A Variational Loop Shrinking Analogy for Handle and Tunnel Detection and Reeb Graph Construction on Surfaces

A. Weinrauch[1,2] and D. Mlakar[1] and H.P. Seidel[2] and M. Steinberger[1] and R. Zayer[2]

[1]Graz University of Technology, Institute of Computer Graphics and Vision, Austria
[2]Max Planck Institute for Informatics Saarland Informatics Campus, Germany

Figure 1: *Computed handle and tunnel loops on the Dragon Tamer model.*

Abstract
The humble loop shrinking property *played a central role in the inception of modern topology but it has been eclipsed by more abstract algebraic formalisms. This is particularly true in the context of detecting relevant non-contractible loops on surfaces where elaborate homological and/or graph theoretical constructs are favored in algorithmic solutions. In this work, we devise a variational analogy to the loop shrinking property and show that it yields a simple, intuitive, yet powerful solution allowing a streamlined treatment of the problem of handle and tunnel loop detection. Our formalization tracks the evolution of a diffusion front randomly initiated on a single location on the surface. Capitalizing on a diffuse interface representation combined with a set of rules for concurrent front interactions, we develop a dynamic data structure for tracking the evolution on the surface encoded as a sparse matrix which serves for performing both diffusion numerics and loop detection and acts as the workhorse of our fully parallel implementation. The substantiated results suggest our approach outperforms state of the art and robustly copes with highly detailed geometric models. As a byproduct, our approach can be used to construct* Reeb graphs *by diffusion thus avoiding commonly encountered issues when using Morse functions.*

CCS Concepts
• *Computing methodologies* → *Shape analysis; Massively parallel algorithms;*

1. Introduction

Probably one of the most inspiring achievements of topology is the classification theorem for surfaces, which establishes equivalence classes based on the *Euler characteristic* and *orientability*. This development triggered a formidable effort focalized on the so called *Poincaré conjuncture* aiming at extending classification to higher dimensions and in particular to the 3−manifold setting

and captured the imagination of generations of mathematicians and the general public alike. Loosely speaking, the classification theorem for surfaces (2-manifolds) tells us that any orientable surface is equivalent to a sphere with a certain number of "handles" sewn onto it. In this respect, the surface can be constructed from a sphere by *topological surgery*, which can be understood as a set of cutting, stitching and deforming operations.

Although a sound theoretical foundation of the subject matter has been laid out in algebraic topology, see e.g. [Mun84], localization of geometrically meaningful handles on surfaces remains a highly challenging topic. Over the last decades, several algorithms for detecting non-trivial loops on general graphs have been proposed in computational geometry, see e.g. [Eri12] and the references therein, but most of these results remain theoretical and have not turned into efficient implementations. The topic of extracting such topological features is not only relevant as a theoretical pursuit but has fundamental applications in geometry processing, covering tasks such as mesh parameterization, mesh repair, and feature recognition, and spills beyond to fields such as biotechnology and bioinformatics, see e.g. [BCG*13; VG10], therefore there is need for efficient practical solutions.
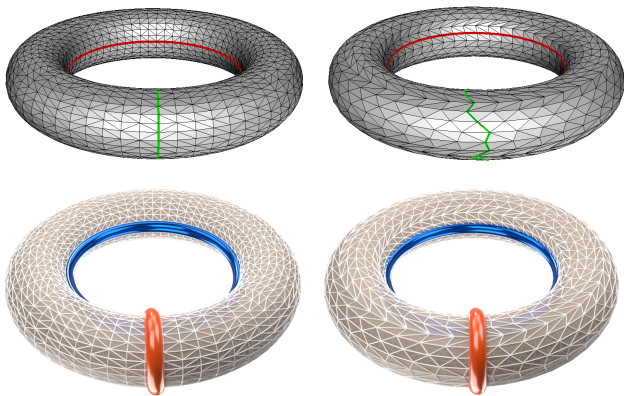

Figure 2: *Sensitivity of graph based methods such as [DFW13] to the underlying triangulation. A small lateral coordinates perturbation of the regular triangulation (top-left) induces a geometrically suboptimal handle loop (top-right). Operating on the piecewise linear manifold, our method (bottom) is robust to such perturbations.*

The pioneering efforts of Dey and coworkers [DLS07; DFW13] formalized the notion of geometrically meaningful topological features such as handle and tunnel loops and presented practical solutions using intermediate structures relying on *persistent homology* and *Reeb graphs*. Nonetheless, the cost of all intermediate constructs and their limitations impedes performance and intensifies memory usage. While many of the above discussed methods trade mesh geometry for a pure graph representation in order to provide some theoretical guarantees, these guarantees do not translate into robustness in practice. In fact, dependence on multiple intermediate constructs and optimizations leads to failures as can be inferred from Table 2 in the results section. Furthermore, the obtained loops are only as good as the underlying graph edges. As illustrated on the single torus case in Figure 2, the method of [DFW13] produces geometrically meaningful loops when a regular triangulation aligned with the principal curvature directions is used, however a simple lateral perturbation of vertex coordinates leads to a non-optimal handle loop, in the sense of not being aligned with the global principal directions and not being necessarily shortest geometrically. For applications such as robotics where affordance is key to interaction such loops are not of practical use. Therefore, there is need

for detection methods capable of working directly on the geometry of the object at hand and not its graph abstraction.

Our goal is to provide a simple, intuitive, yet efficient strategy for detection of handle and tunnel loops. To this end, we re-examine the problem in the light of the humble loop shrinking property which lies behind Poincaré's intuition and we seek to develop a variational analog to it which would allow us to evolve freely on the surface. In doing so, we avoid the difficulties encountered by existing methods in their efforts to marry homotopy classes, elaborate graph constructs, and practical geometric requirements.

Consider a person walking on a given surface while holding a sufficiently long thread from both ends, as illustrated in Figure 3. Had the person been on a sphere, it would be possible to spool it back. On the other hand, if the person is on a torus, it won't be possible to re-spool the thread because it passes through the 2-dimensional hole of the torus.
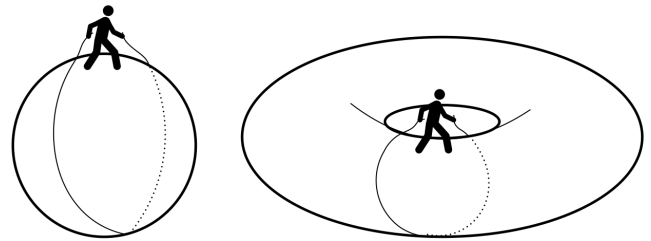

Figure 3: *Loop shrinking property illustrated on simple geometric objects.*

To capture the essence of the loop shrinking property, we consider a diffusion process starting from an arbitrary location *p* on the surface. If we are on the surface of a sphere, the advancing front will grow steadily but eventually it will start shrinking and vanish to one point similar to the thread losing hold. On the other hand, if we are on the surface of a torus, the growing patch will initially have a single boundary (Figure 4-left-top), and eventually, it will meet itself as it folds like a cannoli yielding a tunnel like region with two separate boundaries (Figure 4-left-middle). At this point, we can only confirm that we are evolving on a tubular structure but we cannot infer the existence of a handle. Only at the moment when the two advancing fronts meet (Figure 4-left-bottom), then a handle loop is detected. A tunnel loop can be obtained as a streamline by tracing back from the handle loop along the diffusion gradient, see Figure 5. Clearly here, the number of possible tunnel loops is large and all of them will pass through the initial point *p*, so there is no reason to expect them to exhibit some geometric optimality. This can be remedied by performing a diffusion initiated from the ring of the handle loop (Figure 4-left-bottom). Once the diffusion converges we can trace multiple streamlines sampled along the handle loop, thus obtaining a set of tunnel loops and we can then select the shortest.

This idea extends naturally to higher genus settings. For instance, let us consider the case of the double torus in Figure 4-right-top, the diffusion from the point marked on the surface will behave initially in a similar manner to the torus case, but as the two fronts

merge (Figure 4-right-middle), the resulting single front will eventually split into two fronts (Figure 4-right-bottom) which would meet each other again yielding the second handle loop.
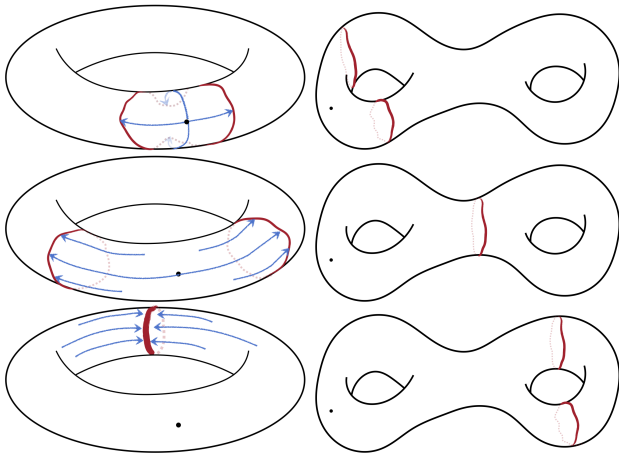

Figure 4: *Handle loop discovery through splitting and collision of diffusion fronts on a simple torus (left) and a double torus (right).*

Despite the inherent simplicity of the process described above, special attention needs to be paid to front tracking, bookkeeping, memory usage and performance in order to build a practical solution suitable for large data sets and complex surface layouts. A possible way forward is to use the level set method for carrying out the front tracking of the diffusion on the surface, however the sharp interface would require substantial bookkeeping and costly computations and checks on top of commonly known issues of the method pertaining to the distance function evaluation and re-initialization. Instead, we rely on the recent layered fields approach proposed as an alternative model to the ubiquitous Voronoi diagrams for modeling natural tessellation modeling [ZMSS18]. We use a narrow band to represent the advancing front to avoid discontinuities across the front. We regard the propagation as field evolving on a layer on the surface. The splitting of the interface corresponds to the splitting of the field into two different fields evolving on different layers. For this purpose, we develop a dynamic approach to layer creation and layer collision, which allows us to keep track of all the handles and tunnels on non-trivial higher genus surfaces.

We represent the different layers as rows of a sparse matrix which both allows for carrying numerical diffusion computations and the tracking of topological features. As our geometry processing operations are channeled through linear algebra we take full advantage of parallel linear algebra primitives. In particular, our implementation operates fully on graphics hardware. A windfall of our approach is the ability to generate a Reeb graph simply by performing diffusion initiated at the detected handle loops. In this way, the commonly encountered shortcomings of working with height based Morse functions are avoided in the first place.

In summary, this work makes the following contributions:

- Novel variational abstraction for the loop shrinking property
- Succinct diffuse interface Model for identifying handle and tunnel loops and capturing front propagation and collision

- Dynamic sparse matrix representation for bookkeeping and processing of multi-frontal branching and merging
- Fully parallel algorithmic realization on graphics hardware
- Simple and intuitive Skeletonization by Reeb graph calculation and embedding

2. Previous work

The body of work on extracting topological primitives on surfaces spans efforts across computational geometry and mesh processing. Theoretical algorithms for extracting non-trivial loops on surfaces have been proposed, *e.g.* [EW05; Kut06; dVE06]. However, there are no existing realizations of these theoretical efforts and no guarantees that the resulting loops are geometrically meaningful handles or tunnels. In the context of mesh simplification, several heuristics for identifying small handles have been proposed, *e.g.* [EV97; GW01]. The use of intermediate graphs for non-trivial loop identification has been explored in terms of *Reeb graphs* in [SL01; WHDS04] and in terms of medial axis in [ZJH07].

More closely related to our work are the efforts directly targeting the localization of handle and tunnel loops. Dey and coworkers addressed the problem by relying on persistent homology in [DLSC08], and then later on by making use of Reeb graphs [DFW13]. In the former, the intermediate tetrahedral tessellation required for carrying out homology computations poses several challenges both to feasibility and performance and bloats the problem size unnecessarily. Dropping persistent homology, in the latter, in favor of Reeb graphs improved both performance and scope, but computational cost and memory requirements remain considerable.

A data structure based on tree-cotree decomposition which allows for constructing generators of fundamental groups of surfaces was proposed by Eppstein [Epp03]. Adjusting edge weights of the tree-cotree decomposition based on principal curvature directions, [DEG09] attempt to inject geometric meaning into those graph cycles by to steering them to align with those directions. These methods are prone to produce redundant non-trivial cycles which require post-processing. An iterative approach alternating between principal directions to find "good" cycles was proposed in [CJG18]. It should be noted that curvature directions on general surfaces do not necessarily match the targeted cycle directions as the surface can be heavily decorated, see for instance, the dragons models in Figure 15 and this limits the scope to overly smooth meshes.

Without loss of generality, the above discussed methods do not operate directly on the 2D manifold but require one or multiple intermediate structures. Subsequently, performance and results quality depend on the quality of these underlying structures. For instance, in [DFW13], the Reeb graph quality depends among other thing on the choice of the initial direction as usually the associated Morse function is some height function. Likewise, [DEG09; CJG18] depend on principal directions estimations which are generally local, sensitive to surface fluctuations and noise. More importantly, the resulting algorithmic pipeline is not streamlined and therefore pose further challenges to code vectorization in view of the ubiquitous parallelism in modern hardware infrastructure.

3. Mathematical representation and data structure

General setting. A majority of existing methods operate in the combinatorial manifold setting where the computation of shortest homology generators is relatively simpler than the more geometrically meaningful piecewise linear manifold. While this simplification offers certain algorithmic guarantees and computational advantages, the results depend largely on the quality of the underlying mesh edges. The challenges of working directly on the piecewise linear manifold have long been recognized, for instance, [EW05] discuss using level-sets for speeding up shortest path computations but acknowledge that "even the simpler problem of finding the shortest non-separating cycle in a piecewise linear manifold appears to be open".

Similar to [DFW13], our work seeks to find geometrically meaningful generating pairs, meaningful is to be understood in the sense of shortest loops. In our illustrations, e.g., Figure 4, as in most common objects, the handle loop is way shorter than the tunnel loop (think a mug handle for instance), so for the sake of presentation flow, we presume in the ensuing discussion that handle loops are found before their tunnel counterparts. Of course, it is possible to imagine scenarios where the opposite happens, e.g., deep screw holes in the drill in Figure 15 but this does not affect the pair extraction per se and the distinction between handle and tunnel loops will be discussed aside.

Diffuse interface. While our approach is driven by diffusion, a pure diffusion alone is not sufficient for tracking different advancing fronts and resolving front collisions. In fact, it would lead simply to a smeared interface which would eventually end up as a flat curve (merged fronts). This issue gets further amplified on higher genus surfaces as multiple branchings yield multiple concurrent fronts. In order to resolve these issues and to avoid the numerical problems commonly encountered with sharp interface such as derivative discontinuities, we adopt a diffuse interface approach where the advancing front is not a sharp line as in the level set method but a narrow band. In this way the growing patch can be defined by a function φ_i defined over the whole mesh and which takes value 1 inside the patch and 0 outside the cell and is smoothly varying on a narrow band where $0 < \varphi_i < 1$. In this way, it is easy to localize the narrow band by simple inspection of field values. Furthermore, at any time, it is possible to extract a sharp boundary as a level set of the narrow band.

Mathematical characterization. In our context, front splitting due to branching, see e.g. Figure 4, leads to the creation of new separate regions and front collision requires colliding fronts to freeze thus blocking each other. These behavior patterns are similar to the natural tessellation model proposed in [ZMSS18; SMS*20] which describes the evolution of the interface of multiple cells growing simultaneously on a surface and yields Voronoi-like surface tessellations.

With the notion of diffuse interface, we are contemplating a situation where multiple scalar functions, associated with each region, are prescribed on the surface. As these different functions start interacting with each other, e.g., when the fronts meet, tracking which front belongs to which regions becomes very hard as we end up just with one complicated scalar function defined on the surface. To resolve this issue, we rely on a layered field representation where multiple scalar functions are defined over the surface as if they live on different layers. In this way, we can allow interaction between multiple fronts and at the same time we can access each scalar function by simply looking at the layer it lives on.

Within this setting, we can regard the diffusion of n individual regions or cells as fields evolving on different layers and their interactions are governed by a set of requirements and weighted energy terms which drive the overall cell interactions. To fix the ideas, each region can be likened to the yolk part of an egg and its narrow-band to the egg white. When multiple eggs are put together, the whites block each other creating an interface and protecting the yolks which remain whole. If an egg white is breached the distinguishability of narrow bands corresponding to the interface at that location gets lost. This can be accounted for by imposing exclusivity on the narrow bands associated with each region. As the narrow bands represent the evolving front of the region, their gradients are non-null, and hence the desired effect can be encoded as the weighted gradients $-\frac{1}{2}a_{ij}\nabla\varphi_i\nabla\varphi_j$. In a similar way, to prevent the yolk of an egg from breaking into multiple parts, or merging with other yolks, we encode the integrity in terms of the weighted product $w_{ij}\varphi_i\varphi_j$. A more physics-inclined interpretation is to look at both terms as the kinetic and potential energy respectively and the goal is to evolve the system while maintaining a low entropy. Besides, by definition of our diffuse interface, we impose partition of unity so that everywhere on the surface, field contributions φ_i sum up to 1. For prescribing the behavior of narrow bands when they enter into contact, we use a function g which will be described shortly.

To account for the partition of unity constraint we introduce the Lagrange multiplier λ. The φ_i's can be treated as independent variables, and the arising Lagrangian then reads

$$\int_S \sum_{i=1}^n \sum_{j=i+1}^n \left(w_{ij}\varphi_i\varphi_j - \frac{a_{ij}}{2}\nabla\varphi_i\nabla\varphi_j + g\right) + \lambda(\sum_{i=1}^n \varphi_i - 1)\, ds,$$

where the integral spans the surface S and the summation is over the number of patches or regions n.

Following [ZMSS18], the minimization of the Lagrangian yields the following time dependent equation which describes the evolution of the individual fields

$$\dot{\varphi}_i = -\sum_{j=1}^n \frac{\mu}{n}\left(\sum_{k=1}^n \left[(w_{ik}-w_{jk})\varphi_k + \frac{1}{2}(a_{ik}^2 - a_{jk}^2)\nabla^2\varphi_k\right]\right.$$
$$\left. + (\frac{\partial g}{\partial\varphi_i} - \frac{\partial g}{\partial\varphi_j})\right). \quad (1)$$

As observed in the above equation, We do not need to define g explicitly, we only need to define the difference term $(\frac{\partial g}{\partial\varphi_i} - \frac{\partial g}{\partial\varphi_j})$. The function g encodes the desired behavior when two boundary bands meet each other. In the simple case, where only two bands φ_i, φ_j meet, we have by virtue of partition of unity, $\varphi_j = 1 - \varphi_i$. In practice , we seek a function that produces a symmetric behavior in the interval $(0, 1)$ and vanishes at 0 and 1 and does not change sign.

One such function is $\sqrt{\varphi_i(1 - \varphi_i)}$, in this way, we have

$$\dot{\varphi}_i = -\sum_{j=1}^{n} \frac{\mu_{ij}}{n} \left(\sum_{k=1}^{n} \left[(w_{ik} - w_{jk})\varphi_k + \frac{1}{2}(a_{ik}^2 - a_{jk}^2)\nabla^2 \varphi_k \right] \right.$$
$$\left. - e_{ij}\sqrt{\varphi_i \varphi_j} \right).$$

In the above equation, we recognize the pure diffusion term characterized by the Laplacian which guarantees the smoothness of the growing interface. The first term in the equation ensures a stable and stationary interface and therefore balances the effect of the diffusion term. The last term characterizes the driving force of the interface between the ith and the jth patch, namely the behavior chosen for g. The growth rate is controlled by the mobility term μ_{ij}. The solution can then be carried by a simple explicit Euler stepping scheme $\varphi_i(t + \Delta t) = \varphi_i(t) + \dot{\varphi}_i \Delta t$. For a discussion on stability or more elaborate numerical schemes, the reader is referred to [ZMSS18] or standard numerical methods textbooks. In all our experiments, a constant time step was sufficient.

Data representation. The numerical modeling is carried out by means of linear finite element approximations. Our experimental results suggest there is no need for going to higher order approximations. The layered fields abstraction lends itself to a compact and efficient representation as a sparse matrix Φ where rows span the fields φ_i and columns span the mesh vertices. This representation is convenient in the sense that we can precompute the Laplacian matrix of the mesh and then at each time step the evaluation of the Laplacian of the fields amounts to a sparse matrix-matrix multiplication with Φ. The Laplacian of φ_i is just the ith row in the resulting product. This formulation offers a clear advantage in view of efficient parallel implementation. Furthermore, checking if two patches share a boundary amounts to checking their corresponding rows in Φ for common nonzeros along the columns (vertices). Partition of unity across the layers can be conveniently enforced by normalizing the columns of the field matrix Φ after each iteration.

At initialization, field values for vertices within a given cell are set to 1, and an additional base layer is created and set to 1 everywhere. This layer serves two purposes, it triggers the overall evolution and serves as an inexpensive means for handling interface interactions implicitly. In fact, as two or more fronts enter into contact, the field values on the base layer at the corresponding locations reflect the contact as they do not flag 1's anymore. By virtue of this observation, we set the mobility term μ_{ij} and boundary interaction term e_{ij} to 0 except when either indices refer to the base layer. In that case, they are set to $\frac{1}{4}$ and $\frac{1}{30}$ resp. The coefficients of the gradient energy $a_{i,j}$ are set to $\frac{1}{25}$, and those for the penalty term $w_{i,j}$ to $\frac{a_{i,j}}{5}$ when $i \neq j$, and to 0 otherwise. We did not see any need for changing the parameter values proposed in [ZMSS18; SMS*20] despite the problem settings being completely different.

4. Variational Loop Shrinking

From the outline given in the introduction, our method can be regarded as a sequence of three distinct passes. The initial pass registers the number of handle and tunnel pairs and gives an initial
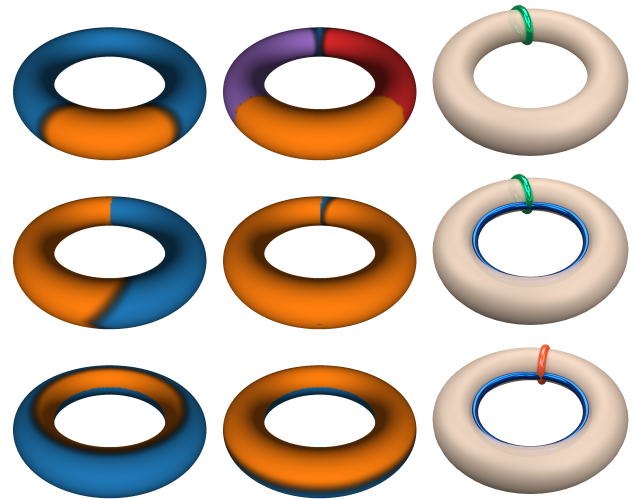

Figure 5: *In the initial pass (top), the field interface (rim of the orange region) splits into two different components (top-left), both of which propagate till they reach each other (top-middle) yielding an initial handle estimate (top-right). In the second pass (middle), a diffusion initiated leftwards of the handle (middle-left) progresses till it reaches the opposite side of the handle (middle-middle). A streamline tracing against the diffusion gradient field yields the tunnel loop (middle-right). Similarly, in the third pass (bottom), a diffusion from the tunnel loop upwards (bottom-left) reaches the opposite side of the tunnel (bottom-middle) yielding a refined handle loop (bottom-right).*

estimate for each handle loop. In the second pass, tunnel loops corresponding to each initial handle are created. In the third pass, the tunnel loop is used to generate a refined handle loop. Figure 5 depicts these three stages on the simple case of a torus. Throughout these passes, we mitigate the effect of the random seeding of the diffusion process in the initial pass and we improve the quality of the reported loops. If the focus is only on producing topologically correct loops without any geometric considerations, the final pass may be skipped and the estimates from the initial pass can be used as the handle loops.

4.1. Initial Pass

The diffusion process is initialized from a random point on the surface and carried out by a simple explicit Euler stepping scheme. After each update, we check the state of the boundary of each advancing front (by inspecting nonzeros along the columns of Φ which is much cheaper than computing isolines at each iteration. Isolines are computed only when a collision or branching is detected). We encounter a branching on the surface when there is more than one boundary loop (narrow band), see Figures 5-top-left and 6-left. In this case, each boundary loop is transferred to a new layer (row) in Φ. As Φ is modeled as a compressed sparse row (CSR) matrix, the transfer means we only have to adjust the column index for each vertex part of a specific connected component. The parent layer which gave birth to these new fronts can then be ignored in future
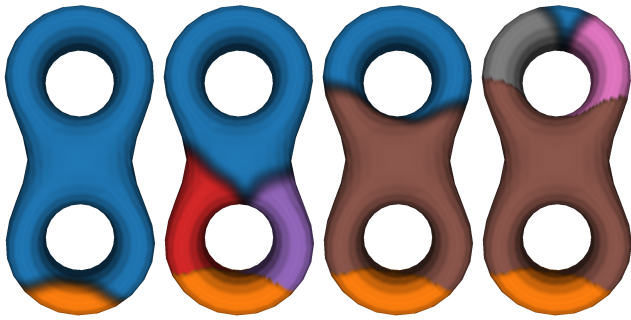
Figure 6: *Handle detection on the double torus during the initial pass. The creation of new colors, reflect dynamic changes to multi-layered field representation Φ such as layer creation or merging as the advancing fronts split or merge.*

iterations, because it has no space left to diffuse further, see e.g. Figure 4 for an illustration. Please note that for visualizations purposes, the transfer is visualized in a vertex-only coloring and not interpolated, therefore it looks jagged as in Figures, 5 and 6; the transfer itself is fully integrated in the smooth diffusion process.

A handle is detected when multiple advancing fronts meet each other, Figures 5-top-middle and 6-second-left. This can be inferred from Φ by inspecting which advancing fronts (rows) contribute field values to the same vertices (columns). Therefore, if a vertex has contributions from two or more layers, those layers should be merged. The number of handles detected is the number of layers involved minus one, e.g., two layers yield one handle. Inversely, to the scattering of multiple advancing fronts to multiple layers, now we gather all involved layers into a single new layer. This time the energy values per vertex of all involved layers are accumulated into the energy value of the last involved layer, clearing all the others. Afterwards, the row index of the accumulation result is shifted to represent the new layer. This process avoids any expensive sparsity pattern changes and is fast to apply. The handle loops are formed by the involved advancing fronts before they are merged into a single layer, see Figures 5-top-right, 6-right, and 7. Since the number of handles equals the number of involved advancing fronts minus one, one boundary loop has to be ignored. Figure 5-top-right visualizes the formed handle loop as a green line. In all of our experiments, the longest boundary loop is ignored, as this choice seems to give the best visual results. In general, this is not required and our final results are independent of this choice, as long as one layer is ignored.

4.2. Tunnel loop pass

It is possible to generate tunnel loops by back-tracing streamlines against the diffusion gradient in the initial pass. However, this turns out to be a poor algorithmic choice because the diffusion emanates from a single seed. In order to produce well-behaved tunnel loops we take a more judicious approach. We set up a diffusion process to start on one side of the handle loop whereas the other acts as a border preventing propagation in the opposite side. In this way, the diffusion process can get from one side of the handle loop to the

other, see Figure 5-middle. If the other side is reached, we can generate streamline back to the start points. The first point reached on the other side is used as the start point for the streamline trace. Due to the assumption that the diffusion happens at a constant speed on the surface, this will give us an approximation of the shortest path from one side to the other. This naive approach, however, would not work for complex geometries, as there may be multiple tunnel candidates for a single handle estimate, as illustrated in Figure 7. Whenever a handle estimate lies on a tubular connection between two holes, as highlighted by the red rectangle, there will be two possibilities for a tunnel, one on each side. Without further restricting the diffusion process we cannot guarantee which tunnel will be found by the Streamlines. Consequently, the same tunnel may be reported by different handle estimates, if they lie along the same tunnel. To fix this problem, we introduce an implicit ordering of the reported tunnels. The second diffusion process for each handle estimate is restricted to the surface area which was already covered at the time the handle estimate was detected. This ensures that a handle estimate finds the shortest tunnel fully covered by the initial diffusion process and no tunnel is reported more than once. Multi-frontal branching and collisions, as the ones observed in Figure 7, are therefore handled correctly and efficiently without the need for elaborate data structures or intermediate representations thanks to the already discussed sparse matrix field representation.

4.3. Handle refinement pass

There is a priori no reason for the handle loop generated in the initial pass to be optimally placed or shaped, for instance, perfectly round at the thinnest section of handle. Often, it will be skewed or
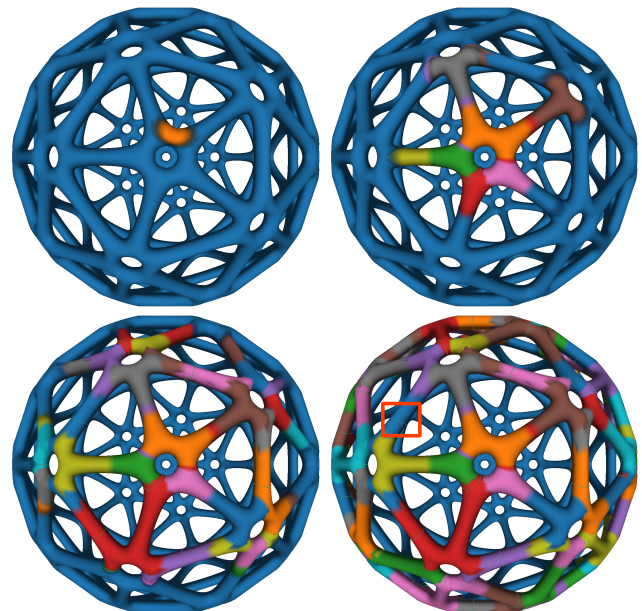

Figure 7: *Visualization of the progression of the initial pass on the ball mesh. Initiated at a single seed (top-left), the front expands and branches out yielding multiple fronts which conquer half of the ball (bottom-right).*
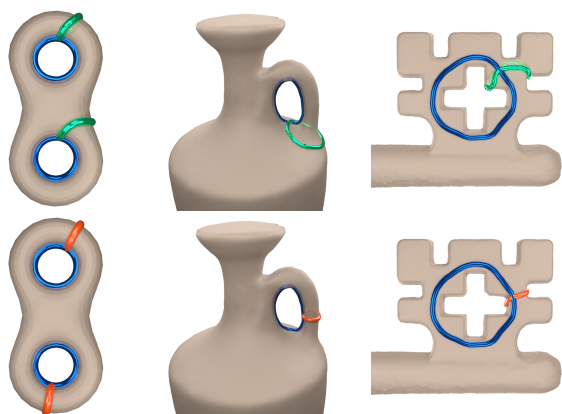
Figure 8: *Examples for improved handle loops (bottom) compared to the estimates obtained from the initial pass (top). Estimates are rendered as green tubes, tunnel loops are in blue and improved handle loops as orange.*

twisted, see examples in Figure 8-top. By applying the same steps as to generate the tunnel loop we can generate a refined handle loop. We start from one side of the tunnel loop and after we reach the other side, we trace backwards starting from the first point reached, see Figure 5-bottom. This time the diffusion process is restricted by the corresponding tunnel loop only. The streamline tracing will yield an improved, ideally placed handle loop, see Figure 8-bottom. Please note that although our approach is purely heuristic, it is well justified and avoids any costly numerical optimization.
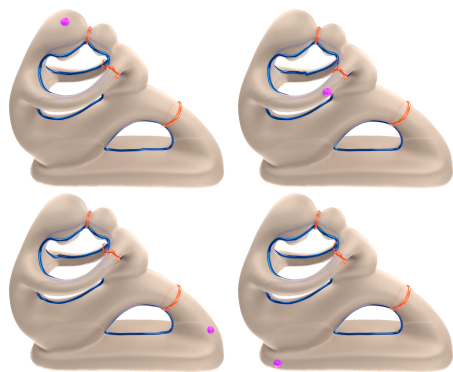

Figure 9: *Detected handle and tunnel loops for different initializations. The purple point shows the starting location of the diffusion.*

4.4. Robustness to initial seed placement

As discussed above, the multiple pass strategy makes our approach robust to the initial random seed placement. This is confirmed empirically in our test results. A typical scenario is shown in Figure 9. Starting from different locations on the surface of the *mother and child* model, our method generates nearly identical results. Please note that the handle loops, in orange, slightly differ. This is expected because there are multiple location having similar cross-

section thickness alongside the arms. Therefore, there are multiple candidate locations which produce meaningful handle loop.
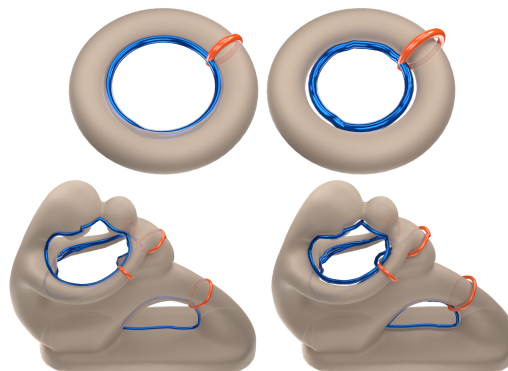

Figure 10: *Distinguishing handle and tunnel loops by offsetting them along their corresponding surface normals (right). As illustrated, tunnel loops tend to shrink whereas handle loops expand.*

4.5. Distinguishing handle and tunnel loops

So far, we have assumed that loop handles are found first. This is not always the case, especially for deep holes as for instance in the drill in Figure 15. Our algorithmic flow does not depend on this distinction, therefore, we can do identification at the end. Keeping to the simplicity of our overall approach, basic checks such as slightly offsetting the final loop along its corresponding surface normals and checking the changes in its chord length, as illustrated in Figure 10, is sufficient for making the distinction in our extensive test cases. Tunnel loops favor hyperbolic parts and will tend to shrink after offsetting whereas handle loops tend to expand. It might be possible to engineer scenarios where this basic distinction fails however we believe this is a reasonable price for a trade-off between fast practical solution and costly bulletproof robustness.

5. Performance Considerations

While it would have been easier to implement our current algorithmic pipeline in a multithreaded fashion, our aim it to harness the tight memory requirement on the GPU and fine grained parallelism pertaining to modern graphics hardware. The two major challenges we faced are finding ways to parallelize the tracking of advancing fronts as well as keeping the communication and synchronization between CPU and GPU at a minimum.

In all times, our method needs to keep an eye on the evolution of the advancing fronts. The extraction of the advancing fronts on a layer requires a list of all triangles which are part of the advancing fronts. A triangle is part of the advancing front, if one or two vertices of that triangle have an energy level above a given threshold and below 1. By applying this condition while iterating over all triangles in parallel, we atomically build the list of all triangles part of the advancing front.

Identifying the number of separate advancing fronts, while an easy serial exercise is not straightforward in parallel. It can be modeled as a connected component labeling problem on an unstructured

Figure 11: *Reeb graph creation on the genus-0 Deer model. Please note how the branching structure of the deer horns is well captured.*
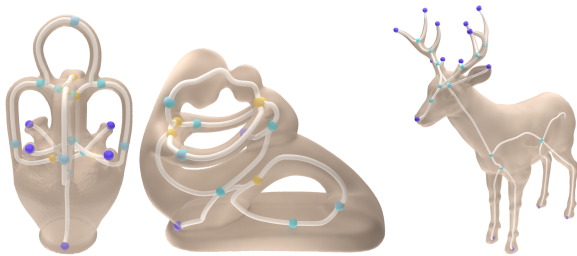

Figure 12: *Geometrically embedded Reeb graphs on a selection of relevant test cases.*

Figure 13: *Comparison of Skeletons produced by our method (left) and from the Topology Toolkit [TFL*18](right).*

6. Reeb Graph Extraction

While our objective has been the detection of handle tunnel and loops, looking at Figure 6 it is clear that the diffusion process somewhat capture all the branching in the surface. While the topological information encoded by a function defined on a given surface by tracking the connected components of it level sets is generally captured in the notion of Reeb graphs.

Interest into Reeb graphs in computer graphics goes back to the work of [SKK91] which opened the door for a steady research effort on the topic. This includes performance oriented solutions, e.g. [PSBM07], mainstream implementations such as the on available from within the *Topology Toolkit* [TFL*18], improvements to the algorithmic complexity, eg. [DN09], and extensions and generalizations e.g. [BFS00].

In this digression, our goal is not produce a state-of-the-art Reeb graph algorithm but to show the versatility of our approach, in the sense, that by the same token we get the relevant handle and tunnel loops and meaningful skeleton by geometric embedding of the Reeb graph associated with our diffusion on the surface.

The critical events forming the Reeb graph, namely splitting, merging and vanishing of connected components (fronts), are already detected in our initial pass for handle and tunnel detection. The embedding of those events are obtained by simply taking the mean of the narrow band at the time those events occurred. Additionally, the mean of each advancing front is captured every few iterations to get points between critical events. This process is visualized in Figure 11 where the sampled points from the boundary and the critical events are shown as white spheres.

Just using the diffusion initialized at a random point as described in the subsection 4.1 would naturally yield a topologically correct graph as shown in Figure 11. However, the embedding especially around critical points will have jumps because the mean of the front jumps when a front splits or two or more merge together. For example at the nose of the deer or at the front legs, where the embedding even intersects with the surface. To improve the embedding in those regions we apply a similar principle as for the handle refinement pass from subsection 4.3. For each front (children) created by splitting a front (parent) we start a separate diffusion process which can only populate backwards compared the initial diffusion process. This ensures that the front will propagate towards the parent and along the jump in the embedding. During this pass we insert sampled points similar to before in between the connection of the child and the parent to improve the quality of those regions. The diffu-

grid where triangles represent nodes and their connectivity from the edges between nodes. Connected component labeling is a challenging task to implement efficiently on the GPU. The nature of this problem requires scattered memory accesses as well as multiple passes, which both are problematic in terms of performance on the GPU. Our method uses the algorithm described in the work by Soman et al. [SKN10], which minimizes the communication between CPU and GPU and reduces the scattered write operations to improve performance.

The time stepping during the initial pass is dominated by the Laplacian evaluation which is encoded as sparse matrix matrix multiplication (SpGEMM) and can readily take advantage of existing numerical kernels.

The tunnel loop pass (subsection 4.2) and handle refinement pass (subsection 4.3) only differ in the initialization of the diffusion process and are discussed as one. Compared to the diffusion process during the initial pass, in these passes, the diffusion process has exactly one single layer. No scattering or merging is performed. This allows us to replace the sparse resizable system matrix Φ by a dense vector containing the energy value of each vertex. The base layer, required by the diffusion process, is implicitly modeled as one minus the energy value of the first layer. Furthermore, the SpGEMM during the diffusion update is replaced by a simple sparse matrix vector multiplication (SpMV). This performance optimization of the diffusion process is particularly important because each tunnel and handle loop requires a separate diffusion process. To fully utilize the GPU, multiple gradient and handle refinement passes run in parallel.
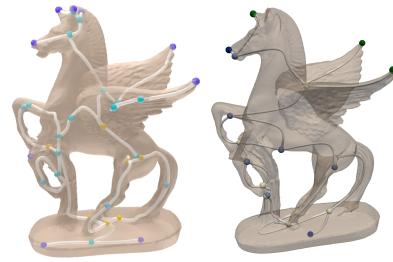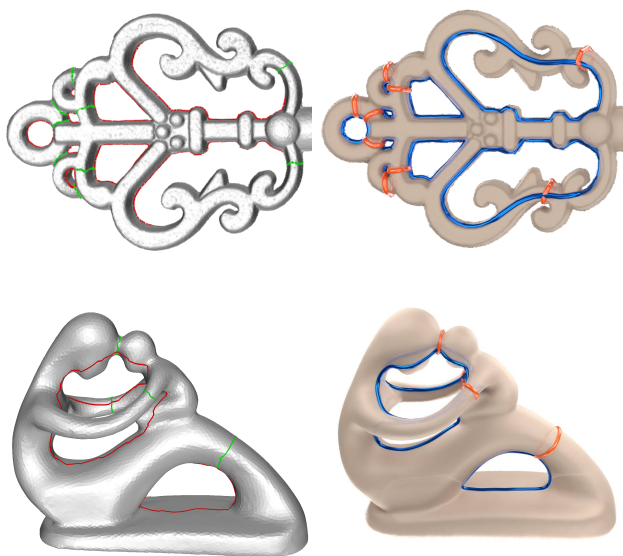
Figure 14: *Side by side comparison between results of [DFW13] (left) and our approach (right) on the Key [Art21] and Mother with Child meshes.*

Mesh	Ours		[DFW13]
	Total (#threads)		Total
MotherWithChild(28k, 4)	92(1)	18 (4)	76
Botijo(82k, 5)	108(1)	108 (5)	193
Casting(93k, 9)	114(1)	134 (9)	210
Happy Buddha(98k, 8)	97(1)	112 (8)	189
Ball(184k, 120)	152(1)	458 (10)	2135
Metal Key(390k, 10)	210(1)	552 (10)	810
Wooden Chair(400k, 7)	214(1)	472 (7)	921
Dragon Phoenix(750k, 11)	340(1)	1062 (10)	1540
Grayloc(921k, 5)	366(1)	1155 (5)	0
V745 Sco Nova(923k, 184)	502(1)	1402 (10)	DNF
Metal Table(950k, 198)	336(1)	882 (10)	16891
Dancing children(1.4M, 8)	462(1)	612 (8)	4992
Drill(1.5m, 13)	496(1)	1844 (13)	4142
Dragon Tamer(2M, 335)	636(1)	1410 (10)	>32562
Dragon Ball(2.4M, 5)	728(1)	1452 (5)	4557
Napoleon(6.5M, 25)	1844(1)	7744 (10)	>15000
Statue(10M, 3)	496(1)	1844 (13)	4142

Table 1: *Memory consumption in megabytes for various test models. For our method peak GPU memory usage is reported with and without running the handle and tunnel passes in parallel. For [DFW13] peak CPU memory consumption is reported.*

sion is stopped when a vertex from the parent front is reached. The same process is applied for fronts (children) which were merged into a single one (parent). Typical results of our approach after the refinement are shown in Figure 12 and Figure 13. Figure 13 shows that our results are comparable to skeletons produced by existing state of the art, however, an objective comparison is difficult due to the different nature of the algorithms.

An estimate of the performance can be inferred by comparing the cost of our method from table 2 to one of the top performing methods [PSBM07] since it is the method used in the Reeb graph step in [DFW13] or [BR21] for a state-of-the-art Reeb graph skeleton embedding method.

7. Results

All our experiment were carried on an Intel(R) Core(TM) i7-7700 with 32 GB system memory and an NVIDIA RTX 2080ti 11 GB. As initialization point for the initial diffusion process we used the first vertex of the mesh.

Figure 15 shows a collection of non-trivial test case meshes featuring: large loops as in the Pegasus model and the dancing children, object with thin features such as the key and the chairs and the mechanical parts. In all these cases, our approach detects the handles and tunnels correctly. The handles and tunnels in the dragon models are heavily decorated with ornaments. The reported tunnel loops correctly avoid following the ornaments which face inwards compared to the tunnel loop. Following those ornaments would increase the length of the loop. The handle loops are located at the smallest parts despite the ornaments. The Napoleon mesh [Art21] features very large tunnels, e.g. the tunnel covering the stone plate, two legs and the body of the horse, and also very tiny ones, e.g. at the top part of the foot rest or at the reins. The drill model hash very

small tunnels combined with large handles for the screw holes. All tunnels and handles are reported correctly, because our method is not based on any presumptions on the length of handle or tunnel loops. Note that not all screw holes feature a tunnel in the mesh, since it is a 3D scan.

Table 2 compares the runtime of our method with the state-of-the-art approach [DFW13]. The results show that our method scales significantly better with larger meshes and higher genus. Our experiments included five meshes with a high genus: Ball, Metal Table [Art21], V745 Sco Nova and Dragon Tamer [Art21]. The results of these meshes are visualized in Figure 15 and Figure 1. In these cases our method outperformed the other method by over an order of magnitude. Grayloc, V745 Sco Nova, Napoleon and Dragon Tamer, which has the highest genus of all tested meshes, did not complete with their method. The metal table required multiple runs until their method completed without an error, we report only the runtime of the successful run.

The number of pairs detected by both methods is the same. As it is cumbersome to match the pairs of both methods automatically for comparison, we compare the total length of all loops. The length ratio in Table 2 suggests that our method yields shorter loops in general. Visual comparison of reported loops between our method in Figure 15 and [DFW13] in Figure 14 confirm that especially handle loops reported by our method follow the area with the smallest arc length around the handle more smoothly, i.e. at the base of the Mother with Child mesh.

Table 1 contains the memory consumptions obtained during our experiments. We report peak GPU memory consumption with and without running handle and tunnel refinement passes in parallel kernels on the GPU. Except for very small meshes, our method

Figure 15: *Typical results of our approach on multiple data sets: From top left, Pegasus, Dancing Children, Chair, Grayloc, Thai Statue, Dragon Phoenix, Dragon Ball, Metal Key, Casting, Botijo Jar, the Napoleon model (with zoomed views), the Drill model (double-sided view) and very high genus test cases, Ball, Metal Table, and a model of the double star system V745 Sco Nova (courtesy of NASA).*

uses less memory even when multiple handle and tunnel refinement passes run in parallel.

The memory consumption does not directly scale by the number of parallel runs because some read only information, such as the mesh, can be shared between these passes. Our initial tests showed that running more than 10 passes in parallel did not improve the performance further. This is probably caused by driver overhead scheduling the parallel kernel runs on the GPU.

While more elaborate numerical schemes can be used with our method, our experiments suggest that the standard time stepping strategy adopted herein is sufficient in practice. In fact, the method is generally robust to time step perturbation. For instance, scaling the time step by one order of magnitude, yields the same number of loops. Computing the Hausdorff distance between each corre-

sponding pair of loops suggest only negligible variations. Typical values for worst pair difference in terms of Hausdorff distance RMS(%) are 0.006 for the Mother with Child model 0.050 for the Dragon Ball model, 0.010 for the Drill model, and 0.012 for the Pegasus.

The performance of [DFW13] depends largely on the fast Reeb graph computations of [PSBM07] which in turn uses "free" height functions (coordinates) as Morse functions. Certainly, other functions can be used, the "streaming meshes" format has been used in the latter but then the cost of the underlying Fiedler vector computation should be factored in. Similarly, when geodesic or harmonic functions, e.g., [HSKK01] are used, their cost will impede performance. Therefore, the comparisons we are providing are representative.

	Ours			[DFW13]					Length ratio
Mesh(#faces, Genus)	**Total**	Initial P.	Handles/Tunnels	**Total**	Reeb graph	Map/Link	Edge annot.	Shorten.	ours/[DFW13]
MotherWithChild(28K,4)	0.42	0.29	0.12	0.52	0.05	0.05	0.02	0.40	0.988
Botijo(82K,5)	1.78	0.65	1.23	1.81	0.21	0.17	0.11	1.32	0.960
Casting(93K,9)	1.31	0.70	0.60	3.10	0.40	0.10	0.16	2.44	0.999
Happy Buddha(98K,8)	1.13	0.61	0.52	2.04	0.24	0.10	0.15	1.55	0.988
Ball (184K, 120)	17.65	13.01	3.13	401.77	0.28	34.76	3.16	363.57	0.918
Metal Key(390k,10)	10.48	6.87	3.61	36.52	10.35	9.52	0.52	16.13	0.984
Wooden Chair(400K,7)	7.49	4.45	3.05	31.21	2.77	18.55	0.43	9.46	0.982
Dragon Phoenix (750K, 11)	12.89	8.75	4.14	100.83	9.77	50.66	1.73	38.67	0.942
Grayloc (921k, 5)	22.25	14.23	8.01	DNF					
V745 Sco Nova (923K, 184)	273.99	104.21	168.66	DNF					
Metal Table(950K, 198)	151.33	92.31	57.34	3,732.92	4.49	35.74	16.68	3,676.01	0.954
Dancing children(1.3M,8)	20.57	13.29	7.14	102.54	38.26	7.60	3.16	53.52	0.989
Drill (1.5m, 13)	18.77	13.92	4.85	90.32	13.52	1.14	2.74	72.92	0.846
Dragon Tamer(2M, 335)	429.57	254.93	172.53	>20,000.00	16.47	885.50	249.81	Crash	
Dragon Ball (2.4M, 5)	37.29	21.88	15.41	91.70	38.14	1.28	2.84	49.44	0.975
Napoleon (6.5M, 25)	705.34	376.50	328.87	>2,970.16	2730.67	239.49	Crash		
Statue (10M, 3)	167.71	152.18	15.53	491.69	253.45	19.12	7.91	211.21	0.976

Table 2: *Runtime comparison for various test models. Timings are measured in seconds. The length ratio compares the total handle and tunnel loops length of our method against [DFW13].*
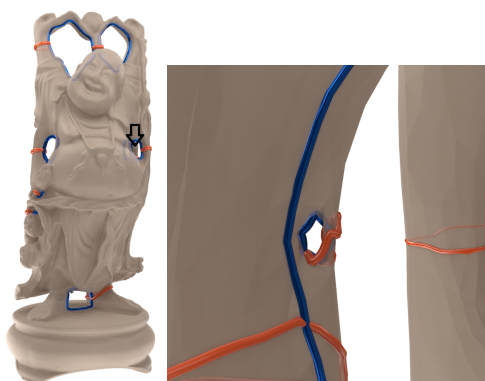

Figure 16: *The Happy Buddha model (left) and a zoom-in on the area marked by the arrow showing the effect of a tiny handle spanned by only three triangles on the output of our variational method.*

7.1. Limitations and discussion.

Our approach operates on the piecewise linear manifold setting, and cannot offer the same algorithmic guarantees of well and long-established combinatorial approaches. However, our method relies on first principles in topology embodied in the loop shrinking property and therefore inherits the same theoretical guarantees. Our method builds on the ability to conduct standard numerical simulations on surface meshes and is hence bound by fairly well-known requirements on methods such as the finite element method. We do not see this as a limitation but rather a motivation for exploring common grounds between combinatorial and variational approaches, especially that ample empirical evidence shows that our method produces the same number of pairs as [Dey2013].

There can be scenarios where tiny and coarsely meshed handles can affect the quality of our results or even escape the width of our advancing front. The only test case where we encountered such a scenario is depicted in Figure 16. Although the handle is only spanned by three triangles, our method still captures it but the loop is clearly suboptimal. Although this does not respect the smoothness assumptions of the linear manifold required in our variational setting, it can be addressed by adaptive refinement along the front.

8. Conclusion

We have re-abstracted the shrinking loop property as a continuous growth process steered by diffusion. The diffusion field was modeled using diffuse interface and the dynamics of the advancing fronts were captured by assuming a multilayered field representation where the creation and merging of layers are respectively steered by front splitting and collision. Our approach is simple and versatile allowing the detection of handle and tunnel loops as well as the creation of Reeb graphs. Furthermore, the time stepping nature of our approach allows for a close control of the whole process by simple adjustment of the field and regions of diffusion. We foresee that this control will help extend our approach to other application such topological surgery, mesh segmentation, and rigging. We hope that adopting our methodology will help break the deadlock in performance and bring new machinery to help to understanding and harnessing geometric problems. The variational nature of our approach allows future improvements by means of improved numerical schemes such as adaptive mesh refinement along the front.

References

[Art21] ARTEC3D. *3D Scans, Content Hub, Metal Table, Napoleon, Copper key, Dragon and phoenix statuette, Statue Dragonfly tamer*. 2021.

URL: http://https://www.artec3d.com/3d-models (visited on 09/30/2021) 9.

[BCG*13] BREZOVSKY, JAN, CHOVANCOVA, EVA, GORA, ARTUR, et al. "Software tools for identification, visualization and analysis of protein tunnels and channels". *Biotechnology advances* 31.1 (2013), 38–49 2.

[BFS00] BIASOTTI, SILVIA, FALCIDIENO, BIANCA, and SPAGNUOLO, MICHELA. "Extended Reeb Graphs for Surface Understanding and Description". *Discrete Geometry for Computer Imagery*. Ed. by BORGEFORS, GUNILLA, NYSTRÖM, INGELA, and di BAJA, GABRIELLA SANNITI. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, 185–197. ISBN: 978-3-540-44438-1 8.

[BR21] BÆRENTZEN, ANDREAS and ROTENBERG, EVA. "Skeletonization via Local Separators". *ACM Trans. Graph.* 40.5 (Sept. 2021). ISSN: 0730-0301. DOI: 10.1145/3459233. URL: https://doi.org/10.1145/3459233 9.

[CJG18] CHEN, JIA, JESTER, JAMES, and GOPI, M. "Fast Computation of Tunnels in Corneal Collagen Structure". *Proceedings of Computer Graphics International 2018*. CGI 2018. Bintan, Island, Indonesia: Association for Computing Machinery, 2018, 57–65. ISBN: 9781450364010. DOI: 10.1145/3208159.3208175. URL: https://doi.org/10.1145/3208159.3208175 3.

[DEG09] DIAZ-GUTIERREZ, P., EPPSTEIN, D., and GOPI, M. "Curvature Aware Fundamental Cycles". *Computer Graphics Forum* (2009). ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2009.01580.x 3.

[DFW13] DEY, TAMAL K., FAN, FENGTAO, and WANG, YUSU. "An Efficient Computation of Handle and Tunnel Loops via Reeb Graphs". *ACM Trans. Graph.* 32.4 (July 2013). ISSN: 0730-0301. DOI: 10.1145/2461912.2462017. URL: https://doi.org/10.1145/2461912.2462017 2–4, 9–11.

[DLS07] DEY, TAMAL K., LI, KUIYU, and SUN, JIAN. "On Computing Handle and Tunnel Loops". *Proceedings of the 2007 International Conference on Cyberworlds*. CW '07. USA: IEEE Computer Society, 2007, 357–366. ISBN: 0769530052 2.

[DLSC08] DEY, TAMAL K., LI, KUIYU, SUN, JIAN, and COHEN-STEINER, DAVID. "Computing Geometry-Aware Handle and Tunnel Loops in 3D Models". *ACM Trans. Graph.* 27.3 (Aug. 2008), 1–9. ISSN: 0730-0301. DOI: 10.1145/1360612.1360644. URL: https://doi.org/10.1145/1360612.1360644 3.

[DN09] DORAISWAMY, HARISH and NATARAJAN, VIJAY. "Efficient Algorithms for Computing Reeb Graphs". *Comput. Geom. Theory Appl.* 42.6–7 (Aug. 2009), 606–616. ISSN: 0925-7721. DOI: 10.1016/j.comgeo.2008.12.003. URL: https://doi.org/10.1016/j.comgeo.2008.12.003 8.

[dVE06] De VERDIÈRE, ÉRIC COLIN and ERICKSON, JEFF. "Tightening Non-Simple Paths and Cycles on Surfaces". *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*. SODA '06. Miami, Florida: Society for Industrial and Applied Mathematics, 2006, 192–201. ISBN: 0898716055 3.

[Epp03] EPPSTEIN, DAVID. "Dynamic Generators of Topologically Embedded Graphs". *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '03. Baltimore, Maryland: Society for Industrial and Applied Mathematics, 2003, 599–608. ISBN: 0898715385 3.

[Eri12] ERICKSON, JEFF. "Combinatorial optimization of cycles and bases". *Advances in Applied and Computational Topology* 70 (2012), 195–228 2.

[EV97] EL-SANA, JIHAD and VARSHNEY, AMITABH. "Controlled Simplification of Genus for Polygonal Models". *Proceedings of the 8th Conference on Visualization '97*. VIS '97. Phoenix, Arizona, USA: IEEE Computer Society Press, 1997, 403–ff. ISBN: 1581130112 3.

[EW05] ERICKSON, JEFF and WHITTLESEY, KIM. "Greedy Optimal Homotopy and Homology Generators". *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '05. Vancouver, British Columbia: Society for Industrial and Applied Mathematics, 2005, 1038–1046. ISBN: 0898715857 3, 4.

[GW01] GUSKOV, IGOR and WOOD, ZOË J. "Topological Noise Removal". *Proceedings of Graphics Interface 2001*. GI '01. Ottawa, Ontario, Canada: Canadian Information Processing Society, 2001, 19–26. ISBN: 0968880800 3.

[HSKK01] HILAGA, MASAKI, SHINAGAWA, YOSHIHISA, KOMURA, TAKU, and KUNII, TOSIYASU. "Topology matching for fully automatic similarity estimation of 3D shapes". Jan. 2001, 203–212. DOI: 10.1145/383259.383282 10.

[Kut06] KUTZ, MARTIN. "Computing Shortest Non-Trivial Cycles on Orientable Surfaces of Bounded Genus in Almost Linear Time". *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*. SCG '06. Sedona, Arizona, USA: Association for Computing Machinery, 2006, 430–438. ISBN: 1595933409. DOI: 10.1145/1137856.1137919. URL: https://doi.org/10.1145/1137856.1137919 3.

[Mun84] MUNKRES, J.R. *Elements Of Algebraic Topology*. Advanced book classics. Perseus Publishing, 1984. ISBN: 9780201054873 2.

[PSBM07] PASCUCCI, VALERIO, SCORZELLI, GIORGIO, BREMER, PEER-TIMO, and MASCARENHAS, AJITH. "Robust On-line Computation of Reeb Graphs: Simplicity and Speed". *ACM SIGGRAPH 2007 Papers*. SIGGRAPH '07. San Diego, California: ACM, 2007. DOI: 10.1145/1275808.1276449. URL: http://doi.acm.org/10.1145/1275808.1276449 8–10.

[SKK91] SHINAGAWA, Y., KUNII, T. L., and KERGOSIEN, Y. L. "Surface coding based on Morse theory". *IEEE Computer Graphics and Applications* 11.5 (Sept. 1991), 66–78. ISSN: 1558-1756. DOI: 10.1109/38.90568 8.

[SKN10] SOMAN, JYOTHISH, KOTHAPALLI, KISHORE, and NARAYANAN, P J. "Some GPU Algorithms For Graph Connected Components and Spanning Tree". *Parallel Processing Letters* 20.04 (2010), 325–339. DOI: 10.1142/S0129626410000272. eprint: https://doi.org/10.1142/S0129626410000272. URL: https://doi.org/10.1142/S0129626410000272 8.

[SL01] SHATTUCK, D. W. and LEAHY, R. M. "Automated graph-based analysis and correction of cortical volume topology". *IEEE Transactions on Medical Imaging* 20.11 (2001), 1167–1177. DOI: 10.1109/42.963819 3.

[SMS*20] STADLBAUER, P., MLAKAR, D., SEIDEL, H.-P., et al. "Interactive Modeling of Cellular Structures on Surfaces with Application to Additive Manufacturing". *Computer Graphics Forum* 39.2 (2020), 277–289. DOI: https://doi.org/10.1111/cgf.13929. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13929. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13929 4, 5.

[TFL*18] TIERNY, JULIEN, FAVELIER, GUILLAUME, LEVINE, JOSHUA A., et al. "The Topology ToolKit". *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2018), 832–842. DOI: 10.1109/TVCG.2017.2743938 8.

[VG10] VOSS, NEIL R and GERSTEIN, MARK. "3V: cavity, channel and cleft volume calculator and extractor". *Nucleic acids research* 38.suppl_2 (2010), W555–W562 2.

[WHDS04] WOOD, ZOË, HOPPE, HUGUES, DESBRUN, MATHIEU, and SCHRÖDER, PETER. "Removing Excess Topology from Isosurfaces". *ACM Trans. Graph.* 23.2 (Apr. 2004), 190–208. ISSN: 0730-0301. DOI: 10.1145/990002.990007. URL: https://doi.org/10.1145/990002.990007 3.

[ZJH07] ZHOU, QIAN-YI, JU, TAO, and HU, SHI-MIN. "Topology Repair of Solid Models Using Skeletons". *IEEE Transactions on Visualization and Computer Graphics* 13.4 (July 2007), 675–685. ISSN: 1077-2626. DOI: 10.1109/TVCG.2007.1015. URL: https://doi.org/10.1109/TVCG.2007.1015 3.

[ZMSS18] ZAYER, RHALEB, MLAKAR, DANIEL, STEINBERGER, MARKUS, and SEIDEL, HANS-PETER. "Layered Fields for Natural Tessellations on Surfaces". *ACM Trans. Graph.* 37.6 (Dec. 2018). ISSN: 0730-0301. DOI: 10.1145/3272127.3275072. URL: https://doi.org/10.1145/3272127.3275072 3–5.