Some Heuristics for the Homological Simplification Problem

Erin W. Chambers * Tao Ju † David Letscher ‡ Mao Li § Christopher Topp ¶ Yajie Yan ‖

Abstract

In this paper, we consider heuristic approaches for solving the homological simplification problem. While NP-Hard in general, we propose an algorithm that in practice significantly reduces topological noise from large datasets, such as those from medical or biological imaging.

1 Introduction

In this paper, we will consider the homological simplification problem. Introduced in [5], this asks: given a pair of simplicial complexes (C, N) where $C \subset N$, can the persistent homology group of this pair be realized as the homology of some intermediate complex? This problem is one way to approach the problem of topologically accurate simplification, where the goal is to take a "noisy" shape and simplify it to reach some desired topological structure. Such algorithms are useful in a wide range of applications, as any surface or region reconstruction algorithm on scanned input data is apt to contain errors, and hence a post processing phase to simplify it is necessary.

For filtrations of closed and orientable 2-manifolds, the homological simplification problem is solvable [11]; this work actually solves the more general problem of finding an ϵ-simplification in a filtration. However, such existence results do not hold in 3-manifolds since there are filtrations of manifolds that do not have ϵ-simplifications [11]. For the homological simplification problem, it is NP-Hard to determine if a simplification exists for 3-manifolds, even if the complex is embedded in $\mathbb{R}^3$ [5].

In this paper, we consider a 2-phase heuristic algorithm to simplify voxelized shapes. We first use a persistent homology-based algorithm to identify candidate simplifications. This approach is inspired by prior work to find "nice" generators for homology groups on surfaces [8], but we expand to find not just the generators of homology group, but also representatives in the larger space that kills those generators. Phase 2 is then a validation: given such a candidate simplification, we must check that it does result in a global simplification, since adding such things can introduce new topological features. We apply this algorithm to a number of types of input data, to assess how successful the heuristic is in practice. We find that our simplifications are able to remove over 99% of topological errors in several real-world data sets.

2 Related Work

2.1 Cubical complexes

A *cubical complex* is built from a collection of cells that are points, intervals, squares, cubes and higher dimensional analogs, where the intersection of any two cells is also a cell of the cubical complex. Formally, the cells are built from products of intervals, either the unit interval $[k, k+1]$ or a degenerate interval $[k, k]$, where k is an integer (so that our complex is aligned with an integer lattice); in d-dimensional space, a cube is a product of d elementary intervals. Given 2 cubes x and y, x is a face of y if $x \subseteq y$. A cubical complex of dimension d is a collection of cubes that is closed under taking faces such that the intersection of any two faces is a common subface. In this paper, we will use cubical complexes to represent the topology of our shapes.

2.2 Digital topology

In imaging and computer graphics, voxelizations are among the most common shape representations. With 3-dimensional Euclidean space divided using a cubical lattice, each individual cube is a single *voxel* and a voxelization of a shape is a set of these voxels. A common connectivity model of voxels is called 6-connectivity, which considers two voxels adjacent if they share a common 2-dimensional face [15]. This connectivity model results in a different topology than just taking the actual union of the voxels, which would typically connect anything that shares an edge or vertex as well. To be topologically consistent with the union of voxels, we will work in a dual complex where there is a vertex for each

*Department of Computer Science, Saint Louis University, echambe5@slu.edu.

†Department of Computer Science and Engineering, Washington University, St. Louis taoju@cse.wustl.edu.

‡Department of Computer Science and Engineering, Saint Louis University, letscher@slu.edu.

§Donald Danforth Plant Science Center, mli@danforthcenter.org .

¶Donald Danforth Plant Science Center, ctopp@danforthcenter.org .

‖Department of Computer Science and Engineering, Washington University, St. Louis yajieyan@wustl.edu.

voxel, and edge whenever two voxels share a common face, a square or 2-cell for any 4 voxels around an edge, and a 3-cell or voxel whenever there are 8 voxels surrounding a common vertex. Note that this is still a cubical complex, but it is not a pure cubical complex, since not every cell of dimension 1 or 2 belongs to 3-cell.

2.3 Homology

Homology groups and persistent homology on filtrations are commonly used tools to find topological features in spaces; due to space constraints, we refer the reader to recent books covering the topic for definitions of homology and persistent homology groups [9, 17]. For two spaces $C \subseteq N$, these persistent homology groups simply capture some of the topological features that are present in C and still remain in N. Most relevant to our setting, cubical persistent homology has been considered in some prior work [21], including optimized data structures to compute persistent homology groups of such complexes.

The p^{th} Betti number of a space X, $\beta_p(X)$ is defined to be the rank of the p^{th} homology groups. Given an inclusion map $f : C \to N$, we can define a persistent notion of Betti number, where the p^{th} Betti number of f is the rank of the induced map on homology: $\beta_p(C \to N) = rk(f^*(H_p(C)))$. Extending this to filtrations of more than 2 spaces precisely gives the notion of persistent homology groups (or their ranks).

2.4 Homological simplification

Consider two spaces $C \subset N \subset \mathbb{R}^3$. The homological simplification problem is generally phrased in terms of Betti numbers: we wish to find a space X such that the p-dimensional Betti number, $\beta_p(X)$ is equal to the Betti number of the inclusions $C \to N$, $\beta_p(C \to N)$. In three dimensions, this problem has been shown to be NP-Hard [5].

2.5 Topological repair

Various methods have been developed in different research communities for removing topological errors of surfaces in $\mathbb{R}^3$. In computer graphics, algorithms exist for modifying a given surface to either remove features smaller than a given size or achieve a specific topology (e.g., a single connected component with a prescribed genus) [22, 24, 6]. These methods make decisions of where and how to modify the surface solely based on the shape of the given surface, whereas homological simplification bases its decision on the persistence of topological features between two spaces. In medical image analysis, there has been active research on rectifying the topology of reconstructions of biological structures, such as the cortical surface in the human brain [19, 13, 18, 23]. Most of these methods are specialized for removing redundant handles and cannot deal with other types of topological noises such as disconnected components or cavities. Finally, in scientific visualization, a line of research aims at simplifying the topological structure (e.g., the Morse-Smale complex) of a scalar function [7, 11, 12, 20]. While these methods effectively remove topological noise on *all* level sets of the function, they are unnecessarily expensive if the goal is to fix the topology of one level set. In addition, methods designed to work on functions in $\mathbb{R}^3$ are limited to reducing the set of maxima or minima, and hence are less useful for removing topological handles on the level sets.

3 Heuristic Shape Simplification

Since we will use voxelized representatives of our input shapes, we will focus on the restriction of the homology simplification problem to voxelized shapes. We restate the problem as follows where we refer to C as the core, and N as the neighborhood, and we wish to find a space X which is somehow "in between" them with homology equal to the persistent homology of $C \to N$.

3.1 Voxelized homological simplification

Suppose $C \subset N \subset \mathbb{R}^3$ are voxel regions. Determine if there exists X such that $C \subset X \subset N$ and $\beta_p(X) = \beta_p(C \to N)$ for all p.

Proposition 1 *The voxelized homological simplification problem is NP-hard.*

Proof. [Proof sketch] We omit details due to space constraints, but just briefly note that the proof for simplicial complexes embedded in $\mathbb{R}^3$ is a reduction from 3SAT [5]. They construct gadgets that are easily modified to be built from voxels on an $O(m)$ by $O(n)$ grid, where n is the number of booleans variables and m is the number of clauses in the 3SAT instance. □

It is typically impractical to solve the homological simplification problem; however, we are concerned with simplifying a shape as much as possible. In particular, we will try to find a shape X where $C \subset X \subset N$ with each of the Betti numbers β_p as close as possible to the persistent Betti numbers $\beta_p(C \to N)$. If they are equal, we have a solution to the homological simplification problem. Even if we do not achieve maximal simplification, we will typically simplify our shape significantly.

Our simplification procedure will be a two phase process. It starts with the core C, which our algorithm will always include in the result, and then tries to expand to remove topological noise. First, a modified version of the standard persistence algorithm [10] is used to find

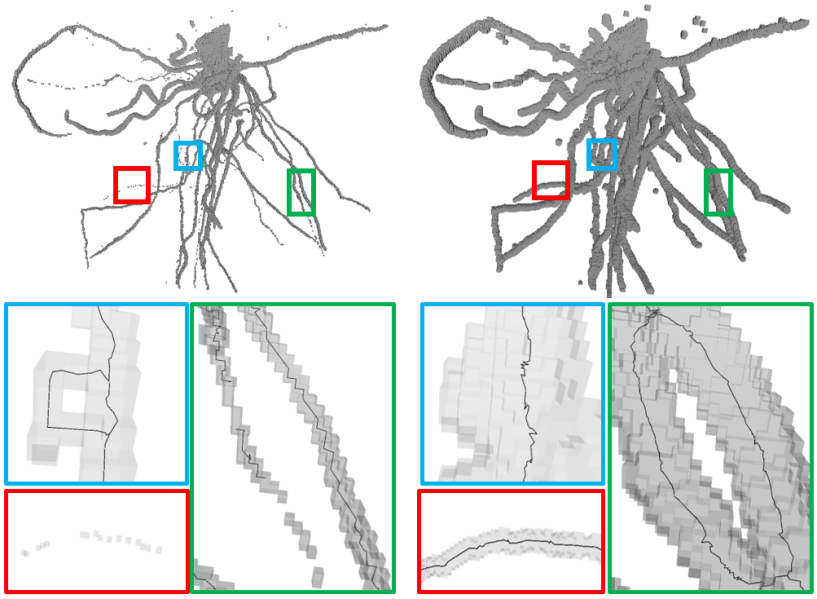

Figure 1: The core and neighborhood for the root system of corn. Note that in the inclusion of core into the neighborhood some loops are filled (blue inset), components connect (red inset) and new loops form (green inset). A solution to the homological simplification problem will accept the first two types of modifications and reject the creation of new loops and voids.

sets of voxels whose addition would remove topological features that are not in the neighborhood. However, these candidates might add new features that are not desired. The second phase examines the modified shape and checks if the persistence Betti numbers have been reduced. If so, the changes are kept. This process of candidate generation and validation is repeated until no more valid additions can be found. There are no guarantees that this will remove all of the undesired topological features; however, our experiments in Section 5 show that in real world examples the accuracy of the simplification is very high.

3.2 Types of topological errors

We now examine the types of topological errors in more detail, and our candidate simplifications. Consider $C \subset N \subset \mathbb{R}^3$ and the maps induced by inclusion $f_p^* : H_p(C) \to H_p(N)$ for $p = 0, \ldots, 3$. If every f_p^* is an injection then C is already a solution to the homological simplification problem. Assuming we do not have a solution, we know that not all the f_p^* are injective; in addition, we know that any extraneous topological noise must appear in the kernel of some f_p^*. In order to get a solution to the homological simplification problem, we therefore need to remove all such cycles.

To this end, consider a cycle $\alpha \neq 0$, where $f_p(\alpha) = 0$. In this 3-dimensional setting, we have several possibilities: α might connect two components, or it might fill a void or loop in the large space. To demonstrate what these look like, assume α is *irreducible*, that is it cannot be written as a non-trivial sum of other elements of the kernel. Depending on the dimension p, we can interpret the different types of noise we observe, and our algorithm will suggest different ways of removing that α in order to construct a better candidate solution (or near-solution, if we cannot generate an optimal one) to the homological simplification problem. We note that each of these cases can be observed in the root data in Figure 1, and hence we expect them all to exist in practice, depending on the data set. The cases to consider:

- $\alpha \in \ker f_0^*$: Here α is represented by two points lying in different components of C. These components can be connected by a path γ in N that connects the two points. In this case, γ is a one dimensional chain in N. By adding γ to C we obtain a new space $C \cup \gamma$ where the kernel of $H_0(C \cup \gamma) \to H_0(N)$ has rank one less than $H_0(C) \to H_0(N)$.
- $\alpha \in \ker f_1^*$: In this case α is a curve that follows a non-trivial topological feature in C, such as going around a handle, but which‘’ bounds some surface in N that kills that homological feature. If this surface is a disk D then $H_1(C \cup D) \to H_1(N)$ has rank one less than $H_1(C) \to H_1(N)$. In other words, we can add the disk D to remove the handle that is present in C. However, if the curve bounds a surface with genus, then if we were to add the surface in, a generator of the kernel is removed but two or more generators are added (since this surface has

its own handles). Hence the resulting object is not simpler in terms of homology, and we cannot use this chain to simplify the space.

- $\alpha \in \ker f_2^*$: Here α is a surface bounding a hollow region (a "void") in C, but which is not hollow in N. This void could be a simple ball, but could also contain topology. In either case, we can fill this in to simplify the topology. If the surface is a sphere, Alexander's theorem implies that it will bound a ball B in N [14]. In this case, it is easy to simplify since $H_2(C \cup B) \to H_2(N)$ has rank one less than $H_2(C) \to H_2(N)$. If the surface does not bound a ball, then the simplification may also simplify $\ker f_1$, as some loops will disappear when the void disappears.

The commonality of all of these cases is that our algorithm must find a $(k+1)$-dimensional structure in N whose boundary is in the kernel of f_k^* and whose addition simplifies the topological structure. Our techniques are based on this observation, repeatedly adding simplifications until no more can be found; we describe the generation of these candidates in the next subsection. Ideally, after adding these $(k+1)$-dimensional structures, we obtain a space X with $C \subset X \subset N$ and $H_k(C) \to H_k(X)$ is a surjection and $H_k(X) \to H_k(N)$ is an injection for all k; in this case, we would actually have a solution to the homological simplification problem. Our algorithm will only simplify the shape, although we have no guarantees of optimality or approximation ratio. In Section 5, we will discuss our implementation and show that it works well on several data sets.

3.3 Candidate generation

We now describe our algorithm to find generators in the kernel as well as to find candidate cycles that they bound in order to simplify the shape. Consider $C \subset X \subset N$ where X is initially equal to C but will be extended to remove topological features that are not shared by C and N. We will build a boundary matrix, ∂_p, for calculating the p-dimensional persistent homology of the filtration $C \subset X \subset N$. There is a column of ∂_p for each $(p+1)$-cell of X and a non-zero entry in that column for each p-cell that is a face of the $(p+1)$-cell. The rows and columns are ordered so that cells of C occur first, then cells of X and finally cells of N. The standard persistence algorithm [10] adds columns to ones to their right to obtain a canonical reduced form.

After these matrix reductions, the columns of ∂_p represent cycles involving the cells with non-zero entries. If that column corresponds to cells of N and the non-zero entries involved include d-cells of C, then this cycle, z, is a feature of C that does not persist in N and should

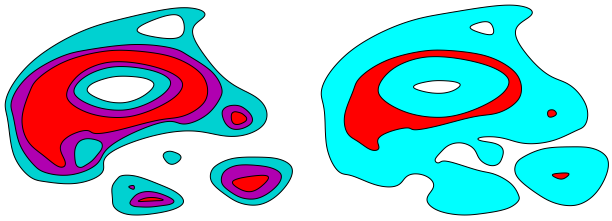

Figure 2: (top) Three intensity threshold input: $t-\epsilon, t, t+\epsilon$. (bottom) Corresponding core and neighborhood.

be removed. To remove this feature we need to find a $(d+1)$ chain, b, in X such that $\partial_p b = z$. We note that of course there are potentially many candidates for z; we find one by creating a copy of the identity matrix M with an entry for each column of ∂_p and performing the same column operations as we did in the reduction. An invariant of these column operations is that if v is the i-th column of M, then $\partial_p v$ is the i-th column of ∂_p. So we choose b to be the union of cells with non-zero entries in the column of M corresponding to the cycle z; recall that b is actually a chain in the dual cubical complex. We build a candidate simplification S which is the set of voxels that contain b. When S is added to X, the cycle z will become trivial; however, additional features could be added, so we cannot add S and guarantee reduction of Betti numbers.

We note that many improvements to the standard persistence algorithm have been made to make this calculation more efficient [2, 1]. These algorithms focus only on finding birth and death times, however, and discard any non-essential information in order to make calculations faster. In particular, they either discard or never compute information about generators in order to speed up the calculations; we need not only the generators, but also the cycles they bound, which we know of no way to track directly in the faster algorithms.

In addition, it is worth re-iterating that there are many possible cycles that could be added in order to kill generators in the kernel; we have merely computed the ones which can be calculated with a simple modification of the standard persistence algorithm. Surprisingly, in our experiments the algorithm nonetheless produces geometrically pleasing results, particularly for smaller topological errors which are common in real-world data.

3.4 Candidate validation

To determine if the addition of S will actually simplify the shape, recall that our algorithm tries to minimize the sum of Betti numbers $B(X) = \sum_{p=0}^{3} \beta_p(X)$, with the goal of reducing this complexity to be equal to $\sum_{p=0}^{3} \beta_p(C \to N)$. If S is set of voxels that form a candidate simplification, we will calculate $B(X \cup S)$. If $B(X \cup S) < B(X)$ and $\beta_p(X \cup S) \leq \beta_p(X)$ for all

p, then we will consider this a *valid simplification*. We cannot be sure that the addition of S has not created new topological feature, but our check does guarantee that more features are removed than were introduced.

Our algorithm We will repeat the process of finding candidate simplifications using the modified standard persistence algorithm and adding them one at a time if they are valid simplifications, until no more are possible.

4 Removing Topological Errors in 3D Imaging

In a 3D imaging technology such as CT or MRI, the output is a grid of voxels each with an intensity value. A typical representation would be a map $f : \{0, \ldots, k-1\}^3 \to \mathbb{R}$. Shapes of interest, in theory, can be extracted by selecting voxels in a given intensity range. A typical segmentation might try to extract all voxels with intensity over a specified threshold, e.g. $X = f^{-1}([t, \infty))$. However, there can be a variety of errors in this segmentation.

Define $N_\delta(X)$ as a neighborhood of the set X for positive δ, or as points with a $|\delta|$ neighborhood contained in the set X for negative δ. Formally:

$$N_\delta(X) = \begin{cases} \cup_{x \in X} B_\delta(x) & \delta \geq 0 \\ \{x \mid B_{-\delta(x)} \subset X\} & \delta < 0 \end{cases}$$

Then there is a natural two dimensional filtration of the region where $X_{t,\delta} = N_\delta\left(f^{-1}([t, \infty))\right)$. Here, t specifies the intensity threshold, and N_δ is in fact a morphological dilation when $\delta > 0$ or erosion when $\delta < 0$, using the ball of radius $|\delta|$ as the structuring element[16].

If t is a threshold for an initial segmentation $X = f^{-1}([t, \infty))$, we can give two parameters, ϵ and δ, representing noise levels on the threshold value and geometric scale, respectively. We will define the core $C = X_{t+\epsilon,-\delta}$ and the neighborhood $N = X_{t-\epsilon,\delta}$. This collapses the two dimensional filtration into a pair of spaces $C \subset N$. In this case the core C represents voxels that we want to include in the final shape, they all have neighborhoods meeting a higher intensity threshold. And the neighborhood N contains all the voxels near some voxel with a lower intensity threshold. See Figure 2 for an example.

A good solution to the homological simplification with this C and N would have features that do not appear or disappear with a small change in threshold, expanding or contraction of the shape, or a combination of these changes. In practice, a partial solution to the homological simplification problem might only remove some of the noise; in our experiments (described in the next section), this still results in an improvement over simple thresholding techniques. This improvement is quite evident in Figure 1, for example, where thresholding any any level (in the top pictures) leaves errors such as disconnected fragments and cycles; since this is a root and hence a simply connected space, these must be sampling errors and not actual features of the data. In the following section, we will discuss our observations of a reduction in more than 99% of the noise in several real world datasets.

5 Experimental Results

We experimented on three different collections of data: CT scans of corn root systems, synthetic root systems and brain volumes reconstructed from histological sections. The corn data was from a single variety of corn, with three different scans each viewed at two different resolutions. The synthetic root was designed to roughly resemble a root system and was studied at nine different resolutions. The brain scans were of the BigBrain dataset [4] downsampled at ten different resolutions. Figure 3 gives some information about the datasets, and our code is available [3]. The largest regions had close to 400 million voxels; this forced some additional techniques described below to handle the scale of the data sets. The largest core in these experiments has 2.5 million voxels and the largest neighborhood had over 4 million voxels. The most complicated example had approximately 8 thousand components, 17 thousand loops and one thousand voids. All but approximately 100 of those where noise features. In general, the complexity of the initial shapes was very high and there were very few features that were shared between the core and the neighborhood.

5.1 Practical concerns

The shapes that we have considered have regions as large as several hundred million voxels. As a result, we utilize a sparse representation for our shapes, so that the neighborhoods have about 6 million voxels at most; instead of storing actual values of the intensity, we have 3 hash sets that simply mark the core C, the current shape, and the neighborhood N. Even with this compression, there are speed issues. While there are faster persistent homology implementations [1, 2] that can handle inputs of this size, the standard persistence algorithm [10] has trouble on this size of inputs. However, we have no way to generate candidate simplifications using these faster algorithms, as discussed in Section 3.3. In order to deal with the size of the data, we did this calculation on windows of data at most 250^3 voxels in size, where the windows were chosen to overlap and cover the shape. This of course makes it likely we will miss some larger errors, but made the algorithm computationally feasible. We performed the calculations on the largest shapes in under five hours on a Linux desktop with a 2.20 GHz processor I5 processor and 64 GB RAM; see Figure 4.

	Region size	Core size	Neighborhood size	Complexity
Corn	$2.3 \times 10^6 - 1.5 \times 10^8$	$1.6 \times 10^3 - 2.4 \times 10^5$	$2.4 \times 10^3 - 4.6 \times 10^6$	$109 - 6,713$
Synthetic	$3.9 \times 10^6 - 3.8 \times 10^8$	$1.8 \times 10^3 - 1.7 \times 10^5$	$2.3 \times 10^3 - 2.7 \times 10^6$	$208 - 26,605$
Brain	$3.9 \times 10^5 - 1.7 \times 10^7$	$5.1 \times 10^4 - 2.5 \times 10^6$	$6.1 \times 10^5 - 3.6 \times 10^6$	$388 - 15,393$

Figure 3: Characteristics of the data analyzed, including the number of voxels in the region examined, core and neighborhood and the initial complexity of the shapes.

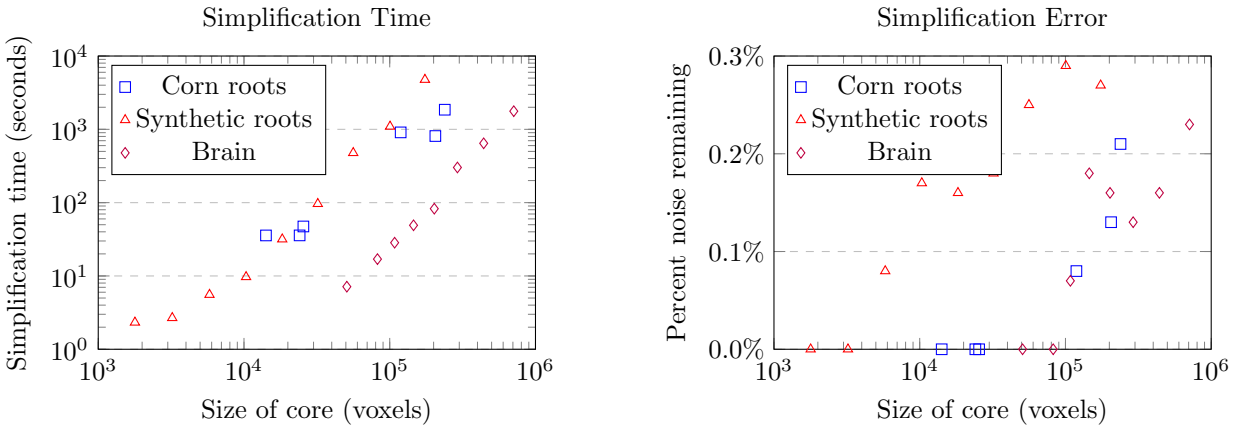

Figure 4: (a) Experimental runtime and (b) error rates on three datasets.

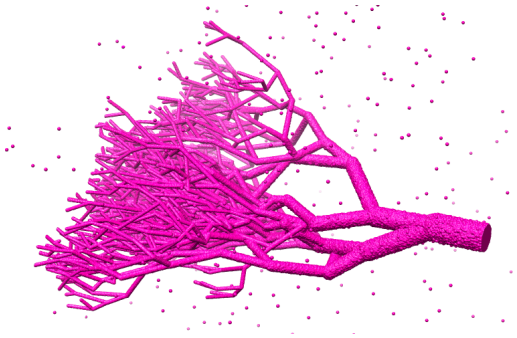

Figure 5: Synthetic root input shape with noise.

5.2 Accuracy

For all of the test shapes, over 99.7% of the topological errors were removed by our method; it is worth noting that much of this success is perhaps because many of the errors were quite small, consisting of just a few missing or extra voxels. In several cases, our algorithm was able to find solutions to the homological simplification problem. See Figure 4 for the error rates in the experiments. We note that it is computationally infeasible to determine in the other cases if solutions to the homological simplification problem exist.

6 Future Work

There are several natural directions to pursue next. First, we note that the algorithm described here picks some candidate to remove a particular error, but does not necessarily choose a geometrically nice generator. In fact, we have found examples where our algorithm adds a cycle that is obviously not ideal, or misses larger topological features due to the windows used in our algorithm. We plan to consider a more robust set of candidate simplifications that might be able to reduce the errors that are not repaired; at the same time, we would like to restrict to simplifications that have nice geometric properties as well as topological properties. In general, calculating "optimal" generators is impossible. However, for data sets such as the roots, we can take advantage of prior knowledge about the physical structure to prefer certain reconstructions, such as those that would reconnect two nearly crossing roots so as to maintain roughly the same local feature size along each. Second, on the more practical end, we would like to continue scaling the algorithms to larger datasets through optimizations and potential parallelization. Finally, more on the theoretical side, it is interesting to consider the notion of hardness of approximation or any approximation guarantees of heuristics such as our greedy approach.

Acknowledgements: This material is based upon work supported by the National Science Foundation under Award numbers: (PGRP) IOS-1638507, (EPSCoR) IIA-1355406, (AF) 1614562, and the linked collaborative awards (ABI) 1759807, 1759836 and 1759796.

References

[1] DIPHA. https://github.com/DIPHA/dipha.

[2] GUHDI c++ library. http://gudhi.gforge.inria.fr/.

[3] Voxelized homological simplification implementation. http://git.cs.slu.edu/public-repositories/shape-simplification-software.

[4] Katrin Amunts, Alan Evans, and Karl Zilles. Big-brain dataset.

[5] Dominique Attali, Ulrich Bauer, Olivier Devillers, Marc Glisse, and André Lieutier. Homological reconstruction and simplification in r3. In *Proceedings of the Twenty-ninth Annual Symposium on Computational Geometry*, SoCG '13, pages 117–126, New York, NY, USA, 2013. ACM.

[6] Marco Attene, Marcel Campen, and Leif Kobbelt. Polygon mesh repairing: An application perspective. *ACM Computing Surveys (CSUR)*, 45(2):15, 2013.

[7] P-T Bremer, Bernd Hamann, Herbert Edelsbrunner, and Valerio Pascucci. A topological hierarchy for functions on triangulated surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(4):385–396, 2004.

[8] T. K. Dey, K. Li, and J. Sun. On computing handle and tunnel loops. In *Cyberworlds, 2007. CW '07. International Conference on*, pages 357–366, Oct 2007.

[9] Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. AMS Press, 2009.

[10] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 454–463. IEEE, 2000.

[11] Herbert Edelsbrunner, Dmitriy Morozov, and Valerio Pascucci. Persistence-sensitive simplification functions on 2-manifolds. In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 127–134. ACM, 2006.

[12] David Günther, Alec Jacobson, Jan Reininghaus, Hans-Peter Seidel, Olga Sorkine-Hornung, and Tino Weinkauf. Fast and memory-efficienty topological denoising of 2d and 3d scalar fields. *IEEE transactions on visualization and computer graphics*, 20(12):2585–2594, 2014.

[13] Xiao Han, Chenyang Xu, Ulisses Braga-Neto, and Jerry L Prince. Topology correction in brain cortex segmentation using a multiscale, graph-based algorithm. *IEEE Transactions on Medical Imaging*, 21(2):109–121, 2002.

[14] John Hempel. *3-manifolds*, volume 349. American Mathematical Soc., 2004.

[15] Reinhard Klette and Azriel Rosenfeld. *Digital geometry: Geometric methods for digital picture analysis*. Elsevier, 2004.

[16] Laurent Najman and Hugues Talbot. *Mathematical morphology: from theory to applications*. John Wiley & Sons, 2013.

[17] Steve Y. Oudot. *Persistence Theory: From Quiver Representations to Data Analysis*, volume 209 of *Mathematical Surveys and Monographs*. American Mathematical Society, 2015.

[18] Florent Ségonne, Jenni Pacheco, and Bruce Fischl. Geometrically accurate topology-correction of cortical surfaces using nonseparating loops. *IEEE transactions on medical imaging*, 26(4):518–529, 2007.

[19] David W Shattuck and Richard M Leahy. Automated graph-based analysis and correction of cortical volume topology. *IEEE transactions on medical imaging*, 20(11):1167–1177, 2001.

[20] Maxime Soler, Melanie Plainchault, Bruno Conche, and Julien Tierny. Topologically controlled lossy compression. *arXiv preprint arXiv:1802.02731*, 2018.

[21] Hubert Wagner, Chao Chen, and Erald Vuçini. *Efficient Computation of Persistent Homology for Cubical Data*, pages 91–106. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[22] Zoë Wood, Hugues Hoppe, Mathieu Desbrun, and Peter Schröder. Removing excess topology from isosurfaces. *ACM Transactions on Graphics (TOG)*, 23(2):190–208, 2004.

[23] Rachel Aine Yotter, Robert Dahnke, Paul M Thompson, and Christian Gaser. Topological correction of brain surface meshes using spherical harmonics. *Human brain mapping*, 32(7):1109–1124, 2011.

[24] Qian-Yi Zhou, Tao Ju, and Shi-Min Hu. Topology repair of solid models using skeletons. *IEEE Transactions on Visualization and Computer Graphics*, 13(4), 2007.