Eurographics Symposium on Geometry Processing 2021
K. Crane and J. Digne
(Guest Editors)

Volume 40 (2021), Number 5

Practical Computation of the Cut Locus on Discrete Surfaces

C. Mancinelli[1] and M. Livesu[2] and E. Puppo[1]

[1]DIBRIS - University of Genoa, Italy
[2]IMATI - National Research Council, Italy

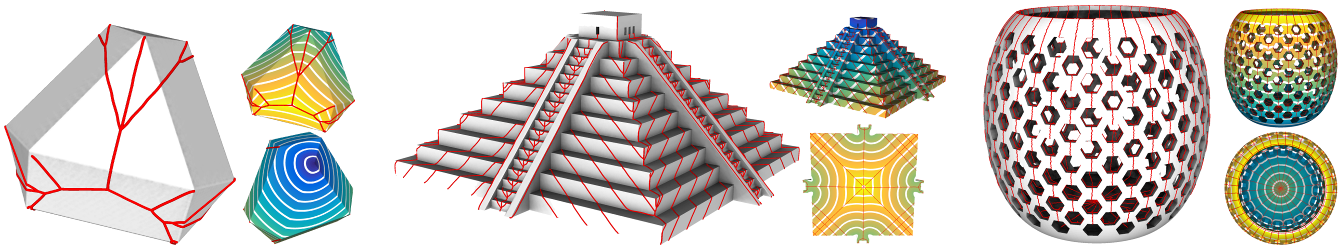

Figure 1: *We can compute the cut locus of the geodesic distance function on any shape. From the simplest ones (left) to the most complex, both geometrically (middle) and topologically (right). Our output can be as smooth as the real cut locus (left, middle), or encoded in the edges of the underlying mesh (right). Both results are practically relevant for applications.*

Abstract

We present a novel method to compute the cut locus of a distance function encoded on a polygonal mesh. Our method exploits theoretical findings about the cut locus and – with a combination of analytic, geometric and topological tools – it is able to compute a topologically correct and geometrically accurate approximation of it. Our result can be either restricted to the mesh edges, or aligned with the real cut locus. Both outputs may be useful for practical applications. We also provide a convenient tool to optionally prune the weak branches of the cut locus, simplifying its structure. Our approach supersedes prior art, in that it is easier to use and also orders of magnitude faster. In fact, it depends on just one parameter, and it flawlessly operates on meshes with high genus and very high element count at interactive rates. We experiment with different datasets and methods for geodesic distance estimation. We also present applications to local and global surface parameterization.

CCS Concepts

• ***Computing methodologies*** → ***Shape analysis;*** *Mesh models;* • ***Mathematics of computing*** → *Continuous functions;*

1. Introduction

Computational problems in the geodesic metric of Riemannian manifolds are becoming more and more relevant in geometry processing [AOCBC15, KCPS15, MRCK21, Sch13, SSC19b], optimal transport [BvdPPH11, LD11, SRGB14, SDGP*15, Sol18], and machine learning [BYF*19, MKK21, MBBV15, RGA*20, SRC*20]. Many results, however, make assumptions about well definiteness of differential quantities of the distance function, uniqueness of shortest paths, injectivity of the exponential map, etc. But such properties are violated at the *cut loci* of points.

Roughly speaking, the cut locus of a point x consists of all points that can be connected to x with more than one shortest path, and it may have a non-trivial structure even for the simplest shapes [GMST05]. Moreover, the minimum distance of the cut locus from the source x can be arbitrarily small [Sak97]. Therefore, methods that make assumptions about staying away from the cut locus, without properly knowing it, might be hindered not just on a global basis, but on a local basis too. For example, the cut locus sets a tight bound to the injectivity of the exponential map in local surface parametrization [HA19] and it influences the smoothness of the solution to the Monge problem in optimal transport [Vil08, Vil11].

Curiously enough, there exist very few algorithms for computing the cut locus. Besides, existing algorithms are slow, or dependent on several parameters, or limited to specific classes of surfaces, or suffer from all such limitations together (see Sec. 2).

In this paper, we focus on real analytic surfaces represented with polygonal meshes, and propose a practical and efficient algorithm for finding the cut locus in this setting. Our algorithm is independent of the method used to estimate the distance function, which is taken in input at the vertices of the mesh. We follow a mixed analytical, geometric, and topological approach. Starting from the point farthest from the source (which surely belongs to the cut locus), we exploit facts about the Laplacian of the distance function to grow a spanning tree that floods the entire mesh, and locally aligns with the cut locus. The method relies on a unique parameter that the user can tune interactively to obtain an approximation of the cut locus by pruning the spanning tree. For surfaces of genus higher than zero, we restore the correct topology by closing the necessary loops. As a result, we obtain a cut locus that is geometrically approximated, being made of edges of the mesh; and topologically accurate, having the same homology of the underlying manifold. For geometrically more accurate results, we also provide an algorithm to smooth the curves that compose the cut locus.

Our method has three key practical advantages with respect to prior art: (i) the user can intuitively tweak a single parameter in real time with immediate feedback; (ii) it is orders of magnitude faster, thus permitting to operate on discrete manifolds with much higher complexity, both geometric and topological; (iii) it is independent of the algorithm used to compute the distance function, thus allowing the user to trade-off between accuracy and speed.

We evaluate our method on several shapes as well as different algorithms for computing the distance function. We demonstrate its applications to the computation of the radius of injectivity of the exponential map, and to visibility-aware mesh cutting for texture mapping. Besides, we believe this is the first practical tool that allows to compute the cut locus on general meshes. Considering the amount of precious information that the cut locus encodes in a concise yet low dimensional structure, we expect scholars from the geometry processing community to readily adopt our tool, with new applications arising to complement the proposed ones. The code is released in the public domain at `https://github.com/Claudiomancinelli90/CutLocus`.

2. Related Works

Theory. Since Poincaré introduced the concept of cut locus [Poi05], this subject was studied by different researchers under different perspectives and in different times. Results are sparse, quite complex, and often subject to strong assumptions on the manifold.

Concerning topological properties, Myers proved that the cut locus of a closed real-analytic surface is a graph with a finite number of branches [Mye35]. Buchner generalized this result to higher dimensions, and proved that in dimension two the local structure is that of a tree [Buc77]. Sakai stated a strong equivalence between the homology of the manifold and that of the cut locus [Sak97]. We leverage such results to ensure that the estimated cut locus has the correct local structure and global topology.

Concerning differential properties, it is well known that the distance function is smooth everywhere but at the source x and at the cut locus [Sak97]. Second order differential properties at the cut locus have been investigated only more recently in a weak sense, like in the sense of distributions, or of viscosity, or of barriers [GOV20, MMU14, Nee07]. Roughly speaking, all such methods study the behavior at the cut locus by approximating the distance function arbitrarily well with a smooth function. Mantegazza and colleagues show the equivalence of analyses in the sense of distributions and of viscosity, and that the analysis in the sense of barriers implies the other two [MMU14] . In this context, Générau shows that indeed the Laplacian of the distance function is $-\infty$ in the sense of barriers at the cut locus [Gén20]. We apply the latter result to locally align to the cut locus.

To the best of our knowledge, the only analytical solutions for the cut locus – except the trivial case of the sphere – were given in [IK04] for the case of a tri-axial ellipsoid, and in [GMST05] for the case of a torus of revolution.

Algorithms. The first two tools for computing the cut locus that were proposed in the literature are Loki [ST02] and Thaw [IS04]. They both have limited capabilities, and were mainly developed with the purpose of supporting theoretical investigations. Loki is based on a polynomial approximation of the exponential map defined from a periodic parametrization of the surface, and supports only surfaces with genus one. Thaw supports only convex surfaces.

Misztal and colleagues [MBAM11] compute a retraction of the surface to the cut locus by means of an advecting front, represented as a piecewise linear curve. The cut locus is detected from self-intersections of the propagated front. Geodesics are computed by relying on a local parametrization, and using finite differences. Results are presented just on parametrized tori, and reported running times are about half an hour. While in principle this method scales to arbitrary surfaces, such an extension would require developing more general ways to compute geodesics, and to propagate the advecting front. Specifically, many geodesics are radially cast during front propagation, hence it may become very expensive to maintain the front accurate, especially far from the source, where it stretches.

Dey and Li [DL09] compute a subset of the cut locus, defined as the set of points where two minimizing geodesics meet after spreading apart by a certain amount. Geodesics are discretized with shortest paths on a graph, and are traced from all pairs of sampled points that lie closer than a given threshold. If two geodesics starting at nearby points diverge, then such points are added to the cut locus. The method depends on several parameters, but it is proven to converge to the cut locus by increasing the density of sampling and reducing the thresholds for distances between adjacent points and spread between geodesics. The output is just a discrete collection of points, which may be connected to form a complex. The authors report about ten minutes to compute the cut locus on a mesh of about 280K triangles.

Générau and colleagues [Gén20, GOV20] provide a comprehensive account of the subject, and propose a method to compute an approximation of the cut locus on a rigorous mathematical basis. They define the λ-*cut locus* as a regularized subset of the cut locus. They prove that such set can be approximated arbitrarily well with the solution of a variational problem, depending in turn on another parameter m. The solution converges to the exact cut locus when m goes to infinity and λ goes to zero. The approximation is obtained by resolving the variational problem with finite element methods.

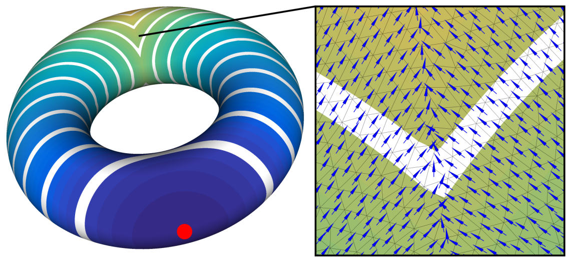

Figure 2: *The distance function emanating from the red source is smooth everywhere except at the cut locus, where its gradient is discontinuous (closeup).*

The method can work on general surfaces, but results are presented just on simple shapes (multi-tori of genus 1, 2, 3); reported times are of about an hour to process a mesh with 100K triangles.

None of these previous techniques is capable of combining the performances of our method with its ability to compute the cut locus on discrete manifolds with any geometric or topological complexity. In Sec. 7, we validate our geometric accuracy by comparing against the ground truth on the torus [GMST05], as well as against numerical methods that converge to the exact solution [DL09, Gén20].

3. Preliminaries

We give the formal definition of cut locus and we report some relevant facts about it. The reader not familiar with basic concepts of Riemannian geometry can refer to [GHL04, Sak97].

Let $\mathcal{M}$ be a compact real-analytic Riemannian 2-manifold (i.e., a closed surface) endowed with a smooth metric. The metric allows us to define the length of any smooth curve γ on $\mathcal{M}$. For any pair of points $x, y \in \mathcal{M}$, their *geodesic distance* $d(x,y)$ is defined as the length of the shortest curve γ on $\mathcal{M}$ joining x and y. Whenever x is fixed, we define its *distance function*

$$d_x : \mathcal{M} \to \mathbb{R} \qquad d_x(y) := d(x,y).$$

The *cut locus* $\mathcal{C}(x)$ of x can be defined geometrically or analytically in two equivalent ways as (the closure of) the locus of points where: a shortest curve to x fails to be unique; the distance function d_x fails to be differentiable. A point of the cut locus will be called a *cut point*. The following properties are relevant to our work:

(P1) Let Δd_x be the Riemannian Laplacian of the distance function, then Δd_x is $-\infty$ in the sense of barriers at the cut locus [Gén20];
(P2) The cut locus is a finite graph, having the local structure of a tree [Buc77];
(P3) The cut locus has the same homology as $\mathcal{M}$: it is a tree for genus zero surfaces, and it contains $2g$ cycles otherwise, with g being the genus of $\mathcal{M}$ [Sak97];
(P4) The cut locus is piece-wise smooth. Specifically, it is C^∞ at all regular points [MM02].

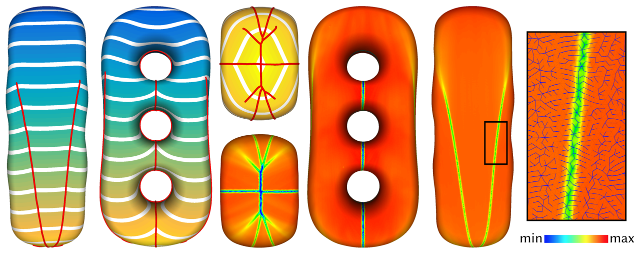

Figure 3: *Left: cut locus of a three torus with respect to a distance function sourced at its topmost vertex. Right: the valleys of the Laplacian of the distance function clearly demarcate the paths of the cut locus. We catch these paths with a spanning tree that grows by locally prioritizing lower Laplacian values (closeup).*

4. Method

Our method alternates discrete differential geometry and topology computations. We take in input an approximation of a smooth surface $\mathcal{M}$ represented with a polygonal mesh $M = (V, F)$, where V and F are the vertices and faces, respectively. Without lack of generality, we assume all faces of F to be triangles. In the following, $x \in M$ is the *source* for which we evaluate the cut locus, and d_x is the distance function from it. We assume that a method is given to compute the distance function from any given point on M to all the vertices of V, as this computation is orthogonal to our contribution. In Section 7 we experiment with different methods.

Our method consists of two steps, plus one optional smoothing step. The steps jointly address the properties (P1-P4) listed in Sec. 3. We first compute the cut locus in the form of a tree, exploiting its relation with the Laplacian of the distance function (Property P1) and its local structure (Property P2). This step is already sufficient to provide a valid solution for objects of genus zero. In the second step, we ensure the correct homology for objects of higher genus (Property P3). In the third (optional) step, we smooth the cut locus, following the gradient of the distance function (Property P4). In the following subsections we provide the technical details.

4.1. Cut locus from spanning tree

We know that the distance function d_x is not differentiable at the cut locus. In fact, the gradient ∇d_x points towards the cut locus and breaks at it, as illustrated in Fig. 2. The distance function d_x can be approximated arbitrarily well with a smooth function $\tilde{d}_x$ (barrier), whose Laplacian diverges to $-\infty$ at the cut locus as the approximation improves (Property P1). Therefore, if we estimate the discrete Laplacian of the distance function sampled at the vertices of M, we expect it to become highly negative close to the cut locus. We base our construction on the above observation, designing a spanning tree that floods the entire mesh while aligning to the valleys of the discrete Laplacian, and then pruning its branches in order to retain only the portions of it that are at the cut locus (Fig. 3).

Laplacian and gradient estimators. We adopt a quadratic estimator for all first and second order differential quantities of function

d_x in the Riemannian metric [WYLC13]. This approach is known to converge to the true values of the sampled function as the average length of edges tends to zero. The Laplacian Δd_x is a fundamental component of the method, whereas the gradient ∇d_x is used for filtering, and can also be substituted with an alternative filter based on the Laplacian itself. This method is stable and reliable even on moderately irregular meshes. If the meshing is regular, then a classical cotan Laplacian [PP93] could be a simpler yet valid alternative. Conversely, linear estimators of the gradient result fragile close to singularities, hence at the cut locus [MLP19].

We briefly review the method, referring to [WYLC13] for further details, and we discuss our implementation. The metric tensor of the underlying surface at any given point y is estimated first, by considering a set of points $(y_0,\dots,y_{k-1})$ in its neighborhood, and fitting a quadric Q to describe the portion of surface around y. The partial derivatives of Q in a local reference system centered at y are used to approximate the metric tensor g and its inverse g^{-1}. Next, five vectors $\mathbf{a}_0,\dots,\mathbf{a}_4$ are computed, which also depend on Q, and estimate the partial (Euclidean) derivatives at y of any real-valued function f sampled at $(y,y_0,\dots,y_{k-1})$, with respect a local reference system (u,v) at y, as:

$$\frac{\partial f}{\partial u} \approx \mathbf{a}_0\mathbf{f}_y, \qquad \frac{\partial f}{\partial v} \approx \mathbf{a}_1\mathbf{f}_y,$$
$$\frac{\partial^2 f}{\partial u^2} \approx \mathbf{a}_2\mathbf{f}_y, \qquad \frac{\partial^2 f}{\partial v \partial u} \approx \mathbf{a}_3\mathbf{f}_y, \qquad \frac{\partial^2 f}{\partial v^2} \approx \mathbf{a}_4\mathbf{f}_y$$

where $\mathbf{f}_y = (f_y, y_0,\dots,y_{k-1})$ is the sampled function. All the above quantities depend only on M and are computed once for all its vertices during preprocessing. The metric tensor, its inverse, and the vectors $\mathbf{a}_i$ are encoded for all vertices in sparse matrices. Given any function $\mathbf{f}$ sampled at the vertices of M, its Riemannian differential quantities of the first and second order are computed at all vertices of M with simple matrix-vector multiplications. We exploit this construction to evaluate the Riemannian gradient and the Laplace-Beltrami operator of d_x.

In order to ensure a reliable estimate, we have noticed that particular care must be taken in choosing the neighbors of each vertex y. To obtain a fairly uniform sampling, we consider an extended neighborhood, computed as depicted in Fig. 4. Note that neighbors, other than vertices in the 1-ring, are expressed with barycentric coordinates with respect to the triangles containing them. In terms of building the quadric Q and the $\mathbf{a}_i$ vectors, this means that we will use the vertices of triangles in the extended 1-ring of Fig. 4 with different weights.

ORG Spanning Tree. Knowing that the cut locus has the local structure of a tree (Property P2), to retain its branches we rely on the construction of an Ordered Region Growing (ORG) spanning tree T. For this, we were inspired by techniques that extract line structures from higher dimensional data, such as blood vessels from medical images [YCS00], and curve-skeletons [LGS12].

We set the root of T at the mesh vertex y^* that maximizes function d_x. Being the global maximum, this point is guaranteed to be in the cut locus. We then initialize a priority queue $\mathcal{Q}$ with y^*, and we grow T by iteratively extracting the top element from $\mathcal{Q}$, creating new arcs with all its neighbors that have not been included in T

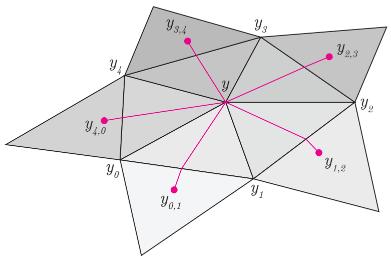

Figure 4: *To compute the Laplacian at a vertex y with valence k, we consider an extended 1-ring consisting of $2k$ points. We enrich the set of vertices in the 1-ring of y, with additional k points computed as follows: for each triangle y, y_i, y_{i1+1}, we trace a geodesic line with length ℓ_y along the bisector at y, where ℓ_y is the 1-ring average edge length. This locates an additional point $y_{i,i+1}$ inside the triangle opposite to y along edge $y_i y_{i+1}$. In case the line extends beyond the boundary of such triangle, we just clamp it at its border.*

yet, and inserting such points into $\mathcal{Q}$. The process stops when the whole mesh is flooded, hence the queue $\mathcal{Q}$ becomes empty. We set the priority of each point y at $-\Delta d_x(y)$. This ensures that the tree expands following the deepest valleys of the Laplacian, thus aligning to the branches of the cut locus (Property P1). See the closeup to the right side of Fig. 3.

Thinning. By construction, tree T spans the whole surface. Since we are operating in a discrete setting, if parallel branches of T are connected by transversal edges of mesh M, we consider them to span the portion of surface between them. On the other hand, we know that the cut locus has Hausdoff dimension one (Property P2). Therefore, we should not allow nodes of T to contain in their 1-ring in M any node of T, other than their parent and siblings in T. We prune T by removing the weaker nodes (in terms of their Laplacian) that do not fulfill this property. In order to preserve the integrity of the tree, we apply a thinning filter, akin [RKS00], again prioritized on the value of the Laplacian. We insert all the leaves y of T in a priority queue $\mathcal{Q}$, this time by using $\Delta d_x(y)$ as priority. When an element y is extracted from $\mathcal{Q}$, we check whether any of its neighbors, different from its parent and siblings in T, also belong to T. If there exits one such neighbor y' such that $\Delta d_x(y') < \Delta d_x(y)$, then we remove y from T, and we add its parent to $\mathcal{Q}$ if it has become a leaf node. Although this filter does not guarantee to fulfill the constraint above everywhere, it maintains the integrity of the tree, avoiding to discard whole branches which might contain strongly negative values of the Laplacian, just because some of their intermediate nodes are weak. In practice, we found it to produce better results than other thinning strategies that we have tried.

Filtering. Having flooded the entire mesh, even after thinning, the tree T contains many spurious branches, which do not belong to the cut locus. Spurious branches are not easy to remove automatically, because the paths of the cut locus may be very unstable. Consider for instance the example in Fig. 5. It is well known that the cut locus of a point x on a sphere consists just of its antipodal point $\bar{x}$. However, if the sphere is perturbed with a bump, the cut locus of x is extended to a geodesic curve that connects the top of the bump to

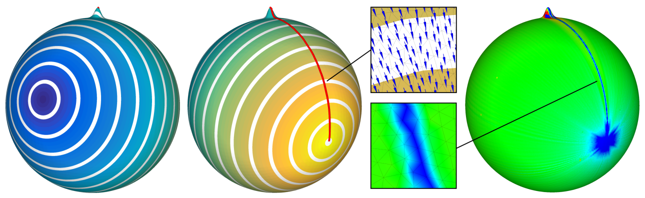

Figure 5: *On a sphere with a bump, the cut locus of a point x (blue) is a geodesic line from the top of the bump to the antipodal point $\bar{x}$ (yellow). Despite weak in terms of discontinuity of gradients, our Laplacian-based detection system clearly defines a valley along the whole cut locus (closeups). Branches that accumulate around the antipodal point $\bar{x}$ may be filtered out.*

$\bar{x}$. Note that the bump can be arbitrarily small and arbitrarily close to x, hence the cut locus on the bumpy sphere can be as extended as a maximal semi-circumference.

Since we are working on a discrete approximation M of the manifold $\mathcal{M}$, we may easily miss existing bumps or, conversely, create artificial bumps. Branches caused by small bumps will be weak in terms of Laplacian, since geodesic lines from x reach them at a very narrow angle, hence easily confused with noise.

We thus decided to let the user clean the tree T from spurious branches, pruning it from the leaves, by means of a simple filter that can be tuned interactively in real time. The filter is controlled with a geometric parameter, which is independent of the specific dataset and is very intuitive: the angle between gradients that approach the cut locus from opposite sides. For each leaf node y of $G_{\mathcal{C}}$, we consider the gradients $\nabla d_x(y_i)$ for all neighbors y_i in the 1-ring of y, we parallel transport such gradients to the tangent plane at y, and we compute the maximum angle between any two of them. If this angle is below a given threshold θ, then we prune y from T, and we proceed recursively along the branch it belongs to. To parallel transport vectors, we use the method proposed in [KCPS13].

Note that, by pushing threshold θ towards relatively large angles, we may also remove the weaker branches of the cut locus, where function d_x is *nearly* differentiable, thus obtaining progressively simpler cut loci (Fig. 7). This may be useful in a variety of applications, since it is equivalent to consider a smoothed version of the manifold, or of its distance function.

An alternative to the above filter consists of pruning the tree based on a threshold on the value of the Laplacian. While the two filters provide similar results, we privilege the one based on gradients because it is most intuitive and independent of the underlying shape. However, in some cases the filter based on the Laplacian allows the user to obtain better results. We therefore support switching between the two filters, or combining them.

Moreover, the filtering process can use two alternative policies: one based on pruning, in which the leaves that exceed the threshold are recursively removed; and the other based on growing, in which the tree is expanded from the root, including nodes that fulfill the given threshold. Pruning is more conservative, while growing is more aggressive. Consider again the example in Fig. 5: both the gradient and the Laplacian are “strong" only at the antipodal point and at the tip of the bump. If pruning is applied, then the whole ridge shown in the image would be retained; conversely, if growing is applied, just the antipodal point would be retained, and the bump would be deemed noise. The default policy, which has been applied in most of our experiments, is the conservative one. However, with models containing many tiny details, such as the ones in Fig. 13, the aggressive strategy provides cleaner results.

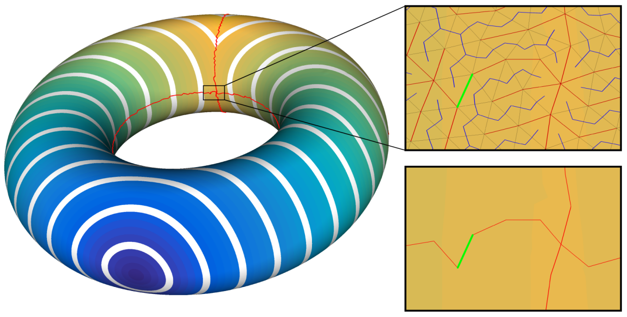

Figure 6: *The cut locus has the same homology of the underlying manifold. We use a field-aware variant of the tree cotree algorithm [EW05] to transform the spanning tree into a system of loops, always guaranteeing the correct topology. In the upper closeup, red and blue lines denote the tree and cotree, respectively. Light green edges are the homology generators. The bottom closeup shows a detail of the output cut locus.*

In our experiments, we experienced the presence of spurious tiny branches incident at the main ridges even after filtering. We provide an additional filter, which can be used to remove short branches to obtain clean paths. The filter can be tuned either based on the number of edges in a branch (for high resolution meshes, we found that removing branches shorter than three edges is a reasonable choice in all cases), or on the length of the branch relative to the size of the object (for coarser meshes, removing branches shorter than 0.01 the length of the diagonal of the bounding box is a reasonable choice).

The filtered tree $T_{\mathcal{C}}$ provides our final approximation of the cut locus for objects of genus zero.

4.2. Homology

If M has genus $g > 0$, we know that its cut locus must have the same homology (Property P3), thus it must contain exactly $2g$ cycles. To restore the correct homology, we employ a variation of the greedy homotopy basis algorithm proposed in [EW05]. The original algorithm finds the shortest homotopy basis centered at a mesh vertex in $\mathcal{O}(n \log n)$, by first growing a shortest spanning tree emanating from a source node, and then growing a spanning tree in the dual mesh (a.k.a. cotree [Epp03]), covering the edges not in the primal tree. This procedure leaves exactly $2g$ edges that are covered neither by the primal nor by the dual trees. These edges are the generators of the homology basis, and – bridging disjoint branches in the primal tree – form the wanted system of loops.

In our specific case, we are interested in finding the system of

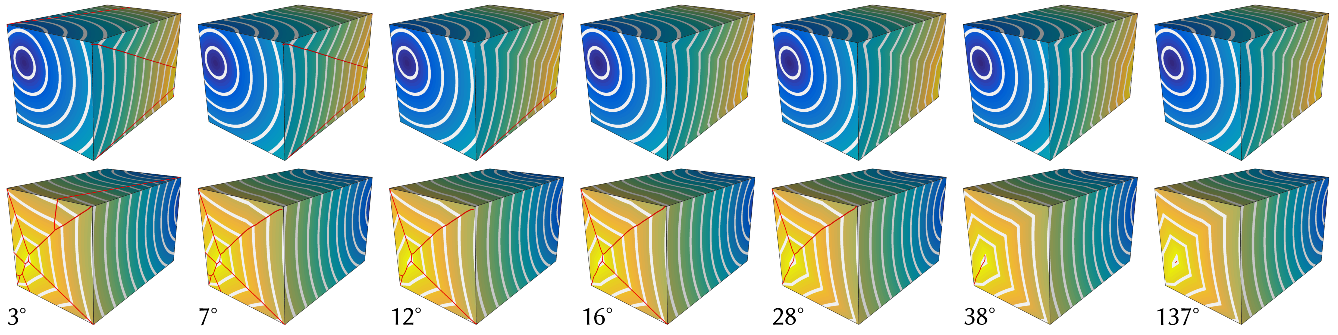

Figure 7: *Increasing the threshold on the gradient angle, we obtain progressively simpler version of the cut locus, where "weak" branches that are nearly differentiable are pruned. Pushing the threshold to the extreme, we retain only the point antipodal to the source (right).*

loops that best aligns with the cut locus. To this end, we initialize the tree with $T_{\mathcal{C}}$, and extend it so as to flood the entire mesh with a new tree $\bar{T}$. Next, we set the root of the cotree $\bar{C}$ at one of the triangles incident to the source x, and we expand it across edges that do not belong to the primal tree $\bar{T}$. Our goal is to position the homology generators as close as possible to the cut locus. To do so, we ensure that both the tree and the cotree grow at constant speed in all directions, thus forcing opposite fronts of the trees to cover roughly the same distance before they collide. We obtain the desired result by prioritizing the growth of the tree $\bar{T}$ with values of d_x, and the growth of the cotree $\bar{C}$ with $-d_x$ (interpolating the function at the centroid of each triangle). In most cases, the endpoints of a generator are already nodes of $T_{\mathcal{C}}$, and it is sufficient to add such edges to the graph to close the loop (Figure 6). In case a generator e is not connected to $T_{\mathcal{C}}$ at its endpoints, we climb the branches of $\bar{T}$ until we reach a node of $T_{\mathcal{C}}$, and we close the loop by adding the corresponding paths to $T_{\mathcal{C}}$.

In the latter case, the portion of cut locus that closes the loop is a subset of tree $\bar{T}$, hence misaligned with the valleys of the Laplacian. We explain here why our choice to grow $\bar{T}$ according to values of d_x not only does not sacrifice geometric accuracy, but rather enhances it. Remembering that the Laplacian Δd_x is the divergence of the gradient ∇d_x, the previous step of the algorithm is extremely effective at detecting *strong* branches of the cut locus, characterized by strongly convergent gradients, but is less effective at detecting *weak* branches, characterized by nearly parallel gradients. Weak branches of the cut locus, which may have been missed in the previous step (e.g., due to aggressive thresholding), roughly align with the local gradient, which provides a valid guidance for tracing them (if a valid starting point in the cut locus is known).

Tree $\bar{T}$ is initialized with the subset of the cut locus selected at the previous step, and is then expanded according to growing values of d_x, hence aligning with ∇d_x. As a whole, this hybrid tree aligns to the valleys of the Laplacian where the gradients clearly converge, and to the gradients where they are nearly parallel. This allows us to have the best of both indicators, using each one of them in the places where it is more appropriate. With this technique, we were able to reconstruct high quality cut loci of shapes with non trivial topology even operating with approximated distance functions, often characterized by less accurate Laplacian fields (Fig. 12). The

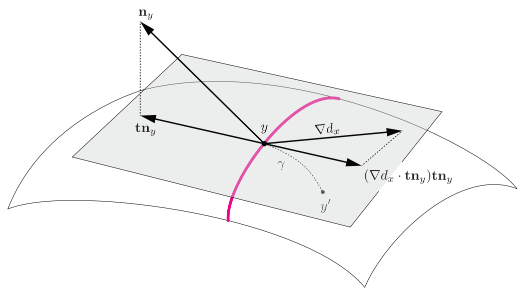

Figure 8: *Construction for gradient-driven smoothing. We estimate the discrete tangential normal* $\mathbf{tn}_y$ *of the cut locus at y. The displacement vector is proportional to the component of the gradient* $\nabla d_x(y)$ *along* $\mathbf{tn}_y$. *Curve* γ *is a geodesic line cast from y in direction* $\mathrm{sign}(\nabla d_x(y) \cdot \mathbf{tn}_y)\mathbf{tn}_y$ *for length* $\lambda|\nabla d_x(y) \cdot \mathbf{tn}_y|$.

result of this step is a graph $G_{\mathcal{C}}$ providing a discrete approximation of the cut locus.

4.3. Smoothing

Graph $G_{\mathcal{C}}$ is made of edges of mesh M, thus it necessarily contains wiggly paths. This approximation is suitable to many applications, e.g., if we want to prevent any computational technique from crossing the cut locus. However, we know that the cut locus of a smooth manifold $\mathcal{M}$ consists of smooth lines (Property P4). We try to obtain a smoother (yet still piecewise-linear) structure by pushing the nodes of $G_{\mathcal{C}}$ closer to the true cut locus, freeing them from the edges of M, while remaining on its surface.

We apply a tangent space smoothing algorithm driven by gradient ∇d_x. We know that the gradient of d_x is oriented towards the cut locus from both sides of it (Fig. 2); since we actually estimate the gradient of the smooth barrier $\nabla\tilde{d}_x$, we expect its component orthogonal to the cut locus to be null at the cut locus. Thus, we iteratively displace each node y of $G_{\mathcal{C}}$ in the direction of the normal to the curve through y in its tangent plane, for an amount that depends on the component of $\nabla\tilde{d}_x(y)$ along such normal, until such component becomes null. Note that, a displaced node y is no longer a vertex of M, but it lies inside a triangle t_y of M, and it is encoded with barycentric coordinates wrt t_y; we estimate the gradient at y by linearly interpolating the gradients at the vertices of t_y, parallel transported at y. Consecutive nodes are connected with shortest geodesic paths to obtain the final cut locus.

Refer to Fig. 8 for a visual explanation of the smoothing step. Let y_p, y, y_n be three consecutive nodes of $G_\mathcal{C}$, y being a regular node. We first estimate the normal of the 3D curve through y_p, y, y_n with standard finite differences, as in [LBS05], then we project such vector to the tangent plane at y to recover the tangential normal $\mathbf{tn}_y$. We displace y by casting a geodesic path from y in tangent direction $\mathbf{tn}_y$ for a length $\lambda\ell(\nabla d_x(y)\cdot\mathbf{tn}_y)$ where λ is a damping parameter, which is initialized at 0.5 and halved at each iteration, and ℓ is the average edge length. Note that the sign of the dot product may reverse the direction of $\mathbf{tn}_y$ according to the component of the gradient along it. Note also that the geodesic path will hardly cross more than one edge per iteration, while it provides a safe way to follow the intrinsic metric.

For a leaf node y_l the algorithm is analogous, just the displacement of y_l occurs in the opposite direction wrt the parallel transport of the normal at its neighbor. A branching node y_b is simply displaced tangentially towards the centroid if its neighbors on $G_\mathcal{C}$, as the gradient estimated at branching points is usually not reliable.

The proposed approach readily pushes the discrete cut locus to the smooth cut locus in areas where the former has a non vanishing angle with the local gradient $\nabla d_x(y)$. Extremely weak portions of the cut locus are less affected, because the tangent curve and the local gradient are nearly parallel. We eventually apply one step of classical tangent space smoothing to relax these areas, too.

5. Results

We have implemented our algorithm in C++, using libraries Yocto/GL [PNC19], CinoLib [Liv19] and libigl [JP*18] for geometry processing. We have validated our results in a variety of experiments reported in this section. The method performs efficiently on meshes with a size up to a few million triangles, and demonstrates to produce plausible results on all models, correctly locating the cut locus even in places where the distance function seems rather smooth at a visual analysis (see, e.g., Fig. 6 and Fig. 11).

Validation. From a topological standpoint, our method is always guaranteed to produce the correct result, meaning that the cut locus has the same homology of the underlying manifold. From a geometric point of view, the algorithm has a strong theoretical foundation in the continuous, but in practice it relies on a discretization of the Laplace-Beltrami operator, and on heuristics to build the connectivity of the cut locus. Therefore, we cannot guarantee the exact location of the cut locus on the manifold. In Figures 9 and 10, we validate the geometric accuracy of our algorithm by comparing our outputs with the ground truth on a torus [GMST05], and with the output of prior approximated methods, which are guaranteed to converge to the exact solution [DL09, Gén20]. We obtain visually indistinguishable results on the torus, and a very similar result on the kitten. For the latter, based on our understanding and practical experience, we ascribe the tiny differences in the actual paths to the different mesh tessellations.

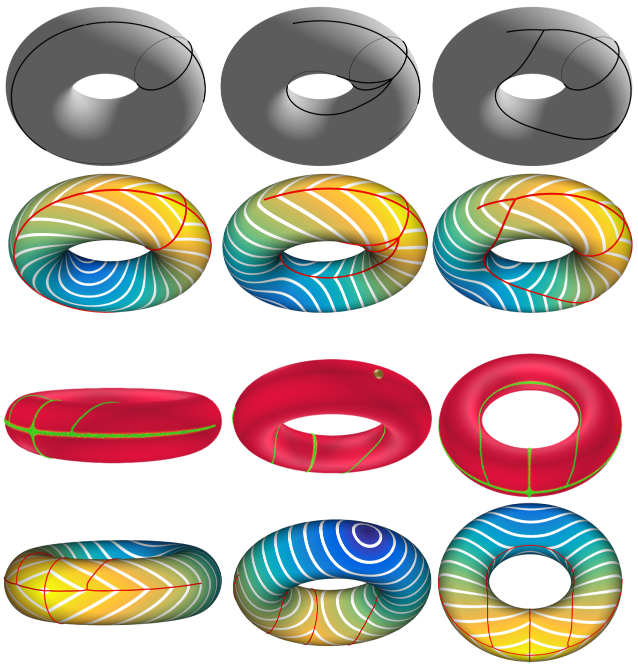
Figure 9: *Visual comparison with the analytic cut loci showed in [GMST05] (top), and with the numerical method described in [Gén20] (bottom). We manually tried to replicate the same sources, obtaining visually indistinguishable results. For the comparison with [Gén20], both his mesh and ours contain* 100*K triangles. We computed our cut locus in less than one second. Our competitor is based on a convoluted numerical FEM scheme, which requires 17 Gauss quadrature points per element. The author declares that the computation terminated in less than 1 hour.*

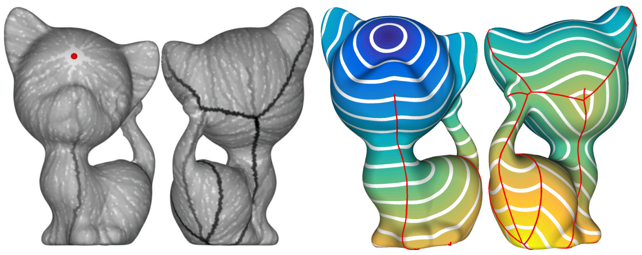
Figure 10: *Visual comparison with the method proposed in [DL09] (left). We manually pinpointed the source on a different mesh. The mesh used in [DL09] contains about 60K triangles and the computation takes about 12 seconds, while our model contains 250K triangles and computation takes about 3 seconds. Reported times in [DL09] are about ten minutes for other meshes of size comparable to ours.*

Scalability and performances. We experimented our method on a variety of meshes, ranging from about 100K to about 2M triangles. High-resolution meshes for all models have been obtained with isotropic remeshing in Meshlab [CCC*08], to warrant a stable estimation of differential properties. All experiments are executed

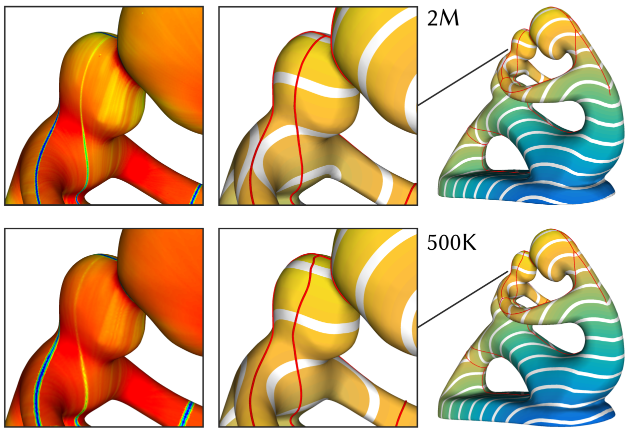

Figure 11: *Mesh resolution may impact the result of our algorithm. In the denser mesh, the Laplacian field (left closeup) clearly identifies a weak path. In the coarse mesh the same path is more blurred. Since that portion belongs to a closed loop, our topological step correctly reconstructs it, but if the same path was open, it would have been difficult to retain it because the gradients are almost parallel, as the level sets of the distance function suggest.*

on a laptop equipped with a 2.9 Ghz quad-core Intel Core i7 and 16 GB RAM, running on a single core. Table 1 reports running times for the various phases of the method. Considering time for pre-processing (once per mesh) and computation of the distance function (once per source point, possible with three different methods), on large meshes our method runs at least two and up to three orders of magnitude faster than any other competitor. The time for the part due to our contribution (i.e., excluding the computation of the distance function) is interactive, and several orders of magnitude faster. Note that this is relevant because, while the other methods require to run the algorithm again from the beginning each time the parameters are changed, we can do parameter tuning interactively after the time consuming part (i.e., computation of the distance function) is complete.

Impact of mesh resolution. As any other technique that approximates continuous entities with a finite discretization, our method performs best on dense and regular samplings. In particular, in our experiments we noticed that even though theory ensures that the Laplacian goes to $-\infty$ at the cut locus, the discrete Laplace-Beltrami may yield values near to zero where the local gradients are almost parallel, hence the cut locus is very weak. In Figure 11 we show a typical failure case. Notwithstanding the feeble signal in the Laplacian, our method was able to reconstruct the corresponding loop and enforce the correct topology. Similar issues that arise outside closed loops cannot be recovered. Considering the computational efficiency of our method, the easiest way to overcome these issues is to operate on dense meshes.

Impact of distance computation. We have experimented with three different methods for computing the distance function: VTP [QHY*16], which provides an efficient variant of the original MMP algorithm [MMP87]; the heat method [CWW13]; and a simple graph-based solver [NPP20]. VTP requires no pre-processing, and provides an exact solution in the polyhedral metric, but it is rather slow on large meshes. The heat method requires solving a linear system of the same size of the mesh. The matrix can be pre-factorized once per mesh, and the solution after factorization amounts to a matrix vector multiplication, hence it is quite fast. It provides an approximated estimate of the distance function for an underlying smooth manifold, and converges to the exact solution as a parameter t tends to zero. We used the implementation in [JP*18] with the default parameter, as recommended by the authors [CWW13]. The graph-based solver relies on a graph with one node per vertex, and one arc for each edge and dual edge of the input mesh. The graph is built during pre-processing, and a solve consists of a Dijkstra-like visit, which is quite fast, too. We have used the implementation in [PNC19]. Compared with the exact polyhedral solution, the accuracies of the heat method and of the graph solver are similar, but with different artifacts, as discussed below.

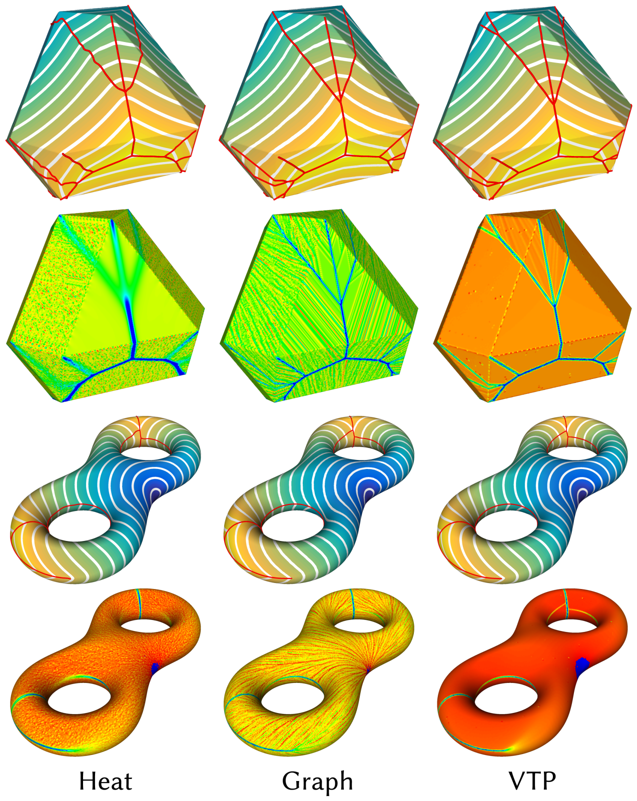

Figure 12: *Cut loci computed with two approximated methods (the heat method [CWW13] and a graph solver [NPP20]), and an exact polyhedral method (VTP [QHY*16]). We show the distance function with the output (top), and its Laplacian (bottom). The exact method produces fields that clearly demarcate the branches of the cut locus. With approximated methods, noise occasionally blurs the Laplacian around the branches of the cut locus.*

All images in the paper, except Fig. 12, were generated by using VTP. In Fig. 12, we compare results obtained with the three methods on two objects. We were able to successfully compute the

			Pre-processing	Distance function			Cut locus computation		Total
model	triangles	genus	min-max	VTP	heat	graph	spanning tree	homology	min-max
torus	100K	1	1.13 - 2.78	0.78	0.03	0.01	0.06	0.10	0.17 - 0.95
diamond	160K	0	1.86 - 5.46	3.11	0.06	0.01	0.12	–	0.13 - 3.23
block	170K	0	1.90 - 5.62	3.20	0.06	0.01	0.13	–	0.14 - 3.33
2-torus	200K	2	2.35 - 6.41	2.55	0.07	0.01	0.16	0.21	0.38 - 2.91
3-torus	200K	3	2.17 - 6.84	2.29	0.07	0.01	0.14	0.19	0.33 - 2.88
raindeer	200K	0	2.45 - 5.08	1.92	0.06	0.02	0.18	–	0.20 - 2.10
sphere	240K	0	2.82 - 10.04	5.64	0.10	0.02	0.22	–	0.24 - 5.86
kitten	250K	1	2.70 - 7.73	2.67	0.09	0.02	0.18	0.24	0.44 - 3.09
nefertiti	460K	0	5.37 - 17.77	3.97	0.19	0.04	0.42	–	0.46 - 4.39
bunny	500K	0	5.52 - 19.45	8.16	0.19	0.04	0.39	–	0.43 - 8.55
fertility	500K	4	5.55 - 19.21	7.92	0.20	0.04	0.41	0.50	0.95 - 8.83
pyramid	660K	0	8.46 - 38.24	11.64	0.30	0.08	0.73	–	0.81 - 13.45
bust	700K	0	9.18 - 39.89	10.65	0.32	0.09	0.78	–	0.87 - 11.43
octopus	800K	0	9.93 - 28.10	14.42	0.31	0.09	0.87	–	0.95 - 15.29
basket	1.1M	260	13.94 - 97.98	13.87	0.62	0.12	1.18	1.36	2.66 - 16.41
fertility	2.0M	4	22.92 - 173.68	66.09	2.99	0.20	1.81	2.10	4.11 - 70.00

Table 1: *Statistics on models used in the experiments and related running times in seconds. Preprocessing includes times for: evaluating metric tensors and related vectors for differential computations; pre-factorization in case the heat method is used; construction of the graph in case the graph method is used (the latter is one order of magnitude smaller than the rest). Spanning tree computation includes also the thinning filter. The total time depends heavily on the method used for the distance function (min with graph, max with VTP) and does not include pre-processing times; graph achieves minimum times even including pre-processing; while heat becomes the most expensive for large meshes because of the cost of pre-factoring. Applying a filter upon parameter tuning works in real time, taking less than 0.01 seconds on all models, and it is not included in the table.*

cut locus with all three methods. Nevertheless, we could observe some interesting differences between the alternative approaches, especially in terms of the Laplacian they yield. VTP consistently produces a neat Laplacian that is completely free from noise. The paths of the cut locus are very sharp and easy to identify. The graph based approach produces a Laplacian with biased noise, where the propagation paths can be clearly identified. Apart from that, the paths of the cut locus remain sharp and easy to identify. On the positive side, the noise-level stripes produced by this method turn out to be useful to trace the weak branches of the cut locus, because they align with the gradient of the function. The heat method was the most challenging for us because, by solving a Poisson problem, it generates a function that is smooth everywhere, cut locus included. This results in a blurred Laplacian field, where strong branches of the cut locus are still clearly identified, but weaker ones fade away (e.g. the three green paths in the upper part of the diamond), or completely disappear (e.g. the closure of the inner loops in the double torus). As a result, the paths of the cut locus computed with the heat method tend to slightly misalign from the other methods, albeit the result is still acceptable for most applications.

In terms of speed, we report timings in Table 1. The cost of VTP is dominant over all other phases of our algorithm. The heat method and the graph solver are quite fast (after pre-processing); the graph method scales better on large meshes, though, in terms of both pre-processing and solve times. All in all, for non time critical applications where quality dominates timing, VTP is by far the best solution. On the other hand, the graph based approach provides a good balancing between accuracy and running times, also proving to scale better than the other methods on meshes containing millions of elements.

Impact of filtering. The interactive filter described in Sec. 4.1 ranks each edge of the ORG spanning tree according to the angle formed by the gradients of the distance function at its two sides. For free branches, this is a quite reliable estimate of their "strength" in the cut locus, namely of whether or not they are caused by relevant features of the model, or just by noise. However, even important branches may fade into regions with nearly parallel gradients.

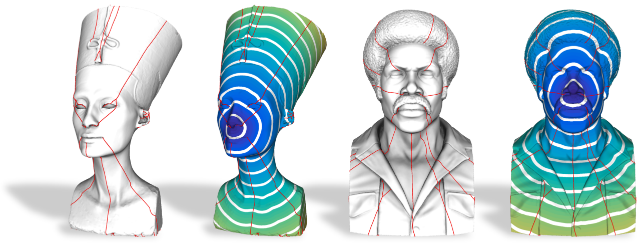
Figure 13: *Our method can cope with complex shapes with small details and a rough surface, too. In these cases, it may be necessary to tweak the filter to a higher threshold in order to remove spurious branches and obtain a clean cut locus.*

The reconstruction of homology loops is quite robust and independent of filtering. We consistently experienced correct reconstructions even if parts of the loops were filtered out before the homology part is performed. On the contrary, weak free branches are quite unstable and may need to be recovered by tweaking the filter properly. The exact position of terminal points of such branches, which are all conjugate points in the exact cut locus, are hard to detect. Therefore, the length of such branches is necessarily approximated. This is especially evident for rough surfaces containing many small details, as the ones depicted in Fig. 13.

Meshes with open boundaries. Our current implementation supports watertight meshes only. In principle, its extension to meshes with open boundaries is straightforward. Of course, the methods

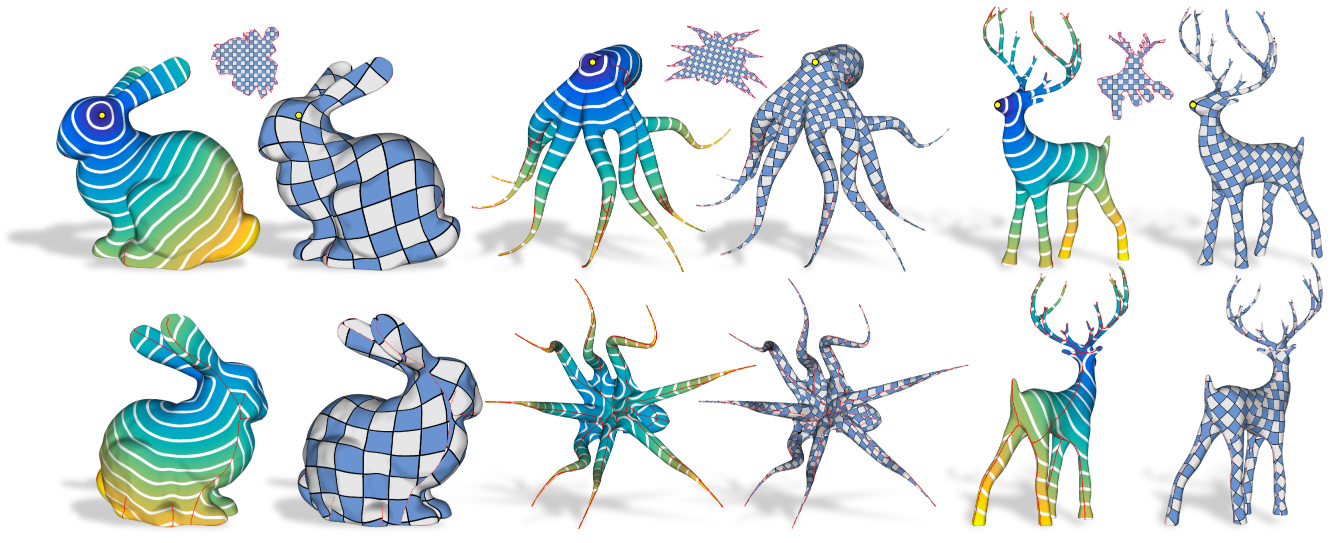

Figure 14: *Given a mesh and a point of view (yellow dot), cutting the mesh along the cut locus positions texture seams furthest from it, making discontinuities least visible. We provide three alternative examples on complex shapes containing elongated structures. UV maps (shown in the middle insets) were computed with ARAP [LZX*08].*

for computing the distance field must support meshes of this kind; and some shortest paths would contain boundary edges and not be geodesics in a geometric sense. In terms of the algorithm, the point furthest from the source is no longer guaranteed to belong to the cut locus, if it lies on the boundary. In that case, the root of the ORG spanning tree could be set at the point with the lowest Laplacian, which is the best guess in this scenario. Besides these tiny details, perhaps the most critical issue concerns the reliable estimate of the Laplacian field. As already observed for differential quantities of the first order, the absence of a complete neighborhood for boundary points makes the estimate of gradients unreliable [MLP19]. We expect the estimate of second order differential quantities to be even worse, possibly affecting the efficacy of our method. Thus, special care would be needed to process boundary points.

5.1. Applications

Being a fundamental descriptor of a distance function living on a surface manifold, the cut locus lends itself to a variety of alternative uses. While demonstrating them all is outside of the scope of this paper, in this section we showcase two practical applications involving the cut locus, in the context of global and local surface parameterization.

POV-aware texture mapping. In texture mapping, objects that are not topological disks must be cut open prior being flattened to the plane. Cuts introduce discontinuities in the map, and often accumulate distortion, resulting in visual artifacts. Techniques for texture mapping try to hide cuts in the least visible parts of the surface, so that the weak spots of the map are not immediately perceived [SH02]. To this end, the cut locus reveals itself to be a practical tool. For objects that are largely observed from a known viewpoint – e.g., for digital sculptures in a virtual museum, or for

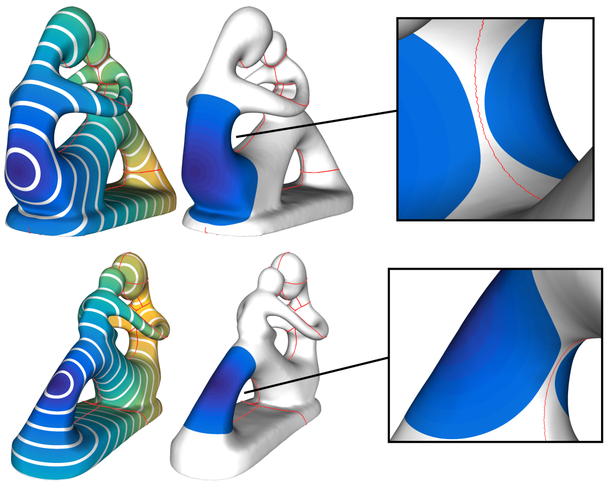

Figure 15: *Starting from a geodesic distance function emanating from a single source, our method allows to precisely retrieve its distance from the cut locus, thus defining the radius of the maximal ball under which the exponential map is injective. This construction is relevant for many techniques in machine learning and optimal transport (Sec. 1).*

objects for which a visual saliency map is known – one can initialize a distance function that emanates from a specific point of interest, and cut the mesh open through the cut locus of such function. There are two nice consequences: (i) since the cut locus and the manifold are homotopic, cutting through the cut locus will provably generate a topological disk, suitable to texture mapping; (ii) since the cut locus maximizes the distance from the source, cuts

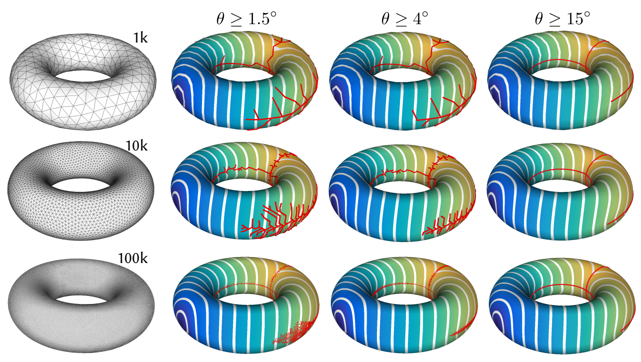

Figure 16: *Alternative cut loci computed with our method using progressively finer meshes (rows) and wider angle thresholds θ (colums). With enough filtering, our method recovers the cut locus correctly on all models. Short branches have been filtered and smoothing have been applied to models in the last column only.*

will be naturally hidden when the mesh is observed from the selected viewpoint. We implemented an interactive tool that allows the user to select a saliency point on a mesh, and automatically cuts it at the cut locus of the distance function that emanates from it. Fig. 14 shows a few results obtained with our tool.

Maximal injectivity disk. Methods that exploit a local parameterization of the surface typically rely on a heuristically computed radius, under which the exponential map is supposed to remain injective (Sec. 1). Considering its minimal computational overhead, our method allows to enhance these methods, providing a precise estimate of the maximum radius that verifies this assumption. The check is pretty straightforward: considering a point x and the distance function d_x emanating from it, the maximum radius $r_{\max}$ can be computed as

$$r_{\max} = \arg\min_p d_x(p) \quad s.t. \quad p \in \mathcal{C}(x)\,.$$

Note that a disk having radius $r_{\max}$ centered at x will be tangent to itself at the cut locus. In practice, one may want to consider a slightly smaller radius $r' = r_{\max} - \varepsilon$, which guarantees the full injectivity of the map. In Fig. 15 we show a few examples of nearly maximal disks centered at different points of a manifold with complex genus.

6. Limitations

Our method assumes a fairly accurate estimate of first and second order differential quantities of the distance function. The discrete methods discussed in Sec. 4.1 achieve this on meshes with high resolution and isotropic elements. Since our method is fast, we found it easy to remesh unsatisfactory models prior processing.

On coarse meshes, our algorithm still reconstructs the cut locus correctly, as long as the resolution is not too low and the mesh elements are slightly regular (Fig. 16). On coarse meshes, or when the elements are poor, the weak branches of the cut locus are hard to distinguish from noise. In that case, while the homology is always recovered correctly, the geometry may become imprecise, some

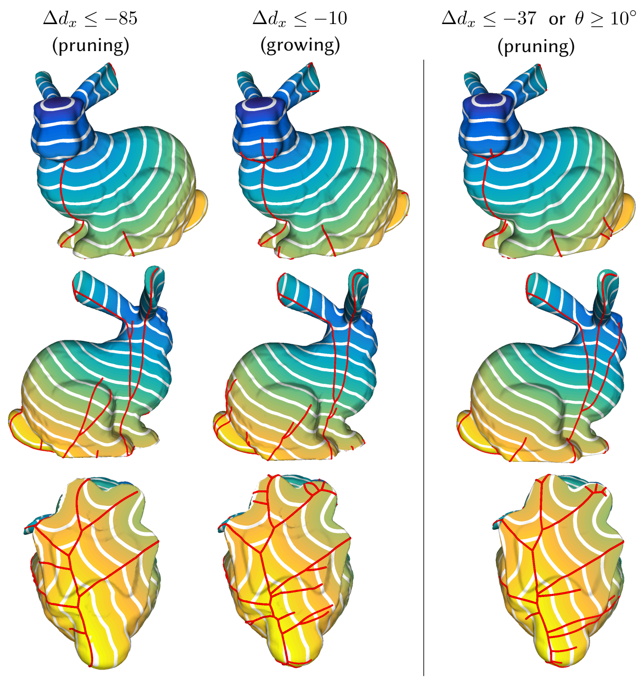

Figure 17: *Cut loci computed on the Stanford bunny with its original tessellation and filled holes (left) and a regular isotropic remeshing of it (right), computed with Meshlab [CCC*08]. Left: because of the artifacts shown in the closeups in Fig. 18, the pruning and the growing policies are not able to retain the cut locus and, at the same time, filter the spurious branches of the spanning tree. Right: on a regular mesh all versions of the filter perform equally well.*

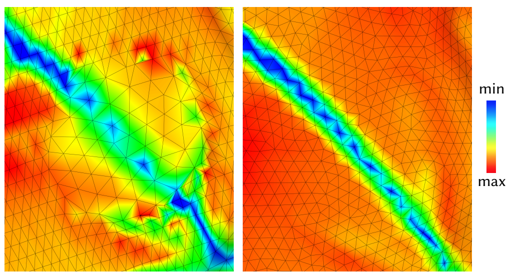

Figure 18: *Estimating the Laplacian on an irregular (left) and regular (right) meshing: a poor tessellation may break the valleys of the Laplacian (discontinuous blue line) and introduce spurious local minima. Both types of artifacts may hinder the action of our filters, as exemplified in Fig. 17.*

weak branches may be lost, and some spurious branches may be retained. For instance, the Stanford bunny has a complicated cut locus with many free branches that are hard to detect (Fig. 17). On the original mesh, the irregular tessellation may interrupt the valleys of the Laplacian that demarcate the cut locus, or introduce bumps along them, or spurious local minima (Fig. 18). Such a poor estimation may cause some points to become either too strong, or too weak, thus hindering the action of filters. Note that the "strength" of a branch depends on the location of the source, the geometry of the shape, and the discretization. Therefore, in general it is not possible to determine a priori which mesh density is suitable to detect all branches. The pruning policy needs an aggressive threshold to filter the branches ending at spurious local minima, also missing the weak branches of the cut locus (Fig. 17, left). On the other hand, with the growing policy some branches of the cut locus are truncated too soon, because of bumps and cracks along the valleys of the Laplacian (Fig. 17, middle). On a regular tessellation, both policies allow us computing an accurate estimate of the cut locus with reasonable thresholds (Fig. 17, right).

We did not experiment with extreme cases characterized by very sparse meshing and long and skinny elements, such as many of the meshes found in the Thingi10k repository [ZJ16]. In that case, it may be convenient to use intrinsic triangulations [SSC19a] and related differential estimators [SC20]. The rest of our method would work unchanged, but it needs being implemented in the framework of intrinsic triangulations.

7. Concluding remarks

We have presented a novel method to compute the cut locus that is practical and fast. The method depends on a unique intuitive parameter that can be tuned interactively to filter out artifacts arising from small details of the surface, or from discretization. The method works on surfaces of any genus, always recovering the correct topology of the cut locus; it works on shapes with sharp creases; and on rough shapes with many small details, too.

We conjecture that out method converges to the true cut locus as the mesh becomes denser, but proving this fact requires further work. In summary, all methods we adopted for computing the distance field can be shown to converge to the true geodesic distance; and the method to compute the differential quantities also converges for smooth functions. By applying such estimators to a denser and denser mesh, the estimated Laplacian should converge to the Laplacian of the distance function away from the cut locus, and to the Laplacian of a smooth barrier function near the cut locus. Thus, we expect that for any given value $A < 0$ there exist a mesh dense enough that $L(y) < A$ for all y at the cut locus. However, since no bound from below to the Laplacian away of the cut locus is known, in general the Laplacian alone is not sufficient to characterize all and only the points of the cut locus. This fact further motivates the additional criteria that we apply in our method.

We foresee two interesting avenues for future works. For the computation of the cut locus, we plan to improve on our current method to achieve a reliable fully automatic detection that works well in all practical scenarios. The most challenging issue in this direction is to determine where the weak free branches end. It would also be interesting to exploit the boundary structure provided by MMP-like algorithms, as in [LCT11]. Based on such structure, the exact cut locus in the polyhedral metric can be computed. However, such a cut locus would consist of a dense tree, with one leaf at each parabolic vertex, thus being useful only for strictly polyhedral objects without any curved surface. It is an open problem how to define suitable pruning strategies to obtain a cut locus that is descriptive for curved objects approximated with a mesh, too.

Finally, we plan to explore the capabilities of our approach for the computation of the medial axis, which is a widely popular shape descriptor used for shape compression, matching and skeletonization [TDS*16]. Indeed, the medial axis can be defined as the cut locus of a distance field emanating from the boundaries of a geometric domain, growing inwards. Despite in this work we focused our attention on distance fields emanating from a single (point-like) source, in his work Générau showed that the Laplacian goes to $-\infty$ also for distance fields emanating from a general hypersurface embedded in the manifold domain [Gén20], creating a connection with the λ-medial axis [CL05]. In its current state, our algorithm is not able to reconstruct a proper connectivity for this more general case, but since the theoretical foundation still holds, it would be interesting to work at different tools to filter the Laplacian field and generate the medial connectivity, both for 2D and 3D manifolds.

Acknowledgments

The authors wish to thank Carlo Mantegazza for useful discussions, and the anonymous reviewers for their comments and suggestions. The work of Marco Livesu was partly supported by EU ERC Advanced Grant CHANGE No. 694515.

References

[AOCBC15] AZENCOT O., OVSJANIKOV M., CHAZAL F., BEN-CHEN M.: Discrete derivatives of vector fields on surfaces – an operator approach. *ACM Trans. Graph. 34*, 3 (May 2015). 1

[Buc77] BUCHNER M. A.: Simplicial structure of the real analytic cut locus. *Proc. Amer. Math. Soc. 64*, 1 (1977), 118–121. 2, 3

[BvdPPH11] BONNEEL N., VAN DE PANNE M., PARIS S., HEIDRICH W.: Displacement interpolation using lagrangian mass transport. *ACM Trans. Graph. 30*, 6 (2011), 1–12. 1

[BYF*19] BUDNINSKIY M., YIN G., FENG L., TONG Y., DESBRUN M.: Parallel transport unfolding: A connection-based manifold learning approach. *SIAM Jou. Appl. Algebra and Geo. 3*, 2 (2019), 266–291. 1

[CCC*08] CIGNONI P., CALLIERI M., CORSINI M., DELLEPIANE M., GANOVELLI F., RANZUGLIA G.: MeshLab: an Open-Source Mesh Processing Tool. In *EG Ital. Chap. Conf.* (2008), pp. 129–136. 7, 11

[CL05] CHAZAL F., LIEUTIER A.: The "λ-medial axis". *Graphical Models 67*, 4 (2005), 304–331. 12

[CWW13] CRANE K., WEISCHEDEL C., WARDETZKY M.: Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Trans. Graph. 32*, 5 (2013). 8

[DL09] DEY T. K., LI K.: Cut locus and topology from surface point data. *Proce. Symp. on Comp. Geo.* (2009), 125–134. 2, 3, 7

[Epp03] EPPSTEIN D.: Dynamic generators of topologically embedded graphs. In *Proc. ACM-SIAM Symp. Disc. Alg.* (2003), pp. 599–608. 5

[EW05] ERICKSON J., WHITTLESEY K.: Greedy optimal homotopy and homology generators. In *Proc. 16th ACM-SIAM Symp. Disc. Alg.* (2005), vol. 5, pp. 1038–1046. 5

[Gén20] GÉNÉRAU F.: *On a stable variational approximation of the cut locus, and a non-local isoperimetric problem.* PhD thesis, Université Grenoble Alpes, June 2020. 2, 3, 7, 12

[GHL04] GALLOT S., HULIN D., LAFONTAINE J.: *Riemannian geometry.* Springer, Berlin New York, 2004. 3

[GMST05] GRAVENSEN J., MARKVORSEN S., SINCLAIR R., TANAKA M.: The Cut Locus of a Torus of Revolution. *Asian Journal of Mathematics 9*, 1 (2005), 103 – 120. 1, 2, 3, 7

[GOV20] Générau F., Oudet E., Velichkov B.: Cut locus on compact manifolds and uniform semiconcavity estimates for a variational inequality, 2020. arXiv:2006.07222. 2

[HA19] Herholz P., Alexa M.: Efficient Computation of Smoothed Exponential Maps. *Computer Graphics Forum* (Jan. 2019). 1

[IK04] Itoh J.-i., Kiyohara K.: The cut loci and the conjugate loci on ellipsoids. *Manuscripta Mathematica 114* (2004), 247–264. 2

[IS04] Itoh J.-i., Sinclair R.: Thaw: A Tool for Approximating Cut Loci on a Triangulation of a Surface. *Experimental Mathematics 13*, 3 (2004), 309 – 325. 2

[JP*18] Jacobson A., Panozzo D., et al.: libigl: A simple C++ geometry processing library, 2018. https://libigl.github.io/. 7, 8

[KCPS13] Knöppel F., Crane K., Pinkall U., Schröder P.: Globally optimal direction fields. *ACM Trans. Graph. 32*, 4 (2013). 5

[KCPS15] Knöppel F., Crane K., Pinkall U., Schröder P.: Stripe patterns on surfaces. *ACM Trans. Graph. 34*, 4 (2015), 1–11. 1

[LBS05] Langer T., Belyaev A., Seidel H.-P.: Asymptotic analysis of discrete normals and curvatures of polylines. In *Proc. 21st Spring Conf. on Comp. Graph.* (2005), pp. 229–232. 7

[LCT11] Liu Y. J., Chen Z., Tang K.: Construction of iso-contours, bisectors, and Voronoi diagrams on triangulated surfaces. *IEEE Trans. Pattern Analysis Machine Intelligence 33*, 8 (2011), 1502–1517. 12

[LD11] Lipman Y., Daubechies I.: Conformal wasserstein distances: Comparing surfaces in polynomial time. *Advances in Mathematics 227*, 3 (2011), 1047–1077. 1

[LGS12] Livesu M., Guggeri F., Scateni R.: Reconstructing the curve-skeletons of 3d shapes using the visual hull. *IEEE Trans. Vis. Comp. Graph. 18*, 11 (2012), 1891–1901. 4

[Liv19] Livesu M.: cinolib: a generic programming header only c++ library for processing polygonal and polyhedral meshes. *Trans. Comp. Sci. XXXIV* (2019). https://github.com/mlivesu/cinolib/. 7

[LZX*08] Liu L., Zhang L., Xu Y., Gotsman C., Gortler S. J.: A local/global approach to mesh parameterization. In *Computer Graphics Forum* (2008), vol. 27, Wiley Online Library, pp. 1495–1504. 10

[MBAM11] Misztal M. K., Bærentzen J. A., Anton F., Markvorsen S.: Cut locus construction using deformable simplicial complexes. *Proc. Int. Symp. Vor. Diag. Sci. Eng. 2* (2011), 134–141. 2

[MBBV15] Masci J., Boscaini D., Bronstein M. M., Vandergheynst P.: Geodesic convolutional neural networks on riemannian manifolds. In *IEEE ICCV Workshop* (2015), pp. 832–840. 1

[MKK21] Mitchel T. W., Kim V. G., Kazhdan M.: Field Convolutions for Surface CNNs, 2021. arXiv:2104.03916. 1

[MLP19] Mancinelli C., Livesu M., Puppo E.: A comparison of methods for gradient field estimation on simplicial meshes. *Computers & Graphics 80* (2019), 37 – 50. 4, 10

[MM02] Mantegazza C., Mennucci A. C.: Hamilton—Jacobi Equations and Distance Functions on Riemannian Manifolds. *Applied Mathematics & Optimization 47*, 1 (2002), 1–25. 3

[MMP87] Mitchell J. S. B., Mount D. M., Papadimitriou C. H.: The discrete geodesic problem. *SIAM J. Comput. 16*, 4 (Aug. 1987), 647–668. 8

[MMU14] Mantegazza C., Mascellani G., Uraltsev G.: On the Distributional Hessian of the Distance Function. *Pacific J. of Math. 270* (2014), 151–166. 2

[MRCK21] Mitchel T. W., Rusinkiewicz S., Chirikjian G. S., Kazhdan M.: Echo: Extended convolution histogram of orientations for local surface description. *Comp. Graph. Forum 40*, 1 (2021), 180–194. 1

[Mye35] Myers S. B.: Connections between differential geometry and topology I,II. *Duke Mathematical Journal 1*, 3 (1935), 276 – 391. 2

[Nee07] Neel R.: The small-time asymptotics of the heat kernel at the cut locus. *Comm. in Analysis and Geometry 15*, 4 (2007), 845–890. 2

[NPP20] Nazzaro G., Puppo E., Pellacini F.: Decosurf: Recursive geodesic patterns on triangle meshes, 2020. arXiv:2007.10918. 8

[PNC19] Pellacini F., Nazzaro G., Carra E.: Yocto/GL: A Data-Oriented Library For Physically-Based Graphics. In *EG Ital. Chap. Conf.* (2019). https://github.com/xelatihy/yocto-gl. 7, 8

[Poi05] Poincare H.: Sur les lignes geodesiques des surfaces convexes. *Trans. American Math. Soc. 6*, 3 (1905), 237–274. 2

[PP93] Pinkall U., Polthier K.: Computing discrete minimal surfaces and their conjugates. *Experimental Math. 2*, 1 (1993), 15–36. 4

[QHY*16] Qin Y., Han X., Yu H., Yu Y., Zhang J.: Fast and exact discrete geodesic computation based on triangle-oriented wavefront propagation. *ACM Trans. Graph. 35*, 4 (2016), 125:1–125:13. 8

[RGA*20] Rakotosaona M.-J., Guerrero P., Aigerman N., Mitra N., Ovsjanikov M.: Learning delaunay surface elements for mesh reconstruction, 2020. arXiv:2012.01203. 1

[RKS00] Rössl C., Kobbelt L., Seidel H., 2000: Extraction of feature lines on triangulated surfaces using morphological operators. In *AAAI Symp. on Smart Graphics* (2000). 4

[Sak97] Sakai T.: *Riemannian Geometry*, vol. 149. American Mathematical Society, 1997. 1, 2, 3

[SC20] Sharp N., Crane K.: A Laplacian for Nonmanifold Triangle Meshes. *Computer Graphics Forum (SGP) 39*, 5 (2020). 12

[Sch13] Schmidt R.: Stroke Parameterization. *Comp. Graph. Forum 32*, 2pt2 (May 2013), 255–263. 1

[SDGP*15] Solomon J., De Goes F., Peyré G., Cuturi M., Butscher A., Nguyen A., Du T., Guibas L.: Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Trans. Graph. 34*, 4 (2015), 1–11. 1

[SH02] Sheffer A., Hart J. C.: Seamster: inconspicuous low-distortion texture seam layout. In *IEEE Vis.* (2002), pp. 291–298. 10

[Sol18] Solomon J.: Optimal transport on discrete domains. *AMS Short Course on Discrete Differential Geometry* (2018). 1

[SRC*20] Sun Z., Rooke E., Charton J., He Y., Lu J., Baek S.: Zernet: Convolutional neural networks on arbitrary surfaces via zernike local tangent space estimation. *Comp. Graph. Forum 39*, 6 (2020), 204–216. 1

[SRGB14] Solomon J., Rustamov R., Guibas L., Butscher A.: Earth mover's distances on discrete surfaces. *ACM Trans. Graph. 33*, 4 (2014), 1–12. 1

[SSC19a] Sharp N., Soliman Y., Crane K.: Navigating intrinsic triangulations. *ACM Transactions on Graphics 38*, 4 (2019). 12

[SSC19b] Sharp N., Soliman Y., Crane K.: The vector heat method. *ACM Trans. Graph. 38*, 3 (June 2019), 1–19. 1

[ST02] Sinclair R., Tanaka M.: Loki: Software for Computing Cut Loci. *Experimental Math. 11*, 1 (2002), 1 – 25. 2

[TDS*16] Tagliasacchi A., Delame T., Spagnuolo M., Amenta N., Telea A.: 3d skeletons: A state-of-the-art report. In *Comp. Graph. Forum* (2016), vol. 35, pp. 573–597. 12

[Vil08] Villani C.: *Optimal Transport Old and New*. Springer, 2008. 1

[Vil11] Villani C.: Regularity of optimal transport and cut locus: From nonsmooth analysis to geometry to smooth analysis. *Discrete and Continuous Dynamical Systems 30*, 2 (2011), 559–571. 1

[WYLC13] Wang R., Yang Z., Liu L., Chen Q.: Discretizing Laplace-Beltrami Operator from Differential Quantities. *Comm. Math. Stati. 1*, 3 (2013), 331–350. 4

[YCS00] Yim P. J., Choyke P. L., Summers R. M.: Gray-scale skeletonization of small vessels in magnetic resonance angiography. *IEEE Trans. Medical Imaging 19*, 6 (2000), 568–576. 4

[ZJ16] Zhou Q., Jacobson A.: Thingi10k: A dataset of 10,000 3d-printing models. arXiv:1605.04797. 12