Surface Mapping using Consistent Pants Decomposition

Xin Li, *Student Member, IEEE,* Xianfeng Gu, *Member, IEEE,* and Hong Qin, *Member, IEEE*

Abstract—Surface mapping is fundamental to shape computing and various downstream applications. This paper develops a *pants decomposition* framework for computing maps between surfaces with arbitrary topologies. The framework first conducts pants decomposition on both surfaces to segment them into consistent sets of *pants patches* (a pants patch is intuitively defined as a genus-zero surface with three boundaries), then composes global mapping between two surfaces by using harmonic maps of corresponding patches. This framework has several key advantages over existing techniques. First, it is automatic. It can automatically construct mappings for surfaces with complicated topology, guaranteeing the one-to-one continuity. Second, it is general and powerful. It flexibly handles mapping computation between surfaces with different topologies. Third, it is flexible. Despite topology and geometry, it can also integrate semantics requirements from users. Through an simple and intuitive human-computer interaction mechanism, the user can flexibly control the mapping behavior by enforcing point/curve constraints. Compared with traditional user-guided, piecewise surface mapping techniques, our new method is less labor-intensive, more intuitive, and requires no user's expertise in computing complicated surface map between arbitrary shapes. We conduct various experiments to demonstrate its modeling potential and effectiveness.

Index Terms—Mathematics of Computing, Computer Graphics, Computational Geometry and Object Modeling, Geometric algorithms, languages, and systems.

1 Introduction

Computing bijective surface mappings is one of the most fundamental problems in modeling and simulation fields and their engineering applications. Its primary goal is to build up a one-to-one correspondence from one shape to another. This mapping has been widely used as an enabling tool for numerous applications such as shape analysis, retrieval, shape morphing, texture/attribute/motion reuse, recognition, etc.

Techniques for surface mapping computation can be classified into implicit methods and explicit methods. Implicit methods typically make use of the volumetric concept. Such methods usually pay less attention to the underlying topology. In addition, they do not require surface models' generation from many real-world raw data acquired from scanners. However, their drawbacks are also obvious – they are computationally more expensive, because volume-based techniques must consider one more dimension. The lack of efficiency significantly restrains their application scopes in practice. In contrast, the majority of surface mapping techniques is based on the explicit approach (our method presented in this paper falls into this category). Such an approach only uses surface's information (e.g., mesh's connectivity and vertices' positions) for the mapping computation. Compared with volume-based techniques, it is more efficient and direct for most graphics/modeling/visualization applications.

Explicit methods need to consider surface topology carefully. When the input surfaces are with complicated topology, i.e., with a large amount of handles or boundaries, the computation is much more challenging and a large portion of existing research focuses on cases dealing with surfaces with trivial topology. Furthermore, when we try to map a surface to another surface with different topological type, the one-to-one continuity could not be guaranteed, and some tearing has to happen in some specific regions. Much less current work tackles this case due to its technical difficulties.

We further classify the explicit surface mapping techniques into two general approaches. One is local approach, or the piecewise method (segmentation + local mapping); and the other is the global approach (global mapping without segmentation). The global approach works well for genus-0 surfaces but meets great challenge when dealing with cases of non-trivial topological surfaces; [1] presented a method to compute the globally optimized surface mapping. However, the majority of surface mapping methods ([2], [3], [4], [5], [6]) follow the direction of local approach due to its efficiency and controllability. The basic idea of local approach is to firstly partition two surfaces into two consistent sets of sub-regions with simple topology, then compute locally optimized mapping between corresponding sub-regions, and finally compose the local results to a global continuous map.

Although surface mapping is important and has been widely studied, state-of-the-art surface mapping techniques are far from adequate and perfect. A more desirable and powerful surface mapping method is needed

- *X. Li is with Department of Computer Science, Stony Brook University (SUNY), Stony Brook, NY 11790. E-mail: xinli@cs.sunysb.edu*
- *X. Gu and H. Qin are with Stony Brook University.*

Manuscript received July 24, 2008.

and should have the following properties.

The first one is **generality**. The mapping methodology should be general, i.e., it should be able to handle surfaces with arbitrary topology, with or without boundaries. The generality also includes another important issue – being capable of accommodating topology changes. We can see the importance of topology change in surface mapping from its applications. When we use surface mapping for shape comparison and difference analysis, data to be registered could easily have different topology due to shape variations and accompanying noises (e.g., small boundaries and tiny handles). Moreover, when we use surface mapping to drive the animation of a morphing sequence, we usually transform one object to another based on their intrinsic semantics, regardless of whether they have the same topology or not (see Fig. 14(a)).

Many existing surface mapping techniques primarily focus on genus-zero surfaces, and recent works start to aim at general surfaces, yet much fewer techniques have been devised to flexibly work for arbitrary topological changes. In this work, we aim at a general framework that can handle arbitrary mesh inputs.

The second property is **automation**. Most current surface mapping techniques heavily rely upon large amount of user intervention. Although in many applications, the requirements of object semantics forbid us from entirely ignoring user intentions, the primary reason for the lack of fully automatic methods in this research field are still due to technical difficulties. Real-world shapes could be complex in both topology and geometry. To our best knowledge, if the given surfaces are topologically non-trivial (neither sphere-like nor disk-like), even with the same topology, no existing techniques are able to compute the mapping in a fully automatic way. A key difficulty stems from that, although most current mapping methods depend on a preprocessing stage of mesh segmentation, few surface segmentation techniques have been devised for automatically providing consistent segmentation on different surfaces.

When mapping is used in applications dealing with large amount of data, such as analysis and comparison on shapes in database, user involvement on every registration trial could not be practical. Therefore, we definitely need a surface mapping technique that works for general inputs, yet is as automatic as possible.

The third property is **controllability**. Although automation makes the mapping process much less labor-intensive, in applications where the semantics plays a critical role, such as morphing (requiring feature points matching), automatic methods based on pure topology and geometry fail. We must have a new mechanism that can provide an easy way to let the user manage the behavior according to semantics-specific requirements. Indeed, current surface mapping techniques oftentimes provide limited control to the user; but for surfaces with complicated topology, they either require a large number of markers [5] or need user's great efforts to design the base mesh as a good starting point [2], [7]. In principle, a good mapping framework should provide an intuitive and easy-to-use human computer interaction.

Fourth, it is also important to emphasize **completeness**. The global continuity is typically required for the underlying mapping. However, between given surfaces, there may exist many continuous yet topologically different mappings, i.e., mappings could have different homotopy types (see Fig. 9). Two surface mappings belong to the same homotopy type if and only if they can continuously deform to each other without degeneracy. Among so many legitimate choices, there are no viable ways to select the best ones from all candidates, since different homotopy may represent different semantics. In such a case, being able to let user easily and intuitively determine arbitrary topological type of a mapping not only demonstrates the completeness of the mapping algorithm, but also has practical importance.

In this paper, we design a new surface mapping framework in order to unify the above four properties. We conduct our experiments on several challenging examples to demonstrate the power and potential of our method. Our **contributions** are as follows.

- Generality: Our framework flexibly handles surfaces with arbitrary topology, with or without boundaries. It also handles surface mapping with topological changes.
- Automation: Our framework has great automation. When a set of surfaces are with complicated topology, our decomposition can generate the consistent segmentation automatically.
- Controllability: When for any semantics reasons, feature alignment is necessary; user interaction can be easily applied. Our framework coherently aligns constraint points or curves to enforce constraints, and provides users a simple and intuitive mechanism to control the mapping behavior.
- Completeness: Our framework can enumerate different homotopy types of mappings. It shows the completeness (as well as controllability) of our framework, demonstrating the rigorousness of this method from the mathematical point of view.
- Efficiency: The technical core of the decomposition is simple and efficient, the algorithm primarily relies on the Dijkstra algorithm, and only the triangular metric of given surfaces is employed.

The remainder of this paper is organized as follows. We briefly review the prior work in Section 2, then introduce theoretic background as well as necessary terms and definitions in Section 3. The fundamental idea of our framework is illustrated in Section 4, which is a two-step pipeline, as discussed in Section 5 and Section 6, respectively. Some implementation details and discussion are given in Section 7. Finally, we demonstrate experimental results with various applications in Section 8 and conclude the paper in Section 9.

2 Related Work

Surface mapping is a fundamental problem in computer graphics/modeling fields. In order to build up a one-to-one mapping from one surface to another, we can use a straightforward yet effective method that uses a regular domain as the bridge. Mappings from a surface to common regular domain such as plane or sphere is usually called *surface parameterization*. Surface parameterization has been extensively studied, and have been playing an important role in the modern graphics, modeling and geometric processing pipeline. A survey of parameterization is beyond the scope of this work, and we refer the reader to [8] and [9]. In the following, we skip the review of parameterization, and only briefly recap the surface mapping techniques.

Earlier work on computing inter-surface mappings is mostly motivated by the need of shape blending. The natural and intuitive approach, which uses the canonical planar or spherical domain and establishes surface correspondence using parameterization techniques, is best suitable on the genus-zero case.

For genus-zero meshes, the sphere (for closed surfaces) and the plane (for open surfaces) are naturally widely-used intermediate domains. Kent et al. [10] projected star-shaped surfaces onto spheres, and merged them by clipping one sphere to the other. Kanai et al. [3] used harmonic maps to build the correspondence from surfaces to the unit disk domain, therefore not only the star-shaped surfaces, but also all genus-zero closed or open surfaces can be mapped easily. However, it only allowed one constraint point from users. Alexa [11] proposed to match multiple feature points between genus-0 surfaces. His work wrapped two surfaces onto a unit sphere by minimizing a distance function, and feature points on the surface were aligned and the resultant embedding was used for the surface mapping. They started to aim for matching multiple feature points. However, its limitation is that no bijectivity is guaranteed and hard constraints may not be fully enforced. More recently, Asirvatham et al. [4] used their constrained spherical parameterization to map genus-zero surfaces onto the sphere, the progressive mesh was used to get a simple base mesh and to enforce constraints at certain positions on the sphere. This method allows multiple hard constraint points between genus-0 surfaces.

For surfaces with more general topology, common canonical domains such as disks and spheres become unavailable. Directly solving intra-surface mapping usually fails. Most techniques, as we mentioned in the previous section, first segment surfaces into consistent sets of sub-regions, then compose or refine the global result from the sub-region mappings. DeCarlo and Gallier [2] designed a surface mapping framework based on user-specified base meshes. When base meshes are carefully designed, the framework is flexible, and mapping between surfaces with different topology can be computed. However, deep domain knowledge in topological surgery may be required to manually design consistent base meshes; and when the surface has high genus, the design are usually quite complicated. Only examples up to genus-2 were provided in their work. Gregory et al. [7] and Zöckler et al. [12] also used the base mesh approach. When the consistent "base mesh" have been manually designed, harmonic or barycentric mappings are used to correspond these sub-regions accordingly. More earlier surface mapping work for morphing applications can be found in the survey [13].

Recent work has been trying to seek more automatic methods to consistently generating the base mesh. Lee et al. [14] used their "MAPS" algorithm to hierarchically map fine meshes onto a common base mesh. Praun et al. [15] introduced a graph tracing algorithm to transfer the coarse base mesh from one surface to another with the same topology. Kraevoy and Sheffer [5] designed another algorithm to trace out base meshes consistently on different surfaces. To build up the base meshes, many feature points have to be provided by users for high genus surfaces. For example, at least four points are required for each topological handle to proceed the base mesh tracing algorithm. Schreiner et al. [6] first traced original surfaces into a corresponding set of triangular patches, with feature points as path endpoints, and created original surfaces' progressive mesh representations. Then they created a trivial map onto the base mesh, and iteratively refined the map back to the original surfaces.

Base mesh construction (consistent segmentation) in many of these aforementioned works consumes a large amount of human labor, which motivates the recent research direction of the automatic generating segmentation on surfaces and its subsequent mapping framework. This automation becomes very challenging for surfaces with complicated topology, where little work has been explored. Furthermore, when given surfaces with different topologies are present, it is even more difficult. Manual base mesh design [2] requires greater effort and stronger expertise from the user. This is the motivation for us to seek the automatic decomposition for consistent shape segmentation for surfaces with complicated topology.

A topological issue should be considered for mapping between surfaces with nontrivial topology. It is the so-called homotopy type of mappings. In [16] and [1], canonical homology bases [17] and systems of loops [18] were used to study this issue and build mappings of different homotopy types. [19] defined terms *handle loops* and *tunnel loops*, which provide another intuitive way to study the topological handles on surfaces; they also introduced a practical computation algorithm to compute tunnel and handle loops respectively. In our work, if a given surface has non-trivial topology, our algorithm takes the surface as well as its handles and tunnel loops as inputs.

Surface Pants decomposition has been widely studied [20]. Work has been done to investigate the optimal segmentation of a given surface into pants [21]. For

surface mapping purpose, instead of decomposing just one surface, we need to compute consistent decomposition on several surfaces, or find canonical decomposition for the same types of surfaces. Less work has been accomplished along this direction.

3 Theoretical Foundation

3.1 Definition of Pants Decomposition

We briefly introduce the related background in topology and geometry and make necessary definitions in this section.

A *surface* M is a topological Hausdorff space in which each point has a neighborhood homeomorphic to either the plane or the closed half-plane. Points with closed half-plane neighborhood are defined as the *boundary* of M.

A *path* is a continuous map $p : [0, 1] \to M$. A *loop*(cycle) is a closed path, meaning that the endpoints $p(0)$ and $p(1)$ coincide. The *concatenation* of two paths p and q, with $p(1) = q(0)$ is the path $p \circ q$ defined by

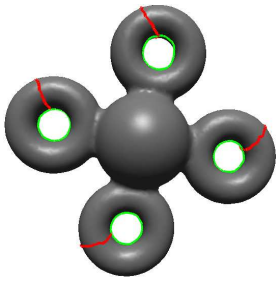
$$(p \circ q)(t) = \begin{cases} p(2t), & t \leq 1/2; \\ q(2t-1), & t \geq 1/2. \end{cases}$$

When we say two paths are *homotopic*, it means one path can continuously evolve to the other one through a family of paths on the surface. Rigorously speaking, a homotopy between paths p and q is a continuous map $h : [0, 1] \times [0, 1] \to M$ s.t. $h(0, \cdot) = p$, $h(1, \cdot) = q$, $h(\cdot, 0) = a$, $h(\cdot, 1) = b$, where a and b are two paths joining $p(0)$ with $q(0)$ and $p(1)$ with $q(1)$, respectively. We denote the homotopy equivalence class of path p as $[p]$.

All homotopy classes under the product $[p] \circ [q] = [p \circ q]$ form a group called the *fundamental group*, denoted as $\pi_1(M)$. Suppose $f : M \to M'$ is a continuous map, p is a loop on M, then $f \circ p$ is a loop on M'. f maps the homotopy class $[p]$ to the homotopy class $[f \circ p]$, and f induces a homomorphism $f_* : \pi_1(M) \to \pi_1(M')$. Suppose $f^1, f^2 : M \to M'$ are two continuous maps between M and M', we say f_1 and f_2 are homotopic, if and only if they induce the same homomorphism between the fundamental groups $f^1_* = f^2_*$.

A pair of *pants* is a genus-0 surface with 3 boundaries. A *pants decomposition* of M is an ordered set of simple, pairwise disjoint cycles that splits M into pairs of pants. Every compact orientable surface, except the sphere, disk, cylinder, and torus, has pants decomposition. If M is of genus G and has B boundaries, a pants decomposition is made of $3G + B - 3$ cycles [20]. In this work, we present an automatic decomposition algorithm to cut surface apart iteratively along certain non-trivial loops. The $3G + B - 3$ cycles segment the given surface M to $2G-2+B$ pairs of pants $(M_0, \ldots, M_{2G-2+B-1})$. Each pair of pants M_i (for simplicity, we also call such a surface patch a *pants patch* in the remaining part of this paper) has three boundaries, which are denoted as the waist Γ^0_i, and two legs Γ^1_i, Γ^2_i. Two pants M_i and M_j are adjacent if they share boundaries.

3.2 Handle and Tunnel Loops



Suppose a closed embedded surface $M \subset \mathbb{R}^3$ separates $\mathbb{R}^3$ into a bounded space $\mathbb{I}$ and an unbounded space $\mathbb{O}$. Handle and tunnel loops of M can be defined as follows (see also [19]). A loop b_i is a handle if it spans a disk in the bounded space $\mathbb{I}$; if one cuts M along b_i and fills the boundary with that disk, one eliminates a handle. A loop a_i is a *tunnel* if it spans a disk in the unbounded space $\mathbb{O}$; and its removal eliminates a tunnel. The handle and tunnel loops characterize important topological information of the surface, and we use them to determine the homotopy types of our mappings. An intuitive illustration is shown in the above figure. Red curves represent the handle loops while the green ones are tunnel loops. More details about handle and tunnel loops are addressed in [19]. All handle loops form a basis of $\pi_1(\mathbb{I})$, and all tunnel loops form a basis of $\pi_1(\mathbb{O})$. The union of their homology classes form a basis of $\pi_1(\mathbb{M})$. Respectively, tunnel loops and handle loops can be effectively computed using techniques presented in [19]. Our algorithm takes surfaces, their handle and tunnel loops as input.

4 Overview of Key Ideas

Pants as Decomposition Elements. The core of our mapping framework is consistent pants decomposition. Given a set of arbitrary surfaces of the same topology, our decomposition scheme canonically segment these surfaces into consistent sets of patches with "pants" topology. The reason that "pants" is used as the decomposition element rooted in topology. Surfaces are topologically classified by their handle numbers and boundary numbers, and characterized by Euler numbers. For example, a genus-G surface with B boundaries has its Euler number $\chi = 2 - 2G - B$. The Euler numbers for surfaces of most topological types are negative integer (except for some trivial cases, e.g. $G = 0, B = 0$, see below for detailed discussion). The Euler number of a pants patch is -1, and therefore pants decomposition provides a canonical decomposition scheme, partitioning this surface into $2G + B - 2$ pants patches.

Decomposing a Surface. The first step of our mapping framework is pants decomposition. It segments the given surface into a set of pants. Once the indexed handle and tunnel loops $(a_i, b_i, 0 \leq i < G)$ are provided in the preprocessing stage 7.1, the subsequent decomposition is unique and we will obtain an ordered set of pants.

To give an intuitive overview, we start from a closed genus-G ($G \geq 2$) surface M. In Fig. 1, a genus-4 torus example is used to illustrate key steps of our pipeline. First, we remove G handle patches from M (a), and get a set of genus-one surfaces $M_i, (0 \leq i < G)$ with one boundary and (if $G \geq 3$), a topological sphere M'

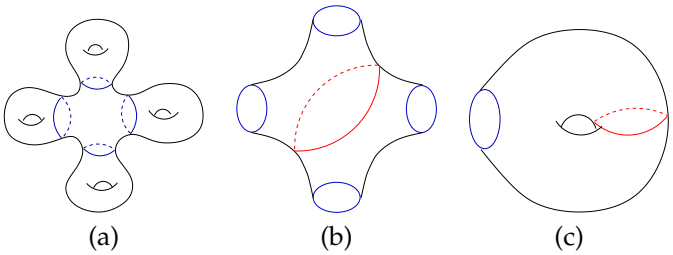


Fig. 1. Pants Decomposition Pipeline. (a) Find and remove "waists" of handles. (b) and (c) Decompose the base patch and handle patches.

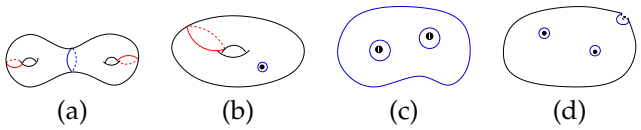


Fig. 2. Pants Decomposition on Surfaces with Simple Topology. (a) A genus-2 surface has 2 handle patches, no base patch. (b) A closed genus-1 surface ($\chi = 0$) needs one punch. (c) A topological disk ($\chi = 1$) needs two punches. (d) A topological sphere ($\chi = 2$) needs three punches.

with G boundaries. We call M' the *base patch* and these boundaries *waists*. Second, we decompose the base patch M' and all the handle patches M_i into pants patches ((b) and (c)). For genus-2 surfaces, no base patch exists, 2 handle patches M_0 and M_1 compose the segmentation (Fig. 2(a)).

When surfaces have high genus ($G \geq 2$) with boundaries, we leave boundaries on the base patch M', treat them as usual "waists", and apply base patch decomposition similarly.

We can also decompose surface M with more trivial topology ($G < 2$) with some extra "holes". The basic idea is, the Euler number of a pair of pants is -1, so if M's Euler number $\chi = 2 - 2G - B$ is negative (for example, $G \geq 2$ will guarantee $\chi < 0$), M can be decomposed directly. Otherwise, we punch holes on M until M gets a minus Euler number. One punch decreases the Euler number by 1, and since the Euler number of a surface can not exceed $+2$, we at most need 3 punches.

More specifically, if M is genus-1 and has a boundary, $\chi_M = -1$, it is directly processed as a handle patch M_i. If the surface M is genus-1 and closed (Fig. 2(b)), $\chi_M = 0$, one marker is required from the user. We punch a hole on the marker, get a boundary and make $\chi_M = -1$ so that it can also be decomposed like M_i. If M is a genus-0 surface with a boundary (Fig. 2(c)), like a disk, then $\chi_M = 1$. We already have one "waist", and need two more punches as "legs". If the surface is a closed genus-0 surface (Fig. 2(a)), three markers are required to form a pair of pants.

When a genus-zero or genus-one surface has more than one boundary, similarly we compute its Euler number and check whether extra punches are necessary. As we will discuss in the later part of this paper, these markers can be used as feature points in the surface mapping because their exact correspondence is guaranteed.

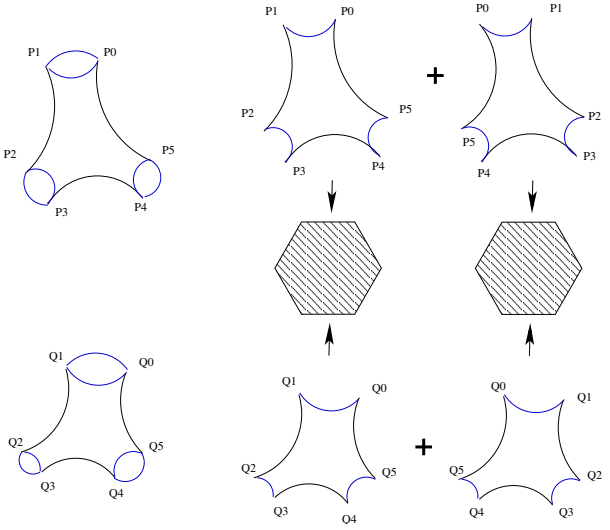


Fig. 3. Mapping Two Pants Patches. Each pair of pants is decomposed into two topological hexagon patches. Harmonic maps from these patches to regular hexagons are used to compose the pants mapping.

Pants Mapping. When surfaces are decomposed into a set of sub-regions, each with "pants" topology, the mapping computation becomes easier. To make sure the map has global continuity and bijectivity, boundary mappings on neighboring sub-regions have to be consistent. Many mapping techniques with fixed boundary conditions, such as harmonic map [22], mean value coordinate map [23] and so on can all be the choice for pants mapping. Free boundary mapping techniques are not preferred for sub-regions mapping here because if we cannot control the boundary mapping behavior, it will fail to satisfy continuity and bijectivity along the sub-regions' boundaries.

In this work, we use the harmonic map, because it is physically natural and can be computed efficiently. Like other fixed boundary mapping techniques, the shape of the target regions should be convex to guarantee the map's existence and validity. Such a direct convex domain for shapes with pants-topology does not exist; therefore, we simply decompose the pants into two patches with disk-like topology to make the mapping computation stable. As illustrated in Fig. 3, each pants patch is decomposed into two topological hexagons, and each hexagon is harmonically mapped to a regular hexagon. The pants mapping is then composed through these hexagonal domains.

5 Consistent Pants Decomposition

This section introduces our algorithm and implementation of the consistent pants decomposition on surfaces. The pipelines are introduced in Section 5.1, Section 5.2, and Section 5.3, respectively.

5.1 Removing Handles

The first step of the pipeline is to remove handle patches from a given surface M. We iteratively trace a special shortest cycle w_i that topologically bounds a handle and slice M along it to separate this handle patch (which becomes a pants patch M_i later) from M. w_i indicates that we make it the "waist" of M_i. Given the handle loop (a_i) and tunnel loop (b_i), the cycle w_i is homotopic to $c_i = a_i^1 \circ b_i^1 \circ a_i^{-1} \circ b_i^{-1}$. The computation of w_i is not trivial, which is illustrated using the following example.

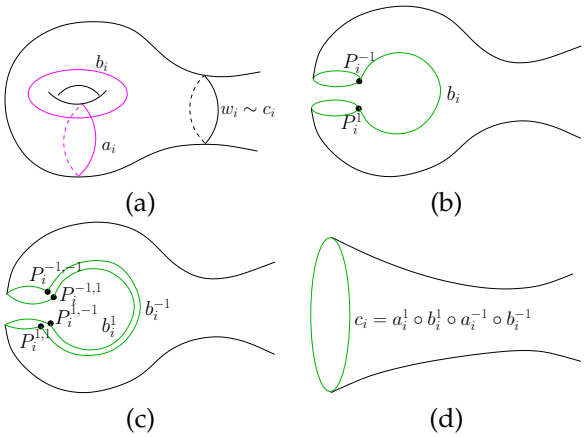


Fig. 4. Computing c_i of the handle i. (a) w_i is the waist, but we need to compute c_i first. (b) Slice a_i apart, get P_i^1 and P_i^{-1}. (c) Slice b_i apart, P_i^1 and P_i^{-1} split to $P_i^{1,1}$, $P_i^{1,-1}$, $P_i^{-1,1}$ and $P_i^{-1,-1}$ separately. (d) The newly generated boundary is c_i.

Step One: Compute c_i.
Fig. 4 shows a surface patch near the handle and illustrates the idea. In this step, we compute the curve $c_i = a_i^1 \circ b_i^1 \circ a_i^{-1} \circ b_i^{-1}$ which is homotopic to w_i. As (b) and (c) show, we slice a_i and b_i along their intersection point, and get the resultant green boundary in (d): $c_i = P_i^{1,1} \to P_i^{1,-1} \to P_i^{-1,-1} \to P_i^{-1,1} \to P_i^{1,1}$.

Step Two: Shrink c_i to the "Waist" w_i.
As shown in Fig. 5, in step one, we sliced apart M ((a)) and get all its c_i ((b)). Now we iteratively shrink each c_i to its shortest homotopic cycle w_i. It is computed through the following algorithm 1:

Algorithm 1: Shrink c_i to w_i.

In: Surface M, c_i.
Out: The shortest loop w_i homotopic to c_i.

1. Connect all existing boundaries except c_i together using shortest paths (dashed green curves in Fig. 5(b)).
2. Slice these paths, we get one new large boundary c_i' (Fig. 5(c)). M becomes a topological cylinder.
3. Compute the shortest path γ (green curve in Fig. 5(c)) connecting the cylinder's two boundaries.
4. (The shortest cycle w_i at least intersects once with γ) Slice γ apart, every point $p \in \gamma$ splits to a pair $(P, \tilde{P})$. Find the point pair $(P, \tilde{P})$ having the minimal length of shortest connecting path. This shortest path is w_i (blue curve in Fig. 5(c)).

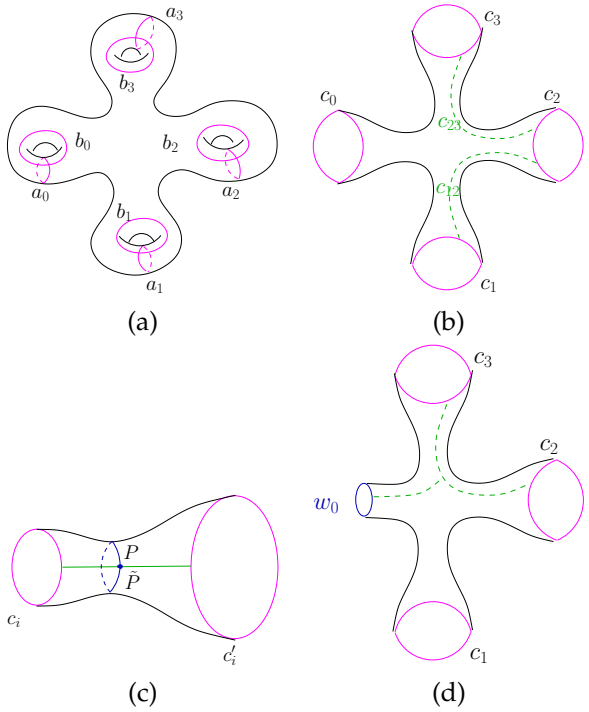


Fig. 5. Computing the "Waist". (a) Slice M apart along its handle/tunnel loops, get boundaries c_i. (b) Connect all other boundaries $j, (j \neq i)$ to a large boundary c_i', and get a topological cylinder. (c) Slice apart γ (green) connecting c_i and c_i'; get a topological "trapezoid"; compute w_i (blue) as the shortest path connecting boundary point pairs. (d) Continue the process on other handles.

Here, the shortest path connecting two curves can be straightforwardly computed in $O(n \log n)$ by a slightly modified Dijkstra algorithm: first, set all points on one curve as the initial starting set, second, keep propagating until a vertex on the target curve is hit, finally, trace the path from this hit point on the target curve back to the point on the source curve and this is the shortest path that we want.

When we get w_0 from c_0, we remove the sub-region bounded by c_0 and w_0 from M, and denote it as M_0. We go on processing the above algorithm on other $c_i, i = 1 \dots G-1$ on the new M, only substituting c_0 by w_0 as shown in (d).

This process ends after G steps, and we get G handle patches $M_0 \dots M_{G-1}$. Each waist w_i is the shortest cycle on $M \setminus \cup_{j=0}^{i-1} M_j$, making the segmentation geometrically optimal under this order. We also get a leftover patch M, which is a topological sphere with G holes, denoted as the base patch M'. We further decompose M' and all the M_i into pants patches in the following sections.

5.2 Decomposing the Base Patch

The base patch M' is a topological sphere with $G + B$ holes (G is the genus, B is the number of original boundaries on surface M). As we mentioned previously, when there are less than 3 boundaries (for example, $G < 3, B = 0$), there is no base patch. In those cases, this

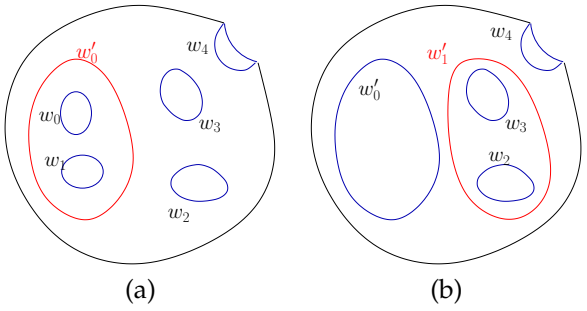


Fig. 6. Decomposing the Base Patch. (a) Slice w_0', get a new pair of pants. Boundary number decreases by 1. (b) Set w_0' as a new boundary, go on to compute w_1'.

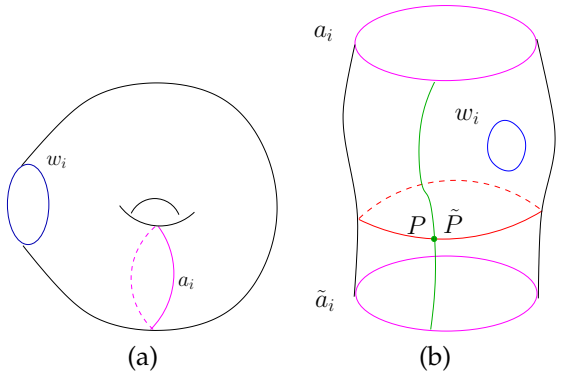


Fig. 7. Decomposing a Handle Patch. (a) Slice a_i apart. (b) Slice apart of the green curve that connects two outer boundaries. Then find the shortest path (red) connecting corresponding point pairs on the green curve.

step is skipped. Fig. 6 illustrates illustrates an example of our base patch decomposition, the algorithm is as follows.

Algorithm 2: Base Patch Decomposition.

In: Base Patch M' and all the waists $w_{i=0\ldots G+B}$.
Out: $G+B-2$ pants patches.

0. Put all boundaries w_i of M' into a queue Q.
1. If Q has ≤ 3 boundaries, end; else goto Step 2.
2. Compute a shortest loop w' homotopic to $w_i \circ w_j$. (Red curves in Fig. 6(a) and (b))
3. w', w_i and w_j bound a pair of pants, remove it from M'. Remove w_i and w_j from Q. Put w' to Q. Goto Step 1.

During each iteration, we decrease the number of boundaries on M' by 1 because two boundaries w_i and w_j are removed, one new boundary w' is created. Therefore, this algorithm will process for $G+B-3$ iterations, and we get $G+B-2$ pants patches ($G+B-3$ from iterations, and base patch becomes the last one).

In step 2, we need to trace a shortest loop w' homotopic to $w_i \circ w_j$. The computation follows the idea of the previous algorithm of shrinking waists (Fig. 5(b) and (c)).

Algorithm 3: Compute shortest loop w' homotopic to $w_i \circ w_j$.

In: Surface M' and two waists w_i, w_j on it.
Out: Shortest (under the given metric) loop w' homotopic to $w_i \circ w_j$.

1. Connect w_i and w_j together by a shortest path w_{ij}.
2. Connect all other loops in Q together with shortest paths without intersecting w_{ij}. These loops together form a new big boundary w_{ij}'
3. w_{ij} and w_{ij}' bound a cylinder (same as in Fig. 5(c)). Compute the shortest cycle w' using the same idea of step 3 and step 4. in Algorithm 1.

5.3 Decomposing Handles Patches

After we find each waist w_i in the pipeline's step one, we remove the handle patches M_i from M, each of which is a genus-1 surface with one boundary. We can simply find a shortest cycle homotopic to the handle loop a_i and slice it apart to make M_i a pants patch.

The idea is illustrated in Fig. 7. We first slice a_i to get a cylinder with an inner boundary w_i; then we find the shortest path γ (green curve in (b)) connecting two outer boundaries. Then we slice γ, and find the shortest "shortest paths" connecting P and $\tilde{P}$, $P \in \gamma, \tilde{P} \in \tilde{\gamma}$. Now we make M_i a pants patch by slicing this shortest cycle (red cycle in (b)).

6 Matching Pants Patches

After we perform consistent pants decomposition on two surfaces M and M' respectively, we get two consistent pants sets $M_i, (i = 0\ldots n)$ and $M_i', (i = 0\ldots n)$. In this section, we map each corresponding pairs of pants patches: $f_i(M_i) \rightarrow M_i'$.

To assure global continuity, mappings across the pants' boundaries should be consistent. Rigorously speaking, if a curve segment γ is on two adjacent pants M_i and M_j: $\gamma \subset \partial M_i, \gamma \subset \partial M_j$, then we should have $f_i(\gamma) = f_j(\gamma)$. Therefore, as we discussed in Section 4, we slice a pair of pants into two patches and compute their boundary-fixed harmonic maps.

As shown in Fig. 3, slicing a pants patch M_i into two hexagons needs 3 curves connecting M_i's boundaries. We simply use the shortest paths to connect each boundaries pair. Then these three paths slice M_i into two patches. The 6 intersection points among these curves and pants' boundaries are mapped to 6 corners of the regular hexagon. To assure the mapping is continuous across the boundary, when corners of M_i have been determined, its adjacent pants' corners on their shared boundary should be consistently determined. The following algorithm computes all corners on a set of pants consistently.

Algorithm 4: Computing corners for a set of pants $M_i, i = 0\ldots n$.

In: A set of pants patches M_i.
Out: All corners on each of M_i.

1. For handle patches $M_0 \ldots M_{G-1}$:
   (1.1) Connect shortest cycles between legs, we get corners P_3, P_4 (the index follows Fig. 3).
   (1.2) Set P_2 as the point on Γ_i^1 farthest from P_3. The

farthest point on Γ_i^2 from P_4 is P_5.
(1.3) Trace the shortest path from P_5 to Γ_i^0; its end point on Γ_i^0 is P_0. The farthest point on Γ_i^0 from P_0 is P_1.

2. Propagate existing corners to adjacent pants: check every M_i's adjacent pants M_j; if corners on M_j's shared boundaries have not been determined, fix them.
3. For each newly propagated M_j, if M_j's corners on Γ_j^0 have not been decided. Use Step (1.3) above to fix them.
4. Stop if all corners have been fixed, otherwise Goto Step 2.

We first go through all handle patches because their corners are freely determined. Then we propagate their corners to the adjacent pants. Each step of the base patch decomposition combines two waists to generate a new pants patch, so the above propagation will not get stuck, and it ends within several iterations with all corners consistently fixed.

Now each pants patch M_i can be sliced into two patches M_i^0 and M_i^1, as Fig. 3 shows. We compute their harmonic maps to the regular hexagon $\mathcal{H}$, $h_j(M_i^j) \rightarrow \mathcal{H}, (j = 0, 1)$ with the following boundary conditions: set each patch's 6 corners' UV coordinates to the regular hexagon's corners; for other boundary points between each pair of corners, interpolate their UV coordinates using the arc-length ratio. Each harmonic map is computed after solving a sparse linear system [22]. On the pants M_i', the same harmonic maps $h_j'(M'^j_i) \rightarrow \mathcal{H}, (j = 0, 1)$ are computed. Then we can immediately get the final composed patch mapping $f(M_i^j) = h'^{-1}_j \circ h_j$ by barycentric point locations. Mapping on boundaries across neighboring patches and pants is consistent. Because each boundary point's image is determined by the corners and corresponding arc length ratios, and both neighboring regions arrive at the same result.

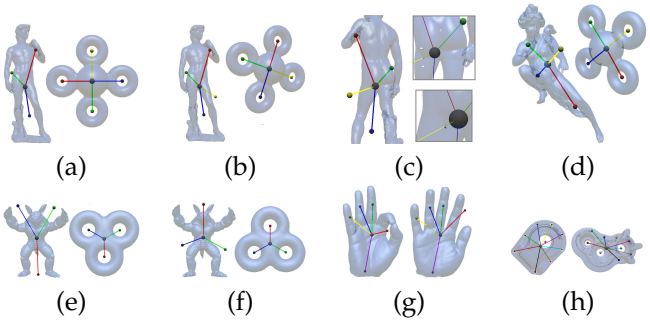


Fig. 8. Handle Matching. (a) Spherical vectors of original surfaces (David and Torus). (b) Handles matched, the target surface rotated. (c) Closed up view. (d)-(h) Some more handle matching for examples used in this paper.

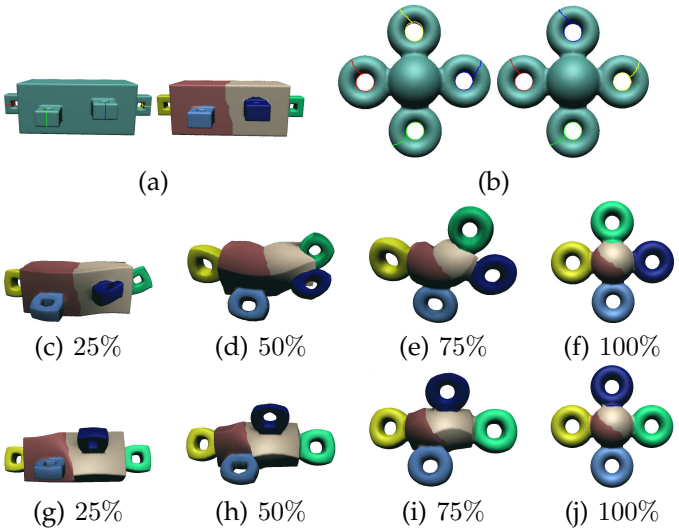


Fig. 9. Surface Mapping with Different Homotopy Types. (a) The source surface and its canonical decomposition. (b) Users can design different homotopy types of mappings by using different indexing of handle/tunnel loops. (c)-(f) First homotopy type: the "green" handle goes up. (g)-(j) Second homotopy type: the "blue" handle goes up.

7 Implementation and Discussion

In this section, we address implementation details of our algorithm, and also discuss the balance between the automation and user involvement within our framework.

7.1 Handle Correspondence

The correspondence of topological handles on two surfaces determines the topological type of the mapping. Different handle correspondences lead to mappings in different homotopic types. Bijective mapping exists in each homotopic type, and generally there is no rigorous way to compare mappings that are topologically different since they could all lead to satisfactory result while carrying different semantics meanings (Fig. 9 shows an example). Our framework can automatically generate a homotopic type (handle correspondence) that is good from geometry aspect, however, we also provide an intuitive interfaces that allows user to enumerate and select different homotopic types of mappings for the semantic purpose.

Handle Matching. The handle correspondence between two surfaces is determined by the indexing of their handle and tunnel loops. Our framework computes default indexing for these loops (as an implicit handle matching) through a rigid alignment.

Once we get all tunnel loops on both surfaces $a_i \subset M$ and $a_i' \subset M'$, we compute a rigid rotation between M and M' to get this alignment. First, on surface M, we compute its mass center M_c as well as mass centers r_i for each tunnel loop a_i, and we get a set of vectors $T = \{t_i\}, t_i = r_i - M_c$ started from the origin. Intuitively, since our pants decomposition treats each surface with handles as a topological "sphere" (base patch) with many "antennas" (handle patches). Each t_i indicates the relative spatial position of a handle to the base patch.

Fig. 8(a) shows these vectors on David and 4-torus models.

A natural optimized matching between T (for M) and T' (for M') is that, under some rigid rotation $\mathbf{R}$ applied on M', pairwise matched vectors between $\mathbf{R}(T')$ and T have their spatial angular distances reached the least square sum. This can be formalized as follows.

Given $T = \{t_0, \cdots, t_{g-1}\}$, and $T' = \{t'_0, \cdots, t'_{g-1}\}$, we seek a $3D$ rigid rotation $\mathbf{R}$ and a bijective mapping matrix $\mathbf{S} : T \to T'$ which defines the bijective correspondence between T and T', optimizing:

$$min \sum_{u=0}^{g-1} \sum_{v=0}^{g-1} \mathbf{S}(u,v)||t_u, \mathbf{R}(t'_v)||$$

s.t.

$$\mathbf{S}u, v \in \{0,1\} \tag{1}$$

$$\sum_{u'=0}^{g-1} \mathbf{S}(u,u') \leq 1, \forall u = 0, \ldots, g-1 \tag{2}$$

$$\sum_{v'=0}^{g-1} \mathbf{S}(v',v) \leq 1, \forall v = 0, \ldots, g-1 \tag{3}$$

where $||\cdot||$ is a measure of the similarity between the two aligned vectors , which the Euclidean distance is used here. Equation (2) and (3) say on each row and collum of the $\{0,1\}$ matrix $\mathbf{S}$, all elements are 0 except the one that equals 1, which guarantee the bijectivity of the correspondence.

Under a specific quaternion, the computation of $\mathbf{S}$ could be considered as a so called "Optimal Assignment Problem" ([24], Chapter 5.5), which can be very efficiently solved using Kuhn-Munkres algorithm in $O(G^3)$ operations where G is the genus number.

To avoid getting stuck on local optimal, we uniformly sample the space of quaternions (using the method presented in [25]), then pick the quaternion with the minimized matching error, and use that corresponding $\mathbf{S}$ as the handle matching. In implementation, we use the software of [25] to uniformly generate 360 quaternion vectors using "layered Sukharev" grid sequence.

This above handle matching preprocessor provides us very well approximated optimal rigid alignment and handle matching. Its advantages are three-folds. First, it automatically and efficiently generates a reasonable homotopy type of the mapping. Second, it directly leads to a generalization on input surfaces that are with different topologies (we will show in the coming section). Third, this pre-alignment can improve the end effects of mapping applications such as morphing (when linear interpolation is used to generate the intermediate shapes as we do in this paper), surface comparison (if the error-matching is conducted on explicit geometric terms such as vertex positions) and so on.

While the default handle matching can be automatically computed, users are allowed to interactively change the homotopy type of the mapping. Fig. 9 and Fig. 13 are two examples in real applications: users may want to enumerate and design different homotopy types of mappings. Within our framework, this is flexibly achieved: users enumerate different mappings by switching indexing numbers[1] of arbitrary two handles.

Base Patch Decomposition Sequence. Sequences used to decompose the two base patches should be consistent as well. The default order is to sequentially remove boundaries from small indices to large ones. Users can also provide their decomposition sequence script when desired. Given a decomposition sequence, the unique pants adjacency relationship can be rigorously deduced. For a surface with G topological handles and B boundaries, we get $2G-2+B$ pairs of pants, and $6G-6+2B$ adjacency relations.

7.2 Topological Surgery and Evolution

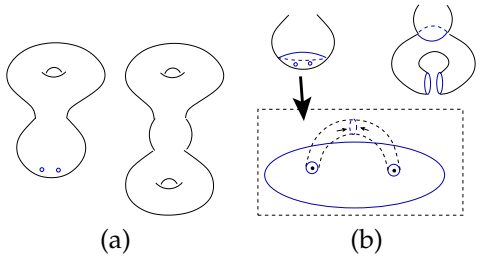


Fig. 10. Consistent Decomposition for Surface with Different Topologies. (a) User specifies a pair of markers to correspond to a handle. (b) Each pair of markers generate a new pants patch, which is matched with a handle patch from the second surface.

Another key advantage of our framework is that it can also flexibly map surfaces with different topologies. Due to the topological difference, "tearing" is inevitable and we call it *topological surgery*. Pants decomposition framework provides a natural scheme to generate such surgery locally on some points, called *surgery points*.

Fig. 10 illustrates an example of topological surgery (Many real examples are presented in the experimental section). When we want to evolve a region to a handle (a), for example, correspond the bottom area of the left torus to the bottom handle on 2-torus. A pair of surgery points are picked and punched, and their one-ring neighbors become boundaries c_1 and c_2. Then similar to the previous process introduced in Algorithm 3, we find a cycle c_3 homotopic to $c_1 \circ c_2$. These three curves bound a pants patch (as shown in (b)), which is then matched to the corresponding handle patch on the second surface.

An automatic way to generate surgery points comes from the handle matching preprocess. Fig. 8(a)-(c) illustrate our idea. Suppose the vector set T, representing handles of surface M (David in (a)), is matched with T' of M' who has higher genus (torus in (a)), then some vectors in T' are not matched. Each such vector t'_i

1. In our system interface, indexing are color-coded and visualized on handle/tunnel loops, while users can simply do the switching.

indicates the spatial position of an extra handle on M', and we directly "transplant" the vector to T. Then we trace the ray from the mass center m_M of M towards the direction of $m_M + t'_i$, and compute this ray's intersected regions on M using the technique of [26]. The farthest intersected region from m_M is a reasonable suggestion to place surgery points for the corresponding handle (David in (b)). Another example is shown in Fig. 8(d). For better semantic control, users can also interactively pick two points on the mesh through the interface and assign to them the index number of the handle that they correspond.

7.3 More User Interaction

As we showed in the previous two sections, automation is achieved in our framework for surface mapping. On the other hand, semantics plays important roles in some applications, in which situation users want to have refined control on the mapping behavior. Designing the special homotopy mapping type is an example. In this section, we show that our framework also integrates more precise user involvement such as feature alignment (Fig. 14(c)) and user-guided segmentation (Fig. 15(g)).

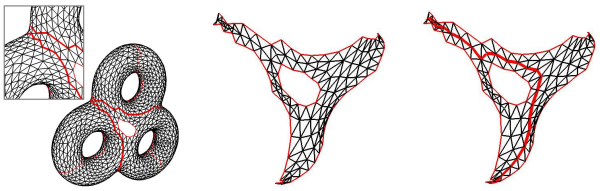
Feature Alignment. An intuitive way to control mapping behavior is through feature alignment. User want to set up corresponding feature points on both surfaces and have them exactly matched under the map. Our framework can naturally enforce correspondence of feature points. It firstly punches a hole on each feature point and makes its 1-ring neighbor a boundary; then the canonical decomposition treats these "feature boundaries" as usual boundaries. As discussed in Section 6, since all boundaries in the pants set are consistently matched, **hard constraint** will be guaranteed. Similarly, the system cuts user-provided feature curves into "feature boundaries", and guarantees their correspondence in the same way.

7.4 User-Guided Segmentation

Canonical pants decomposition must follow the topological consistency, and as discussed above, segmenting paths are determined by geometry (shortest paths) of the surface. However, for any semantic purpose, users are also allowed to adjust these paths by sketching.

Note that decomposition should be topologically correct to assure validity and consistency of segmentation. Therefore, our system takes user's sketches (or called guiding curves) as **soft constraints**, and try to follow the guidance while at the same time guarantees the traced cycles to be homotopic to original ones. This can ease user's operations, eliminate the necessity of user's expertise, and greatly improve our system's robustness.

When the user sketches a set of guiding curves (or curve segments) Γ on the mesh to guide the segmentation (i.e. waists tracing), the system designs a special metric m_M to attract the shortest cycle towards Γ. The following algorithm computes m_M based on vertices' distances from Γ. Intuitively, it "shrinks" Γ's nearby regions, and preserves or expands distant regions, so that the shortest paths are attracted towards Γ:



(a) (b) (c)

Fig. 11. Adaptive Edge-Split. (a) Two waists (thick red curves) are close to each other. No path can cross through the upper left region. (b) Edge-split on base patch before tracing. (c) Shortest cycles pass through successfully.

Algorithm 5: Compute the Guiding Metric.

In: Surface mesh M, guiding curves set Γ, threshold D, parameter α.

Out: Guiding metric m_M defined on each edge $e \in E(M)$.

1. Set an initial metric $m_M(e) = 1$ for all edges. Set all vertices on guiding curves as starting points.
2. Perform the Dijkstra algorithm, using $m_M(e)$ as the edge length of the graph. We get the hop distance $d(v)$ from all vertices to the guiding curves.
3. Set the weight function:
$$w(v_i) = \begin{cases} 1, & d(v_i) > D; \\ (\frac{d(v_i)}{D})^\alpha, & o/w. \end{cases},$$
where D is a hop distance threshold, α is a parameter to control the strength of the attraction.
4. Set
$$m_M(e_{ij}) = (\frac{w(v_i) + w(v_j)}{2}) * |e_{ij}|,$$
where $|e_{ij}|$ is the original edge length of e_{ij}.

7.5 Robust Shortest Path Tracing

Shortest cycles tracing generates the "pants" topology of each patch. To avoid degeneracy, it should prevent shortest cycles from intersecting each other or boundaries. We slightly modify the Dijkstra algorithm to prevent shortest paths from reaching boundaries (or user-specified curves). In the Dijkstra algorithm, when a vertex v is visited, we enqueue it and relax its neighbors ([27], page 595). Now if v is on boundaries (or on some specific curves we want to circumvent), we do not enqueue v nor update its neighbors. The new algorithm prevents any shortest cycle from intersecting boundaries.

The Dijkstra algorithm always succeed to trace a shortest path for an arbitrary vertex on a connected mesh, and our modified algorithm only fails when two boundaries are too close to each other. An example is illustrated in Fig. 11. (a) shows a three-hole torus with a boundary, and its waists w_0 and w_1 (thick red curves) are close to each other, therefore the upper left region is error-prone: there are some edges spanning these two boundaries. So although topologically a path should be able to go through this region between two boundaries without any intersections, a discrete path will inevitably intersect

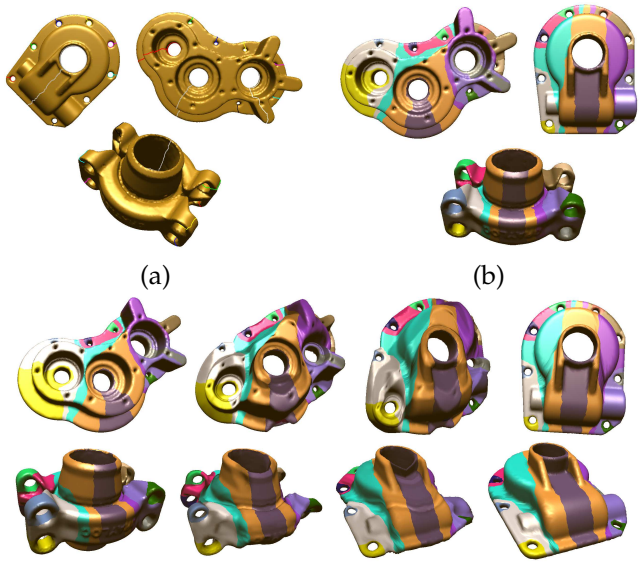


Fig. 12. Mapping Genus-9 Mechanical Parts. (a) shows genus-9 surface models and color-codes their homotopy group bases. (b) illustrates the corresponding canonical decomposition result. The next two rows visualize mappings through morphing sequences.

boundaries under the current connectivity. We call this kind of spanning edges *dangerous edges*. We perform "edge-split" on all *dangerous edges* before computing shortest cycles/paths, as shown in (b). This robustly guarantees the success of our path tracing (c).

8 Experimental Results

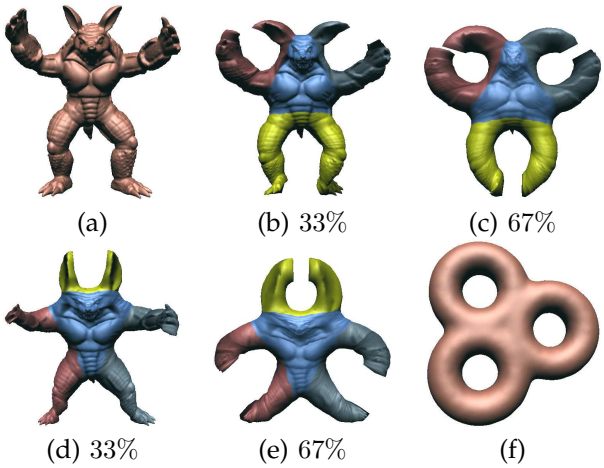


Fig. 13. Different Homotopy Types of Mapping from Armadillo to 3-Torus. (a) Armadillo Model. (b),(c) Mapping/Morphing of one homotopy type. (d) and (e) Mapping/Morphing of another homotopy type. (f) 3-Torus Model.

This section experimentally demonstrates our mapping framework. In most examples, the mapping is visualized using linear-interpolated morphing sequence, all of which can be found in our accompanying video.

Automatic Mapping Genus-9 Mechanical Parts. Consistent pants decomposition is automatically performed on three genus-9 mechanical parts. As shown in Fig. 12, no user involvement is necessary for generating mapping between these models with very complicated topology and geometry. This demonstrates great automation of our framework and its availability in registering complicated objects.

Surface Mappings with Different Homotopy Types. Fig. 9 and Fig. 13 illustrate the completeness of our mapping framework. As previously mentioned, different homotopy classes can be enumerated by users, by simply switching the indexing of the homotopy group basis (as shown in Fig. 9(b)). In Fig. 9, the morphing sequences from the source surface (a) to the target surface based on different mappings are illustrated in the next two rows. They are both geometrically good, and it is up to the user to select which one they really want. Our framework provides a rigorous way for users to decide an arbitrary homotopy type of the mapping. Fig. 13 shows another example. The mapping and morphing illustrated in (b) and (c) follow the homotopy type of Fig. 8(e); while morphing of (d) and (e) follow the homotopy type of Fig. 8(f).

Deforming Hands: "Five" to "Okay". In this example, we map a human hand (Fig. 14(a) (left)) to another hand (right). It demonstrates that our framework integrates feature alignments for semantic mapping purpose. The source hand is an open genus-0 surface and the target hand is genus-1 with a boundary (red curves are its handle/tunnel loops). To make the topological evolution, at least a pair of surgery points is required on the first hand. Here we can manually set them on tips of the indexing finger and thumb (red points in (b)). Now without any further feature points, each hand is decomposed into one pants patch by default. This leads to a global one-patch mapping that follows no shape semantics, visualized by morphing in (d): three fingers shrink while three new fingers grow from somewhere else, because this mapping does not match fingers to fingers. User can control the mapping behavior by feature alignment. (e) shows the refined decomposition on both hands based on the feature setting of (c). The new resultant mapping guarantees the finger correspondence and therefore generates a more semantically natural morphing as illustrated in (f) and (g).

Genus-4 Greek Model to Genus-3 David Model. Fig. 15 illustrates the mapping between Greek and David models and shows the user interaction on setting guiding curves. (a) shows the original surfaces and their homotopy group bases. According to the handle matching of (b), a pair of surgery points, as shown in (c), is generated corresponding to the small handle in the lower right part of the Greek sculpture. In (d), we place two feature points on the base patch of each model, to semantically guarantee correspondence between "head" regions. We show the canonical decomposition result in (e). Now semantically, we do not like the segmentation around the right hand (blue patch) of the Greek because the shortest cycle goes through his wrist. The segmentation of the

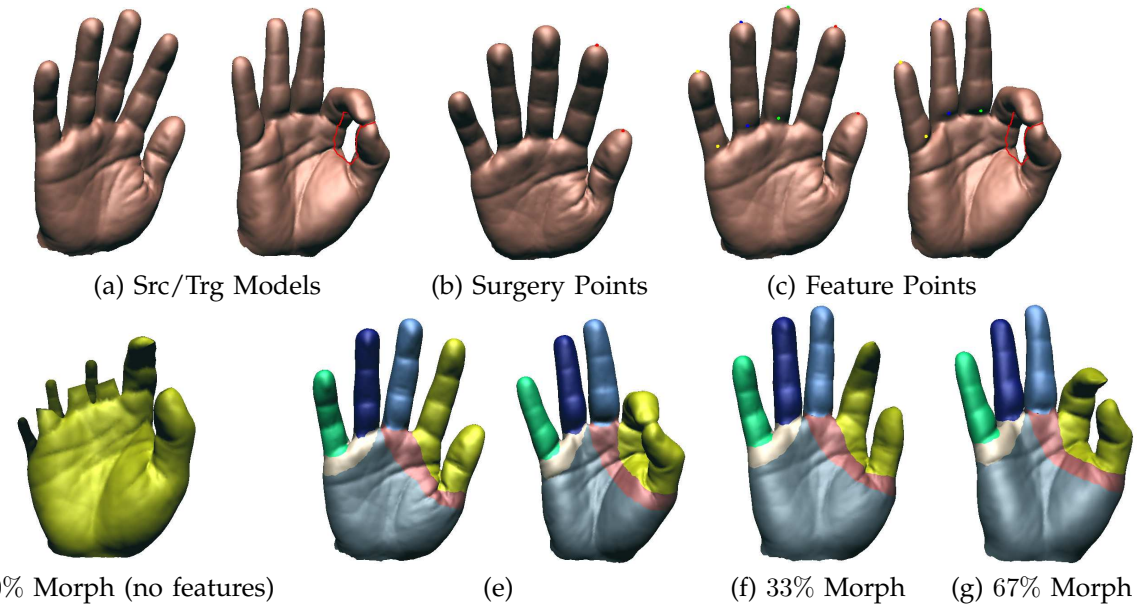


Fig. 14. Mapping Hands: "Five" to "Okay". (a) Source and target surfaces. (b). Two surgery points are the least requirements due to the topological difference. (c) Users define more feature points for semantics purpose. (d) Without feature points in (c), 3 fingers are not matched, the morphing is ugly. (e) The refined decomposition results (with feature points). (f) and (g) show the newly generated morphing.

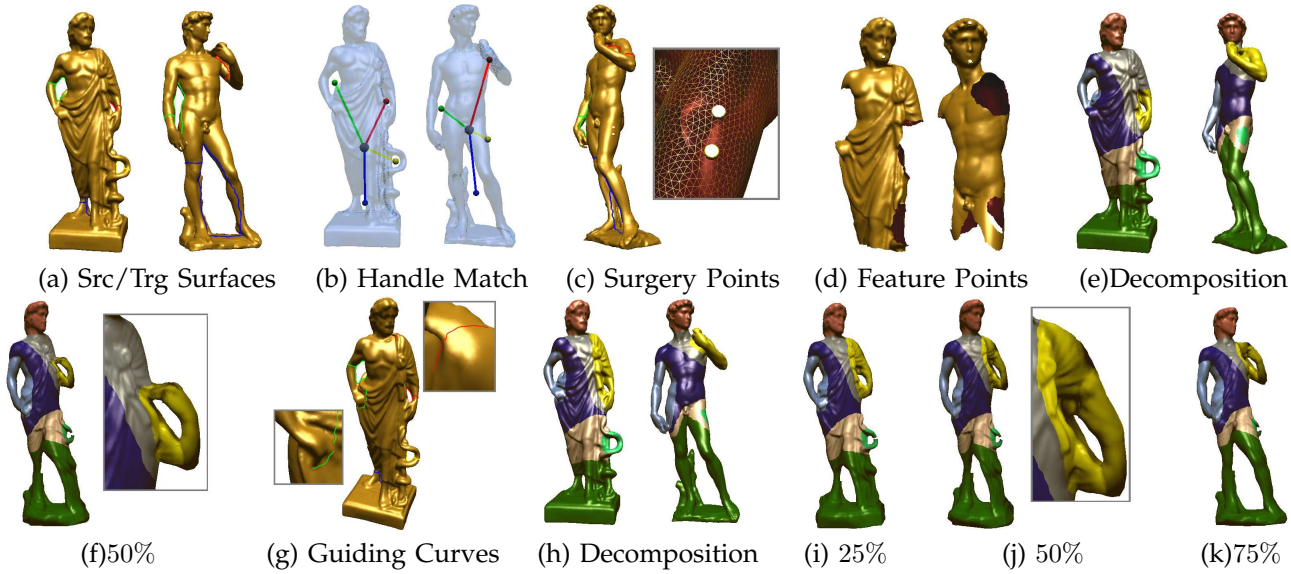
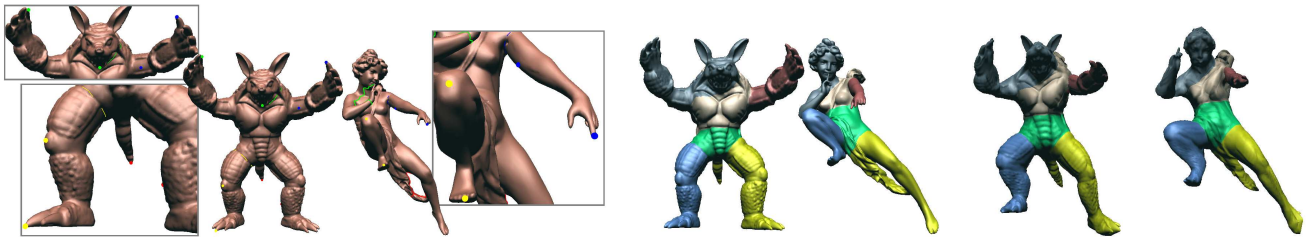


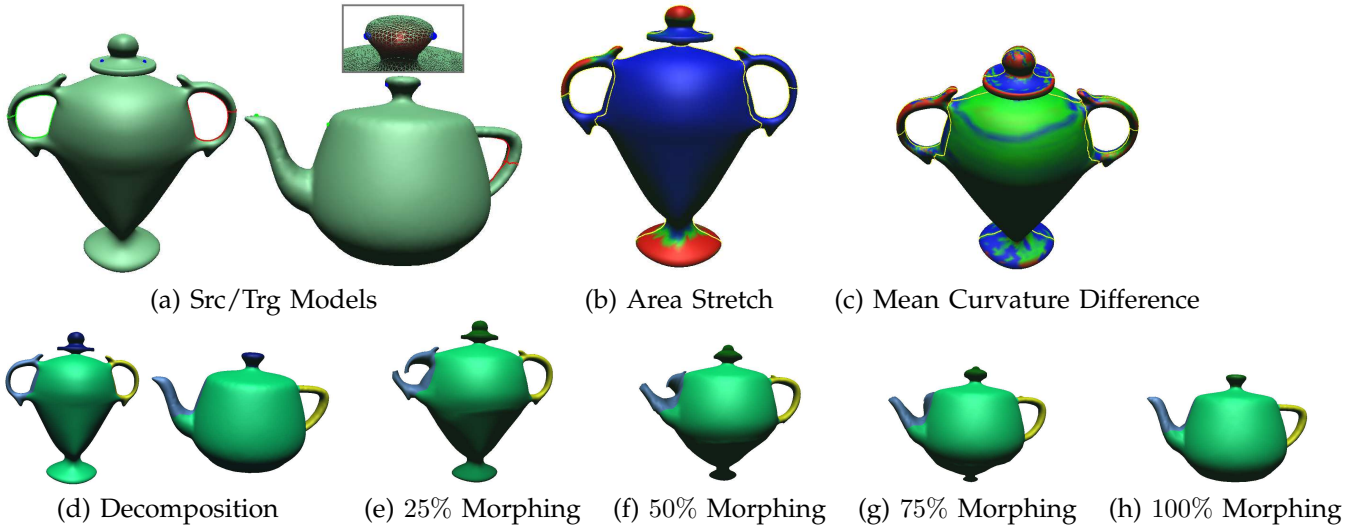
Fig. 15. Mapping Greek Sculpture to David. (a) Two surfaces and their homotopy group bases. (b) Two surgery points (matched with the lower right green handle on the Greek). (c) Base patches of both models, and two feature points to assure correspondence on head regions. (d) The decomposition result without further user involvements. (e) Geometrically optimal decomposition may have poor semantics effect (yellow regions). (f) Users sketch some guiding curves. (g) The new decomposition result with guided segmentation. (h)-(j) A more visually natural morphing sequence.

Fig. 16. Mapping two Dragons. Feature/surgery points are placed on both dragons (red and green markers on the head and legs). The morphing sequence is generated.



(a) (b) (c) 33% (d) 67%

Fig. 17. Mapping Armadillo to Angel. (a) The initial setting. (b) Decomposition Result. (c),(d) Morph Armadillo to Angel.



(a) Src/Trg Models (b) Area Stretch (c) Mean Curvature Difference

(d) Decomposition (e) 25% Morphing (f) 50% Morphing (g) 75% Morphing (h) 100% Morphing

Fig. 18. Vase vs. Teapot. (a) Surfaces with handle/tunnel loops and surgery/feature points. The matching's area stretching (b) and mean curvature difference (c) are color-coded. (d) Pants Decomposition Result. (e)-(h) Morphing.

left arm (yellow patch) is even worse; it cuts through the elbow. A resultant morphing that maps the forearm of the Greek to the whole arm of David is shown in (f). Users can easily remedy this in our framework and get a refined result by simply sketching two short guiding curve segments on the Greek model, as shown in (g). The new decomposition result is shown in (h), where we get a morph with better visual effects as shown in (i)-(k).

Morphing Dragons. In Fig. 16, we perform decomposition on two dragons. Several surgery points and feature points are used at the same time to guide the mapping, as depicted on the source and target models. The morphing sequence is shown to demonstrate the mapping effect.

Shape Matching and Error Analysis. With one-to-one correspondence between two matched surfaces, we can measure point-by-point the shape difference using some geometric properties, and color-code the error distribution, which is potentially useful for shape comparison and visual analysis.

Fig. 18 illustrates our mapping from a genus-2 vase model to a genus-1 teapot model. (a) shows the models and their handle/tunnel loops; and user-provided surgery/feature points are also depicted. The decomposition results are shown in (d). (e)-(h) show the morphing sequence generated by our mapping.

In (b) and (c), we color-code the shape matching error distribution. We use two terms, one is the *area stretching ratio* while the other is mean curvature difference. In (b), we compute total area of one-ring triangles around each point on the vase model; when the vase is mapped onto the teapot, we also compute each point's corresponding one-ring area. The ratio of these two areas represents the *stretching* of the mapping, which is color-coded on the surface: red represents the maximum while blue represents the minimum. From this figure, we can see that the cap, the left handle, the tips of handles, and the bottom of the vase have larger stretching values, because these regions shrink to a relatively small area on the teapot model. In (c), we color-code the mean curvature difference on every point: the regions with large curvature difference (for example, left handle, the rim of the cap) are red. Integration of these two terms over the whole surface has been proved ([28]) to provide an intrinsic energy that measures the shape difference. Therefore, our surface mapping/registration can be used for shape comparison and shape analysis.

The complexity of our algorithm is theoretically bounded by $O(n^{3/2}\log n)$ from the the Dijkstra algorithm in the "waists" tracing step, where the square root of n comes from the length of the shortest path between two

Model	Topology	Vertex/Pants #	Time
Hand-1	$G=0, B=1$	19832/7	24.5s
Hand-2	$G=1, B=1$	21055/7	26.1s
Teapot	$G=1, B=0$	22012/4	27.0s
Vase	$G=2, B=0$	10014/4	10.1s
4-Torus	$G=4, B=0$	7994/6	6.3s
David	$G=3, B=0$	26138/8	140.3s
Greek	$G=4, B=0$	43177/8	380.5s
Asian Dragon	$G=0, B=0$	26562/10	110.1s
Casting	$G=9, B=0$	34116/16	423.4s

handles, which is of the complexity of $O(\sqrt{n})$. Runtime performance of our algorithm on most real examples presented here is given in the following table.

9 Conclusion and Future Work

We have developed a consistent pants decomposition framework for mapping surfaces with arbitrary topology. The consistent generation of sets of pants is a key component to ensure the subsequent high quality surface mapping. Our novel mapping framework has been demonstrated to be efficient, robust, and powerful on examples with arbitrary types of surfaces. Also, the mapping framework simultaneously provides great automation and accommodates intuitive user control. Therefore, our new modeling framework can serve as an enabling tool for many visual computing tasks.

Besides surface mapping, we believe our pants decomposition framework has many other potential applications. First, pants decomposition provides a piecewise representation for any given surface. When we have the semantically meaningful patch segmented from the original surface, we can easily perform the "cut-and-paste" operation from a "part" database ([29]) to produce more meaningful shapes from examples. Since all our segmented patches are pants-like shapes, we could streamline many modeling and simulation tasks. Also, pants decomposition can be extended to a consistent segmentation of volumetric data. Compared with directly computing harmonic maps between volumetric shapes with complicated topology and geometry [30], this decomposition should make the process more robust and general, and it will also provide more semantics control. Our generated morphing sequence has the connectivity of the source surface, whose limitation is that the sampling may not be good for the target surfaces. A postprocess adaptive remeshing technique [1] could be applied to improve the quality of the final mesh, however, a dynamic online remeshing method will be an interesting research direction for the morphing application.

References

[1] X. Li, Y. Bao, X. Guo, M. Jin, X. Gu, and H. Qin, "Globally optimal surface mapping for surfaces with arbitrary topology," *IEEE Trans. on Visualization and Computer Graphics*, vol. 14, no. 4, pp. 805–819, 2008.

[2] D. DeCarlo and J. Gallier, "Topological evolution of surfaces," in *Proc. Graphics interface*, 1996, pp. 194–203.

[3] T. Kanai, H. Suzuki, and F. Kimura, "Three-dimensional geometric metamorphosis based on harmonic maps," *The Visual Comput.*, vol. 14, no. 4, pp. 166–176, 1998.

[4] A. Asirvatham, E. Praun, and H. Hoppe, "Consistent spherical parameterization," in *International Conference on Computational Science (2)*, 2005, pp. 265–272.

[5] V. Kraevoy and A. Sheffer, "Cross-parameterization and compatible remeshing of 3D models," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 861–869, 2004.

[6] J. Schreiner, A. Asirvatham, E. Praun, and H. Hoppe, "Inter-surface mapping," *SIGGRAPH.*, vol. 23, no. 3, pp. 870–877, 2004.

[7] A. Gregory, A. State, M. Lin, D. Manocha, and M. Livingston, "Feature-based surface decomposition for correspondence and morphing between polyhedra," in *Proc. of the Computer Animation*, 1998, pp. 64–71.

[8] M. S. Floater and K. Hormann, "Surface parameterization: a tutorial and survey," in *Advances in Multiresolution for Geometric Modelling*, ser. Mathematics and Visualization, 2005, pp. 157–186.

[9] A. Sheffer, E. Praun, and K. Rose, "Mesh parameterization methods and their applications," *Found. Trends. Comput. Graph. Vis.*, vol. 2, no. 2, pp. 105–171, 2006.

[10] J. Kent, W. Carlson, and R. Parent, "Shape transformation for polyhedral objects," in *SIGGRAPH '92*. ACM Press, 1992, pp. 47–54.

[11] M. Alexa, "Merging polyhedral shapes with scattered features," in *Proc. of the Intl Conf. on Shape Modeling and Applications*, 1999, pp. 202–210.

[12] M. Zöckler, D. Stalling, and H.-C. Hege, "Fast and intuitive generation of geometric shape transitions," *Visual Computer*, vol. 16, no. 5, pp. 241–253, 2000.

[13] F. Lazarus and A. Verroust, "Three-dimensional metamorphosis: a survey," *The Visual Computer*, vol. 14, no. 8/9, pp. 373–389, 1998.

[14] A. Lee, D. Dobkin, W. Sweldens, and P. Schröder, "Multiresolution mesh morphing," in *Proc. SIGGRAPH*, 1999, pp. 343–350.

[15] E. Praun, W. Sweldens, and P. Schröder, "Consistent mesh parameterizations," in *Proc. SIGGRAPH*, 2001, pp. 179–184.

[16] C. Carner, M. Jin, X. Gu, and H. Qin, "Topology-driven surface mappings with robust feature alignment." in *IEEE Visualization*, 2005, pp. 543–550.

[17] X. Gu and S.-T. Yau, "Global conformal surface parameterization," in *Proc. Symp. Geometry Processing*, 2003, pp. 127–137.

[18] J. Erickson and K. Whittlesey, "Greedy optimal homotopy and homology generators," in *ACM-SIAM Symp. on Discrete algorithms*, 2005, pp. 1038–1046.

[19] T. K. Dey, K. Li, and J. Sun, "On computing handle and tunnel loops," in *International Conf. on Cyberworlds*, 2007, pp. 357–366.

[20] A. Hatcher, P. Lochak, and L. Schneps, "On the Teichmüller tower of mapping class groups," *J. Reine Angew. Math.*, vol. 521, pp. 1–24, 2000.

[21] E. Verdière and F. Lazarus, "Optimal pants decompositions and shortest homotopic cycles on an orientable surface," *J. ACM*, vol. 54, no. 4, p. 18, 2007.

[22] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution analysis of arbitrary meshes." in *SIGGRAPH*, 1995, pp. 173–182.

[23] M. S. Floater, "Mean value coordinates." *Computer Aided Geometric Design*, vol. 20, no. 1, pp. 19–27, 2003.

[24] J. Bondy and U. Murty, *Graph Theory with Applications*. North Holland, 1982.

[25] A. Yershova and S. Lavalle, "Deterministic sampling methods for spheres and SO(3)," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2004, pp. 3974–3980.

[26] D. Badouel, "An efficient ray-polygon intersection," pp. 390–393, 1990.

[27] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*. The MIT Press, 2001.

[28] X. Gu, Y. Wang, T. Chan, P. Thompson, and S. T. Yau, "Genus zero surface conformal mapping and its application to brain surface mapping," *IEEE Trans. Med. Imaging*, vol. 23, no. 8, pp. 949–958, 2004.

[29] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin, "Modeling by example," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 652–663, 2004.

[30] X. Li, X. Guo, H. Wang, Y. He, X. Gu, and H. Qin, "Harmonic volumetric mapping for solid modeling applications," in *Proc. ACM symp. on Solid and physical modeling*, 2007, pp. 109–120.