Choking Loops on Surfaces

Xin Feng and Yiying Tong, *Member*, *IEEE*

Abstract—We present a method for computing "choking" loops—a set of surface loops that describe the narrowing of the volumes inside/outside of the surface and extend the notion of surface homology and homotopy loops. The intuition behind their definition is that a choking loop represents the region where an offset of the original surface would get pinched. Our generalized loops naturally include the usual $2g$ handles/tunnels computed based on the topology of the genus-g surface, but also include loops that identify chokepoints or bottlenecks, i.e., boundaries of small membranes separating the inside or outside volume of the surface into disconnected regions. Our definition is based on persistent homology theory, which gives a measure to topological structures, thus providing resilience to noise and a well-defined way to determine topological feature size. More precisely, the persistence computed here is based on the lower star filtration of the interior or exterior 3D domain with the distance field to the surface being the associated 3D Morse function.

Index Terms—Computer graphics, computational geometry and object modeling, curve, surface, object representations

1 Introduction

Loops that cannot shrink to a point by deforming over the surface play an important role in topology. In practice, such noncontractible loops have a multitude of potential applications in segmentation, parameterization, topological simplification and repair, path planning, detection of geometrical and topological features, biomedical imaging, and determining integrability of partial differential equations; see, e.g., [1], [2], [3], [4], [5], [6], [7], [8]. A number of algorithms based on surface homology (equivalence classes of such loops, equivalent when they form the boundary of a patch) and homotopy (equivalence classes of such loops, equivalent when they can deform continuously from one to the other) have been proposed. Most of them find $2g$ such loops that form a set of generators for the first homology or homotopy groups of a surface with genus g. Some algorithms can provide geometrically shortest loops for such bases, and some can further classify the loops in a basis into g handles (loops around the solid inside) and g tunnels (loops around the void outside). However, although $2g$ loops are enough to form a basis, there are often still a lot more than $2g$ nontrivial loops that are candidates for topologically and geometrically important structures of the object in various applications. For example, the genus for the buckyball model as shown in Fig. 1 is 31, but there are 90 equally short loops that can replace any of those in the handle-type homology basis.

One way to find these additional loops is to examine different homology classes spanned by combinations of the loops in the basis. A few algorithms allow for the discovery of the shortest loop within a single homology class (i.e., loops that correspond to the sum of several loops in the basis). However, there is no predetermined way of telling which combinations should be used or where to start the search. Even for objects with the same genus, there can in fact be different numbers of useful nontrivial loops depending on the geometry. Even if one opts to count all possible combinations of the $2g$ loops and finds an oracle that distinguishes useful loops, it will still miss handle-like or tunnel-like structures of a genus-0 object: In this case, the basis contains not a single loop to begin with.

One possible solution is to allow the test of whether a loop is contractible to be performed in a local region, for instance, the intersection of the surface with a ball-like local neighborhood of a certain point. This may solve the problem when the global topology of the surface is trivial as it discovers loops that are nontrivial locally. However, the location of the center point of the designated neighborhood is not easy to determine automatically. Another issue with this method is that we may potentially find a lot of nearby locally nontrivial loops even if we add the constraint that they must be also locally shortest, for example, by considering long cylinder-like handles (such as the tail in Fig. 4).

Finding a complete set of shortest nontrivial loops is considered an open problem [9]. However, finding such a set of loops is desirable in many occasions. For instance, when editing or filtering topological noises, such as a thick membrane (like the buckyball), with 32 tiny holes on it, filling the 31 tunnel loops in the homology basis can only turn the model to a genus-0 structure, instead of fixing the topology to a membrane enclosing a ball-like empty space inside. Similarly, objects may be incorrectly connected by a thin tube-like topological noise to form a genus-0 model, and the empty homology basis will not help find a separating loop. To detect where a certain sized object will get stuck at some bottleneck location inside a volume enclosed by a surface can be crucial to motion planning, going through short loops in a homology basis is insufficient, e.g., in a genus-0 model. When geometrically analyzing the easy-to-break handle-like structures in a mechanical part, the handles in the homology basis will only provide a subset of these structures. In biomolecules, finding tunnel like structures is important to identify ion

• *The authors are with the Department of Computer Science and Engineering, Michigan State University, 428 S. Shaw Lane Room 3115, East Lansing, MI 48824. E-mail: {fengxin, ytong}@msu.edu.*

Manuscript received 17 July 2012; revised 27 Nov. 2012; accepted 6 Dec. 2012; published online 9 Jan. 2013.
Recommended for acceptance by L. Kobbelt.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2012-07-0138.
Digital Object Identifier no. 10.1109/TVCG.2013.9.

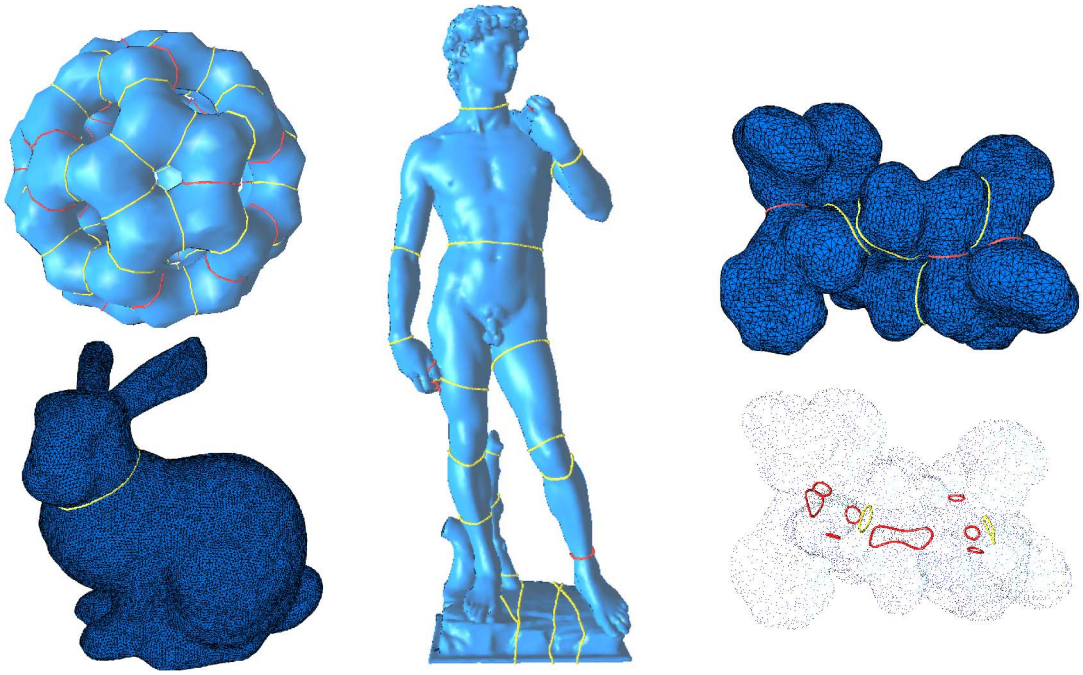

Fig. 1. Upper left: C_{60} "buckyball" is of genus-31, but our algorithm finds 59 more "choking" loops (yellow) than the usual 31 homology generators (red). Lower left: Although the bunny model has a trivial topology, we still find topological features corresponding to the narrowing of the neck. Middle: This David statue model is of genus-5, with three handles near the right hand, one formed by the legs and pedestal, and the last one by the left arm; our approach extracts other topologically relevant handles, e.g., around the waist or the neck. Right: Focusing only on the shortest 1-homology generators of a 1mag protein (top) fails to identify important ion channel loops (bottom, yellow) that our algorithm easily extracts from the surface description.

channels, crucial in determining biological functions and in drug design. As shown in the 1mag model above, shortest tunnel loops actually miss both loops in the ion channel. In all these cases, to be able to detect a complete set of bottleneck loops is a prerequisite. Other potential applications in defining and computing such a full set of loops include analysis of shape and topology of 3D objects, surface parameterization, meshing, and feature detection.

In this paper, we propose a practical definition of topologically and geometrically useful loops using the theory of persistent homology on volumes to address these issues. We also provide an efficient algorithm that produces all of these loops fully automatically. Some of them are indeed topologically trivial in surface topology, but we will make their 3D topological relevance precise through the definitions in Section 3.

The remainder of the paper is organized as follows: In Section 2, we briefly go over the most relevant methods in computational topology related to nontrivial loops on curved surfaces. In Section 3, we provide the necessary mathematical definitions used in persistent homology before giving our definition of choking loops. We describe the procedure of detecting such topological structures in Section 4.1, and provide a method to identify, within an equivalent homotopy class, a discrete approximation of the choking loop on the surface in Section 4.2. We show results in Section 5, and conclude in Section 6.

2 Related Work

There have been a wide variety of algorithms proposed for the purpose of computing homology basis. Some of these methods compute nontrivial loops on the surface mesh directly. The greedy homotopy and homology algorithm [10] gives an optimal solution in theory. Other methods rely on a tetrahedralization of the interior/exterior volume; the HanTun algorithm [4] is the first of these volumetric methods to show results with automatic detection of loops on surfaces, categorized into either handles or tunnels taking geometric measurements into consideration. Our method is also volume based as we need to identify chokepoints.

A number of algorithms have been proposed to find the shortest loop within a single homology class. The problem has been proven NP-hard with coefficients of the homology taken as integer modulo 2 [11]. However, optimal codimension-1 cycle in integer homology classes can be computed in polynomial time for manifold meshes of arbitrary dimension [12]. Given constant genus, the shortest cycle in Z_2-homology classes can be found in $O(n \log \log n)$ time [13], [14]. Most of these methods find shortest loops restricted to closed paths along mesh edges (including, e.g., [15], [16], [17]), with the exception of a few recent methods for computing local minimal loops within a given homotopy class. One of them computes geodesic loops using a discrete geodesic curvature flow [18] based on level set functions. Another method is based on iteratively computing the shortest path inside a triangle strip loop and updating the triangle strip; it is highly efficient with an empirical time complexity of $O(mk)$, where m is the number of vertices in the original loop, and k is the average number of edges in the sequences of edges the loop swept through during the shortening process [19]. We use the latter, but only to refine our results. The constriction loops in [20] are also defined as geodesic loops, but instead of detecting the true narrowing

of the volume inside, they find initial vertices on the surface with large negative Guassian curvature or through a progressive surface simplification [21].

Our definition of choking loops and the associated algorithm to compute them are based on persistent homology theory [22], which measures the life span of each noncontractible loop or membrane when the mesh is incrementally built starting from a single vertex to the full-blown mesh. Here, we define a membrane as a triangle patch in the mesh, homeomorphic to a 2D disk. The persistence endows all such topological structures with a measure, providing a continuous importance scale for topological features as well as resilience to noise. Topological persistence measures are no strangers to computational science, graphics, and visualization. For example, the use of 3D Morse-Smale complexes in the construction of a clean distance field from noisy data is in fact using persistence to filter out critical points that cancel out in pairs through perturbation of the original field [23]. The persistence concept applied to volumetric data can also be used to extract medial structures more robustly than previous methods [24]. Their thinning process can be seen as the complement of the process we use. The topological filtering algorithm based on Reeb graphs [2] can also be seen as using persistence from height functions, although height is often not the most natural choice, even for the tiny nontrivial loops representing topological noise.

As mentioned above, to the best of our knowledge, no existing method attempts to compute the set of all possible candidates of topologically nontrivial loops that are reasonably apart from each other, or even just to formulate them mathematically. Segmentation is a potential application, where the topologically relevant loops can be incorporated as part of patch boundaries; segmentation methods also implicitly generate boundary loops (e.g., [8], [25], [26], [27]). Chazal et al. [27] actually employs 0th persistent homology of a filtration based on a scalar function defined on a point cloud. However, the emphasis of other segmentation methods is often more on surface geometric features such as ridges and valleys (or peaks), and a thorough discussion on these methods is beyond the scope of this paper.

3 Mathematical Background

In the first half of this section, we briefly introduce the concept of homology in topology, and the concept of persistent homology, which provides a way to geometrically measure the topological features. These concepts are crucial to the definitions that we give in the second half of this section, which can indeed be seen as a specifically designed, yet straightforward special case of persistent homology.

3.1 Preliminaries

For a more formal and detailed treatment of persistent homology theory, please refer to [22], [28]. We roughly follow the notation therein in the following brief review.

Given a simplicial complex (tetrahedral mesh) K, which, roughly speaking, is a concatenation of p-simplices (convex hulls of $p+1$ vertices, i.e., vertices for $p=0$, edges for $p=1$, faces for $p=2$, and tetrahedra for $p=3$), we define a p-chain $c=\sum_i a_i\sigma_i$ as a formal linear combination of p-simplices in K, where $a_i \in Z_{/2}$ is an integer modulo 2 and σ_i is a p-simplex. We define the usual boundary operator for each p-simiplex spanned by vertices v_0 through v_p as

$$\partial_p[v_0,\ldots,v_p]=\sum_{i=0}^{p}[v_0,\ldots,\hat{v_i},\ldots,v_p],$$

where $\hat{v_i}$ indicates that v_i is omitted. If we linearly extend the operator to the spaces of chains, we can map p-chains to $p-1$-chains. p-boundaries are p-chains in the image of ∂_{p+1}, and p-cycles are p-chains in the kernel of ∂_p. The pth homology can be defined as the quotient linear space $H_p(K)=Ker\ \partial_p/Im\ \partial_{p+1}$, which is certainly also a group. The quotient can be taken since p-boundaries are always p-cycles. Intuitively speaking, 0-homology generators are connected components; 1-cycles are loops, and 1-homology generators (vectors in a basis for the linear space of 1-chains) are a set of independent nontrivial loops; 2-homology generators are independent membranes, each enclosing one connected component of the "inside" of the surface.

As in [4], we define two types of loops, given a closed surface M separating the 3D space into an inside I (with finite volume) and an outside O (in practice, the volume between the surface and a bounding box).

Definition 1. *A loop in $H_1(M)$ but not $H_1(M\cup I)$ is a handle.*

Definition 2. *A loop in $H_1(M)$ but not $H_1(M\cup O)$ is a tunnel.*

Intuitively speaking, a handle loop can shrink through the inside of the object into a point, and a tunnel loop can shrink through the outside of the object into a point.

Homology groups are topological invariants, and as such, they are not influenced by the lengths defined on the surface or the embedding in 3D space. To enable measurements, persistent homology can be used to give a notion of persistence for each homology generator by measuring its life span in a filtration, which is a nested sequence of subcomplexes of K:

$$\emptyset = K_{-1}\subset K_0\subset \ldots \subset K_n = K.$$

The inclusion map from K_i to K_j $(i\le j)$ induces a mapping from homology groups of earlier subcomplexes to those of later subcomplexes. If we assume that we build the filtration by adding one simplex at a time, each p-simplex will either create a nontrivial p-cycle in the homology of the new subcomplex, or eliminate a nontrivial $p-1$-cycle in the homology of the previous subcomplex. This can be seen as a consequence of the fact that it increases the Euler characteristic ($\chi=\#V-\#E+\#F-\#T$, the alternating sum of numbers of simplices of different dimensions) by $(-1)^p$, and the fact that $\chi=dim(H_0)-dim(H_1)+dim(H_2)-dim(H_3)$, the alternating sum of dimensions of homology groups of different orders. Using positive simplices to represent the homology generators (nontrivial cycles), we can mark the birth time of each homology generator by the order i of the subcomplex K_i. We can pair each negative simplex with the positive simplex representing the nontrivial cycle that it kills, and mark the death time of that nontrivial cycle, j of the sumcomplex K_j. The difference between the two times $j-i$ is the persistence of that nontrivial cycle.

In our algorithm, we use *lower star filtration*, defined by the nested sequence of complexes with simplices added in

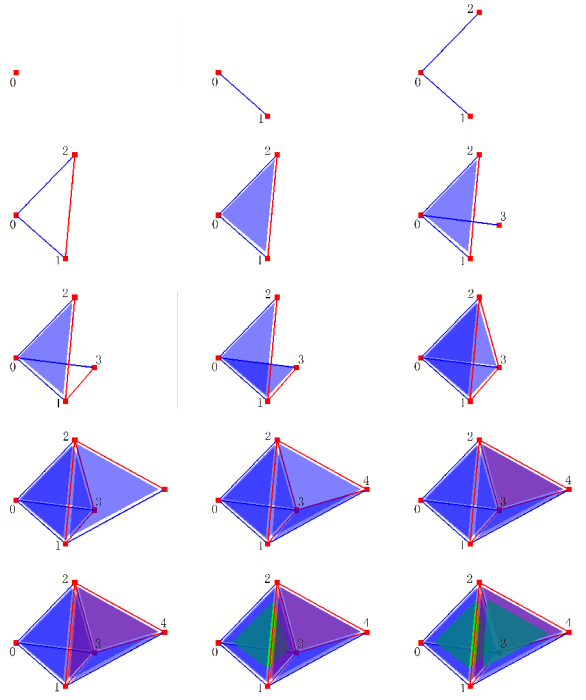

Fig. 2. Here, we show a 3D simplicial complex (tetrahedral mesh) containing two tets. We show the process of building persistent homology using the pairing algorithm. The red simplices are positive, and blue ones negative. We use transparent rendering. Thus, dark blue will show when a negative face is covered by another negative face, and purple faces are positive faces covered by negative faces. To make the negative tets visible, we render both in green.

ascending order of the Morse function d (a function without degenerate critical points, discretely, it can be a function that takes different values at different vertices after symbolic perturbation). One important fact of the pairing algorithm in [22] is that we always kill the *youngest* cycle among all the cycles that could be killed by a negative simplex (the elder's rule); we, thus, avoid converting an important persistent topological structure into a sequence of short-lived nontrivial cycles. In the case of lower star filtration, this rule can also be interpreted as a way to measure the smallest amount of perturbation to the Morse function that is necessary to cancel out a topological structure by its paired simplex. Thus, the persistence of a topological feature is measured by the difference in d, to reduce the dependence on the discretization of the objects.

Here, we show a complex with two tetrahedra as an example for the pairing algorithm in persistent homology (Fig. 2). We have five vertices in this case. The simplicial complex K consists of all the simplices contained in $\{0,1,2,3\}$ and $\{4,1,2,3\}$. The filteration is constructed as follows:

1. positive $\{0\}$;
2. positive $\{1\}$ (two connected components now);
3. negative $\{0,1\}$ killing the younger connected component $\{1\}$;
4. positive $\{2\}$, and negative $\{0,2\}$;
5. positive edge $\{1,2\}$ creating a nontrivial loop, subsequently killed by negative face $\{0,1,2\}$;
6. a few more pairs: $(\{3\},\{0,3\}),(\{1,3\},\{0,1,3\}),(\{2,3\},\{0,2,3\}),(\{4\},\{1,4\}),(\{2,4\},\{1,2,4\}),(\{3,4\},\{1,3,4\})$;
7. positive face $\{2,3,4\}$ on the surface creating one piece of void inside;
8. positive face $\{1,2,3\}$ cutting the void into two pieces;
9. negative tetrahedron $\{0,1,2,3\}$ killing the younger membrane $\{1,2,3\}$;
10. negative tetrahedron $\{1,2,3,4\}$ killing $\{2,3,4\}$ and filling the inside space.

In this example, the only homology structures with persistence greater than 1 (not immediately killed after birth) are the connected component represented by $\{0\}$, and the closed membrane represented by $\{2,3,4\}$. See [4], [22], if the reader wishes to see example pseudocode of the pairing algorithm.

3.2 Definition of Choking Loops

We observe that a handle can be detected as a narrow passage inside the material side of the surface. As we offset the surface toward the interior, the swollen surface will "choke" off the air passage inside the handle. These would naturally include the g first homology generators of the surface, as well as other similar candidates. In fact, these additional choking locations are, formally, second homology generators of the swollen surface, as these membranes now divide the volume enclosed by the surface into more than one connected components. The locations for handle-type 1-homology generators can actually also be seen as where the membranes cut the topologically nontrivial inside volume into a topologically trivial ball-like volume.

More practically, we create a filtration of the tetrahedral mesh of the inside of the surface $M \cup I(=K)$ as follows: First, we assume that a parameter d denoting the distance of each vertex to the surface is stored for each interior vertex. We can use symbolic perturbation to determine the order of vertices at a same distance [29]. We then build a filtration of the surface mesh. Next, we add one interior vertex at a time in ascending order of distance, and add any simplices containing the vertex that can be added without violating the condition of forming a subcomplex. When all vertices within distance d are added, the current subcomplex is the solid object between M and its offset by d toward the interior, which we denote by K_d. In other words, we build the aforementioned lower star filtration using the distance field to the surface for the inside volume. We now define choke face and its associated choking loop.

Definition 3. *A choke face* (3D choke point) *at a distance d from the surface M with persistence δ is a negative face killing a 1-cycle nontrivial in $H_1(M)$ but trivial in K_d, or a positive face representing a 2-cycle nontrivial in both K_d and $K_{d+\delta}$.*

These are locations where an object with a diameter greater than $2d$ will get stuck. If f is a positive face, it separates $K \backslash K_d$ into more connected components. If f is a negative face, $K \backslash K_d$ is cut into a topologically simpler volume (e.g., cutting a torus into a topological ball). The red

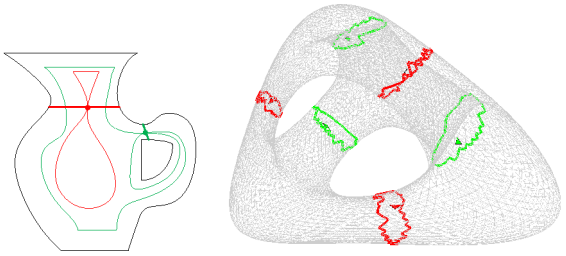

Fig. 3. Left: 2D illustration, when green offset curve is reached, a handle is detected, and when red offset curve is reached, an additional choke point is detected). Right: Display of six 3D choke points (three handles corresponding to the genus-3 in red, and three additional handles in green) with their associated choking loops. On this model, we show the loops before the postprocessing shortening.

triangles and the yellow triangles in Fig. 3 are example positive and negative choke faces, respectively.

We intentionally left out the requirement for persistence δ on the negative faces (corresponding to the original $2g$ nontrivial loops on the surface): Depending on the application they may still be important to the surface topological structure, no matter how nonpersistent they are. However, if required, a condition that would place these negative faces on the same footing as the positive faces is easy to formulate: We measure the persistence of the negative faces by the difference between the distance at which they are found and the distance at which the volume inside that piece of the original surface is filled. This persistence means that these handle loops are cutting the inside volume into a topologically simpler volume long before the void is completely filled up, as the inside passage would not mean much if the inside volume linked by it was about to disappear as a whole. A similar requirement applies to the original tunnels. In this case, we use the difference between the distance to fill up the bounding box representing the outside space within which we put the object and the distance at which the negative face is found. It means that if the persistence is small, i.e., the whole room is about to be filled up before we kill the tunnel of the object in that room, the tunnel would have been almost as wide open as the room to begin with.

Definition 4. *A choking loop associated with a choke face f (first added to the filtration in K_d) is the loop* on the surface M, *formed as the boundary of the smallest membrane B containing f, such that $B\backslash f$ is homotopic to the boundary of f in $K_d\backslash f$.*

Intuitively speaking, we deform f to the membrane B by growing it inside K_d, turning it from what locally separates $K\backslash K_d$ into what locally separates K, the entire volume inside (e.g., Fig. 5). Boundary of the smallest membrane going through the choke point is not necessarily a geodesic loop, although very close to one in practice. However, this can be a more reasonable requirement in finding the narrowing in the volume, e.g., in medical applications for detecting constrictions of airway or blood vessels, as the area of the membrane limits the capacity of fluid flow roughly speaking.

We can also define all tunnel-like nontrivial loops similarly, which can be named "external" choking loops. In this case, we offset the surface along positive normal direction to build the filtration, and again examine its persistent homology. In practice, we create a large bounding box of the surface, and treat the space between the surface and the bounding box as the volume in the above definition. The external choking loops include the g tunnels (that cut the volume outside into a topological ball if we see the 3D space as part of the 3D ball S^3) as well as other membranes, cutting the space outside into pieces, enclosing most of the pieces, and leaving only one outside.

In the above definition, we used two parameters d and δ, both of which are rather intuitive. d denotes how far we have to offset the surface to create the choking loop. The use of δ avoids creating many duplicate loops that would have been merged when the offset is changed by δ, providing resilience to geometric noise. Thus, our definition can create useful loops even for genus-0 objects, but it will not create cluttered clusters of loops. For high genus models, one might occasionally find a choking loop associated with a positive choke face before finding any handles, as there may be a very narrow passage connecting the bulk of the inside volume to another large piece of volume (roughly speaking, with radius greater than δ) enclosed by a topologically trivial patch of the surface.

4 Choking Loop Calculation

Built on persistent homology with the filteration ordered by a distance function, our algorithm requires a volume mesh with sufficient internal vertices to discern the distances at which the choking loops are detected. In contrast to the HanTun algorithm [4], also based on persistent homology but with a volume mesh containing only surface vertices, we employs 2-homology as well as 1-homology to detect the additional choking loops. Furthermore, our distance-based homology gives the loops a geometrically relevant ordering and associated persistence.

4.1 Detection of Nontrivial Topology

We now present the procedure used to compute choke faces. Without loss of generality, we only describe handle-like choking loops here. Before we start building the filtration, we first preprocess the surface representation. If we are given a surface triangle mesh, we create a tetrahedralization of the boundary surface using Tetgen [30]. The distance field to the surface can be estimated by a number of different ways. For example, we may run Closest Point Transform [31] to create a distance field on a regular grid of the bounding box of the object, followed by trilinear interpolation of the distance field on the grid for the internal vertices of the tetrahedral mesh. Alternatively, we can run a multisource Dijkstra's algorithm computing the shortest distance through edges from the surface for each vertex. For implicit surfaces, fast marching can be performed to create the distance field if the level set function is not already a signed distance field, and we perform marching cubes to create a surface mesh, and proceed with the tetrahedralization and trilinear interpolation.

The filtration parameter/time is the distance to the surface. Thus, at time 0, we start the filtration by adding all cells of the boundary surface, e.g., in a breadth-first traversal from a seed vertex. When the whole closed surface is in the filtration, we will have $2g$ positive edges left representing the

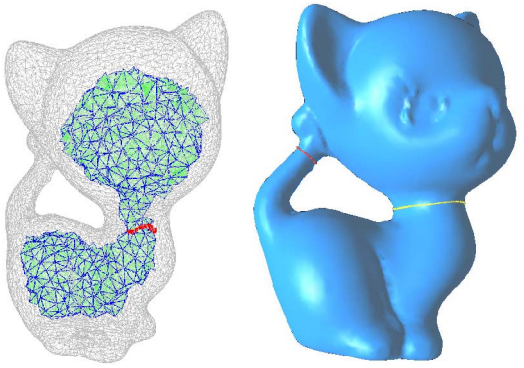

Fig. 4. Our filtration is built in the order of distance from the surface. The filtration when the choking loop shown in yellow to the right is detected is the volume between the offset surface shown in green and the original surface. The 1-homology handle shown in red to the right (around the tail) has already been killed when we reach this offset distance.

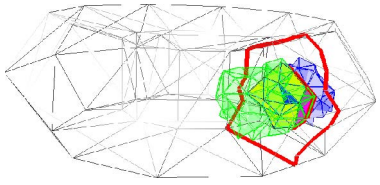

Fig. 5. Starting from the blue triangle representing the choke face, we gradually expand the membrane separating the left (blue tets) and the right (green tets) internal space toward the surface, until the loop is entirely on the boundary. In the step shown here, the purple faces will be added to the yellow membrane. The final result is the red loop on the surface of the torus.

$2g$ homology generators. Next, we add all interior edges with both vertices on the surface into the filtration, followed by interior faces with all three vertices on the boundary in the ascending order of the number of edges inside the volume, and the tets with all vertices on the boundary. We then add the vertex with the smallest d, followed by the simplices in the tetrahedral mesh containing the new vertex, i.e., the edges connecting it to vertices already in the current subcomplex, the faces formed by the vertex and two previous vertices, and the tets formed by the vertex and three previous vertices. We repeat the process, until we reach the distance d_{max}. Any positive faces with persistence greater than δ_{min} and any negative faces paired with positive edges on the surface will be identified as choke faces.

This is to our knowledge, the first application of persistent 2-homology for extracting surface loops. Unlike [4] (using only persistent 1-homology), we need to handle the pairing between tets and faces as well as the pairing between faces and edges. With our particular order of adding simplices to the filtration, most of the positive simplices will only have a small persistence.

4.2 Computation of the Associated Surface Loops

We now present a robust method to compute an approximation of the membrane B starting from a choke face. Intuitively speaking, we gradually deform the boundary loop of the membrane approximately along the gradient of the distance field to reach the surface in a fastest (greedy) way, so that the membrane swept will be a minimal one touching the boundary. We can see the procedure as partitioning the nearby tets in K into one connected cluster to the left of the membrane, and another to the right of the membrane, as in min-cut of the dual graph. We use the following fast procedure:

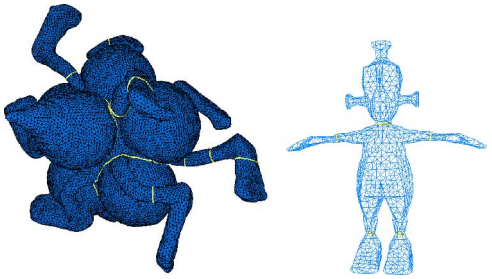

Fig. 6. A few more genus-0 models and their choking loops.

1. Add the choke face f to the membrane, and add the two tets adjacent to f to the two clusters.
2. Pick the vertex v_k in the boundary loop $(v_0v_1 \ldots v_nv_0)$ with the largest distance d from the surface.
3. Find the local optimal membrane patch within the one-ring neighborhood of v_k (within the filtration before the seed face is included), such that it morphs $v_{k-1}v_kv_{k+1}$ to a path connecting v_{k-1} to v_{k+1} on the boundary of the one ring. This membrane patch partitions tets in the one ring into left and right, consistent with those already classified as left or right.
4. Merge the local membrane patch separating the two classes of the one ring to the membrane.
5. Repeat Steps 2, 3, and 4 until all boundary edges of the membrane are on the surface.

In Step 3, the membrane is optimal in terms of an average of the length of its edges on the boundary of the one ring (to reduce area) and the distance of the corresponding vertices along the path to the boundary (to reach the surface fast).

Finally, to improve the geometric shape of the result, we employ the method in [19] to allow the path to go through the surface triangles instead of being restricted on edges, thus creating a smoother loop. We can allow it to reach a locally minimal length in terms of geodesic distance, or stop after few iterations to smooth the loop without deviating far. In practice, the edge loops are very close to geodesic loops to begin with.

5 Results and Discussion

We ran our algorithm on a few genus-0 models (shown in Fig. 6). These surface loops, e.g., shown on the bunny model, have well-defined topological meaning of local separating membranes as given in the mathematical definition of choking loops.

For high genus models, there can be a lot of legitimate candidates for the shortest loops that can form a basis of the 1-homology group, as well as additional surface loops that do not belong to combinations of this bases, as in the genus-0 case. As shown on the C_{60} model (the Buckyball, genus 31), our algorithm produces all 90 useful handle loops, as opposed to only 31 of them produced by other methods. We also find the complete set of 32 tunnels.

We observe in our tests that there are often many more handle-type choking loops than homology generators (Figs. 7, 8, and 9), because there are often more narrowing passages for the inside of the object. However, for protein models (e.g.,

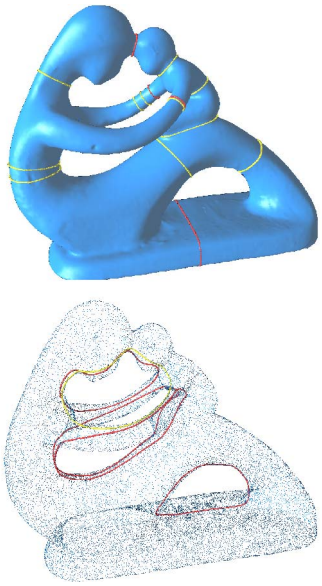

Fig. 7. The handles and tunnels on fertility model. We render only the vertices when showing the tunnels to make them more visible. The red loops can be obtained by homology generators, and the yellow loops are the additional choking loops.

Fig. 8. A number of additional topologically choking loops can be found on the Neptune model.

Fig. 9. More models showing the tunnels detected by our algorithm.

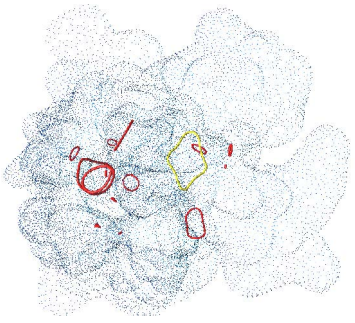

Fig. 10. We find one additional tunnel loop aside from the 1-homology generators for 2kix protein. Some of the red loops here can be seen as topological noise.

Fig. 10), there can be more tunnels than those obtained by 1-homology of surfaces. Those tunnels are important in automatic detection of ion channels crucial in analysis of the biomolecular surfaces. Those tunnels allow ions to flow past membranes of cells, and they play important biological roles, e.g., in nerve impulse of the nervous sytem. For models with knots (boundary of Seifert's surface [32]), we did not find additional choking tunnels.

The maximum offset distance d_{max} to specify how far we want to go inside the volume, and the minimum persistence δ_{min} can be used to filter out noise-like structures. In most of tests, we found the default setting of $d_{max} = 50\%$ and $\delta_{min} = 10\%$ to produce satisfying results, with all the important loops included without introducing a high density of loops. Alternatively, we can set $d_{max} = 100\%$ with a low δ_{min} to extract all useful chokepoints, and allow the user to adjust them interactively after the first run. The most costly step in our implementation is the persistent homology pairing algorithm (see Table 1). While the worst case of computing persistent homology has upper bound of $O(m^3)$, where m is the size of the simplicial complex, the practical running time appears to be nearly linear [22]. All the other steps (computing persistence to detect choke faces, tracing back to find choking loops, and postprocessing) take mostly less than a second. If we are interested only in smaller features for a particular application such as filtering, we can set d_{max} to a small number, which greatly improves the efficiency.

The tessellation of the inside and outside space generates numbers of vertices roughly proportional to the numbers of surface vertices, which was enough to compute the choking loops, since the efficient postprocessing can improve the geometric shape. We have tested on both interpolated signed distance field and Dijkstra distance from the edge graph, and found similar results even at low resolution. As long as the distance field is accurate enough to discern the life cycles of persistent choking loops, the results are insensitive to tessellation differences in our tests, especially after the proposed postprocessing. See Fig. 11 for a typical example. In case the application requires better precision for feature size and persistence, a high-resolution tetrahedral mesh can be used.

6 Conclusion

We present a method to compute nontrivial loops that are not possible to produce using conventional topological methods for surfaces. Our contribution is twofold: We first provide a mathematical definition for such loops using persistent homology theory; we then provide an efficient algorithm based on our definition. Theoretically, our definitions can be seen as related to topological structures in an extension of proximity complexes. While proximity

TABLE 1
Statistics of the Results (All Time Measurements in Milliseconds, Taken on a Windows 7 System with Intel Core i7@2.8 GHz and 12-GB RAM)

model	surf#v	total#v #f #t	TetGen	prep	surf	inside	basis	choke	g	k	param	$d_{max}(\%)$	$\delta_{min}(\%)$
armadillo	40K	70K,656K,307K	13,011	12,948	3,292	79,763	0	531	0	4	pfq1.2	50	6
1mag	17K	33K,328K,155K	10,437	4,773	1,358	19,094	390	671	8	6	pfq1	75	15
bunny	26K	98K,1088K,531K	17,846	11,060	2,387	91,026	0	640	0	1	pfq1	75	2.5
david	26K	46K,430K,202K	8,955	5,725	2,761	25,709	1,154	234	5	3	pfq1.2	50	10
fertility	47K	99K,985K,469K	29,235	17,924	3,245	79,935	811	344	4	2	pfq1.2	50	10
kitten	8K	17K,166K,79K	3,728	1,482	546	10,187	47	140	1	1	pf1.2	50	15
neptune	32K	52K,471K,219K	9,890	8,549	9,516	130,963	94	4,805	3	7	pfq1.2	50	10
tangle	7K	11K,97K,45K	2,809	1,107	546	5,335	47	94	5	7	pfq1.2	50	10

From left to right: Surface vertex number, total vertex number, tetrahedralization time by TetGen, preprocessing time (distance field construction), time running persistent homology for surface mesh, time running persistent 1-homology and 2-homology for inside volume, time to find and postprocess all the homology generators in the basis and all additional choking loops, genus g, number of additional choking loops k, TetGen parameters used, maximum distance in building the filteration, and persistence threshold for the loops shown. Only timing for handles is reported, as that for tunnels is similar.

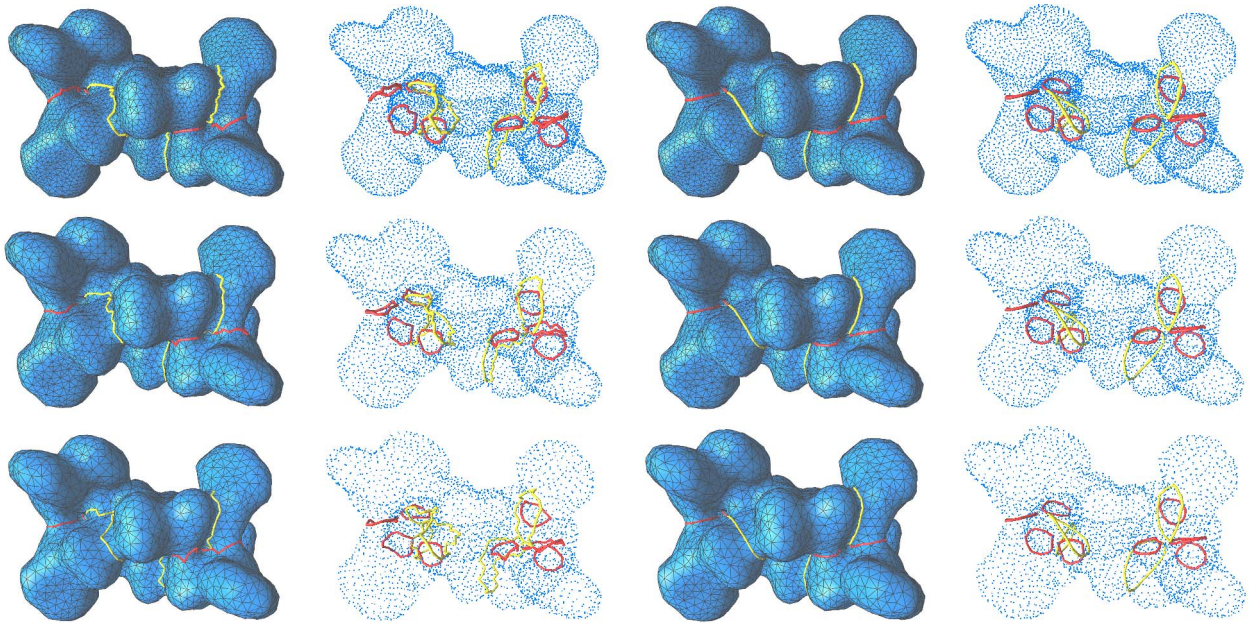

Fig. 11. Comparison of the handle loop results on 1mag at different resolutions. All the loops (eight homology generators in red and four additional choking loops in yellow) are captured by setting $d_{max} = 70\%$ and $\delta_{min} = 25\%$. Top to bottom: Meshes with surface vertex counts 7.7k, 4.8k, and 3.4k (TetGen parameters pfq1.2a0.5, pfq1.2a1, and pfq1.2a1.5, resp.); left to right: Choking loops, their unoccluded view, postprocessed loops, and their unoccluded view.

complexes are built from unions of balls with radius d in a finite point set, we use all the points on a surface.

A potential limitation of the algorithm is that if there are two nearby short loops, the shorter one might push the other slightly away from the geometric optimal location, depending on the persistence δ one chooses—although we found in practice that the postprocessing loop shortening step can usually move them to decent locations (e.g., loops near the left knee of David statue in Fig. 1). Another potential problem is that the loops discovered first can be around a cross section in the shape of a long thin rectangle instead of a cross section in the shape of a disk, (e.g., those on the basis of David statue). However, we can eventually detect all these loops, and simply sort them again based on lengths. We can also easily discard such loops if required by the application, by allowing the loop around the choke face to deform only within a certain distance, and eliminating those failed to reach the surface. On the other hand, for motion planning or constriction detection, this long narrow passage should be detected earlier than shorter geodesic loops, as a ball with radius d will get stuck.

For future work, we plan to explore the possibilities of improving computational time for level sets (used by many biomedical or biomolecular applications), leveraging the regular grid structure of the inside and outside domains, or using the implicit representation directly. We also wish to explore other 3D Morse functions (e.g., distance from the medial axes, diffusion times, or diffusion distances) to guide the construction of the filtration used in persistent homology. An obvious application that derives from our method is topological filtering, where we only need to set a short distance for the surface offset to kill tiny handles (offset inward) and to kill tiny tunnels (offset outward).

ACKNOWLEDGMENTS

The authors thank the anonymous reviewers for their valuable suggestions, and Beibei Liu for her help on the postprocessing of the loops. This work was supported in part by National Science Foundation (NSF) Grants CCF-0936830 and IIS-0953096.

REFERENCES

[1] D. Letscher and J. Fritts, "Image Segmentation Using Topological Persistence," *Proc. 12th Int'l Conf. Computer Analysis of Images and Patterns (CAIP '07),* http://dl.acm.org/citation.cfm?id=1770904.1770985, pp. 587-595, 2007.

[2] Z. Wood, H. Hoppe, M. Desbrun, and P. Schröder, "Removing Excess Topology from Isosurfaces," *ACM Trans. Graphics,* vol. 23, pp. 190-208, http://doi.acm.org/10.1145/990002.990007, Apr. 2004.

[3] P.-T. Bremer, E.M. Bringa, M.A. Duchaineau, A.G. Gyulassy, D. Laney, A. Mascarenhas, and V. Pascucci, "Topological Feature Extraction and Tracking," *J. Physics: Conf. Series,* vol. 78, no. 1, http://stacks.iop.org/1742-6596/78/i=1/a=012007, p. 012007, 2007.

[4] T.K. Dey, K. Li, J. Sun, and D. Cohen-Steiner, "Computing Geometry-Aware Handle and Tunnel Loops in 3D Models," *ACM Trans. Graphics,* vol. 27, pp. 45:1-45:9, http://doi.acm.org/10.1145/1360612.1360644, Aug. 2008.

[5] M. Desbrun, E. Kanso, and Y. Tong, "Discrete Differential Forms for Computational Modeling," *Proc. ACM SIGGRAPH ASIA Courses,* pp. 15:1-15:17, http://doi.acm.org/10.1145/1508044.1508059, 2008.

[6] D. Boltcheva, D. Canino, S.M. Aceituno, J.-C. Léon, L. De Floriani, and F. Hétroy, "An Iterative Algorithm for Homology Computation on Simplicial Shapes," *Comput.-Aided Design,* vol. 43, no. 11, pp. 1457-1467, Nov. 2011.

[7] E. Zhang, K. Mischaikow, and G. Turk, "Feature-Based Surface Parameterization and Texture Mapping," *ACM Trans. Graphics,* vol. 24, no. 1, pp. 1-27, http://doi.acm.org/10.1145/1037957.1037958, Jan. 2005.

[8] S. Katz and A. Tal, "Hierarchical Mesh Decomposition Using Fuzzy Clustering and Cuts," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 954-961, http://doi.acm.org/10.1145/882262.882369, July 2003.

[9] T.K. Dey, K. Li, and J. Sun, "Computing Handle And Tunnel Loops with Knot Linking," *Computer-Aided Design,* vol. 41, no. 10, pp. 730-738, http://linkinghub.elsevier.com/retrieve/pii/S0010448509000153, Oct. 2009.

[10] J. Erickson and K. Whittlesey, "Greedy Optimal Homotopy and Homology Generators," *Proc. 16th Ann. ACM-SIAM Symp. Discrete Algorithm,* http://dl.acm.org/citation.cfm?id=1070432.1070581, pp. 1038-1046, 2005.

[11] C. Chen and D. Freedman, "Quantifying Homology Classes," *Science,* pp. 169-180, http://arxiv.org/abs/0802.2865, 2008.

[12] T.K. Dey, A.N. Hirani, and B. Krishnamoorthy, "Optimal Homologous Cycles, Total Unimodularity, and Linear Programming," *SIAM J. Computing,* vol. 40, no. 4, pp. 1026-1044, 2011.

[13] E. Chambers, J. Erickson, and A. Nayyeri, "Minimum Cuts and Shortest Homologous Cycles," *Proc. Symp. Computer Geometry,* 2009.

[14] G.F. Italiano, Y. Nussbaum, P. Sankowski, and C. Wulff-Nilsen, "Improved Algorithms for Min Cut and Max Flow in Undirected Planar Graphs," *Proc. 43rd Ann. ACM Symp. Theory of Computing (STOC),* pp. 313-322, 2011.

[15] S. Cabello, E. Colin de Verdière, and F. Lazarus, "Finding Shortest Non-Trivial Cycles in Directed Graphs on Surfaces," *Proc. Ann. Symp. Computational Geometry,* pp. 156-165, http://doi.acm.org/10.1145/1810959.1810988, 2010.

[16] E.C.d. Verdière and F. Lazarus, "Optimal System of Loops on an Orientable Surface," *Proc. 43rd Symp. Foundations of Computer Science (FOCS '02),* pp. 627-636, http://dl.acm.org/citation.cfm?id=645413.652186, 2002.

[17] F. Lazarus, M. Pocchiola, G. Vegter, and A. Verroust, "Computing a Canonical Polygonal Schema of an Orientable Triangulated Surface," *Proc. 17th Ann. Symp. Computer Geometry (SCG '01),* pp. 80-89, http://doi.acm.org/10.1145/378583.378630, 2001.

[18] C. Wu and X. Tai, "A Level Set Formulation of Geodesic Curvature Flow on Simplicial Surfaces," *IEEE Trans. Visualization and Computer Graphics,* vol. 16, no. 4, pp. 647-662, http://dx.doi.org/10.1109/TVCG.2009.103, July 2010.

[19] S.-Q. Xin, Y. He, and C.-W. Fu, "Efficiently Computing Exact Geodesic Loops within Finite Steps," *IEEE Trans. Visualization and Computer Graphics,* vol. 18, no. 6, pp. 879-889, June 2012.

[20] F. Hétroy, "Constriction Computation Using Surface Curvature," *Eurographics (Short Paper),* pp. 1-4, http://hal.inria.fr/inria-00001135, 2005.

[21] F. Hétroy and D. Attali, "Detection of Constrictions on Closed Polyhedral Surfaces," *Proc. Symp. Data Visualisation (VISSYM '03),* pp. 67-74, http://dl.acm.org/citation.cfm?id=769922.769929, 2003.

[22] H. Edelsbrunner, D. Letscher, and A. Zomorodian, "Topological Persistence and Simplification," *Proc. 41st Ann. Symp. Foundation of Computer Sciences,* pp. 454-, http://dl.acm.org/citation.cfm?id=795666.796607, 2000.

[23] A. Gyulassy, M. Duchaineau, V. Natarajan, V. Pascucci, E. Bringa, A. Higginbotham, and B. Hamann, "Topologically Clean Distance Fields," *IEEE Trans. Visualization and Computer Graphics,* vol. 13, no. 6, pp. 1432-1439, http://dx.doi.org/10.1109/TVCG.2007.70603, Nov. 2007.

[24] L. Liu, E.W. Chambers, D. Letscher, and T. Ju, "A Simple and Robust Thinning Algorithm on Cell Complexes," *Computer Graphics Forum,* vol. 29, no. 7, pp. 2253-2260, 2010.

[25] L. Shapira, A. Shamir, and D. Cohen-Or, "Consistent Mesh Partitioning and Skeletonisation Using the Shape Diameter Function," *Visual Computer,* vol. 24, no. 4, pp. 249-259, http://dx.doi.org/10.1007/s00371-007-0197-5, Mar. 2008.

[26] A. Golovinskiy and T. Funkhouser, "Consistent Segmentation of 3D Models," *Computers and Graphics (Shape Modeling Int'l 09),* vol. 33, no. 3, pp. 262-269, June 2009.

[27] F. Chazal, L.J. Guibas, S.Y. Oudot, and P. Skraba, "Analysis of Scalar Fields over Point Cloud Data," *Proc. 20th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA '09),* pp. 1021-1030, http://dl.acm.org/citation.cfm?id=1496770.1496881, 2009.

[28] H. Edelsbrunner and J. Harer, *Computational Topology: An Introduction,* series Applied Math. Am. Math. Soc., http://books.google.com/books?id=MDXa6gFRZuIC, 2010.

[29] H. Edelsbrunner and J. Harer, *Computational Topology: An Introduction.* Am. Math. Soc., 2010.

[30] H. Si, "Constrained Delaunay Tetrahedral Mesh Generation and Refinement," *Finite Elements Analysis Design,* vol. 46, pp. 33-46, http://dl.acm.org/citation.cfm?id=1660153.1660230, Jan. 2010.

[31] S. Mauch, "Closest Point Transform to a Triangle Surface," http://www.cacr.caltech.edu/sean/projects/cpt/html3/, 2004.

[32] J.J. van Wijk and A.M. Cohen, "Visualization of Seifert Surfaces," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 4, pp. 485-496, http://dl.acm.org/citation.cfm?id=1137246.1137503, July 2006.

Xin Feng received the BS and MS degrees from Harbin Institute of Technology, China, and is currently working toward the PhD degree in the Department of Computer Science and Engineering at Michigan State University. His research interests include geometric modeling, shape analysis, and computer graphics.

Yiying Tong received the PhD degree from the University of Southern California in 2004. He is an assistant professor at Michigan State University. Prior to joining MSU, he was a postdoctoral scholar at Caltech. His research interests include discrete geometric modeling, physically based simulation/animation, and discrete differential geometry. He received the US National Science Foundation (NSF) Career Award in 2010. He is a member of the IEEE.