Washington University in St. Louis
Washington University Open Scholarship

McKelvey School of Engineering Theses & Dissertations

McKelvey School of Engineering

Summer 8-15-2022

Geometric Algorithms for Modeling Plant Roots from Images

Dan Zeng
Washington University in St. Louis

Follow this and additional works at: https://openscholarship.wustl.edu/eng_etds

 Part of the Computer Sciences Commons

Recommended Citation

Zeng, Dan, "Geometric Algorithms for Modeling Plant Roots from Images" (2022). *McKelvey School of Engineering Theses & Dissertations*. 813.
https://openscholarship.wustl.edu/eng_etds/813

This Dissertation is brought to you for free and open access by the McKelvey School of Engineering at Washington University Open Scholarship. It has been accepted for inclusion in McKelvey School of Engineering Theses & Dissertations by an authorized administrator of Washington University Open Scholarship. For more information, please contact digital@wumail.wustl.edu.

WASHINGTON UNIVERSITY IN ST. LOUIS

Mckelvey School of Engineering
Department of Computer Science and Engineering

Dissertation Examination Committee:
Tao Ju, Chair
Jeremy Buhler
Ayan Chakrabarti
Ulugbek Kamilov
Alvitta Ottley
Christopher Topp

Geometric Algorithms for Modeling Plant Roots from Images

by
Dan Zeng

A dissertation presented to
the McKelvey School of Engineering
of Washington University
in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

August 2022
St. Louis, Missouri

© 2022, Dan Zeng

Table of Contents

List of Figures

List of Tables

Acknowledgments

My PhD has been a life-changing experience. As such, I have many people to thank for their support and guidance.

I would like to first thank my doctoral advisor, Professor Tao Ju, for your incredible patience and communication in mentoring, career guidance, and helpful expertise in formulating and executing my research. Your judicious feedback pushed me to work at a higher level and persist through challenging problems.

I would like to acknowledge my colleagues at the Donald Danforth Plant Science Center for their fruitful collaboration. I thank Dr. Christopher Topp for broadening my intellectual horizons to the world of plant phenotyping, giving me opportunities to further my research, and providing me with career guidance. I would also like to thank Dr. Mao Li, Dr. Mon-Ray Shao, Dr. Ni Jiang, and Dr. Elizabeth Kellogg for their expertise. You have been my role models for effective communication across disciplines while we worked on several projects together. I thank members of Chris Topp's lab for offering me significant help by way of knowledge and both mental and physical labor in data collection, both during and after my internship with the lab in 2018: Tim Parker, Keith Duncan, August (Gus) Thies, Dhineshkumar Thiruppathi, Elisa Morales, Mitchell Sellers, Tiffany Hopkins, Shayla Gunn, Alexander Liu, Marcus Griffiths, Nate Ellis, Zhengbin Liu, Adam Bray, and Eric Floro.

Figure 1: "Teamwork is the root of growth"

I would like to acknowledge my colleagues at Saint Louis University, Erin Chambers, David Letscher, and Hannah Schreiber, for their guidance in formulating my research, tutoring me in the field of computational geometry, and offering me useful career advice.

I would like to thank my past and present labmates for their mentorship and friendship, including Yajie Yan, Hang Dou, Zhiyang Huang, Xingyi Du, Gustavo Gratacos, Yiwen Ju, Yishan Zheng, Wenzhen Zhu, Chunyuan Li, Mahlon Farbanish, Jade Kandel, Yue Wu,

Huayi Zeng, Hang Yan, and Chen Liu. From lab dinners to snowboarding, you all have made the past five years such an enjoyable experience, and are the reason why I will remember these as the "good old days".

I thank Washington University in St. Louis Division of Biology and Biomedical Sciences for providing me with an Imaging Sciences Fellowship, which funded my research from 2019 through 2021.

I thank my family for being my greatest supporters through thick and thin. We are one.

Dan Zeng

Washington University in St. Louis

August 2022

This thesis is dedicated to:

My parents, who from humble beginnings came to America seeking a better life. They are my role models of perseverance.

My brother and sister, who have been my lifelong companions.

My relatives, many of whom live an agricultural life. I thought of them when considering the broader impact of our work.

My PhD advisor, Dr. Tao Ju, who provided me with a career direction, valuable skills, and lifelong lessons.

My labmates, colleagues, and friends, who have enriched my life in immeasurable ways.

My teachers and mentors from preschool through PhD, who have taught me that learning is lifelong.

My birth city of St.Louis, where I have called home up to the present. Wherever I may go, here are my roots.

ABSTRACT OF THE DISSERTATION

Geometric Algorithms for Modeling Plant Roots from Images

by

Dan Zeng

Doctor of Philosophy in Computer Science

Washington University in St. Louis, 2022

Professor Tao Ju, Chair

Roots, considered as the "hidden half of the plant", are essential to a plant's health and productivity. Understanding root architecture has the potential to enhance efforts towards improving crop yield. In this dissertation we develop geometric approaches to non-destructively characterize the full architecture of the root system from 3D imaging while making computational advances in topological optimization. First, we develop a global optimization algorithm to remove topological noise, with applications in both root imaging and computer graphics. Second, we use our topology simplification algorithm, other methods from computer graphics, and customized algorithms to develop a high-throughput pipeline for computing hierarchy and fine-grained architectural traits from 3D imaging of maize roots. Finally, we develop an algorithm for consistently simplifying the topology of nested shapes, with a motivating application in temporal root system analysis. Along the way, we contribute to the computer graphics community a pair of topological simplification algorithms both for repairing a single 3D shape and for repairing a sequence of nested shapes.

Chapter 1: Introduction

1.1 Roots

1.1.1 Motivation

Figure 1.1: Some motivating applications of plant phenotyping. (A) Yuan Longping, the father of hybrid rice. (B) Growing drought-resistant sorghum varieties in sub-saharan Africa (C) low-input agriculture minimizes use of pesticides and fertilizers.

Roots are the primary means by which the plant absorbs water and nutrients. Increasing the capacity of roots to obtain resources from the soil is vital to improve crop yields and minimizing farmer's use of fertilizers, which consume large amounts of energy and cause environmental pollution [1]. As the climate warms, the role of roots in crop productivity has become especially important in arid regions such as sub-Saharan Africa, where depletion of soil fertility has contributed to food insecurity in the region [2]. Besides absorbing resources from the soil, roots also form a symbiotic relationship with the soil by releasing products of photosynthesis (e.g. sugars, amino acids, and proteins) to surrounding bacteria and fungus, which in turn control the proliferation of harmful pathogens, thus producing a healthy environment around the roots. Finally, roots provide anchorage to the plant and reduce soil erosion by holding the soil together [3]. Roots are of particular interest to those who study cereal crops, such as maize, wheat, rice, sorghum, and millet, which account for more than 50 percent of the world's daily caloric intake [4]. Understanding the role

that roots play in improving the production and survivability of these crops will be one of the keys to the future of food security as the world's population grows and the climate changes.

1.1.2 Root system architecture

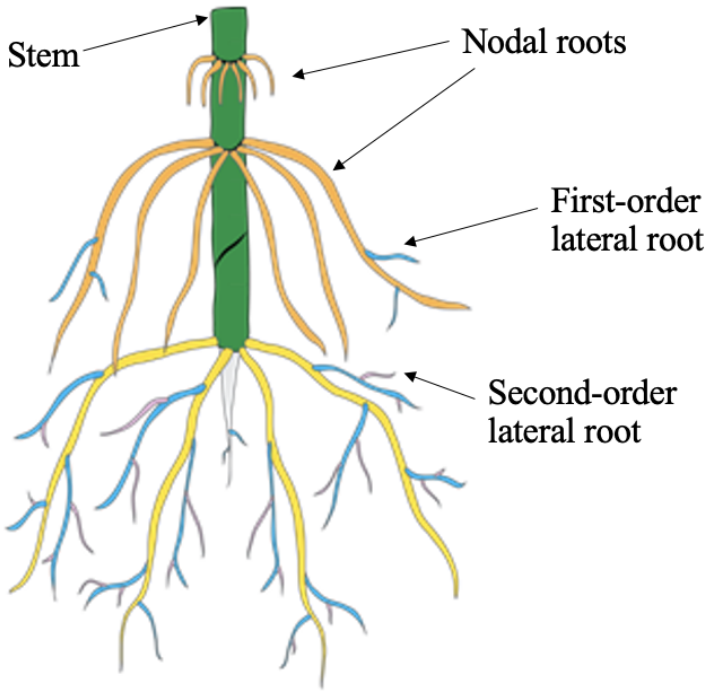

Figure 1.2: The root system architecture of fibrous root systems such as that of Maize.

These functions are closely related with root system architecture (RSA) [5, 6, 7], which both describes the geometry of individual roots and their hierarchical relationships. Each root is typically classified as one of several types depending on hierarchy level (Fig. 1.2). The hierarchy begins at the stem, which forms the main axis of the root system in a direction roughly perpendicular to the soil. Nodal roots emerge from the stem. For some species such as Maize and Rice, the starting points of the nodal roots will form whorls which are groups of roots emanating from roughly the same point along the stem. First-order lateral roots emerge from nodal roots, second-order lateral roots emerge from first-

order ones, and so forth. Root architectures can be further classified into tap root systems and fibrous root systems. Tap root systems have a single main root that grows down, while fibrous root systems form a dense network of roots that is closer to the soil surface. Quantitatively describing RSA enables efforts to discover the genetic basis that controls root traits. Knowing which traits are favourable for field growth, and breeding crops with those traits, has the potential to improve crop yield while minimizing adverse environmental effects [7].

1.2 Capturing root architecture

1.2.1 Traditional methods

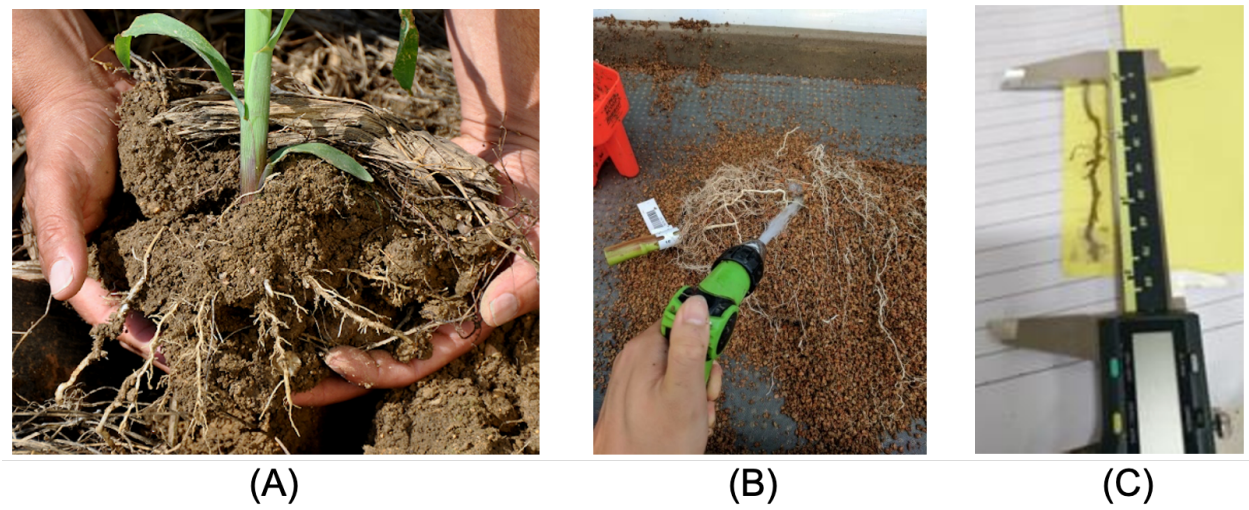

Figure 1.3: Traditional methods for capturing root architecture often involve a combination of (A) Digging up the roots (B) Washing the roots (picture from Topp Lab internship, 2018) , and (C) dissecting and measuring individual roots using devices such as calipers. Such methods are *destructive*.

In contrast to the visible half of the plant above the soil, roots are considered as the "hidden half" of the plane due to their poor accessibility, which make it much more challenging to investigate RSA. Traditional methods for analyzing root structure either involve pick-

ing the root apart or only provide a very limited 2D view of the plant (Fig. 1.3). In Shovelomics, the root is excavated from the soil and traits such as the numbers of roots, angles, densities, and lengths are measured using devices such as calipers, protractors, and scales [8]. Often times roots are clipped off in order to measure them individually, both altering the geometry and killing the plant, making it impossible to measure traits over time. Measurements of lateral roots by hand are also very challenging due to their miniscuility. Lastly, roots may be accidentally lost when the root system is washed after being dug up from the soil, which affect not only root counts, but also average statistics for other traits.

1.2.2 2D imaging

In contrast to destructive methods, non-destructive methods for investigating root architecture involve either 2D or 3D imaging. The root could be placed on a flatbed scanner, providing a compressed 2D view of the root's structure. Due to their simplicity, 2D photographs taken from the side of the plant are also used for large-scale studies [9]. In a Rhizotron setup, a laboratory is constructed underground, with viewing windows to the soil compositions coming from all sides of a central corridor, to study the root's interactions with the soil, other plants, and animals. These allow the root to be grown and studied in its natural soil environment. However, all of these mentioned methods fail to fully capture the full 3D geometry of the root architecture due to occlusions between roots when viewed from a single 2D perspective [10].

1.2.3 3D imaging

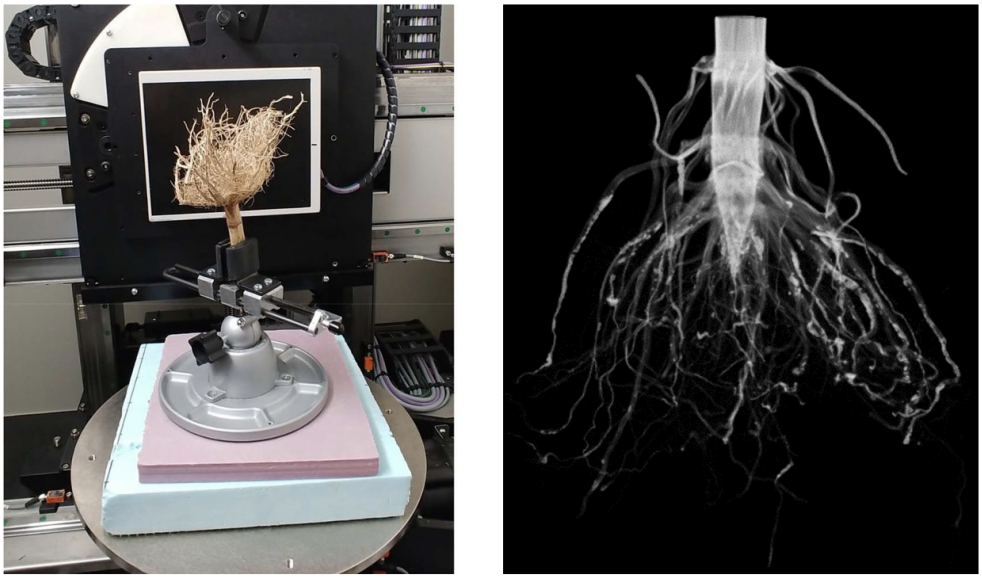

Figure 1.4: X-ray CT imaging setup used throughout this dissertation. The excavated root is placed on a turntable, with a radiograph being taken every few degrees of rotation (left). Customized software converts the projects into a 3D grayscale image (right). Picture provided courtesy of the Topp Lab.

More recently, 3D imaging techniques including CT, MRI, and optical imaging have been developed [11, 12, 13], which can address the limited views provided by 2D imaging by non-destructively capturing the entirety of 3D root structures in their natural environments. Among these techniques, X-ray CT has been used to study both the visible and the hidden half of the plant with high resolution and throughput [14, 15]. In an X-ray setup (Fig. 1.4), the pot with the root inside is placed on a turntable, which is rotated 360 degrees while X-ray source shoots beams at it which are received by a flat panel detector on the other side. By illuminating the "hidden" half of the plant, 3D imaging methods including CT have paved the way for recent efforts towards automatic quantification of root system architecture [16, 17].

1.3 Challenges in root image analysis

While high resolution 3D root images can now be produced with high throughput, there are still many computational challenges before any useful architectural traits can be obtained from those images. These include the tasks of segmenting the root from the rest of the image and computing the architecture from that segmentation.

1.3.1 Root segmentation

Reconstructing a shape from the image which accurately captures its architecture, through image segmentation, is a challenging task for several reasons. Due to the insufficient contrasts between the roots and their surroundings (e.g. soil or pebbles), the resulting segmentation may contain parts which are not a part of the root. Just as important, a host of topological issues can be present in the segmentation due to resolution and power limits of the imaging. Individual roots may not be adequately separated from each other in the segmentation, resulting in topological handles. Given that the desired hierarchy is inherently an acyclic construct, these handles must be removed correctly according to the known semantics of the root architecture. If the imaging resolution is not high enough, the images also may fail to fully capture thinner roots, which can become disconnected in the segmentation, hence affecting both the root counts and lengths. Lastly, Limits in the power of the imaging may result in the interior of the root to not be fully captured, resulting in topological voids. As we will see in second part of the dissertation, these voids can prevent the stem from being fully captured, resulting in cascading errors in the hierarchy computation. Most segmentation techniques which have been used for roots have involved either global or local thresholding, which work well when there is a clear contrast between the roots and its surroundings, but struggle to separate the root from the soil [18, 19]. Other methods follow a region-growing approach [20], or attempt to track the shape of the root one image slice at a time [21, 22]. More recently, deep learning techniques have also been

applied to label each voxel as either root or soil [23]. In situations where the root system is not complex, semi-automatic segmentation can also be performed using image processing software such as ImageJ [24] and Volume Graphics Studio Max [25, 26]. Few of these methods address topological issues, and the few that do only guarantee that the root is a single connected component [27].

1.3.2 Computing root traits

Even after image segmentation, further steps are needed to compute a full description of the architecture. While global traits such as volume, convex hull volume, surface area, total length can be easily measured from the segmentation, other more fine-grained traits, including branching hierarchy, root lengths, angles, and individual root growth over time, require a much more involved analysis. Because of this, most studies and analysis tools have only outputted global traits [27, 22, 28, 24], being limited by the presence of topological features, and by the lack of a method to compute the hierarchy. Only the method of [29] attempts to compute the hierarchy. However, the method depends on the quality of the image segmentation, and by itself can not compute traits from a grayscale image or resolve topological issues. In the second part of the proposal, we provide a detailed comparison of our method with DynamicRoots.

1.3.3 Capturing root growth

Figure 1.5: A motivating application of the final aim of this dissertation seeks to topologically simplify reconstructions of root systems captured across multiple time points.

As the imaging throughput increases, it has become more possible to use 3D imaging to capture the growth of the root system across multiple time points. How a root dynamically interacts with its environment plays a critical role in its absorption of water and nutrients, which is in turn determined by the plant's genetics [30, 31, 32]. Given limited resources from the soil, a plant must decide how to allocate its growth in terms of the expansion of existing roots, the creation of new ones, and the directions of these growth patterns. [33] Understanding the genetic basis behind root growth would support efforts at improving crop survivability in the context of sustainable agriculture development and climate change [34, 35]. Both 2D and 3D phenotyping platforms have been used to mathematically model root growth, but just like in the static case, 3D platforms have seen an increase in popularity due to the avoiding of occlusion issues. Aside from the mentioned challenges for obtaining fine-grained architecture for the static case, capturing dynamic growth is even more challenging due to the consistency of the shape and architecture which must be maintained at different time points. The root segmentation at a previous time point must be a strict subset of the root at a later time. Also, roots which are classified to be one hierarchy level at an earlier time point must be at the same hierarchy level at

a later time. There are very few works in either the computer graphics or root analysis community for addressing the first issue, and only the work of [29] attempts to address the second issue, with limited success.

1.4 Applications outside roots: computer graphics and imaging

While the original motivation for the work in this dissertation was for root architecture analysis, the underlying topological issues which are present in root imaging are also prevalent in other fields.

1.4.1 Topological Simplification

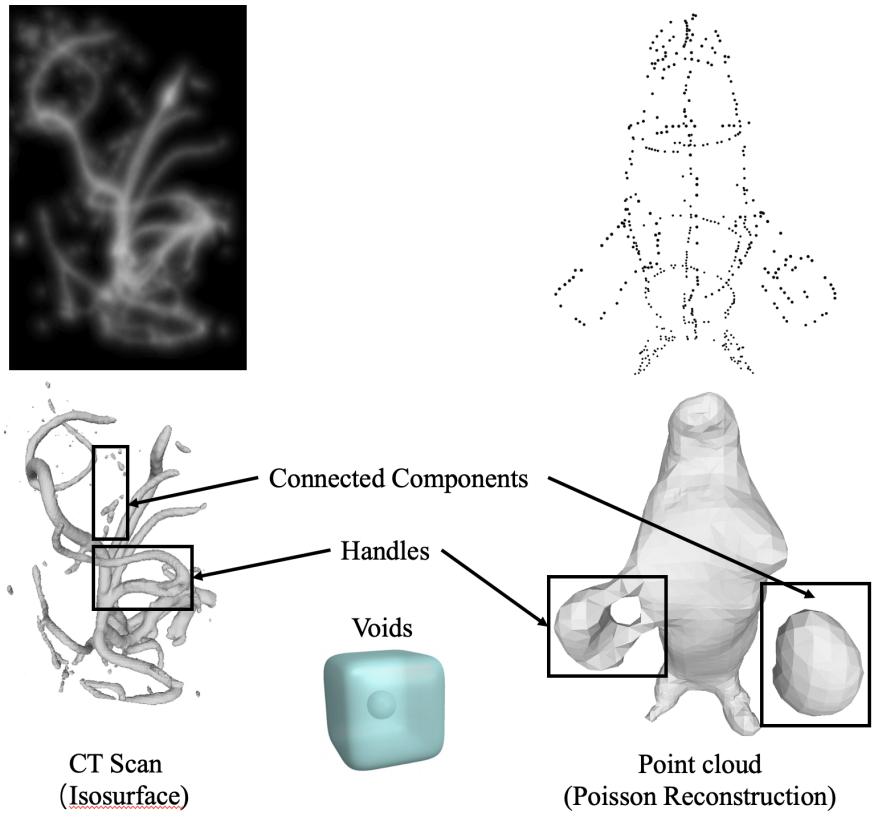

Figure 1.6: Reconstructing shapes from raw data is a common task within computer graphics. Whether its a grayscale image such as a CT scan (left), or a point cloud (right), the surfaces produced by these reconstructions often have an excessive number of topological features in the form of connected components, handles, and cavities.

Many applications within computer graphics and biomedical imaging require reconstructing a shape from raw data, such as a CT scan or a point cloud. Examples of such applications include brain MRI segmentation, indoor reconstructions for virtual reality, and 3D scene reconstruction from laser scans for autonomous driving. The shapes, which can be represented as a binary volume or a surface, often contain excessive topological features (components, handles, and cavities), some of which are intended, and others which are noisy artifacts. Topological noise can be detrimental to many geometry processing tasks, including mesh parameterization, shape analysis (as in the roots application), and physical simulation.

The problem of *topological simplification* thus aims to reduce the number of connected components, handles, and cavities of a shape while minimizing the change to the geometry. Prior to this dissertation, existing methods for this task limited their modifications to be only cutting, or only filling, or chose a greedy approach. These methods often resulted in excessive changes to the shape. In contrast, the topological simplification algorithm presented in chapter 2 of this thesis uses a global optimization approach to minimize the number of topological features while also minimizing the change to the geometry. The algorithm demonstrates robust results on both complex root images, other biomedical images, and 3D graphics models.

1.4.2 Topological Simplification of Nested Shapes

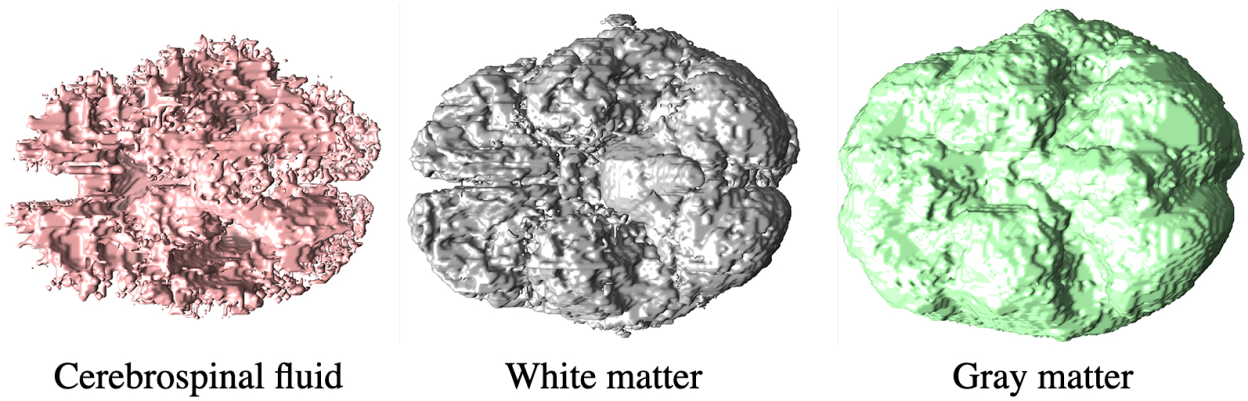

Figure 1.7: Outside of growing roots, the different layers of the brain as captured here by a CT scan also represent a sequence of nested shapes, each with its own potential topological issues.

Just like how topological issues in growing roots need to be repaired consistently, many other applications also require reconstructing a series of *nested* shapes with minimal topological features across all of the shapes. Example applications include different iso-surfaces of a scalar field (e.g. X-ray CT scan of a bone or other biomedical image), or shapes which are inherently layered (e.g. Fig. 1.7 gray matter, white matter, cerebrospinal fluid of a

brain). Ideally, each shape in the nested sequence should be *simply connected* (with a single connected component and no cycles or cavities), but minimal changes to the geometry should be used to achieve such a shape.

In chapter 4, we introduce an algorithm for topological simplification of nested shapes. While prior works either could not guarantee consistency or repair all three types of topological issues, the algorithm presented in that chapter consistently repairs a sequence of nested shapes while removing all three types of topological features in a geometrically minimal fashion.

1.4.3 Goals

This dissertation seeks to (1) fill the need of computational methods for root architectural analysis from 3D imaging (2) develop approaches for topological simplification for both a single shape and a sequence of nested shapes. We first address topological issues in the imaging, then develop a full image-to-analysis pipeline, and lastly extend this pipeline for time-series imaging. In the process, we develop topological simplification algorithms with applications both within and outside root analysis.

- Aim 1: Topological simplification: To address topological errors in shapes reconstructed from raw data, we develop a global optimization approach to topological simplification. Our algorithm seeks to maximally simplify all three types of topological features (connected components, handles, and voids) while minimizing the change to the geometry.

- Aim 2: Computing fine-grained root architecture: to address the need for accurate computation of root architecture from 3D imaging with high throughput, we develop a pipeline of geometric algorithms (TopoRoot) for automatically computing architectural traits from the grayscale images. Our pipeline demonstrates improved accuracy and variety of traits compared to previous methods.

- Aim 3: Topological simplification of nested shapes: Given a sequence of nested shapes, we aim to simplify the topology of all shapes in the sequence in a consistent fashion while minimizing the change to the geometry.

1.4.4 Timeline

All three goals have been achieved at the time of this writing. Aim 1 was achieved in 2020, published to SIGGRAPH Asia. Aim 2 was achieved in 2021, published to Plant Methods Journal. Aim 3 was achieved in Spring 2022, with the manuscript currently under review for publication to the European Symposium on Geometry Processing.

Chapter 2: Topological Simplification

2.1 Introduction

Reconstructing shapes from raw data is an important task in many graphics and imaging applications. Whether it's a grayscale image such as a CT scan, or a point cloud, obtaining surface reconstructions from these data can be a very challenging task. The resulting shapes often have many topological features in the form of connected components, handles, and cavities. While some of these features are intended as part of the natural semantics of the shape, other features are artifacts of the reconstruction. Excessive topological features make it quite challenging for many tasks in imaging and geometry processing, such as shape analysis, parameterization, and physical simulations. These challenges have led to the growing field of topological data analysis (TDA), which seeks to provide topological and geometric tools to determine the most salient features for complex, noisy data. A key component of TDA is topological simplification, which seeks to remove excessive topological features from a shape by either cutting contents from the shape or filling in content. Cutting and filling can both have the same topological effects, but can have different effects on the geometry of the shape. Ideally, a topological simplification algorithm should maximally simplify the shape in terms of its topology while minimizing the change to its geometry. A variety of monotonic and non-monotonic algorithms have emerged to address topological simplification. Most methods fall under the monotonic category, performing only cutting or only filling to the shape. These methods can make excessive changes to the shape when there are some features that should be cut and others which should be filled in the same shape. A few other methods take a non-monotonic approach, cutting some topological features away, while filling others in. However, these methods perform heuristics which only locally minimize the change to the geometry, that can result in globally suboptimal decisions (Fig. 2.1).

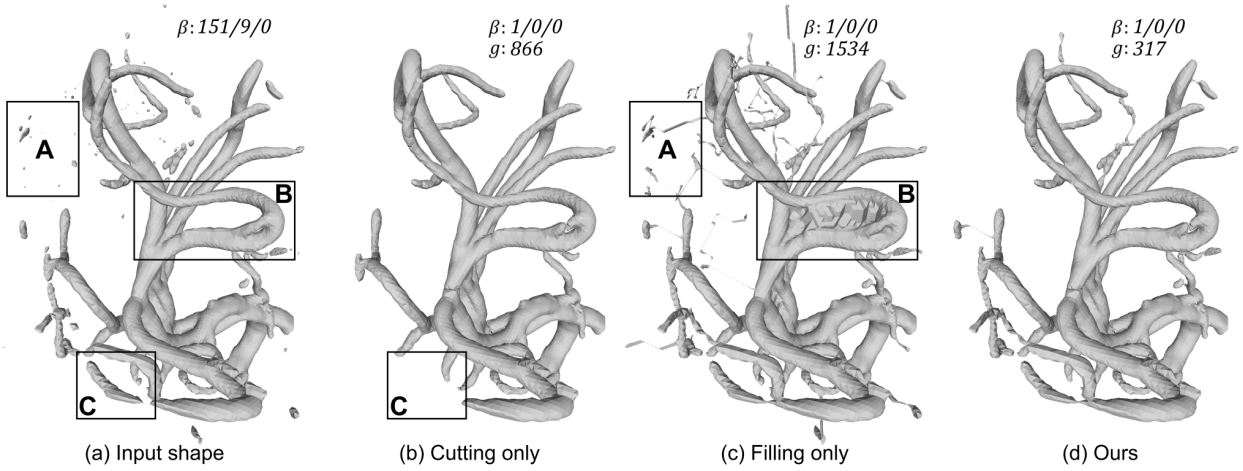

Figure 2.1: To simplify the topology of a 3D shape (a), performing cutting alone (b) or filling alone (c) results in excessive changes, such as removing large components (box C in (b)), creating long bridges to distant islands (box A in (c)) and large patches to fill in a handle (box B in (c)). Given a set of pre-computed cuts and fills, our method optimally selects a subset of them to maximally simplify topology while minimizing the impact on the geometry (d). (β: number of connected components, handles, and cavities; g: geometric cost)

2.2 Previous works

Here we provide a brief review of existing methods for topological simplification of 3D shapes, with an emphasis on how the decisions of where to cut and fill are made. An in-depth review of many of these methods can be found in the survey [36].

2.2.1 Monotonic methods

These methods perform either only cutting or only filling on the 3d shape. Several methods in this category remove topological handles by representing the shape as a connectivity graph, and weight the edges by a geometric cost of cutting or filling. The graphs can be constructed in a variety of ways: axis-aligned reeb graphs [37, 38], an adjacency graph of segmented parts [39], and a curve skeleton [40]. A minimum spanning tree (MST) can then be computed from the graph. This approach solves the handle-cutting or handle-

fitting problem optimally, but does not make globally optimal decisions between cutting and filling. To simplify all three types of topological features, morphological opening or closing can also be applied to voxelized shapes [41]. A structuring element can be used to perform opening to cut away voxels from the shape or closing to add voxels to the shape. However, morphological operations can add new topological features to the shape, and there is no direct way to control the resulting topology. [42, 43, 44] control simplification by either deflating or inflating an initial seed towards the shape while not allowing (or only conditionally allowing) topological changes. [45] uses a heuristic to explore candidate fills in order to maximally simplify the topology. However, since all of these methods are monotonic, they can lead to excessive shape modifications.

2.2.2 Non-Monotonic methods

These methods perform both cutting and filling on the shape. Most non-monotonic methods only remove topological handles, and minimize different metric of geometric changes, including the lengths of loops on the surface [46], the volume of voxels to be cut or filled [43], and the volume weighted by the image intensity [47]. Prior to our work, only the MendIT method of [48] can remove all three types of topological features. All of these methods use a greedy heuristic to decide which feature should be removed in the current step and whether it should be cut or filled. However, greedy strategy often fails to make globally optimal decisions. An example can be seen in Figure 2.2.

2.3 Definitions

Our algorithm operates on shapes represented as cell complexes (Figure 2.3). We provide here a brief review on their definitions and properties. An in-depth discussion can be found in [49]. A cell complex decomposes a space into topologically simple units, called cells. A k-dimensional cell is an open set homeomorphic to an open k-dimensional ball. A

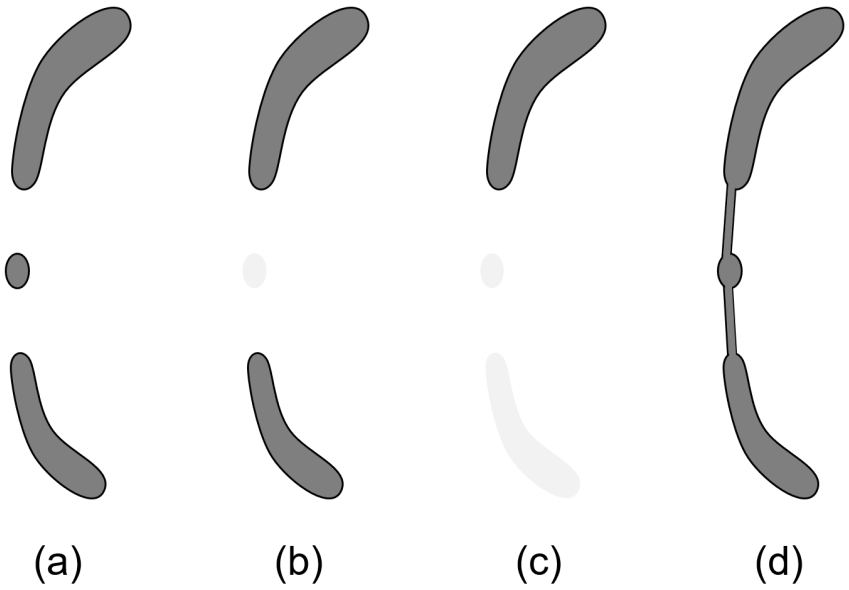

Figure 2.2: A scenario where a greedy heuristic for deciding where to cut and fill produces a suboptimal result. To reduce the number of connected components of the shape in (a), a greedy heuristic would cut away the smaller middle island in (b), but this simplification creates a large gap between the bottom and top components, which forces the greedy approach to remove the bottom component in (c). The optimal solution, with a lower overall change to the shape, would instead keep the middle island in order to bridge the larger two components (d).

cell x is a face of cell y if x is contained in the boundary of y. A set of disjoint cells is a cell complex if, for each cell in the complex, all its faces are also in the complex. Two sets of cells are connected if some cell in on set is the face of a cell in the other set.

The n-th Betti number β_n of a cell complex is the rank of the n-th homology group. $\beta_0(X)$ is the number of connected components of X, $\beta_1(X)$ is the number of topological handles of X, and $\beta_2(X)$ is the number of cavities in X. The alternating sum of Betti numbers defines the Euler characteristic γ:

$$\gamma(X) = \beta_0(X) - \beta_1(X) + \beta_2(X) \tag{2.1}$$

The euler characteristic can also be found by the alternating sum of the number of cells in

X in different dimensions:

$$\gamma(X) = k_0(X) - k_1(X) + k_2(X) - k_3(X) \tag{2.2}$$

where $k_d(X)$ represents the number of d-dimensional cells in X.

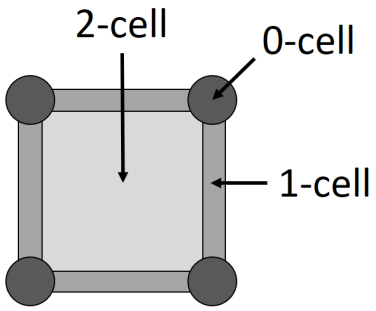

Figure 2.3: A simple sketch of a 2D cell complex.

2.4 Problem Statement

Topological simplification aims to remove as many topological features as possible (connected components, handles, cavities). However, since the number of possible modifications to the shape is virtually infinite, we instead consider a limited space of shape modifications in the form a predefined set of cuts and fills. These cuts and fills can be computed by running any of the monotonic simplification methods from section 2.2.1, or by performing morphological opening or closing. Using the notion of cuts and fills, the problem reduces to choosing a subset of the cuts and fills with the lowest total geometric cost such that the resulting shape has the simplest topology.

Formally, we represent each shape in the sequence by a 3D cell complex Ω. Let O be the set of all 3-cells (e.g. hexahedra or tetrahedra). Our input is a set of *shape* cells $T \subset O$, *cut* cells $C \subset T$ and *fill* cells $F \subset O \quad T$, and a geometric cost $g(v)$ for each cut or fill cell v. For the purpose of counting topological features, we define the shape as the closure $\Omega(T)$, which is the cell complex composed of 3-cells in T and their lower-dimensional faces. We

seek a binary labelling of the cut and fill cells (label 1 for object, label 0 for background) such that the resulting shape has the lowest number of topological features, and the sum of the geometric costs of all selected cuts and fills is minimal. Given the labelling as L and the cut cells as C (resp. fill cells F), the cut cells which are selected to be removed from the shape and fill cells which are selected to be added to the shape are denoted as $C_{L,0}$ and $F_{L,1}$. For each shape in the input sequence, we seek to minimize the following vector *lexicographically*:

$$\{\beta_0(X) + \beta_1(X) + \beta_2(X), \sum_{v \in C_{L,0} \cup F_{L,1}} g(v)\} \tag{2.3}$$

This objective may be expressed as minimizing the following scalar energy:

$$E(L) = 2\lambda * (\beta_0(X) + \beta_2(X)) - \lambda * \gamma(X) + \sum_{v \in C_{L,0} \cup F_{L,1}} g(v) \tag{2.4}$$

where λ is any constant greater than $\sum_{v \in C_{L,0} \cup F_{L,1}} g(v)$ and $\gamma(X)$ is the Euler characteristic.

2.5 Graph formulation

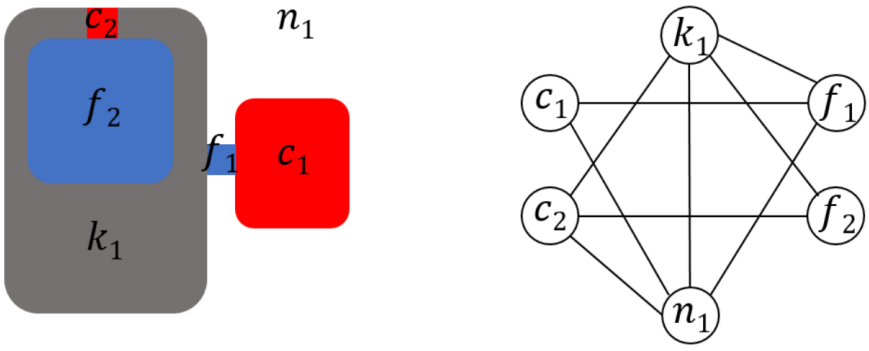

Figure 2.4: Given a shape (left) partitioned into kernel ($k1$), cut ($c1$ and $c2$), fill ($f1$ and $f2$), and neighborhood (n1) regions, we construct a graph (right) representing its connectivity.

To solve our labeling problem, we first re-formulate it as a graph labelling problem. To faithfully capture the topology of the shape, our graph represents a partitioning of the 3D space into regions of different types (Figure 2.4). These are cut cells, fill cells, kernel cells (shape cells subtracted by the cut cells), and neighborhood cells (the complement of the shape cells plus fill cells). Our graph is constructed such that each connected component of the cells also corresponds to a connected component in our graph. To guarantee this, we first provide a ranking of the four types of cell in the decreasing order of kernel, cut, fill and neighborhood. We define a rank-based connectivity, called R-connectivity, in which two 3-cells u,v are R-connected if they share a common face that is not the face of another 3-cell whose rank is higher than both u and v. In the appendix, we prove the following proposition:

Proposition 1. *A labelling L is called* proper *if it satisfies the following constraint: if a cut node s and a fill node t are connected by an edge, than either $L(s) = 1$ or $L(t) = 0$. Denote the modified shape as $X = (\cup_{L(s)=1} O_s)$. For any proper labelling L,*

1. $\beta_0(X) = \beta_0(G_{L,1})$.
2. $\beta_2(X) = \beta_0(G_{L,0}) - 1$.

where $\beta_0(S)$ counts the number of connected components in a graph S.

By this proposition, our graph, based upon R-connectivity, captures the topology of the shape such that the connected components of the 1-labelled subgraph (resp 0-labelled subgraph) are equivalent to the connected components of the 1-labelled cells (resp. 0-labelled cells). Each cut or fill node is associated with a node-wise cost, that is obtained from the euler characteristic and geometric cost of the cells which compose the node. A cut node with a 0-labelling or a fill node with a 1-labelling both have positive costs since these are opposite to their labellings in the original shape. We seek a 1/0-labeling that minimizes the sum of the number of connected components of the 1-labelled and 0-labelled

subgraphs, and the total labelling costs of all nodes. The goal of the topological labeling (TL) problem is to find a proper labelling L of graph G that minimizes

$$E_G(L) = 2\lambda * (\beta_0(G_{L,0}) + \beta_0(G_{L,1})) + \sum_{s \in V_C \cup V_F} h(s, L(s)). \tag{2.5}$$

2.6 Graph Labeling

2.6.1 As-connected-as-possible (ACAP) Labeling

Given that the TL problem involves connectivity, this naturally led us in the direction of Steiner Trees, a problem for which mature optimizers have been developed. To make this transformation, we come up with a variant of TL in which the two labelled subgraphs $G_{L,0}$ and $G_{L,1}$ are constrained to be as connected as possible. We introduce the concept of a reachable set, which is a set of kernel (resp. neighborhood) nodes that are connected when all cut and fill nodes are labelled 1 (resp. 0). We call a labelling as-connected-as-possible (ACAP) if any reachable set of kernel (resp. neighborhood) nodes are connected in the 1 (resp. 0)-labelled subgraph. ACAP-TL differs from TL in two ways. First, the solution to ACAP-TL may not be the solution of TL, because a labelling that minimizes the TL energy may not have to be ACAP. For example, the solution of TL may have more connected components than the solution of ACAP-TL, but still have fewer handles. Second, a solution of ACAP-TL may not exist, if no labelling can satisfy the ACAP constraint. For example, a cut or fill node may be both an articulation node in the 1-labeled and 0-labeled subgraph. These differences are addressed in section 2.6.3.

2.6.2 Reduction to Node-weighted Steiner Tree

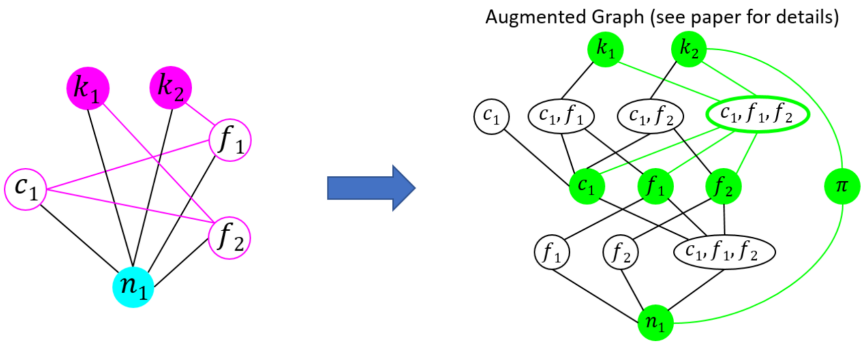

Figure 2.5: Constructing our augmented graph from the initial graph of our formulation.

We reduce the ACAP-TL problem to the Node-Weighted Steiner Tree (NWST) problem through the construction of a new graph (H) (Figure 2.5) derived from the original graph of our formulation (G). We provide here a brief summary, with further details being provided in the paper. We create a terminal node in H for each kernel node and neighborhood node in G. For each group of terminals representing a reachable set of kernel nodes, we select a terminal in that group and connect it with an auxiliary terminal node. We further create two types of non-terminal nodes in H, each representing a group of cut and fill nodes in G. Selecting a node representing a group of nodes from the original graph corresponds to labelling all cut and fill nodes within the group as 1 or 0, and that any NWST solution will result in a proper labelling of the shape (as defined in proposition 1). A new type of terminal node is constructed for each cut and fill node. These terminals ensure that in a subgraph of H that connects all terminal nodes (the result of a valid NSWT), each cut or fill node of G will be represented by at least one non-terminal node, and hence provided with a labelling.

2.6.3 Algorithm

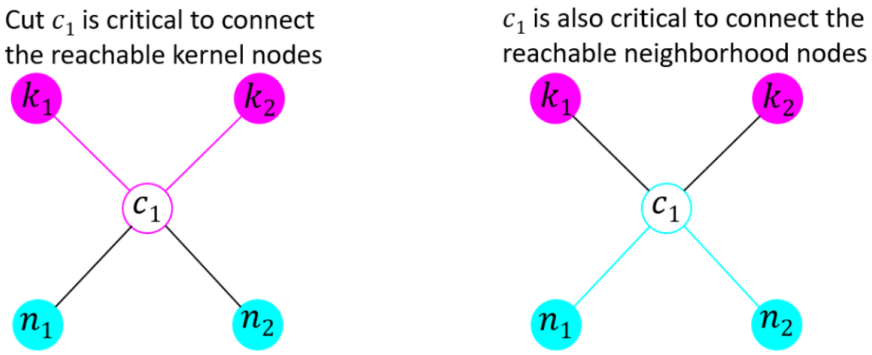

Figure 2.6: Cut c_1 is an articulation node in both the 1 and 0-labeled subgraphs. In such a case, the steiner tree solver, when run on the augmented graph, would label c_1 to be both 1 and 0 in order to connect both subgraphs.

While the reduction to NWST solves ACAP-TL, a few adjustments are needed to reconcile the differences with respect to the original TL problem. First, a cut or fill node may be labeled to both 1 and 0 if either a solution to ACAP-TL does not exist, or the Steiner Tree solver produces a suboptimal solution. An example would be a double-articulation node, a node which is necessary to connect the reachable kernel and neighborhood nodes in both the 1 and 0-labelled subgraphs. Second, a labelling that minimizes the energy of ACAP-TL may be different from that which minimizes the different energy for TL.
To resolve the first issue, doubly-labelled nodes are greedily sent to the object or background, depending on which has the lowest lexicographical cost in the topological change and geometric change. For the second issue, the cuts and fills are not doubly labeled, but rather the solution is suboptimal with respect to TL. In these cases, we greedily iterate through the cut and fill nodes, and flip the labeling if doing so results in a more optimal solution in terms of the TL energy.

2.7 Results

Experiments were performed on cuts and fills computed using two methods: intensity-based and morphological opening and closing-based. Intensity-based cuts and fills may be produced by running existing monotonic simplification methods [42, 43, 50] in either fill-only or cut-only modes. All of these methods operate on a voxel grid. We compare our method to TopoMender [40], a method that removes handles using an MST-based approach, and MendIT [48], a non-monotonic method that follows a greedy approach to remove all three types of topological features. We solve the NWST problem using the the recent branch-and-bound solver of [51], which performed well on public benchmarks (e.g. [52]) as well as on our own data. We found that our solver is able to return an optimal solution within seconds for medium-sized graphs, in some cases containing thousands of nodes. Presented here are results using intensity-based cuts and fills.

2.7.1 Intensity-based cuts and fills

We first provide results computed using intensity-based cuts and fills.

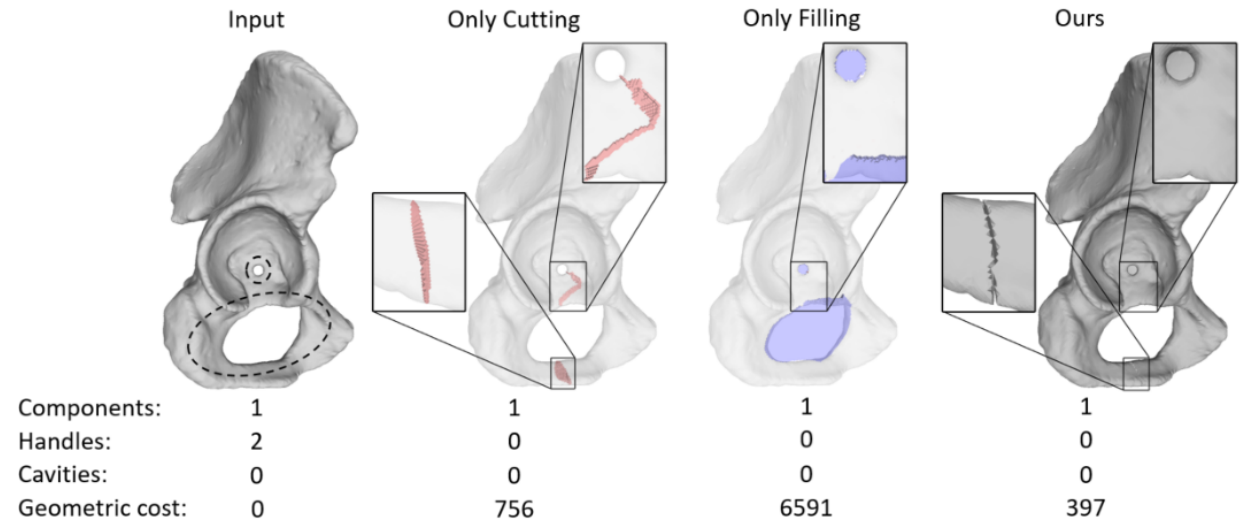

Figure 2.7: Hip model: Comparing the amount of topological simplification and geometric cost for only cutting, only filling, and our method. Our method performs smaller geometric changes by using a combination of cutting and filling.

On a hip model with two handles, only cutting results in an excessive removal of voxels for the smaller handle, while only filling results in an excessive addition for the larger one. Our method is able to use a combination of cutting and filling to make a smaller total geometric change than either method.

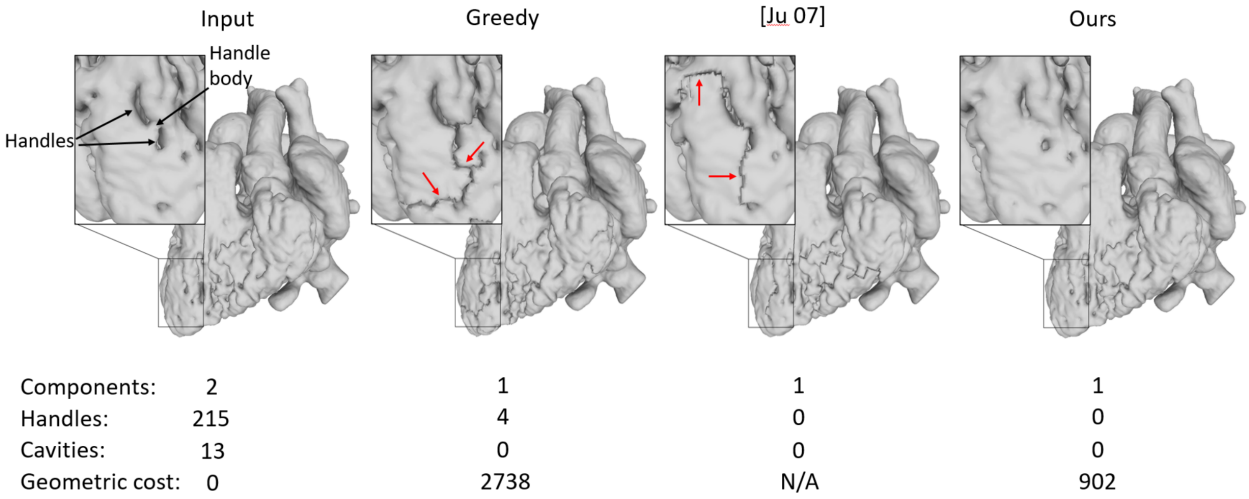

Figure 2.8: Heart iso-surface: Comparing the amount of topological simplification and geometric cost for a greedy method, MendIT [27], and our method. Our method performs smaller geometric changes by using a combination of cutting and filling.

On a heart example with multiple components, handles, and cavities, our method is able to fully simplify the topology, while the greedy strategy incurs a larger geometric cost. The greedy method iteratively goes through all cut and fill nodes and flips them to the opposite of their original labeling as long as their is a topological or (secondarily) a geometric benefit. The arrows show regions where both the greedy method and MendIT [48] make excessive geometric changes.

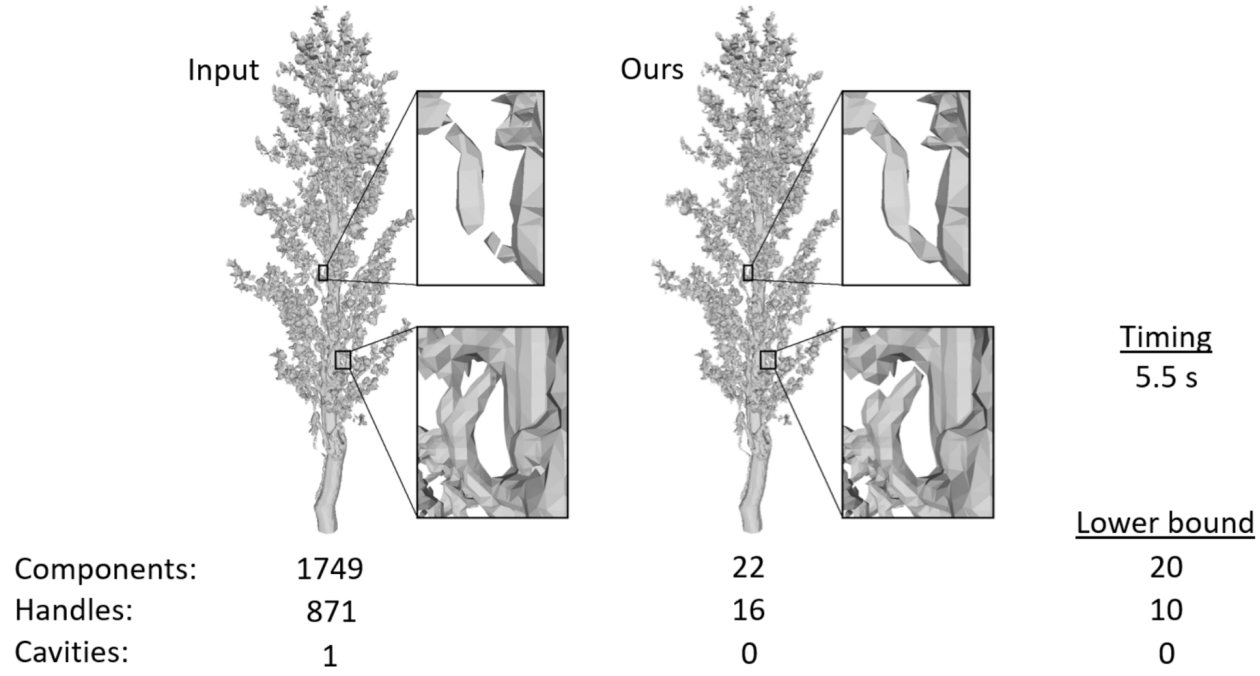

Figure 2.9: Iso-surface of a sorghum panicle, before and after topological simplification.

On an iso-surface from a CT scan of a sorghum panicle with thousands of topological features, our method is able to perform a combination of cutting and filling to drastically reduce the number of topological features within a matter of seconds.

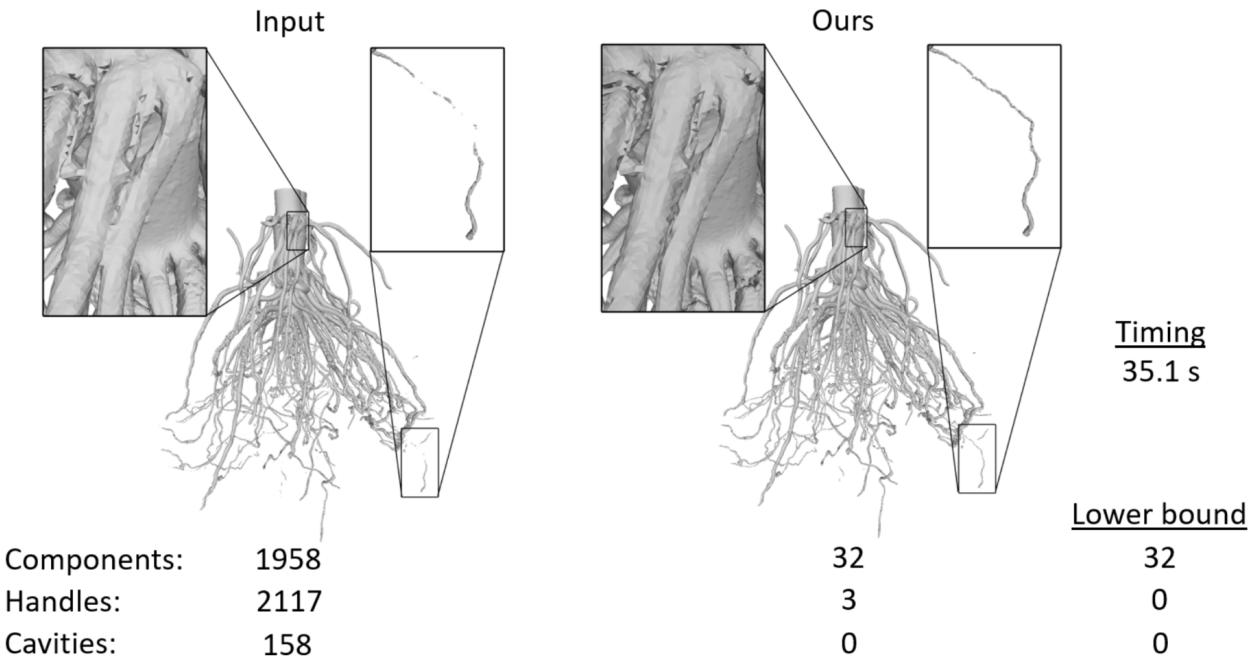

Figure 2.10: Iso-surface of a corn root, before and after topological simplification..

On an even more topologically example, an iso-surface from a CT scan of a corn root with thousands of topological features, our method is able to perform a combination of cutting and filling to drastically reduce the number of topological features within a matter of seconds.

2.7.2 Morphological cuts and fills

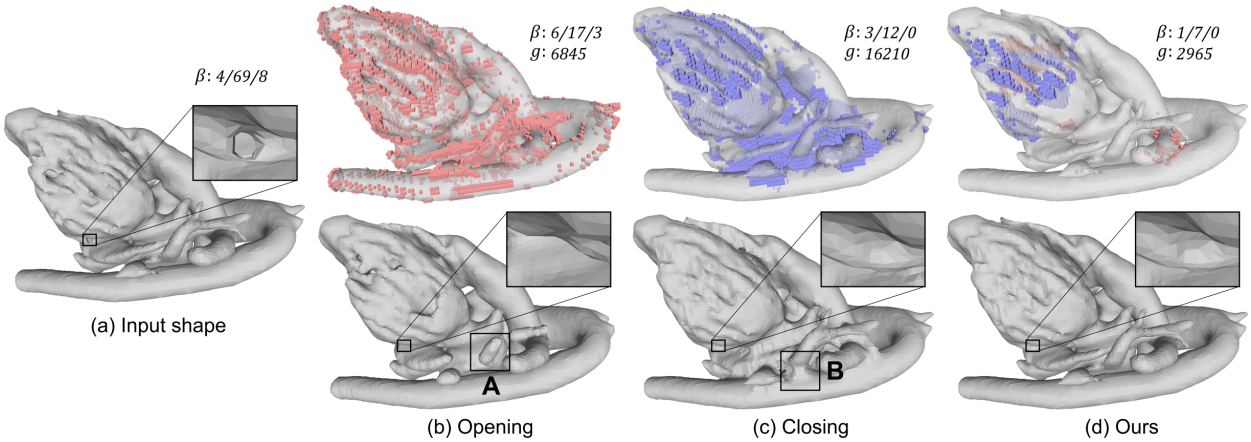

Figure 2.11: (a) A heart segmentation. (b): Result of opening (bottom), which contains several new islands (box A), and the corresponding cut voxels (top). (c): Result of closing (bottom), which merges nearby vessels (box B) and thereby creating new handles, and the corresponding fill voxels (top). (d): Voxels selected by our algorithm (top) and the modified shape (bottom). The insert examines a handle that is removed in all three methods.

We next provide results computed using cuts and fills computed using morphological opening and closing. Morphological opening and closing are often used to simplify the topologies of voxelized shapes. However, they may introduce new topological features that are not present in the original shape. As an example, opening may create new components (box A in Figure 2.11). Opening and closing may also result in excessive modifications to the shape, most of which do not contribute to simplifying the toplogy (see cut and fill voxels at the top of Figure 2.11 (b,c)). Additionally, opening and closing are monotonic operations, since opening only performs cutting, while closing only performs filling. We address these shortcomings by running our algorithm on the results of both opening and closing, taking the difference between the shape and its opening (resp. closing) as the cut (resp. fill) voxels, and assign each voxel a unit geometric cost. Our algorithm removes topological features and incurs a lower geometric cost than either opening or closing alone (Figure 2.11 (d)).

2.7.3 Conclusion

We developed a global optimization algorithm which attempts to maximally simplify all three types of topological features (connected components, handles, and voids) of a 3D shape, while minimizing geometric change. While existing methods limited their modifications to be only cutting or only filling, or take a greedy heuristic, we consider the problem of finding a globally optimal solution. We show that the problem can be formulated as graph labelling, and solve it through a transformation to the Node-Weighted Steiner Tree problem. We show significant improvements over existing simplification methods in both topological simplicity and geometric distortion, and perform these simplifications within seconds even on very complex examples.

Chapter 3: Computing root traits

With an algorithm to repair topological noise which is often present in 3D imaging of maize roots, we are better equipped to address goal 2 of this dissertation.

3.1 Introduction

As discussed in Chapter 1, root system architecture (RSA) plays a crucial role in the health and growth of the plant. Analyzing the genetic control of RSA traits enables potentially leads to the development of highly productive crops while minimizing adversarial environmental effects. However, traditional studies of RSA involve excavating the roots from the soil, washing them, and measuring them using devices such as rulers, calipers, and protractors. This process is not only labor-intensive but also prone to human errors. More importantly, many aspects of RSA, particularly those pertaining to lateral roots of higher order, are almost impossible to measure by hand.

Advances in 3D imaging, including X-ray CT, MRI, and optical imaging [13, 11, 12], have allowed root shapes to be captured digitally either after excavation or in situ. However, most image-based root phenotyping methods only compute overall traits such as the volume, depth, convex hull volume, total root length, and root number [28, 53, 22, 54]. Though useful, these traits do not capture the branching structure or the hierarchical organization of individual roots, which provide a much more comprehensive description of RSA.

To our knowledge, DynamicRoots [29] is the only published and validated root phenotyping method that produces a full branching hierarchy and root traits associated with each hierarchy level in 3-dimensions. DynamicRoots is designed for a time-series of root systems grown in transparent gel [55]. These seedling-stage root systems tend to have a relatively simple geometry and structure, which makes it possible to obtain high quality 3D voxelized reconstructions using multi-view imaging [56]. While, in theory, DynamicRoots can process any segmented 3D root image, its accuracy can be significantly affected by the

number of topological errors in the segmentation, such as disconnected root components and root branches forming loops due to touchings. Although such errors are scarce in the multi-review reconstruction of simple seedling-stage roots, they can be abundant in 3D images of more complex root systems. In addition, DynamicRoots requires a time series to obtain the correct root hierarchy.

3.2 Background

3.2.1 Topology simplification

As mentioned in the first part of the proposal, root segmentations often contain many topological errors in the form of excessive connected components (e.g. disconnected roots), cycles (e.g. false intersections between branches), and voids (empty regions within the stem and roots). These errors are detrimental to recovering the correct branching structure of the root, which ideally should be composed of a single connected component with zero genus and no internal voids. Few works have attempted to resolve topological issues in the context of root imaging. [56] proposed a 3D reconstruction method from 2D images of roots growing in gel, which can ensure that the segmentation is a single connected component. However, the result can have a high genus. In the graphics community, many methods were developed to simplify the topological structure of a 3D shape [40, 45, 37, 38, 39, 57]. In this pipeline, we employ the global optimization method of the first part of the proposal in order to fix all three types of topological errors, through a combination of cutting (removing voxels from the shape) and filling (adding voxels) that minimizes the change to the geometry, while also considering the image intensity.

3.2.2 Skeletonization

Skeletons provide for a concise representation of the geometry and organization of the root structure. As such, several existing image-to-analysis pipelines have used skeletons to extract coarse-grained traits [27, 28, 24]. There are generally two types of skeletons: voxelized and geometric. Methods to generate voxelized skeletons [58, 59, 60, 61] are usually easy to implement and numerically robust, but are often expensive in both time and memory since they compute the skeleton from the interior of the shape. Voxel-based methods also lack theoretical guarantees on the ability to approximate the geometry of the input shape. In contrast, geometric skeletons save time and space by sampling points along the boundary of the shape instead of from the interior, and are equipped with strong theoretical guarantees on preserving the topology and geometry of the input shape [44]. In our work, we compute a geometric skeleton in a two-step process, first thinning the image segmentation to its 2D medial axis using the Voxelcore method [62] and further down to a one-dimensional, noise-pruned curve skeleton using the erosion thickness method proposed in [63]. Compared to other methods for computing geometric skeletons, our combination of [62] and [63] is more scalable to large volumes, numerically robust, and specifically designed for voxelized shapes, while retaining similar theoretical guarantees as other geometric skeletons.

3.3 Our Contributions

In this part of the dissertation we present TopoRoot, a method for obtaining the complete root hierarchy and associated fine-grained traits of a mature maize root system (or crown) from a single 3D image. Compared with DynamicRoots, TopoRoot is designed to deal with topological errors, which are common in images of complex root systems, and to infer the hierarchy without the need for a time series. TopoRoot builds on several state-of-the-art algorithms from computer graphics, including topological simplification and 3D

skeletonization, and introduces customized heuristics tailored to the maize root structure. TopoRoot is validated on both real and simulated data. On a set of 45 X-ray CT scans of excavated maize root crowns, TopoRoot shows dramatic improvements in accuracy over DynamicRoots in counting the number of nodal roots. On another set of 495 synthetically generated images of maize root systems simulated by OpenSimRoot [64] with varying age, complexity, and noise level, TopoRoot exhibits improved accuracy in a variety of coarse-grained and fine-grained traits over DynamicRoots and GiaRoots [54].

TopoRoot is completely automated and requires setting only three thresholds for each image. On a standard desktop computer, TopoRoot runs within a few minutes for images at the resolution range of 400^3. This makes TopoRoot suited for batch processing a large set of images in a high-throughput analysis pipeline. The software package is freely distributed on GitHub with our X-ray CT dataset.

3.4 Method

The input to the TopoRoot pipeline is a 3D grayscale image, I, of a maize root crown or root system, represented as a stack of 2D image slices. The pipeline assumes that I has sufficient contrast between the roots and their surroundings (e.g., soil), so that intensity thresholding can be used to segment the roots. TopoRoot produces a root hierarchy and fine-grained root traits in four steps.

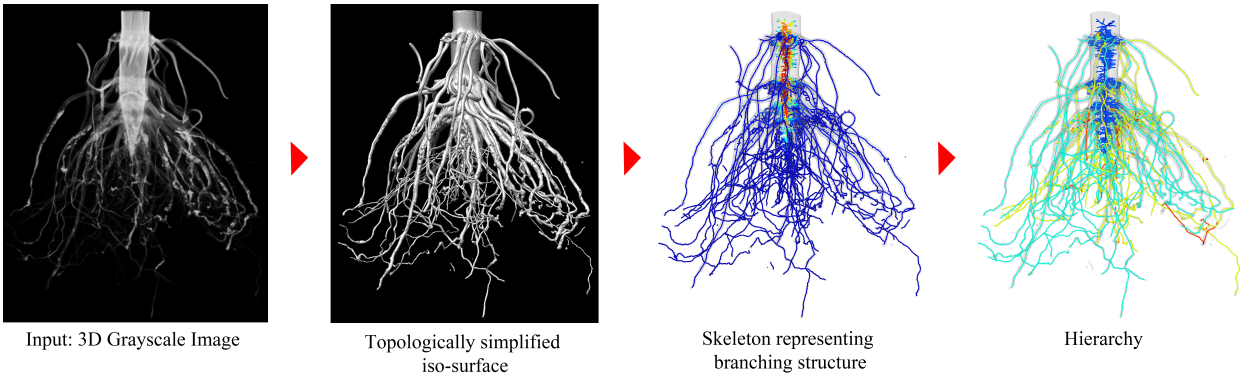

Figure 3.1: The pipeline of TopoRoot for computing fine-grained traits from a 3D image. Beginning from a 3D grayscale image I represented as a stack of image slices (A), TopoRoot first computes a topologically simple segmentation B (B), then it creates a geometric skeleton S capturing the branching structure (C), from which a hierarchy H is obtained (D) and the traits are subsequently computed.

3.4.1 Segmentation

Each maize root system has a simple topology: it is connected, free of handles ("loops") or voids ("air bubbles"). However, due to limits in imaging resolution, contrast, and/or noise level, simple thresholding of the input grayscale image often yields a segmentation with numerous disconnected components, handles and voids (see Fig. 3.2A). These erroneous topological features pose significant challenges for the subsequent hierarchy inference. Another issue with simple thresholding is that due to the relatively low intensity in the interior of thick roots (e.g., the stem), only the outer shell of these roots is included in the segmentation (see Fig.3.2A). These hollow shells would lead to complex skeletons that do not accurately capture the tubular shape of the roots. Given an input image I, the first step extracts a segmentation B that fills the hollow root interior and has minimal topological errors.

We start by filling the hollow interior of roots using an erosion approach. The observation is that these hollow spaces become voids (i.e., closed off by the root's shell) when the image I is thresholded by a sufficiently low value. Our heuristic identifies these voids at a

low threshold and fills them in after "growing" them back to the normal threshold. The heuristic takes in two user-specified thresholds, t_{mid} and t_{low}, such that t_{mid} best captures the shape of the root while $t_{low} < t_{mid}$ closes off most of the hollow spaces within roots. We denote the segmented, voxelized shapes at these two thresholds as B_{mid} and B_{low} respectively (see Fig. 3.3A). To grow the voids in B_{low} back to the hollow spaces in B_{mid}, we maximally erode the voxels in B_{low} while maintaining its topology and preventing voxels in B_{mid} from being eroded. This results in another shape, denoted as B'_{mid}, which is a minimal superset of B_{mid} with its hollow spaces closed off (see Fig. 3B). We then take all voxels in the voids of B'_{mid}, together with those voxels in $B'_{mid} \setminus B_{mid}$ adjacent to the voids, and "fill" them by setting their intensity values to be t_{mid} (see Fig. 3.3C). We denote the resulting image as I'.

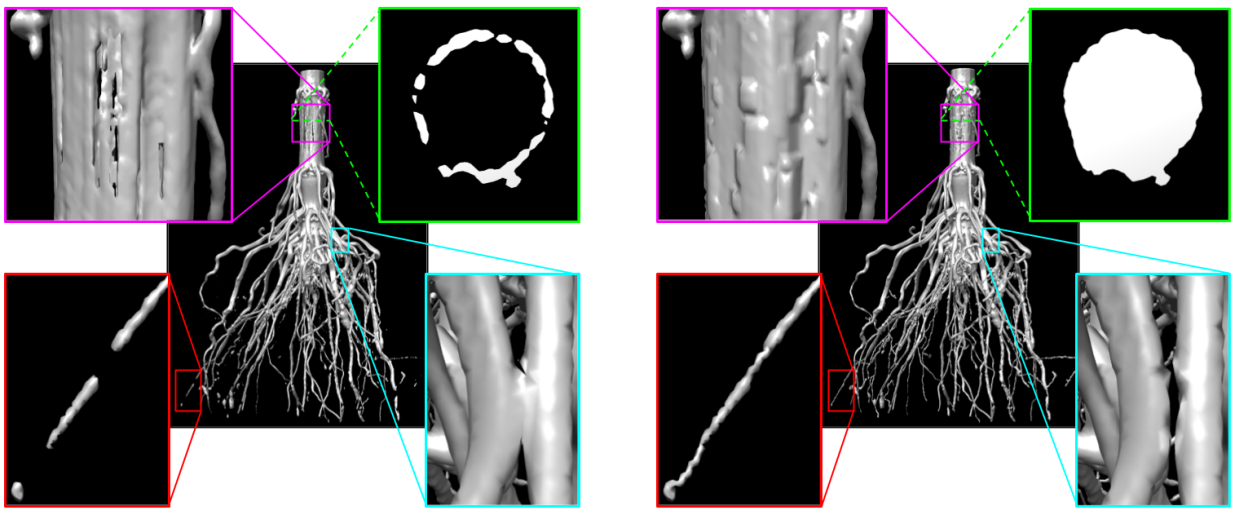

Figure 3.2: Segmenting a root image. (A) Thresholding the image I yields numerous topological errors such as disconnections (red box) and handles (cyan and purple boxes), and the stem has a hollow interior (green box). (B) Applying our filling heuristic followed by the algorithm of [57] yields a segmentation B with these topological errors removed and the stem filled

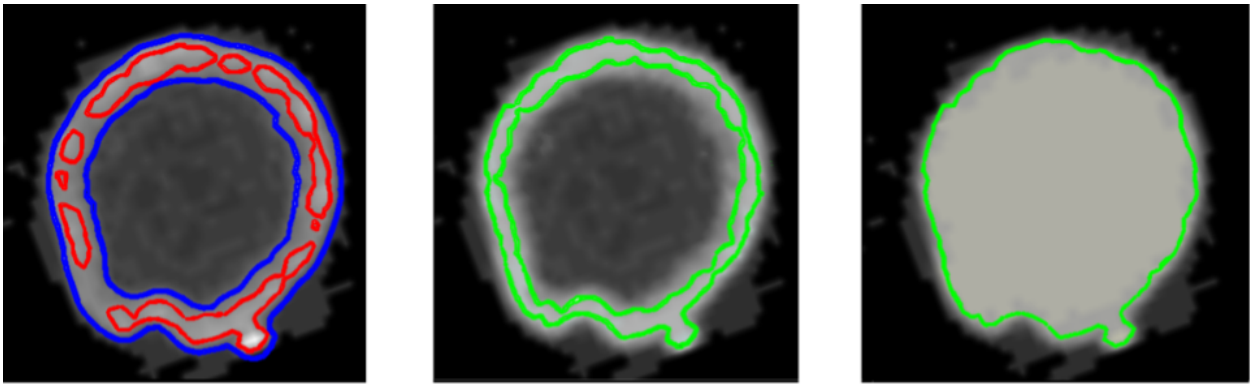

Figure 3.3: Filling the hollow space inside the stem and thick roots. (A) A slice of I showing the cross-section of the stem with the outline of segmentations B_{mid}, B_{low} (red and blue) at thresholds t_{mid}, t_{low}. The hollow space within the stem is connected to the outside in B_{mid} but is closed off by the stem's shell in B_{low}. (B) Eroding B_{low} onto B_{mid}, while preserving its topology, results in the shape $B^{'}_{mid}$ (green), which adds a minimal set of voxels to B_{mid} to "close off" the hollow space. C The hollow space is filled by raising the intensity value of the voxels in the void of $B^{'}_{mid}$ and the voxels in $B^{'}_{mid} \setminus B_{mid}$ adjacent to the void to B_{mid}. This creates a new image $I^{'}$.

Next, we compute a segmentation of the hollow-space-filled image $I^{'}$ using the algorithm of [57]. This algorithm uses global optimization to extract a segmentation bounded between two intensity thresholds that has the least number of topological features. It takes three thresholds with increasing values, t_{low} ¡ t_{mid} ¡ t_{high}, where t_{low},t_{mid} are the same as in the filling algorithm above. Let the segmented shapes of $I^{'}$ at these thresholds be B_{low}, B_{mid}, B_{high}. The algorithm of [20] computes a shape B such that $B_{high} \subseteq B_{mid} \subseteq B_{low}$ and the following vector energy is minimal in lexicographical order,

$$\{\beta_0(B) + \beta_1(B) + \beta_2(B), diff(B, B_{mid})\} \tag{3.1}$$

Here, β_0, β_1, β_2 counts the number of connected components, handles and voids of B, and $diff$ is a difference measure between two voxel sets that considers both the number and intensity of voxels that are in one of the sets but not the other. Intuitively, B makes the least change to the shape B_{mid} (in terms of $diff$) to remove as many topological features

on B_{mid} as possible while sandwiched between B_{high} and B_{low}.

An example result of this step (filling hollow spaces and then applying the algorithm of [57]) is shown in Fig. 3.2B. The three thresholds t_{low}, t_{mid}, t_{high} control the trade-off between topological simplicity and geometric fidelity of the segmentation B. A larger gap between t_{mid} and t_{low}, t_{mid} gives the algorithm [57] more room to remove topological features, and hence the result has fewer topological errors. But this comes at the cost of possibly large and undesirable geometric changes; for example, a root may be broken in the middle to remove a topological handle. We found that the best results are obtained by setting t_{low} to be the highest value such that thin roots remain connected in B_{low} and setting t_{high} to be the lowest value before roots start to merge in B_{high}. The resulting segmentation B will not be completely free of topological errors but fixing these remaining errors in a geometrically correct way requires a more global context of the root shape. This will be addressed in the next step and with the help of a geometric skeleton.

3.4.2 Skeletonization

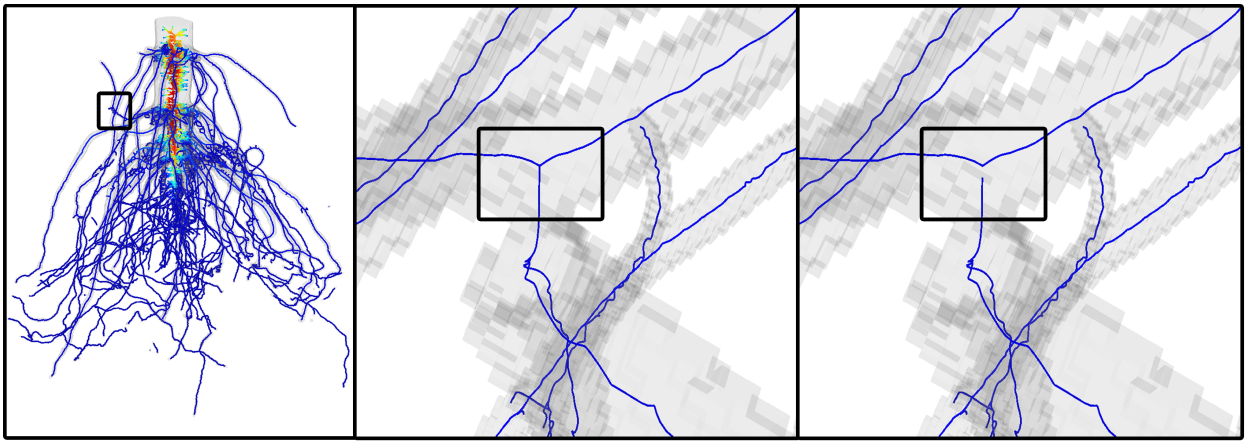

Figure 3.4: Computing a cycle-free skeleton. A An initial curve skeleton S0 computed by the methods of [62, 63], where color indicates the thickness of the roots (redder curves lie in thicker roots). B A skeleton junction (in the boxed region in (A)) caused by two roots touching in the segmentation, which leads to a cycle in the skeleton. C The same region on the final skeleton S where the cycles have been removed using a graph-based algorithm

The tubular shape of roots makes them representable by curve skeletons. The graph structure of a skeleton is the key that enables subsequent analysis of branching hierarchy and traits. Given the segmentation B produced by the previous step, this step produces a geometric skeleton S capturing the shape and branching structure of the roots. We will utilize the structural information provided by the skeleton to resolve the topological errors that remain from the previous step, so that S is connected and free of cycles.

We first compute an initial curve skeleton S_0 from the voxel shape B using the algorithms described in [62, 63]. These methods have been recently used in skeleton-based phenotyping of sorghum panicles [14]. Specifically, the Voxel Core method of [62] extracts an approximate medial axis of B, which is a triangulated 2-dimensional non-manifold that lies at the center of B. Taking the medial axis as the input, the Erosion Thickness method of [63] reduces it to a polygonal 1-dimensional skeleton S_0. Compared with other means for computing skeletons, this approach has several features important for root analysis. Both

methods [62, 63] are robust to irregular shape boundaries, and hence spurious skeleton branches are minimized. Both methods also preserve the topology of B exactly, and hence S_0 carries the same set of topological features as B without adding new features. Both methods are highly optimized and capable of processing large 3D images. Unlike methods that produce skeletons made up of voxels, S_0 is made up of vertices and edges, and hence it can be conveniently processed by graph algorithms. Finally, the method of [63] also outputs a "thickness" measure for each edge of S_0, which will be utilized later. An example of this initial skeleton is shown in Fig. 3.4A (also in Fig. 3.1C).

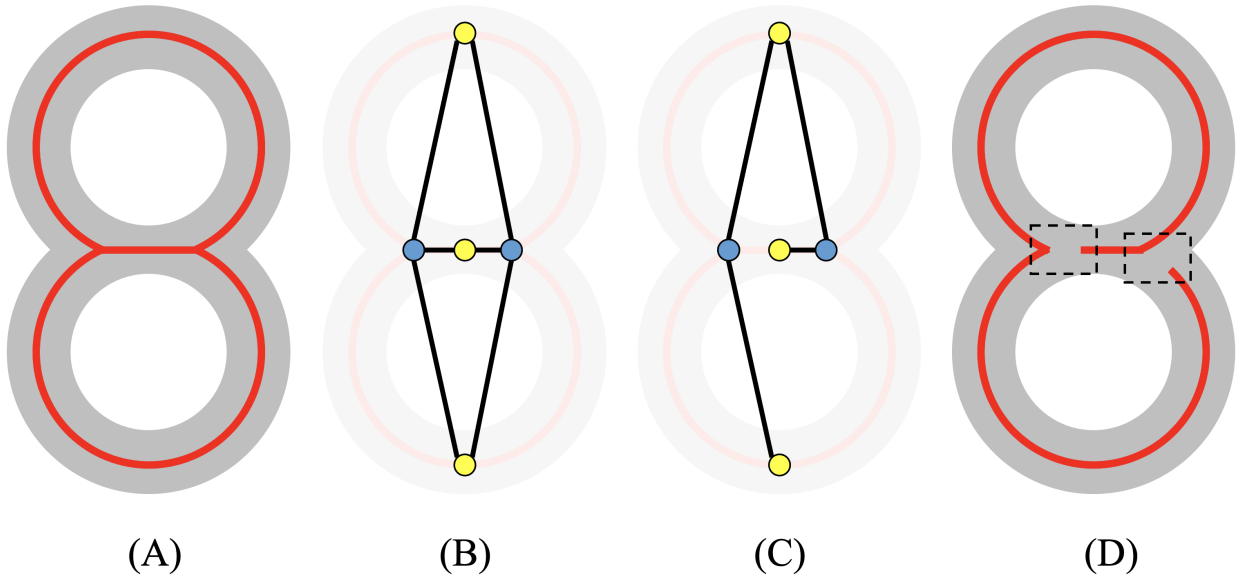

Figure 3.5: Illustration of the graph-based algorithm for cycle removal. (A) A synthetic segmentation (gray) and its skeleton with two cycles (red). (B) A graph G where each node represents either a skeleton junction (blue) or a skeleton branch (yellow) and each arc (black) connects two nodes representing a junction and an adjacent branch. (C) A spanning tree of G excludes two arcs. (D) The resulting skeleton after detaching two junction-branch pairs (dashed boxes) corresponding to the excluded arcs in C. Note that the spanning tree is not unique, and the one shown in C is just an example and not necessarily the optimal one.

The remaining topological features on the segmentation B manifest as disconnected components and cycles on the initial skeleton S_0. Figure 3.4B shows an example of a cycle caused by two touching roots. To remove these features, we take the largest component

of S_0, denoted by S_1, and remove cycles in S_1 using a graph-based approach. We first construct a graph $G = \{N, A\}$ with nodes N and arcs A as follows: each node represents either a junction (a vertex incident to three or more skeleton edges) or a branch (a sequence of skeleton edges between two junctions or between a junction and an end vertex) on S_1, and each arc connects two nodes representing a junction and a branch at that junction (see Fig. 3.5A, B). Next, we extract a spanning tree of G (see Fig. 3.5C). The final skeleton, S, is then obtained from $A^{'}$ as follows: for each arc $a \in A$ that does not exist in $A^{'}$, the pair of skeleton branch and junction presented by a is "detached" from each other (see Fig. 3.5D). Note that cycle-removal using this approach prevents a skeleton branch from being broken in the middle.

To find a spanning tree that best captures the structure of the root, we associate each arc $a \in A$ with a positive weight $w(a)$ and compute the spanning tree $A^{'}$ with the minimal total arc weights. $A^{'}$ is known as the minimal spanning tree (MST) of G, and it can be computed efficiently using standard algorithms such as Prim's or Kruskal's. The weight $w(a)$ measures the likelihood that the pair of skeleton junction and branch represented by a should be detached. It is defined as:

$$w(a) = w_{angle}(a) + \lambda w_{dis}(a) \tag{3.2}$$

The first term $w_{angle}(a)$ measures the continuity of skeleton orientation at the junction. Let j,b be the skeleton junction and branch represented by the arc a, ω be the set of all branches at j, and $\vec{b}$ be the unit vector representing the tangent direction of the branch b oriented towards j. The angle term is defined as:

$$w_{angle}(a) = \min_{b' \in \Omega}(1 + \vec{b} \cdot \vec{b'}) \tag{3.3}$$

This term reaches the minimum of 0 if there is some other branch $b^{'}$ at junction j that has the same tangent direction as b. The second term $w_{dis}(a)$ measures the distance from the

junction j represented by the arc a to the root stem. This term exists to discourage detaching branches representing nodal roots from the stem, which would have a great impact on the branching hierarchy. We use the algorithm reported in [14] to identify the stem as the longest non-branching sequence of skeleton edges on the skeleton S_1 whose thickness measure is above a given threshold. Figure 3.4C shows the skeleton obtained from the MST of the weighted graph. Observe that the touching between two roots in the highlighted view are correctly detached.

3.4.3 Inferring hierarchy

Given the cycle-free skeleton S obtained from the previous step, we next label each vertex of the skeleton as either part of the stem, a nodal root, or a lateral root of a specific order. This results in a branching hierarchy denoted as H. The hierarchy plays a key role in the final step to obtain traits concerning each type of roots.

We first identify the path of skeleton edges capturing the stem using the heuristic of [14], as already done in the previous step. This path is called a stem path. We then consider all skeleton edges within a cylindrical region around the stem path, where the radius of the cylinder varies along the stem path and is set to be 1.6 times the thickness measure stored on the skeleton vertices on the path. These skeleton edges make up the stem region. The factor 1.6 was chosen empirically to ensure that all skeleton junctions representing the beginning of nodal roots are included in the stem region. Both stem path and stem region are labelled as hierarchy level 0.

To identify nodal roots and lateral roots of different orders, we make the following two assumptions, which are shared by DynamicRoots and have been supported by studies of root systems [12, 64, 65]. First, roots higher up in the hierarchy are generally longer. For example, nodal roots are generally longer than 1st-order lateral roots, which in turn are generally longer than 2nd-order lateral roots, and so on. Second, the maximum number of hierarchy levels in a root system is generally kept low. With these assumptions, we developed

a heuristic that minimizes the depth of the hierarchy while favoring longer roots higher up in the hierarchy.

Our heuristic proceeds in two stages, a bottom-up traversal of the skeleton and then a top-down traversal. They are illustrated in Fig. 3.6 using a cartoon example. We start with a skeleton S labelled only by the stem region (Fig. 3.6A). Recall that a branch is a sequence of skeleton edges between two junctions or between a junction and an end vertex. Since S has no cycles, it is a "tree". We consider the stem region as the "root" of this tree, and this induces a partial ordering on the skeleton such that each junction (outside the stem region) is incident to exactly one parent branch and two or more children branches. The first stage of the heuristic computes, at each skeleton junction, the association between the parent branch with one of the children branches as the continuation of the same root (see arrows in Fig. 3.6B). The association is computed by visiting the skeleton branches from the leaves of the skeleton tree towards the stem and updating a depth d(b) (numbers in Fig. 3.6B) and a distance $l(b)$ at each visited branch b, as follows. First, for each branch b incident to an end vertex, we set $d(b) = 0$ and $l(b)$ as the length of b. We then iteratively visit parent branches whose children branches have already been visited. For a parent branch b whose children are $b_1, \ldots, b_n$, we associate b with the child b_i that has the maximal depth $d(b_i)$. If multiple children have the same maximal depth, b is associated with the b_i with maximal length $l(b_i)$. We then set $d(b)$=$d(b_i)$+1 and $l(b)$ to be $l(b_i)$ plus the length of b. In the second stage, we visit all branches from the stem to the leaves and assign the hierarchy levels. We assign each branch attached to the stem region a hierarchy level of 1 (i.e., nodal roots). For each parent branch assigned with level k, we assign level k to the child branch associated with the parent (computed from the first stage) and level k+1 to all other children branches. The resulting hierarchy labelling is shown in Fig. 3.6C.

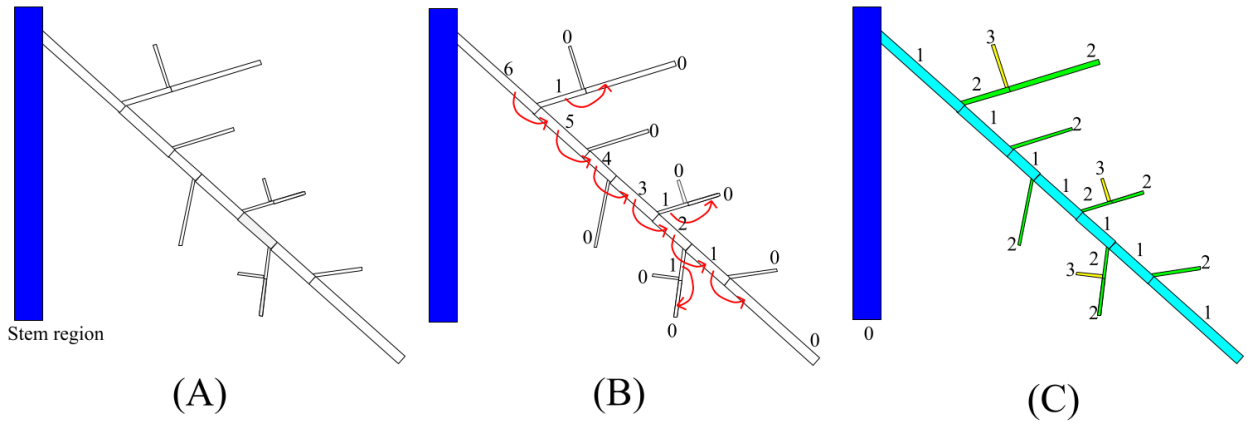

Figure 3.6: Illustration of the heuristic for inferring hierarchy. (A) The input skeleton S with only the stem region labelled (blue). (B) The first stage associates each parent branch with one of its children branches (indicated by arrows). The numbers are the depth $d(b)$ stored at each branch b, an intermediate quantity used to determine the parent–child association. (C) The second stage assigns hierarchy levels (shown as numbers and grouped by colors) to each skeleton branch segment based on the parent–child association. These level labels make up the hierarchy H

3.4.4 Computing traits

Given the skeleton S and the hierarchy labelling H, TopoRoot computes a suite of coarse-grained and fine-grained traits. Like existing works (e.g. [54]), we compute global traits which are aggregated over all roots regardless of their location in the hierarchy, including the total root length, number of roots, and average root length. For fine-grained traits, for each hierarchy level (e.g., nodal roots, 1st-order lateral roots, 2nd-order lateral roots, etc.), we compute the root count, average and total root length, average root tortuosity, average root thickness, average number of children, and the average emergence, midpoint, and tip angle. We also report the length and thickness of the stem. Some of these traits, such as stem length and per-level angle traits, have not been previously reported by existing tools (including DynamicRoots [29]). Details on how each of these traits is computed can be found in the supplementary section of the paper.

3.5 Results

3.5.1 Data preparation

TopoRoot is tested on two sets of maize root images, one consisting of 45 X-ray CT scans of excavated root crowns, and another consisting of 495 synthetic images of simulated maize root systems. For validation, we collected ground truth data of nodal root counts for the CT data set and a variety of fine-grained traits for the synthetic data set.
A cohort of 59 maize seeds were planted in June 2020 at Planthaven Farms in O'Fallon, Missouri (latitude 38.84871204483824, longitude -90.68711352048403) in silt loam soil. The cohort consists of both wild-types (Rt1-2.4 WT) and mutants (Rt1-2.4 MUT) with mutation on the Rootless1 gene, which are known to have decreased nodal root counts [66]. Seeds were planted using jabtype planters into 3.65-m long rows (25.4-cm within row spacing) on 0.9144-m row-to-row spacing. Fields received fertilization with ammonium nitrate. After 54–57 days of growth, the roots were excavated using the Shovelomics protocol [26] in September 2020 and washed to remove large soil chunks (the impact of soil on our method is discussed in the Discussion section). An X5000 X-ray imaging system and efX-DR software (NSI, Rogers, MN) were used to collect X-ray computed tomography (XRT) data (see in Fig. 3.7). The X-ray source was set to a voltage of 70 kV, current of 1700μA, and focal spot length of 119 μm. Samples were clamped and placed on a turntable for imaging at a magnification of 1.17X and 10 frames per second (fps), collecting 1800 16-bit digital radiographs over a 3 min scan time. efX-CT software was used to reconstruct the scan into a 3D image at 109 μm voxel resolution. This 3D image was exported as a 16-bit RAW file, then ImageJ was used to store the RAW file as a stack of 2D image slices perpendicular to the z-axis. Finally, 14 3D images were removed from the analysis due to excessive soil present in the imaging or missing whorls, resulting in a total of 45 3D images or validating TopoRoot. We performed manual counting of nodal roots

for each of the 45 root crowns. Each sample was dissected starting at the highest node (stalk end) moving downward to the root tips. Only attached roots were counted towards the total number of developed roots at each node.

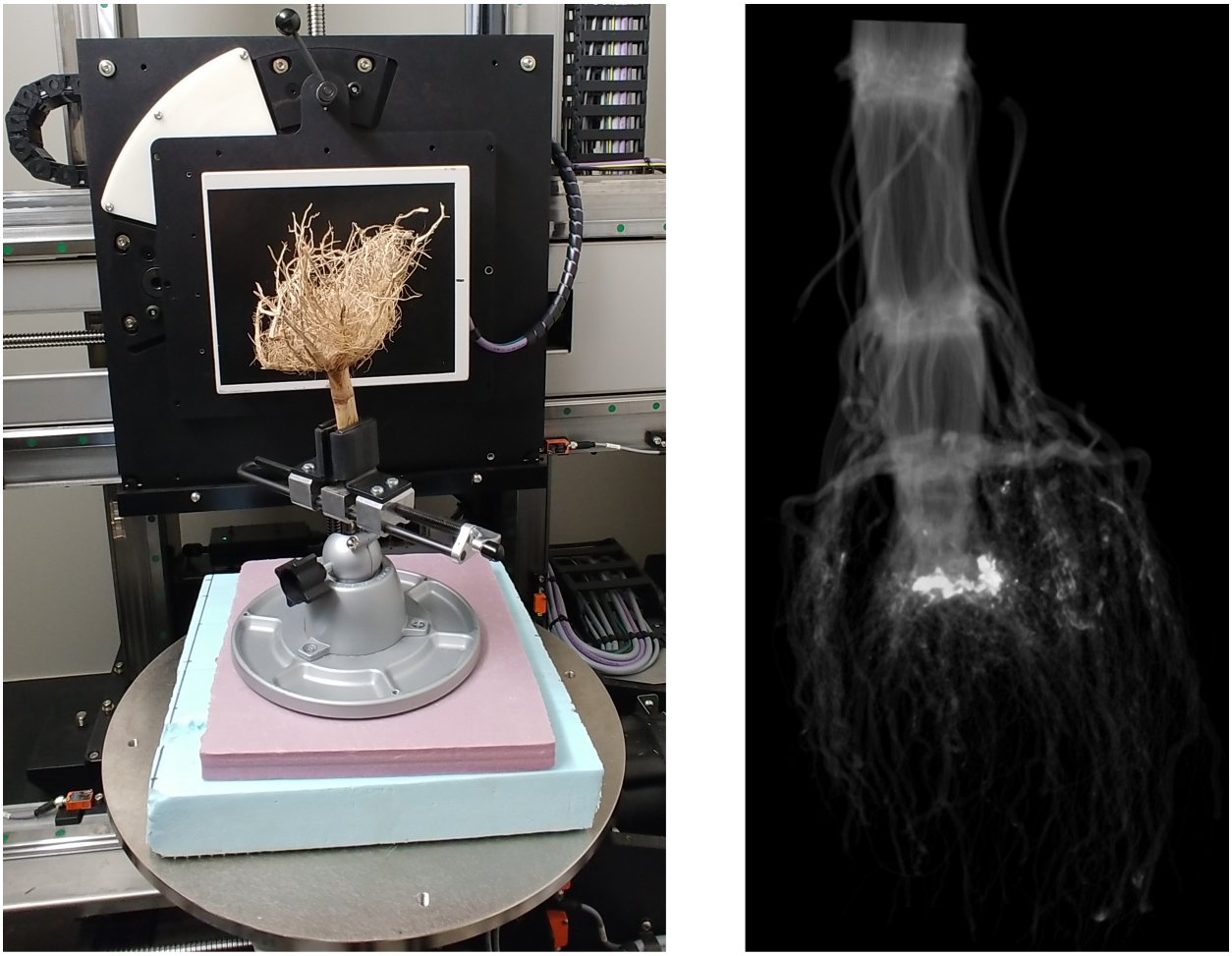

Figure 3.7: X-ray CT imaging of root crowns. (A) Each maize root crown was clamped and placed on a turntable, which was rotated for 3 min while radiographs were collected at a rate of 10 frames per second. (B) efX-CT software was used to reconstruct a 3D grayscale image from the scan, which is represented as a stack of image slices perpendicular to the z-axis

It is generally difficult to obtain manual measurement of fine-grained root traits beyond counting the nodal roots. To validate other fine-grained traits produced by our method, we supplement the CT data set with a large benchmark of synthetically generated root

images. We adopt OpenSimRoot [64], a highly customizable 3D root growth simulation software that has been widely used in modeling and visualizing root growth [67, 68]. We used OpenSimRoot to create 55 maize root systems ranging in days of growth from 30 to 40 days, numbers of nodal roots ranging from 31 to 69, number of whorls from 5 to 6, and lateral root branching frequency from 0.3 to 0.7 cm / branch. The diameter of the stem was set to be 2 cm, starting diameter for nodal roots is 0.3 cm (gradually decreasing to 0.1 cm after 10 days of growth), lateral roots is 0.04 cm, and fine lateral roots is 0.02 cm. OpenSimRoot provides a detailed hierarchy for each of the simulated roots, from which we obtain the ground-truth traits (roots less than one voxel long in the ground truth model were excluded). For each simulated root system, we synthesize a 512^3 image by computing the signed distance field from the surface of the root using the method of [69] with the inside of the surface having positive values and the outside having negative values. To mimic various levels of image noise, we randomly perturb the distance value at each voxel, with the amount of perturbation ranging from 0 to 0.08 cm in 0.01 increments. This results in 9 images at increasing noise levels for each of the 55 roots, and thus 495 images in total. Figure 3.8 shows images of one simulated root (at day 40) synthesized at different noise levels. Note that the amount of geometric irregularity and topological noise (e.g., disconnected components and loops) increase with the noise level.

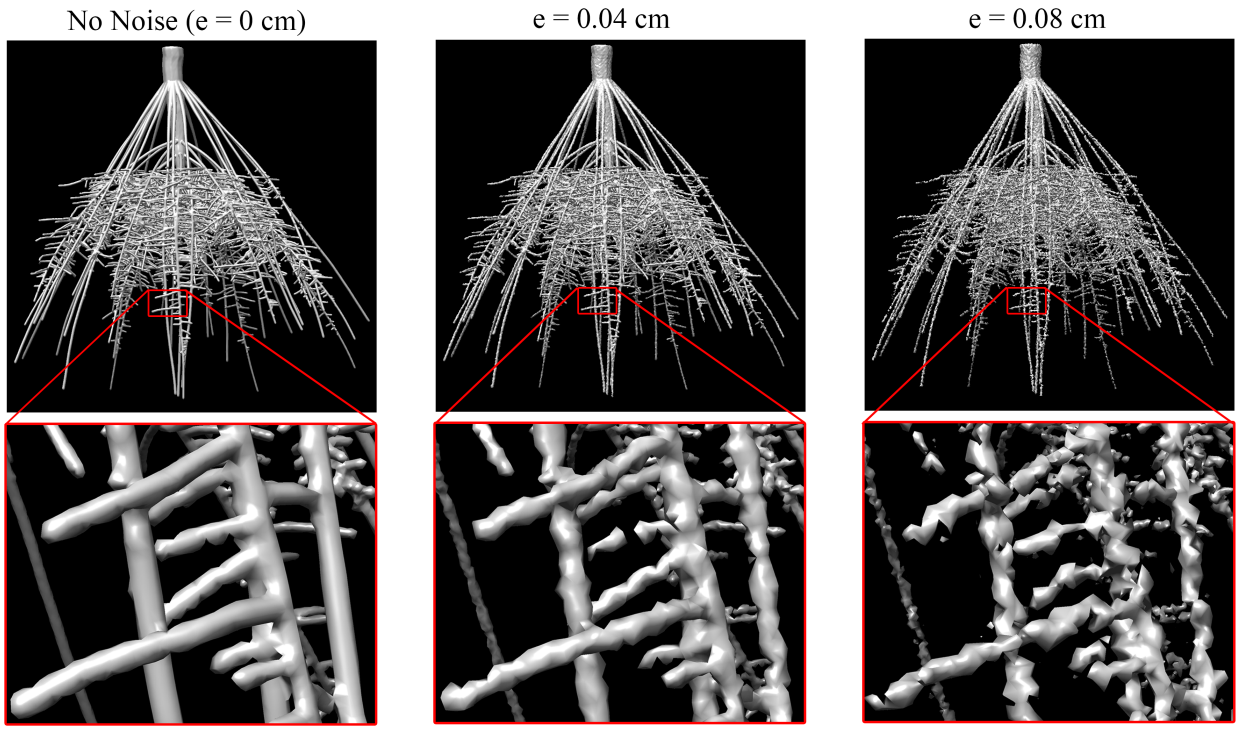

Figure 3.8: Synthetic maize root images with increasing amounts of noise (e) generated from a simulated root system at 40 days of growth. The closeups show fine lateral roots. With increased noise, the roots exhibit less regular geometry and more topological errors (e.g., disconnections and loops)

3.5.2 Experiment settings

The only parameters of TopoRoot that need to be individually tuned for each input image are the three thresholds used in the segmentation step. As explained earlier, t_{mid} should be chosen to best capture the overall root shape, t_{low} should be low enough to connect most of the roots, and t_{high} should be high enough to prevent roots from touching. Due to the varying contrast in the CT images, the thresholds tend to vary as well (the values are included in our online data distribution). For the synthetic images, we found that setting t_{low}=0.15,t_{mid}=0,t_{high}=0.03 works well on all images.

We compared TopoRoot with two previous tools, DynamicRoots [16] (for both global and fine-grained traits) and a 3D version of GiaRoots [54] first published in [27] (for global

traits only). We used default parameters for both tools. Since both tools take in a binary segmentation, we ran them after thresholding each image at the threshold t_{mid}. To study the impact of topological errors on these tools, we also experimented with first performing the segmentation step of TopoRoot and then running DynamicRoots or GiaRoots on the topologically simplified segmentation (instead of naive thresholding at t_{mid}). We call the new protocols DynamicRoots+ and GiaRoots+ respectively.

3.5.3 Experimental results: excavated root crowns

Figure 3.9 visually compares the root hierarchies computed by TopoRoot, DynamicRoots, and DynamicRoots+ on an example root crown. DynamicRoots produces a point cloud where each point represents an input voxel and is labelled by its hierarchy level (0, 1, 2, etc.). Observe DynamicRoots mis-labelled many roots, such as those highlighted in the black boxes. This is primarily since the methodology of DynamicRoots is designed for much less complex, seedling-stage roots. In addition, a significant portion of the root is missing, as highlighted by the red box. This is because DynamicRoots takes in the naively thresholded image, which has many disconnected parts. Although performing the topological simplification step in TopoRoot allows DynamicRoots+ to recover a more complete root shape, mislabelling of hierarchy levels remains (black boxes). In contrast, TopoRoot produces a more visually plausible hierarchy separating the stem (region), nodal roots, and lateral roots.

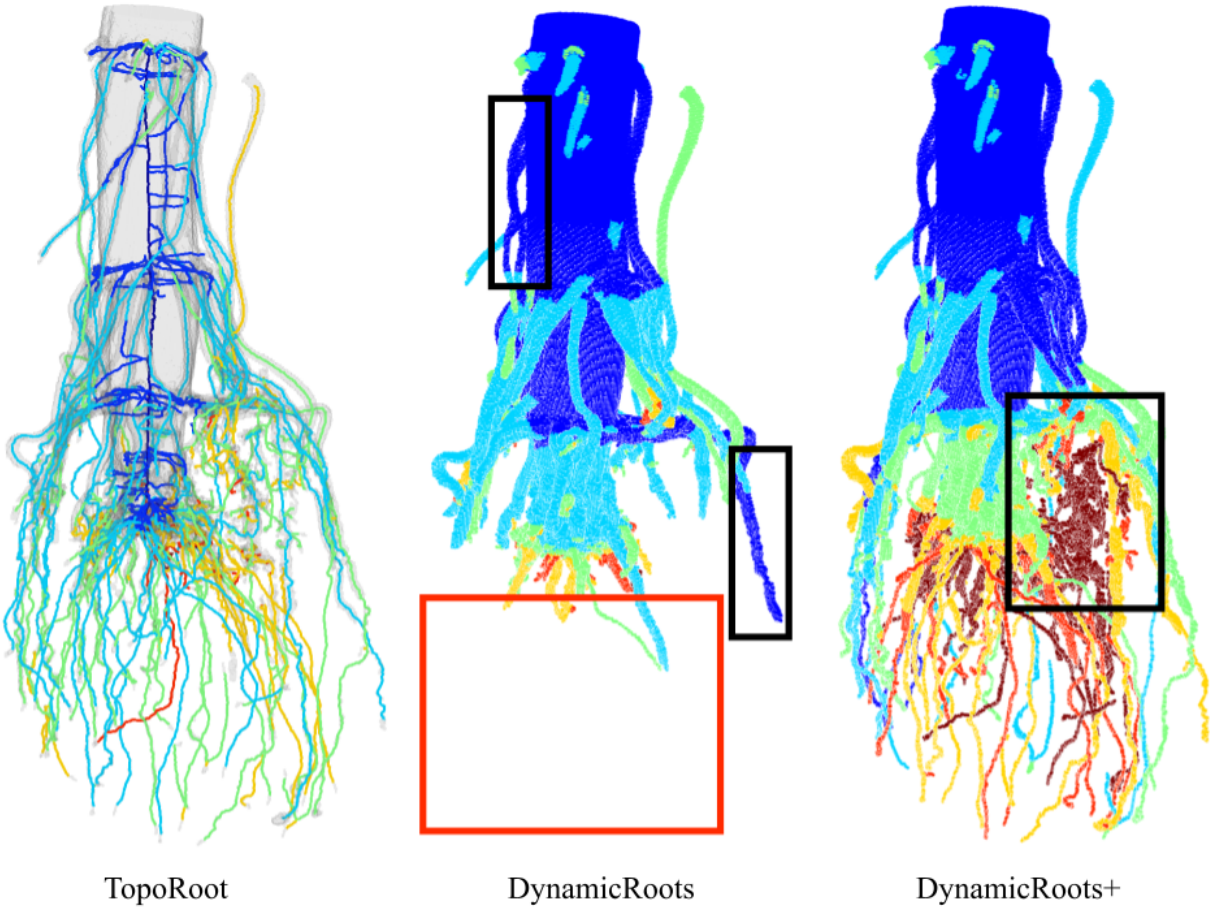

Figure 3.9: Visual comparison of root hierarchies computed by TopoRoot, DynamicRoots and DynamicRoots+ from the X-ray CT scan of an excavated maize root crown. Hierarchy levels are colored as follows: 0 (stem): dark blue, 1 (nodal roots): light blue, 2 (1st-order lateral roots): green, 3 (2nd-order lateral roots): orange, 4 (3rd-order lateral roots): red,$\geq$ 5: dark red. Black boxes highlight incorrect levels obtained by DynamicRoots and DynamicRoots+, and the red box highlights a missing component in DynamicRoots

We next compare the nodal root count obtained by various tools and by hand measurement. Figure 3.10 plots the per-sample hand-measured nodal root counts and those computed by TopoRoot (A), DynamicRoots (B) and DynamicRoots+(C) for all 45 samples. The plots also report Pearson's correlation coefficients (ρ) and the normalized root mean squared errors (RMSEn). TopoRoot exhibits a much higher correlation (ρ=0.951) and lower error (RMSEn=0.141) than either DynamicRoots (ρ=$-$0.0452, RMSEn=2.637) or DynamicRoots+(ρ=0.160, RMSEn=3.090). We also computed the per-sample relative er-

ror of a tool as the ratio of the difference between the tool's and the hand measurements over the hand measurement, and found that TopoRoot exhibits a much lower mean and standard deviation (σ) of the relative error (mean=8.3%, σ=5.6%) than either DynamicRoots (mean=159.5%, σ=190.0%) or DynamicRoots+(mean=235.4%, σ=244.4%). The significant over-counting of DynamicRoots+ is mostly caused by the mislabeling of nodal roots as level-0 roots, as shown in Fig. 3.9, which leads to many lateral roots being labelled as level-1 roots. Furthermore, both the nodal root counts computed by TopoRoot and the hand measurements exhibited a significant difference between the mutant and wild type samples, as measured by the independent two-sided Wilcoxon rank sum test (p=0.00013 for TopoRoot, p=0.00349 for hand measurements). Neither DynamicRoots (p=0.126) nor DynamicRoots+(p=0.0199) showed a significant difference between the mutant and wild-type. This shows that TopoRoot can be useful for differentiating the root system architecture between these two varieties.

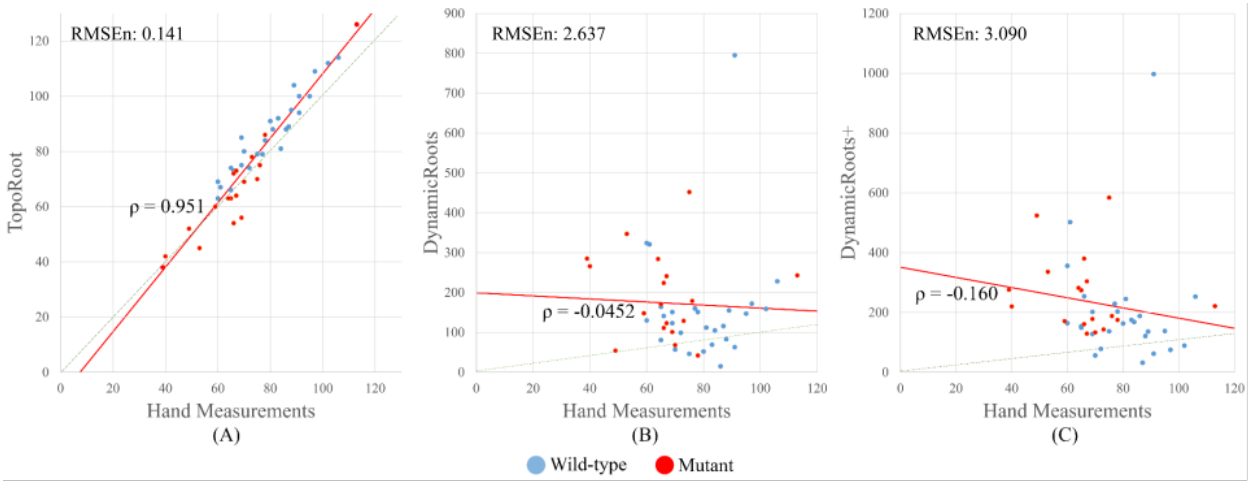

Figure 3.10: Correlation plots of nodal root count between hand measurements and those obtained by TopoRoot (A), DynamicRoots (B), and DynamicRoots+(C). Blue and red dots indicate wild type and mutant samples. The regression line is in red, and the dashed green line indicates the ideal correspondence between the two measurements. Also reported are Pearson's correlation coefficients (ρ) and the normalized root mean squared errors (RMSEn)

3.5.4 Experimental results: simulated roots

Figure 3.11 visually compares the root hierarchies produced by TopoRoot and DynamicRoots/DynamicRoots+ as well as the voxelized skeletons produced by GiaRoots/GiaRoots+ on three synthetic root images at different noise levels (0, 0.04 cm, 0.08 cm). This root system is simulated to be 34 days old, with five whorls, 34 nodal roots, and a lateral root branching frequency between 0.3 and 0.7 cm / branch. As the noise level increases, DynamicRoots and GiaRoots miss more root parts, whereas TopoRoot as well as the extended protocols, DynamicRoots+ and GiaRoots+, retain much of the root shape. Observe that, like the CT dataset, the hierarchies produced by DynamicRoots+ incorrectly label many nodal roots as level 0 (black boxes). In contrast, the hierarchies produced by TopoRoot are more visually plausible.

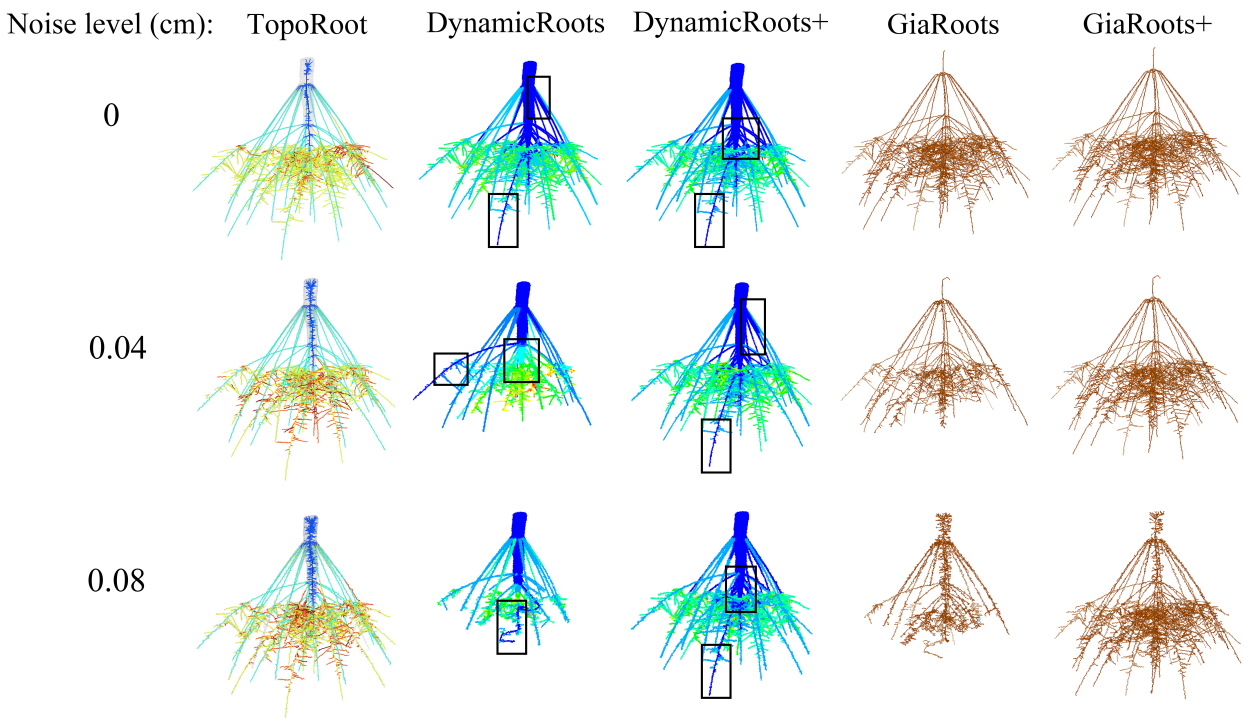

Figure 3.11: Comparing hierarchies and skeletons computed by different tools from images synthesized at increasing noise levels from a simulated maize root. Hierarchy levels 0, 1, 2, 3 and 4 produced by TopoRoot and DynamicRoots/DynamicRoots+ are colored dark blue, light blue, green, orange, and red. The voxelized skeletons produced by GiaRoots/-GiaRoots+ are colored brown. Black boxes highlight mislabeling of nodal roots as level 0 roots by DynamicRoots and DynamicRoots+

We report the mean and deviation of the relative errors of these tools for each fine-grained or coarse-grained trait in Tables 1, 2, 3, 4 of the paper (GiaRoots/GiaRoots are considered for global traits only). In the supplementary files of the paper, we take a closer look at the accuracy of TopoRoot and the other tools as a function of the noise level of the input images. In general, we observe that higher image noise leads to larger mean errors and/or greater variance by TopoRoot. For most of the traits, TopoRoot maintains a lower error than other tools across all noise levels.

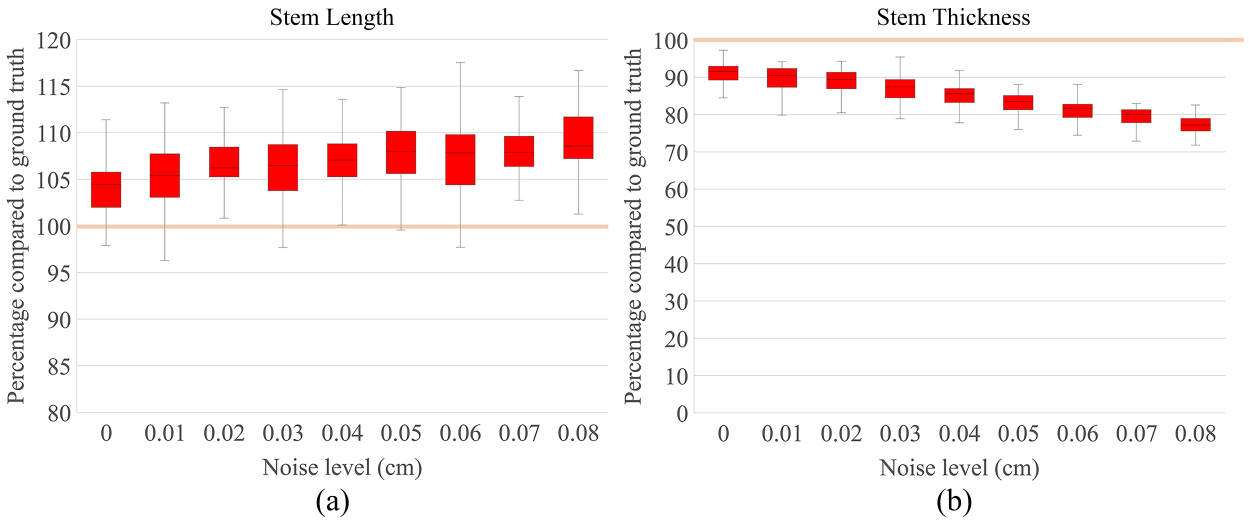

Figure 3.12: Box plots of relative errors in stem traits computed by TopoRoot. Each box shows the quantiles of relative errors over all 55 synthetic samples at each noise level.

As the base of the hierarchy, the stem traits are among the most accurate (Paper Table 1, Fig. 3.12). As the noise increases, portions of the stem region are lost, resulting in a thinner stem. Increased noise also causes the stem to wiggle more in the direction perpendicular to its main path, resulting in an increased stem length.

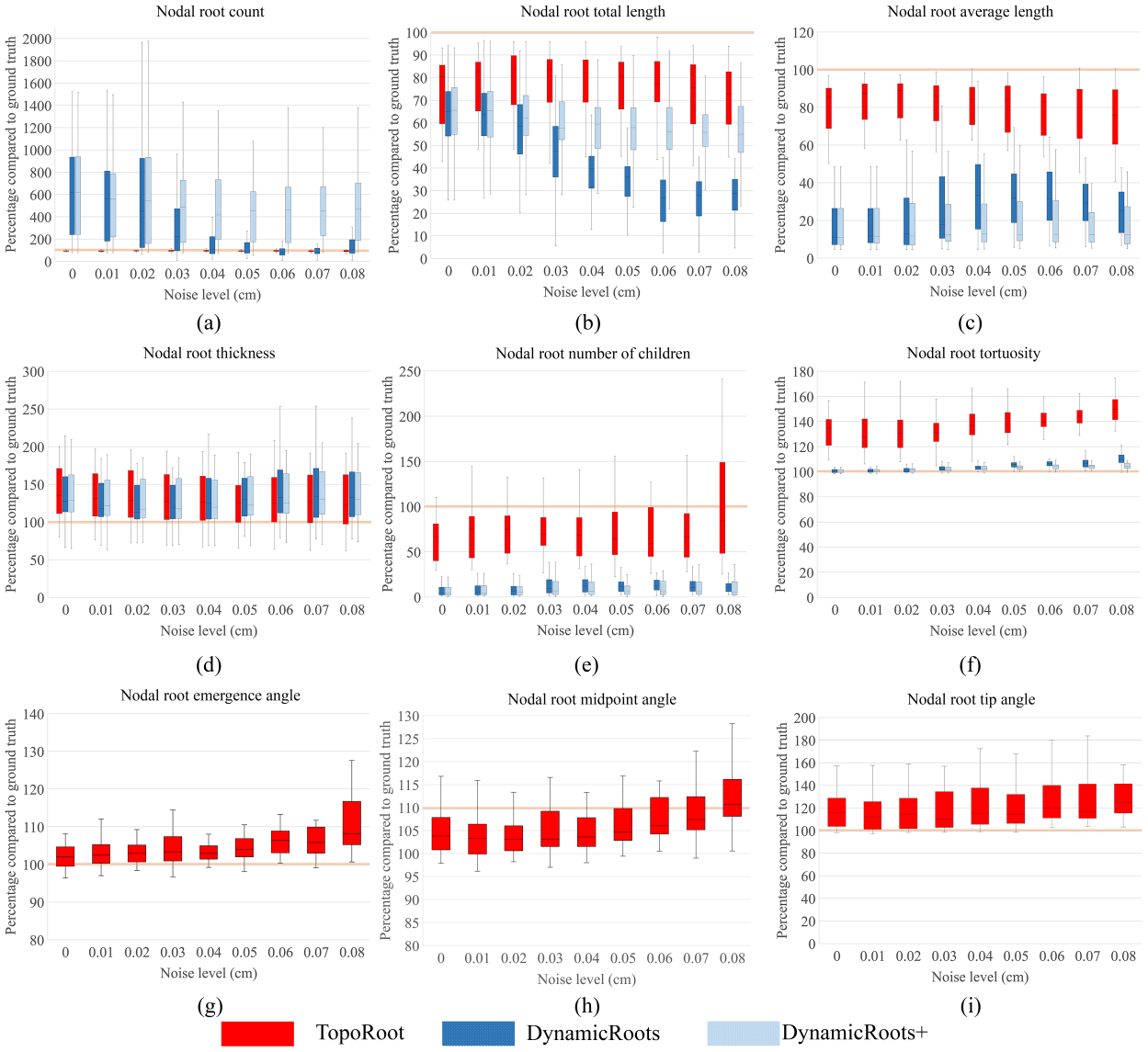

Figure 3.13: Box plots of relative errors in nodal root traits computed by TopoRoot, DynamicRoots and DynamicRoots+. Each box shows the quantiles of relative errors over all 55 synthetic samples at each noise level.

Among the nodal root traits (Paper Table 2, Fig. 3.13), the most accurate ones are the root count (mean error 8.3% up to noise level e=0.04, 10.3% up to e=0.08) and emergence/midpoint angles (mean error 5.7/7.1% up to e=0.04, 9.1/9.5% up to e=0.08). The lowest accuracy is seen for the number of children (mean error 40.0% up to e=0.04, 48.6% up to e=0.08) and thickness (mean error 39.2% up to e=0.04, and 38.2% up to e=0.08).

These errors are due to misclassifications when the nodal root becomes entangled with higher-order lateral roots. TopoRoot slightly underestimates the average and total length due to faulty cycle breaking and misclassification errors in portions of nodal roots further away from the stem. TopoRoot's error in nodal root tortuosity is higher than that of DynamicRoots for two reasons. First, the ground truth tortuosity is close to 1 (only for the simulated data, but not for the real maize roots), and DynamicRoots coincidentally produces values close to this because it mistakes many shorter lateral roots as nodal roots, as evidenced by its much shorter average nodal root length and the black boxes of Fig. 3.11. Second, nodal roots sometimes are misclassified by TopoRoot closer to their tips due to the large number of intersections between roots of different hierarchy levels, resulting in excessive winding. TopoRoot slightly overestimates angle measurements due to misclassification errors further away from the stem which bend the detected paths sideways; these explain the errors in the tip angle measurements.

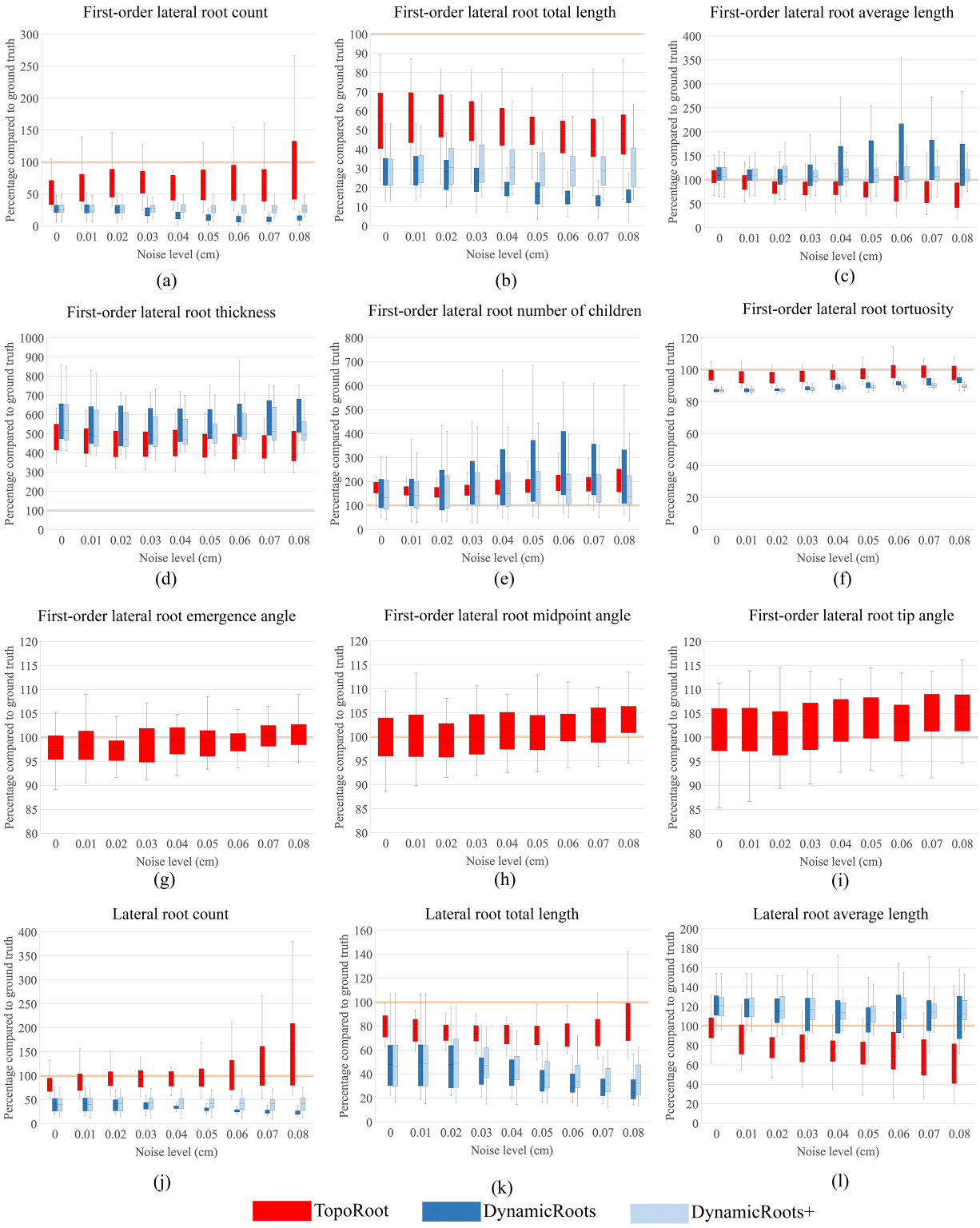

Figure 3.14: Box plots of relative errors in lateral root traits computed by TopoRoot, DynamicRoots and DynamicRoots+. Each box shows the quantiles of relative errors over all 55 synthetic samples at each noise level.

The errors for the lateral root traits (Paper Table 3) are generally larger than nodal root traits, primarily because the imaging noise has a greater impact on the thinner roots more than the thicker ones. There is a greater underestimation of both the total first-order lateral roots and their total length (Fig. 3.15), due to both the misclassification of the hierarchy levels and the loss of many thin roots in the distance field. On the other hand, the misclassified first-order lateral roots are counted as lateral roots of higher orders, and hence less errors lie in the total lateral root count (mean error 22.2% up to e=0.04 and 37.0% up to e=0.08) and lengths (mean error 24.3% up to e=0.04 and 24.4% up to e=0.08) over all orders. All methods significantly overestimate the first-order lateral root thickness due to limits in the resolution, but TopoRoot produces the lowest error. The lowest errors are seen in the first-order lateral emergence/midpoint/tip angles (mean error 3.4%/4.1%/5.4% up to e=0.04 and 3.0%/4.0%/5.9% up to e=0.08) and tortuosity (mean error 4.6% up to e=0.04 and 4.5% up to e=0.08).

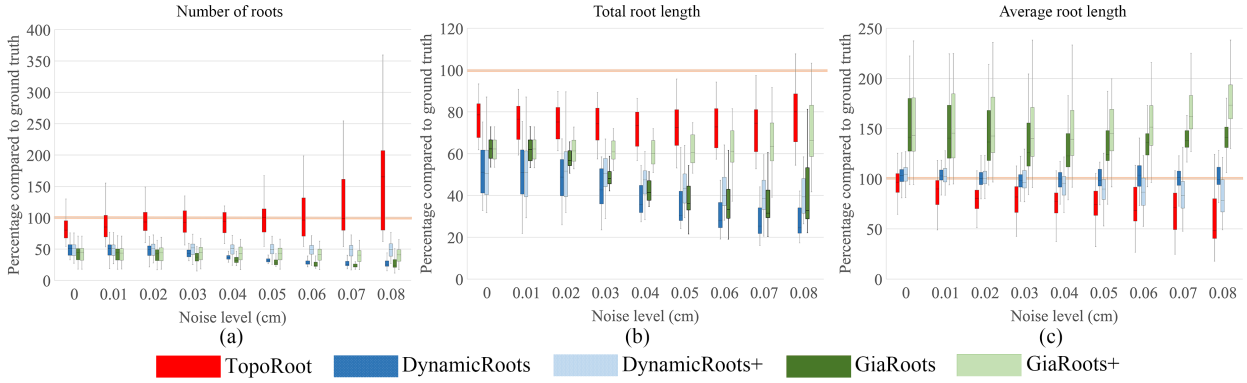

Figure 3.15: Box plots of relative errors in global root traits computed by TopoRoot, DynamicRoots, DynamicRoots+, GiaRoots and GiaRoots+. Each box shows the quantiles of relative errors over all 55 synthetic samples at each noise level.

Finally, combining nodal and lateral roots, TopoRoot produces on average 35.4% relative error (21.5% up to e=0.04) in the total root count and 25.4% relative error (25.0% up to e=0.04) in the total root length, which are much lower than DynamicRoots/DynamicRoots+ and GiaRoots/GiaRoots+ (Paper Table 4). Note that both DynamicRoots and

GiaRoots significantly underestimate the root count and total length, even after topological simplification, and the amount of underestimation generally increases with the level of noise (Fig. 3.15). The only global trait that TopoRoot does not have the lowest error is the average length, due to a combination of DynamicRoots being coincidentally closer due to its underestimation of both the total length and number of roots, and TopoRoot counting an excessive number of roots at higher noise levels. These are the same reasons why the two methods have similar lateral root average length errors.

3.6 Discussion

A gap exists in the phenotypic measure of root system architecture between fine-grained analyses that can be conducted on entire seedling root systems in laboratory settings, and much coarser global analyses available to field researchers. Since root systems are an emergent property of their many hundreds, thousands, or tens of thousands of constituent roots, this gap is a major hindrance to a comprehensive understanding of root system development, environmental interaction, and the genetics that influence these processes. In previous work, we showed that when global 3D analysis of field excavated maize root crowns was compared to 3D seedling analysis in gellan gum, genetically encoded differences were consistent despite major differences in developmental stage and the growth environment. Whereas DynamicRoots was previously developed for fine scale measurements of 3D seedling root systems containing dozens to hundreds of roots, no similar tool existed for more complex mature root crowns, containing hundreds to thousands of roots. The orders of magnitude of increased complexity motivated unique solutions using both state-of-the-art techniques in computer graphics [20,21,22] and novel algorithms which eventually led to the development of TopoRoot.

3.6.1 Error analysis

The steps of our pipeline that are most prone to errors are segmentation and skeletonization. While the algorithm of [20] is very effective in reducing the topological complexity, it may occasionally do so at the cost of altering the structure of the roots (e.g., by breaking a thin root or merging two nearby roots), thereby introducing errors in the hierarchy and fine-grained traits. The problem can be alleviated by pushing thresholds T_{low}, T_{high} closer to T_{mid}, thus limiting the amount of changes that the algorithm can make. This solution, however, will increase the number of topological errors in the segmentation that need to be resolved in the skeletonization stage. On the other hand, our method for extracting a connected and cycle-free skeleton has several limitations itself. First, we only consider the largest connected component of the skeleton and hence would miss any root parts that are not connected to the main roots in the segmentation stage. Second, our hand-crafted weights (w(a)) on the graph arcs in the cycle-removal algorithm may not correctly distinguish between real and false junctions, and as a result the algorithm may detach branches from junctions in the wrong places. Third, the MST formulation of cycle removal cannot recover branches that are already broken in the segmentation stage. These and other limitations all lead to downstream errors in hierarchy and trait analysis. Improving the accuracy of these two steps calls for development of more robust and shape-aware algorithms for topological simplification and tree extraction from a skeleton.

Our method can be sensitive to the amount of soil attached to the roots being imaged. The method works well when the roots are reasonably clean and free of large dirt chunks. Small dirt particles are expected to remain after washing, and our method is designed to handle images containing localized topological noise caused by the dirt particles. If the root contains large chunks of dirt, and due to the similarity in intensity between the dirt and roots, a naive segmentation method (such as thresholding) may create many false and/or cluttered roots. Our method cannot fix these large-scale errors and will produce

incorrect hierarchies and traits. Besides more thorough cleaning, an alternative solution is to employ advanced image segmentation methods that are capable of separating soil from roots prior to the application of our method.

3.6.2 Running time

On average, TopoRoot completes in 7 min and 13 s for each sample in the CT scan dataset (downsampled by a factor of 4 to the resolution of 369×369×465). Since this is much shorter than the time spent imaging and reconstructing one sample, TopoRoot is well suited for high-throughput analysis. The computation time is dominated by the first two steps, topological simplification (3 min and 6 s) and skeletonization (3 min and 44 s). The time complexity of both these steps may increase quickly with the image resolution. For example, running TopoRoot on the original CT images downsampled by a factor of 2, which results in 3D images of resolution 737×737×931, would take 63 min and 39 s, with 32 min and 25 s spent on topological simplification and 29 min and 9 s on skeletonization. On the other hand, we have not observed a notable improvement in the accuracy of the nodal root count for this data set with the increased image resolution.

3.6.3 Extensions

In addition to the per-level traits reported in this work, the hierarchy obtained by TopoRoot potentially enables computation of other fine-grained traits, such as inter-whorl distances, per-whorl measurements, and the numbers of nodal roots above and below the soil. Preliminary experiments show promising results of whorl detection by clustering the nodal roots branches along the stem path of the skeleton. The soil plane can be potentially identified by the emergence of a large cluster of 1st-order lateral roots along the direction of the stem.

While TopoRoot is designed for high-contrast images (e.g., CT scans) of maize roots, it can be adapted to other types of root systems and images. For root crowns with multi-

ple tillers (e.g., sorghum), we offer a mode of TopoRoot which extends the stem-detection heuristic (during the skeletonization step) by producing a stem path within each region of the skeleton above a given thickness threshold. Preliminary visual experiments show that TopoRoot's multiple-tiller mode produces plausible hierarchies at a quality like that seen in the single-tiller mode (Fig. 3.16). Further expanding the stem-detection heuristic to identify the primary root would make the pipeline applicable to taprooted systems as well. Finally, TopoRoot can be extended to work on 3D images that lack a sufficiently high contrast between the roots and the surroundings (e.g., soil), such as in situ imaging of growing roots, provided that the roots can be segmented from its surroundings using a third-party image segmentation algorithm. Examples of such algorithms include region-growing [70], tracking tubular features [19, 71], deep learning [72, 23], and semi-automatic annotation [53, 24]. Some of these methods (e.g., deep learning) produce a probability density field, which can be fed into TopoRoot as the input 3D grayscale image. Others (e.g., region growing) produce a binary 3D image, and TopoRoot can be applied after converting the binary 3D image into a Euclidean distance field (e.g., using [69]).

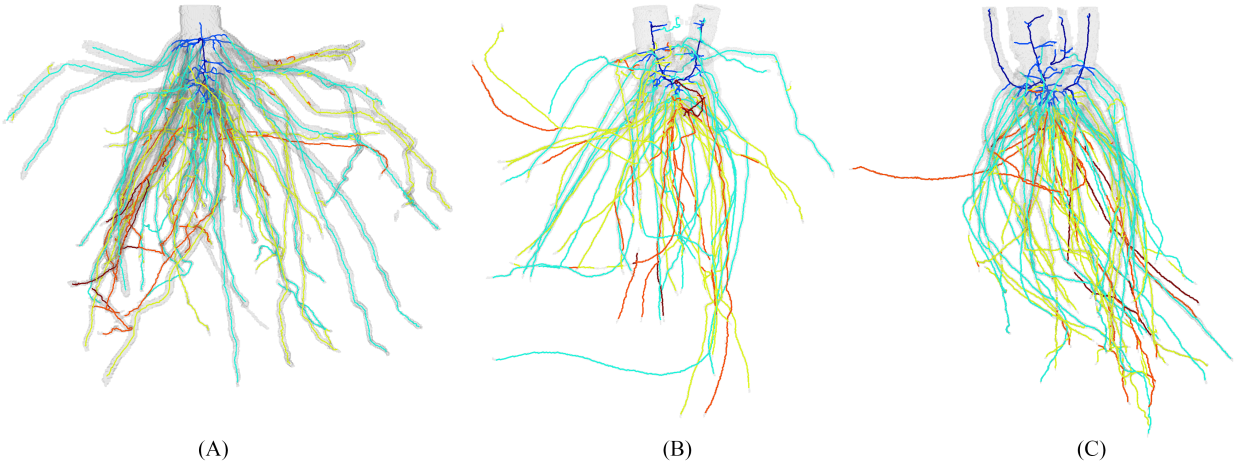

Figure 3.16: Hierarchies of sorghum roots computed by TopoRoot, showing one tiller (A), two tillers (B), and four tillers (C). Hierarchy levels 0, 1, 2, 3 and 4 are colored dark blue, light blue, green, orange, and red.

3.7 Conclusions

We introduced TopoRoot, a high-throughput method for computing the hierarchies and fine-grained traits from 3D images of maize roots. TopoRoot specifically addresses topological errors, which are common in segmenting 3D images and are barriers for obtaining accurate root hierarchies. Our method combines state-of-the-art methods developed in computer graphics with customized heuristics to compute a wide variety of traits at each level of the root hierarchy. When tested on both 3D scans of excavated maize root crowns and synthetic images of simulated root systems with artificially added noise, TopoRoot exhibits superior accuracy over existing tools (DynamicRoots and GiaRoots) in both coarse-grained and fine-grained traits. Furthermore, the efficiency and automation of TopoRoot makes it suited for a high-throughput analysis pipeline. The results are readily compatible with the Root System Markup Language (RSML; [73]), and major plant structural–functional modelling frameworks such as CRootBox [74] and OpenSimRoot [64].

Chapter 4: Nested Shape Simplification

4.1 Introduction

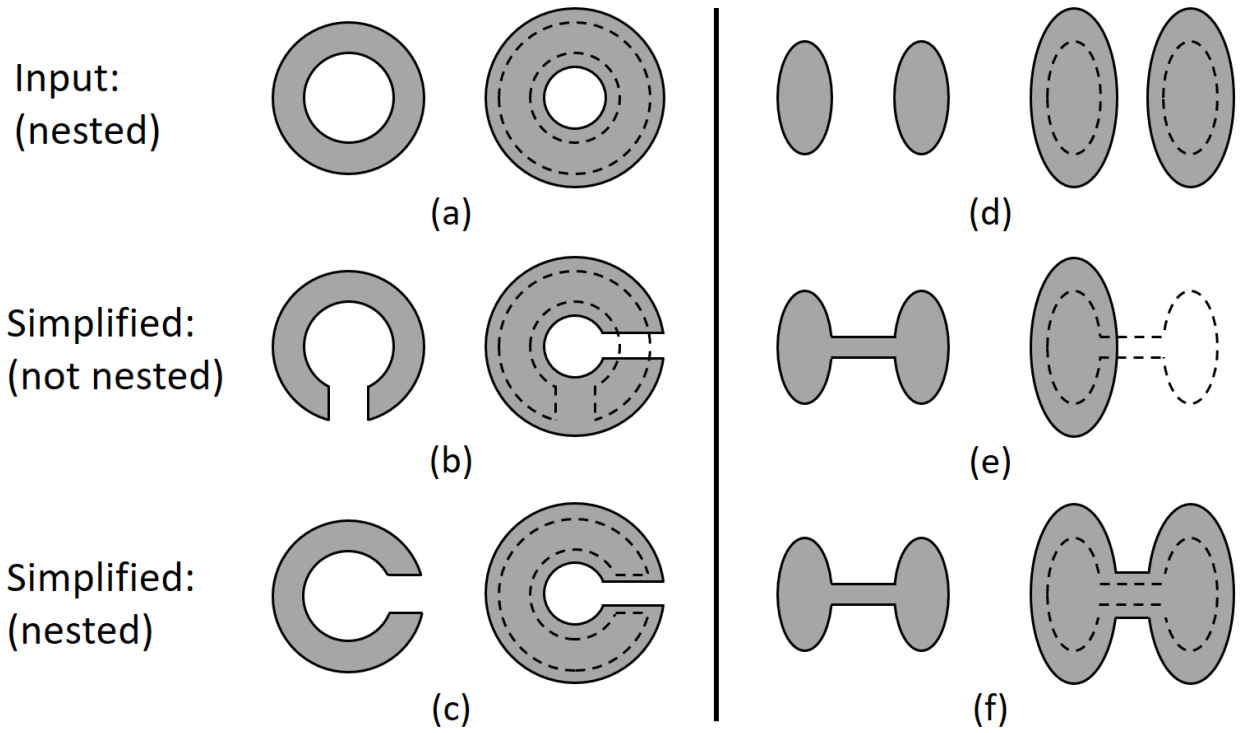

Figure 4.1: Two examples (left and right) of a pair of nested shapes containing topological features (top), and after topological simplification that violates (middle) or preserves (bottom) nesting. The outline of the first shape in each pair is overlayed on the second.

As seen in the first part of this dissertation, topological simplification is the problem of reducing unwanted topological features from 3D shapes. These features include islands (i.e., connected components), cavities (i.e., voids inside the shape), and handles (i.e., loops). These artifacts are typically resulted from reconstructing the shape from noisy or incomplete raw inputs, such as images or point clouds. Without simplification, spurious topological features may present significant challenges for many geometry processing tasks such as mesh simplification, surface parameterization, shape matching, and physical simulations. Many methods have been proposed to simplify the topology of a single 3D shape. How-

ever, for shapes that exist in a collection, the processing of individual shapes may be subject to additional constraints among the shapes. In this work, we consider a sequence of *nested* shapes, such that each shape is completely contained in the next shape in the sequence. One example of nested shapes is the layers of a composite material, such as coatings, deposits, and biological tissues (e.g., skin, fat, muscle), where inner layers are nested within the outer layers. Another example is the time-series of an expanding structure, such as the growing roots of a plant. For such shape sequences, maintaining the nesting relations between successive shapes is important for downstream analysis, whether it is physical simulation on a layered material or computing the time function of an expanding structure.

Performing topological simplification independently on each shape in a nesting sequence could result in shapes that are no longer nested. For example, a handle may be cut in different locations in two consecutive shapes (Figure 4.1(b)), whereas two islands may be bridged in one shape but one of them is removed in the next shape (Figure 4.1 (e)). Such violations of nesting are often found on real-world data, as shown in Figure 4.2 (middle row) which applies a recent single-shape topology simplification method [57] to a time-series of growing pennycress roots.

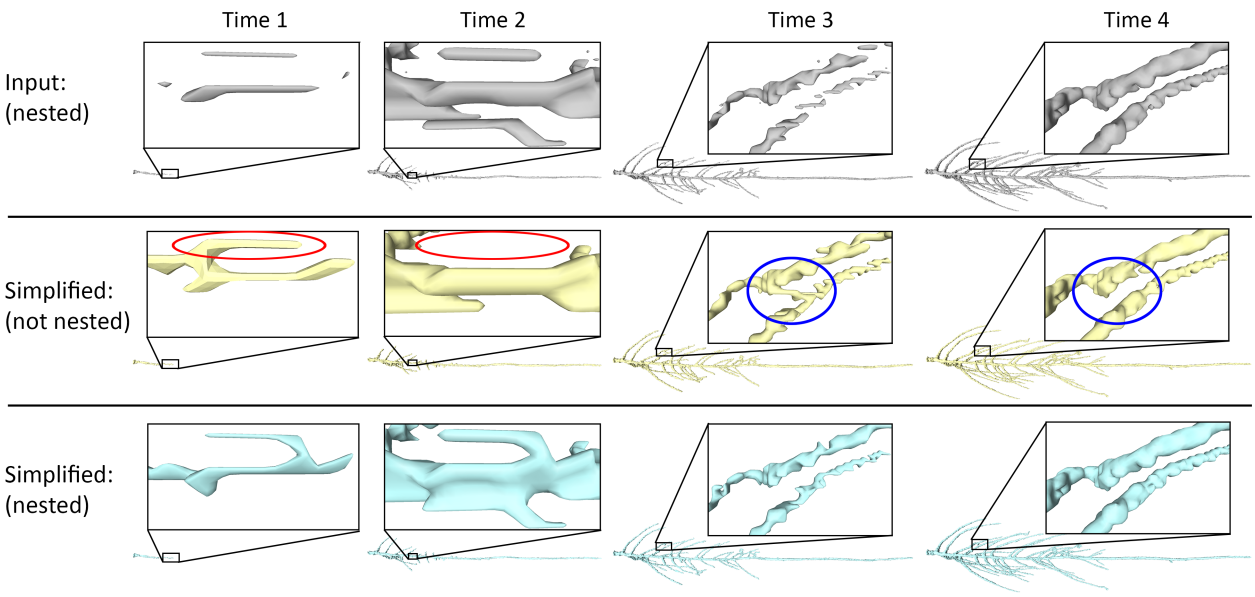

Figure 4.2: Simplifying the topology of a nested sequence of 4 shapes capturing a growing pennycress root (top row) by applying the simplification method [57] to each shape independently (middle row) and by our proposed method (bottom row). While both methods fully simplify the topology of each shape, the results of [57] are no longer nested (e.g., regions highlighted in red and blue circles) whereas ours are.

In this work, we propose a novel topological simplification method designed for nested shapes. Our method simplifies all three types of topological features (islands, handles, cavities) on each shape while maintaining their nesting relation. Our method builds upon an existing single-shape simplification method [57], which adopts a global optimization approach to maximally simplify topology while minimizing geometric changes. We make two main technical contributions:

- We formulate nested simplification as a combinatorial optimization problem (Section 4.4). Similar to [57], our formulation makes use of a set of pre-computed, candidate modifications to each shape, known as *cuts* (deletions from the shape) and *fills* (additions to the shape). Our formulation extends the one in [57] by imposing nesting constraints between shapes and computing *nesting-aware* cuts and fills that locally respect those constraints.

- We propose several strategies for solving the optimization problem (Section 4.5), including a greedy heuristic that propagates solutions from one shape to the next, a state-space search algorithm that is provably optimal, and a beam-search variant that trades off optimality for efficiency.

We tested our method on both synthetic and real-world shape sequences. We found that all optimization strategies are effective in reducing topological complexity of individual shapes while maintaining nesting. On complex data, we found that, among the three optimization strategies, the beam-search algorithm strikes the best balance between solution quality and performance.

4.2 Related Work

4.2.1 Topological simplification of shapes

Topological simplification of 3D shapes has been extensively studied in the past. While the general problem of maximal simplification within an error bound is known to be NP-hard [75], many practical heuristics have been proposed. Some methods only remove topological handles, for which a common approach is computing the Minimum Spanning Tree on a graph [38, 37, 76, 40]. Methods to remove all three types of topological features (islands, handles, and cavities) include morphological opening and closing [41], inflation or deflation from a topologically simple *seed* [43, 42, 50], or tools from persistent homology [45]. However, these above methods either only add to, or only delete from, the shape to remove its topological features. This may lead to excessive geometric changes, since some topological features are either to be cut (e.g., a thin loop) whereas others are easier to be filled (e.g., a narrow tunnel).

Methods that allow both addition and deletion often apply some local heuristics to decide where to add or delete [43, 46, 77, 48]. However, these heuristics do not consider the

global optimality of the changes. The recent work of Zeng et al. [57] tackles topological simplification as a global optimization problem. After computing a set of candidate cuts and fills, this method formulates a graph labelling problem on these candidates that aims at maximally simplifying the topology while minimizing the amount of geometric changes. The problem is then solved by a transformation to the Node-Weighted Steiner Tree (NWST) problem.

While these methods can simplify individual shapes in a collection, they do not respect any notion of consistency (e.g., nesting) among the shapes. In this regard, our method makes a first step towards consistency-constrained topological simplification of a shape collection. Our method is an extension of [57] in two ways: (1) we extend its candidate-based formulation to multiple shapes under the nesting constraint, and (2) we use its label optimization algorithm as a building-block in our own optimization.

4.2.2 Topological simplification of scalar fields

Topological abstractions, such as the Morse-Smale complex and persistence diagram, are useful for analyzing scalar fields in Topological Data Analysis (TDA) [78]. Since practical data is often noise-ridden, such noise needs to be removed from a scalar field prior to analysis. Numerical methods [79, 80, 81, 82] simplify a scalar field by maximally removing its local extrema (minima and maxima) except for those in a given constraint set. Combinatorial methods [83, 84, 85, 86] remove those extrema whose importance, given by some measure such as persistence [87], is below a threshold while maintaining an error bound. A major limitation of scalar field simplification methods is that they are generally not effective in removing saddle points in a 3D scalar field. Since saddles are responsible for topological handles on the level sets, these methods have limited utility in solving our nesting-constrained simplification problem by representing the shape sequence as level sets (see Figure 4.12).

4.3 Background

Our algorithm represents and analyzes shapes represented by cells within a cell complex, just like our work in the first part of the dissertation. We briefly review key concepts in cell complex topology relevant to the discussion of our method. Readers are referred to standard literature such as [49] for in-depth discussions on the subject.

A k-dimensional cell, or k-cell, is an open set homeomorphic to an open k-dimensional ball. A set of disjoint cells is called a *cell complex* if, for each cell in the complex, its boundary is completely covered by other lower-dimensional cells in the complex. For example, a cell complex may consist of a cube (a 3-cell), its six bounding squares (2-cells), twelve edges (1-cells), and eight vertices (0-cells). A cell x is said to be a *face* of another cell y if x lies in the boundary of y.

Given a cell complex C covering a d-dimensional space, we denote the set of all d-cells of C as C_d and call them the *top-dimensional cells*. For example, if C is a cubical or tetrahedral decomposition of space, C_d are the cubes or tetrahedra (excluding their low-dimensional faces). Our algorithm represents a shape as a subset of C_d. Strictly speaking, the top-dimensional cells are open sets and disjoint from each other. For the purpose of analyzing topology, we need to define a connected shape from them. See a 2D illustration in Figure 2.

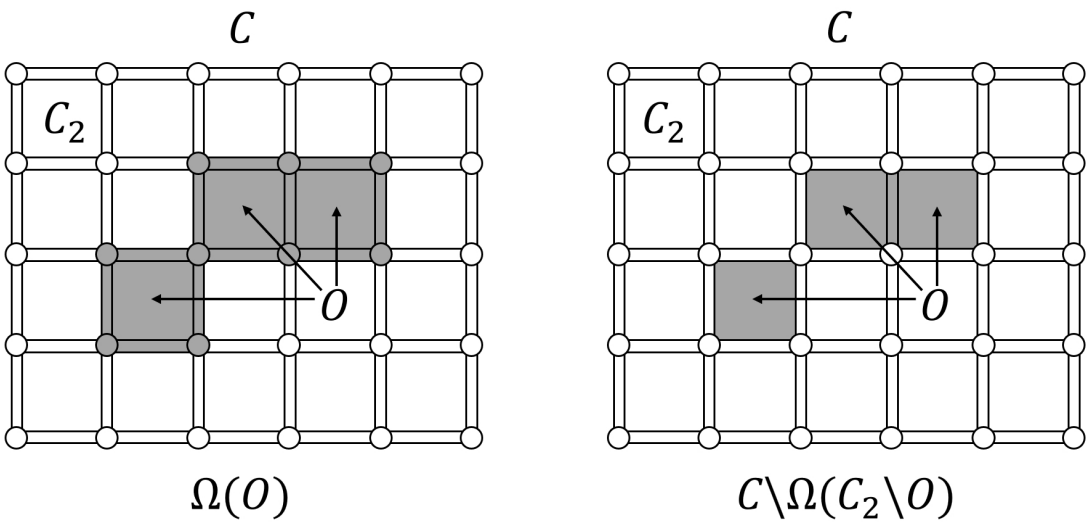

Figure 4.3: Two ways of defining a shape (shaded cells) from three 2-cells O in a 2D cell complex C. C_2 is the set of all 2-cells in C.

Adapting the approach of the work in the first part of the dissertation, a topological feature (e.g., connected component, handle or cavity) can be created or removed using one of the two types of topological surgeries, namely *cutting* (i.e., removing contents from the shape) and *filling* (i.e., adding contents to the shape).. For example, a component can be removed by either deleting the entire component (i.e., cutting) or connecting it with another component via a bridge (i.e., filling). A handle can be removed by either breaking the handle ring (i.e. cutting) or filling the handle hole. A cavity can be removed by either connecting it with the exterior by a tunnel (i.e. cutting) or filling the entire cavity. These surgeries are illustrated in 3D in Figure 3.

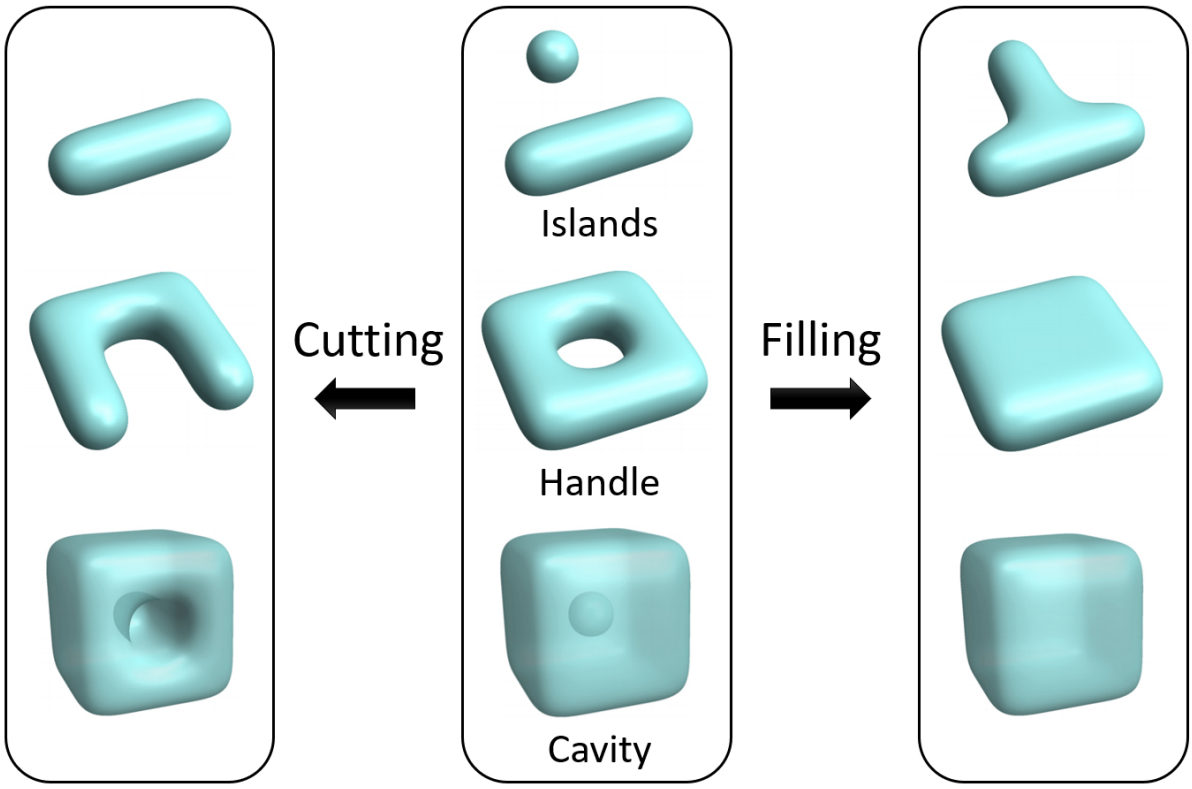

Figure 4.4: Removing islands (top), handles (middle), and cavities (bottom) by cutting or filling. Adapted from Figure 2 of [57] and courtesy of the authors.

4.4 Problem formulation

Given a nesting sequence of 2D or 3D shapes, our goal is to modify each shape in the sequence such that (1) the modified shape's topology is as simple as possible, (2) the modified shape differs from the original shape as little as possible, and (3) the modified shape sequence remains nesting.

We first present a general formulation of the problem (Section 4.4.1). To make the optimization task more trackable, and inspired by the formulation in [57], we next introduce a simplified formulation that considers only a pre-computed set of candidate cuts and fills (Section 4.4.2). Finally, we characterize candidates that are compatible with the nesting constraint (which we call *nesting-aware* candidates), develop an algorithm for finding such candidates, and further tailor the problem formulation to such candidates (Section 4.4.3).

4.4.1 General formulation

The main input to our method is a shape sequence $\{T_1, \ldots, T_n\}$ defined on a common cell complex C in d-dimensional space for $d = 2, 3$. Each T_i is a subset of the top-dimensional cells (d-cells) of C, that is, $T_i \subseteq C_d$. The sequence is nesting in the sense that $T_i \subseteq T_{i+1}$ for all $i = 1, \ldots, n-1$. Our method, in theory, can be applied to any type of cell complexes (e.g., cubical, tetrahedral, mixed-element, etc.) and either one of the two ways of shape definition from top-dimensional cells (see Figure 4.3). Our current implementation is specialized to quadrilateral (in 2D) or cubical (in 3D) cell complexes, and it uses the definition such that the connected shape represented by T_i is $C \setminus \Omega(C_d \setminus T_i)$ (Figure 4.3 (b)). This is equivalent to using 4-connectivity (in 2D) and 8-connectivity (in 3D) in digital topology. For simplicity, henceforward we shall use T_i to denote both the composing d-cells and the connected shape represented by them.

To suit different application scenarios, our method can take in two additional and optional inputs from the user. First, the user may define a per-cell *geometric cost* $g_i(c)$ for each d-cell $c \in C_d$ and each $i = 1, \ldots, n$. This cost measures the geometric change when c is added to or deleted from T_i. By default, $g_i(c)$ is set to be the area (in 2D) or volume (in 3D) of c, but it can be customized. For example, it can be weighted by an additional scalar field (e.g., confidence map or SDF) that may come with each shape. For notational simplicity, we write the sum of costs over a set of d-cells C as $g_i(C)$.

Secondly, the user may provide two additional sets of d-cells, a *kernel* K and a *neighborhood* N, such that $K \subseteq T_1$ and $T_n \subseteq N$. These two sets define an "envelope" within which the modified shape sequence will be constrained. Note that, if both K, N have non-trivial topology, the modified shapes will *preserve* those topological features that persist from K to N. If not provided by the user, K is set to be a single d-cell inside T_1 and $N = C_d$. Since K has a simple topology (a single connected component without handles or cavities), our method will attempt to remove all topological features in the shape sequence.

Given the shape sequence $\{T_1, \ldots, T_n\}$, the geometric cost functions g_i, the kernel K and neighborhood N, we seek a modified sequence of shapes $\{T'_1, \ldots, T'_n\}$ that minimizes, lexicographically, the following vector energy that measures (firstly) the total number of topological features and (secondly) the total geometric change,

$$E(\{T'_1, \ldots, T'_n\}) = \{\sum_{i=1}^{n} \sum_{k=0}^{d} \beta_k(T'_i), \quad \sum_{i=1}^{n} g_i(T'_i \ominus T_i)\}, \tag{4.1}$$

where β_k is the k-th Betti number and $\ominus$ is the symmetric difference operator, subject to the constraint that the modified sequence is nesting and sandwiched within K, N:

$$K \subseteq T'_i \subseteq \ldots \subseteq T'_n \subseteq N \tag{4.2}$$

4.4.2 Candidate-based formulation

The problem formulated above is NP-hard even when $n = 1$ and without considering the geometric cost [75]. In the case of $n = 1$, an alternative and more computationally friendly formulation was proposed in [57], which restricts the shape modifications to a precomputed set of *candidate cuts and fills*. A candidate cut (resp. fill) is a group of d-cells in the original shape (resp. its complement) whose simultaneous deletion (resp. addition) removes one or more topological features of the shape. For example, a cut could be all cells that make up a connected component of the shape, whereas a fill could be a group of cells that form a bridge connecting two components. The energy E (Equation 4.1) can then be expressed as a function of a binary labelling on these candidates, where a label of 0 (resp. 1) means a candidate is included in (resp. excluded from) the shape.

We extend the candidate-based formulation of [57] to $n > 1$ with the nesting constraint. We assume that there exists a set of candidate cuts and fills for each shape T_i, denoted as X_i (their computation will be discussed shortly). Given a 0/1 labelling L of the candi-

dates, the modified shape, denoted by T_i^L, can be written as:

$$T_i^L = (T_i \setminus X_i^{L,0}) \cup X_i^{L,1} \tag{4.3}$$

where $X_i^{L,0}$ and $X_i^{L,1}$ are the unions of candidates of X_i that are labelled by L respectively as 0 and 1. If we replace the modified shapes T_i' in the general formulation (Equations 4.1,4.2) by T_i^L, the problem becomes seeking a labelling L that minimizes the energy

$$E(L) = E(\{T_1^L, \ldots, T_n^L\}), \tag{4.4}$$

where the righthand side is defined in Equation 4.1, while satisfying the constraint that

$$K \subseteq T_i^L \subseteq \ldots \subseteq T_n^L \subseteq N. \tag{4.5}$$

4.4.3 Nesting-aware candidates

The formulation above simplifies the problem by limiting the shape modifications to the use of the pre-computed candidates. The choice of the candidate cuts and fills therefore plays an important role in how well the solution of the simplified problem minimizes the original energy (Equation 4.1).

Naturally, we would like to have candidates that can maximally simplify each shape while having low geometric costs. Such candidates can be obtained, for each shape in the sequence, using the inflation/deflation approach in [57]. To compute the candidate cuts on a shape T (restricted to the envelop within a kernel K and a neighborhood N), their method expands K towards T by iteratively adding cells of T while preserving the topology of K. When no more cells can be added without incurring a topological change, each connected component of the difference between T and the expanded kernel becomes a cut[1]. The candidate fills are constructed in a symmetric fashion by maximally shrinking

[1]The connected components are based on the so-called *R-connectivity* defined in [57] to ensure consis-

the neighborhood N towards T while preserving the topology of N and taking the connected components of the difference between the shrunken neighborhood and T. By construction, such cuts and fills are sufficient to fully simplify the topology of T (except for those topological features shared by both K and N). Furthermore, the order of cells being added or deleted is chosen so that the cuts and fills tend to consist of cells with low per-cell geometric costs.

However, candidates constructed independently for each shape may not be useful towards satisfying the nesting constraint (Equation 4.5). Let X_i be the union of all candidates in X_i, and define the *per-shape kernel* as $K_i = T_i \setminus X_i$ and the *per-shape neighborhood* as $N_i = T_i \cup X_i$. Observe that, regardless of the labelling L, the modified shape T_i^L is sandwiched between K_i and N_i. Now, consider a candidate cut in the next shape T_{i+1} which shares some common cells with K_i (e.g., the cut c_3 in Figure 4.5 (a)). Deleting that cut from T_{i+1} will result in a shape that does not completely contain K_i and hence fails to contain T_i^L for any L. Symmetrically, if a candidate fill in the previous shape T_{i-1} contains some cells that are not in N_i (e.g., the fill f_1 in Figure 4.5 (b)), then adding the fill to T_{i-1} will result in a shape that is not completely contained within N_i and in turn fails to nest inside T_i^L for any L. In both cases, the cut or fill is "useless" in the sense that it cannot be applied without violating the nesting constraint.

tency with their graph formulation.

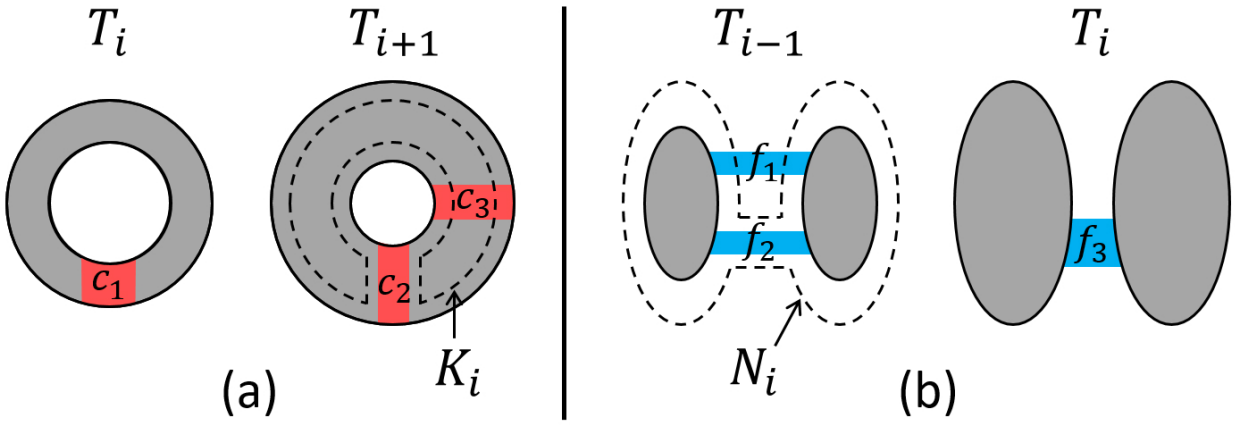

Figure 4.5: Two examples (a,b) of pairs of consecutive shapes with candidate cuts (red) and fills (blue). In (a), the cut c_2 is nesting-aware because it is disjoint from the previous per-shape kernel K_i (shown in outline), but c_3 is not. In (b), the fill f_2 is nesting-aware because it is contained within the next per-shape neighborhood N_i (shown in outline), but f_1 is not.

The discussion above shows that a candidate $x \in X_i$ is useful towards satisfying nesting only if it is disjoint from the previous per-shape kernel (i.e., $x \cap K_{i-1} = \emptyset$) and nested within the next per-shape neighborhood (i.e., $x \subseteq N_{i+1}$)[2]. We call such candidates *nesting-aware*. For example, in Figure 4.5, the cut c_2 and fill f_2 are both nesting-aware, whereas the cut c_3 and fill f_1 are not.

To compute nesting-aware candidates, we first give an alternative characterization using per-shape kernels K_i and neighborhoods N_i:

Proposition 2. *All candidates in* $\{X_1, \ldots, X_n\}$ *are nesting-aware if and only if*

$$K \subseteq K_1 \subseteq \ldots \subseteq K_n \tag{4.6}$$

and

$$N_1 \subseteq \ldots \subseteq N_n \subseteq N \tag{4.7}$$

where $K_i = T_i \setminus X_i$ *and* $N_i = T_i \cup X_i$.

Proof. We first show sufficiency. Let $K_0 = K$ and $N_{n+1} = N$. Since $K_{i-1} \subseteq K_i$ and $X_i \cap$

[2]We assume $K_0 = K$ and $N_{n+1} = N$

$K_i = \emptyset$ for $i = 1, \ldots, n$, $x \cap K_{i-1} = \emptyset$ for any $x \in X_i$. Similarly, Since $N_i \subseteq N_{i+1}$ and $X_i \subseteq N_i$ for $i = 1, \ldots, n$, $x \subseteq N_{i+1}$ for any $x \in X_i$. To show necessity, suppose on the contrary that all candidates are nesting-aware but $K_{i-1} \nsubseteq K_i$ for some $i \in [1, n]$. Hence there exists some d-cell $c \in K_{i-1}$ but $c \notin K_i$. Since $K_{i-1} \subseteq T_{i-1} \subseteq T_i$, $c \in T_i$, and hence c must belong to some cut $x \in X_i$. However, x overlaps with K_{i-1} at c, which contradicts that x is nesting-aware. A similar contradiction can be reached if all candidates are nesting-aware but $N_i \nsubseteq N_{i+1}$ for some $i \in [1, n]$. □

The characterization leads to a variation of the inflation/deflation approach of [57] to compute nesting-aware candidates that are also sufficient for simplifying topology and low in geometric costs. To compute the candidate cuts, instead of expanding the same kernel K to every shape T_i, we expand the previous per-shape kernel K_{i-1} towards T_i while preserving the topology of K_{i-1} and guided by the geometric cost g_i. The result of maximal expansion becomes the current per-shape kernel K_i, and the connected components of $T_i \setminus K_i$ become the candidate cuts in X_i. Our algorithm starts from T_1 and iteratively creates a expanding sequence of per-shape kernels $\{K_1, \ldots, K_n\}$ satisfying Equation 4.6. The candidate fills are created in a symmetric fashion, this time working backwards from T_n. For each shape T_i, we shrink the next per-shape neighborhood N_{i+1} towards T_i to produce the current per-shape neighborhood N_i, and the connected components of $N_i \setminus T_i$ become the candidate fills in X_i. The process results in a sequence of shrinking per-shape neighborhoods $\{N_n, \ldots, N_1\}$ satisfying Equation 4.7. The inflation and deflation processes are illustrated in 2D in Figure 4.6.

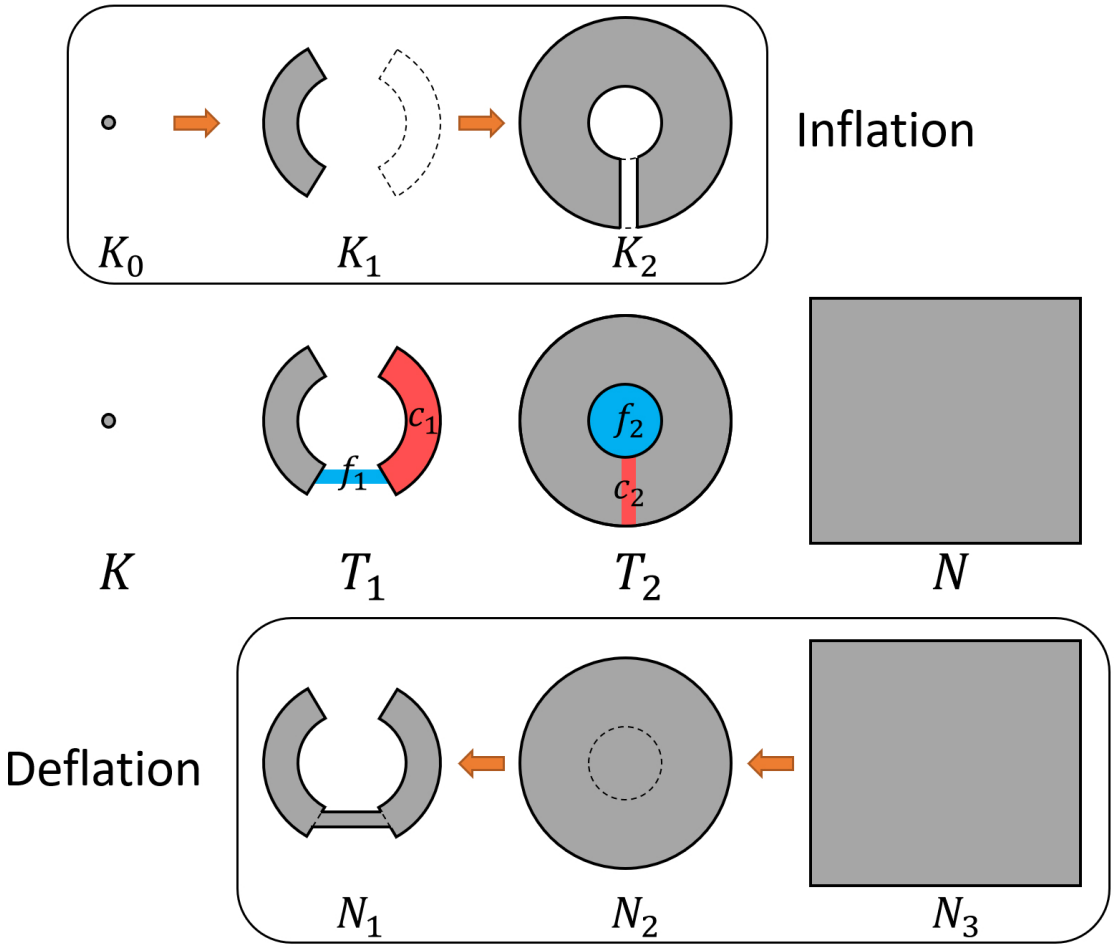

Figure 4.6: Computing nesting-aware candidates. Middle row: An input sequence consisting of two shapes T_1, T_2, the kernel K, and neighborhood N. Top: Inflating $K_0 = K$ towards successive shapes in the sequence, while preserving topology, results in per-shape kernels K_1, K_2. Bottom row: Deflating $N_3 = N$ towards successive shapes in the reverse order, while preserving topology, results in per-shape neighborhoods N_2, N_1. Components of $T_i \setminus K_i$ and $N_i \setminus T_i$ become the candidate cuts (red) and fills (blue) in the middle row.

Finally, we show that the use of nesting-aware candidates further simplifies the problem formulation. In particular, the inclusion constraint of Equation 4.5 can be transformed into labelling constraints on pairs of *overlapping* candidates:

Proposition 3. *If all candidates* $\{X_1, \ldots, X_n\}$ *are nesting-aware, then Equation 4.5 holds if and only if the following holds: for any* $i = 1, \ldots, n-1$ *and any two candidates* $x \in X_i$ *and* $y \in X_{i+1}$ *such that* $x \cap y \neq \emptyset$*, either* $L(x) = 0$ *or* $L(y) = 1$*.*

Proof. We first note that

$$T_i^L = K_i \cup X_i^{L,1} = N_i \setminus X_i^{L,0}.$$

Hence $T_i^L \subseteq T_{i+1}^L$ is equivalent to

$$K_i \cup X_i^{L,1} \subseteq N_{i+1} \setminus X_{i+1}^{L,0}.$$

Since all candidates are nesting-aware, and by Proposition 2, $K_i \subseteq N_i \subseteq N_{i+1}$. Hence the inequality above holds if and only if $X_i^{L,1} \cap X_{i+1}^{L,0} = \emptyset$. The latter, in turn, is equivalent to asking that, for any two candidates $x \in X_i$ and $y \in X_{i+1}$ such that $x \cap y \neq \emptyset$, $\{L(x), L(y)\}$ cannot be $\{1, 0\}$. □

Note that a pair of overlapping candidates $x \in X_i$ and $y \in X_{i+1}$ can be both cuts (e.g., c_1 and c_2 in Figure 4.5 (a)), both fills (e.g., f_2 and f_3 in Figure 4.5 (b)), or a fill and a cut (e.g., f_1 and c_2 in Figure 4.6 middle row). We call the simultaneous assignment of label 1 to x and 0 to y a *conflict*. Assuming that all candidates are nesting-aware, our optimization problem can now be stated simply as *finding a conflict-free labelling L that minimizes $E(L)$ defined in Equation 4.4.*

4.5 Optimization

We explore several strategies to solve the conflict-free labelling problem formulated above. We start with a heuristic approach that sequentially optimizes the labels in successive shapes while avoiding conflicts with previous shapes (Section 4.5.1). We then introduce a state-space search algorithm and prove that it always returns the optimal labelling, although with possibly exponential complexity (Section 4.5.2). Finally, we discuss a beam-search variant of the optimal algorithm that trades off optimality for efficiency (Section 4.5.3).

In all these strategies, we use the method of [57] as a black-box solver to optimize labels

on individual shapes without considering conflicts. For convenience of discussion, we introduce two notations. Given two subsets F_0, F_1 of candidates X_i, we use $L_i(F_0, F_1)$ to denote the labels of X_i computed by [57] to minimize $E(\{T_i^L\})$ (Equation 4.1) under the constraints that F_0 are all labelled 0 and F_1 are all labelled 1. Given subsets F_0, F_1 of all candidates on all shapes, we denote by $L(F_0, F_1)$ the labelling of the entire sequence made up of per-shape labels $L_i(F_0 \cap X_i, F_1 \cap X_i)$ for $i = 1, \ldots, n$.

4.5.1 Propagation

One idea is to "propagate" the labels from one shape to the remaining shapes in a sequential manner. During propagation, the labels of candidates X_i, denoted as L_i, are obtained by minimizing the labelling energy of L_i while constraining some labels to avoid conflicts with previously propagated labels in either L_{i-1} or L_{i+1}.

Specifically, starting from some $j \in [1, n]$, we first obtain the labels L_j of X_j as $L_j(\emptyset, \emptyset)$ (i.e., without constraining any labels). The labels are then propagated forward and backward to adjacent shapes. In the forward direction, for each $i = j + 1, \ldots, n$, we define the constraint set $F_{i,1}$ as all candidates of X_i that overlap with some 1-labelled candidates in X_{i-1}:

$$F_{i,1} = \{x \in X_i | x \cap y \neq \emptyset, \exists y \in X_{i-1}, L_{i-1}(y) = 1\}$$

Candidates in this set must be labelled 1 to avoid conflicts with L_{i-1}. Hence we obtain $L_i = L_i(\emptyset, F_{i,1})$. Similarly, in the backward direction, for each $i = j - 1, \ldots, 1$, we define the constraint set $F_{i,0}$ as all candidates of X_i that overlap with some 0-labelled candidates in X_{i+1}:

$$F_{i,0} = \{x \in X_i | x \cap y \neq \emptyset, \exists y \in X_{i+1}, L_{i+1}(y) = 0\}$$

Since candidates in this set must be labelled 0 to avoid conflicts with L_{i+1}, we obtain $L_i = L_i(F_{i,0}, \emptyset)$. The forward and backward processes are illustrated in Figure 4.7 top and bottom.

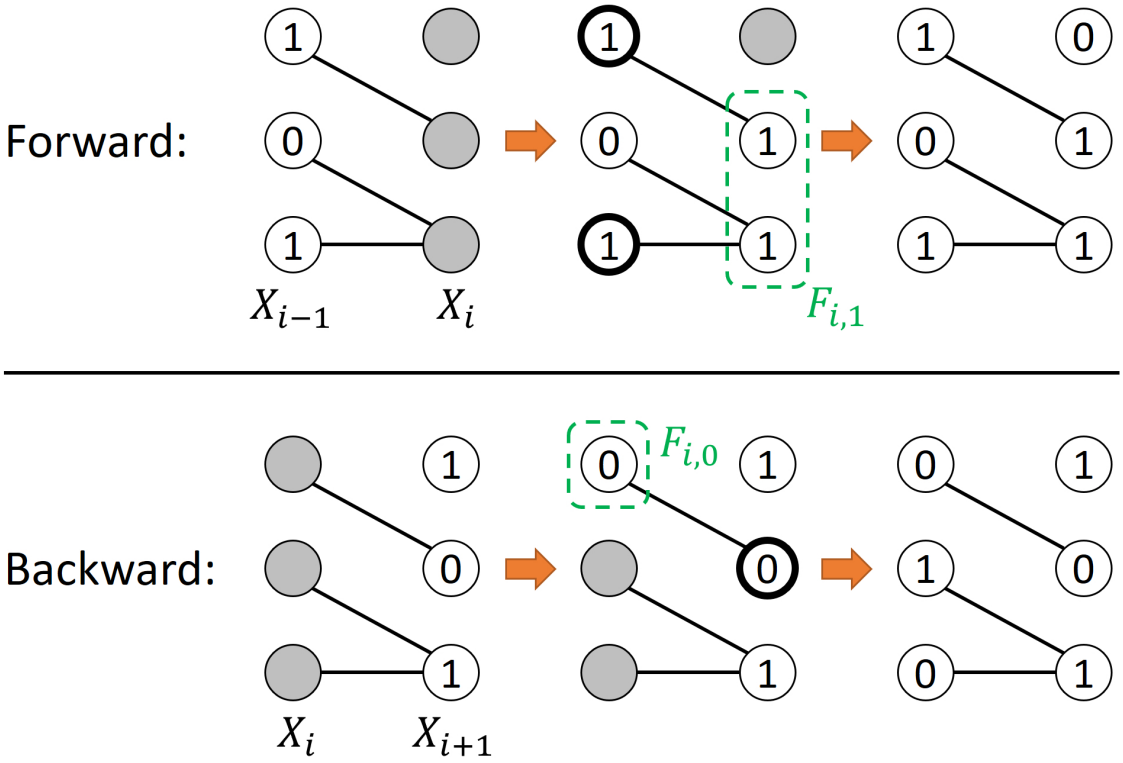

Figure 4.7: Forward (top) and backward (bottom) propagation of labels. Candidates are shown as circles with their labels (gray circles have unknown labels) and overlapping candidates are connected by edges. Top: to get labels of candidates X_i from X_{i-1}, the subset $F_{i,1} \subseteq X_i$ that overlaps with 1-labelled candidates of X_{i-1} (thick outlines) is labelled 1, and the remaining labels are computed by energy minimization. Bottom: to get labels of candidates X_i from X_{i+1}, the subset $F_{i,0} \subseteq X_i$ that overlaps with 0-labelled candidates of X_{i+1} (thick outlines) is labelled 0, and the remaining labels are computed by energy minimization.

Let L^j be the labelling over the entire sequence composed of the per-shape labels $\{L_1, \ldots, L_n\}$ propagated from L_j. We perform propagation once from each starting index $j = 1, \ldots, n$, which results in n labellings $\{L^1, \ldots, L^n\}$, and output the one with the least energy.

4.5.2 Optimal search

While the propagation method creates a conflict-free labelling, it is not guaranteed to be optimal in terms of its energy. We next describe an *optimal* algorithm based on state-space exploration. Here, a *state* is a triple $\{F_0, F_1, L\}$ where F_0 are candidates constrained to have label 0, F_1 are candidates constrained to have label 1, and $L = L(F_0, F_1)$ is the

energy-minimizing labelling subject to these constraints. The algorithm starts with a single state $\{\emptyset, \emptyset, L(\emptyset, \emptyset)\}$, whose labelling minimizes the energy on each shape without enforcing nesting. At each iteration, the algorithm removes the state whose labelling achieves the least energy among all existing states. Let this state be $\{F_0, F_1, L\}$. If L is free of conflicts, the algorithm terminates and L is returned as the output. Otherwise, a conflict of L is chosen, and new states are created to resolve that conflict by adding more constraints to either F_0 or F_1.

The key step in the algorithm is the creation of new states. Consider a labelling L and a pair of overlapping candidates $x \in X_i$ and $y \in X_{i+1}$ with conflicting labels in L, that is, $L(x) = 1$ and $L(y) = 0$. A simple way to resolve the conflict is to constrain either x to have label 0 or y to have label 1. However, such constraints alone may lead to new conflicts with other shapes. For example, labelling x as 0 may conflict with 1-labelled candidates in X_{i-1} that overlap with x, and labelling y as 1 may conflict with 0-labelled candidates in X_{i+2} that overlap with y. To avoid such potential conflicts, our algorithm constrains not one, but possibly a set of candidates when resolving each conflict.

Specifically, we call two candidates $u \in X_i$ and $v \in X_j$ where $i < j$ *path-connected* if there is a sequence of candidates $\{w_i, \ldots, w_j\}$ such that $w_i = u$, $w_j = v$, $w_k \in X_k$ for $k = i, \ldots, j$, and each pair of consecutive candidates overlap (i.e., $w_k \cap w_{k+1} \neq \emptyset$ for $k = i, \ldots, j-1$). For each candidate $x \in X_i$, we define its 0-*set*, $z(x)$, as the union of x and all candidates in X_j for $j < i$ that are path-connected with x, and its 1-*set*, $o(x)$, as the union of x and all candidates in X_j for $j > i$ that are path-connected with x. To resolve the conflict involving a pair of candidates $\{x \in X_i, y \in X_{i+1}\}$, we constrain either all candidates in $z(x)$ to have label 0 or all candidates in $o(y)$ to have label 1. The constraints are appended to existing constraints in either F_0 or F_1, and the remaining labels are updated using energy minimization. For efficiency, we only update the labels of X_k, for $k = 1, \ldots, n$, if it has some constrained candidate z whose constraint label is different from its current label $L(z)$. This process is illustrated in Figure 4.8.

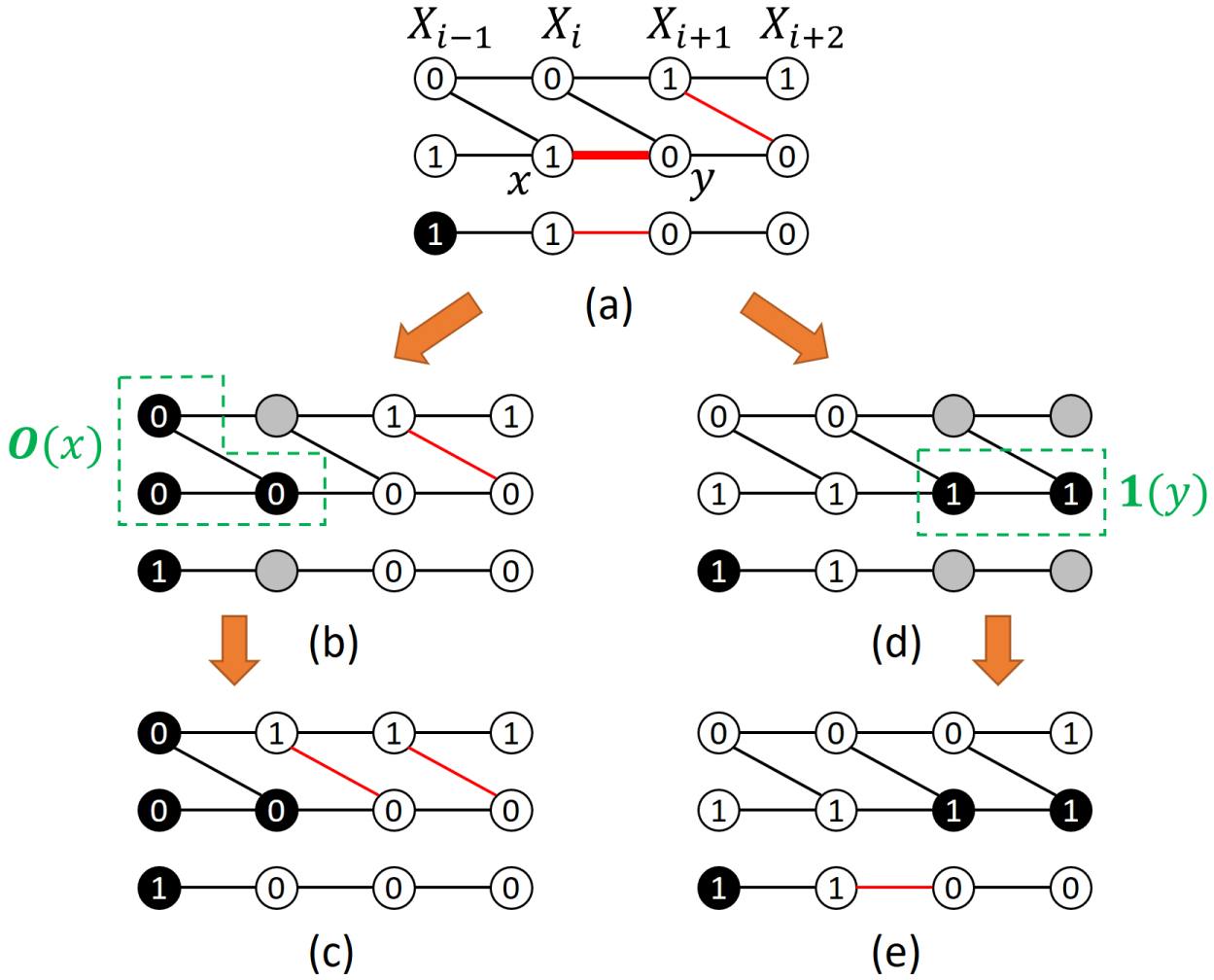

Figure 4.8: State expansion. Given a state (a) consisting of candidate labels, constrained candidates (black circles), candidate pairs with conflicting labels (red edges), and a chosen pair $\{x, y\}$, two new states are created by expanding the constraints to include either the 0-set of x (b) or the 1-set of y (d) and updating the remaining labels on the affected shapes using energy minimization (c,e).

Algorithm 1 State-space search algorithm

Initialize empty queue Q
Q.push($\{\emptyset, \emptyset, L(\emptyset, \emptyset)\}$)
while Q *is not empty* **do**
  $\{F_0, F_1, L\} \longleftarrow Q$.pop()
  if L *has no conflict* **then**
    return L
  else
    $\{x, y\} \longleftarrow$ a conflict pair of candidates in L
    $F_0' \longleftarrow z(x) \cup F_0$
    $F_1' \longleftarrow o(y) \cup F_1$
    Q.push($\{F_0', F_1, L(F_0', F_1)\}$)
    Q.push($\{F_0, F_1', L(F_0, F_1')\}$)
  end
end

The search algorithm is summarized as pseudo-code in Algorithm 1. The algorithm maintains current states in a queue Q sorted by increasing labelling energy. Each iteration of the algorithm removes the state in Q with the least labelling energy and either returns the labelling, if it is conflict-free, or adds two new states to Q. We prove in the appendix that the algorithm always terminates, and that the result is optimal when the solver L is optimal:

Proposition 4. *Algorithm 1 terminates and produces a conflict-free labelling L in finite number of iterations. Furthermore, if $E(L(F_0, F_1))$ is minimal among all labellings that label F_0 as 0 and F_1 as 1 for any two disjoint sets of candidates F_0, F_1, then $E(L)$ is minimal among all conflict-free labellings.*

We make a final comment on the choice of the conflict candidate pair $\{x, y\}$ (Line 8 in Algorithm 1). Although the choice does not affect the optimality of the algorithm, it has an impact on the order of states being explored and therefore the overall running time. We adopt the following choice in our implementation. For each conflict $\{x, y\}$ of L, we count the number of 1-labelled candidates in $z(x)$ and the number of 0-labelled candidates in $o(x)$. We choose the conflict where the absolute difference between those two numbers is the smallest (ties are broken arbitrarily). This strategy prioritizes conflicts where one of

the two ways of resolving it (either constraining $z(x)$ to 0 or $o(y)$ to 1) has a clearer benefit over the other in terms of minimizing the number of label-flippings in L.

4.5.3 Beam search

While being optimal, the search algorithm described above may have a prohibitive running time. As shown in the appendix, the states explored by the algorithm form a binary tree whose depth can be as large as the total number of candidates in all shapes. Denoting this number by m, the algorithm therefore may require $O(2^m)$ number of iterations. This can be impractical for inputs with many candidates, either due to a high level of topological noise or a large number of shapes in the sequence.

To make the algorithm more practical, at the cost of losing optimality, we can replace the full search by a beam search with a limited (and controllable) size of memory. Specifically, the maximum number of states that can be held in the queue Q is restricted to be a user-provided constant B. When Q is full and a new state needs to be pushed, the state whose labelling has the highest energy among the $B + 1$ states is dropped from Q.

In contrast to the full search, the beam search requires only $O(Bm)$ number of iterations. In practice, we found that even a small B (e.g., 1) can produce near-optimal results but with significantly improved efficiency over the full search. In the extreme case that $B = 1$, Q holds only a single state at any time, and the algorithm simply replaces the current state at each iteration by one of the two newly created states that has a smaller labelling energy.

4.6 Results

We present experimental results that evaluate the effectiveness of our method, compare the different optimization strategies, and compare with alternative simplification methods. As mentioned earlier, while our method in theory can be applied to any type of cell com-

plexes, our current implementation assumes that the input shapes are defined on a pixel (or voxel) grid using 4-connectivity (or 6-connectivity).

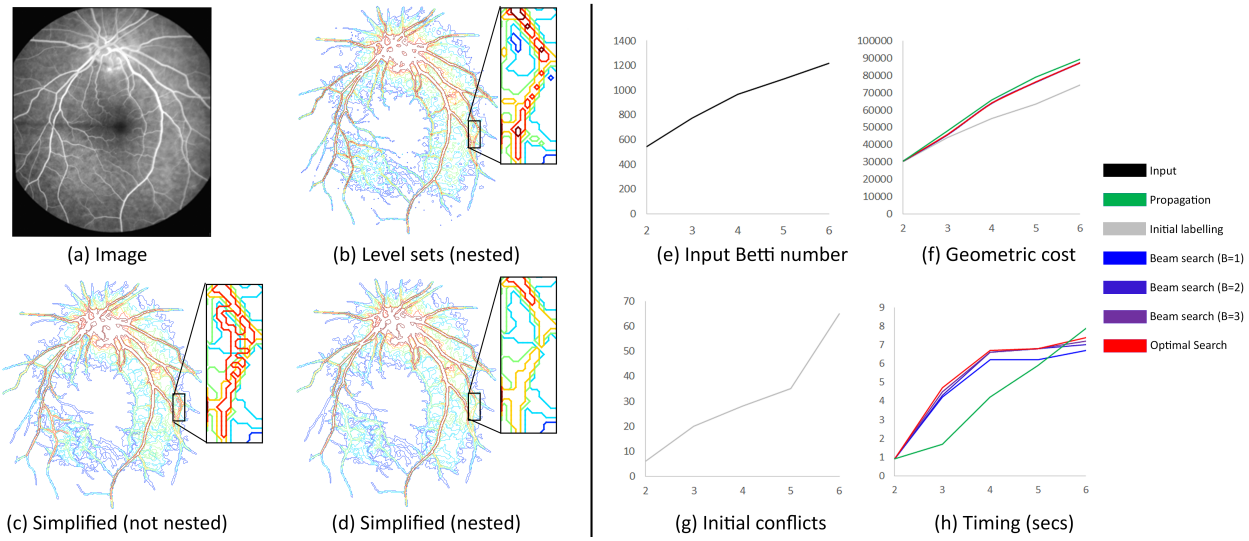

Figure 4.9: Left: a grayscale retinal vessel image (a), 6 level sets used as input shapes (b), shapes simplified using the initial labelling (c) and conflict-free final labelling produced by our beam search ($B = 1$) (d). Right: plotting total Betti number of the input sequence (e), geometric costs of various optimization approaches (f), number of conflicts of the initial labelling (g), and running time of various optimization approaches (h) over experiments that use increasingly long input sequences.

4.6.1 Level sets

To compare different optimization options, we use the level sets of a grayscale 2D or 3D image at decreasing (or increasing) levels. Such "synthetic" sequences are always nested, and their complexity can be easily adjusted by changing the number of levels. Assuming the sequence consists of level sets with decreasing levels, we set the kernel K to be one of the pixels or voxels with the highest intensity (and hence is contained in the first shape) and the neighborhood N to be the entire image. Following [57], we set the geometric cost function $g_i(v)$ to be the magnitude of the intensity gradient of the image at a pixel or voxel v, which penalizes strong intensity edges, and prioritize pixels or voxels with higher (resp. lower) intensities during inflation (resp. deflation) to compute the nesting-aware candidates.

2D level sets (Figure 4.9): We first test on level sets extracted from a 2D retinal vessel image shown in (a). We took 6 expanding level sets from the image, as shown in Figure (b) with colors ranging from red to blue. Observe that these level sets have a large amount of topological noise including islands and cavities. We set up 5 experiments with increasingly longer input sequences, so that the i-th experiment uses the last $i + 1$ level sets as the input. As seen in (e), the topological complexity of the input sequence, measured by the total Betti numbers ($\beta_0 + \beta_1$) for all shapes in the sequence, increases as the experiment uses longer sequences.

In all five experiments, each of the three optimization approaches, namely propagation, optimal search, and beam search (we tried $B = 1, 2, 3$), is able to fully simplify the topology of the input sequence, meaning that each simplified shape has a single component without cavities. We visually compare the initial labelling in the search algorithm, which is obtained by [57] without considering nesting, and the labelling returned by beam search with $B = 1$ in the last experiment (using all 6 level sets) in (c,d). While the initial labelling also results in topologically simple shapes, the simplified shapes could intersect (e.g., red, cyan and yellow curves in the close-ups of (c)), whereas the final labelling restores the nesting relation while maintaining topological simplicity. Beyond topology, different optimization approaches result in similar geometric costs, as plotted in (f), although propagation produces slightly higher costs. These costs become increasingly higher than the costs of the initial labelling, which correlates with the increase in the number of conflicts in the initial labelling as shown in (g).

In terms of performance, (h) shows that both optimal search and beam search have similar running time, while beam search with smaller B is slightly faster. The propagation approach is generally faster than both search algorithms, although the advantage decreases with the length of the input sequence. This is because the propagation approach requires computing the energy-minimizing labels on all shapes for each starting shape, making its complexity quadratic to the number of shapes. In contrast, the beam search's complexity

depends on a variety of factors such as the number of conflicts and the particular search path in the state space. Note that the time for computing the nesting-aware candidates is not included here, because they are common for all optimization approaches.

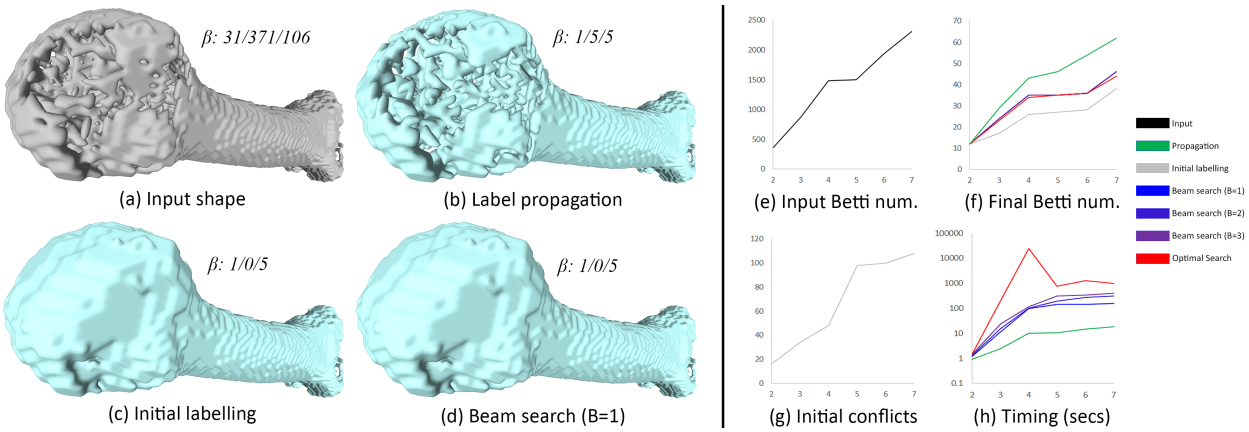

Figure 4.10: Left: one of the seven level sets of a 3D CT image of a human foot bone that are used as input shapes (a), the shape simplified by label propagation (b), the initial labelling (c) and final conflict-free labelling of beam search ($B = 1$) (d). Betti numbers of each shape ($\beta_0/\beta_1/\beta_2$) are shown. Right: plotting total Betti numbers of the input sequence (e), total Betti numbers of the simplified sequence using various optimization approaches (f), number of conflicts of the initial labelling (g), and running time of various optimization approaches (h) over experiments that use increasingly long input sequences.

3D level sets (Figure 4.10): Our next test uses 7 level sets from a 3D CT image of a human foot bone, one of which is shown in (a). Observe that the bone shape has numerous handles, cavities, and islands in its interior. Similar as above, we set up 6 experiments using increasingly longer sequences of these level sets. The increase in topological complexity (as the total Betti number $\beta_0 + \beta_1 + \beta_2$ over all shapes) in these experiments is plotted in (e). To make the problem more challenging, we further removed some candidate cuts and fills so that the shapes cannot be fully simplified. Specifically, for each level set T_i, we removed all candidates from X_i that contain some voxel whose intensity is more than 40 away from the level of T_i (assuming the entire intensity range is 255).

Observe from (f) that both optimal search and beam search produce shapes with significantly fewer topological features than label propagation. Also, beam search (even with $B = 1$) produces only slightly more complex topology than optimal search. We visually

compare in (b,c,d) the results of the propagation approach, initial labelling and final labelling of beam search for the input shape in (a). Observe that beam search results in fewer handles than propagation and better retains the appearance of the initial, energy-minimizing labelling. Performance-wise (h), propagation is notably faster than beam search, which is in turn significantly faster than optimal search, sometimes by two orders of magnitude. As a result, beam search (with $B = 1$) appears to offer the best compromise - it produces simplifications much closer to optimum than propagation but at a fraction of cost of the optimal search.

4.6.2 Real-world data

We applied our method to several real-world data sets including tissue layers in a brain scan and time series of growing plant roots. In each input sequence, all shapes are given as binary masks on a common voxel grid. We set the kernel K as one of the voxels inside the first shape that are furthest from the shape's boundary, and the neighborhood N as the entire grid. The geometric cost $g_i(v)$ is set to be constant for each voxel v, and the inflation and deflation prioritize voxels that are further away from the boundary of the shape being inflated or deflated towards.

We visually compare our method using the beam search strategy ($B = 1$) with the single-shape simplification method of [57] and the scalar-field simplification method of [85] implemented in the Topology Toolkit (TTK) [88]. For the latter, we first constructed a real-valued grayscale volume that interpolates the input shape sequence at integers equal to the indices of the shapes, then maximally simplified the extrema of the volume using TTK, and finally extracted the level sets of the simplified volume at the original integer values.

Brain layers (Figure 4.11): The input sequence (a) consists of 3 layers in a segmented human brain scan, namely (from inner to outer) the cerebrospinal fluid, white matter, and grey matter. Numerous topological errors exist in this input, particularly on the two inner layers (as seen in the close-ups), due to the complex anatomical structure. Both our

method and [57] can remove all topological features on all three shapes, but the nesting relation between the layers is only preserved in our method. The close-ups in (b,c) visualize the two inner layers simplified by each method. In the result of [57] (b), the inner most layer (red) can be seen outside the middle layer (gray) indicating a violation of nesting, while in our result (c) the inner layer is completely hidden inside the middle layer.

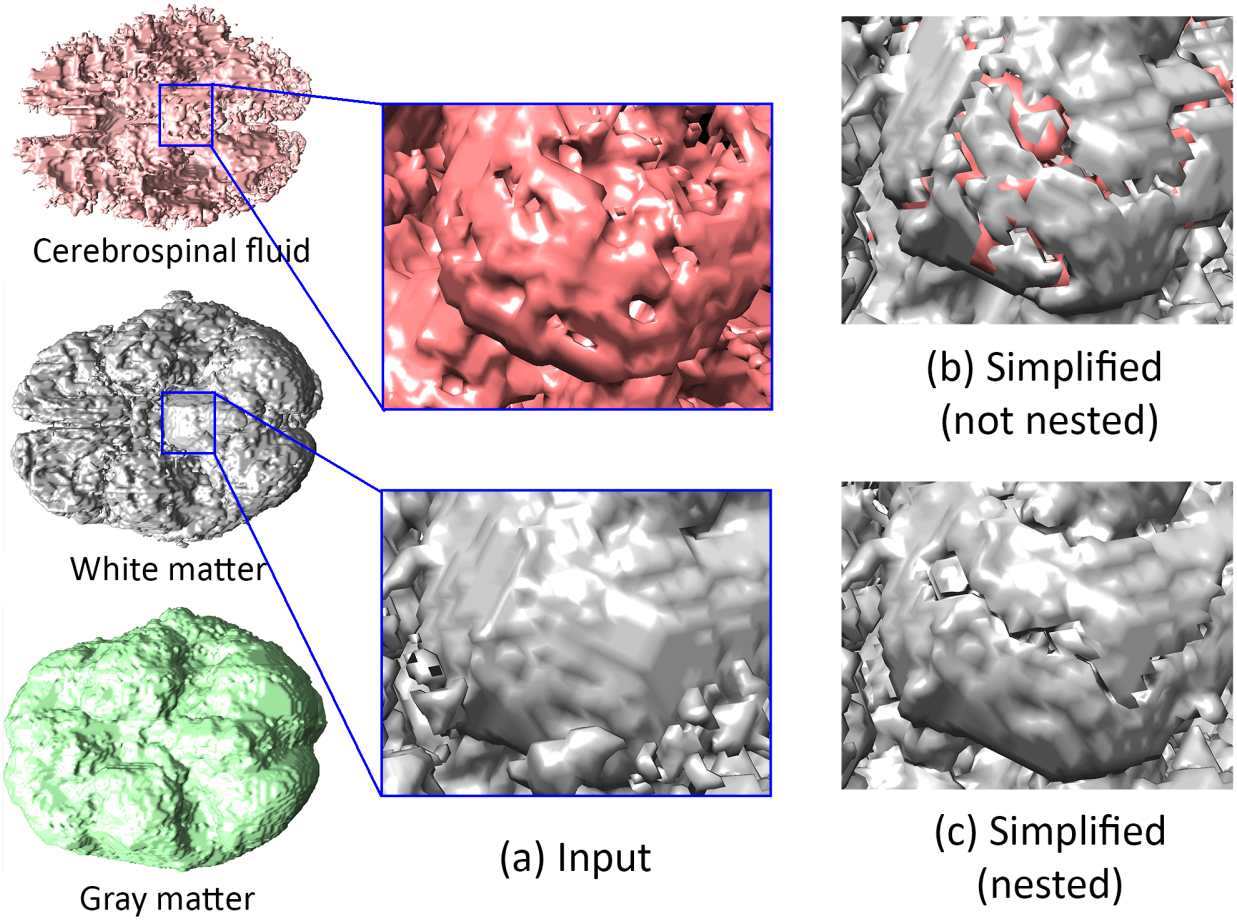

Figure 4.11: (a) Three layers of a brain scan and a close-up on a region with complex topology in the two inner layers. (b) Two inner layers after applying the method of [57]; note the innermost layer extrudes outside the middle layer. (c) Our method keeps the innermost layer nested inside the middle layer.

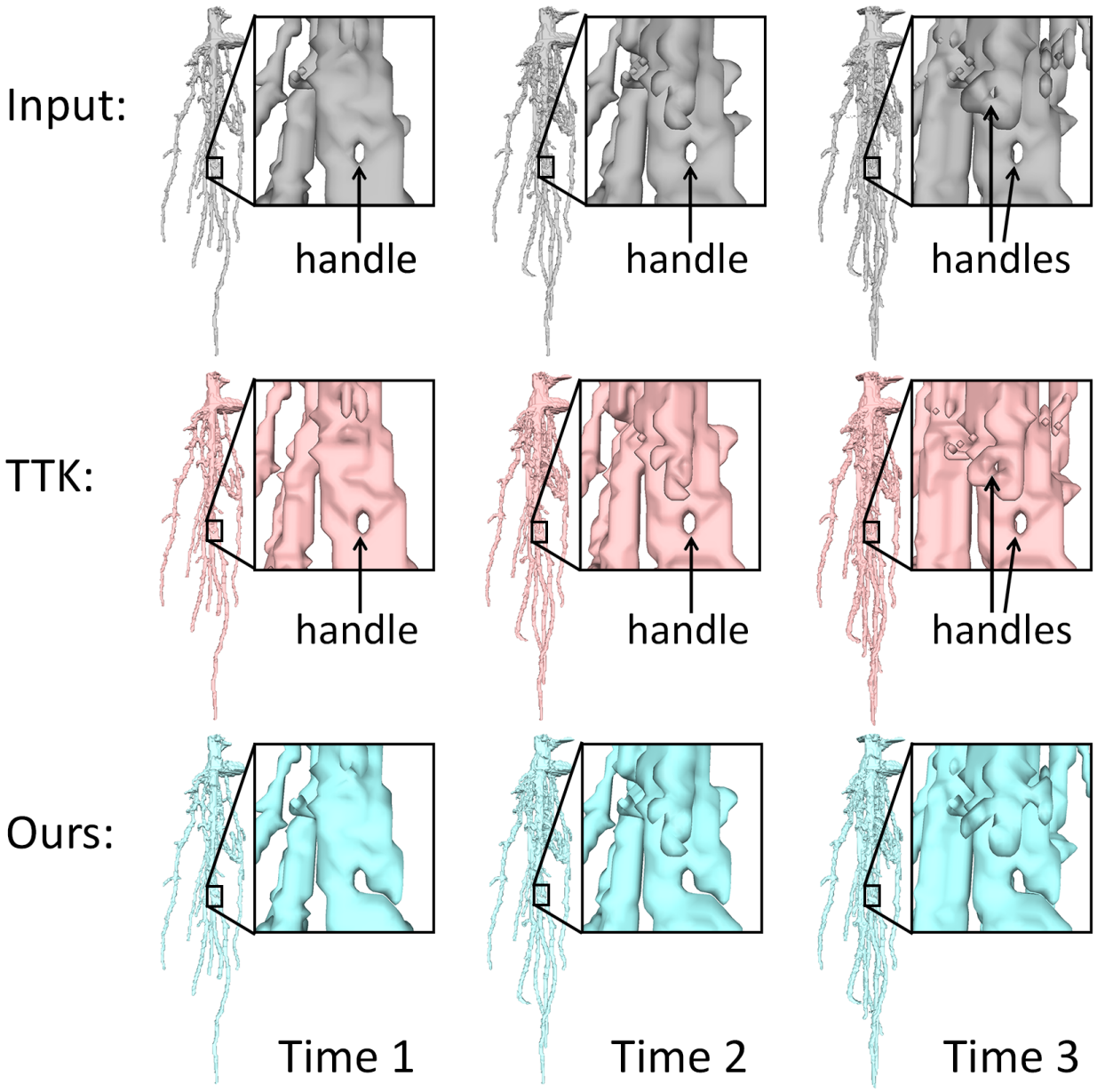

Figure 4.12: Simplifying a sequence of 3 time points of a growing rice root system (top row) using TTK (middle row), which fails to remove most of the handles, and our method (bottom row) which fully simplifies the topology of each shape while maintaining nesting.

Growing roots (Figures 4.2,4.12): As roots grow in the soil or other media, and due to resistance of their surroundings, their shape become nested over time. Generally, roots are connected and free of handles or cavities, but the reconstruction of root shapes is prone to topological errors due to the presence of thin and nearby branches. Figure 4.2 compares our method with that of [57] on a sequence of 4 time points of a growing pennycress

Table 4.1: Statistics for the four real-world data sets used in this paper (Figures 4.11, 4.12, 4.2, 4.13), showing the Betti numbers of the input sequence, time for computing the candidate cuts and fills, and the resulting topological complexity, geometric cost, and running time of the three optimization strategies.

	Brain (3 layers)			Rice root (3 times)			Pennycress root (4 times)			Maize root (41 times)		
	$\beta_0/\beta_1/\beta_2$	Geom. cost	Time (sec)	$\beta_0/\beta_1/\beta_2$	Geom. cost	Time (sec)	$\beta_0/\beta_1/\beta_2$	Geom. cost	Time (sec)	$\beta_0/\beta_1/\beta_2$	Geom. cost	Time (sec)
Input	1247/863/80	-	-	352/133/4	-	-	408/254/0	-	-	12290/6780/201	-	-
Candidates	-	-	71.2	-	-	19.1	-	-	64.3	-	-	549.0
Propagation	3/0/0	897437.7	30.9	3/0/0	9931.2	1.3	4/0/0	3981.4	7.1	41/0/0	989392.5	26106.8
Opt. search	3/0/0	760882.1	248.6	3/0/0	9844.6	2.8	4/0/0	2765.3	43.1	N/A	N/A	N/A
Beam search	3/0/0	760882.1	71.4	3/0/0	9844.6	2.8	4/0/0	2773.3	9.1	41/0/0	709947.5	2664.1

root system. Again, while both methods fully simplify the topology, only our method preserves the nesting between the time points. Figure 4.12 shows another sequence of 3 time points of a growing rice root system and compares our method with TTK. Observe from the close-ups that, while the result of TTK remains nested (as they are level sets), TTK fails to remove the handles on the input shapes. This is a common limitation of field-based simplification methods (see Section 4.2). In contrast, our method removes all topological features (handles included) and keeps the shapes nested.

Quality and performance: Table 4.1 reports, for each real-world data set, the topological complexity of the input, running time for computing the nesting-aware candidates, and the cost (topology and geometry) and running time of the three optimization strategies ($B = 1$ for beam search). We additionally include a sequence of 41 time points of a growing maize root system (four time points are shown in Figure 4.13), which is our most complex data set. Observe that the optimal search is by far the most expensive of the three optimization approaches, and it failed to finish within ten hours for the maize data. Propagation is faster than beam search for the shorter sequences, but it takes significantly longer on the maize data due to the much longer sequence. Upon completion, all three strategies fully simplify the topology of these sequences, while optimal search and beam search have similar geometric costs that are lower than propagation. Once again, beam search seems to strike a better balance between optimality and speed.

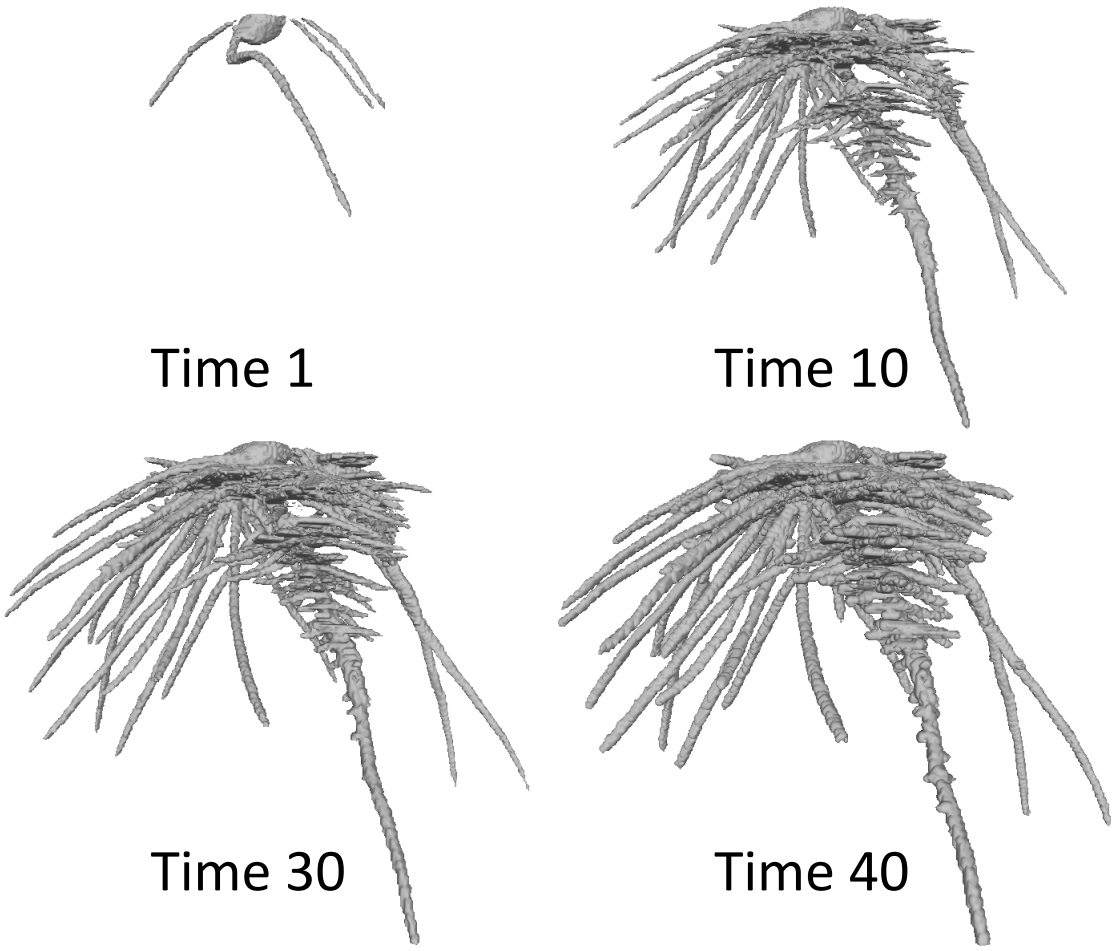

Figure 4.13: Four time points of a growing maize root system from a sequence of 41 time points.

4.7 Conclusions

We present, to the best of our knowledge, the first method for simultaneous topological simplification of a sequence of nested shapes while preserving nesting. Our method extends a recent global optimization method [57] to nested shapes by defining and computing nesting-aware candidate cuts and fills, formulating a constrained labelling problem using these candidates, and exploring several solution strategies that offer different trade-offs between optimality and efficiency. The method is shown to be effective on both synthetic and real-world shape sequences.

While our method produces topologically simple and nested shapes, the topological changes

made by the algorithm may not be geometrically pleasing (e.g. the long "scar" on the surface of the white matter in Figure 4.11 (c)) and may destroy the semantics of the shape (e.g., the branching structure of the roots). The key to improving the quality of the results is obtaining candidate cuts and fills that better respect the geometry and semantics of the input shapes. We would also like to improve the efficiency of the current inflation/deflation method for computing the candidates, which currently accounts for a significant portion of the overall runtime, and extend the implementation to handle non-cubical complexes.

Conclusion

5.1 Summary

The aim of this dissertation was to develop algorithms for topological optimization, using plant root phenotyping as a primary motivator. An algorithm for topological simplification of a single shape represented as a cell complex was developed which uses a global optimization approach to maximally simplify the number of topological features while minimizing the change to the geometry. The algorithm laid the foundation for the development of TopoRoot, which captures the hierarchy and fine-grained traits of excavated Maize Roots from 3D imaging, through a computer graphics-based pipeline. The final part of this dissertation was the development of an algorithm for topological simplification of nested shapes which maximally simplifies the number of topological features of every shape in the sequence while minimizing the change to the geometry and maintaining consistency.

5.2 Future Work

There remains significant room for research in each of the topics covered in this dissertation. While the global optimization algorithm developed in the first part of this thesis is able to improve on prior methods in reducing the number of topological features with minimal geometric change, a few topological features remain in the final result for extremely complex examples such as iso-surfaces of Maize roots extracted from X-ray CT scans (with thousands of topological features). These noisy topological features remain due to the suboptimality of the Node-Weighted Steiner Tree (NWST) solver used in our implementation. As such, a possible direction to explore would be the use of more optimal solvers for the NWST problem, or different graph formulations to solve the same problem besides NWST.

Furthermore, though the cuts and fills simplify the topology, they sometimes are not geometrically pleasing, such as cutting roots along their cross sections. By incorporating geometric smoothness criteria or domain-specific criteria (e.g. root biology), the visual quality of the cuts and fills can be improved to better reflect the natural semantics of the underlying shape being repaired. Another direction of improvement would be to extend the algorithm to Octrees and non-cubical complexes such as irregular grids.

The *TopoRoot* pipeline developed in part 2 of this dissertation has diverse areas for improvement and exploration, and we envision incorporating these changes into future versions of the pipeline. Since the cuts and fills from the topological simplification algorithm of the first part of the dissertation are not generated using criteria specific to root biology, the algorithm sometimes performs changes to the root shape which do not reflect its natural growth patterns. Improvements to those cuts and fills would thus improve the quality of the output hierarchy. Upon close examination of the hierarchy in our graphical user interface, there are still misclassifications of the roots at different hierarchy levels (e.g. portions of nodal roots being mistaken as lateral roots). Most of these mistakes occur at hierarchy levels greater than 1 (e.g. first-order lateral roots and higher), and are due to either faulty cycle breaking along the curve skeleton, or errors in the hierarchy labeling algorithm when running on the acyclic skeleton. Two potential directions to explore would be a global optimization approach to cycle breaking (as opposed to the greedy minimum spanning tree approach currently used in the pipeline) which utilizes better geometric criteria, as well as a hierarchy labeling algorithm with a different strategy than the current one (e.g. recursively maximizing the total length of the root as opposed to minimizing the maximum hierarchy level). The graphical user interface can also be modified to allow for edits to the skeleton if the hierarchy labelling is incorrect. A variety of new traits can also be added to future versions of the pipeline, including whorl locations, inter-whorl distances, and the location of the soil plane. Whorls could potentially be identified either before the skeleton is produced in our pipeline, or after it. In the former case, the identi-

fied whorls may help to distinguish between ”real roots” which emerge from whorls, and ”fake roots” which do not come from any whorl, and which are actually topological artifacts from the imaging. This may help with cycle breaking further down in the pipeline. Finally, TopoRoot can also be potentially extended to other species such as Sorghum, Rice, and Pennycress. This would involve a tiller identification algorithm for multi-tiller species such as Sorghum, and a potential starting point of exploration would be to extend the algorithm used in [14] to identify multiple thick regions on the skeleton. Species such as Pennycress are taproot systems (as opposed to fibrous root systems), and will require identification of the primary root as the base of the hierarchy instead of a stem. To begin, the primary root can potentially be identified as the longest path within the acyclic skeleton produced by TopoRoot, found by extending the current hierarchy labelling algorithm to start at the base of the hierarchy (e.g. the level below nodal roots).

The final part of this dissertation covered topological simplification of nested shapes. Many improvements to our algorithm for the topological simplification of a single shape can also be potentially applied to the nested shape case, including the improvement of the visual quality of the cuts and fills, the optimality of the per-level solver, and an extension of the implementation to non-cubical complexes. Finally, the inflation / deflation method which is used to generate the cuts and fills currently accounts for a significant portion of the runtime, especially due to the potentially high number of shapes involved. More efficient methods for generating cuts and fills would allow for application of the algorithm to nested sequences with a greater number of shapes with higher topological complexities.

References

[1] Anthony Bishopp and Jonathan Paul Lynch. "The hidden half of crop yields". In: *Nature Plants* 1 (2015).

[2] Luis O. Duque and Arthur Villordon. "Root Branching and Nutrient Efficiency: Status and Way Forward in Root and Tuber Crops". In: *Frontiers in Plant Science* 10 (2019). ISSN: 1664-462X. DOI: 10.3389/fpls.2019.00237. URL: https://www.frontiersin.org/article/10.3389/fpls.2019.00237.

[3] Wouter Vannoppen et al. "How do root and soil characteristics affect the erosion-reducing potential of plant species?" In: *Ecological Engineering* 109 (2017), pp. 186–195.

[4] Joseph Awika. "Major Cereal Grains Production and Use around the World". In: *Advances in Cereal Science: Implications to Food Processing and Health Promotion* 1089 (Nov. 2011), pp. 1–13. DOI: 10.1021/bk-2011-1089.ch001.

[5] J. Lynch. "Root Architecture and Plant Productivity". In: *Plant Physiology* 109.1 (Sept. 1995), pp. 7–13. ISSN: 0032-0889. DOI: 10.1104/pp.109.1.7. eprint: https://academic.oup.com/plphys/article-pdf/109/1/7/35533131/plphys_v109_1_7.pdf. URL: https://doi.org/10.1104/pp.109.1.7.

[6] E. Adeleke et al. "Assessing Root System Architecture of Wheat Seedlings Using A High-Throughput Root Phenotyping System". In: *bioRxiv* (2019). DOI: 10.1101/677955. eprint: https://www.biorxiv.org/content/early/2019/06/21/677955.full.pdf. URL: https://www.biorxiv.org/content/early/2019/06/21/677955.

[7] Iko T. Koevoets et al. "Roots Withstanding their Environment: Exploiting Root System Architecture Responses to Abiotic Stress to Improve Crop Tolerance". In: *Frontiers in Plant Science* 7 (2016). ISSN: 1664-462X. DOI: 10.3389/fpls.2016.01335. URL: https://www.frontiersin.org/article/10.3389/fpls.2016.01335.

[8] Larry M. York et al. "Wheat shovelomics I: A field phenotyping approach for characterising the structure and function of root systems in tillering species". In: *bioRxiv* (2018). DOI: 10.1101/280875. eprint: https://www.biorxiv.org/content/early/2018/03/12/280875.full.pdf. URL: https://www.biorxiv.org/content/early/2018/03/12/280875.

[9] Narendra Narisetti et al. "Semi-automated Root Image Analysis (saRIA)". In: *Scientific Reports* 9 (Dec. 2019), p. 19674. DOI: 10.1038/s41598-019-55876-3.

[10] H. F. DOWNIE et al. "Challenges and opportunities for quantifying roots and rhizosphere interactions through imaging and image analysis". In: *Plant, Cell & Environment* 38.7 (2015), pp. 1213–1232. DOI: https://doi.org/10.1111/pce.12448.

eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/pce.12448. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/pce.12448.

[11] J.S. Perret, M.E. Al-Belushi, and M. Deadman. "Non-destructive visualization and quantification of roots using computed tomography". In: *Soil Biology and Biochemistry* 39.2 (2007), pp. 391–399. ISSN: 0038-0717. DOI: https://doi.org/10.1016/j.soilbio.2006.07.018. URL: https://www.sciencedirect.com/science/article/pii/S003807170600321X.

[12] Hannes Schulz et al. "Plant Root System Analysis from MRI Images". In: *Computer Vision, Imaging and Computer Graphics. Theory and Application*. Ed. by Gabriela Csurka et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 411–425. ISBN: 978-3-642-38241-3.

[13] Sacha Mooney et al. "Developing X-ray Computed Tomography to non-invasively image 3-D root systems architecture in soil". In: *Plant and Soil* 352 (Mar. 2011), pp. 1–22. DOI: 10.1007/s11104-011-1039-9.

[14] Mao Li et al. "Comprehensive 3D phenotyping reveals continuous morphological variation across genetically diverse sorghum inflorescences". In: *New Phytologist* 226.6 (). DOI: 10.1111/nph.16533. URL: https://par.nsf.gov/biblio/10158432.

[15] Mon-Ray Shao et al. "Complementary Phenotyping of Maize Root Architecture by Root Pulling Force and X-Ray Computed Tomography". In: (Mar. 2021). DOI: 10.1101/2021.03.03.433776.

[16] Miguel A. Piñeros et al. "Evolving technologies for growing, imaging and analyzing 3D root system architecture of crop plants". In: *Journal of Integrative Plant Biology* 58.3 (2016), pp. 230–241. DOI: https://doi.org/10.1111/jipb.12456. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/jipb.12456. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/jipb.12456.

[17] Ana Paez-Garcia et al. "Root Traits and Phenotyping Strategies for Plant Improvement". In: *Plants* 4.2 (2015), pp. 334–355. ISSN: 2223-7747. DOI: 10.3390/plants4020334. URL: https://www.mdpi.com/2223-7747/4/2/334.

[18] Melinda Lontoc-Roy et al. "Advances in the acquisition and analysis of CT scan data to isolate a crop root system from the soil medium and quantify root system complexity in 3-D space". In: *Geoderma* 137 (Dec. 2006), pp. 231–241. DOI: 10.1016/j.geoderma.2006.08.025.

[19] Maxime Phalempin et al. "An improved method for the segmentation of roots from X-Ray computed tomography 3D images: Rootine v.2". In: *EGU General Assembly Conference Abstracts*. EGU General Assembly Conference Abstracts. May 2020, 5225, p. 5225. DOI: 10.5194/egusphere-egu2020-5225.

[20] Sebastian Blaser, Steffen Schlüter, and Doris Vetterlein. "How much is too much?—Influence of X-ray dose on root growth of faba bean (Vicia faba) and barley (Hordeum vulgare)". In: *PLOS ONE* 13 (Mar. 2018), e0193669. DOI: 10.1371/journal.pone.0193669.

[21] Randy Clark et al. "Three-Dimensional Root Phenotyping with a Novel Imaging and Software Platform1[C][W][OA]". In: *Plant Physiology* 156 (2011), pp. 455 –465.

[22] Stefan Mairhofer et al. "RooTrak: Automated Recovery of Three-Dimensional Plant Root Architecture in Soil from X-Ray Microcomputed Tomography Images Using Visual Tracking". In: *Plant physiology* 158 (Dec. 2011), pp. 561–9. DOI: 10.1104/pp.111.186221.

[23] Abraham Smith et al. "Segmentation of roots in soil with U-Net". In: *Plant Methods* 16 (Feb. 2020). DOI: 10.1186/s13007-020-0563-0.

[24] Richard Flavel et al. "An image processing and analysis tool for identifying and analysing complex plant root systems in 3D soil using non-destructive analysis: Root1". In: *PLOS ONE* 12 (May 2017), e0176433. DOI: 10.1371/journal.pone.0176433.

[25] Johannes Pfeifer et al. "Rapid phenotyping of crop root systems in undisturbed field soils using X-ray computed tomography". In: *Plant Methods* 11:14 (Aug. 2015). DOI: 10.1186/s13007-015-0084-4.

[26] Sascha Oswald et al. "Combining Neutron and Magnetic Resonance Imaging to Study the Interaction of Plant Rootsand Soil". In: *Physics Procedia* 69 (Dec. 2015), pp. 237–243. DOI: 10.1016/j.phpro.2015.07.033.

[27] Taras Galkovskyi et al. "GiA Roots: Software for the high throughput analysis of plant root system architecture". In: *BMC plant biology* 12 (July 2012), p. 116. DOI: 10.1186/1471-2229-12-116.

[28] Wei Gao et al. "A shape-based method for automatic and rapid segmentation of roots in soil from X-ray computed tomography images: Rootine". In: *Plant and Soil* 441 (2019), pp. 643–655.

[29] Olga Symonova, Christopher N Topp, and Herbert Edelsbrunner. "DynamicRoots: A Software Platform for the Reconstruction and Analysis of Growing Plant Roots". In: *PLOS One* 10.6 (2015).

[30] HANS LAMBERS et al. "Root Structure and Functioning for Efficient Acquisition of Phosphorus: Matching Morphological and Physiological Traits". In: *Annals of Botany* 98.4 (June 2006), pp. 693–713. ISSN: 0305-7364. DOI: 10.1093/aob/mcl114. eprint: https://academic.oup.com/aob/article-pdf/98/4/693/339786/mcl114.pdf. URL: https://doi.org/10.1093/aob/mcl114.

[31] Angela Hodge. “Root decisions”. In: *Plant, Cell Environment* 32.6 (2009), pp. 628–640.

[32] Yun Bao et al. “Plant roots use a patterning mechanism to position lateral root branches toward available water”. In: *Proceedings of the National Academy of Sciences of the United States of America* 111 (June 2014). DOI: 10.1073/pnas.1400966111.

[33] Jonathan P. Lynch. “Steep, cheap and deep: an ideotype to optimize water and N acquisition by maize root systems”. In: *Annals of Botany* 112.2 (Jan. 2013), pp. 347–357. ISSN: 0305-7364. DOI: 10.1093/aob/mcs293. eprint: https://academic.oup.com/aob/article-pdf/112/2/347/17008517/mcs293.pdf. URL: https://doi.org/10.1093/aob/mcs293.

[34] Jonathan Lynch. “Roots of the Second Green Revolution”. In: *Australian Journal of Botany - AUST J BOT* 55 (Jan. 2007). DOI: 10.1071/BT06118.

[35] Mark Tester and Peter Langridge. “Breeding Technologies to Increase Crop Production in a Changing World”. In: *Science (New York, N.Y.)* 327 (Feb. 2010), pp. 818–22. DOI: 10.1126/science.1183700.

[36] Marco Attene, Marcel Campen, and Leif Kobbelt. “Polygon Mesh Repairing: An Application Perspective”. In: *ACM Comput. Surv.* 45.2 (2013). ISSN: 0360-0300. DOI: 10.1145/2431211.2431214. URL: https://doi.org/10.1145/2431211.2431214.

[37] Lin Chen and Gudrun Wagenknecht. “Automated Topology Correction for Human Brain Segmentation”. In: *Medical image computing and computer-assisted intervention : MICCAI ... International Conference on Medical Image Computing and Computer-Assisted Intervention* 9 Pt 2 (2006), pp. 316–23.

[38] David W. Shattuck and Richard M. Leahy. “Automated graph-based analysis and correction of cortical volume topology”. In: *IEEE Transactions on Medical Imaging* 20 (2001), pp. 1167–1177.

[39] Lin Chen and Gudrun Wagenknecht. “Topology Correction for Brain Atlas Segmentation Using a Multiscale Algorithm”. In: *Bildverarbeitung für die Medizin 2007*. Ed. by Alexander Horsch et al. 2007.

[40] Qian-Yi Zhou, Tao Ju, and Shi-Min Hu. “Topology Repair of Solid Models Using Skeletons”. In: *IEEE transactions on visualization and computer graphics* 13 (July 2007), pp. 675–85. DOI: 10.1109/TVCG.2007.1015.

[41] F.S. Nooruddin and G. Turk. “Simplification and repair of polygonal models using volumetric techniques”. In: *IEEE Transactions on Visualization and Computer Graphics* 9.2 (2003), pp. 191–205. DOI: 10.1109/TVCG.2003.1196006.

[42] Stephan Bischoff and Leif P. Kobbelt. *Isosurface Reconstruction with Topology Control*. 2002.

[43] Nikolaus Kriegeskorte and Rainer Goebel. "An Efficient Algorithm for Topologically Correct Segmentation of the Cortical Sheet in Anatomical MR Volumes". In: *NeuroImage* 14.2 (2001), pp. 329–346. ISSN: 1053-8119. DOI: https://doi.org/10.1006/nimg.2001.0831. URL: https://www.sciencedirect.com/science/article/pii/S1053811901908316.

[44] Andrea Tagliasacchi et al. "3D Skeletons: A State-of-the-Art Report". In: *Computer Graphics Forum* 35 (May 2016). DOI: 10.1111/cgf.12865.

[45] Erin W. Chambers et al. "Some Heuristics for the Homological Simplification Problem". In: *CCCG*. 2018.

[46] Zoë Wood et al. "Removing Excess Topology from Isosurfaces". In: *ACM Trans. Graph.* 23 (Apr. 2004), pp. 190–208. DOI: 10.1145/990002.990007.

[47] Florent Ségonne, Jennifer L. Pacheco, and Bruce R. Fischl. "Geometrically Accurate Topology-Correction of Cortical Surfaces Using Nonseparating Loops". In: *IEEE Transactions on Medical Imaging* 26 (2007), pp. 518–529.

[48] Tao Ju, Qian-Yi Zhou, and Shi-Min Hu. "Editing the Topology of 3D Models by Sketching". In: *ACM Trans. Graph.* 26 (July 2007), p. 42. DOI: 10.1145/1276377.1276430.

[49] Allen Hatcher. *Algebraic topology*. Cambridge: Cambridge University Press, 2002, pp. xii+544. ISBN: 0-521-79160-X; 0-521-79540-0.

[50] A. Szymczak and J. Vanderhyde. "Extraction of topologically simple isosurfaces from volume datasets". In: *IEEE Visualization, 2003. VIS 2003.* 2003, pp. 67–74. DOI: 10.1109/VISUAL.2003.1250356.

[51] Markus Leitner et al. "A Dual Ascent-Based Branch-and-Bound Framework for the Prize-Collecting Steiner Tree and Related Problems". In: *INFORMS J. on Computing* 30.2 (2018), 402–420. ISSN: 1526-5528. DOI: 10.1287/ijoc.2017.0788. URL: https://doi.org/10.1287/ijoc.2017.0788.

[52] *11th dimacs implementation challenge in collaboration with icerm: Steiner tree problems.* Tech. rep. url https://dimacs11.zib.de/. 2014.

[53] Stefan Gerth et al. "Semiautomated 3D Root Segmentation and Evaluation Based on X-Ray CT Imagery". In: *Plant Phenomics* 2021 (Feb. 2021), pp. 1–13. DOI: 10.34133/2021/8747930.

[54] Christopher N. Topp et al. “3D phenotyping and quantitative trait locus mapping identify core regions of the rice genome controlling root architecture”. In: *Proceedings of the National Academy of Sciences* 110.18 (2013), E1695–E1704. DOI: 10.1073/pnas.1304354110. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1304354110. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1304354110.

[55] Ni Jiang et al. “Three-Dimensional Time-Lapse Analysis Reveals Multiscale Relationships in Maize Root Systems with Contrasting Architectures”. In: *The Plant Cell* 31.8 (May 2019), pp. 1708–1722. ISSN: 1040-4651. DOI: 10.1105/tpc.19.00015. eprint: https://academic.oup.com/plcell/article-pdf/31/8/1708/36995977/plcell_v31_8_1708.pdf. URL: https://doi.org/10.1105/tpc.19.00015.

[56] Ying Zheng et al. “Detailed reconstruction of 3D plant root shape”. In: Nov. 2011, pp. 2026–2033. DOI: 10.1109/ICCV.2011.6126475.

[57] Dan Zeng et al. “To cut or to fill: a global optimization approach to topological simplification”. In: *ACM Transactions on Graphics* 39 (Nov. 2020), pp. 1–18. DOI: 10.1145/3414685.3417854.

[58] Punam Saha, Gunilla Borgefors, and Gabriella Sanniti di Baja. “A Survey on skeletonization algorithms and their applications”. In: *Pattern Recognition Letters* 76 (Apr. 2015). DOI: 10.1016/j.patrec.2015.04.006.

[59] André Sobiecki, Andrei Jalba, and Alexandru Telea. “Comparison of curve and surface skeletonization methods for voxel shapes”. In: *Pattern Recognition Letters* 47 (2014). Advances in Mathematical Morphology, pp. 147–156. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2014.01.012. URL: https://www.sciencedirect.com/science/article/pii/S0167865514000269.

[60] Punam K. Saha and Bidyut. B. Chaudhuri. “Detection of 3-D Simple Points for Topology Preserving Transformations with Application to Thinning”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 16 (1994), pp. 1028–1032.

[61] Gilles Bertrand and Grégoire Malandain. “A New Characterization of Three-Dimensional Simple Points”. In: *Pattern Recogn. Lett.* 15.2 (1994), 169–175. ISSN: 0167-8655. DOI: 10.1016/0167-8655(94)90046-9. URL: https://doi.org/10.1016/0167-8655(94)90046-9.

[62] Yajie Yan, David Letscher, and Tao Ju. “Voxel Cores: Efficient, Robust, and Provably Good Approximation of 3D Medial Axes”. In: *ACM Trans. Graph.* 37.4 (2018). ISSN: 0730-0301. DOI: 10.1145/3197517.3201396. URL: https://doi.org/10.1145/3197517.3201396.

[63] Yajie Yan et al. “Erosion thickness on medial axes of 3D shapes”. In: *ACM Transactions on Graphics* 35 (July 2016), pp. 1–12. DOI: 10.1145/2897824.2925938.

[64] Johannes Postma et al. “OpenSimRoot: widening the scope and application of root architectural models”. In: *New Phytologist* 215 (Apr. 2017). DOI: 10.1111/nph.14641.

[65] M. A. Ali Khan, Dorcus C. Gemenet, and Arthur Villordon. “Root System Architecture and Abiotic Stress Tolerance: Current Knowledge in Root and Tuber Crops”. In: *Frontiers in Plant Science* 7 (2016).

[66] MERLE T. JENKINS. “HERITABLE CHARACTERS OF MAIZE: XXXIV—Rootless”. In: *Journal of Heredity* 21.2 (Feb. 1930), pp. 79–80. ISSN: 0022-1503. DOI: 10.1093/oxfordjournals.jhered.a103287. eprint: https://academic.oup.com/jhered/article-pdf/21/2/79/2454166/21-2-79.pdf. URL: https://doi.org/10.1093/oxfordjournals.jhered.a103287.

[67] Daniel Gonzalez, Johannes Postma, and Matthias Wissuwa. “Cost-Benefit Analysis of the Upland-Rice Root Architecture in Relation to Phosphate: 3D Simulations Highlight the Importance of S-Type Lateral Roots for Reducing the Pay-Off Time”. In: *Frontiers in Plant Science* 12 (2021). ISSN: 1664-462X. DOI: 10.3389/fpls.2021.641835. URL: https://www.frontiersin.org/article/10.3389/fpls.2021.641835.

[68] Jonathan P Lynch. “Rightsizing root phenotypes for drought resistance”. In: *Journal of Experimental Botany* 69.13 (Feb. 2018), pp. 3279–3292. ISSN: 0022-0957. DOI: 10.1093/jxb/ery048. eprint: https://academic.oup.com/jxb/article-pdf/69/13/3279/25054709/ery048.pdf. URL: https://doi.org/10.1093/jxb/ery048.

[69] H. Xu and J. Barbič. “Signed distance fields for polygon soup meshes”. In: *Proceedings - Graphics Interface* (Jan. 2014), pp. 35–41.

[70] Sophie de Dorlodot et al. “Root system architecture: opportunities and constraints for genetic improvement of crops”. In: *Trends in Plant Science* 12.10 (2007), pp. 474–481. ISSN: 1360-1385. DOI: https://doi.org/10.1016/j.tplants.2007.08.012. URL: https://www.sciencedirect.com/science/article/pii/S1360138507001914.

[71] Hannes Schulz et al. “3D Reconstruction of Plant Roots from MRI Images”. In: vol. 2. Jan. 2012.

[72] Mohammadreza Soltaninejad et al. “Three Dimensional Root CT Segmentation Using Multi-Resolution Encoder-Decoder Networks”. In: *IEEE Transactions on Image Processing* PP (May 2020), pp. 1–1. DOI: 10.1109/TIP.2020.2992893.

[73] Guillaume Lobet et al. “Root System Markup Language: Toward a Unified Root Architecture Description Language”. In: *Plant physiology* (Jan. 2015). DOI: 10.1104/pp.114.253625.

[74] Andrea Schnepf et al. “CRootBox: A structural-functional modelling framework for root systems”. In: *Annals of botany* 121 (Feb. 2018). DOI: 10.1093/aob/mcx221.

[75] Dominique Attali et al. “Homological reconstruction and simplification in R3”. In: *Computational Geometry* 48.8 (2015), pp. 606–621. ISSN: 0925-7721. DOI: https://doi.org/10.1016/j.comgeo.2014.08.010. URL: https://www.sciencedirect.com/science/article/pii/S0925772114000923.

[76] Xiao Han et al. “Topology correction in brain cortex segmentation using a multi-scale, graph-based algorithm”. In: *IEEE Transactions on Medical Imaging* 21.2 (2002), pp. 109–121. DOI: 10.1109/42.993130.

[77] Florent Ségonne, Jenni Pacheco, and Bruce Fischl. “Geometrically Accurate Topology-Correction of Cortical Surfaces Using Nonseparating Loops”. In: *IEEE transactions on medical imaging* 26 (Apr. 2007), pp. 518–29. DOI: 10.1109/TMI.2006.887364.

[78] C. Heine et al. “A Survey of Topology-based Methods in Visualization”. In: *Computer Graphics Forum* 35.3 (2016), pp. 643–667. DOI: https://doi.org/10.1111/cgf.12933. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12933. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12933.

[79] P.-T. Bremer et al. “A topological hierarchy for functions on triangulated surfaces”. In: *IEEE Transactions on Visualization and Computer Graphics* 10.4 (2004), pp. 385–396. DOI: 10.1109/TVCG.2004.3.

[80] Tino Weinkauf, Yotam Gingold, and Olga Sorkine. “Topology-based Smoothing of 2D Scalar Fields with C1-Continuity”. In: *Computer Graphics Forum* 29.3 (2010), pp. 1221–1230. DOI: https://doi.org/10.1111/j.1467-8659.2009.01702.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2009.01702.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2009.01702.x.

[81] Giuseppe Patanè, Michela Spagnuolo, and Bianca Falcidieno. “Topology- and Error-Driven Extension of Scalar Functions from Surfaces to Volumes”. In: *ACM Trans. Graph.* 29.1 (2009). ISSN: 0730-0301. DOI: 10.1145/1640443.1640447. URL: https://doi.org/10.1145/1640443.1640447.

[82] David Günther et al. “Fast and Memory-Efficient Topological Denoising of 2D and 3D Scalar Fields”. In: *IEEE Transactions on Visualization and Computer Graphics* 20 (Dec. 2014), to appear. DOI: 10.1109/TVCG.2014.2346432.

[83] 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*. SCG ’06. Sedona, Arizona, USA: As-

sociation for Computing Machinery, 2006, 127–134. ISBN: 1595933409. DOI: 10.1145/1137856.1137878. URL: https://doi.org/10.1145/1137856.1137878.

[84] Ulrich Bauer, Carsten Lange, and Max Wardetzky. "Optimal Topological Simplification of Discrete Functions on Surfaces". In: *Discrete and Computational Geometry* 47 (Mar. 2012), pp. 347–377. DOI: 10.1007/s00454-011-9350-z.

[85] Julien Tierny and Valerio Pascucci. "Generalized Topological Simplification of Scalar Fields on Surfaces". In: *IEEE Transactions on Visualization and Computer Graphics* 18.12 (2012), pp. 2005–2013. DOI: 10.1109/TVCG.2012.228.

[86] Jonas Lukasczyk et al. "Localized Topological Simplification of Scalar Data". In: *IEEE Transactions on Visualization and Computer Graphics* PP (Oct. 2020). DOI: 10.1109/TVCG.2020.3030353.

[87] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. "Topological Persistence and Simplification". In: *Discrete & Computational Geometry* 28 (2002), pp. 511–533.

[88] Julien Tierny et al. "The Topology ToolKit". In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2018), pp. 832–842. DOI: 10.1109/TVCG.2017.2743938.

Appendix

A.1 Proof of Proposition 1

We start by associating a set of cells (at all dimensions) to each node and showing some properties of these sets. Define $_s$ as the subset of cells in the closure (O_s) that are not faces of any 3-cell ranking higher than O_s. Note that $O_s \subseteq_s \subseteq (O_s)$ by definition. Recall that two sets of cells are *connected* if some cell in one set is the face of a cell in the other set. We can show that, The following properties hold:

1. For any cell $c \in$, there exists a unique node $s \in V$ such that $c \in_s$. That is, $_s$ of all nodes $s \in V$ form a disjoint decomposition of .

2. For any node $s \in V$, $_s$ is connected.

3. For any two nodes $s, t \in V$, they are connected by an edge if and only if $_s$ and $_t$ are connected.

Proof. We prove each property in turn.

1. Consider the set H of highest-ranking 3-cells that c is a face of. If H contains more than one 3-cell, then all 3-cells in H are R-connected, because c is not a face of any other higher-ranking 3-cells. Hence all 3-cells in H belong to the same R-connected component O_s for some node $s \in V$.

2. Since $O_s \subseteq_s$ and the remaining cells $_s \setminus O_s$ are faces of (and hence connected to) some 3-cell in O_s, we only need to show that any two 3-cells in O_s are connected via a path of cells in $_s$. Since any two 3-cells in O_s are connected via a path of R-connected 3-cells, we only need to show that two R-connected 3-cells $u, v \in O_s$ share a common face that is in $_s$. By R-connectivity, u, v share a common face c that is

not the face of any 3-cell with rank higher than that of u, v. Hence $c \in_s$, and the property holds.

3. If s, t are connected, then there exists 3-cells $u \in O_s$ and $v \in O_t$ such that they share a common face c that is not a face of another 3-cell ranking higher than u or v. Hence c belongs to either $_s$ or $_t$, implying that $_s$ and $_t$ are connected. Conversely, if $_s$ and $_t$ are connected, and without loss of generality, there exists a cell $c \in_s$ that is a face of a 3-cell $v \in O_t$. Let u be the 3-cell in O_s that c is a face of. Since $c \in_s$, there is no other 3-cell with rank higher than u that has c as a face. Hence u, v are R-connected, and hence nodes s, t are connected by a graph edge.

□

We now prove Proposition 1:

Proof. (**Proposition 1**) We first show that the following statement is true for any *proper* labelling L: if two nodes $s, t \in V$ are connected such that s has a higher rank than t, and if $L(t) = 1$, then $L(s) = 1$. Note that s can be either a kernel node or cut node. If s is a kernel node, then $L(s) = 1$ regardless of L. If s is a cut node, then t can only be a fill node, and therefore $L(s) = 1$ since L is proper.
To prove the Proposition, it suffices to show that the union of $_s$ over all 1-labelled nodes s is the modified shape X, or:

$$\cup_{L(s)=1\ s} = (\cup_{L(s)=1} O_s) \tag{A.1}$$

for any proper labelling L. If this is true, then the three properties in Lemma A.1 ensure that the connected components of X, and of its complement $\backslash X$, have one-to-one correspondence with the connected components of the 1-labelled and 0-labelled subgraphs.
To show that equality A.1 holds, we need to show that for any node $s \in V$ such that $L(s) = 1$, and any 3-cell $u \in O_s$, all faces of u (at all dimensions) must have been included in the union $\cup_{L(s)=1} s$. We consider two cases for each face c of u. If $c \in_s$, then obviously

$c \in \cup_{L(s)=1} s$. Otherwise, by Lemma A.1, there exists some other node $t \in V$ such that $c \in_t$, t has higher rank than s, and s, t are connected. As shown at the beginning of this proof, we conclude that $L(t) = 1$, and hence $c \in \cup_{L(s)=1} s$. □

A.2 Proof of Proposition 4

We start with a few observations on the 0-sets and 1-sets. Recall that, for each candidate $x \in X_i$, its 0-set (x) (resp. 1-set (x)) is the union of x and all candidates in X_j for $j < i$ (resp. $j > i$) that are path-connected with x.

We introduce two more definitions. Given a set of candidates S (possibly across multiple shapes), we say it is *0-inclusive* (resp. *1-inclusive*) if $(x) \subseteq S$ (resp. $(x) \subseteq S$) for any candidate $x \in S$. We show that: The following statements hold for any candidates x, y and candidate sets S_1, S_2:

1. $x \in (y)$ if and only if $y \in (x)$.
2. (x) is 0-inclusive and (x) is 1-inclusive.
3. If S_1, S_2 are both 0-inclusive (resp. 1-inclusive), than $S_1 \cup S_2$ is also 0-inclusive (resp. 1-inclusive).

Proof. We prove each statement in turn. We denote by $\pi(x)$ the index of the shape to which the candidate x belongs (i.e., $x \in X_{\pi(x)}$).

1. By definition of the 0-set, $x \in (y)$ if and only if x and y are path-connected and $\pi(x) < \pi(y)$. These are exactly the conditions for $y \in (x)$.
2. To show (x) is 0-inclusive, we need to show that $(y) \subseteq (x)$ for any $y \in (x)$. Consider such a y and any $z \in (y)$. Since z is path-connected with y, which is in turn path-connected with x, z must be path-connected with x. Furthermore, $\pi(z) < \pi(y) <$

$\pi(x)$. We conclude that $z \in (x)$, and hence $(y) \subseteq (x)$. A symmetric argument shows that (x) is 1-inclusive.

3. We only need to show the 0-inclusive case; the 1-inclusive case is symmetric. For any $x \in S_1 \cup S_2$, x must either lie in S_1 or S_2. Suppose the former (the latter case is identical). Since S_1 is 0-inclusive, $(x) \subseteq S_1$ and hence $(x) \subseteq S_1 \cup S_2$, and therefore $S_1 \cup S_2$ is 0-inclusive as well.

□

Using these observations, we can show that each state created by the algorithm has two disjoint constraint sets that are respectively 0-inclusive and 1-inclusive: The following statements hold for any state $\{F_0, F_1, L\}$ pushed into the queue Q in Algorithm 1:

1. $F_0 \cap F_1 = \emptyset$
2. F_0 is 0-inclusive and F_1 is 1-inclusive.

Proof. We will prove by induction. Both (1,2) trivially hold for the initial state, where $F_0 = F_1 = \emptyset$. Consider a state $\{F_0, F_1, L\}$ popped by the algorithm. Assuming that (1,2) hold for F_0, F_1, we will show that F_0' obtained in Line 9 is (i) 0-inclusive and (ii) satisfying $F_0' \cap F_1 = \emptyset$. A symmetric argument would then show that F_1' obtained in Line 10 is 1-inclusive and satisfies $F_1' \cap F_0 = \emptyset$.

- (i): Since F_0 is 0-inclusive and so is (x) (be Lemma A.2 (2)), their union F_0' is also inclusive (be Lemma A.2 (3)).
- (ii): Since $F_0 \cap F_1 = \emptyset$, we only need to show that $(x) \cap F_1 = \emptyset$. We first show that $x \notin F_1$. Suppose, on the contrary, that $x \in F_1$. Since $y \in (x)$ (because x, y overlap), and F_1 is 1-inclusive, we have $y \in F_1$, which contradicts with $L(y) = 0$. We next show that $(x) \cap F_1 = \emptyset$. Suppose on the contrary that there exists some $y \in (x)$ such that $y \in F_1$. By Lemma A.2 (1), $x \in (y)$, and hence $x \in F_1$, which contradicts to the earlier statement.

□

We are finally ready to prove Proposition 4:

Proof. We first show termination. By Lemma A.2, the two constraint sets in each state are always disjoint, and hence a labelling under these constraints can always be found. Therefore the algorithm can always proceed. The states explored by the algorithm can be organized into a binary tree, where the popped state at each iteration becomes the parent of the two pushed states. Note that, since $F_0 \subset F_0'$ and $F_1 \subset F_1'$, the union of each child state's sets of constrained candidates is strictly larger than that of its parent. Since the total number of candidates is finite, the depth of the binary tree is finite, and so is the size of the tree. On the other hand, it is easy to see that the labelling associated with a state at the very bottom level of the tree, where all candidates are constrained, must be conflict-free (due inclusiveness of the constraints). Therefore the algorithm must terminate and return a conflict-free labelling (on Line 6) within finite number of iterations.

Next we show optimality. We call a state $\{F_0, F_1, L\}$ *compatible* with a labelling L' if L' labels all candidates in F_0 as 0 and all candidates in F_1 as 1. Note that, assuming optimality of , we always have $E(L) \leq E(L')$, because $L = (F_0, F_1)$ has the least energy among all labellings that satisfy the constraint sets F_0, F_1, and L' is one of such labellings.

Let L^* be a conflict-free labelling with minimal energy. We make a key observation that, at any time during the algorithm, there is at least one state in the queue Q that is compatible with L^*. We prove by induction. At the beginning of the algorithm, Q holds a single state where $F_0 = F_1 = \emptyset$, which is compatible with any labelling. As the algorithm proceeds, we need to show that, after a state $\{F_0, F_1, L\}$ compatible with L^* is popped from Q, at least one of the two pushed states remains compatible with L^*. To do so, note that since L^* is conflict-free, either $L^*(x) = 0$ or $L^*(y) = 1$ (both could hold). We first consider the case that $L^*(x) = 0$. By definition of the 0-set, L^* must assign label 0 to all candidates in (x) to avoid conflicts. Since L^* already labels all of F_0 as 0 (due to compatibility), L^* therefore labels all of $F_0' = (x) \cup F_0$ as 0. This makes one of the pushed states,

$\{F_0', F_1, (F_0', F_1)\}$, compatible with L^*. Similarly, if $L^*(y) = 1$, the other pushed state, $\{F_0, F_1', (F_0, F_1')\}$, is compatible with L^*.

To complete the proof, let L' be the labelling of the state in Q compatible with L^* when the algorithm terminates. Note that L' could be the same as the labelling L being returned. We have the following relation:

$$E(L) \leq E(L') \leq E(L^*)$$

The first inequality is due to the fact that L is the least-energy labelling in Q, and the second inequality is due to compatibility. By minimality of $E(L^*)$ and since L is also conflict-free, the equality must hold in this relation, which implies that L also has the minimal energy among all conflict-free labellings. □