

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/327023647>

AN EFFICIENT DATA RETRIEVAL PARALLEL REEB GRAPH ALGORITHM

Article · August 2018

CITATIONS
0

READS
8

2 authors:



[Mustafa Hajij](#)
The Ohio State University
29 PUBLICATIONS **39** CITATIONS

SEE PROFILE



[Paul Rosen](#)
University of South Florida
70 PUBLICATIONS **393** CITATIONS

SEE PROFILE

AN EFFICIENT DATA RETRIEVAL PARALLEL REEB GRAPH ALGORITHM

MUSTAFA HAJIJ AND PAUL ROSEN

ABSTRACT. The Reeb graph of a scalar function defined on a domain gives a topological meaningful summary of that domain. Reeb graphs have been shown in the past decade to be of great importance in geometric processing, image processing, computer graphics and computational topology. The demand to compute large data sets has increased in the last decade and hence the consideration of parallelization of topological computations. We propose a parallel Reeb graph algorithm on triangulated meshes with and without a boundary. Furthermore, we give a description for extracting the original manifold data from the Reeb graph structure. As an application, we show how our algorithm can be utilized in mesh segmentation algorithms.

CONTENTS

1. Introduction	1
1.1. Prior work and contribution	2
2. Morse theory and Reeb graphs	3
2.1. Morse theory	3
2.2. Reeb graphs	4
3. Reeb graphs on triangulated surfaces	5
3.1. Morse function in the piece-wise linear setting	5
3.2. Reeb graphs of general simplicial complexes	6
4. Reeb graph sequential algorithm on triangulated surfaces	6
4.1. Critical sets and ascending paths	6
4.2. Outline of the sequential algorithm	8
4.3. Surfaces with boundaries	10
4.4. Correctness of the sequential algorithm	11
4.5. Dealing with the degenerate cases	12
5. Parallelization of the algorithm	13
5.1. The partition stage	14
5.2. Computing the Reeb graph of each submanifold	14
5.3. The gluing stage	15
6. Going from the Reeb graph to the manifold	15
7. Run-time analysis and implementation	16
8. Applications	17
9. Conclusion and Future Work	19
ACKNOWLEDGEMENTS	19
References	20

1. INTRODUCTION

Recent years have witnessed an extensive research in topological based method to analyze and study data [10, 11]. The popularity of topological based techniques seem to be coming from the generality and the robustness of the these techniques and their applicability to a wide range of areas.

The *Reeb graph* [62] has been one of the most successful topological tools in the data analysis and data understanding. The Reeb graph is a data structure associated to a scalar function defined on a manifold. It gives an efficient topological summary for the manifold by encoding the evolution of the connectivity of its level sets. Reeb graphs, and their loop-less version, contour trees [9], are of fundamental importance in computational topology, geometric processing, image processing, computer graphics, and more recently, data analysis and visualization. Examples of Reeb graph applications include quadrangulation [37], shape understanding [1], surface understanding and data simplification [6], parametrization [60, 85], segmentation [79], animation [40], feature detection [70], data reduction and simplification [13, 65], image processing [41], visualization of isosurfaces [2] and many others.

The past decade has witnessed an increase of large geometric data on which a scalar field is defined. This has yielded several challenges for the time efficiency computation of topological structures usually computed on such data. Parallelization of the utilized algorithms is a natural direction one usually takes in order to improve the computational-time efficiency. In this article we introduce an efficient shared memory parallel algorithm to compute the Reeb graph a scalar function defined on a triangulated surface with or without a boundary. In addition, our algorithm provides a fast method for retrieving the surface data from the constructed Reeb graph. Existing Reeb graph algorithms lack explicit description of manifold information from the Reeb graph. For this purpose we define an explicit map that associates the Reeb graph data to its corresponding data on the manifold. As an application we show how Reeb graph can be used to identify and calculate curves with certain homological properties on a surface. Finally, we show how the data retrieval aspect of our algorithm along with the curves that can be extracted from the Reeb graph structure can be used for mesh segmentation and mesh parametrization.

1.1. Prior work and contribution. Reeb graph literature has is vast and ranges from computational accuracy of the graph to its applications in data analysis and visualization. The first provably correct algorithm to compute a Reeb graph on a triangulated surface was presented by Shinagawa and Kunii in [66]. Computing Reeb graph in $O(n^2)$ times where n in the number of triangles in the mesh. This time was improved to $O(n \log(n))$ by Cole-McLaughlin et al. [18].

Reeb graphs have also been studied for higher dimensional manifolds and simplicial complexes. An algorithm for computing Reeb graph for a 3-manifold embedded in \mathbb{R}^3 is proposed in [72]. The first Reeb graph algorithm on an arbitrary simplicial complex is given in [17]. Pascucci et al. [18], presented an online method that computes the graph by calculating the Reeb graph of the restriction of the input scalar field on the 2-skeleton of the input mesh. This algorithm can handle a non-manifold input, but its worse case time complexity is quadratic. Reeb graph for a varying scalar function is studied in [28]. Other Reeb graphs algorithms can be found in [25, 26, 35, 57, 59]. Approximating Reeb graphs algorithms can be found in [6, 38]. However, such algorithms may lead to inaccurate results. A loop-free Reeb graph, also called a contour tree, has been used extensively in data analysis and data visualization. Algorithms for computing such graphs can be found in [17, 27, 61, 71, 76]. Contour tree have been used for scientific visualization [58], volume rendering [78], terrain applications [5, 31]. For a thorough introduction to contour tree and its applications the reader is referred to [12, 61] and the references within.

Reeb graphs have been also used to study and analyze points cloud data. The applications are numerous including data skeletonization [29], retrieving topological information from point data such as homology group computation [16, 23], locus cut [20], data abstraction [54], and recovering structural information of a scalar function on a point data [15]. In the context of point cloud, a relatively recent constructed named Recently, *Mapper* [67] has received a lot of attention, as it generalizes both the Reeb graph and contour tree. Mapper has found numerous applications [46, 56, 63, 64] and has been studied from multiple perspectives [14, 22, 34, 53].

There is rich literature in computer graphics regarding the use of the Reeb graph. Reeb Graph has been used in mesh segmentation [81], shape similarity [75], shape matching [49], feature-extraction [2], surface reconstruction [8], extracting tunnel and handle loops of a surface [21], removing tiny handle in an iso-surface [80] and shape matching [38]. See also [7] for further applications of Reeb graph in computer graphics. Our algorithm relies on Morse theory to find a finite set of paths traced concurrently inside parts of the manifold where the topology of the manifold with respect to a given scalar function does not change. The sequential algorithm that we present here is essentially similar to Reeb graph algorithm given in [25], where tracing paths inside cylinders were used to construct the Reeb graph. We provide the sequential version of the parallel algorithm in this paper because it has some major differences from the algorithm given [25]. However our main contributions and focus, which are listed below, are not the sequential Reeb graph algorithm.

The demand to compute large data sets has increased in the last decade and hence the consideration of topological computations parallelization. Multiple attempts have been made in this direction, including multicore homology computation [43] spectral sequence parallelization [45], distributed contour tree [51], distributed merge tree [50], and distributed Mapper [32]. In this paper, we give a parallel Reeb graph algorithm on arbitrary triangulated mesh with and without a boundary. We prove the correctness of our method using fundamental theorems in Morse Theory. Moreover, we discuss the performance results that compare our approach to a reference sequential Reeb graph algorithm. We then show how we can use the Reeb graph to retrieve certain curves on the manifold. Finally, we utilize the data retrieval aspect of our algorithm and give an application to surface segmentation.

This article has the following contributions:

- (1) We give an efficient parallel algorithm that computes the Reeb graph of piece-wise linear function defined on a triangulated 2-manifold with and without a boundary.
- (2) Our method can be used to retrieve the manifold data from the Reeb graph. In other words, given a point in the Reeb graph, we give an efficient method to retrieve the manifold data that corresponds to that point.
- (3) We show how the homological properties of a Reeb graph can be used to extract certain curves on a surface and we utilize our algorithms to give a mesh segmentation algorithm.
- (4) The algorithms presented here are easy to implement and require minimal memory storage.

2. MORSE THEORY AND REEB GRAPHS

In this section, we review the basic background needed in this paper. We start by reviewing the basics of Morse theory and Reeb graphs on a smooth manifolds. Then we discuss the corresponding piece-wise linear version. For more details on Morse theory the reader is referred to [4, 47].

2.1. Morse theory. Morse Theory is a tool in differential topology that is concerned with the relations between the geometric and topological aspects of manifolds and the real-valued functions defined on them. One of the primary interest in this theory is the relationship between the topology of a smooth manifold M and the critical points of a real-valued smooth function f defined on M . Intuitively, Morse theory studies the topological changes of the level sets of a real-valued smooth function as the height of f varies. Morse theory was first introduced by Morse [52] for infinite dimensional spaces. A comprehensive introduction to Morse theory on finite-dimensional manifolds is given [48]. Also see [4, 47]. Morse theory has been proven to be a very useful tool in computer graphics, geometric data processing and understanding. The theory was extended to triangulated 2-manifolds by [3]. Recently, Morse theory has found applications in global surface parameterization [30], finding a fundamental domain of a surface [55], surface quadrangulation [24], topological matching [38], implicit surfaces [69], surface segmentation [83], spline construction [77] and many other applications.

Let M be a compact and smooth n -manifold and let $I = [a, b] \subseteq \mathbb{R}$, where $a < b$, be a closed interval. Let $f : M \rightarrow I$ be a smooth function defined on M . A point $x \in M$ is called a *critical point* of f if the differential df_x is zero. A value c in \mathbb{R} is called a *critical value* of f if $f^{-1}(c)$ contains a critical point of f . A point in M is called a *regular point* if it is not a critical point. Similarly, if a value $c \in \mathbb{R}$ is not a critical value then we call it a regular value. The inverse function theorem implies that for every regular value c in I the level set $f^{-1}(c)$ is a disjoint union of $n - 1$ manifolds. In particular, when $n = 2$, then $f^{-1}(c)$ is a disjoint union of simple closed curves. A critical point is called *non-degenerate* if the matrix of the second partial derivatives of f , called the *Hessian matrix*, is non-singular. A differentiable function $f : M \rightarrow I$ is called *Morse* if all its critical points are non-degenerate and all critical values are distinct. If the manifold M has boundary, i.e. $\partial M \neq \emptyset$, then we will also require two other conditions: (1) $f^{-1}(\partial I) = \partial M$ and (2) there are no critical points on ∂M . In other words, the boundary points in the interval I , the values in ∂I , are regular values for the function f . The *index* of a critical point x of f , denoted by $index_f(x)$, is defined to be the number of negative eigenvalues of its Hessian matrix. For instance the Hessian of a scalar function on smooth surface is a 2×2 symmetric matrix. Hence, the index of f on a critical point takes the values 0, 1 or 2. In this case an index 0, 1 and 2 of a critical point of a function f is nothing more than a local minimum, a saddle and a local maximum for f , respectively.

If f is a Morse function on a surface then, up to a change of coordinates, the surface around a critical point f takes one of the three shapes illustrated in Figure 1. More formally we have the following important Lemma:

Lemma 2.1. (Morse Lemma) Let M be a smooth surface, $f : M \rightarrow \mathbb{R}$ be a smooth function and p be a non-degenerate critical point of f . We can choose a chart (ϕ, U) around p such that $f \circ \phi^{-1}$ takes exactly one of the following three forms:

- (1) Local minimum — $f \circ \phi^{-1}(X, Y) = X^2 + Y^2 + c$.
- (2) Saddle — $f \circ \phi^{-1}(X, Y) = X^2 - Y^2 + c$.
- (3) Local maximum — $f \circ \phi^{-1}(X, Y) = -X^2 - Y^2 + c$.

An analogous lemma holds for Morse functions on higher dimensional smooth manifolds. See [47] for more details. Morse Lemma implies that a Morse function around a critical point looks simple, and it is exactly one of the forms given in the Lemma, above up to change of coordinates. Notice that the number of minus signs in the standard form of the function f around p is equal to the index of the critical point p .

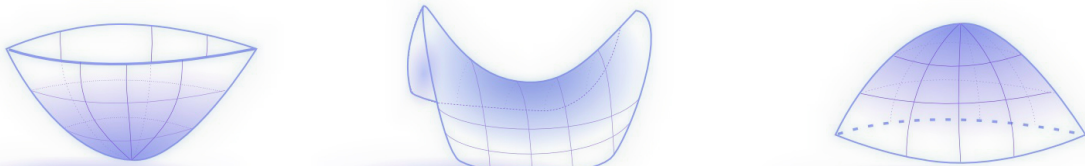


FIGURE 1. Minimum, Saddle and Maximum, respectively.

2.2. Reeb graphs. Let M be a topological space and let $f : M \rightarrow \mathbb{R}$ be a scalar function defined on M . The Reeb graph of the pair f and M gives a summary of the topological information

encoded in by tracking changes occurring to the connected components of the level sets of f . More precisely, the *Reeb graph* of M and f is the quotient space $R(M, f)$ of M defined as follows. We say the x and y are equivalent in M , and write $x \sim y$, if and only if they belong to the same connected component of $f^{-1}(r)$ for some $r \in \mathbb{R}$. The quotient space $M/\sim = R(M, f)$ with the quotient space topology induced by the quotient map $\pi : M \rightarrow R(M, f)$ is called the Reeb graph of M and f . Recall here that the map π takes a point x in M to its equivalence class $[x]$ in $R(M, f)$. Given a point p in $R(M, f)$, it is often important in practice to retrieve the the set of points in M that map to p via π . In this article, we provide an efficient method to retrieve the data in X associated to a point on a Reeb graph.

The map π induces a continuous function $\bar{f} : R(M, f) \rightarrow \mathbb{R}$, where $\bar{f}(p) = f(x)$ if $p = \pi(x)$. This map is well defined since $f(x) = f(y)$ whenever $\pi(x) = \pi(y)$.

When M is a manifold, and f is a Morse, then $R(M, f)$ exhibits certain additional properties. For instance, in this case every vertex of $R(X, f)$ arises from a critical point of f or a boundary component. Furthermore, every maximum or minimum of f gives rise to a degree 1-node of $R(M, f)$. Saddle points for a Morse function f defined on a 2-manifold have degree 3-node. This is not guaranteed if the scalar function is not Morse. See Figure 2 for an example of a Reeb graph.

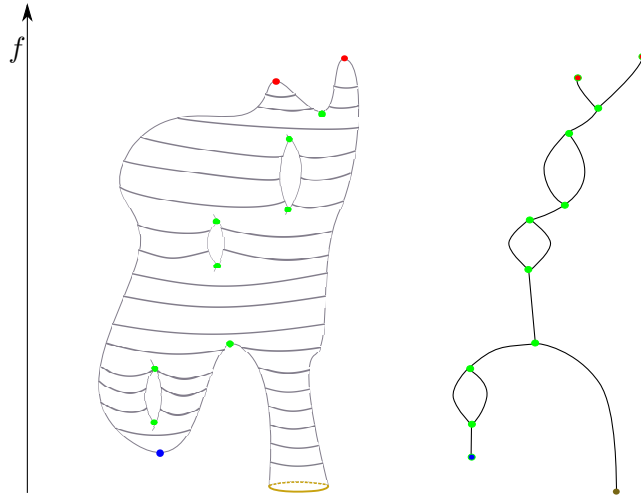


FIGURE 2. An example of a Reeb graph (right) of a scalar function (left) defined on a surface with a boundary.

3. REEB GRAPHS ON TRIANGULATED SURFACES

3.1. Morse function in the piece-wise linear setting. In this paper we will be working with triangulated surfaces. We will denote to the vertices, edges and faces of a triangulated mesh M by $V(M)$, $E(M)$ and $F(M)$ respectively. The extension of Morse theory to triangulated manifolds was by given Banchoff [3]. Let M be a triangulated 2-manifold and let $f : M \rightarrow [a, b] \subset \mathbb{R}$ be piece-wise linear function on M . The star of v , denoted by $star(v)$, is the set of simplices that intersect with the vertex v . The closure $\overline{star(v)}$ of $star(v)$ is the smallest simplicial complex of M that contains $star(v)$. The link $Lk(v)$ consists of the set of vertices of M that belong $\overline{star(v)}$ but do not belong to $star(v)$. The upper link of v is defined to be the set:

$$Lk^+(v) = \{u \in Lk(v) : f(u) > f(v)\} \cup \{[u, v] \in Lk(v) : f(u) > f(v)\},$$

the lower link is defined similarly by:

$$Lk^-(v) = \{u \in Lk(v) : f(u) < f(v)\} \cup \{[u, v] \in Lk(v) : f(u) < f(v)\},$$

and *mixed link*

$$Lk^\pm(v) = \{[u_1, u_2] : f(u_1) < f(v) < f(u_2)\}.$$

Using the link definitions, we classify vertices of M as follows. A vertex v in M is *PL regular* if the cardinality $|Lk^\pm(v)|$ of $Lk^\pm(v)$ is equal to 2. If $|Lk^+(v)| = 0$, then v is a *PL maximum* vertex with index 1, and if $|Lk^-(v)| = 0$, then v is a *PL minimum* with index 0. If $|Lk^\pm(v)| = 2 + 2m$, then v is a PL saddle with index 1 and multiplicity $m \geq 1$. See Figure 3. A PL function $f : V \rightarrow [a, b] \subset \mathbb{R}$ is *PL Morse function* if each vertex is either PL regular or PL simple, and the function values of the vertices are distinct. Similar to the smooth case, when $\partial M \neq \emptyset$, we assume further that f satisfies: (1) $f^{-1}(\partial[a, b]) = \partial M$ and (2) there are no critical points on ∂M .

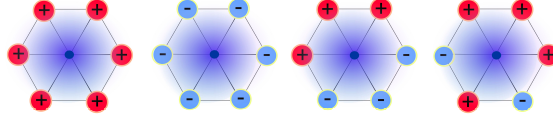


FIGURE 3. The types of vertices on a triangulated mesh. From left to right: minimum, maximum, regular vertex, saddle.

Remark 3.1. There are other ways to define a Morse function of a manifold with a boundary. See for instance [19]. Our choice here will be utilized the parallelization of the Reeb graph algorithm and the application section 8.

3.2. Reeb graphs of general simplicial complexes. Reeb graph can be defined naturally on arbitrary simplicial complexes. Let K be a simplicial complex and $f : K \rightarrow \mathbb{R}$ be a simplicial map defined on the vertices of K . The map f can be extended linearly to all simplices of K to PL function which we will denote also denote by f . Using this function, the Reeb graph of (K, f) can be defined as before.

4. REEB GRAPH SEQUENTIAL ALGORITHM ON TRIANGULATED SURFACES

In this section we assume that we are given a PL function $f : M \rightarrow \mathbb{R}$ defined on a triangulated surface M possibly with a non-empty boundary ∂M . This includes the case when f is a Morse function. Then we discuss the degenerate case, when f has non-simple saddles.

As mentioned, the sequential algorithm that we present here is similar to Reeb graph algorithm given in [25]. We provide the sequential version of the algorithm because it has some keys differences from the algorithm given [25] that will be utilized later in our parallel algorithm.

The main idea of the algorithm is the construction of a sub-simplicial complex X of M such that the Reeb graph $R(X, f|_X)$ of X with respect to $f|_X$, the restriction of f on X , is identical to the Reeb graph $R(M, f)$. The constructed simplicial complex X does not only provide us with a Reeb graph of (M, f) but also implies immediately an algorithm to compute the map $F : R(M, f) \rightarrow M$ that allows us to extract the manifold data given the corresponding Reeb graph points. The main two ingredients of the algorithm are the *critical sets* and the *ascending paths*. We introduce these two concepts next.

4.1. Critical sets and ascending paths. We start by giving the definition of critical sets then we give the definition of ascending paths.

Critical sets. Let p be a saddle point of f , and let t_p be its corresponding critical value. Consider the connected components of the set $f^{-1}(t_p)$. The connected components of $f^{-1}(t_p)$ consist of a collection of simple closed curves embedded in M , as well as a single component, which contains a singularity. This *singular set* consists of multiple circles that intersect at the critical point p . We will denote this singular set by C_p . See Figure 4 for an example. Note that for critical value t $f^{-1}(t_p)$ might consist merely of the critical set C_p (with no other simple closed curves).

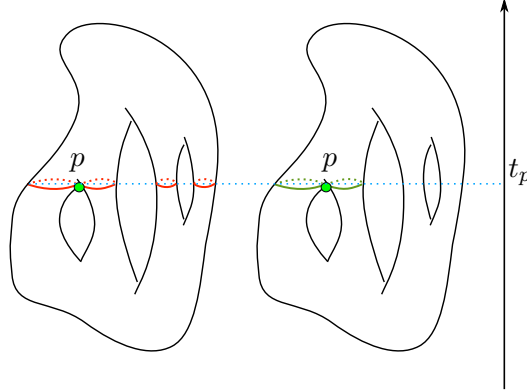


FIGURE 4. For t_p a critical value, the left figure shows the set $f^{-1}(t_p)$, which consists of a singular set C_p and two of simple closed curves. On right only the critical set C_p is shown, which is the connected component of $f^{-1}(t_p)$ that contains the critical point p .

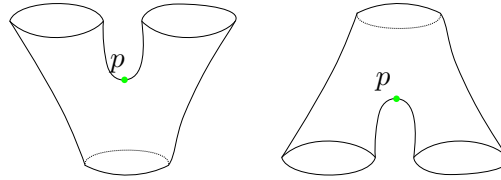


FIGURE 5. A split saddle on the left and a merge saddle on the right.

Choose $\epsilon > 0$ small enough such that the interval $[t_p - \epsilon, t_p + \epsilon]$ has only the critical value t_p . As we move from t_p to $t_p - \epsilon$, the singular set C_p becomes a non-singular one consisting of a disjoint union of simple closed curves A_1, \dots, A_n , for $n \geq 1$. By convention we will consider the sets A_1, \dots, A_n to be the *connected components of the singular set C_p* , and we will refer to them as such for the rest of the paper. We talk more about about the components of a critical set and show exactly how to determine them in the piece-wise linear setting in Section 4.2.1. The following Lemma asserts that the number of connected components of C_p of for a simple saddle point of a Morse function defined on surface is either 1 or 2.

Lemma 4.1. Let M be a compact connected orientable surface with a Morse function $M \rightarrow \mathbb{R}$. If f has a unique saddle point then M is homeomorphic to a pair of pants. See Figure 5.

For a regular value t , we will denote to the number of simple closed curves of $f^{-1}(t)$ by $|f^{-1}(t)|$. Lemma 4.1 implies that for a sufficiently small enough ϵ and for any saddle point p on a Morse function f , one has $|f^{-1}(t_p + \epsilon)| - |f^{-1}(t_p - \epsilon)| = \pm 1$. In other words, as we are passing through a saddle point p two circles merge or split. Figure 5 shows two types of saddles: a split saddle and a merge saddle. A saddle point p is a split saddle if $|f^{-1}(t_p + \epsilon)| - |f^{-1}(t_p - \epsilon)| = 1$, and it is a merge saddle if $|f^{-1}(t_p + \epsilon)| - |f^{-1}(t_p - \epsilon)| = -1$.

The sequential version of the algorithm relies on Lemma 4.1 to handle the case when the function f is Morse and has only simple saddle points. The case when f has saddle points with higher multiplicities will be handled in Section 4.5.

Remark 4.2. If p is a maximum and minimum point, then by definition, C_p will be the set that consists of the point p itself.

Ascending paths. The second main ingredient of the sequential Reeb graph algorithm is a collection of curves that we trace inside the manifold M using the function values. more precisely,

an *ascending path* from a non-maximum vertex v_0 , denoted by $apath(v_0)$, is defined to be a finite sequence of consecutive edges $\{[v_0, v_1], \dots, [v_{k-1}, v_k]\}$ on M , such that $[v_i, v_{i+1}]$ is an edge on M for $0 \leq i \leq k-1$, $f(v_{i+1}) > f(v_i)$, and v_k is a maximum, a boundary or a saddle vertex.

4.2. Outline of the sequential algorithm. We present now the outline of the sequential Reeb graph algorithm. We assume that we are given a triangulated PL Morse function $f : M \rightarrow [a, b] \subset \mathbb{R}$ defined on triangulated mesh M without boundary. The case when the function f is not Morse, or when the M has a boundary, will be discussed in later sections.

The sequential Reeb graph algorithm is given in the following steps:

- (1) We start by sorting the critical points of f by their critical values. Let CP be the set of the sorted critical points of f in an ascending order.
- (2) For each critical point of the function f we define a node in the in the Reeb graph $R(M, f)$. In other words, the node set of the graph $R(M, f)$ corresponds precisely to the set of critical points of the function f defined on M .
- (3) For each critical point v in CP we compute the critical set C_v .
- (4) For each saddle or a minimum vertex v in CP , we associated one or two ascending paths on the mesh: one ascending path if v is a minimum or a merge saddle and two paths if v is a split saddle. For each ascending path, we march with until this path intersects with the first critical C_w set with a higher critical value than of t_v . At this point we insert an edge for the Reeb graph $R(M, f)$ between the vertex v and the vertex w .

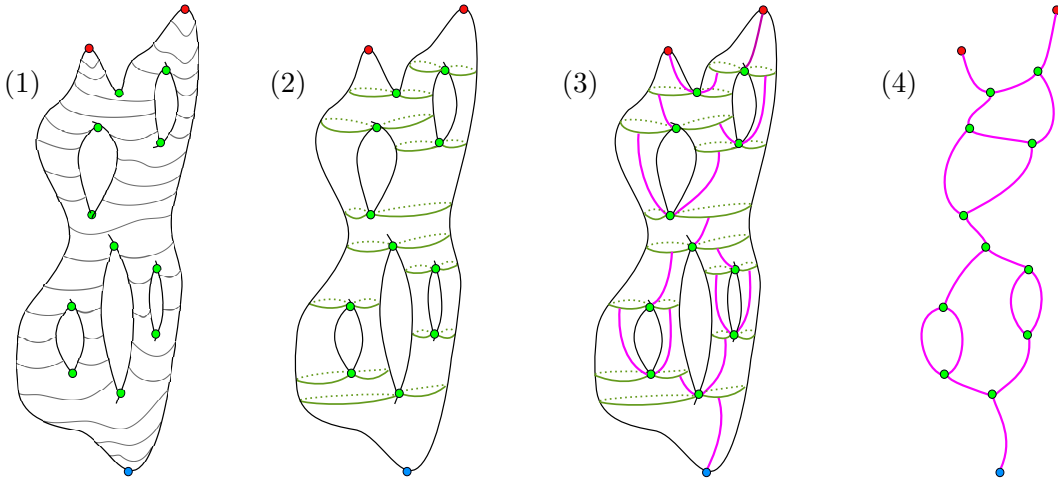


FIGURE 6. Summary of the sequential algorithm. (1) The input of the algorithm is a manifold M with a scalar function f . (2) Computing the critical sets. (3) For each saddle point or minimum we compute the ascending paths (4) Creating the edges of the Reeb graph by the information encoded in the start vertex of the ascending vertex and its termination critical set.

Figure 6 illustrates the steps of the algorithm. It remains to describe two aspects of the previous algorithm : the construction of the critical sets mentioned in step (3) and the construction of an the ascending paths mentioned in step (4).

4.2.1. Construction of the critical sets. We now describe how to compute the critical set of a critical point in the piece-wise setting. As before we assume that $f : M \rightarrow [a, b] \subset \mathbb{R}$ a piece-wise linear function defined on a triangulated surface M , and it takes distinct values on the vertices of M . This assumption will guarantee that for a given value t , the level curve $f^{-1}(t)$ intersects with at most one vertex of M .

Now let $t \in \mathbb{R}$. The *cross simplices* $CR_f(t)$ of the value t is the union of all simplices of M , which intersect with the level curve $f^{-1}(t)$. This is the set of vertices, edges and faces in M , which intersect with the level curve $f^{-1}(t)$. If $f(t) = v$ for a vertex v in $V(M)$, then we also define $\overline{CR_f(t)}$ to be $CR_f(t) \cup \overline{star(v)}$.

When t is the maximal or the minimal value, then $CR_f(t)$ consists of a single vertex v_t . In this case $\overline{CR_f(t)}$ is simply $\overline{star(v_t)}$. When t is a regular value, $CR_f(t)$ is a disjoint union of topological cylinders, that is $CR_f(t)$ appears as a "thickened" band around the curve $f^{-1}(t)$. Note that when the value t corresponding to a vertex v in $V(M)$ with $f(v) = t$ for some v in M , then $CR_f(t) \cap f^{-1}(t) = \{v\}$. When t is critical value of a saddle point then the curve $f^{-1}(t)$ consists of a finite collection of simple close curves that meet at the saddle point and the the set $\overline{CR_f(t)}$ can be seen as the thickened band of these curves. See Figure 7.

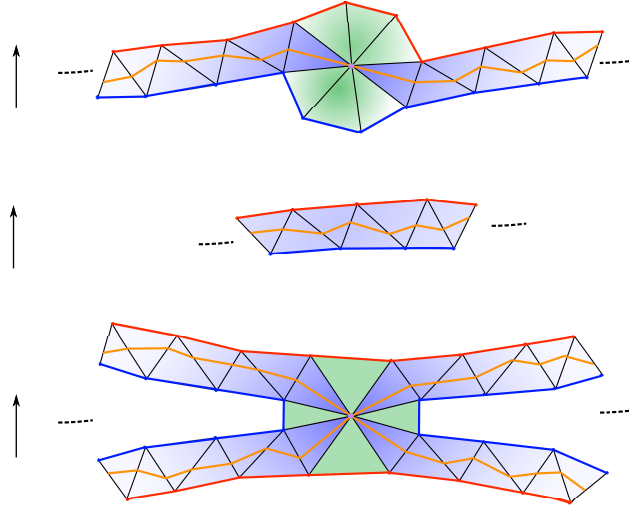


FIGURE 7. The set of vertices, edges and faces in the mesh M , which $f^{-1}(t)$ intersects. The top 2 figures depicts cases where t is a regular value. The top figure shows an example of when there exists a vertex v in M , such that $v \in f^{-1}(t)$. The second figure shows the case where no such a vertex exists, in other words $f^{-1}(t)$ intersects only edges and faces of M . The third figure is an example of the local neighborhood a critical point of f or when $f^{-1}(t)$ contains a critical point. The purple simplices represent the set $CR_f(t)$. The union of the green and purple simplices represent the set $\overline{CR_f(t)}$. The blue edges and nodes are the edges in the complexes shown in the figure and nodes in $\overline{CR_f(t)}$, which have f -values less than or equal to t . The red edges and nodes are the edges in the complexes shown in the figure and nodes in $CR_f(t)$, which have f -values higher than or equal to t .

The *lower level* of the cross subsimplex of a value t , denoted by $L(\overline{CR_f(t)})$, is the set of vertices and edges in $\overline{CR_f(t)}$ which have values less than or equal to t . Similarly, the *higher level* of the cross subsimplex of a value t , denoted by $H(\overline{CR_f(t)})$, is the set of vertices and edges in $\overline{CR_f(t)}$ which have values higher than or equal to t . See Figure 7.

Recall the notion of the connected component of a critical set from Section 4.1. Specifically, for a critical vertex v with a critical value t_v , we talked about the connected components of the critical set C_v . Using the definitions we introduced in this section we can computed the connected components of the critical set C_v in the piece-wise linear setting by considering the connected components of $L(\overline{CR_f(t_v)})$.

4.2.2. *Construction of the ascending paths.* The ascending paths from a critical point v_0 are specified as follows:

- (1) If v_0 is a merge saddle or a minimum then we initiate a single ascending path $apath(v_0)$ specified as follows. Let v_1 be a vertex in $Lk^+(v_0)$ such that $f(v_1) > f(v_0)$. At the k^{th} iteration $apath(v_0)$ consists of $\{[v_0, v_1], \dots, [v_{k-1}, v_k]\}$ with $f(v_{i+1}) > f(v_i)$ for $0 \leq i \leq k-1$.
- (2) If v_0 is a split saddle, then we start two ascending paths P_1 and P_2 starting from the point p specified as follows. Divide the set $Lk^+(p)$ into two disconnected components A and B . Choose the vertex v_A in A , such that $f(v_A) > f(v)$ for all $v \in A$, and choose the vertex v_B in a similar manner. At the k^{th} iteration P_1 consists of $\{[v_0, v_A], \dots, [v_{k-1}, v_k]\}$ with $f(v_i) > f(v_{i-1})$ for $0 \leq i \leq k-1$ (here we assume $v_1 = v_A$). The path P_2 is constructed similarly.

4.2.3. *Termination of an ascending path.* The condition at which we terminate the ascending paths we initiated in step (4) is specified as follows. Assume that at the k^{th} iteration an ascending path starting from the vertex p is $\{[v, v_1], \dots, [v_{k-1}, v_k]\}$. We continue this iteration until we arrive at an edge $E_n = [v_{n-1}, v_n]$ with $f(v_n) > f(v)$ where v is the critical point with a critical value right after the critical value of the critical point v . At this point we check the condition $C_w \cap E_n \neq \emptyset$. If this condition is satisfied then we insert an edge for the Reeb graph $R(M, f)$ between the vertex v and the vertex w . If $C_w \cap E_n = \emptyset$ then we keep marching until the critical point w with the lowest critical value that satisfies the condition $f(w) > f(v)$ as well as $C_w \cap E_n \neq \emptyset$. Note that w can be either a saddle or a maximum vertex.

The check of intersection between the edge E_n in an ascending path and a critical set C_w , mentioned in step (4), can be done by checking if E_n belongs with $CR_f(w)$.

Remark 4.3. It is important to notice how an ascending path corresponds to an edge in the Reeb graph. The ascending path starts at a critical point p with a critical value t_p and terminates at a critical set that corresponds to a critical point q with a critical value t_q with $t_q > t_p$. More precisely, an ascending path starts at one of the connected components of the upper link of a critical point p and ends at the one of the connected components of the critical sets C_q . If two ascending paths starts at two different connected components of the upper link of p but still end up in the *same connected component* of C_q then these two paths correspond to the exact same edge in the Reeb graph. Therefore only one of these ascending paths corresponds to an edge in the final Reeb graph. For this reason we say that each ascending path starting from a connected component of the upper star of a vertex gives rise to a *potential* edge in the Reeb graph. We give more details on this point in Section 4.5.

4.3. **Surfaces with boundaries.** In the case when the surface M has a boundary we modify the previous algorithm as follows. In this case $f^{-1}(\partial I) = f^{-1}(a) \cup f^{-1}(b)$ is not empty and consists of a finite collection of simply closed curves. We treat each connected component of $f^{-1}(a)$ as a minimal point, and we treat with the boundary $f^{-1}(b)$ as a maximum point. More precisely, the following modifications are added to the previous algorithm from Section 4.2.

- In step (2), each connected component in $f^{-1}(\partial I)$ is considered a vertex in the Reeb graph vertex set.
- In step (3), for each boundary component in $f^{-1}(a)$, we pick an arbitrary vertex on the boundary and initiate an ascending path starting.
- In step (4), if an ascending path starting at a vertex v reaches a boundary vertex w in one of the connected components, say $Bndry_w$, of $f^{-1}(b)$, then we insert an edge in the Reeb graph $R(M, f)$ between the vertex v and the vertex in $R(M, f)$ that corresponds to $Bndry_w$.

We denote the subcomplex obtained from M using the previous algorithm by $X_{M,f}$. In other words, $X_{M,f}$ consists of the critical sets C_p for all critical point p , as well as the ascending paths

we initiated at saddle, minimum or boundary vertices. When M and f are clear from the context, we will denote to $X_{M,f}$ simply by X .

4.4. Correctness of the sequential algorithm. For a function $f : M \rightarrow [a, b]$ and a $c \in \mathbb{R}$ define:

$$M_c := \{x \in M | f(x) \leq c\}.$$

Note that we allow M_c to be empty. Moreover, define :

$$M_{[c,d]} := \{x \in M | c \leq f(x) \leq d\}.$$

The correctness of our algorithm relies on the following two facts :

- (1) The only topological changes to the level sets of f occur when as pass a critical point. This is stated formally in Theorem 4.4.
- (2) The structure of the manifold around a critical point is completely determined by the index of that critical point. We give this in Theorem 4.5.

The proof of Theorems 4.4 and 4.5 can be found in [48].

Theorem 4.4. Let $f : M \rightarrow [a, b]$ be a smooth function on a smooth surface M . For two reals a, b , with $a < b$, if f has no critical values in the interval $[a, b]$, then the surfaces M_a and M_b are homeomorphic.

Our algorithm relies on Theorem 4.4. Namely, as we trace an ascending paths, we assume that no topological change occur until we reach the next critical point. The ascending path may not terminate at the next critical point, provided the part of the manifold in which this path is traveling in has not changed its topology. This will more evident after we provide the next theorem, which shows the exact structure of $M_{[c,d]}$ around a critical point.

Theorem 4.5. Let M be a compact connected surface, possibly with boundary and let $f : M \rightarrow [a, b]$ be a Morse function on M . Let p be a critical point, and let t be its corresponding critical value. Let $\epsilon > 0$ be small enough so that $I_\epsilon := [t - \epsilon, t + \epsilon]$ has only the critical value t .

- (1) If $index(p)$ is equal to 0 or 2, then M_{I_ϵ} is homeomorphic to a disjoint union of a disk and a finite number of topological cylinders (that is a genus zero surface with two boundary components).
- (2) If $index(p) = 1$, then M_{I_ϵ} is homeomorphic to a pair of pants and finite number of topological cylinders.

The previous two theorems show that for a given Morse f function on M , we can arrange M so that at each critical point only a single topological event occurs, and this topological event occurs around the critical point. Moreover, we know exactly what topological event occurs by the index of the critical point. This is illustrated in the Figure 8.

This shows that any 2-manifold can be built from the building blocks in Figure 9. Moreover, Theorem 4.5 shows that the restriction of Reeb graph of the restriction of f on M_{I_ϵ} has the shapes given in Figure 10. In other words, this gives us the structure of the Reeb graph around an interval I_ϵ that contains a single critical value.

Moreover, subcomplex $X_{M,f}$ constructed in the algorithm has the same Reeb graph structure of M around the critical points. More specifically, the algorithm creates an ascending path for each minimum and merge critical point and two such paths for a split saddle. Moreover, the algorithm terminates as ascending path when it passes from a critical set. See Figure 10.

Now let $e = (p, q)$ be an edge in a Reeb graph $R(M, f)$ connecting between the two nodes p and q , which correspond to critical points of f . For each such edge, we can find a preimage arc E in M that is mapped to $R(M, f)$ via the map $\pi : M \rightarrow R(M, f)$. The preimages of the points p and q under π are the critical set C_p and C_q in M , respectively. Hence each arc E must start at a point A in C_p and end at a point in B in C_q . Each such an arc on M characterizes the edge e .

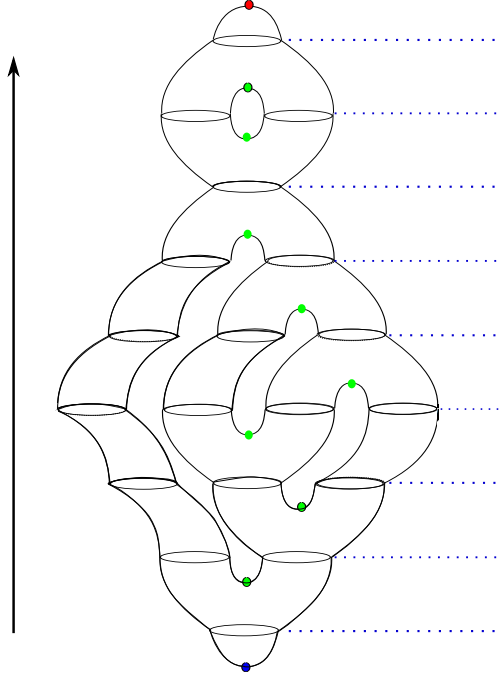


FIGURE 8. Given a Morse function f on a surface M we can slice M so that around each critical value t the submanifold $M_{[t-\epsilon, t+\epsilon]}$ is a disjoint union of simple building blocks: pair of pants, topological cylinders and topological disks.

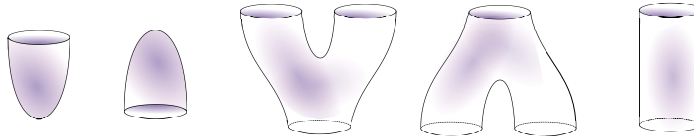


FIGURE 9. The building blocks of a surface. Given a scalar function f defined on a surface M , Theorem 4.5 asserts that we can decompose the surface into the building block pieces appear in above. These pieces are: genus zero surface with a single boundary component (disk), genus zero surface with 3 boundary components (pair of pants) and genus zero surface with 2 boundary components (cylinder).

An ascending path constructed in the sequential algorithm essentially traces an arc in the way described above. Namely, for a critical point p , with a critical value $f(p)$, an ascending path created at a point p will terminate at a critical point point q with $f(q) > f(p)$. This termination occurs when we pass through the critical set of the point q .

4.5. Dealing with the degenerate cases. It is possible in practice to obtain a scalar function with saddle points that have multiplicity $m \geq 2$. The algorithm that we present in Section 4.2 can be extended to handle such cases. We need to do the following modifications.

- In step (3) we calculate the connected components of the critical set C_p . Here it is not enough to calculate the critical set C_p . We need to also to compute the connected components of this set. This was explained in Section 4.2.1.
- When the multiplicity m of a split saddle p is greater than or equal to 2 we create an ascending path for each connected component in $Lk^+(p)$.

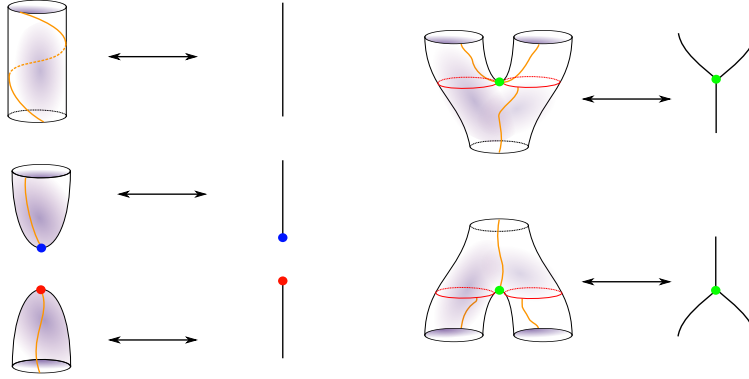


FIGURE 10. The restriction of the Reeb graph on the building blocks of a surface along with the part of the ascending paths in that part of the surface. Around each critical points the structure of the ascending curves (orange) is identical to the structure of the Reeb graph. More precisely the quotient space of the restriction of the function on the ascending curves the critical sets around a critical point (these are the curves highlighted by orange and red int the figure) is identical to the quotient space of the manifold locally.

In the non-degenerate case every ascending path corresponds to an edge in the Reeb graph. This is not the case anymore in the degenerate case. We describe next the sequential algorithm of the Reeb graph when f has degenerate critical points.

- (1) Sort the critical points of f an in ascending order. Let $CP = \{v_1, \dots, v_n\}$ be this set. This represents the vertex set of the Reeb graph, as we did before.
- (2) For each v in CP , compute the connected components of C_v and include all these component in a single container \mathcal{S} . We will denote by B_i^j to be the connected component j of the critical set C_{v_i} . In this way we index all elements in \mathcal{S} .
- (3) Declare each set B_i^j of \mathcal{S} as not visited.
- (4) For each critical point v_i in CP and for each component in $Lk^+(v_i)$, we initiate an ascending path P as described in Section 4.2.2. For each such path, we determine the connected component B_k^l in \mathcal{S} , which the path terminates at as described in Section 4.2.1. We have two cases:
 - (a) If the connected component B_k^l is not visited, then we insert an edge between v_i and v_k and mark the component B_k^l as visited.
 - (b) If the connected component B_k^l is visited, then we do not make any changes to the Reeb graph and terminate the current ascending path. In this case the ascending path corresponds to an edge that already exists in the Reeb graph. See Remark 4.3.

Note that the above procedure can be used to determine an the edges originating from a simple saddle. Namely, we do not need to check the type of the simple saddle point (merge or split) in step (3) of the algorithm given in Section 4.2, and for any saddle point we use the above procedure instead.

5. PARALLELIZATION OF THE ALGORITHM

In this section we give the details of our strategy to compute the Reeb graph in parallel. We describe the three stages of the parallel algorithm as follows.

- (1) **The partition stage.** In this stage we partition the manifold M into submanifolds such that the vertices counts of each submanifold are approximately equal to each other.

- (2) **Computing the Reeb graph for the submanifold stage.** Computing the Reeb graphs for each submanifold obtained from stage one concurrently.
- (3) **The gluing stage.** In this step we glue the Reeb graphs obtained from the stage 2.

5.1. The partition stage. In this first stage of the parallel algorithm we sort the vertices of the manifold with respect to the function f . This step can be done efficiently in parallel [68, 74]. The critical sets for saddle points are then determined by assigning a thread to each saddle point. This computation is only necessarily to determine the number of ascending paths that we need to initiate from that saddle point. Next we partition the manifold M to submanifolds along certain regular values of the scalar function f . More precisely the partition stage is given as follows:

- (1) Compute the critical points of f by assigning a thread to each vertex in M . Let p_1, p_2, \dots, p_n be the list critical points of f , and let t_1, t_2, \dots, t_n be their corresponding critical values.
- (2) We choose k regular values $C = \{c_1, \dots, c_k\}$ of f . These values will be utilized to slice the manifold M into $k + 1$ submanifolds M_1, \dots, M_{k+1} , such that the vertex count of the submanifolds are approximately equal to each other. We also need to determine the connected components $f^{-1}(c_i)$ for c_i in C . The connected components of the regular value $c_i \in C$ can be computed in linear time with respect to the number of edges in M as follows. We visit all edges of M and detect if an edge crosses one of the values c_i . If such an edge is found at a level c_i , then we keep rotating around to find all other edges crossing the value c_i within the same connected component of $f^{-1}(c_i)$. After visiting all edges, we also will have determined the connected components of $f^{-1}(c_i)$ for each $c_i \in C$. We will denote the set of all connected components of $f^{-1}(c_i)$ for $1 \leq i \leq k$ by \mathcal{C}_f . We also call an edge in M that crosses $f^{-1}(c_i)$ a *crossing edge*. See Figure 11 for an illustration.
- (3) Divide the surface M into $k + 1$ partitions along the level sets $f^{-1}(c_i)$ for all $c_i \in C$. We obtain a list of submanifolds $M_{[c_0, c_1]}, \dots, M_{[c_k, c_{k+1}]}$. Here we set $c_0 = t_1$ and $c_{k+1} = t_n$. We will denote to $M_{[c_{i-1}, c_i]}$ by M_i . The set $\{M_i | 1 \leq i \leq k + 1\}$ will be denoted by \mathcal{M}_f .
- (4) Next we extend M_i by adding other vertices in M as follows. Let E_{i-1} and E_i be the edges in M that intersect with $f^{-1}(c_{i-1})$ and $f^{-1}(c_i)$, respectively. The submanifold $M'_i = M'_{[c_{i-1}, c_i]}$ is obtained from $M_{[c_{i-1}, c_i]}$ by adding the vertices from E_{i-1} and E_i . We call the edges E_{i-1} and E_i the boundary edges of $M'_{[c_{i-1}, c_i]}$. Note that every two consecutive submanifolds from \mathcal{M}_f intersect with each other along their boundary edges. See Figure 12 for an illustration.

The purpose of extending the submanifolds M_i to M'_i in step (3) will be justified in the gluing stage in Section 5.3.

5.2. Computing the Reeb graph of each submanifold. The manifold M'_i is in general homeomorphic to the manifold M_i , since the former is obtained from the latter by extending its boundary slightly. However, in the piece-wise linear case the extension specified in the previous section may change the topology of the manifold. This will occur when a crossing edge in E_i or E_{i-1} contain a critical vertex of f . When this case occurs, we exclude this vertex from M'_i in order to keep it homeomorphic to M_i . Using this convention, we can assume that the Reeb graph of the restriction of f on M_i is identical to the Reeb graph of the restriction of the Reeb graph of f on M'_i .

Computing the Reeb graphs $R(M'_i, f)$ for M'_i , for $1 \leq i \leq k + 1$, now goes as follows. To each connected component in M'_i for $1 \leq i \leq k + 1$, we assign a thread and the Reeb graph $R(M'_i, f)$ on the submanifold M'_i , which can be computed concurrently.

An ascending path that starts at a crossing edge or ends at an crossing edge will be treated specially. We call the Reeb graph edge that corresponds to such an ascending path a *crossing arc*. Furthermore if the starting or the ending edge of this ascending path is a crossing edge then we will call the corresponding node in the Reeb graph a *crossing node*. Every crossing node is determined by its crossing edge. In other words, given a crossing edge we can recover its crossing node in the graphs $R(M'_i, f)$ for $1 \leq i \leq k + 1$. In practice we need to be able to do this in constant time,

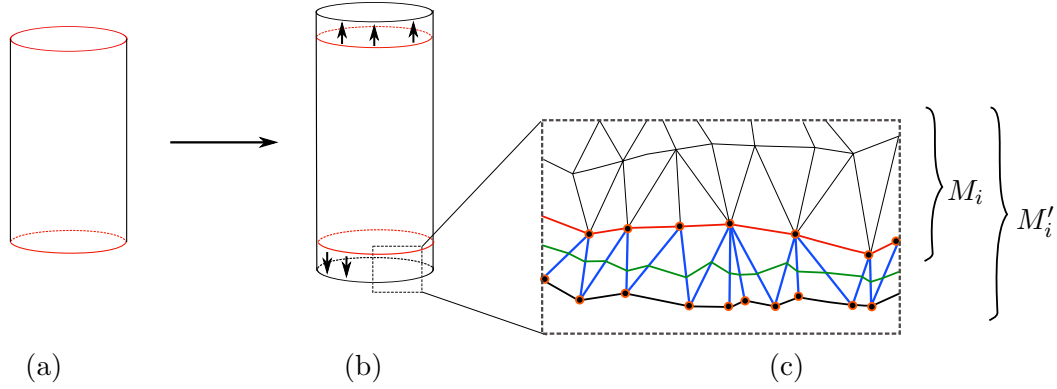


FIGURE 11. An illustration of the partition stage. (a) A connected component of the submanifold M_i is obtained. (b) M_i is extended to M'_i by adding the crossing edges. (c) A closeup of the crossing edges of the manifold M'_i . The blue edges represent the crossings edges of E_{i-1} , and the green curve represents the portion of the curve $f^{-1}(c_{i-1})$ within the closeup region.

so we create a global map G that takes as an input a crossing edge and returns its corresponding crossing node. In the case when a single crossing edge is associated with two crossing nodes from two consecutive submanifolds then the map G associates that edge crossing edge to the two crossing nodes (this occur when is the ending edge of a crossing arc and the starting edge of the crossing arc in the consequent submanifold are the same). If the crossing edge is not associated with any crossing node, then this map returns a constant value indicating that this edge is not a starting or an ending crossing edge for any ascending path. Note that each connected component in \mathcal{C}_f has either two crossing edges that are associated with two crossing nodes or a single crossing edge that is associated with two crossing nodes. This map will be utilized in the gluing stage.

5.3. The gluing stage. For the gluing stage we need to glue the nodes of the Reeb graphs that occur in duplication \mathcal{C}_f . For this purpose we utilize the function G that we constructed in Section 5.2. For each connected component in \mathcal{C}_f , we visit its crossing edges and check if two edges within that connected component have been flagged by G . If this is the case, then we retrieve the crossing nodes that corresponds to these two edges via the function G and glue them. In the case when connected component of \mathcal{C}_f has a single crossing edge, then we retrieve the two crossing nodes that corresponds associated with that edge and glue them. When we finish visiting all connected components of \mathcal{C}_f , all duplicate nodes will have been glued, and the final graph is $R(M, f)$. See Figure 12 for an illustration of this process.

6. GOING FROM THE REEB GRAPH TO THE MANIFOLD

In this section we give an algorithm that describes an explicit description of the map $F : R(M, f) \rightarrow M$. This map associates to every vertex v in the graph $R(M, f)$ to the critical set C_v associated to that critical point. More importantly, for each interior point p of an edge in $R(M, f)$, we want to associate a simple closed curve Cr in M such that $\pi(Cr) = p$.

In the construction above, for our Reeb graph algorithm an edge of the graph $R(M, f)$ is traced as a sequence of edges on the mesh running between two critical sets of the function f . This gives an embedding of the edges of the Reeb graph on the surface. This embedding is used to get the circle the corresponding to any points on the graph. More preciously we have the following correspondence. Let e be an edge of the graph $R(M, f)$. By the construction of our algorithm, every edge in $R(M, f)$ is determined by two critical points and a sequence of oriented edges on the mesh. Let $\mathcal{E}_e := \{E_1, \dots, E_n\}$ be the sequence of oriented edges on the mesh M that corresponds

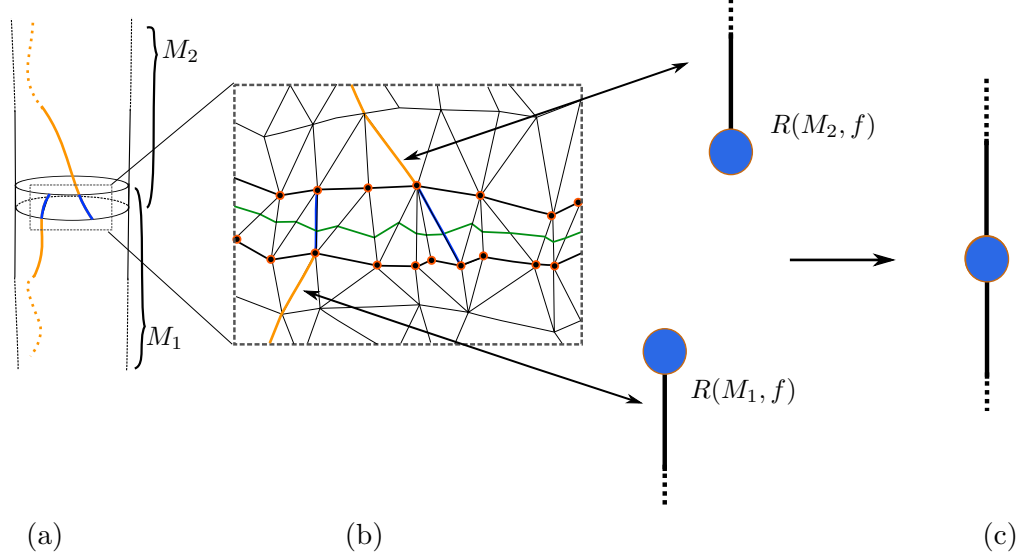


FIGURE 12. An illustration of the gluing stage. (a) At this stage two ascending paths from all submanifolds have been calculated. In the case when two consecutive submanifolds share a crossing edges such as the case in the illustrative figure the ascending path of the lower submanifold M_1 gets terminated at the at a crossing edge. Similarly an ascending path from the higher submanifold M_2 get initiated at a crossing edge. (b) A zoom to the figure (a) shows the two crossing edges that determine uniquely two nodes in the Reeb graphs $R(M_1, f)$ and $R(M_2, f)$. The fact that these two edges belong to the same connected component in the inverse image of the regular crossing value is used to glue the graphs $R(M_1, f)$ and $R(M_2, f)$ along the blue nodes to obtain the graph in Figure (c).

to the edge e . If $E_i = [v_i, v_{i+1}]$ then we will denote by $l_f(E_i)$ to $|f(v_i) - f(v_{i+1})|$. Let $T_j(\mathcal{E}_e)$ be the summation $\sum_{i=1}^j l_f(E_i)$ where $1 \leq j \leq n$.

An interior point p on the edge e is specified by giving a value t_p in the interval $(0, 1)$. To obtain the circle Cr_p on the mesh M that correspond to p we do the following procedure:

- (1) Map the interval $(0, 1)$ linearly to the interval $(0, T_n(\mathcal{E}_e))$.
- (2) Use the previous linear function to map t_p to its corresponding value t' in $(0, T_n(\mathcal{E}_e))$.
- (3) Determine the edge $E_k = (p_k, p_{k+1})$ in \mathcal{E}_e such that $T_{k-1}(\mathcal{E}_e) < t' \leq T_k(\mathcal{E}_e)$.
- (4) Now we need to find the value t , in the range of $[a, b]$, the range of f , such that $f^{-1}(t)$ contains Cr_p . We know that this value corresponds to t' , which lies in the interval $[T_{k-1}, T_k]$. The required t lies in interval $[f(p_k), f(p_{k+1})]$. Hence, we map the interval $[T_{k-1}, T_k]$ linearly to $[f(p_k), f(p_{k+1})]$ and determine the value t in $[f(p_k), f(p_{k+1})]$ that corresponds to t' .
- (5) The required circle Cr_p is the connected component of $f^{-1}(t)$ that contains the edge E_k .

Note that in the previous procedure we rely on the fundamental assumption that f takes different values on the vertices of the mesh.

7. RUN-TIME ANALYSIS AND IMPLEMENTATION

We tested the presented algorithm on meshes with various complexities. Our experiments were done on a AMD FX 6300 6-Core with 32 GB memory. The algorithm was implemented in C++ and Windows platform was used.

We test our test our parallel algorithm with the AIM@SHAPE Repository. This initial attempt did not give us an increase of performance over the sequential algorithm for most meshes available

in the above dataset. In order to take advantage of the parallel implementation, we tested our algorithm on high resolution meshes. Specifically, we uniformly increase the resolution of the meshes available AIM@SHAPE library to $200k$. Our implementation on the described above data gave us a minimum speedup of 3.6, a maximum one of 4.3 and an average speedup of 3.8 on 6 cores. Using this we obtain a 63% average parallel efficiency. The details are given in Figure 13. The x -axis represents the number of processes and the y -axis shows the speedup. The figure shows how the trends in the curves are increasing monotonically as the number of processes increase.

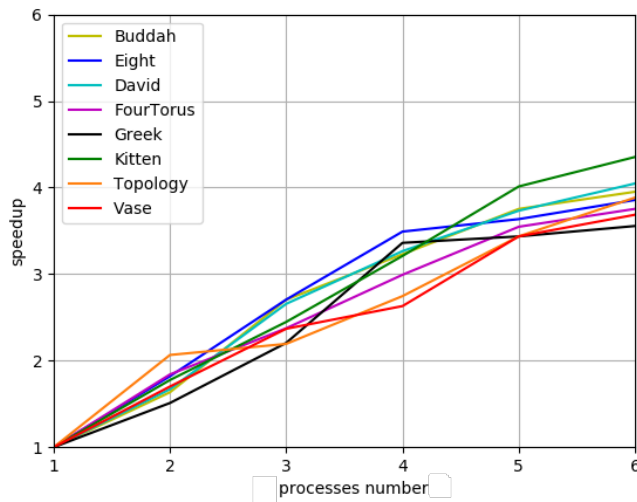


FIGURE 13. Speedups obtained by our parallel Reeb graph algorithm.

8. APPLICATIONS

Reeb graphs on a surface M encodes a rich amount of topological information from the original surface. In this section we show how the Reeb graph of a surface M gives rise to a natural collection of simple closed curves on M . The applications that we present can be described in terms of these curves. We first define these curves and show their relation with the Reeb graph. We then we give a procedure to extract them by utilizing the tools we described in Section 6. Finally, the curves are utilized to obtain two mesh segmentation algorithms.

While almost all Reeb graph-based segmentation algorithm focus on genus zero surfaces, see for instance [73, 81, 82], we present here two mesh segmentation algorithms for surfaces with higher genus.

Let $R(M, f)$ be a surface M and a scalar function f . The edges of the Reeb graph $R(M, f)$ determine the following types of simple closed curves on M :

- Cutting system curves.
- Pants decomposition curves.
- Branch curves.

We describe how a Reeb graph on a surface can be used to realize these curves next.

Cutting System Curves. Reeb graphs can also be used to determined the so called *cutting system*. A cutting system for a connected, closed, orientable surface M of genus g is a collection of unordered disjoint simple closed loops embedded in M whose complement $M \setminus (l_1 \sqcup \dots \sqcup l_g)$ is a sphere with $2g$ boundary components [36]. Segmenting a surface along a cutting system curve yield a genus zero surface with multiple boundary components. Hence this can be used to aid in

mesh parametrization. See for instance [84] and the references therein. We describe here a Reeb graph-based algorithm to obtain a cutting system. The algorithm goes as follows:

- (1) Let T be a spanning tree of $R(M, f)$ and consider the edges e_1, \dots, e_g in $R(f) \setminus T$.
- (2) Select an interior point in e_i , for $0 \leq i \leq g$.
- (3) Each interior point selected in the previous step determines a loop l_i which can be obtained using the Reeb graph algorithm we described here.

Figure 14 show multiple examples of cutting system curves on triangulated surfaces.



FIGURE 14. Examples of cut system curves on triangulated surfaces.

Pants Decomposition Curves. A Reeb graph naturally gives rise to a collection of curves that can be used to decompose a surface into a pair of pants. A pair of pants is a genus zero surface with three boundary components. Beside surface segmentation [33], surface pants decomposition has found applications in mesh parametrization [42], surface matching [44] and surface classification and indexing [39]. The method to obtain a pants decomposition from the Reeb graph is illustrated in Figure 15 and is described as follows:

- (1) Let $R'(M, f)$ be the deformation retract of $R(M, f)$. This graph can be obtained by recursively deleting nodes with valency one from $R(M, f)$ and the edge attached to them until no such indices exist. We exclude from this deletion the 1-valence nodes originating from the boundary of M . We also delete all nodes with valency 2 and combine the two edges that meet at that node to form a single edge. This step is illustrated in step (3) Figure 16.
- (2) We select one interior point from each edge in $R'(M, f)$, provided this edge does not have a node of valency one.
- (3) We use our Reeb algorithm to determine the curves on the surface that correspond to the points that we selected on the graph in the previous step.

The results of the previous algorithm was tested on triangulated meshes with various topological complexities. We show some examples in Figure 16.

Branch Curves. A branch curve on a surface M is a simple closed curve that bounds a topological disk on M . Such a curve is also called null homotopic. A branch curve is determined by a Reeb graph edge which has a vertex of valency one. Note that cutting the surface along a branch curve increases the number of connected components of the surface. Cutting along such curves can be used for segmentation of a genus zero surfaces. There are many Reeb graph-based segmentation algorithms in the literature for segmentation of genus zero surfaces such as the surfaces obtained from a humanoid character. See for instance [73, 81, 82]. Since this type of curves is essentially utilized elsewhere in the literature, we simply list it here for completeness of our discussion.

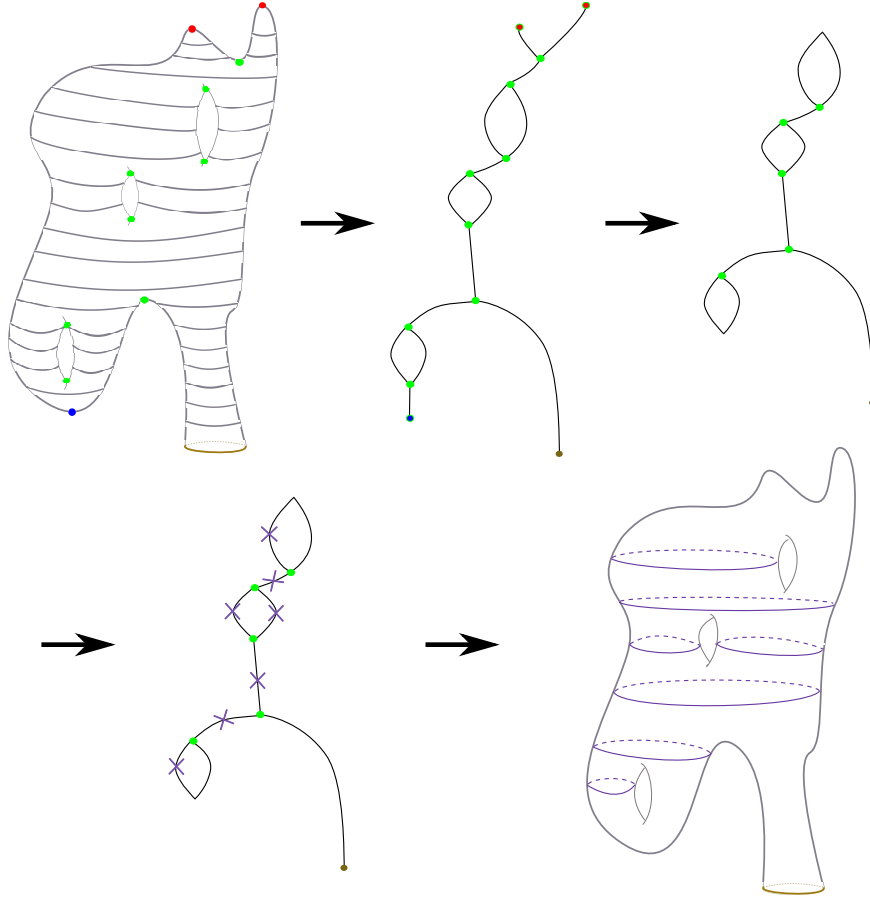


FIGURE 15. A Reeb graph can be used to segment a surface into a collection of topologically consistent patches. (1) The surface M with a scalar function f are given as an input. (2) The Reeb graph of $R(M, f)$ is calculated. (3) The deformation retract graph $R'(M, f)$ of the graph $R(M, f)$ is calculated. (4) For each edge in the graph $R'(M, f)$ we select a point. (5) We select the curves on the manifold M that corresponds to the points selected in the previous step.

All these methods however lack the description of a method to extract the manifold data from the Reeb graph data.

9. CONCLUSION AND FUTURE WORK

Parallelization of topological data analysis algorithms is still in its infancy. There are plenty of existing topological machineries such as Morse theory that offer a plethora of tools which can be utilized to obtain robust and efficient parallel algorithms. In this paper we presented a work that utilizes Morse theory to obtain a parallel algorithm for Reeb graph.

The parallel algorithm that we present here has the elements that makes it generalizable to a Reeb graph algorithm on a general simplicial complex. However, we thought that this would make the discussion more complicated in many parts of the algorithm. We plan to peruse this direction in a future work.

ACKNOWLEDGEMENTS

This work was supported in part by the National Science Foundation (IIS-1513616).

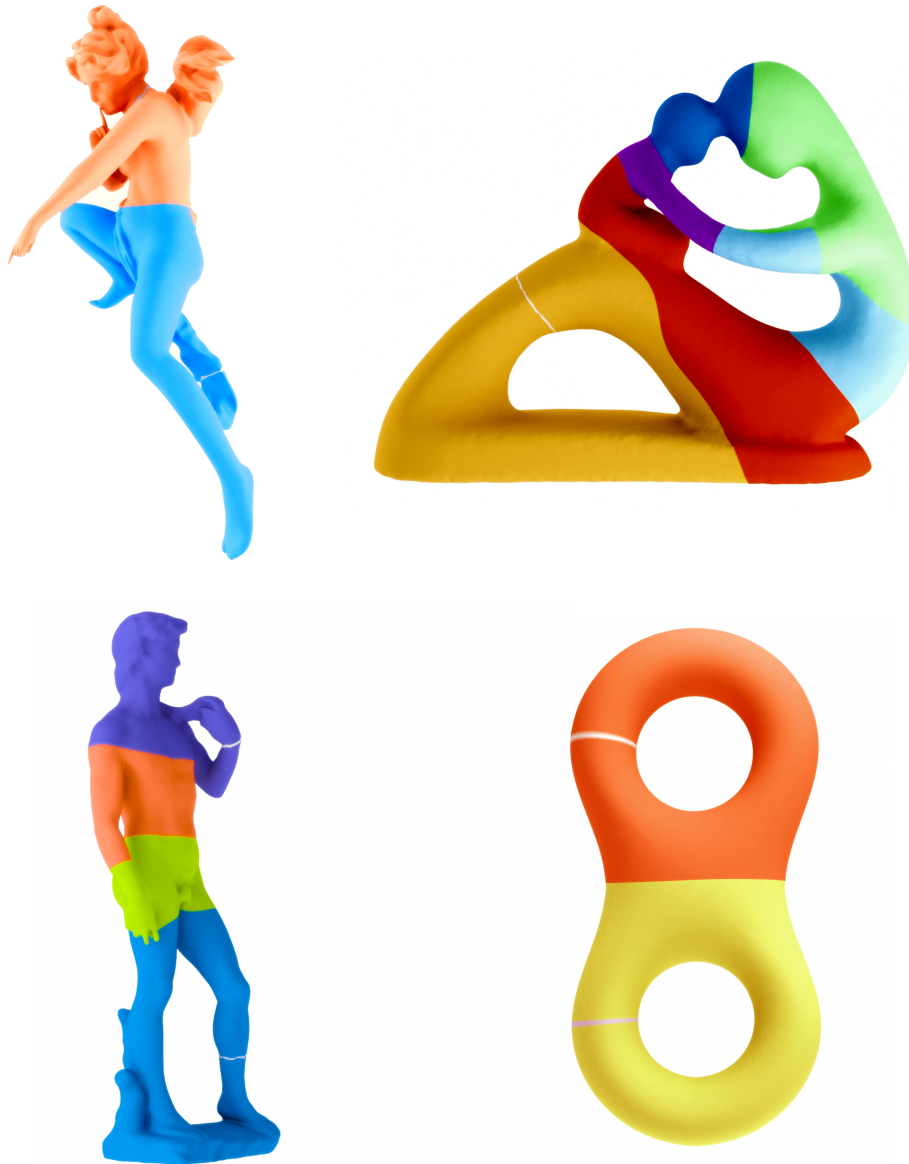


FIGURE 16. Examples of segmentation of higher genus surfaces into pair of pants using our Reeb graph algorithm.

REFERENCES

- [1] Marco Attene, Silvia Biasotti, and Michela Spagnuolo, *Shape understanding by contour-driven retiling*, *The Visual Computer* **19** (2003), no. 2, 127–138.
- [2] Chandrajit L Bajaj, Valerio Pascucci, and Daniel R Schikore, *The contour spectrum*, *Proceedings of the 8th conference on visualization'97*, 1997, pp. 167–ff.
- [3] Thomas Banchoff et al., *Critical points and curvature for embedded polyhedra*, *J. Diff. Geom* **1** (1967), no. 245-256, 34.
- [4] Augustin Banyaga and David Hurtubise, *Lectures on morse homology*, Vol. 29, Springer Science & Business Media, 2013.
- [5] Paul J Besl and Neil D McKay, *Method for registration of 3-d shapes*, *Robotics-dl tentative*, 1992, pp. 586–606.
- [6] Silvia Biasotti, Bianca Falcidieno, and Michela Spagnuolo, *Extended reeb graphs for surface understanding and description*, *Discrete geometry for computer imagery*, 2000, pp. 185–197.

- [7] Silvia Biasotti, Daniela Giorgi, Michela Spagnuolo, and Bianca Falcidieno, *Reeb graphs for shape analysis and applications*, Theoretical Computer Science **392** (2008), no. 1-3, 5–22.
- [8] Silvia Biasotti, Michela Mortara, and Michela Spagnuolo, *Surface compression and reconstruction using reeb graphs and shape analysis*, Spring conference on computer graphics, 2000, pp. 174–185.
- [9] Roger L Boyell and Henry Ruston, *Hybrid techniques for real-time radar simulation*, Proceedings of the november 12-14, 1963, fall joint computer conference, 1963, pp. 445–458.
- [10] Gunnar Carlsson, *Topology and data*, Bulletin of the American Mathematical Society **46** (2009), no. 2, 255–308.
- [11] Gunnar Carlsson and Facundo Mémoli, *Persistent clustering and a theorem of j. kleinberg*, arXiv preprint arXiv:0808.2241 (2008).
- [12] Hamish Carr, *Topological manipulation of isosurfaces*, Ph.D. Thesis, 2004.
- [13] Hamish Carr, Jack Snoeyink, and Michiel van de Panne, *Simplifying flexible isosurfaces using local geometric measures*, Visualization, 2004. ieee, 2004, pp. 497–504.
- [14] Mathieu Carrière and Steve Oudot, *Structure and stability of the 1-dimensional mapper*, arXiv preprint arXiv:1511.05823 (2015).
- [15] Frédéric Chazal, Leonidas J Guibas, Steve Y Oudot, and Primoz Skraba, *Analysis of scalar fields over point cloud data*, Proceedings of the twentieth annual acm-siam symposium on discrete algorithms, 2009, pp. 1021–1030.
- [16] Frédéric Chazal and Steve Yann Oudot, *Towards persistence-based reconstruction in euclidean spaces*, Proceedings of the twenty-fourth annual symposium on computational geometry, 2008, pp. 232–241.
- [17] Yi-Jen Chiang, Tobias Lenz, Xiang Lu, and Günter Rote, *Simple and optimal output-sensitive construction of contour trees using monotone paths*, Computational Geometry **30** (2005), no. 2, 165–195.
- [18] Kree Cole-McLaughlin, Herbert Edelsbrunner, John Harer, Vijay Natarajan, and Valerio Pascucci, *Loops in reeb graphs of 2-manifolds*, Proceedings of the nineteenth annual symposium on computational geometry, 2003, pp. 344–350.
- [19] ———, *Loops in reeb graphs of 2-manifolds*, Discrete & Computational Geometry **32** (2004), no. 2, 231–244.
- [20] Tamal K Dey and Kuiyu Li, *Cut locus and topology from surface point data*, Proceedings of the twenty-fifth annual symposium on computational geometry, 2009, pp. 125–134.
- [21] Tamal K Dey, Kuiyu Li, Jian Sun, and David Cohen-Steiner, *Computing geometry-aware handle and tunnel loops in 3d models*, Acm transactions on graphics (tog), 2008, pp. 45.
- [22] Tamal K Dey, Facundo Memoli, and Yusu Wang, *Topological analysis of nerves, reeb spaces, mappers, and multiscale mappers*, arXiv preprint arXiv:1703.07387 (2017).
- [23] Tamal K Dey, Jian Sun, and Yusu Wang, *Approximating cycles in a shortest basis of the first homology group from point data*, Inverse Problems **27** (2011), no. 12, 124004.
- [24] Shen Dong, Peer-Timo Bremer, Michael Garland, Valerio Pascucci, and John C Hart, *Spectral surface quadrangulation*, Acm transactions on graphics (tog), 2006, pp. 1057–1066.
- [25] Harish Doraiswamy and Vijay Natarajan, *Efficient output-sensitive construction of reeb graphs*, International symposium on algorithms and computation, 2008, pp. 556–567.
- [26] ———, *Efficient algorithms for computing reeb graphs*, Computational Geometry **42** (2009), no. 6, 606–616.
- [27] Harish Doraiswamy and Natarajan Vijay, *Efficient algorithms for computing reeb graphs*, Computational Geometry **42** (2009), no. 6-7, 606–616.
- [28] Herbert Edelsbrunner, John Harer, Ajith Mascarenhas, and Valerio Pascucci, *Time-varying reeb graphs for continuous space-time data*, Proceedings of the twentieth annual symposium on computational geometry, 2004, pp. 366–372.
- [29] Xiaoyin Ge, Issam I Safa, Mikhail Belkin, and Yusu Wang, *Data skeletonization via reeb graphs*, Advances in neural information processing systems, 2011, pp. 837–845.
- [30] Xiaohu Guo, Xin Li, Yunfan Bao, Xianfeng Gu, and Hong Qin, *Meshless thin-shell simulation based on global conformal parameterization*, Visualization and Computer Graphics, IEEE Transactions on **12** (2006), no. 3, 375–385.
- [31] Satyandra K Gupta, William C Regli, and Dana S Nau, *Manufacturing feature instances: which ones to recognize?*, Proceedings of the third acm symposium on solid modeling and applications, 1995, pp. 141–152.
- [32] Mustafa Hajij, Basem Assiri, and Paul Rosen, *Distributed mapper*, arXiv preprint arXiv:1712.03660 (2017).
- [33] Mustafa Hajij, Tamal Dey, and Xin Li, *Segmenting a surface mesh into pants using morse theory*, Graphical Models **88** (2016), 12–21.
- [34] Mustafa Hajij, Bei Wang, and Paul Rosen, *Mog: Mapper on graphs for relationship preserving clustering*, arXiv preprint arXiv:1804.11242 (2018).
- [35] W. Harvey, R. Wenger, and Y. Wang, *A randomized $O(m \log m)$ time algorithm for computing Reeb graph of arbitrary simplicial complexes*, Proc. 25th annu. acm sympos. compu. geom., 2010, pp. 267–276.
- [36] Allen Hatcher and William Thurston, *A presentation for the mapping class group of a closed orientable surface*, Topology **19** (1980), no. 3, 221–237.

- [37] Franck Hétroy and Dominique Attali, *Topological quadrangulations of closed triangulated surfaces using the reeb graph*, Graphical Models **65** (2003), no. 1, 131–148.
- [38] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Tosiyasu L Kunii, *Topology matching for fully automatic similarity estimation of 3d shapes*, Proceedings of the 28th annual conference on computer graphics and interactive techniques, 2001, pp. 203–212.
- [39] Miao Jin, Wei Zeng, Ning Ding, and Xianfeng Gu, *Computing fenchel-nielsen coordinates in teichmuller shape space*, Shape modeling and applications, 2009. smi 2009. ieee international conference on, 2009, pp. 193–200.
- [40] PIZZANU Kanongchaiyos and YOSHIHISA Shinagawa, *Articulated reeb graphs for interactive skeleton animation*, Proceeding Modeling Multimedia Information and System (2000), 451–467.
- [41] In So Kweon and Takeo Kanade, *Extracting topographic terrain features from elevation maps*, CVGIP: image understanding **59** (1994), no. 2, 171–182.
- [42] Tsz-Ho Kwok, Yunbo Zhang, and Charlie CL Wang, *Constructing common base domain by cues from voronoi diagram*, Graphical Models **74** (2012), no. 4, 152–163.
- [43] Ryan H Lewis and Afra Zomorodian, *Multicore homology via mayer vietoris*, arXiv preprint arXiv:1407.2275 (2014).
- [44] Xin Li, Xianfeng Gu, and Hong Qin, *Surface mapping using consistent pants decomposition*, IEEE Transactions on Visualization and Computer Graphics **15** (2009), no. 4, 558–571.
- [45] David Lipsky, Primoz Skraba, and Mikael Vejdemo-Johansson, *A spectral sequence for parallelized persistence*, arXiv preprint arXiv:1112.1245 (2011).
- [46] PY Lum, G Singh, A Lehman, T Ishkanov, Mikael Vejdemo-Johansson, M Alagappan, J Carlsson, and G Carlsson, *Extracting insights from the shape of complex data using topology*, Scientific reports **3** (2013), 1236.
- [47] Yukio Matsumoto, *An introduction to morse theory*, Vol. 208, American Mathematical Soc., 2002.
- [48] John Willard Milnor, *Morse theory*, Princeton university press, 1963.
- [49] Waleed Mohamed and A Ben Hamza, *Reeb graph path dissimilarity for 3d object matching and retrieval*, the visual computer **28** (2012), no. 3, 305–318.
- [50] Dmitriy Morozov and Gunther Weber, *Distributed merge trees*, Acm sigplan notices, 2013, pp. 93–102.
- [51] Dmitriy Morozov and Gunther H Weber, *Distributed contour trees* (2012).
- [52] Marston Morse, *The calculus of variations in the large*, Vol. 18, American Mathematical Soc., 1934.
- [53] Elizabeth Munch and Bei Wang, *Convergence between categorical representations of reeb space and mapper*, arXiv preprint arXiv:1512.04108 (2015).
- [54] Mattia Natali, Silvia Biasotti, Giuseppe Patanè, and Bianca Falcidieno, *Graph-based representations of point clouds*, Graphical Models **73** (2011), no. 5, 151–164.
- [55] Xinlai Ni, Michael Garland, and John C Hart, *Fair morse functions for extracting the topological structure of a surface mesh*, ACM Transactions on Graphics (TOG) **23** (2004), no. 3, 613–622.
- [56] Monica Nicolau, Arnold J Levine, and Gunnar Carlsson, *Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival*, Proceedings of the National Academy of Sciences **108** (2011), no. 17, 7265–7270.
- [57] Salman Parsa, *A deterministic $O(m \log m)$ time algorithm for the Reeb graph*, Acm sympos. comput. geom. (socg), 2012, pp. 269–276.
- [58] Valerio Pascucci, Kree Cole-McLaughlin, and Giorgio Scorzelli, *Multi-resolution computation and presentation of contour trees*, Proc. iasted conference on visualization, imaging, and image processing, 2004, pp. 452–290.
- [59] Valerio Pascucci, Giorgio Scorzelli, Peer-Timo Bremer, and Ajith Mascarenhas, *Robust on-line computation of reeb graphs: simplicity and speed*, Acm transactions on graphics (tog), 2007, pp. 58.
- [60] Giuseppe Patane, Michela Spagnuolo, and Bianca Falcidieno, *Para-graph: Graph-based parameterization of tri-angle meshes with arbitrary genus*, Computer graphics forum, 2004, pp. 783–797.
- [61] Benjamin Raichel and C Seshadhri, *Avoiding the global sort: A faster contour tree algorithm*, arXiv preprint arXiv:1411.2689 (2014).
- [62] G. Reeb, *Sur les points singuliers d’une forme de pfaff complètement intergrable ou d’une fonction numérique (on the singular points of a complete integral pfaff form or of a numerical function)*, Comptes Rendus Acad.Science Paris **222** (1946), 847–849.
- [63] Alejandro Robles, Mustafa Hajij, and Paul Rosen, *The shape of an image: A study of mapper on images*, to appear VISAPP 2018 (2018).
- [64] Paul Rosen, Mustafa Hajij, Junyi Tu, and Tanvirul Arafin, *Using topological data analysis to infer the quality in point cloud-based 3d printed objects*, Proceedings of CAD 2018.
- [65] Paul Rosen, Bei Wang, Anil Seth, Betsy Mills, Adam Ginsburg, Julia Kamenetzky, Jeff Kern, and Chris R Johnson, *Using contour trees in the analysis and visualization of radio astronomy data cubes*, arXiv preprint arXiv:1704.04561 (2017).
- [66] Yoshihisa Shinagawa and Tosiyasu L Kunii, *Constructing a reeb graph automatically from cross sections*, IEEE Computer Graphics and Applications **11** (1991), no. 6, 44–51.

- [67] Gurjeet Singh, Facundo Mémoli, and Gunnar E Carlsson, *Topological methods for the analysis of high dimensional data sets and 3d object recognition.*, Spbg, 2007, pp. 91–100.
- [68] Johannes Singler, Peter Sanders, and Felix Putze, *Mcstl: The multi-core standard template library*, Euro-Par 2007 Parallel Processing (2007), 682–694.
- [69] Barton T Stander and John C Hart, *Guaranteeing the topology of an implicit surface polygonization for interactive modeling*, Proceedings of the 24th annual conference on computer graphics and interactive techniques, 1997, pp. 279–286.
- [70] Shigeo Takahashi, Yuriko Takeshima, and Issei Fujishiro, *Topological volume skeletonization and its application to transfer function design*, Graphical Models **66** (2004), no. 1, 24–49.
- [71] Sergey P Tarasov and Michael N Vyalys, *Construction of contour trees in 3d in $o(n \log n)$ steps*, Proceedings of the fourteenth annual symposium on computational geometry, 1998, pp. 68–75.
- [72] Julien Tierny, Attila Gyulassy, Eddie Simon, and Valerio Pascucci, *Loop surgery for volumetric meshes: Reeb graphs reduced to contour trees*, IEEE Transactions on Visualization and Computer Graphics **15** (2009), no. 6.
- [73] Julien Tierny, Jean-Philippe Vandeborre, and Mohamed Daoudi, *Topology driven 3d mesh hierarchical segmentation*, Shape modeling and applications, 2007. smi'07. ieee international conference on, 2007, pp. 215–220.
- [74] Philippas Tsigas and Yi Zhang, *A simple, fast parallel implementation of quicksort and its performance evaluation on sun enterprise 10000*, Parallel, distributed and network-based processing, 2003. proceedings. eleventh euromicro conference on, 2003, pp. 372–381.
- [75] Tony Tung and Francis Schmitt, *The augmented multiresolution reeb graph approach for content-based retrieval of 3d shapes*, International Journal of Shape Modeling **11** (2005), no. 01, 91–120.
- [76] Marc Van Kreveld, René van Oostrum, Chandrajit Bajaj, Valerio Pascucci, and Dan Schikore, *Contour trees and small seed sets for isosurface traversal*, Proceedings of the thirteenth annual symposium on computational geometry, 1997, pp. 212–220.
- [77] Hongyu Wang, Ying He, Xin Li, Xianfeng Gu, and Hong Qin, *Geometry-aware domain decomposition for t-spline-based manifold modeling*, Computers & Graphics **33** (2009), no. 3, 359–368.
- [78] Gunther H Weber and Gerik Scheuermann, *Topology-based transfer function design*, Proceedings of the second iasted international conference on visualization, imaging, and image processing, 2002, pp. 527–532.
- [79] Naoufel Werghi, Yijun Xiao, and Jan Paul Siebert, *A functional-based segmentation of human body scans in arbitrary postures*, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on **36** (2006), no. 1, 153–165.
- [80] Zoë Wood, Hugues Hoppe, Mathieu Desbrun, and Peter Schröder, *Removing excess topology from isosurfaces*, ACM Transactions on Graphics (TOG) **23** (2004), no. 2, 190–208.
- [81] Yijun Xiao, Paul Siebert, and Naoufel Werghi, *A discrete reeb graph approach for the segmentation of human body scans*, 3-d digital imaging and modeling, 2003. 3dim 2003. proceedings. fourth international conference on, 2003, pp. 378–385.
- [82] Yijun Xiao, Naoufel Werghi, and Paul Siebert, *A topological approach for segmenting human body shape*, Image analysis and processing, 2003. proceedings. 12th international conference on, 2003, pp. 82–87.
- [83] Ichitaro Yamazaki, Vijay Natarajan, Zhaojun Bai, and Bernd Hamann, *Segmenting point sets*, Shape modeling and applications, 2006. smi 2006. ieee international conference on, 2006, pp. 6–6.
- [84] Wei Zeng, Xiaotian Yin, Min Zhang, Feng Luo, and Xianfeng Gu, *Generalized koebe's method for conformal mapping multiply connected domains*, 2009 siam/acm joint conference on geometric and physical modeling, 2009, pp. 89–100.
- [85] Eugene Zhang, Konstantin Mischaikow, and Greg Turk, *Feature-based surface parameterization and texture mapping*, ACM Transactions on Graphics (TOG) **24** (2005), no. 1, 1–27.

OHIO STATE UNIVERSITY

E-mail address: hajij.1@osu.edu

UNIVERSITY OF SOUTH FLORIDA

E-mail address: prosen@usf.edu