This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3211781

Fast Computation of Neck-like Features

Hayam Abdelrahman,Yiying Tong, *Member, IEEE*

Abstract—Locating neck-like features, or locally narrow parts, of a surface is crucial in various applications such as segmentation, shape analysis, path planning, and robotics. Topological methods are often utilized to find the set of shortest loops around handles and tunnels. However, there are abundant neck-like features on genus-0 shapes without any handles. While 3D geometry-aware topological approaches exist to find neck loops, their construction can be cumbersome and may even lead to geometrically wide loops. Thus we propose a "topology-aware geometric approach" to compute the tightest loops around neck features on surfaces, including genus-0 surfaces. Our algorithm starts with a volumetric representation of an input surface and then calculates the distance function of mesh points to the boundary surface as a Morse function. All neck features induce critical points of this Morse function where the Hessian matrix has precisely one positive eigenvalue, i.e., type-2 saddles. As we focus on geometric neck features, we bypass a topological construction such as the Morse-Smale complex or a lower-star filtration. Instead, we directly create a cutting plane through each neck feature. Each resulting loop can then be tightened to form a closed geodesic representation of the neck feature. Moreover, we offer criteria to measure the significance of a neck feature through the evolution of critical points when smoothing the distance function. Furthermore, we speed up the detection process through mesh simplification without compromising the quality of the output loops.

Index Terms—Computer Graphics, Computational Geometry, and Object Modeling, Curve, Surface, Object Representations

1 Introduction

When we grab objects, we naturally reach for thin parts between thicker ends. This defines the concept of "neck" as in [1], where such structures help a robot determine how to manipulate 3D shapes in its environment. "Lassoing" around such neck features leads to closed geodesics, which is helpful in a wide range of computer graphics and geometric modeling applications, such as segmentation, parameterization, shape analysis, topological filtering and repair, structural weakness detection, wrapping 3D objects, etc. (e.g., [2], [3], [4], [5], [6], [7], [8], [9]).

Many existing methods for calculating shortest loops are based on the computation of a set of $2g$ noncontractible loops that can cut a surface with genus-g into a topological disk. These loops can be further cleaned and classified into g handles (contractible through the interior volume) and g tunnels (contractible through outside space). However, the number of neck-like features does not only depend on g. For example, as shown in Fig. 7, in the Kitten model with genus-1, there are two detected handles, one is a handle around the tail, and the other is a handle around the neck; the latter cannot be detected using existing methods. Similarly, in Fig. 1, there are many more neck loops in the genus-4 Fertility model than the expected 4 handles; as for the 4-genus model, there should be 2g homology generators; 4 handles, and 4 tunnels. Moreover, all loops on genus-0 models are contractible, but some such models contain prominent neck features, as in the Bunny and Toy models in Figure 8.

One recent method [10] proposed a topological algorithm for computing all possible neck loops. It provided a mathematical definition of such loops based on the lower-star filtration of the distance to the surface. This method uses persistent homology to measure the life span of each noncontractible loop during the filtration process. However, the construction is complex and can lead to features that do not resemble a neck.

Instead, we propose using the critical points of a processed distance function as a Morse function to find both the location and evaluate the significance of a possible neck-like feature. Critical points of a Morse function defined on a volume provide rich topological and geometric information about the structure of the shape. Thus, they are closely related to the above lower-star filtration-based approach. However, we take a shortcut based on the geometry and directly construct planes that cut through the neck feature, resulting in initial neck loops on surfaces. We further employ regularly used Laplacian smoothing to remove noise-like features in addition to direct geometric criteria such as loop size and distance to nearby loops.

We briefly discuss the most relevant work in Sec. 2, then provide the mathematical background in Sec. 3 on Morse functions (Sec. 3.1, (Sec. 3.2), and Laplacian smoothing (Sec. 3.3). Finally, in Sec. 4, we explain our algorithm in detail and show our results in Sec. 5 before concluding in Sec. 6.

2 Related Work

Many algorithms have been proposed for computing a homology basis. Some methods use a tetrahedralization of the interior/exterior volume to detect noncontractible loops on surfaces automatically. Among these, the HanTun algorithm [11] is the first volumetric method to compute and categorize surface loops into either handles or tunnels with geometric measurements taken into account for large practical models. A more efficient extension to HanTun was proposed in [12], which computes a basis for handle and tunnel loops on a surface mesh based on Reeb graphs; the

- *Hayam Abdelrahman and Yiying Tong are with the Department of Computer Science and Engineering, Michigan State University, 428 S. Shaw Lane Room 3115., East Lansing, MI 48824.*
E-mail: ytong@msu.edu

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3211781

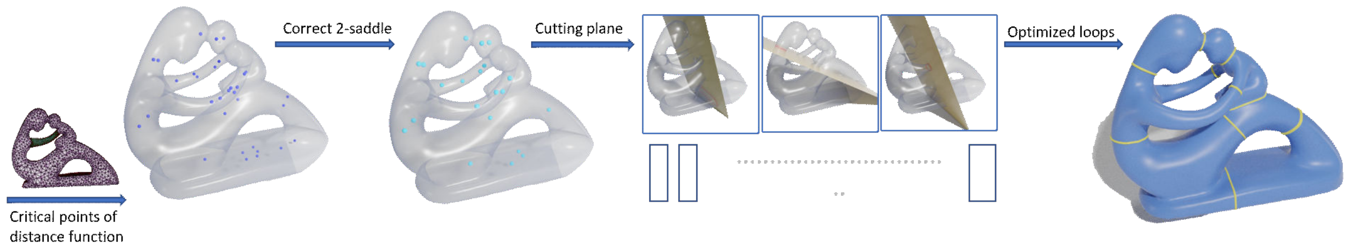


Fig. 1: **Neck-like feature.** Our pipeline starts with calculating the candidate neck feature locations as a specific type of critical points (2-saddles) of the distance to the surface in the volume. The candidates that passed the selection criteria are chosen as the centers of cutting planes that produce loops around the neck features, which are then tightened and cleaned up further.

loops are further tightened for geometric relevance. For optimal codimension-1 cycles (including 1-cycles on surfaces) in integer homology classes, [13] proved a surprising result, namely that the problem can be calculated in polynomial time, despite the NP-hard nature of $\mathbb{Z}_2$ coefficients. These methods aim to find only a set of noncontractible loops, thus missing neck features like those on genus-0 shapes.

To evaluate a complete set of neck features, [10] proposed to use tetrahedral meshes to identify 3D choke points represented as triangle faces and turn them into homologous "choking loops" on surfaces. Their method provided a definition of choking loops around neck features as well as a measurement of their significance based on persistent homology. The specific topological construction they use is the lower-star filtration of a Morse function that evaluates the distance of a mesh vertex to the boundary surface. During the filtration, the surface thickness grows until the inside tunnel is blocked at some choke points. The importance of these points is measured by, roughly speaking, the size of the separate chambers created by these blockages. While a number of software packages exist for the calculation of the lower-star filtration [14], [15], [16], the construction is not efficient for large tet meshes. Our method skips this construction, although we also use volumetric distance fields to identify seed locations of neck features as in [10]. Our measurement criteria for neck features are more geometrically based. There are surface-based definitions of constriction loops [17] as closed nearly planar geodesics. While undoubtedly valuable for specific applications, such definitions are not as closely related to the topology of the volume as in [10].

While our detection procedure differs from [10] in that we use critical points of a Morse function instead of discrete critical simplices in the lower-star filtration, they are, in fact, linked, as pointed out in [18]. In addition, several studies investigated the role of Morse theory in shape analysis to explore the topological features of discretized spaces [18], [19]. Several other applications in shape segmentation and graph reconstruction used Morse complexes (see, e.g., [20], [21]).

The final output of shortest loops on surfaces in many methods (including [22], [23], [24]) is restricted to shortest closed edge paths. Instead, we use the method proposed in [25] to efficiently compute the shortest path inside a triangle strip loop by updating the triangle strip iteratively. The method has a time complexity of $O(mk)$, where m is the number of vertices in the original loop, and k is the average number of edges the loop swept through during the shortening process. A recent alternative proposed in [26] computes exact geodesic paths by flipping edges to create a shorter path within their local neighborhood, which may run even faster. However, as we only have few geodesic loops to evaluate, this post-processing step does not impact the performance much. Another method proposed in [27] computes the shrinking loops by tracking the evolution of a diffusion from a single location on the surface.

3 Mathematical Background

This section briefly reviews a few relevant concepts in Morse theory, particularly the critical points of a Morse function. Our algorithm relies on particular types of critical points to locate neck features of 3D shapes. Then we describe volumetric Laplacian smoothing, which provides a measurement of the importance of these critical points and a denoising preprocessing step. Implementation details of these methods can be found in Sec. 4.

3.1 Morse Function

Morse functions form a dense subset of smooth functions defined on a smooth manifold M. Such functions can often be used to analyze the topological information of a manifold and construct auxiliary structures. Specifically, a function $f : M \rightarrow R$ is a Morse function if and only if all critical points of f are non-degenerate. Its discrete analogy defined on simplicial meshes (triangle meshes in 2D and tetrahedral meshes in 3D) are simply piecewise-linear (PL) functions that evaluate to different values on different vertices, which can always be achieved by symbolic perturbation [28].

The critical points of Morse functions, along with the stable and unstable manifolds [28] of the gradient of Morse functions, can reveal essential structures. One such structure is the Morse-Smale complex, useful in, e.g., quadrangulation [29]. An analog structure, called the quasi Morse-Smale complex, can even be computed for piecewise-linear (PL) functions defined on simplicial 3D meshes [30]. In our algorithm, we take a shortcut to avoid the direct calculation of these structures, but leverage the existence of such structures to directly use critical points as candidates for the neck features that lead to corresponding neck loops.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3211781

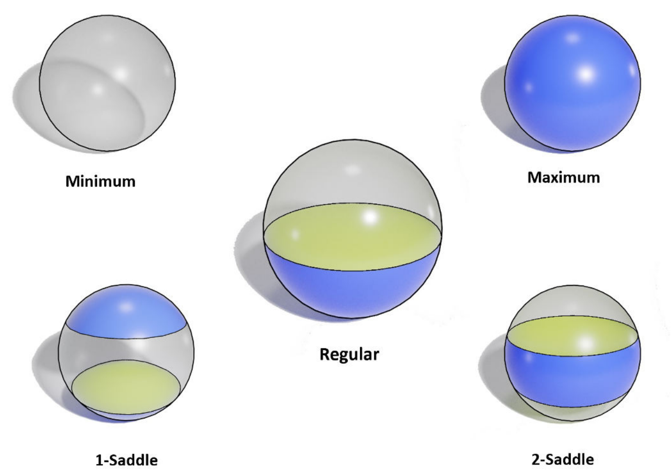


Fig. 2: Different types of critical points. The middle sphere represents a noncritical point with one connected upper part (grey) and one connected lower part (blue). The sufficiently small sphere around a critical point always indicates its type through the component numbers of upper and lower parts; a point with no connected lower part (minimum), a point with no connected upper part (maximum), a point with 1 connected upper part and 2 connected lower parts (1-saddle), and a point with 2 connected upper parts and 1 connected lower part (2-saddle)

3.2 Critical Points

For a smooth d-manifold M, the critical points of a smooth function $f : M \to \mathbb{R}$ are the locations where its differential vanishes. In a local coordinate system $(x_1, x_2, ..., x_d)$, the condition for x to be a critical point of f is given by

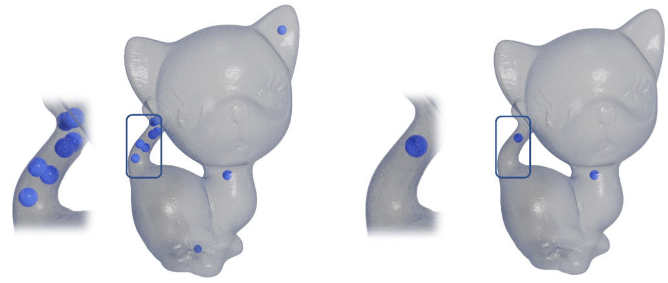
$$\frac{\partial f}{\partial x_1}(x) = \frac{\partial f}{\partial x_2}(x) = ... = \frac{\partial f}{\partial x_d}(x) = 0.$$

For a Morse function f, the Hessian of f at a critical point is nonsingular, i.e., $\det \nabla\nabla^T f \neq 0$. Thus the critical points can be classified by the signature of the Hessian. For a 3-manifold, there are four types of critical points, namely minima with index 0 (Hessian signature $[+,+,+]$), saddles with index 1 ($[+,+,-]$) or 2 ($[+,-,-]$), and maxima with index 3 ($[-,-,-]$).

Each critical point can also be equivalently classified by the topology of a sub-level set around it, as illustrated in Fig. 2. The sphere around each type of point indicates the boundary of a small neighborhood, with the sub-level set shaded in blue. A critical point with no connected lower part is a minimum, a critical point with no connected higher part is a maximum, a critical point with 1 connected upper part and 2 connected lower parts is a 1-saddle, and a critical point with 2 connected upper parts and 1 connected lower part is a 2-saddle. In contrast, any noncritical point has one connected upper part and one connected lower part [28]. We will use this formulation to detect and classify critical points in our implementation.

3.3 Laplacian Smoothing

Laplacian smoothing is a typical tool for denoising functions defined on polygonal meshes. When the function to smooth is the vertex positions of the polygonal mesh itself, it serves as a denoising tool for the surface. We use the volumetric cotangent-based Laplacian in [31]. Note that this discretization of the Laplacian corresponds to $l = -\Delta f$, where $\Delta = \nabla^2$ is the continuous Laplacian.



(a) unsmoothed (b) smoothed

Fig. 3: Singularities before and after smoothing. The blue points are the detected 2-saddles of the Kitten model.

Instead of constructing a filtration of cell complexes as in persistent homology, we employ the notion of persistence under smoothing similar to the Laplace-Beltrami flow-based filtration in [32]. Such a filtering process mimics physical diffusion, described by the heat equation with fixed temperature on the boundary. As heat moves from locations with higher temperature to those with lower temperature, temperature changes at a rate proportional to its Laplacian:

$$\frac{\partial T(x,t)}{\partial t} = \lambda \Delta T(x,t),$$

where λ is the diffusivity. With temporal discretization through the implicit Euler method, we solve the heat equation in an iterative way as

$$T(x, t+h) \approx T(x,t) + \lambda h \Delta T(x, t+h),$$

where h is the time step size. Together with the spatial discretization of $T(x,t)$, the implicit Euler step results in a sparse symmetric linear system. Applying Laplacian smoothing to the distance function defined on all internal vertices of a tetrahedral mesh with a homogeneous Dirichlet boundary is essential in our method. This smoothed distance function provides a measurement of singularity importance and a valid Morse function for the straightforward detection of singularities and the evolution of singularities over the time period in nh, where n is the iteration number. Fig. 3 shows the detected 2-saddle points of the kitten model before and after applying the Laplacian smoothing process.

4 Neck Feature Extraction

Given a closed surface mesh, our algorithm produces a set of loops around 3D neck-like regions for the volume inside or outside the surface. Following the definition in [10], we compute the bottleneck region as the boundary loop of a small surface membrane that alters the connectivity of the inside (or outside) volume. These geometry-aware topologically defined loops can be computed based on seed faces that eliminate a first homology generator (loop) or create a second homology generator when the surface is offset towards the interior of the volume. Based on the

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3211781

observation that the volume between the original surface and its offset corresponds to a lower-star filtration of distance function from the surface, the seed face, in either case, must be incident to a 2-saddle point. Thus, we may bypass the potentially costly 3D persistent homology evaluation without missing a single candidate by examining all 2-saddle points. We select the significant features among the candidates by examining how the 2-saddles evolve under Laplacian smoothing instead of per persistence in the lower-star filtration. After all, the diameter of the resulting loop provides a more straightforward criterion on how the region is neck-like than the difference in distance function values of paired simplices in the lower-star filtration. Smoothing of the distance function also more effectively reduces duplicated detected of neck regions.

In theory, our procedure based on 2-saddle detection guarantees to produce a superset of the seed faces before the distance function is smoothed. Moreover, according to the theory of the Laplace-Beltrami flow-based filtration [32], the smoothing time provides a persistence measure comparable to that of a Vietoris-Rips complex. Thus the set of 2-saddles surviving the smoothing process is close to the seed faces detected above a persistence threshold in [10].

Overview. Our pipeline starts with a preprocessing that constructs a tetrahedral mesh for the volume inside (or outside) the surface mesh and evaluates the distance function on interior vertices. Using the distance function as a Morse function, we evaluate all the critical points with their types. Laplacian smoothing can be leveraged when determining critical points. In a typical calculation, we perform a few rounds of Laplacian smoothing during a preprocessing step, and then track the evolution of the critical points to select significant features among 2-saddles. Finally, we extract the surface loops surrounding all seed 2-saddle points. Those initial loops are shortened into the final output surface loops.

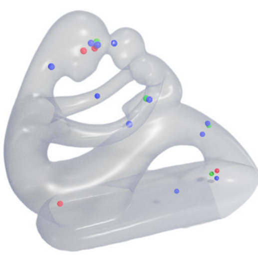
4.1 Critical point identification

To use the distance function of points to the surface as our Morse function, we follow [10] in applying fast marching to compute the initial per-vertex values. We add a numerical perturbation if internal vertices share the same floating-point values as one of its neighbors. For each internal vertex v_i with function value $f(v_i)$, its one-ring neighbors forms a topological sphere around the vertex. Next, all vertices are examined to detect all singularities and classify them. The procedure can be done in parallel since only the one-ring is necessary to classify each vertex.

We follow the usual discrete singularity type definitions [28]. We denote by N_i^{low} the lower link of v_i, i.e., the set of all adjacent vertices with function values less than f_i. Similarly, N_i^{up} is the upper link, set of all the adjacent vertices with function values higher than f_i. The singularity type of each vertex can be defined by the Betti numbers of the lower link. We classify the vertex based on the following equivalent discrete definitions, which can be performed based on the numbers of connected components for both the lower and upper links:

type of v_i	#component of N_i^{up}	#component of N_i^{low}
maximum	0	1
minimum	1	0
noncritical	1	1
1-saddle	1	2
2-saddle	2	1
monkey-saddle	otherwise	

Note that the minima of the distance function can only be on the surface, so in practice, we only need to evaluate the three types of critical points for interior vertices. Common examples of internal critical points are shown in Fig. 4. While other types of monkey-saddles exist, they rarely show up for distance functions. In fact, all monkey-saddles will generally disappear after smoothing. As a side note, while monkey-saddles do not influence the bottleneck calculation, since we use only 2-saddles, it is possible to eliminate them through smoothing combined with local mesh refinement.



The inset figure shows all detected critical points for the Fertility model. The 1-saddles are colored green, 2-saddles blue, and maxima red. Since two eigenvalues of the Hessian located at the 2-saddle are negative, the distance to the surface increases only along one of the three eigenvectors. In the discretized mesh, it indicates a local minimum along a "skeleton" tangential to that eigenvector, and corresponds to a bottleneck, i.e., a narrowing of the volume around it. Intuitively, we may interpret the neck loop as the extension of the ring-like lower link of the 2-saddle to the surface, forming a membrane separating the internal volume.

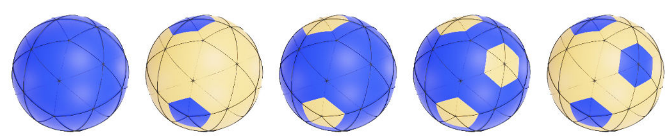


Fig. 4: Discrete critical points. The triangulated sphere represents the one-ring neighbor around an internal vertex, with the lower link colored blue and the upper link colored yellow. Based on the connectivity of the two parts, all internal critical points are classified as a 1-saddle, 2-saddle, monkey-saddle, or maximum.

While the set of 2-saddles of the inside volume (or the volume between the surface and a bounding sphere) includes all the seed locations for handle-like (or tunnel-like) neck loops, they may potentially contain some points due to noise or locations corresponding to large loops. Since we skip the direct calculation for the 3D persistent homology, we found in our experiments that smoothing the distance function and analyzing the spatial relation among neighboring singularities is sufficient for our task of selecting only the significant structures.

4.2 Smoothing-based critical point selection

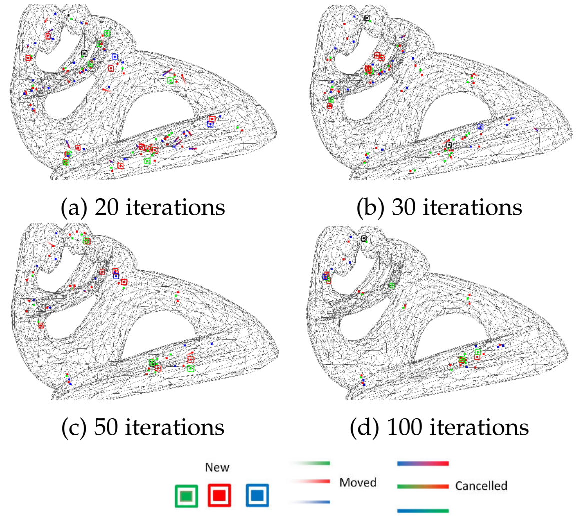
We start with a tet mesh M and rescale it to a unit bounding box to avoid scale dependence. Assuming the discrete Laplacian is assembled into a matrix L, we then set the rows and columns corresponding to boundary vertices to 0, except the diagonal entry is set to 1, to enforce the boundary condition $f|_{\partial M} = 0$. Denote the discrete representation of f as a column vector F, then solve for $(\mathrm{M}+\lambda h\mathrm{L})F^{t+h} = \mathrm{M}F^{t}$ in each iteration. Here M is the mass matrix, λ is the diffusivity, and h is a time step. Setting λh to a constant (0.01 in our experiments) allows us to use the iteration count to measure the amount of smoothing performed. Higher-order Laplacian operators (e.g., [33]) may be used, but lead to similar final results. Note that we are not smoothing the mesh itself, so the sparse matrix remains the same, and Cholesky pre-factorization can be used to speed up the repeated smoothing.

In practice, we found the initial distance function from fast marching often noisy. Since we intend to ignore small-scale bottleneck structures, we perform a few (10 in our experiments) smoothing iterations before any critical point evaluation. We may then perform the critical point calculation algorithm after any number of smoothing iterations. As a sanity check based on Morse theory, we compare the Euler characteristic $\chi = \beta_0 - \beta_1 + \beta_2 - \beta_3$, the alternating sum of Betti numbers β_i, with $\chi_{\partial M} - n_1 + n_2 - n_3$, where $\chi_{\partial M} = 2 - 2g$ is the Euler characteristic of the boundary surface with genus g, n_1 is the number of 1-saddles, n_2 is the number of 2-saddles, and n_3 is the number of maxima. In the rare cases of monkey-saddles, their contribution to the Euler characteristic can be evaluated by comparing the Euler characteristic of its lower-star and that of its lower-link.

Smoothing the function values of all vertices helps eliminate the transient 2-saddles and keep the persistent ones, as shown in Figure 3.

Local approximation-based selection. One heuristic rule we found effective is to perform a local quadratic approximation of the one-ring distance function values, and verify that the Hessian of the approximation has the correct signature of a 2-saddle, i.e., two negative eigenvalues, and one positive eigenvalue. Among all the detected 2-saddle points in any smoothing level, we may use this local heuristic to rule out some 2-saddles that do not correspond to reasonable bottleneck structures. Fig. 1 shows an example of excluded points in the Fertility model. All points in the left figure are classified as discrete 2-saddles, but only the points in the center figure will be further processed. Moreover, the positive eigenvector is reused in our initial loop construction step, as it represents the normal of the cutting plane.

Evolution-based selection. If we choose to compute the critical points after each iteration of smoothing, we can track their continuous changes. The majority of critical points will remain at the same vertex or move to a nearby vertex. With any k-nearest neighbor algorithm, we can track down these changes. In some cases, new critical points are far from any critical points in the previous step, while other existing pairs of nearby critical points of index k and $(k+1)$ cancel out. Fig 5 shows the evolution of the different critical points at increasing smoothing levels. We demonstrate the noisy nature of the initial distance function values on a tet mesh by skipping the initial smoothing. Normally, the number of critical points after the initial smoothing is low, and can be tracked efficiently with negligible time cost compared to the smoothing step. We use blue dots to denote maxima, red dots for 2-saddles, green dots for 1-saddles, and black dots for the rare cases of monkey-saddles. Squares encase new critical points, faded arrows represent moving points, and blended colored lines denote pairwise cancellation. With tracking, we can use the number of smoothing iterations between the first appearance of any 2-saddle until its cancellation to measure persistence under smoothing for seed point selection.



(a) 20 iterations (b) 30 iterations
(c) 50 iterations (d) 100 iterations
(e)

Fig. 5: Critical points evolution. (a) to (d) show the detected critical points of the downsampled Fertility model at 20, 30, 50, and 100 smoothing iterations, resp.

4.3 Surface loops

Any 2-saddle point that passes both selection criteria has the eigenvector associated with the positive eigenvalue of the local Hessian matrix stored. Using this eigenvector as the normal and the seed point, we can compute a cutting plane that intersects the boundary surface as shown in Fig. 1. It may result in multiple intersection loops, and we keep only the loop with the seed 2-saddle point inside it. All three example cutting planes in Fig. 1 result in multiple intersection loops. In the leftmost example, the cutting plane of a 2-saddle is located in the base, and the red loop is chosen as the initial loop. Another cutting plane for a seed point located in the model's right arm is shown in the middle. The loop around the seed in the neck is shown on the right. In our tests, using the eigenvector as the plane normal produces the best guess for the initial loop. We may optionally use the lower link variation to create multiple normals for multiple candidate cutting planes and pick the shortest initial loop.

Following [25], we perform a local shortest loop evaluation that moves continuously on the triangle surface mesh, which improves the geometric shape of the computed surface loops. The final shortest loops are discrete geodesic

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3211781

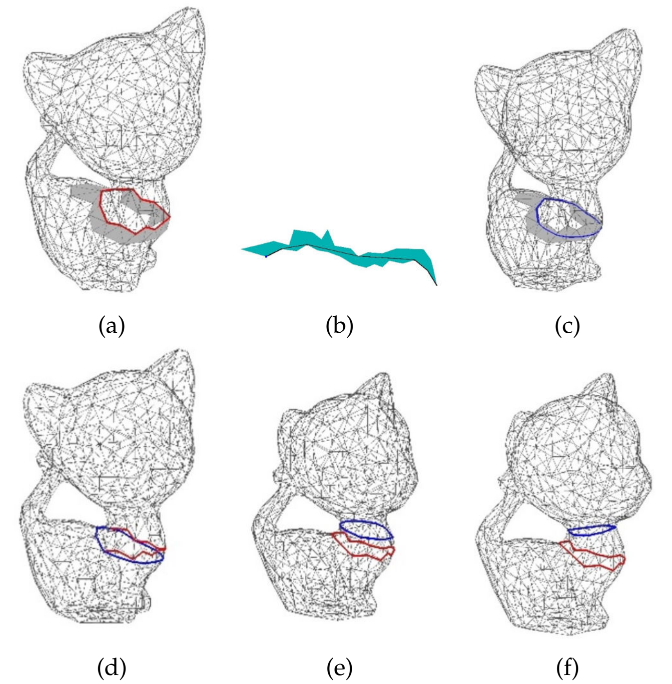


Fig. 6: Computing the shortest loop from the initial loop as in [25]. (a) is the initial loop with the attached triangles (b) is the funnel, and (c) is the shorter loop. The second row shows an example of finding the shortest loop (f) from the initial one , the blue polyline (the shorter loop from a) (d), with an intermediate loop (e).

loops, which are locally straight and only go through vertices with negative discrete Gaussian curvatures. In Fig. 6, the initial loop is shown as the red polyline around the Kitten's neck, considered as part of the triangle strip in grey. The strip of faces on the surface can be unfolded after cutting along one edge, and the shortest path is computed in the next subfigure as the blue polyline. The blue polyline is restricted by a few vertices, which can be used to update the grey strip for further relaxation of the loop. In the bottom row of the figure, the evolution of the initial loop, through an intermediate stage, to the final loop is shown. Fig. 7 shows more examples of the initial and final loops on the Kitten (a sparsely sampled model) and the (densely sampled) Figure-Eight model.

5 Results and Discussion

The resulting surface loops of our algorithm represent neck features akin to the choking loops in [10], which are geometry-aware topological features. These loops can populate a complete set of necks like handles, as well as those narrow regions of the outside space, i.e., tunnels. This differs from finding a shortest set of loops that span the first homology of the boundary surface, as this set may derive from second homology generators of the lower-star filtration of the distance function. Roughly speaking, they are the loops that bound membranes that separate the internal space into lower genus or more components. For instance, genus-0 models can have multiple handle-type neck features like the Bunny and the Toy model in Fig. 8, despite their sphere-like surface geometry.

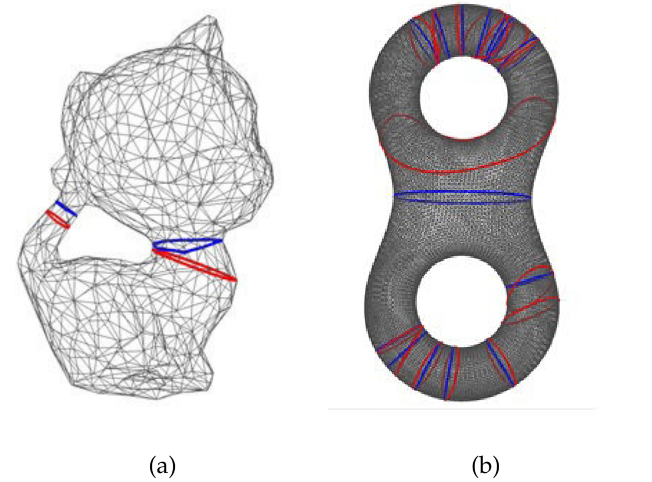


Fig. 7: Initial and final loops; red(initial) , blue(final)

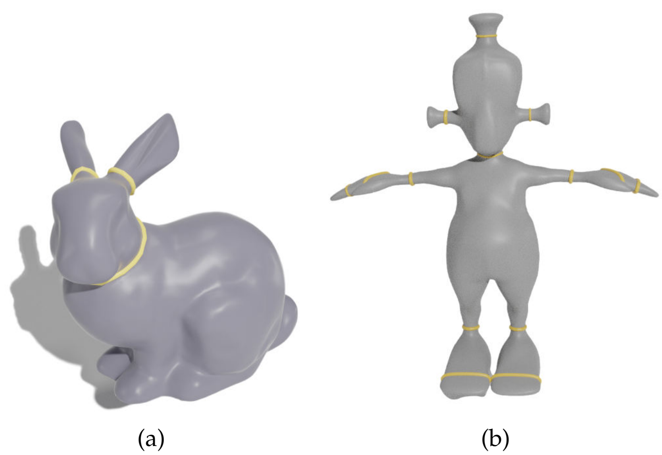


Fig. 8: Neck loops of genus-0 models

Fig. 1 shows that for a nonzero genus model, the number of handle-type neck features can be far more than twice its genus. Here, the Fertility model is genus-4, but there are more than double the number of neck features.

For some high genus models, such as the genus-31 Buckyball, the difference is even more significant; our algorithm can generate all 96 handle-like loops, each of which is a valid candidate for 1st homology generator of the surface, as shown in Fig. 9. It also illustrates a similar structure for a more complicated Protein model.

Fig. 10 shows the tunnel-like neck features computed by our algorithm, which uses the volumetric mesh bounded by the given surface and a bounding sphere. The outside neck loops of the Kitten, Buckyball, Botijo, and EMD models are shown on the surfaces, with some of the surfaces rendered transparent to show the internal structure. Such structures can potentially help evaluate the docking of drug molecules on protein surfaces and the analysis of ion channels. For example, in the EMD model, there are four wide tunnels of the exterior surface, and another 2 tunnels of the double torus shape inside. The 6 tunnels are all detected in addition to another small one, and two narrow passages.

We first aim to generate a comprehensive set of neck features automatically. Then we offer multiple heuristic

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3211781

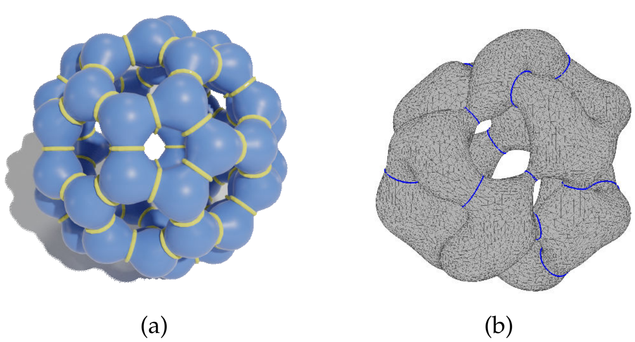


Fig. 9: Neck structures of high-genus models

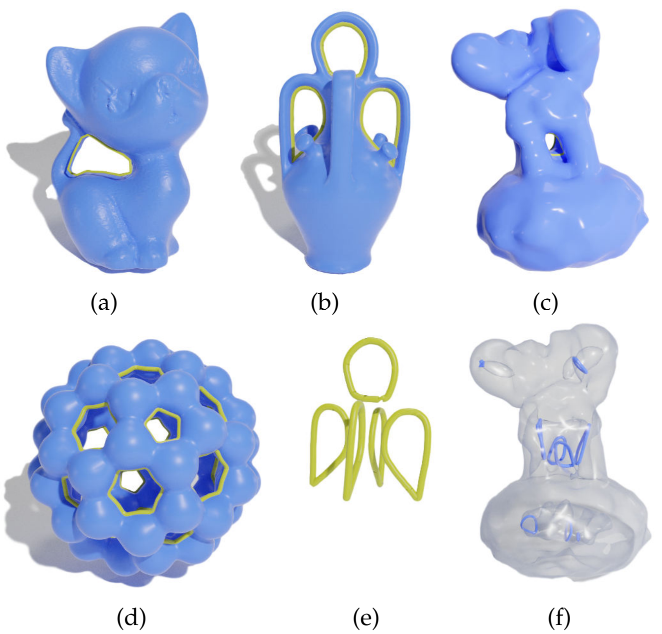


Fig. 10: Tunnels. (a) one tunnel in the Kitten model (d) 32 tunnels in the Buckyball model (b,e) tunnels in the Botijo model (c,f) exterior and interior tunnels of the EMD model

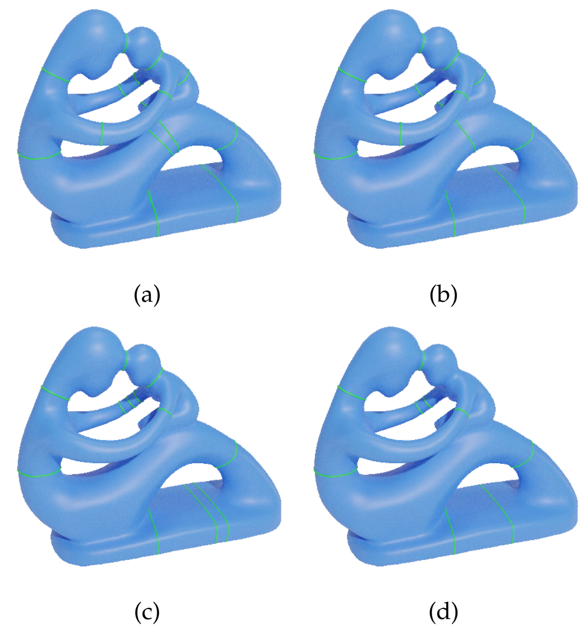


Fig. 11: Detected loops at different smoothing levels (a) 5 iterations (b) 10 iterations (c) 20 iterations (d) 50 iterations

rules to select important ones automatically. The more important selection rule is the smoothing level; we detect different numbers of critical points with varying amounts of smoothing. We offer a default value that works well for all the models shown in the paper, but the user can easily change the smoothing level and choose to include vanished 2-saddles if they persist long enough during the smoothing process. For example, Fig. 11 shows the loops at different smoothing levels of the Fertility model. With 5 iterations of Laplacian smoothing, we can get almost all expected neck loops; by applying more smoothing iterations, some critical points vanish, and their corresponding loops are eliminated from the set of significant neck features.

Comparison to Feng and Tong's method [10]. The key to the efficiency of our algorithm is skipping the persistent homology computation, which was the bottleneck in their system. Our smoothing algorithm runs for a fixed number of iterations, each of which solves a Poisson equation in linear time with multigrid methods. In contrast, the persistent homology typically runs at $O(n^3)$ or $O(n^\omega)$, where n is the total number of simplices, and ω is the matrix multiplication exponent $\omega < 2.4$, according to [34]. We tested our method on many models used in [10] , and the resulting loops on Fertility, Bunny, Toy, Buckyball, Kitten, and two proteins are shown in the respective result figures. Our method and the method in [10] depend on parameters that enable the user to choose the loop feature significance, including smoothing level in our method, and persistence in their method. We show that similar loops could be generated for the same model with comparable parameter choices. More results showing the robustness of our method can be found in the supplemental material. All experiments show that our method is comparable to their method in results, but with a simpler structure and more efficient computation.

In Table 1, we list the computation time of Feng and Tong's method [10] and our method applied for different models at different scales. We tested both methods on Fertility, Kitten, Botijo, Bunny and Buckyball with numbers of interior vertices 21k, 75k, 256k, 90k, and 163k, resp. All results are in milliseconds. The time is broken down for the different parts of both methods. For Feng and Tong's method, the results in the table show the computation time of prepossessing, persistent homology, computing seed faces, and contracting surface loops. For our method, the computation time of prepossessing, Laplacian smoothing, and critical point detection are listed. For both methods, the input meshes are identical and tetrahedralized using TetGen [35] with parameter "pfq1.2".

Figure 12 shows a performance comparison between the two methods on the same model with different internal seed points. We applied both methods with varying densities of sampling of the Botijo model (Figure 10 (b)), with 113k, 144k, 174k, 257k, and 341k internal vertices, resp. The graph shows that our method is scalable in terms of mesh size.

Figure. 13 compares the initial loops generated by Feng

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3211781

Feng and Tong's method [10]		Our Method	
Fertility #v21908, #t90973, #F195923			
param	del=10%, dist= 50%	param	sm=20 iterations
preprocessing	7928	preprocessing	5249
persistent H surf	1576	smoothing	1672
persistent H inside	52927	computing 2-saddles	1349
seed faces	46675		
loop computation	2970	loop computation	821
total time (millisecond)	65,447	total time (millisecond)	8,270
kitten #v75500, #T360282, #F756360			
param	del=20%, dist= 50%	param	sm=20 iterations
preprocessing	24693	preprocessing	24645
persistent H surf	3410	smoothing	8455
persistent H inside	154146	computing 2-saddles	5764
seed faces	138		
loop computation	4881	loop computation	267
total time (millisecond)	187,268	total time (millisecond)	39,131
Botijo #v265523, #t1224316, #F2586212			
param	del=20%, dist= 50%	param	sm=20 iterations
preprocessing	151993	preprocessing	146506
persistent H surf	19972	smoothing	22413
persistent H inside	2049428	computing 2-saddles	19351
seed faces	550		
loop computation	55154	loop computation	1093
total time (millisecond)	2,277,097	total time (millisecond)	189,363
Bunny #v90819, #t431208, #F 906022			
param	del=10%, dist= 50%	param	sm=20 iterations
preprocessing	30582	preprocessing	31307
persistent H surf	4092	smoothing	10187
persistent H inside	309762	computing 2-saddles	6865
seed faces	552		
loop computation	253728	loop computation	3787
total time (millisecond)	598716	total time (millisecond)	52146
Buckyball #v90819, #t431208, #F 906022			
param	del=20%, dist= 50%	param	sm=20 iterations
preprocessing	72288	preprocessing	70681
persistent H surf	10087	smoothing	19336
persistent H inside	318090	computing 2-saddles	12122
seed faces	321		
loop computation	160107	loop computation	6542
total time (millisecond)	560893	total time (millisecond)	108681

TABLE 1: Performance statistics and comparison with Feng and Tong [10] (all time measurements in milliseconds, with Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz).

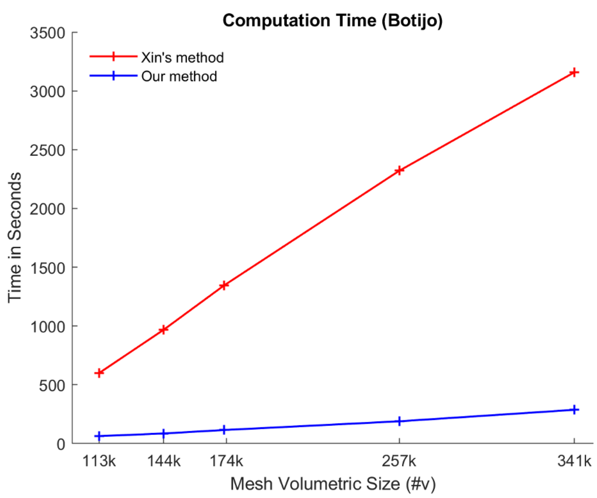


Fig. 12: Computation time of two methods for different scales of the Botijo model

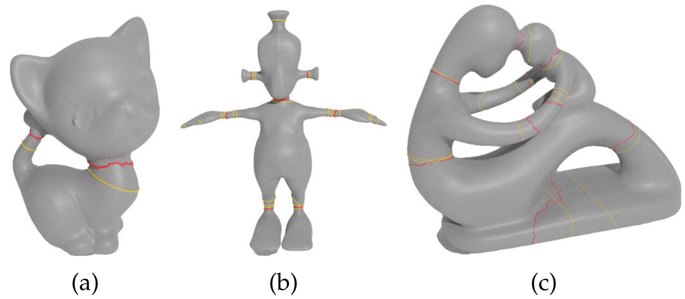


Fig. 13: Initial loops between Feng and Tong's method [10] (in red) and our method (in yellow) for the (a) Kitten (b) Toy, and (c) Fertility models

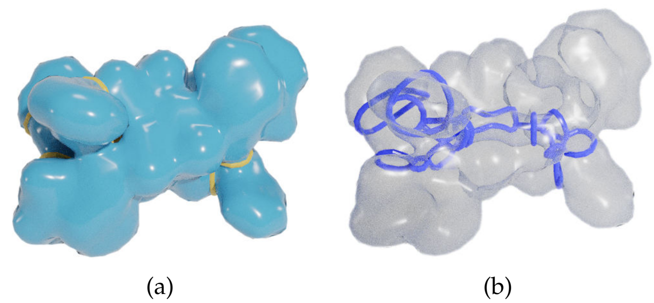


Fig. 14: 1mag results loops

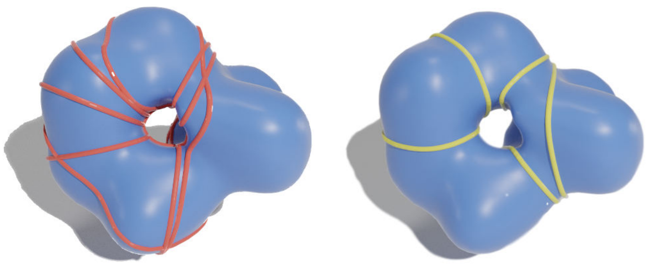
and Tong's method [10] (shown in red) and those generated by our method (in yellow). The yellow loops are generally smoother than the red loops (formed only by edges). While the 2-saddles may drift during the smoothing, they stay close to the corresponding seed faces. Thus, the shrunk red loops form a superset of the shrunk yellow loops.

Computing the shortest loop on the surface as in [25] can be sensitive to the presence of negative Gaussian curvature points, which can trap the movement of the loop at a local minimum. We offer the option of smoothing the surface mesh before our procedure to produce loops closer to actual neck locations. This would also allow duplicated nearby loops to slide to the same neck locations (e.g., Fig. 15). We offer the option to project the loop back onto the original surface, which is always within a small chamfer distance from the smoothed surface.

An additional speedup in the computation of the initial loops can be achieved by using a simplified version of the original mesh. As we search for significant features, the results are always similar. With the mesh simplification, all the procedures involved run faster. For instance, the cutting plane will cut the surface in fewer edges, and shortening the loop will also be more efficient than denser surface meshes. The quality of the loop is no worse than directly running on a dense mesh, as we map the loops back to the original surface. Mapping the loops back to the original mesh is performed by cutting the surface of the original mesh by the cutting planes used on the simplified mesh for creating these loops. Fig. 16 shows an example of using the simplified mesh for computing the surface loops before mapping them back. Some loops in Figure 16(a) are merged in 16(c) as they correspond to the same neck features of the shape (left arm and base).

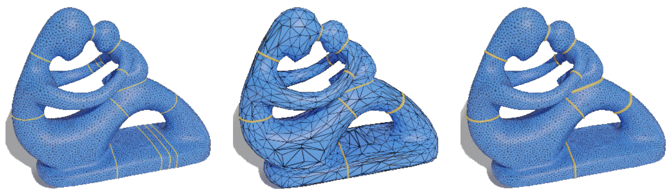
Fig. 17 shows the final neck loops for some additional 0-genus models. Neck loops computed on around 40 models with various genus can be found in the **supplemental material**.

While we investigate all the internal vertices for critical point classification in various smoothing levels, monkey-saddles do appear in our computation. However, our experiments show that those points quickly vanish after applying adequate smoothing steps. Moreover, such monkey-saddles can be easily eliminated with local refinements of the tet



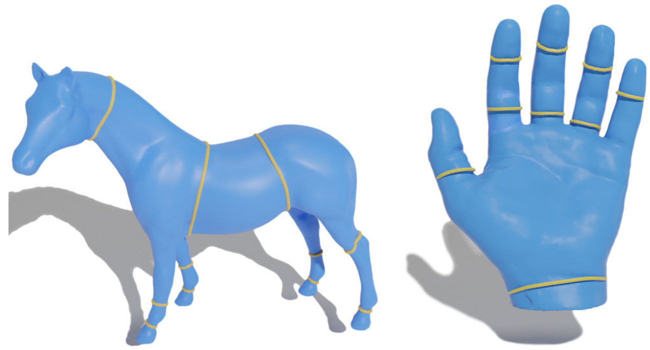
(a) (b)

Fig. 15: (a) Initial loops(red) of a protein model (b) Shortest final loops(yellow)



(a) (b) (c)

Fig. 16: Computing surface loops on simplified meshes. The initial loops are computed on the original mesh(a), while in (b), the loops are computed on the simplified mesh and mapped back to the original mesh (c).

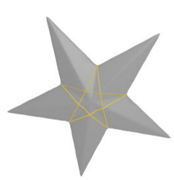


(a) (b)

Fig. 17: More results of genus-0 models

mesh and splitting the monkey-saddle into a few non-monkey-saddle critical points.

Limitations. One limitation of our algorithm is that using cutting planes might not work for some complicated examples. For example, the inset figure shows the computed handles of the torus knots model. Still, the tunnel of this specific case would be a Seifert surface, which cannot be computed with the plane-surface intersection. However, arguably this tunnel does not correspond to a neck feature.



Another type of feature our algorithm is not intended to handle is the closed geodesic loops as shown in the star model, which are nevertheless also candidates for grasping locations of the object. Such loops can be computed by surface-based methods like [17].

6 Conclusion and Future Work

In this work, we introduce a fast method to compute all neck feature loops using both the topological and geometrical properties of the shape. Our key observation is that all neck structures induce 2-saddles in the distance function. One way to define the significance of these features is through persistent homology. However, we found in our experiments that direct smoothing of the distance function, mimicking the heat diffusion procedure with a fixed temperature at boundary method, led to similar but often more reasonable candidate center locations for neck features. A possible explanation is the global and symmetric nature of the smoothing procedure than the elder rule in persistence calculation. Given that the neck features are where the shape can be "lassoed", we loosely fit an initial loop by a cutting plane constructed based on the local Hessian eigenvectors, and tighten it into a geodesic loop to represent the precise location where the lasso will end up.

In addition to our basic procedure, we offer additional selection rules based on the loop size and distance to nearby loops to facilitate the automatic generation of a compact set of neck features. If the application does not require small features, we can further speed up the entire pipeline by using a simplified surface mesh and/or simplified tet mesh to find the neck feature loops before mapping them back onto the original surface. The mesh simplification not only reduces the element count, but also reduces the number of smoothing iterations.

For future work, we wish to investigate further theoretical analysis on the diffusion-induced changes in Morse functions, as well as other types of Morse functions like diffusion distance. In certain scenarios, homology generator loops that form knots may also be extracted as features, which would require an extension from our cutting plane strategy to a connectivity-based approach. Fast calculation of membranes bounded by the neck feature loops can be a relevant computational tool. We also intend to explore alternative ways of defining neck structure through distance functions on skeleton structures such as the medial axes. Applications of neck features in segmentation, meshing, and machine learning can also be explored. A related geometry-aware topological feature to explore is the short line segments through persistent 1-saddles, as they correspond to thin layers that are easy to punch through (interior) or close surface locations with opposite normals that are nearly touching (exterior). In terms of performance improvements, while our work is performed on tet meshes, it can be extended to implicit representations on a Cartesian grid. For that purpose, an efficient smoothing algorithm of the signed distance function with the 0-th level set fixed is required.

Acknowledgments

The authors would like to thank Rundong Zhao for his help in visualization, and Emily Ribando-Gros for proofreading the paper. The authors also thank the anonymous referees for their valuable comments and helpful suggestions. The work was supported in part by NSF grant IIS-1900473.

References

[1] A. Varava, D. Kragic, and F. T. Pokorny, "Caging grasps of rigid and partially deformable 3-d objects with double fork and neck features," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1479–1497, 2016.

[2] E. Zhang, K. Mischaikow, and G. Turk, "Feature-based surface parameterization and texture mapping," *ACM Transactions on Graphics (TOG)*, vol. 24, no. 1, pp. 1–27, 2005.

[3] D. Letscher and J. Fritts, "Image segmentation using topological persistence," in *International Conference on Computer Analysis of Images and Patterns*. Springer, 2007, pp. 587–595.

[4] 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," *Computer-Aided Design*, vol. 43, no. 11, pp. 1457–1467, 2011.

[5] D. Zeng, E. Chambers, D. Letscher, and T. Ju, "To cut or to fill: a global optimization approach to topological simplification," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–18, 2020.

[6] J. Liu, S. Xin, X. Gao, K. Gao, K. Xu, B. Chen, and C. Tu, "Computational object-wrapping rope nets," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 1, pp. 1–16, 2021.

[7] J.-M. Favreau and V. Barra, "Tiling surfaces with cylinders using n-loops," *Computers & Graphics*, vol. 35, no. 1, pp. 35–42, 2011.

[8] É. C. D. Verdière and F. Lazarus, "Optimal pants decompositions and shortest homotopic cycles on an orientable surface," *Journal of the ACM (JACM)*, vol. 54, no. 4, pp. 18–es, 2007.

[9] M. Hajij, T. Dey, and X. Li, "Segmenting a surface mesh into pants using morse theory," *Graphical Models*, vol. 88, pp. 12–21, 2016.

[10] X. Feng and Y. Tong, "Choking loops on surfaces," *IEEE transactions on visualization and computer graphics*, vol. 19, no. 8, pp. 1298–1306, 2013.

[11] T. K. Dey, K. Li, J. Sun, and D. Cohen-Steiner, "Computing geometry-aware handle and tunnel loops in 3d models," in *ACM SIGGRAPH 2008 papers*, 2008, pp. 1–9.

[12] T. K. Dey, F. Fan, and Y. Wang, "An efficient computation of handle and tunnel loops via reeb graphs," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, pp. 1–10, 2013.

[13] T. K. Dey, A. N. Hirani, and B. Krishnamoorthy, "Optimal homologous cycles, total unimodularity, and linear programming," *SIAM Journal on Computing*, vol. 40, no. 4, pp. 1026–1044, 2011.

[14] D. Morozov, "Dionysus Software," *Retrieved December*, vol. 24, p. 2018, 2012.

[15] U. Bauer, M. Kerber, and J. Reininghaus, "Dipha (a distributed persistent homology algorithm)," *Software available at https://github.com/DIPHA/dipha*, 2014.

[16] B. T. Fasy, J. Kim, F. Lecci, C. Maria, D. L. Millman, and M. J. Kim, "Package 'TDA'," 2019.

[17] F. Hétroy, "Constriction computation using surface curvature," in *Eurographics (short paper)*, 2005, pp. 1–4.

[18] U. Fugacci, C. Landi, and H. Varlı, "Critical sets of pl and discrete morse theory: A correspondence," *Computers & Graphics*, vol. 90, pp. 43–50, 2020.

[19] X. Ni, M. Garland, and J. C. Hart, "Fair morse functions for extracting the topological structure of a surface mesh," *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3, pp. 613–622, 2004.

[20] L. De Floriani, U. Fugacci, F. Iuricich, and P. Magillo, "Morse complexes for shape segmentation and homological analysis: discrete models and algorithms," in *Computer Graphics Forum*, vol. 34, no. 2. Wiley Online Library, 2015, pp. 761–785.

[21] T. K. Dey, J. Wang, and Y. Wang, "Graph reconstruction by discrete morse theory," *arXiv preprint arXiv:1803.05093*, 2018.

[22] S. Cabello, É. Colin de Verdière, and F. Lazarus, "Finding shortest non-trivial cycles in directed graphs on surfaces," in *Proceedings of the twenty-sixth annual symposium on Computational geometry*, 2010, pp. 156–165.

[23] É. C. De Verdiére and F. Lazarus, "Optimal system of loops on an orientable surface," *Discrete & Computational Geometry*, vol. 33, no. 3, pp. 507–534, 2005.

[24] F. Lazarus, M. Pocchiola, G. Vegter, and A. Verroust, "Computing a canonical polygonal schema of an orientable triangulated surface," in *Proceedings of the seventeenth annual symposium on Computational geometry*, 2001, pp. 80–89.

[25] S.-Q. Xin, Y. He, and C.-W. Fu, "Efficiently computing exact geodesic loops within finite steps," *IEEE transactions on visualization and computer graphics*, vol. 18, no. 6, pp. 879–889, 2011.

[26] N. Sharp and K. Crane, "You can find geodesic paths in triangle meshes by just flipping edges," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–15, 2020.

[27] A. Weinrauch, H.-P. Seidel, D. Mlakar, M. Steinberger, and R. Zayer, "A variational loop shrinking analogy for handle and tunnel detection and reeb graph construction on surfaces," *arXiv preprint arXiv:2105.13168*, 2021.

[28] H. Edelsbrunner and J. Harer, *Computational topology: an introduction*. American Mathematical Soc., 2010.

[29] X. Fang, H. Bao, Y. Tong, M. Desbrun, and J. Huang, "Quadrangulation through morse-parameterization hybridization," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–15, 2018.

[30] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci, "Morse-smale complexes for piecewise linear 3-manifolds," in *Proceedings of the nineteenth annual symposium on Computational geometry*, 2003, pp. 361–370.

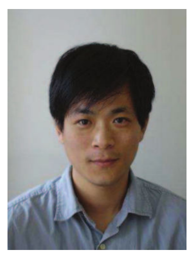
[31] Y. Tong, S. Lombeyda, A. N. Hirani, and M. Desbrun, "Discrete multiscale vector field decomposition," *ACM transactions on graphics (TOG)*, vol. 22, no. 3, pp. 445–452, 2003.

[32] B. Wang and G.-W. Wei, "Object-oriented persistent homology," *Journal of Computational Physics*, vol. 305, pp. 276–299, 2016.

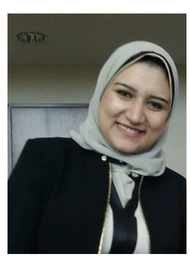
[33] R. Zhao, M. Desbrun, G.-W. Wei, and Y. Tong, "3d hodge decompositions of edge-and face-based vector fields," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–13, 2019.

[34] N. Milosavljević, D. Morozov, and P. Skraba, "Zigzag persistent homology in matrix multiplication time," in *Proceedings of the twenty-seventh Annual Symposium on Computational Geometry*, 2011, pp. 216–225.

[35] S. Hang, "Tetgen, a delaunay-based quality tetrahedral mesh generator," *ACM Trans. Math. Softw*, vol. 41, no. 2, p. 11, 2015.



Yiying Tong is a professor at Michigan State University. Prior to joining MSU, He worked as a postdoctoral scholar at Caltech. He received his Ph.D. degree from the University of Southern California in 2004. His research interests include discrete geometric modeling, physically-based simulation/animation, and discrete differential geometry/topology. He received the U.S. National Science Foundation (NSF) Career Award in 2010.



Hayam Abdelrahman is a Ph.D. student in the Department of Computer Science and Engineering at Michigan State University. She received her M.S. and B.S. degrees from Zagazig University, Egypt .Her research interests include computational topology, shape analysis, and computer graphics.