Heuristic Search for Homology Localization Problem and Its Application in Cardiac Trabeculae Reconstruction

Xudong Zhang[1] , **Pengxiang Wu**[2] , **Changhe Yuan**[1,3] , **Yusu Wang**[4] ,
Dimitris Metaxas[2] and **Chao Chen**[5]
[1] CUNY Graduate Center, New York, NY, United States
[2] Rutgers University, New Brunswick, NY, United States
[3] CUNY Queens College, New York, NY, United States
[4] Ohio State University, Columbus, OH, United States
[5] Stony Brook University, Stony Brook, NY, United States
xzhang5@gradcenter.cuny.edu, pw241@rutgers.edu, changhe.yuan@qc.cuny.edu,
yusu@cse.ohio-state.edu, dnm@cs.rutgers.edu, chao.chen.1@stonybrook.edu

Abstract

Cardiac trabeculae are fine rod-like muscles whose ends are attached to the inner walls of ventricles. Accurate extraction of trabeculae is important yet challenging, due to the background noise and limited resolution of cardiac images. Existing works proposed to handle this task by modeling the trabeculae as topological handles for better extraction. Computing optimal representation of these handles is essential yet very expensive. In this work, we formulate the problem as a heuristic search problem, and propose novel heuristic functions based on advanced topological techniques. We show in experiments that the proposed heuristic functions improve the computation in both time and memory.

1 Introduction

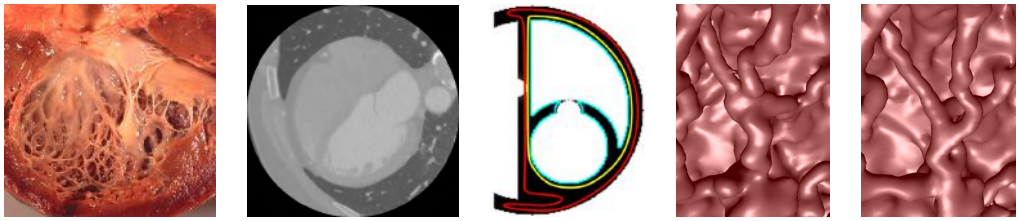
Cardiac trabeculae are fine muscle columns, whose ends attache the ventricular walls (Figure 1(a) [Edwin P. Ewing, 2016]). Those fine structures make up a large portion of left ventricle (23% volume at end-diastolic state), and thus play an important role in diagnosis of cardiac diseases and understanding of cardiac functionality. Modern imaging techniques (e.g., Computed Tomography (CT)) make it possible to capture the trabeculae, within cardiac ventricles (Figure 1(b)). However, it is challenging to accurately extract these complex structures due to the heterogeneity of their intensity and geometry. Existing segmentation models, which typically rely on smoothness and global shape priors [Zhu *et al.*, 1995; Boykov *et al.*, 2001; Cootes *et al.*, 1995], are insufficient for handling this task and tend to remove fine geometric details.

To solve this problem, Gao *et al*. [2013] proposed a topological method, in which salient topological handles (i.e., thickened loops) are detected as trabeculae. Figure 1(c) shows an example of topological handles in a 2D synthetic image. One major weakness with this method is that the detected handles are not ideally represented. A topological handle can be represented by any loop it contains. For example, the handle in Figure 1(c) can be represented by any of the red, green, and cyan cycles. A sub-optimal representative cycle leads to inaccurate geometric description of the handle, and thus impair downstream analysis. Figure 1(d) shows low-quality reconstruction of trabeculae which are not fully stretched as expected. Wu *et al*. [2017] showed that by finding the optimal representative cycle of each handle, we can obtain the ideal reconstruction results (Figure 1(e)).

However, computing the optimal representative cycle is indeed a very challenging problem. It has been proven that this problem, called *homology localization*, is NP-hard, and even NP-hard to approximate within constant ratio [Chen and Freedman, 2010a]. Various approximate algorithms [Chen and Freedman, 2010b] have been proposed, but cannot achieve the optimal result. Known exact algorithms, e.g., [Busaryev *et al.*, 2012], are exponential to the Betti number and thus are impractical. To efficiently solve this problem, we formulate it as a search problem. We assign a binary vector, called the homological annotation, to each edge of the underlying graph. In the graph, a loop represents the handle of interest if and only if its edge annotations sum up to a desired vector. This way, the problem of finding the shortest representative cycle of a handle is reduced into finding the shortest loop in a graph with a given sum of edge annotation.

This problem can be solved using an A* searching framework. Wu *et al*. [2017] proposed a heuristic function for the A* search. It takes the maximum over a set of heuristic functions, each of which focusing on matching one single bit of the given annotation. However, such scheme ignores the relationship between different bits, and may not be a tight approximation of the real cost. In this paper, we design two novel heuristic functions, which merges multiple bits in edge annotations when calculating the heuristic values, thus leading to a tighter estimation. The experimental results demonstrate that the proposed heuristic functions enable the A* algorithm to reduce the searching time by exploring less nodes, and consume less memory than previous methods.

Our improved hierarchical-clustering heuristic is an example of the disjoint pattern database technique developed in [Korf and Felner, 2002]. The optimal homology cycle has to satisfy the constraint that the sum of all bit-vector represen-



(a) (b) (c) (d) (e)

Figure 1: (a): interior of LV. (b): input CT image. (c): topological handles belong to the same homology class and its representative cycles. The yellow one is the shortest and best describes its geometry. (d): trabeculae segmentation without optimal cycles. (e): successfully captures the trabeculae with shortest homology cycles [Wu et al., 2017].

tations of the edges in the cycle satisfy a particular homology cycle class. The naive heuristic in [Wu *et al.*, 2017] relaxes the constraint so that each bit sums to its desired value independently. Our improved heuristic clusters the bits into exhaustive and exclusive groups so that each group of bits sums to desired values, thus producing a tighter lower bound than the naive heuristic. Also, the naive heuristic and the optimal homology cycle can be considered two extreme cases of our heuristic, with the group size equal to one or the total number of bits respectively.

The rest of this paper is organized as follows. Section 2 introduces the background theories in algebraic topology. Section 3 formulate the problem and explore its solutions via annotation and covering graph. Section 4 presents the search method and the proposed heuristic function. Section 5 provides the experimental results and discussions.

2 Background

In this section, we introduce the theoretical background of algebraic topology [Munkres, 1984; Edelsbrunner and Harer, 2010]. We are particularly interested in 1-dimensional topological structures, i.e., handles. A d-dimensional simplex or d-simplex σ, is the convex hull of $d+1$ affinely independent points. A vertex, edge, triangle and tetrahedron are 0-simplex, 1-simplex, 2-simplex and 3-simplex in topology respectively. A face of a d-simplex is the convex hull of a nonempty subset of its $d+1$ vertices. A tetrahedron has four 2-dimensional faces, corresponding to four triangles enclosing the tetrahedron. A triangle has three 1-dimensional faces, i.e., the three edges bounding it. The two end vertices of a edge are its faces. A simplicial complex $K = \{\sigma_1, \sigma_2, ..., \sigma_n\}$ is a collection of simplices satisfying two conditions: 1. any face of a simplex in K is also in K. 2. the intersection of any two simplices in K is either empty or their common face. The dimension of a simplicial complex is the maximum dimension of its elements.

A d-chain c is a formal sum of d-simplices in K, $c = \sum_{\sigma \in K} a_\sigma \sigma$, $a_\sigma \in \mathbb{Z}_2$. Note that $\mathbb{Z}_2$ is the binary field with only 0 and 1 as its elements and uses $\mathbb{Z}_2$ arithmetic as the addition rule (mod 2). A d-chain c is also a subset of the simplicial complex K, $C \subseteq K$. A d-chain can be represented by a n_d long binary vector, in which n_0 is the number of d-simplices in K. The i-th entry is 1 if and only if $\sigma_i \in c$. All d-chains form the group of d-chain, $\mathsf{C}_d(K)$, which is equivalent to a n_d-dimensional binary vector space.

The boundary of a d-simplex is the formal sum of its $(d-1)$-faces. The boundary of a d-chain, c, is the sum of boundaries of all d-simplices in c, $\partial_d(c) = \sum_{\sigma \in c} \partial_d(\sigma)$. When the chain is represented by a n_d-dimensional vector, the boundary operator is equivalent to a $n_{d-1} \times n_d$ binary matrix, whose columns are the boundaries of individual d-simplices. See Figure 2 for an example simplicial complex and its boundary matrices. The boundary of c is equals to multiplying the boundary matrix to c, $\partial_d c$. The group of d-boundaries is the image of the $(d+1)$-dimensional boundary matrix, $\mathsf{B}_d = \mathrm{im}(\partial_{d+1})$.

A d-cycle is a d-chain with zero boundary. The group of d-cycles is the kernel of boundary operator $\mathsf{Z}_d(K) = \ker(\partial_d)$. In fact, A d-boundary z is a d-cycle, formally, $\mathsf{B}_d(K) \subseteq \mathsf{Z}_d(K)$. A d-cycle z is a *non-boundary cycle* if $z \in \mathsf{Z}_d(K) - \mathsf{B}_d(K)$.

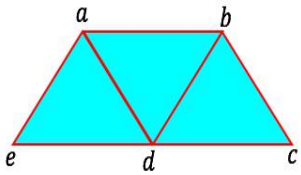
The group of all d-cycles are partitioned into different homology classes. A homology class is an equivalent set of cycles, whose difference is a boundary. Formally, given a cycle z_0 belonging to the class h, the class is $h = \{z \mid z = z_0 + \partial_{d+1} c, c \in \mathsf{C}_{d+1}(K)\}$. The homology class can also be a *coset* $[z_0] = z_0 + \mathsf{B}_d(K)$. Any two cycles in the same homology class are homologous. Any cycle in the homology class can be the *representative cycle*, z_0. When z_0 is a boundary, $[z_0]$ is the boundary group $\mathsf{B}_d(K)$. All the d-dimension homology classes form the d-dimension homology group $\mathsf{H}_d(K)$, which is equal to the quotient of $\mathsf{Z}_d(K)$ over $\mathsf{B}_d(K)$, formally, $\mathsf{H}_d(K) = \mathsf{Z}_d(K)/\mathsf{B}_d(K)$. As mentioned before, $\mathsf{C}_d(K)$ is isomorphic to a vector space $(Z_2)^{n_d}$, where n_d is the number of d-simplex in K. In fact, $\mathsf{Z}_d(K)$, $\mathsf{B}_d(K)$, and $\mathsf{H}_d(K)$ are all vector space $(Z_2)^{n_d}$. The dimension of the homology group $\mathsf{H}_d(K)$ is called *Betti number*, β_d. For convenient, we focus on 1-dimensional homology and denote $g = \beta_1$. A set of g cycles, representing a set of linear independent classes, is called a *homology cycle basis*.

3 Problem

We formulate the localization problem. Given a simplicial complex and a homology class $[z_0]$, the localization problem is to find the shortest cycle within the class $[z_0]$, formally,

$$\operatorname*{argmin}_{z \in [z_0]} length(z). \tag{3.1}$$

For convenience, we assume all edges have length 1. $length(z)$ is then the number of edges in a cycle z. We use



(a)

	ad	cd	de	bd	ae	ab	bc
d	1	1	1	1	0	0	0
a	1	0	0	0	1	1	0
c	0	1	0	0	0	0	1
e	0	0	1	0	1	0	0
b	0	0	0	1	0	1	1

(b)

	ade	abd	bcd
ad	1	1	0
cd	0	0	1
de	1	0	0
bd	0	1	1
ae	1	0	0
ab	0	1	0
bc	0	0	1

(c)

Figure 2: A simplicial complex K (a) and its 1-dimensional (b) and 2-dimensional (c) boundary matrices.

advanced techniques called the homology annotation to convert the localization problem into a shortest path problem.

3.1 Homology Annotations

Let g be the Betti number of dimension 1. A homology annotation is a mapping from edges to binary vectors with length g, $\mathcal{A}(e) \in \{0,1\}^g$, so that for any two cycles, z_1 and z_2, their annotations are identical if and only if z_1 and z_2 belong to the same homology class. The annotation of a path or a cycle is equal to the sum of the edge annotations, $\mathcal{A}(z) = \sum_{e \in z} \mathcal{A}(e)$. The sum is over $\mathbb{Z}_2$ field. The edge annotations can be used to identify whether two cycles belong to the same homology class. For example, Figure 3(a)(right) shows an annotation of the complex in Figure 3(a)(left). An edge with no numbers associated has annotation $(0,0)$. The cycle (e_1, e_2, e_5, e_7) and the cycle (e_1, e_6, e_7) belong to the same homology class, as their annotations are both $(0,1)$.

The annotation can be computed as follows [Busaryev *et al.*, 2012]: Firstly, a spanning tree was created from the input simplicial complex. Any of the rest of edges which do not belong to that spanning tree creates a cycle by connecting two nodes on the tree. Secondly, a homology cycle basis H was calculated from those created cycles. Finally, the annotation of any edge, e, which does not belong to that spanning tree is calculated as a binary vector x, from a linear function $Hx = z$, where H is the homology cycle basis, and z is a cycle composed by a path on the spanning tree and the edge e. Any edge belonging to the spanning tree has annotation zero. Figure 3(a) shows the construction of an annotation.

With the annotation, we may reformulate the localization problem as finding the shortest cycle with the same annotation as z_0:

$$\underset{\mathcal{A}(z)=\mathcal{A}(z_0)}{\operatorname{argmin}} \; length(z). \tag{3.2}$$

Intuitively, we are searching for the shortest cycle with a specific annotation. The annotation can be considered as an advanced constraint one has to satisfy.

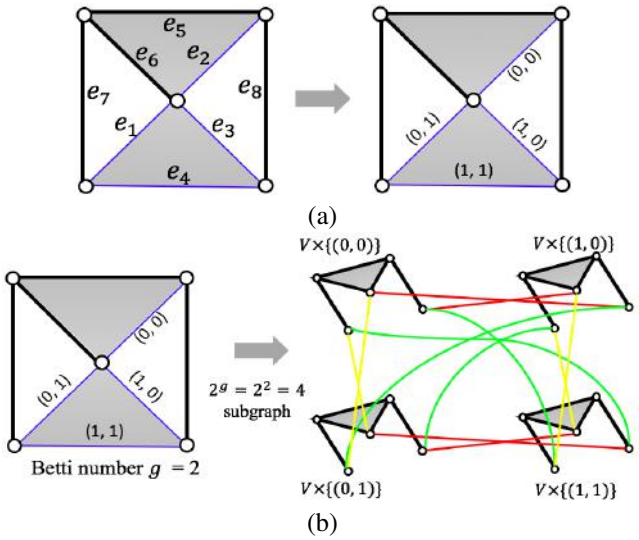


Figure 3: (a) Using a spanning tree $T = \{e_5, e_6, e_7, e_8\}$ (black edges), we compute the annotation. All blue edges, e_1, e_2, e_3, and e_4, have nonzero annotations. (b) A covering graph constructed from this annotation [Busaryev et al., 2012].

3.2 Covering Graph

To compute the shortest cycle within a given homology class, a covering graph with $N2^g$ many nodes, where N is the number of vertices of the simplicial complex. The covering graph is built on a computed annotation. See Figure 3(b) for an example. In the covering graph, 2^g subgraphs are created by copying the spanning tree for $2^g - 1$ times. In this covering graph, vertices in each subgraph corresponds to a vertex in the original complex associated with an annotation, e.g., $u \times (0,0)$. Edges with non-zero annotation connects vertices cross different subgraphs. An edge e connects two vertices $u \times \mathcal{A}_1$ and $v \times \mathcal{A}_2$ if and only if $\mathcal{A}_1 + \mathcal{A}(e) = \mathcal{A}_2$.

3.3 Dijkstra's Algorithm

Using the covering graph, we have an algorithm for the localization problem: for a vertex v, find a shortest path from $v \times (0, \ldots, 0)$ at the subgraph with zero annotation to $v \times \mathcal{A}(z_0)$ at the subgraph with annotation $\mathcal{A}(z_0)$. This path is essentially the shortest cycle with the desired annotation, $\mathcal{A}(z_0)$, that passes v. Repeating this procedure over all possible v's will find the shortest cycle with annotation $\mathcal{A}(z_0)$.

However, this straightforward algorithm is very expensive. In particular, assume the number of vertices in the original complex is N, the covering graph has size $2^g N$. Solving the shortest path problem for N times will need to run the Dijkstra's algorithm for N times over the covering graph, with a total complexity of $O(2^g N g \log N)$.

For the remainder of the paper, we solve this problem using A$*$ search with specially designed heuristic functions. With good heuristic functions, we do not need to exhaust the whole covering graph in order to find the shortest path as desired.

4 Heuristic Search

In this section, we present an A$*$ algorithm to solve the problem. In particular, we design heuristic functions based on the underlying topology and geometry.

In [Wu *et al.*, 2017], A* algorithm was employed to search on the covering graph to find the optimal cycle. In this work, we propose a novel heuristic function which is much tighter approximation of the real cost compared with the previous one. We first introduce the previous heuristic functions introduced in [Wu *et al.*, 2017]. Let $\widetilde{\mathcal{A}} = \mathcal{A}(z_0)$ be the desired annotation. We focus on the search problem where we have a fixed starting point $u = u' \times (0, \dots, 0)$ and end point $\omega = u' \times \widetilde{\mathcal{A}}$ in the covering graph, and need to find the shortest path between them. Denote by $p(u, \omega)$ the path from node u to node ω, and its cost is $c(u, \omega)$. Let $\mathcal{A}(p)$ represent the annotation for path p. Denote the heuristic estimation from ω to v as $h(\omega, v)$. Then the total estimated cost of the target path is $c(u, \omega) + h(\omega, v)$, which needs to be minimized.

Here $h(\omega, v)$ is the maximum of sub-heuristic functions $h_i(\omega, v)$, each of which is the estimated shortest distance between ω and v. Such estimation should satisfy the constraint that the i-th bit of the annotation $\mathcal{A}(p(\omega, v))$ and $\widetilde{\mathcal{A}} - \mathcal{A}(p(u, \omega)$ are identical. Formally,

$$h(\omega, v) = \max\{h_1(\omega, v), h_2(\omega, v), ..., h_g(\omega, v)\}, \quad (4.1)$$

$$h_i(\omega, v) = \min_{\alpha \in P(\omega, v),\ \mathcal{A}_i(\alpha) = \widetilde{\mathcal{A}}_i - \mathcal{A}_i(p(u,\omega))} c(\alpha), \quad (4.2)$$

where $P(\omega, v)$ is the set of paths between ω and v, $\mathcal{A}_i(\alpha)$ is the value of the i-th bit of annotation of path α, $p(u, \omega)$ is the existing optimal path from u to ω.

4.1 The Proposed Heuristic Function

In order to design a tighter heuristic approximation for A* algorithm, we propose to combine two or three annotation bits in a single heuristic function. Given that each annotation bit corresponds to a certain homology class, intuitively it should be more efficient for A* search employing bit combination when some homology classes/handles are tangled. The overall workflow of our method is as follows. We first analyze the input simplicial complex to obtain an optimal homology cycle basis, i.e., a homology cycle basis whole total length is the shortest [Dey *et al.*, 2010]. Then a hierarchical clustering is applied to cycles in this optimal basis. Note that each cycle in the basis corresponds to one annotation bit. By finding a hierarchical clustering of cycles based on their geometrical proximity, we decide which annotation bits should be grouped together. We use single linkage clustering technique to cluster the bases. We set the maximum number of bit combination to 3. The reason is that more bit combinations would lead to an increase of the expense for computing the heuristic values.

The steps of our method are as follows. First, the optimal cycle bases are calculated from the input complex (see Figure 4). Second, we compute the spatial centers of these basis cycles (see Figure 4(b)). Then a tree is created by iteratively merging two basis cycles whose centers are closest, until only one cycle is left. This process builds clusters and progressively creates a merging tree of cycles. Finally, the constructed tree is iteratively cut from the top level to form several sub-trees, until the number of leaves of all sub-trees is less than or equal to 3. The number of leaves in each remaining subtree decides the number of bits combined in the corresponding heuristic function.

After combining the single-bit heuristics based on the hierarchical clustering strategy. The set of g bits of the annotation vector is partitioned into $m \leq g$ disjoint sets, $s_1, \cdots, s_m$, so that $\{1, \dots, g\} = s_1 \cup s_2 \cup \cdots \cup s_m$. The heuristic function can be written as a maximum of m sub-heuristic functions, each of which is constrained by a set of bits, s_i. Formally, we have

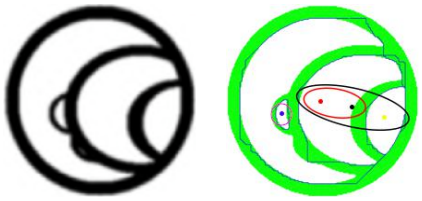
$$\begin{aligned} &h(\omega, v) = \max\{h_{s_1}(\omega, v), h_{s_2}(\omega, v), ..., h_{s_m}(\omega, v)\} \\ &h_{s_i}(\omega, v) = \\ &\min_{\alpha \in P(\omega,v), \mathcal{A}_j(\alpha) = \widetilde{\mathcal{A}}_j - \mathcal{A}_j(p(u,\omega)), \forall j \in s_i} c(\alpha). \end{aligned} \quad (4.3)$$

When s_i only contains a single bit, h_{s_i} is a single-bit function as in Equation (4.2). Others are combinations of k bits. In experiments, we set $k \leq 3$.

Those proposed heuristic functions are able to reduce the searching time, due to theirs intrinsic tighter estimation. In particular, for our problem, the time consumption of A* search mainly consists of three parts, including the time for creating the covering graph, calculating the heuristic value and searching. While the new heuristic function improves search efficiency (thus decreases search time), it is more time-consuming to compute. In particular, to compute a single-bit heuristic function, h_i, as in Equation (4.2) is equivalent to compute the shortest path in a graph of size $2N$. The graph is built similar to the covering graph. We use the original graph and a copy of it, corresponding to $\mathcal{A} = (\cdots, 0, \cdots)$ and $\mathcal{A} = (\cdots, 1, \cdots)$, respectively. The i-th bit with 0 or 1 is the bit of interest. To run Dijkstra's algorithm on this $2N$ size graph takes $O(N \log N)$ time. In the new proposed method, a particular heuristic function h_{s_i} may involve k different bits of interest. To compute it requires building a size $2^k N$ graph, consisting of 2^k different copies of the original graph, with different combinations of annotations on the k bits of interest. The computation is then $O(2^k N(k \log 2 + \log N))$. In other words, our method is able to achieve a better trade-off among different factors and therefore obtain better practical performance. Using smaller k will save heuristic function computation time. Using larger k will lead to tighter bound and thus save search time. We will evaluate the cases when k is upperbounded by 2 and 3, respectively.

The proof of admissibility is relatively straightforward; the problem is essentially a shortest path problem over the covering graph with constraints on the allowable paths (specified in terms of homology annotation). Each heuristic function is the shortest path distance to the destination with only partial constraints (a few bits of the annotation) satisfied. Each heuristic function h_{s_i} is naturally a lower-bound of the true shortest path distance. The admissibility will then follow. Also, because the shortest paths over the covering graph are essentially distances in an euclidean space, which automatically satisfy the triangular inequality, the proof of consistency immediately follows. Since the new heuristic combines several

bits for heuristic function, it is guaranteed to provide a tighter bound than the previous one (which takes the maximum of all single-bit heuristic functions). We will demonstrate the improvement in experiments.



(a) (b)

Figure 4: Illustration of clustering the cycles. (a) The input synthetic image with several cycles. (b) 4 cycles are identified and clustered from input image (the green color represents the topological space).

5 Experimental Results and Discussion

We evaluate the performance of the A* search with proposed heuristic functions. After calculating the persistence diagram, we compare different searching strategies to locate the optimal homology cycles on synthetic and real images, including (1) the NaiveTopo [Gao *et al.*, 2013], which produces non-optimal cycles; (2) OptTopoDij [Wu *et al.*, 2017], which employs Dijkstra's algorithm on searching; (3) OptTopoA* [Wu *et al.*, 2017], which uses A* algorithm with less tighter heuristic function; (4) the proposed methods, which use tighter heuristic functions.

We measure the performance of the following metrics: number of nodes expended (Nodes), and the total time and memory consumption (Mem). In particular, for the total (Total) time cost we further define the following fine-grained metrics: the time on constructing the covering graph (Graph), calculating the heuristic value (H Val) and searching (Search) with A* algorithm, where those terms in the parenthesis, are used from Table 1. Dij, A*, method (1), and method (2), represent the OptTopoDij, OptTopoA*, proposed 2-bit annotation combination in heuristic function, and proposed 3-bit annotation combination in heuristic function, respectively. We conduct the experiments on Intel i5-3427U CPU with 8GB memory.

5.1 Synthetic Experiments

We use randomly generated 2D synthetic images of size 300×300 (see Figure 5). The synthetic images have different numbers of cycles, from 3 (easy) to 7 (hard task). Experimental results are shown in Table 1. It can be observed that, the A* based searching methods, including OptTopoA*, and the proposed method (1) and (2), expand less number of nodes than OptTopoDij, and consume less memory for all the synthetic images.

For creating the covering graph, the time consumption of OptTopoDij is more than that of other methods, and is larger when the cycle number is larger (e.g., on harder task). The reason is that the size of covering graph created by OptTopoDij is exponential to the Betti number, g. However, A* based searching methods only create the covering graph with vertices belonging to the edge of specific homology class, and thus consume much less time.

Figure 5: Randomly generated synthetic 2-dimension images with 3,4,5,6,7 cycles.

OptTopoDij does not need to calculate heuristic values. The proposed method (2) shows better performance on calculating the heuristic. The reason is method (2) uses the heuristic function with 3-bit combination on annotation. The proposed method (1) and method (2) are only effective when the image has sufficiently complex topology. As a result, the advantage of time consumption of method (2) on calculating the heuristic values, varies depending on the images. The searching time consumption mainly depends on how many nodes explored. A* based methods always have better performance, especially method (2).

We can see the proposed heuristic function drastically improved the performance. For the total time consumption and node expended, the speedup of the proposed methods are 2.8, 1.8, 1.3, 1.5 and 1.6 for the 3-, 4-, 5-, 6- and 7-cycle images, respectively. The number of nodes are reduced by 5.6, 1.2, 2.1, 1.2 and 1.7 times, respectively. Except for the 3-cycle case, the improvement is consistent even for larger problems. Those results are the average performance. If we only focus on the most difficult handle (i.e., the one requiring the most time to optimize), the improvement is even more significant. For the 3-, 4-, 5-, 6- and 7-cycle images, the speedup on the most difficulty handle are 3.4, 2.6, 1.6, 4.7 and 3.8, respectively. The number of expanded nodes are reduced by 5.8, 1.3, 16.1, 7.6 and 2.4 times, respectively.

This selective way of combining bits is critical. We can show that it is much more effective than randomly combining bits. For synthetic input images, we evaluate the baseline of randomly combining bits. For the 3-cycle synthetic image, using the baseline of combining two random bits, the total run time and number of expanded nodes are 73.51 seconds and 47769 respectively. This baseline is much worse than the proposed methods, and are even worse than the previous A* heuristic method. For the 7-cycle synthetic image, the baseline of combining two random bits has 280.52 seconds total time and 141309 expanded notes. It is even worse than the previous A* heuristic method. Randomly combining 3 bits only performs worse.

5.2 Real World Data

We also validate the proposed method on trabeculae segmentation on cardiac CT images ($30 \times 17 \times 22$), as shown in Figure 6. A standard segmentation method (Figure 6(a)) fails to segment the trabeculae. Although the NaiveTopo (Figure 6(b)) is able to identify the trabeculae, it produces results with wiggling morphology due to the non-optimal homology cycles obtained. In contrast, heuristic search algorithms generate high quality segmentation (Figure 6(c)). Also, the proposed methods achieves better performance in total time

consumption and node expanded among different A* based searching methods, as shown in Table 1. Note that for the real world image, Dijkstra's algorithm cannot finish. Only heuristic search methods can find optimal cycles. The experimental results indicate that the A* search with proposed heuristic functions, consume less time and explore less nodes than the existing methods, while identify the same topological handles.

6 Conclusion

We propose a novel heuristic function for the computation of optimal representative cycles of topological handles. The method is validated on synthetic data and applied to real world cardiac image data. Topology-inspired methods can be useful in other applications where we need to reconstruct fine-scale structures with large variation in geometry and appearance. Examples include but are not limited to neurons [Uzunbas *et al.*, 2016; Li *et al.*, 2017] and vessels [Rudyanto *et al.*, 2014].

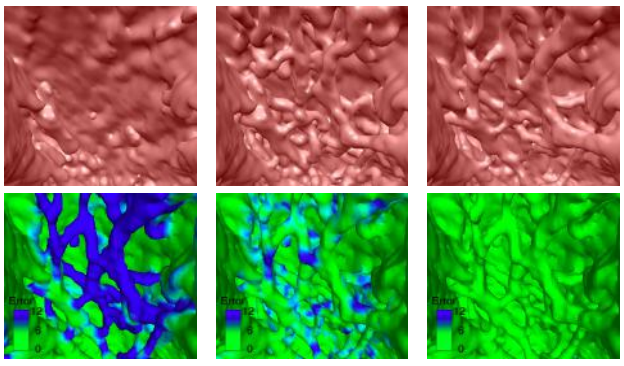
More advanced topological information such as persistent homology can be used in training classifiers [Chen *et al.*, 2019] and improving clustering [Ni *et al.*, 2017]. In these contexts, optimal persistent homology cycles [Wu *et al.*, 2017] add geometric information to topological descriptors and can improve analytical power.

Acknowledgments

We thank reviewers for their insightful comments and suggestions. We thank Dr. David Mount for suggesting us to look into this problem deeper. This work was partially supported by NSF IIS-1855759, CCF-1855760, CCF-1733843, IIS-1829560, CCF-1733798 and CCF-1740761.

References

[Boykov *et al.*, 2001] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1222–1239, 2001.



(a) (b) (c)

Figure 6: Segmentation results by (a) region competition, (b) Naive-Topo, (c) Result using heuristic search (OptTopoA* [Wu et al., 2017] or the proposed methods). The bottom row illustrate the corresponding segmentation error, which is rendered against ground truth.

3-cycle	Dij	A*	Method 1	Method 2
Mem(GB)	0.77	**0.67**	0.68	0.70
Nodes	1.52e6	23979	23979	**4320**
Graph(S)	83.52	2.03	**1.83**	2.23
H Val(S)	0	64.07	43.24	**21.04**
Search(S)	10.66	2.42	2.48	**0.89**
Total(S)	94.29	69.31	48.33	**25.03**
4-cycle	Dij	A*	Method 1	Method 2
Mem(GB)	1.59	**0.89**	0.90	0.93
Nodes	2.24e6	24162	24162	**19400**
Graph(S)	88.85	3.13	**2.89**	3.18
H Val(S)	0	74.48	55.76	**38.12**
Search(S)	21.24	2.13	2.13	**1.62**
Total(S)	110.32	80.74	61.77	**44.07**
5-cycle	Dij	A*	Method 1	Method 2
Mem(GB)	1.63	**0.61**	0.62	0.64
Nodes	2.48e6	22888	22888	**10932**
Graph(S)	90.85	2.13	**1.88**	2.13
H Val(S)	0	86.99	73.92	**66.74**
Search(S)	30.83	4.38	4.40	**4.00**
Total(S)	121.95	94.21	80.90	**73.73**
6-cycle	Dij	A*	Method 1	Method 2
Mem(GB)	15.44	**0.95**	**0.95**	0.97
Nodes	2.58e6	54792	54792	**46915**
Graph(S)	146.18	3.06	**2.47**	2.76
H Val(S)	0	152.63	127.23	**100.66**
Search(S)	57.57	2.59	2.63	**1.81**
Total(S)	204.60	159.17	133.18	**106.15**
7-cycle	Dij	A*	Method 1	Method 2
Mem(GB)	23.79	**1.45**	1.46	1.47
Nodes	3.20e6	81002	81002	**46315**
Graph(S)	217.13	7.15	**6.96**	7.46
H Val(S)	0	218.84	173.41	**131.73**
Search(S)	89.26	9.54	9.51	**5.76**
Total(S)	306.65	237.65	191.89	**147.16**
3D Image	Dij	A*	Method 1	Method 2
Mem(GB)	N/A	**20692**	20694	20725
Nodes	N/A	1521164	973784	**908513**
Graph(S)	N/A	84.69	**81.70**	85.10
H Val(S)	N/A	267.49	220.54	**215.78**
Search(S)	N/A	92.25	56.81	**53.55**
Total(S)	N/A	459.19	374.73	**372.62**

Table 1: Performance comparison between Dij, A* and the proposed methods on a 2D synthetic image with 3, 4, 5, 6 and 7 cycles, as well as real world 3D CT images.

[Busaryev *et al.*, 2012] Oleksiy Busaryev, Sergio Cabello, Chao Chen, Tamal K Dey, and Yusu Wang. Annotating simplices with a homology basis and its applications. In *Scandinavian workshop on algorithm theory*, pages 189–200. Springer, 2012.

[Chen and Freedman, 2010a] Chao Chen and Daniel Freedman. Hardness results for homology localization. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete algorithms*, pages 1594–1604. Society for Industrial and Applied Mathematics, 2010.

[Chen and Freedman, 2010b] Chao Chen and Daniel Freedman. Measuring and computing natural generators for homology groups. *Computational Geometry*, 43(2):169–181, 2010.

[Chen *et al.*, 2019] Chao Chen, Xiuyan Ni, Qinxun Bai, and Yusu Wang. A topological regularizer for classifiers via persistent homology. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2573–2582, 2019.

[Cootes *et al.*, 1995] Timothy F Cootes, Christopher J Taylor, David H Cooper, Jim Graham, et al. Active shape models-their training and application. *Computer vision and image understanding*, 61(1):38–59, 1995.

[Dey *et al.*, 2010] Tamal K Dey, Jian Sun, and Yusu Wang. Approximating loops in a shortest homology basis from point data. In *Proceedings of the twenty-sixth annual symposium on Computational geometry*, pages 166–175. ACM, 2010.

[Edelsbrunner and Harer, 2010] H. Edelsbrunner and J. Harer. *Computational topology: an introduction*. Amer Mathematical Society, 2010.

[Edwin P. Ewing, 2016] Jr. Edwin P. Ewing. Gross pathology of idiopathic cardiomyopathy — Wikipedia, the free encyclopedia, 2016. [Online; accessed 09-December-2016].

[Gao *et al.*, 2013] Mingchen Gao, Chao Chen, Shaoting Zhang, et al. Segmenting the papillary muscles and the trabeculae from high resolution cardiac ct through restoration of topological handles. In *International Conference on Information Processing in Medical Imaging*, pages 184–195. Springer, 2013.

[Korf and Felner, 2002] Richard E Korf and Ariel Felner. Disjoint pattern database heuristics. *Artificial intelligence*, 134(1-2):9–22, 2002.

[Li *et al.*, 2017] Yanjie Li, Dingkang Wang, Giorgio A Ascoli, Partha Mitra, and Yusu Wang. Metrics for comparing neuronal tree shapes based on persistent homology. *PloS one*, 12(8):e0182184, 2017.

[Munkres, 1984] James R Munkres. *Elements of algebraic topology*, volume 2. Addison-Wesley Menlo Park, 1984.

[Ni *et al.*, 2017] Xiuyan Ni, Novi Quadrianto, Yusu Wang, and Chao Chen. Composing tree graphical models with persistent homology features for clustering mixed-type data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2622–2631. JMLR. org, 2017.

[Rudyanto *et al.*, 2014] Rina D Rudyanto, Sjoerd Kerkstra, Eva M Van Rikxoort, et al. Comparing algorithms for automated vessel segmentation in computed tomography scans of the lung: the vessel12 study. *Medical image analysis*, 18(7):1217–1232, 2014.

[Uzunbas *et al.*, 2016] Mustafa Gokhan Uzunbas, Chao Chen, and Dimitris Metaxas. An efficient conditional random field approach for automatic and interactive neuron segmentation. *Medical image analysis*, 27:31–44, 2016.

[Wu *et al.*, 2017] Pengxiang Wu, Chao Chen, Yusu Wang, et al. Optimal topological cycles and their application in cardiac trabeculae restoration. In *International Conference on Information Processing in Medical Imaging*, pages 80–92. Springer, 2017.

[Zhu *et al.*, 1995] S.C. Zhu, T.S. Lee, and A.L. Yuille. Region competition: unifying snakes, region growing, energy/Bayes/MDL for multi-band image segmentation. In *ICCV*, pages 416 –423, June 1995.