Surface Matching Using Consistent Pants Decomposition

Xin Li* Stony Brook University (SUNY)
Xianfeng Gu† Stony Brook University (SUNY)
Hong Qin ‡ Stony Brook University (SUNY)

Abstract

Surface matching is fundamental to shape computing and various downstream applications. This paper develops a powerful *pants decomposition* framework for computing maps between surfaces with arbitrary topologies. We first conduct pants decomposition on both surfaces to segment them into consistent sets of *pants* patches (here a pants patch is intuitively defined as a genus-zero surface with three boundaries). Then we compose global mapping between two surfaces by harmonic maps of corresponding patches. This framework has several key advantages over other state-of-the-art techniques. First, the surface decomposition is automatic and general. It can automatically construct mappings for surfaces with same but complicated topology, and the result is guaranteed to be one-to-one continuous. Second, the mapping framework is very flexible and powerful. Not only topology and geometry, but also the semantics can be easily integrated into this framework with a little user involvement. Specifically, it provides an easy and intuitive human-computer interaction mechanism so that mapping between surfaces with different topologies, or with additional point/curve constraints, can be properly obtained within our framework. Compared with previous user-guided, piecewise surface mapping techniques, our new method is more intuitive, less labor-intensive, 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.

CR Categories: I.3.5 [Computing Methodologies]: Computer Graphics—Computational Geometry and Object ModelingGeometric algorithms, languages and systems;

Keywords: Surface Matching, Shape Analysis and Synthesis

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 registration 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 matching 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 new 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.

It may be noted that, although this topic is important and has been widely studied, current state-of-the-art surface mapping techniques are far from adequate and perfect. A more desirable and powerful surface mapping method is needed 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. 10(a)).

Many existing surface mapping techniques primarily focus on genus-zero surfaces, most recent works start to aim at general surfaces, and 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 depends 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 real applications where the semantics plays a critical role, such as morphing (re-

*e-mail: xinli@cs.sunysb.edu
†e-mail: gu@cs.sunysb.edu
‡e-mail:qin@cs.sunysb.edu

quiring feature points matching), automatic methods based on pure topology and geometry inevitably 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 [Kraevoy and Sheffer 2004] or need user's great efforts to design the base mesh as a good starting point [DeCarlo and Gallier 1996], [Gregory et al. 1998]. In principle, a good mapping framework should provide an intuitive and easy-to-use human computer interaction.

Fourth, it is also important to emphasize **rigorousness**. 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. 14). 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 rigorousness 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.

1. Our framework efficiently handles surfaces with arbitrary topology, with or without boundaries. It also handles surface mapping with topology changes.

2. Our framework has great automation. Our pants decomposition can automatically compute consistent segmentation on a set of surfaces with same but complicated topology. This framework can proceed without any user intervention, and therefore provide canonical decomposition for automatic matching among a large number of acquired or synthetic datasets.

3. When user interaction is necessary for any semantics reasons, our framework coherently aligns constraint points or curves to enforce constraints, and provides users a simple and intuitive mechanism to control the mapping behavior.

4. In practice, our framework generates and enumerates any different homotopy types of mappings. It shows not only the flexibility but also the rigorousness and completeness of our mapping framework from the mathematical point of view.

5. Our algorithm is simple and efficient in practice. As we will elaborate later, the technical core of the decomposition 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. We demonstrate experimental results with various applications in Section 7 and conclude the paper in Section 8.

2 Related Work

Surface mapping is a fundamental problem in computer graphics/modeling fields. A thorough survey is beyond the scope of this work. In the following, we only brief the most related work.

Earlier work on computing inter-surface mappings is mostly motivated by the need of shape blending. A natural and intuitive approach is to establish the shape correspondence through some canonical domains.

For genus-zero meshes, the sphere (for closed surfaces) and the plane (for open surfaces) are two widely-used intermediate domains. [Kent et al. 1992] computed maps between star-shaped surfaces by first mapping them onto spheres, and then merging them by clipping one sphere to the other. Later, [Kanai et al. 1998] used harmonic map to build correspondence from surfaces to the unit disk domain, so that not only the star-shaped surfaces, but all genus-zero surfaces can be mapped easily. However, it only allows one constraint point from users. [Alexa 1999] proposed to match multiple feature points between genus-0 surfaces. This work wrapped two surfaces onto a unit sphere by minimizing a distance function, and feature points on the surface are aligned and the resultant embedding is used for the surface mapping. Its drawback is that no bijectivity is guaranteed and hard constraints may not be fully enforced. More recently, [Asirvatham et al. 2005] used 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 complicated topology, common canonical domains such as disks and spheres become unavailable. Directly solving intra-surface mapping usually fails. Most techniques use another strategy: first segment surfaces with complicated topology into consistent sets of sub-regions; then compose or refine the global result from the sub-region maps. [DeCarlo and Gallier 1996] introduced a flexible surface mapping framework based on user-specified base meshes. When the base meshes are carefully designed, mapping between surfaces with different topology can be computed. However, deeper domain knowledge in topological surgery is required to manually design consistent base meshes; and when the surface are with high genus, the design can be quite complicated. Only examples up to genus-2 are provided in this work. [Gregory et al. 1998] and [Zöckler et al. 2000] also used the base mesh approach. When the consistent "base mesh" have been manually designed. Harmonic or barycentric maps are used to match these sub-regions accordingly. More early surface mapping work for morphing applications can be found in the survey [Lazarus and Verroust 1998].

Recent work has been trying to seek more automatic methods for consistent base mesh generation. [Lee et al. 1999] used their MAPS algorithm to hierarchically map fine meshes onto a common base mesh. [Praun et al. 2001] introduced a graph tracing algorithm to transfer the coarse base mesh from one surface to another with the same topology. [Kraevoy and Sheffer 2004] used another way to trace base meshes consistently on different surfaces. However, generating base meshes for high genus surfaces still needs many feature points from users. For example, to proceed the base mesh tracing algorithm, at least four markers are required on each topological handle. [Schreiner et al. 2004] 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 created a trivial map onto the base mesh, and iteratively refined the map back to the original surfaces.

Base mesh construction (consistent segmentation) could consume a large amount of human labor. Therefore, a recent research direction is the automatic generation of surface maps. This automation becomes very challenging for surfaces with complicated topology, where far less work has been explored. Furthermore, when given

surfaces with different topologies are present, it is even more difficult. Manual base mesh design [DeCarlo and Gallier 1996] requires greater effort and stronger expertise from the user. This motivates us to seek an automatic method 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 [Carner et al. 2005] and [Li et al. 2008], canonical homology bases [Gu and Yau 2003] and systems of loops [Erickson and Whittlesey 2005] were used to study this issue and build mappings of different homotopy types. [Dey et al. 2007] defined terms *handle loops* and *tunnel loops*, which provide an intuitive way to study the topological handles on surfaces. The computation algorithm is also presented in [Dey et al. 2007]. 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 [Hatcher et al. 2000]. Work has been done to investigate the optimal segmentation of a given surface into pants [Verdière and Lazarus 2007]. For surface mapping purpose, instead of decomposing just one surface, we need to compute consistent decomposition on several surfaces, or find canonical decomposition for 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 paper.

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 *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

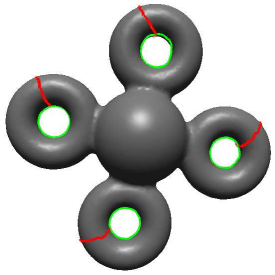
$$(p \circ q)(t) = \begin{cases} p(2t), & t \leq 1/2; \\ q(2t-1), & t \geq 1/2. \end{cases}$$

When we say two paths are *homotopic*, it means one path can continuously evolve to the other through a family of paths on the surface. Rigorously, a homotopy between paths p and q is a continuous map $h : [0,1] \times [0,1] \to M$ s.t. $h(0,\cdot) = p$, $h(1,\cdot) = q$, $h(\cdot,0) = a$, $h(1,\cdot) = b$, 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 split M into pairs of pants. Every compact orientable surface, except the sphere, disk, cylinder, and torus, admits pants decomposition. If M is with genus G and has B boundaries, a pants decomposition is made of $3G + B - 3$ cycles [Hatcher et al. 2000]. 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 apart 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 surface patch a *pants patch* in the following 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 ([Dey et al. 2007]). 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 as well as their automatic computation algorithm can be found in [Dey et al. 2007]. 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 $\pi_1(\mathbb{M})$. Our algorithm takes surfaces, their handle and tunnel loops as input.

4 Overview of Key Ideas

The core of our mapping framework is consistent pants decomposition. Given an arbitrary genus-G surface with B boundaries, pants decomposition provides a canonical segmentation, partitioning this surface into $2G + B - 2$ pants patches (see the following for more details). Given a set of arbitrary surfaces with same topology, our pants decomposition scheme can automatically segment all of them into consistent sets of patches with "pants" topology.

Decomposing a Surface. The first step of our surface mapping is pants decomposition. It segments the given surface into a set of pants. This decomposition can process canonically in an automatic way, i.e. once the indexed handle and tunnel loops $(a_i, b_i, 0 \leq i < G)$ are provided in a preprocessing stage, then the decomposition is unique and we will obtain an ordered set of pants.

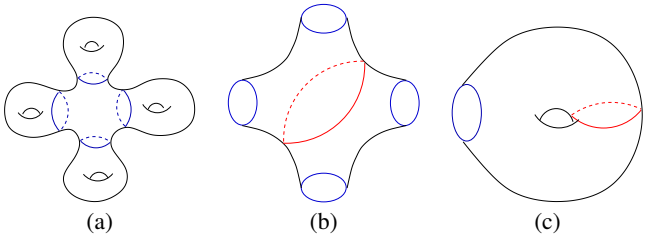


Figure 1: Pants Decomposition Pipeline. (a). Find and remove "waists" of handles. (b),(c). Decompose the base patch and handle patches.

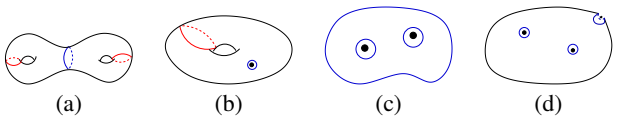


Figure 2: Pants Decomposition on Surfaces with Simple Topology. (a). A genus-2 surface has 2 handle patches, no base patch. (b). A closed genus-1 surface ($\chi = 0$) needs one punch. (c). A topological disk ($\chi = 1$) needs two punches. (d). A topological sphere ($\chi = 2$) needs three punches.

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 our pipeline. First, we remove G handle patches apart 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' 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 ((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 easier 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 mesh 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 mesh (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.

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 [Eck et al. 1995], mean value coordinate map [Floater 2003] 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 pair of pants is decomposed into two topological hexagons, and each of them is harmonically mapped to a regular hexagon. The pants mapping is then composed through these hexagonal domains.

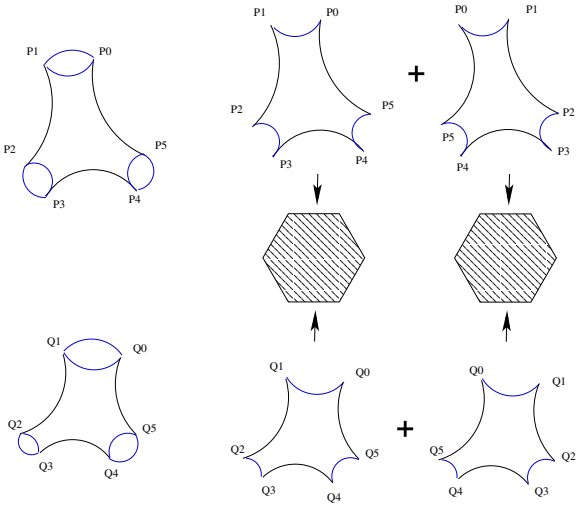


Figure 3: Mapping two Pants Patches. Each pair of pants is decomposed into two topological hexagon patches. Harmonic maps from these patches to regular hexagons are used to compose the pants mapping.

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. To allow topology changes in mapping, surgery points are introduced in our decomposition framework (Section 5.4). Users can control the mapping by setting feature points or curves, we describe the related issues in Section 5.5 and Section 5.6. Our decomposition process is very robust, as addressed in Section 5.8.

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 and slice M along it to separate a handle patch (which becomes a pair of pants M_i later) from M. We denote such shortest cycle as w_i, indicating it is the "waist" of M_i.

Suppose the handle loop and tunnel loop of the current handle are denoted as a_i and b_i, then 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, we illustrate it using the following example.

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 apart along their intersection point, and get the resultant green boundary in (d): $c_i = P_i^{1,1} \rightarrow P_i^{1,-1} \rightarrow P_i^{-1,-1} \rightarrow P_i^{-1,1} \rightarrow P_i^{1,1}$.

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, iteratively, we shrink each c_i to its shortest homotopic cycle w_i. It is computed through the following algorithm 1:

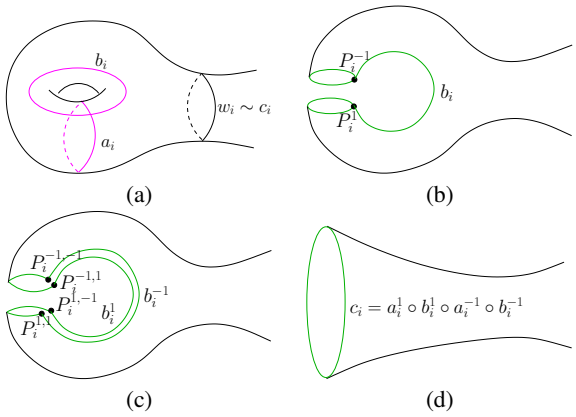


Figure 4: Computing c_i of the handle i. (a). w_i is the waist, but we need to compute c_i first. (b). Slice a_i apart, get P_i^1 and P_i^{-1}. (c). Slice b_i apart, P_i^1 and P_i^{-1} split to $P_i^{1,1}$, $P_i^{1,-1}$, $P_i^{-1,1}$ and $P_i^{-1,-1}$ separately. (d). The newly generated boundary is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

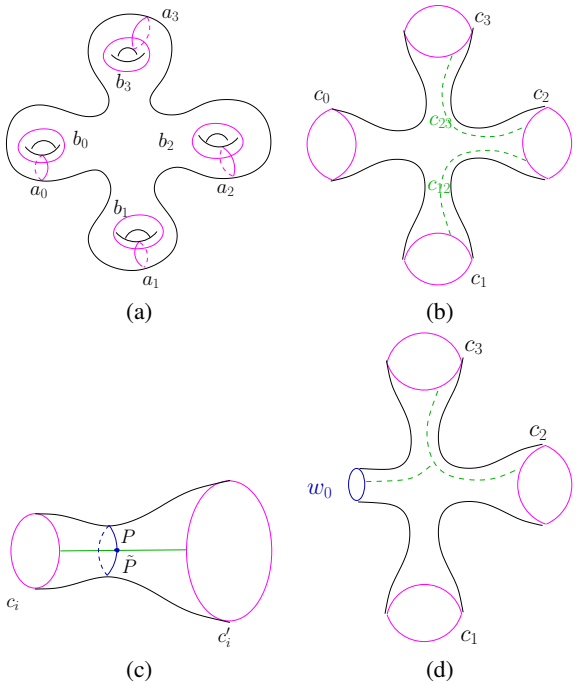


Figure 5: Computing the "Waist". (a). Slice M apart along its handle/tunnel loops, get boundaries c_i. (b) Connect all other boundaries $j, (j \neq i)$ to a large boundary c_i', and get a topological cylinder. (c) Slice apart γ (green) connecting c_i and c_i'; get a topological "trapezoid"; compute w_i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

Algorithm 1 *Shrink c_i to w_i.*

1. *Connect all existing boundaries except c_i together using shortest paths (dashed green curves in Fig. 5(b)).*
2. *Slice these paths apart, 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)).*

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 $M_0 \dots M_{G-1}$. Their waists w_i are 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 in the following sections.

5.2 Decomposing the Base Patch

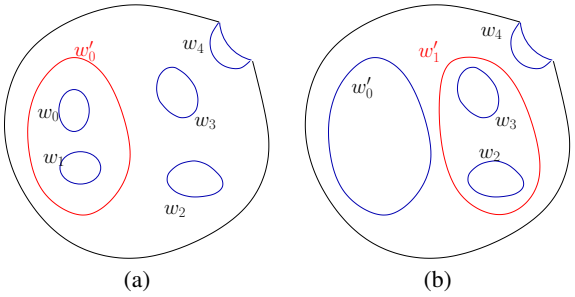


Figure 6: Decomposing the Base Patch. (a) Slice apart w_0', get a new pair of pants. Boundary number decreases 1. (b) Set w_0' as a new boundary, go on to compute w_1'.

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

Algorithm 2 *Base Patch Decomposition.*

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 $w_i \circ w_j$.*

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 4. in Algorithm 1.*

5.3 Decomposing Handles

After we find each waist w_i in pipeline Step one, we remove the handle patches M_i from M, each of which is a genus-1 surface

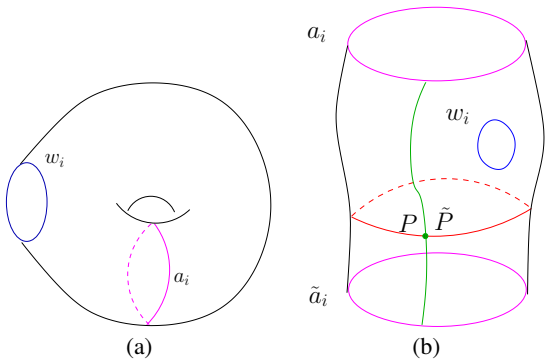


Figure 7: Decomposing a Handle Patch. (a) Slice a_i apart. (b) Slice apart of the green curve that connects two outer boundaries. Then find the shortest path (red) connecting corresponding point pairs on the green curve.

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 apart 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 apart 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 apart this shortest cycle (red cycle in (b)).

5.4 Topological Surgery and Evolution

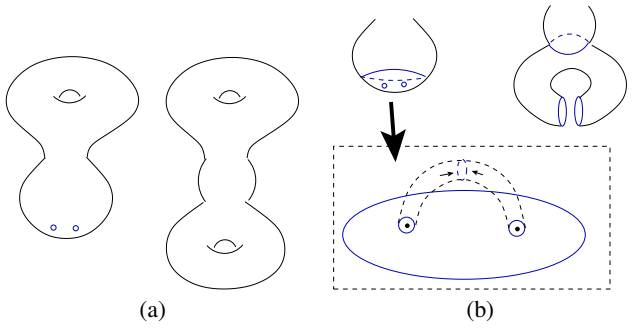


Figure 8: Consistent Decompositions for Surface with Different Topologies. (a) User specifies a pair of markers to correspond to a handle. (b) Each pair of markers generated a new pants patch, which is matched to a handle patch from the second surface.

The pipeline introduced in the above sections provides a canonical (therefore consistent) decomposition for surfaces with the same topology. When the given surfaces are with different topologies, our framework can also flexibly handle it. The user only needs to provide some marker points (denoted as *surgery points*).

As shown in Fig. 8(a), when we want to evolve a region to a handle. For example, we want to match the bottom area of the left torus to the bottom handle on 2-torus. We easily pick a pair of points, and punch two holes there. 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 matched with the corresponding handle patch on the second surface. Many real examples using surgery points to handle topology change in surface mapping will be presented in the experimental section later.

5.5 Matching Feature Points

When the semantics need to be considered during surface mapping, users usually set up corresponding feature points or curves on both surfaces and require them to be matched exactly. To enforce the feature points correspondence as hard constraints, on each feature point, we also punch a hole, and make its 1-ring neighbor a boundary. During the decomposition, we treat these boundaries generated from feature points as usual boundaries. As we will discuss in Section 6, since all the boundaries in the pants set are consistently matched, **hard constraints** are guaranteed. We can easily cut feature curves into boundaries, and guarantee their hard constraints in the same way.

5.6 User-Guided Segmentation

Our pants decomposition follows a topological consistency and all the segmentation paths are determined by geometry (shortest paths) of the surface. Taking semantics into account is sometimes important in segmentation. We therefore also allow users to easily sketch some curve segments to guide decomposition.

Decomposition should be topologically correct to assure validity and consistency of segmentation. Therefore, our system takes user's sketches as **soft constraints**, and try to follow the guidance while at the same time guarantee the newly traced cycles are homotopic to original ones. This can ease user's operations, eliminate the necessity of user's expertise, and greatly improve our system's robustness.

Users can sketch some guiding curves on the mesh for a specific handle or boundary. When we compute the waist of this handle or boundary, we use a special metric m to attract the shortest cycle towards the guiding curves. To design the metric, we first compute each vertex's distance from the guiding curves. A scaling function is defined according to such a distance. The smaller this distance is, the smaller the scale is. Intuitively, regions close to guiding curves shrink while the metric of the distant area preserves or even expands. Under such a metric, the shortest paths will be attracted towards guiding curve segments.

5.7 Decomposition Sequence

The correspondence of handle patches between two surfaces is determined by indexing of handle/tunnel loops. Our system generates a default index sequence for handle and tunnel loops on one surface for its canonical decomposition. The user can reorder the indexing to change the handle correspondence when necessary.

By default, when M and all its handle/tunnel loops (homology group bases) are given as input. We simply project the object onto a plane (e.g. $X - Y$ plane); for each basis (a pair of handle and tunnel loops), we compute their "center" on this plane; then we enumerate the index for each basis according to its center's slope on this plane. As we will show later, an arbitrary indexing order determines a homotopy type of decomposition (and mapping). Users can easily change this order through the interface of our system.

The second issue is the sequence we decompose the base patch. The default sequence is to decompose waists from small indices to large ones. Users can also provide their decomposition sequence script, if desired. Under a specific base patch decomposition sequence, all the pants adjacency relationship can be deduced easily. For a surface with G topological handles (including virtual handles introduced by surgery points) and B boundaries, we can get $2G - 2 + B$ pairs of pants, and totally $6G - 6 + 2B$ adjacency relations.

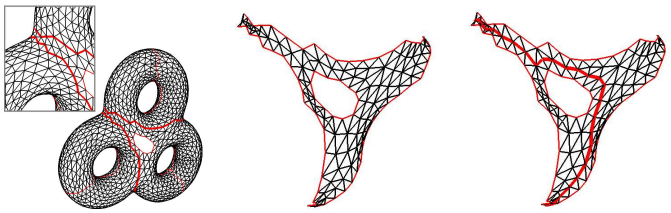


Figure 9: Adaptive Edge-Split. (a) Two waists (thick red curves) are close to each other. No path can cross through the upper left region. (b) Edge-split on base patch before tracing. (c) Shortest cycles pass through successfully.

5.8 Robust Shortest Paths Tracing

We always generate pants by tracing shortest cycles. To assure that the pants do not degenerate, we should prevent shortest cycles from intersecting boundaries. Therefore, here we slightly update the Dijkstra algorithm so that it prevents the shortest paths from hitting boundaries (or from hitting specified curves).

In the Dijkstra algorithm, when a vertex v is visited, we enqueue it and relax its neighbors ([Cormen et al. 2001], page 595). Now if v is on boundaries (or on some specifical curves we want to circumvent), we do not enqueue v nor update its neighbors. Using this updated algorithm, shortest cycles' intersection with boundaries will be prevented.

The original Dijkstra algorithm guarantees to trace a shortest path for an arbitrary vertex on a connected mesh. Our modified algorithm only fails when two boundaries are too close to each other. An example is illustrated in Fig. 9. (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 real go-through path will inevitably intersect boundaries under the current mesh connectivity. We call this kind of spanning edges *dangerous edges*. We perform the edge split on all *dangerous edges* before computing shortest cycles/paths, as shown in (b). This then robustly guarantees the success of our path tracing (c).

There is another technical issue here. Usually we are not only satisfied with the correct topology, but also want the cycles to be apart from boundaries so that the pants will not be too skinny. We may also need to handle similar situations around surgery/feature points and boundaries on the base patch, so that degenerate pants will not be created. Therefore, we also apply a changed metric (as discussed above in Section 5.6) to push shortest paths away from boundaries or some specific curves.

6 Matching Pants Patches

After we perform the consistent pants decompositions on two surfaces M and M' respectively, we get two consistent pants sets $M_i, (i = 0 \dots n)$ and $M'_i, (i = 0 \dots n)$. 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, 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 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 $M_i, i = 0 \dots n$.*

1. *For handle patches $M_0 \dots M_{G-1}$:*
   (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 GO 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 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 [Eck et al. 1995]. 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.

7 Experimental Results

We demonstrate our mapping framework using several examples. The generated surface morphing sequences can be found in our accompanying video.

Automatic Mapping Genus-9 Mechanical Parts. The consistent pants decomposition is automatically performed on three genus-9 mechanical parts. As shown in Fig. 13, although all models in this example have very complicated topology and geometry, once their homotopy group bases are indexed, all the following decomposition is performed in a canonical way without any user involvement. This demonstrates the automation of our framework, and it shows our consistent decomposition is a powerful tool for registering complicated objects. Two morphing sequences generated using our mapping are shown.

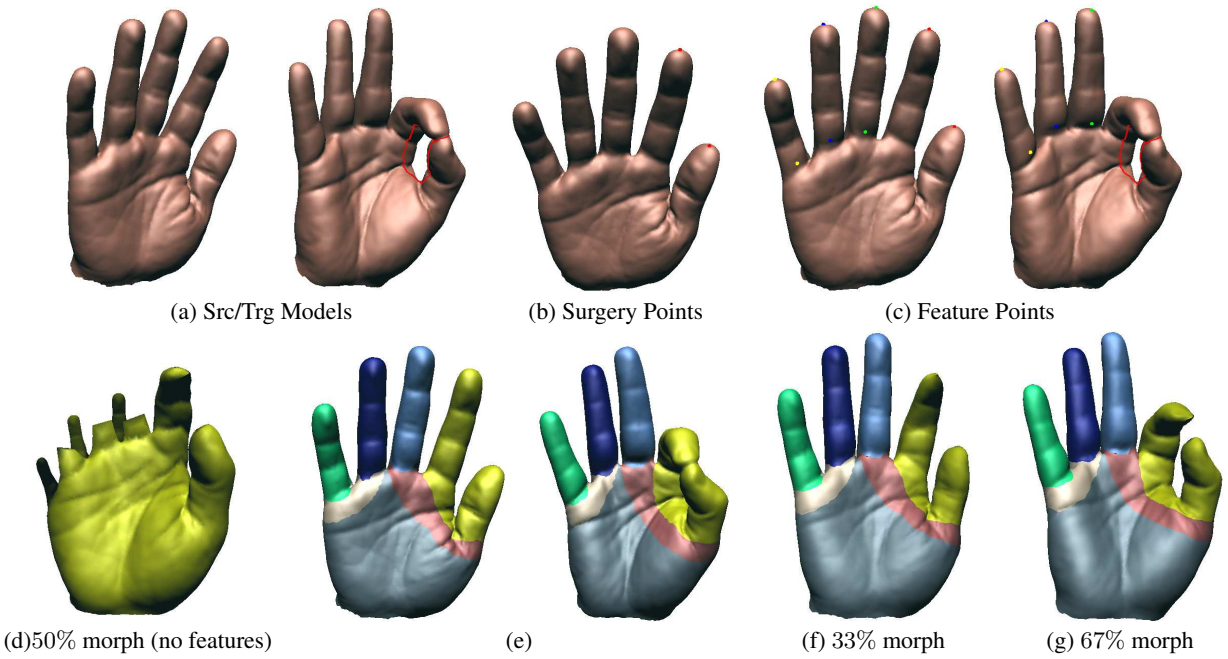


Figure 10: Mapping Hands: "Five" to "Okay". (a). Source and target surfaces. (b). Two surgery points are the least requirements due to the topological difference. (c). Users define more feature points for semantics purpose. (d). Without feature points in (c), 3 fingers are not matched, the morphing is ugly. (e). The refined decompositions result (with feature points). (f) and (g) show the newly generated morphing.

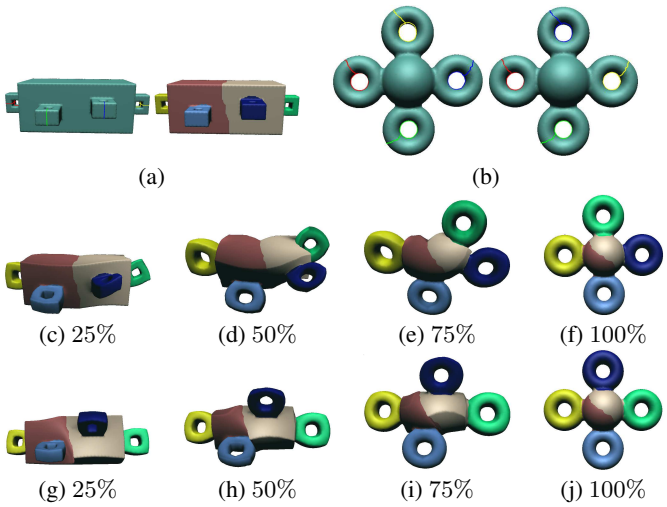


Figure 14: Design Surface Mapping with Arbitrary Homotopy Types. (a). The source surface and its canonical decomposition. (b). User chooses different homotopy types by changing the index of handle/tunnel loops. (c)-(f). First homotopy type: the "green" handle goes up. (g)-(j). Second homotopy type: the "blue" handle goes up.

Deforming Hands: "Five" to "Okay". In this example, we map a human hand (Fig. 10(a) (left)) to another hand (right). This example shows the how semantics are well handled in our framework with intuitive user intervention.

The source hand is an open genus-0 surface, while the target hand is genus-1 with a boundary (red curves are its handle/tunnel loops). To achieve the topological evolution, at least a pair of surgery points is required on the first hand. Naturally, we can set them on tips of the indexing finger and the thumb (Red points in (b)). If we do not provide any other feature points, each hand is decomposed into one pair of pants. The direct mapping between them can be easily computed. However, such a mapping may not follow shape semantics, which can be visualized using a linear interpolated morphing sequence generated from this mapping, as shown in (d): three fingers shrink while three new fingers come out from some places elsewhere. This means those fingers are not matched to each other properly. Our framework allows users to set a few corresponding markers in order to match these fingers, as (c) shows. This results in a finer decomposition as shown in (e). The new mapping is computed using this decomposition result, and guarantees the matching between regions of corresponding fingers; therefore, it generates a more natural morphing sequence as illustrated in (f) and (g).

Genus-4 Greek Model to Genus-3 David Model. In Fig. 11, we map a genus-4 Greek model to a genus-3 David model. The original surfaces and their homotopy group basis correspondence are shown in (a) (each curve's color indicates its index). A pair of surgery points is set on the target model (b), corresponding to the small handle in the lower right part of the Greek sculpture. In (c), when all the handle patches are removed, we get the base patch of both models. Also, we want to semantically guarantee correspondence between head regions, two feature points are placed on each model. We show the canonical decomposition result in (d). From the semantics aspect, 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 left arm (yellow patch)

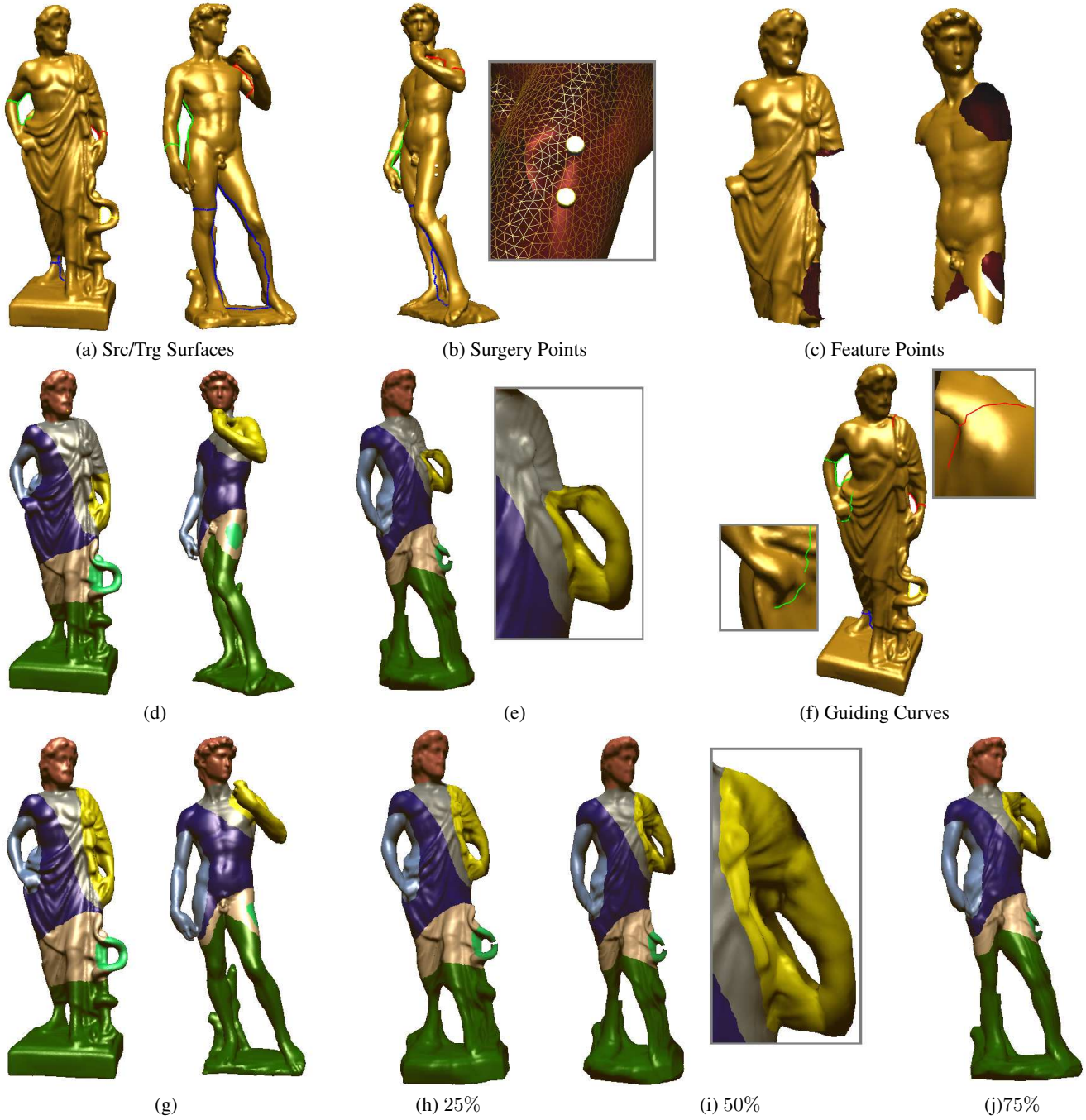
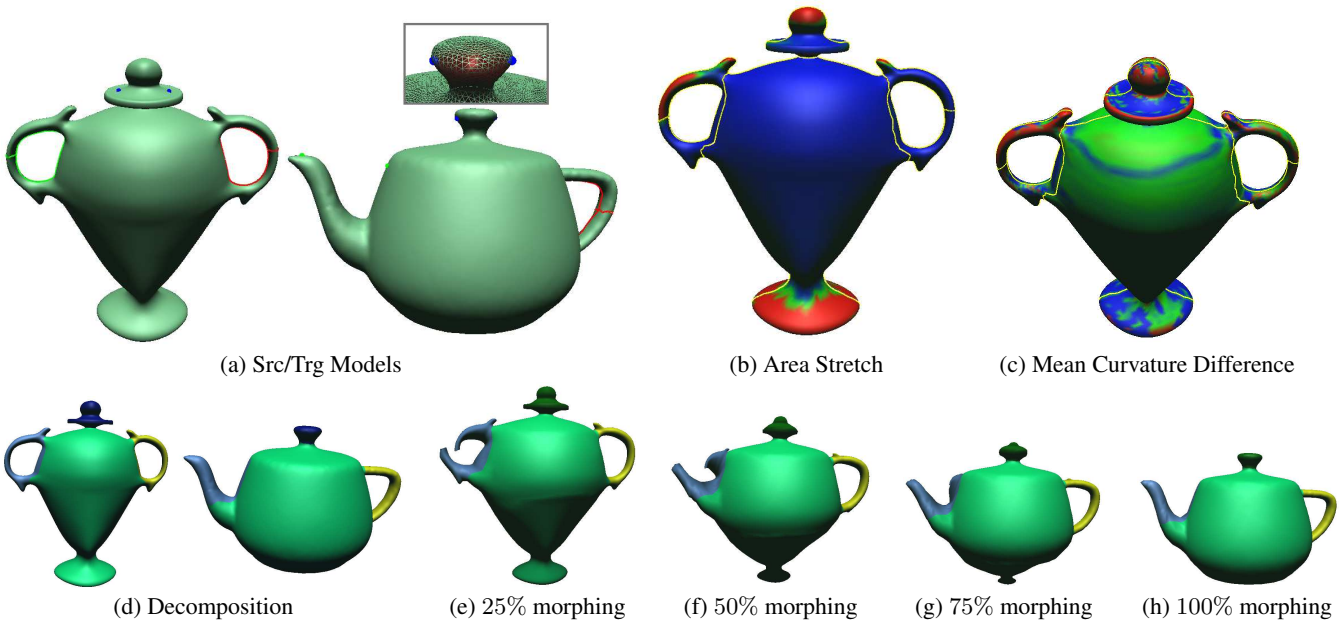


Figure 11: Mapping the Greek Sculpture to David. (a). Two Surfaces and their Homotopy group bases. (b). Two surgery points (matched to the lower right green handle on the Greek). (c). Base patches of both models, and two feature points to assure correspondence on head regions. (d). The decomposition result without further users 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.

Figure 12: 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.

Figure 13: Mapping Genus-9 Mechanical Parts. The initial homotopy group bases on each models are color-encoded in (a). (b) illustrates the canonical decomposition result. The next two rows visualize mappings through morphing sequences.



(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

Figure 15: 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.

is even worse; it cuts through the elbow. An unnatural morph mapping the forearm of the Greek to the whole arm of David is shown in (e). This can be easily remedied if users sketch two short guiding curve segments on the Greek model as shown in (f). The new decomposition result is shown in (g), where we get a morph with much better visual effects as shown in (h)-(j).

Morphing Dragons. In Fig. 12, we perform decomposition on two dragons. Several surgery points and feature points are used to guide the mapping, as depicted on the source and target models. The morphing sequence is shown to demonstrate the mapping effect.

Surface Mappings with Different Homotopy Types. This example demonstrates the rigourousness and completeness of our mapping framework. As shown in Fig. 14, different homotopy classes can be chosen arbitrarily by users, they only need to switch the indexing of the homotopy group basis (as shown in (b)). 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 reasonably good, and it is up to the user to pick up the one they really want. Our framework provides a rigorous way for users to decide an arbitrary homotopy type of the mapping.

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

Fig. 15 illustrates our mapping from a genus-2 vase model to a genus-1 teapot model. (a) shows the models and their handle/tunnel loops; 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 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, and it is color-coded on the surface: red represents the maximum while blue represents the minimum. From the 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([Gu et al. 2004]) that provides an intrinsic energy to measure the shape difference. Therefore, our surface mapping/registration can be used for automatic shape comparison and shape analysis. Before applying our mapping for registering models in a database, each model should go through a preprocessing procedure. For each model, first, its handle and tunnel loops are automatically computed and indexed; second, if necessary for any semantic reason, user can simply reorder indexing of these loops. After this preprocessing, all the following shape comparison and retrieval can be performed automatically.

The complexity of our algorithm is very low, and the performance of our algorithm on most real examples presented here is given in the following runtime table.

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

8 Conclusion and Future Work

We have developed a consistent pants decomposition framework for matching 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 "parts" database ([Funkhouser et al. 2004]) 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 [Li et al. 2007], this decomposition should make the process more robust and general, and it will also provide more semantics control.

Acknowledgement

This work is partially supported by NSF funding CCF-0448399, DMS-0528363, DMS-0626223, IIS-0713145, and IIS-0710819. Most of our data are provided by AIM@SHAPE Shape Repository and Stanford 3D Scanning Repository.

References

ALEXA, M. 1999. Merging polyhedral shapes with scattered features. In *Proc. of the Intl Conf. on Shape Modeling and Applications*, 202–210.

ASIRVATHAM, A., PRAUN, E., AND HOPPE, H. 2005. Consistent spherical parameterization. In *Computer Graphics and Geometric Modeling (CGGM) Workshop*.

CARNER, C., JIN, M., GU, X., AND QIN, H. 2005. Topology-driven surface mappings with robust feature alignment. In *IEEE Visualization*, 543–550.

CORMEN, T., LEISERSON, C., RIVEST, R., AND STEIN, C. 2001. *Introduction to Algorithms*. The MIT Press.

DECARLO, D., AND GALLIER, J. 1996. Topological evolution of surfaces. In *Proc. Graphics interface*, 194–203.

DEY, T. K., LI, K., AND SUN, J. 2007. On computing handle and tunnel loops. In *International Conf. on Cyberworlds*, 357–366.

ECK, M., DEROSE, T., DUCHAMP, T., HOPPE, H., LOUNSBERY, M., AND STUETZLE, W. 1995. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH*, 173–182.

ERICKSON, J., AND WHITTLESEY, K. 2005. Greedy optimal homotopy and homology generators. In *ACM-SIAM Symp. on Discrete algorithms*, 1038–1046.

FLOATER, M. S. 2003. Mean value coordinates. *Computer Aided Geometric Design 20*, 1, 19–27.

FUNKHOUSER, T., KAZHDAN, M., SHILANE, P., MIN, P., KIEFER, W., TAL, A., RUSINKIEWICZ, S., AND DOBKIN, D. 2004. Modeling by example. *ACM Trans. Graph. 23*, 3, 652–663.

GREGORY, A., STATE, A., LIN, M., MANOCHA, D., AND LIVINGSTON, M. 1998. Feature-based surface decomposition for correspondence and morphing between polyhedra. In *Proc. of the Computer Animation*, 64–71.

GU, X., AND YAU, S.-T. 2003. Global conformal surface parameterization. In *Proc. Symp. Geometry Processing*, 127–137.

GU, X., WANG, Y., CHAN, T., THOMPSON, P., AND YAU, S. T. 2004. Genus zero surface conformal mapping and its application to brain surface mapping. *IEEE Trans. Med. Imaging 23*, 8, 949–958.

HATCHER, A., LOCHAK, P., AND SCHNEPS, L. 2000. On the teichmüller tower of mapping class groups. *J. Reine Angew. Math. 521*, 1–24.

KANAI, T., SUZUKI, H., AND KIMURA, F. 1998. Three-dimensional geometric metamorphosis based on harmonic maps. *The Visual Comput. 14*, 4, 166–176.

KENT, J., CARLSON, W., AND PARENT, R. 1992. Shape transformation for polyhedral objects. In *SIGGRAPH '92*, ACM Press, 47–54.

KRAEVOY, V., AND SHEFFER, A. 2004. Cross-parameterization and compatible remeshing of 3d models. *ACM Trans. Graph. 23*, 3, 861–869.

LAZARUS, F., AND VERROUST, A. 1998. Three-dimensional metamorphosis: a survey. *The Visual Computer 14*, 8/9, 373–389.

LEE, A., DOBKIN, D., SWELDENS, W., AND SCHRÖDER, P. 1999. Multiresolution mesh morphing. In *Proc. SIGGRAPH*, 343–350.

LI, X., GUO, X., WANG, H., HE, Y., GU, X., AND QIN, H. 2007. Harmonic volumetric mapping for solid modeling applications. In *Proc. ACM symp. on Solid and physical modeling*, 109–120.

LI, X., BAO, Y., GUO, X., JIN, M., GU, X., AND QIN, H. 2008. Globally optimal surface mapping for surfaces with arbitrary topology. *IEEE Trans. on Visualization and Computer Graphics*, to appear.

PRAUN, E., SWELDENS, W., AND SCHRÖDER, P. 2001. Consistent mesh parameterizations. In *Proc. SIGGRAPH*, 179–184.

SCHREINER, J., ASIRVATHAM, A., PRAUN, E., AND HOPPE, H. 2004. Inter-surface mapping. *SIGGRAPH. 23*, 3, 870–877.

VERDIÈRE, E., AND LAZARUS, F. 2007. Optimal pants decompositions and shortest homotopic cycles on an orientable surface. *J. ACM 54*, 4, 18.

ZÖCKLER, M., STALLING, D., AND HEGE, H.-C. 2000. Fast and intuitive generation of geometric shape transitions. *Visual Computer 16*, 5, 241–253.