather a set of neighboring feature vertices (the  $k$ -closest neighbors of the two endpoints on the two meshes). For each of these neighbors we determine on which side of the path it

lies by computing the side on which the shortest route from the neighbor to the path meets the path. If the side is different between the two meshes, then the path is likely to cause a swirl, so it is penalized in the pool of candidate paths. If only penalized paths are left, we attempt to re-route the lowest-cost path on the correct side of the offending neighbor vertex as follows. Shortest paths between the offending neighbor and the candidate path endpoints are computed (under normal constraints) and temporarily added to the path network. The new path is thus forced to go on the correct side of the connected component of the offending neighbor.

**Handling surface boundaries.** Our map initialization algorithm is easily extended to meshes with boundaries (Figure 7). Each boundary contour is triangulated using a single central point. The point is treated as a feature vertex, and must be associated with a corresponding boundary-center vertex on the other mesh. Once the two path networks are computed, these boundary-center vertices are removed along with the faces used to triangulate the boundaries. The paths connecting to the boundary centers are clipped to the boundary, and these clip points become new feature vertices. We then consistently triangulate the resulting non-triangular patches, and the remaining steps proceed as before.

## 5. Coarse-to-fine map optimization

Like previous work [e.g. Guskov et al 2000; Sander et al 2001], we optimize the map by moving one vertex at a time within its one-ring neighborhood to decrease the distortion metric. This optimization is performed after each vertex split for the new vertex and each of its neighbors, and for all mesh vertices when their total number has increased by a factor of 1.5.

Unlike previous methods, we consider the optimization of vertex neighborhoods not just of  $M^1$  but also of  $M^2$ . This convenient symmetry is necessary since neither mesh is a special “domain”. It also provides finer-grain optimization than previous simplicial parametrization methods.

In our current implementation, we refine only  $M^2$  for a number of steps, while  $M^1$  is held at constant resolution, then swap their roles and optimize  $M^1$ , and then repeat the process. Keeping track of only one refining mesh at a time while the other is static results in lighter-weight data structures and more manageable code. For the scenarios where one of the meshes is very simple (octahedral and simplicial parametrizations), the swaps are unnecessary.

### 5.1 Vertex optimization

The main operation considers a vertex  $v$  of  $M^2$  and optimizes its location  $\tilde{v}$  on  $M^1$ . Let  $\mathcal{N}(v)$  be the 1-ring neighborhood of  $v$  in  $M^2$ , and  $\mathcal{N}(\tilde{v})$  be the pre-image of this neighborhood in  $M^1$  under the map (Figure 3a-b). The optimization only modifies the map inside these corresponding neighborhoods, i.e. by regenerating barycentric coordinates for all meta-mesh vertices within the interior. Therefore we can exactly compute the change in overall distortion.

To perform the relaxation, we make use of a temporary 2D parametrization of the neighborhood  $\mathcal{N}(\tilde{v})$  onto a planar polygon  $\mathcal{N}(\hat{v})$ , constructed as follows (see Figure 3c). We use a one-ring unfolding where  $v$  is initially mapped to the origin  $\hat{v} = (0,0)$ , each neighbor  $w$  of  $v$  is mapped to a point  $\hat{w}$  at a radius equal to the path length  $\tilde{v}\tilde{w}$ , and the angle  $\angle \hat{u}\hat{v}\hat{w}$  between successive neighbors  $\hat{u}, \hat{w}$  is proportional to  $\angle \tilde{u}\tilde{v}\tilde{w}$  (scaled such that their sum equals  $2\pi$ ). The angle  $\angle \hat{u}\hat{v}\hat{w}$  on  $M^1$  is computed using the law of cosines applied to the path lengths  $\tilde{u}\tilde{v}$ ,  $\tilde{v}\tilde{w}$ , and  $\tilde{w}\tilde{u}$  (or to the respective Euclidean distances if the path lengths do not obey the triangle inequality). Importantly, when  $\mathcal{N}(\tilde{v})$  is entirely contained inside a single face of  $M^1$ , the map from  $\mathcal{N}(\tilde{v})$  to  $\mathcal{N}(\hat{v})$  is an isometry.

Once we have computed the boundary of the polygon  $\mathcal{N}(\tilde{v})$ , we delete all the edges incident to  $\tilde{v}$  and the corresponding paths incident to  $\tilde{v}$  (along with all their intersection points on edges of  $M^1$ ). We use the mean-value parametrization scheme of Floater [2003] to relax the 2D locations inside  $\mathcal{N}(\tilde{v})$  of the vertices of  $M^1$  contained within  $\mathcal{N}(\tilde{v})$ . Since the boundary of  $\mathcal{N}(\tilde{v})$  can be concave, flips can occur, or some interior pieces can be non-convex. In those rare cases, we re-map the boundary to a convex circle-inscribed polygon [Guskov et al 2000], and repeat the relaxation, this time guaranteeing no folds.

Next, we optimize the location of  $\tilde{v}$  using repeated line searches as in [Sander et al 2001]. In these searches, we keep  $\tilde{v}$  within the kernel of polygon  $\mathcal{N}(\tilde{v})$  to preserve map bijectivity. For each location of  $\tilde{v}$ , we redraw its incident edges, and map these back to  $M^1$  using the map  $\mathcal{N}(\tilde{v}) \rightarrow \mathcal{N}(\tilde{v})$  (Figure 3d). We apply Constrained Delaunay Triangulation (CDT) to any non-triangular (boundary-adjacent) pieces of  $\mathcal{N}(\tilde{v})$ . For all resulting triangle pieces, we compose the linear maps  $\mathcal{N}(\tilde{v}) \rightarrow \mathcal{N}(\tilde{v}) \rightarrow \mathcal{N}(v)$  to compute the Jacobian used in measuring overall map distortion.

We retain the location of  $\tilde{v}$  that achieves lowest distortion. Note that due to the initial relaxation and deletion of edge-edge crossings when constructing the neighborhood  $\mathcal{N}(\tilde{v})$ , the final distortion may be larger than that before neighborhood optimization. In this case, we discard the whole operation.

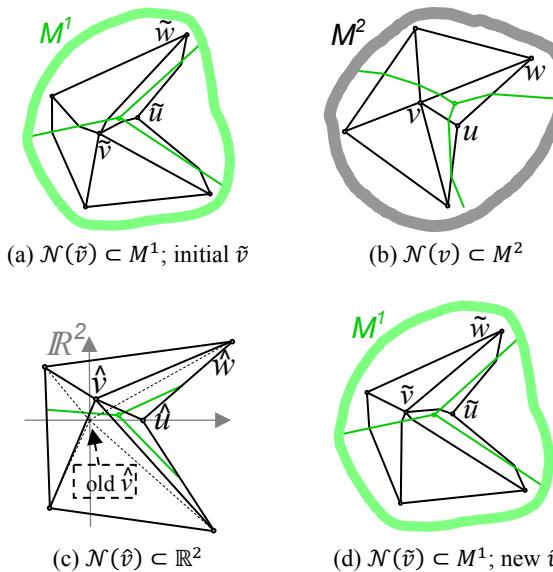


Figure 3: Illustration of neighborhoods in vertex optimization.

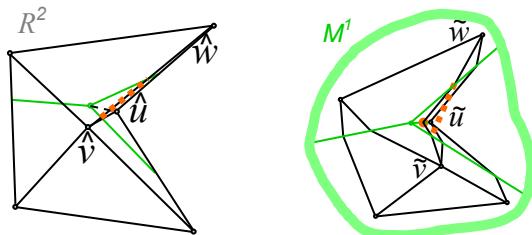


Figure 4. A kink vertex (red, right) is required since a direct segment along  $\tilde{v}\tilde{w}$  (dotted) goes on the wrong side of  $\tilde{u}$ .

**Kink vertices.** Just as Steiner vertices are sometimes necessary to create a valid bijection, in rare cases we need to “kink” the image of an edge of  $M^2$  on  $M^1$ , by breaking it at points other than intersections with edges of  $M^1$ . After optimizing the 2D location  $\tilde{v}$  of a vertex, we must map its incident edges back to  $M^1$ . We map an edge  $\tilde{v}\tilde{w}$  to a path  $\tilde{v}\tilde{w}$  by finding its intersections with pieces of  $\mathcal{N}(\tilde{v})$  in 2D, and mapping these intersection points to  $M^1$  using the split ratios on their supporting segments. Since the pieces of  $\mathcal{N}(\tilde{v})$  on  $M^1$  may have concave vertices, the straight-line segment between the two mapped intersection points may not be contained inside the piece (Figure 4). In these rare cases, we use the CDT diagonals of the concave piece to support additional break points in the path  $\tilde{v}\tilde{w}$ . These kinks are represented as temporary vertices of  $M^2$  with valence 2, and are removed when next optimizing  $\tilde{v}$  or  $\tilde{w}$ . (When swapping  $M^2$  and  $M^1$ , one of these optimizations is forced, in order to remove the kinks).

## 5.2 Distortion metric

Many parametrization distortion measures have been proposed, including angle-preservation (conformal map) [Eck et al 1995; Hormann et al 1999a; Levy et al 2002; Desbrun et al 2002], area-preservation (authalic map) [Desbrun et al 2002], and stretch minimization [Sander et al 2001]. Often, these metrics can be expressed in terms of the singular values  $\Gamma, \gamma$  of the map Jacobian  $J$  (i.e.  $\Gamma^2$  and  $\gamma^2$  are eigenvalues of the metric tensor  $J^T J$ ).

Most previous distortion metrics are asymmetric, in the sense that optimizing  $\phi$  and optimizing  $\phi^{-1}$  would not result in the same map. Two exceptions are the  $(\frac{\Gamma}{\gamma} + \frac{\gamma}{\Gamma})$  metric of Hormann et al [1999a] and the  $\max(\frac{1}{\gamma}, \Gamma)$  metric of Sorkine et al [2002], which have the key property that they are invariant to the substitution  $(\Gamma, \gamma) \leftrightarrow (\frac{1}{\gamma}, \frac{1}{\Gamma})$ .

It is likely feasible to create symmetrized versions of many prior metrics, including the popular discrete conformal map. We have chosen to symmetrize the  $L^2$  stretch of Sander et al [2001] because it smoothly penalizes scale distortion. We do this by summing direct and inverse  $L^2$  stretch:

$$L^2(T)^2 = L_{\text{stretch}}^2(M^1 \rightarrow M^2)^2 + L_{\text{stretch}}^2(M^2 \rightarrow M^1)^2 \\ = A_{\tilde{T}} \frac{A_{M^2}}{(A_{M^1})^2} \left( \frac{1}{\gamma^2} + \frac{1}{\Gamma^2} \right) + A_T \frac{A_{M^1}}{(A_{M^2})^2} (\gamma^2 + \Gamma^2)$$

where  $A$  denotes area,  $\tilde{T}$  is a triangular piece of  $\mathcal{N}(\tilde{v})$ ,  $T$  is a triangular piece of  $\mathcal{N}(v)$ , and  $\Gamma$  and  $\gamma$  are the singular values of the Jacobian  $J$  of the composed map between  $\tilde{T}$  and  $T$ . Our particular definition has the key property of being invariant to the scale of either model. Note that the symmetric formulation obviates the need for a regularizing term as was used in [Praun and Hoppe 2003].

For our results, we report the symmetric stretch efficiency, which is defined simply as  $2/\sum_T L^2(T)^2$  and has an upper bound of 1.

We also experimented with a symmetric conformal metric. However, such a distortion metric is less sensitive to changes in geometry, and therefore does not lead to natural correspondence of major geometric features. As an example, in Figure 5 the whole head of each animal is mapped to a small disc on the neck of the other.

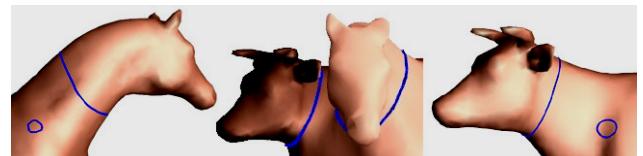


Figure 5. Use of a conformal metric results in a poor inter-surface map.

## 6. Applications and results

### 6.1 Inter-surface mapping

Figures 8, 6, and 1 show mappings between pairs of surfaces of genus 0, 1, and 2 respectively. The horse-cow map in Figure 8 uses 4 feature correspondences, the teapot-cup map in Figure 6 uses 22 feature points (red dots), and the dragon-feline map in Figure 1 uses 8 user-specified points and 7 automatically added.

Generally, the constraint points are used to initialize the map, and are then dropped during the coarse-to-fine optimization, to improve map smoothness. To see what can happen when the constraints are held fixed, let us consider the teapot-cup map of Figure 6. Because the interior of the cup has much more surface area than the teapot lid, it tries to flow around the feature constraints located on the teapot rim, causing distortion (Figure 6a). In contrast, it unfolds nicely when the constraints are relaxed (Figure 6c). If one did desire the cup and teapot rims to remain in correspondence, it would be best achieved by introducing constraint paths (instead of constraint points). For objects that are geometrically similar, such as the heads in Figure 7, point features introduce little distortion.

Figure 8 shows that with only four feature points placed on the hooves of the cow and horse models, we obtain a map where all the important features correspond to each other, as demonstrated by the morph. (If features did not match, they would appear doubled.) Not only did our optimization automatically match the two heads without any user-provided features in their vicinity, but it also matched smaller features such as the horse’s ears to the cow’s horns (as shown in the close-up). Maps obtained by composing two separate parametrizations to simple domains (planar, spherical, or simplicial) cannot easily match features in the absence of user constraints, since this information is only available in the combined map. Figure 9a shows that even with 17 feature points (two on the eyes) the composed map does not achieve the quality of our inter-surface map. Notice the presence of doubled features, such as nostrils, both pairs of ears and the cow’s horns.

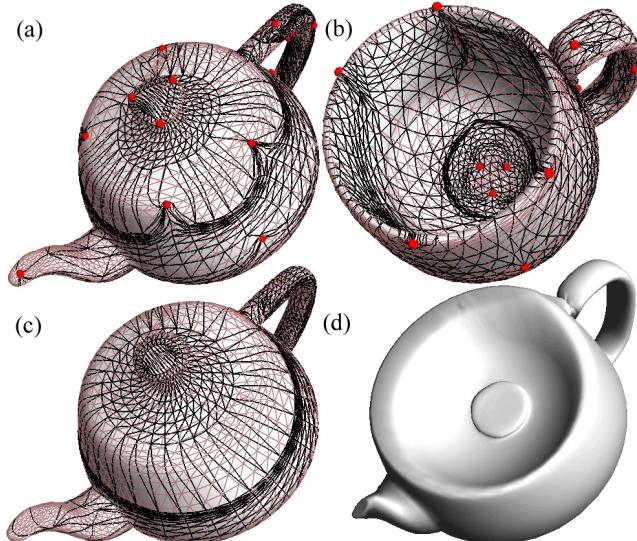


Figure 6. Inter-surface map between two genus-1 objects. (a,b) use fixed constraints while (c,d) drop the constraints after initialization. (a,c) cup edges on teapot. (b) teapot edges on cup. (d) 50% morph. (Sym. stretch efficiencies: (a,b) 0.471, (c,d) 0.598).

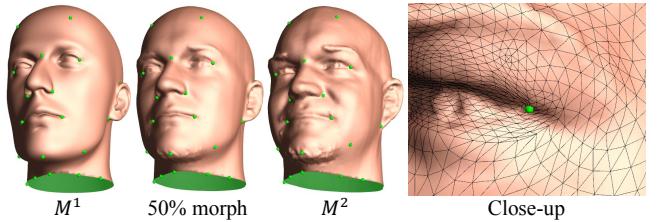


Figure 7. Map between two meshes with boundaries. The close-up on the eye shows low distortion around the feature point ( $M^1$  edges over  $M^2$  geometry). (Symmetric stretch efficiency 0.967).

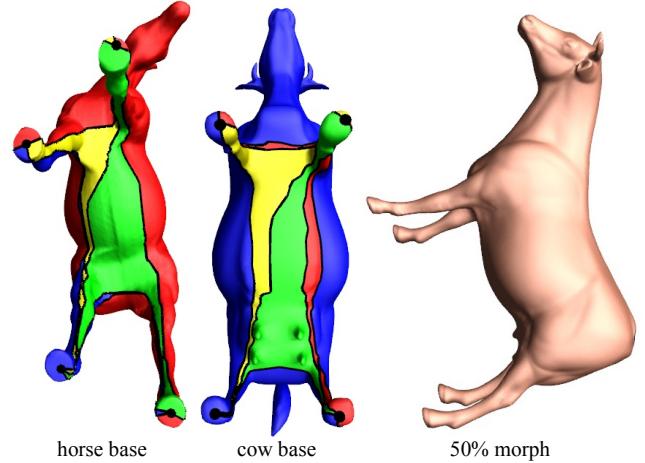
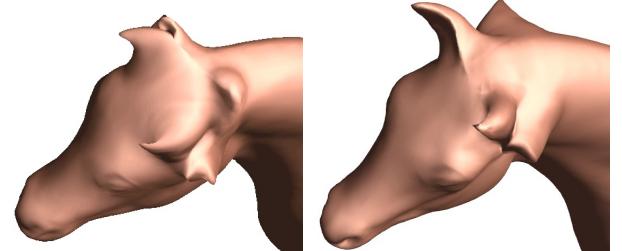


Figure 8: Cow-horse inter-surface map using only 4 features.



(a) composition of 2 simplicial maps      (b) direct inter-surface map

Figure 9. The inter-surface map automatically favors shape correspondence, unlike the composition of two separate simplicial parametrizations, as shown in these morphs. (The simplicial map uses the 17 feature points shown in Figure 2.) (Symmetric stretch efficiencies: (a) 0.416, (b) 0.442).

### 6.2 Simplicial parametrization

In this scenario,  $M^1$  is an abstract domain whose triangle faces are conceptually all equilateral. Although such a domain lacks an isometric embedding in  $\mathbb{R}^3$ , this is not a problem for the algorithm. During the construction of the local neighborhood  $\mathcal{N}(\tilde{v})$  in Section 5.1, the faces in  $\mathcal{N}(\tilde{v})$  are simply taken to be equilateral.

Among previous simplicial parametrization methods, the most advanced is the Globally Smooth Parametrization (GSP) work of Khodakovsky et al [2003], which attains smoothness across domain edges. However, it compresses the parametrization in the vicinity of low-valence irregular vertices, and stretches it near high-valence irregular vertices. As Figure 10 shows, our maps are visually smooth everywhere, and the extraordinary domain vertices have much less influence on the parametrization uniformity.

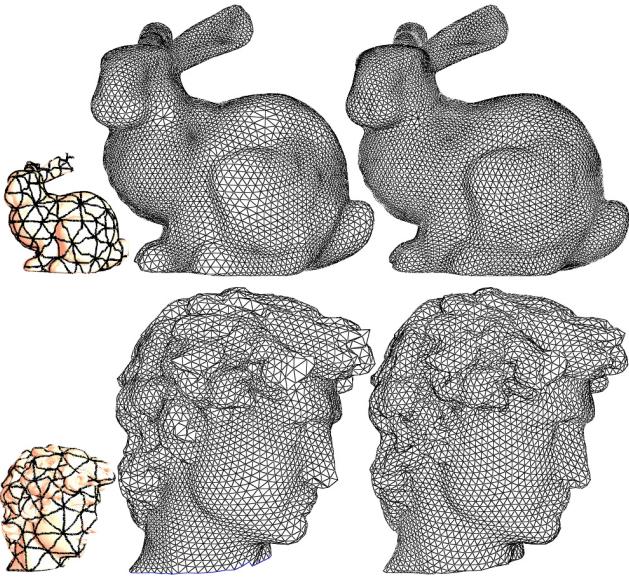
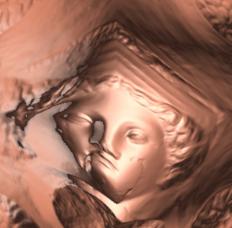


Figure 10. Comparison of semi-regular remeshing using GSP (middle) and our method (right), using the same set of base domain patches (left). (One-way stretch efficiencies: bunny 0.800, 0.915; David 0.761, 0.902).

### 6.3 Octahedral parametrization

Praun and Hoppe [2003] use a sphere as an intermediate domain to parametrize a surface onto an octahedron, for subsequent geometry image remeshing. By directly optimizing the octahedron-to-surface map, we obtain improved results. The inset figure shows the Venus head as a geometry image obtained by unfolding an octahedral parametrization. As shown in Table 1, the parametrization stretch efficiency is improved in all cases, and the geometric accuracy of the remeshes (as measured with PSNR) is also improved for models with many extremities.



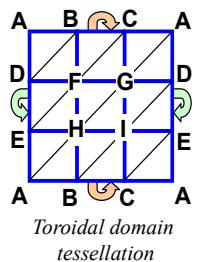
Model	One-way $L^2$ stretch efficiency		Remesh PSNR (dB)	
	$D \rightarrow S \rightarrow M^2$	$D \rightarrow M^2$	$D \rightarrow S \rightarrow M^2$	$D \rightarrow M^2$
Venus	0.943	0.947	83.4	83.2
Bunny	0.706	0.717	80.0	79.9
Gargoyle	0.643	0.679	79.2	79.3
Armadillo	0.454	0.528	72.0	73.0
Horse	0.363	0.398	76.9	77.7
Cow	0.405	0.440	74.9	77.0
tyrannosaurus	0.360	0.418	73.6	74.5

Table 1. Comparison of octahedral remeshing using spherical parametrization ( $D \rightarrow S \rightarrow M^2$ ) [Praun and Hoppe 2003], and using our direct map onto octahedron domain  $D$ .

### 6.4 Toroidal parametrization

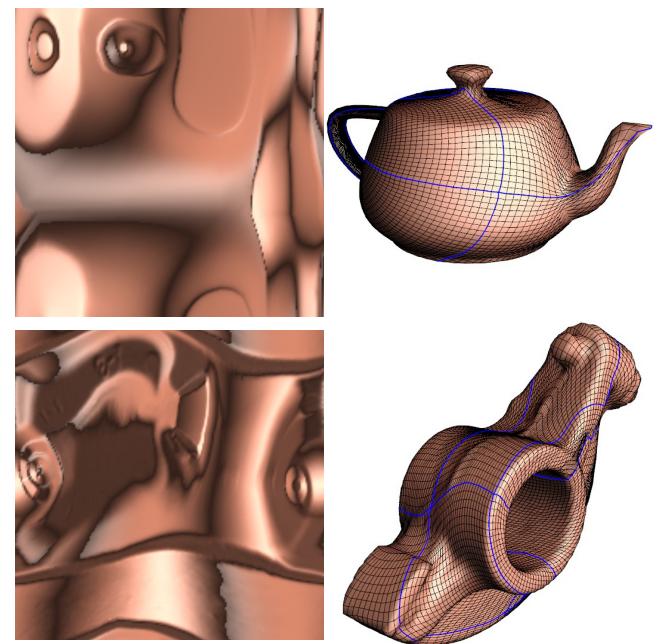
A natural domain for genus-1 surfaces is the toroidal unit square. It is formed by identifying the square’s boundaries left-to-right and top-to-bottom. To apply our framework to this scenario, we let the toroidal domain be represented by a mesh  $M^1$  with 9

vertices and 18 triangles. As in simplicial parametrization, the domain  $M^1$  does not have a *global* isometric embedding in  $\mathbb{R}^3$ , but again we can use the local geometry of the domain when constructing the neighborhoods  $\mathcal{N}(\tilde{v})$  and  $\mathcal{N}(\hat{v})$  in Section 5.1. In this case, the triangles in  $\mathcal{N}(\tilde{v})$  are always right isosceles triangles, and their configuration is such that  $\mathcal{N}(\tilde{v})$  is planar. Thus, the *local* map  $\mathcal{N}(\tilde{v}) \rightarrow \mathcal{N}(\hat{v})$  is always an isometry.



To initialize the parametrization, the user specifies 9 feature points on the input mesh  $M^2$ , to correspond with the domain mesh vertices. To allow maximum freedom for the map, these feature points do not act as constraints during coarse-to-fine optimization. Figure 11 shows some example results.

There has been little work on toroidal parametrizations of arbitrary genus-1 surfaces, which is surprising since the domain is the most “Euclidean” of all closed surface topologies. Gu and Yau [2003] demonstrate their global conformal approach on genus-1 surfaces. Compared to their results, ours exhibit less scale-distortion due to the use of a stretch functional.



Surfaces mapped into toroidal domain (with 2-sided lighting)      Remeshed surfaces  
(all vertices have valence exactly 4)

Figure 11. Examples of toroidal parametrization and remeshing. (One-way stretch efficiencies: teapot 0.458, rocker arm 0.582).

## 7. Discussion

An earlier implementation of our method followed a more traditional parametrization approach, with a static domain and only one mesh being optimized using a coarse-to-fine algorithm. To initialize the map we used conformal maps to establish correspondences between the domain vertices and the large base domain faces of the progressive mesh. This method presented two difficulties: (1) some patches were too large to robustly parametrize using a single linear system and (2) having formed this initial map, there was no way to effectively improve it (since it was “stuck” at a fine level). Our symmetric coarse-to-fine approach overcomes both these difficulties.

An important property of directly optimizing the map between two surfaces is that the correspondence of geometrically similar features is encouraged within the distortion metric itself, thereby requiring fewer manually specified features. For example, only 4 features are sufficient to obtain a good map between the cow and the horse. These 4 features on the hooves are needed to prevent a combinatorial optimization, i.e. which cow leg corresponds to which horse leg. Our mapping problem shares similarities with the problem of obtaining a rigid correspondence between two objects. Mesh registration energy functionals typically have many local minima and thus require initial user guidance, but importantly they have a deep energy well near the global solution.

The major difference between our symmetric coarse-to-fine refinement process and previous simplicial parametrization methods is the opportunity for fine-grain optimization. Simplicial parametrization methods apply linear relaxation operations across coarse domain faces, whereas we apply non-linear optimization on individual vertices of both meshes.

While our method achieves impressive results for a large class of applications, its main current limitation is execution time. The mutual tessellation is more complex than either of the input meshes, and managing it during optimization is time-consuming. Currently our implementation takes a couple of hours to create inter-surface maps between meshes of ~64K faces. For the simplicial, octahedral, and toroidal parametrization scenarios, where  $M^1$  is coarse, it takes about 20 minutes to create the map. The space complexity of the mutual tessellation could theoretically be  $O(n^2)$  for a pathological worst case, but for ordinary models it is about  $8n$ , i.e. a small factor more than the  $2n$  vertices from the two meshes. In practice, memory usage has not been an issue.

Another conceptual drawback of our current implementation (though not of the method in general) is the asymmetry of the data structure, which only allows one of the meshes to be refined at a time. A truly symmetrical implementation allowing fine-grain interleaved refinement of both meshes would be more elegant.

## 8. Future work

There are several avenues for future work. To improve speed we envision using fine-to-coarse propagation of information [Sander et al 2002] to obtain better configurations at low resolutions. Huge meshes could be handled using a hybrid strategy; after running our ISM algorithm to create a good mid-resolution map, we could define the finer map using simplicial map composition, since the simplicial pieces may be small and flat enough to avoid numerical problems and geometric detail mismatch.

One exciting application is the use of inter-surface maps to automatically transfer geometric texture between models. This may allow surface texture synthesis using other surfaces as exemplars.

An interesting open question is how to extend our method to handle multiple models. Simultaneously optimizing an all-to-all map would not scale, while using one model as domain would lose some benefits of directly optimizing inter-surface maps.

Another area of future work is computing maps with singularities to allow correspondences between objects with different topologies. User input may be required to associate topological features and introduce singularities on some of the meshes.

## Acknowledgements

We thank Cyberware and the Digital Michelangelo Project at Stanford University for the 3D models, and Andrei Khodakovsky for sharing his Globally Smooth Parametrization data.

## References

- AKSOYLU, B., KHODAKOVSKY, A., AND SCHRÖDER, P. 2003. [Multilevel solvers for unstructured surface meshes](#). *SIAM J. Sci. Comput.*
- ALEXA, M. 2002. [Recent advances in mesh morphing](#). *Computer Graphics Forum*, 21(2), 173-196.
- DESBRUN, M., MEYER, M., AND ALLIEZ, P. 2002. [Intrinsic parameterizations of surface meshes](#). *Computer Graphics Forum*, 17(2), 167-174.
- ECK, M., DEROSSE, T., DUCHAMP, T., HOPPE, H., LOUNSBERRY, M., AND STUETZLE, W. 1995. [Multiresolution analysis of arbitrary meshes](#). *ACM SIGGRAPH*, 173-182.
- FLOATER, M. 2003. [Mean value coordinates](#). *CAGD*, 20(1), 19-27.
- FLOATER, M., AND HORMANN, K. 2003. [Recent advances in surface parameterization](#). *Multiresolution in Geometric Modeling Workshop*.
- GOTSMAN, C., GU, X., AND SHEFFER, A. 2003. [Fundamentals of spherical parameterization for 3D meshes](#). *ACM SIGGRAPH*, 358-363.
- GU, X., GORTLER, S. J., HOPPE, H. 2002. [Geometry images](#). *ACM SIGGRAPH*, 355-361.
- GU, X., YAU, S. 2003. [Global conformal surface parameterization](#). *Symposium on Geometry Processing*, 127-137.
- GUSKOV, I., VIDIMČE, K., SWELDENDS, W., AND SCHRÖDER, P. 2000. [Normal meshes](#). *ACM SIGGRAPH*, 95-102.
- HAKER, S., ANGENENT, S., TANNENBAUM, S., KIKINIS, R., SAPIRO, G., AND HALLE, M. 2000. [Conformal surface parametrization for texture mapping](#). *IEEE TVCG*, 6(2), 181-189.
- HOPPE, H. 1996. [Progressive meshes](#). *ACM SIGGRAPH*, 99-108.
- HORMANN, K., AND GREINER, G. 1999a. [MIPS: An efficient global parametrization method](#). *Curve and Surface Design*, 153-162.
- HORMANN, K., GREINER, G., AND CAMPAGNA, S. 1999b. [Hierarchical parametrization of triangulated surfaces](#). *Vision, Modeling, and Visualization*, 219-226.
- KHODAKOVSKY, A., LITKE, N., AND SCHRÖDER, P. 2003. [Globally smooth parameterizations with low distortion](#). *ACM SIGGRAPH*, 350-357.
- KRAEVOY, V., SHEFFER, A., AND GOTSMAN, C. 2003. [Matchmaker: constructing constrained texture maps](#). *ACM SIGGRAPH*, 326-333.
- KRAEVOY, V., AND SHEFFER, A. 2004. [Cross-parameterization and compatible remeshing of 3D models](#). *ACM SIGGRAPH*.
- LAZARUS, F., POCCHIOLA, M., VEGTER, G., AND VERROUST, A. 2001. [Computing a canonical polygonal schema of an orientable triangulated surface](#). *ACM Symposium on Computational Geometry*, 80-89.
- LEE, A., SWELDENDS, W., SCHRÖDER, P., COWSAR, L., AND DOBKIN, D. 1998. [MAPS: Multiresolution adaptive parametrization of surfaces](#). *ACM SIGGRAPH*, 95-104.
- LEE, A., DOBKIN, D., SWELDENDS, W., AND SCHRÖDER, P. 1999. [Multiresolution mesh morphing](#). *ACM SIGGRAPH*, 343-350.
- LÉVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J. 2002. [Least squares conformal maps for automatic texture atlas generation](#). *ACM SIGGRAPH*, 362-371.
- MAILLOT, J., YAHA, H., AND VERROUST, A. 1993. [Interactive texture mapping](#). *ACM SIGGRAPH*, 27-34.
- PRAUN, E., SWELDENDS, W., AND SCHRÖDER, P. 2001. [Consistent mesh parametrizations](#). *ACM SIGGRAPH*, 179-184.
- PRAUN, E., AND HOPPE, H. 2003. [Spherical parametrization and remeshing](#). *ACM SIGGRAPH*, 340-349.
- SANDER, P., SNYDER, J., GORTLER, S., AND HOPPE, H. 2001. [Texture mapping progressive meshes](#). *ACM SIGGRAPH*, 409-416.
- SANDER, P., GORTLER, S., SNYDER, J., AND HOPPE, H. 2002. [Signal-specialized parametrization](#). *Eurographics Workshop on Rendering*, 87-100.
- SHEFFER, A., AND HART, J. 2002. [Seamster: Inconspicuous low-distortion texture seam layout](#). *IEEE Visualization*, 291-298.
- SORKINE, O., COHEN-OR, D., GOLDENTHAL, R., AND LISCHINSKI, D. 2002. [Bounded-distortion piecewise mesh parametrization](#). *IEEE Visualization*, 355-362.
- TURK, G. 1992. [Re-tiling polygonal surfaces](#). *ACM SIGGRAPH*, 55-64.