UNIVERSITY OF CALIFORNIA SAN DIEGO

An algorithmic toolbox for plant phenotyping with 3D point clouds

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Illia Ziamtsov

Committee in charge:

Professor Saket Navlakha, Chair
Professor Hao Su, Co-Chair
Professor Henrik Christensen
Professor David Kriegman
Professor Martin Yanofsky

2022

The dissertation of Illia Ziamtsov is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2022

DEDICATION

To my parents.

EPIGRAPH

Discipline is the bridge between

dreams and reality.

—Jim Rohn

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I want to acknowledge the chair of my committee and my adviser Dr. Saket Navlakha for all his help, support and patience for me through the last couple of years. His openness to communicate and discuss my ideas along with giving me freedom of direction proved to be very influential.

Chapters 2, 3, and 4, use materials that appear in Plant Physiology 181(47). Ziamtsov, Illia; Navlakha, Saket, 1425-1440 2019. The dissertation author was the primary investigator and author of this paper.

Chapter 5 has been submitted for publication of the material as it may appear in Remote Sensing 2021, Ziamtsov, Illia; Faizi, Kian; Navlakha, Saket. The dissertation author was the primary investigator and author of this paper.

Chapter 6 uses materials that were published in Bioinformatics 36(12). Ziamtsov, Illia; Navlakha, Saket, 3949-3950 2020. The dissertation author was the primary investigator and author of this paper.

VITA

2011	B. S. in Computer Science *summa cum laude*, University of North Carolina Wilmington
2012-2015	Graduate Teaching Assistant, Purdue University
2015	M. S. in Computer Graphics, Purdue University
2015-2021	Graduate Research Assistant, University of California San Diego
2022	Ph. D. in Computer Science, University of California San Diego

PUBLICATIONS

Yoo, I., Massih, MA., Ziamtsov, I., Hassan, R., Benes, B.: "Motion retiming by using bilateral time control surfaces", *Computers & Graphics (47)*, 59-67, 2015.

Ziamtsov, I., Navlakha, S.: "Machine learning approaches to improve three basic plant phenotyping tasks using three-dimensional point clouds", *Plant Physiology 181(4)*, 1425–1440 2019.

Ziamtsov, I., Navlakha, S.: "Plant 3D (P3D): a plant phenotyping toolkit for 3D point clouds", *Bioinformatics 36 (12)*, 3949-3950, 2020.

Ziamtsov, I., Faizi, K., Navlakha, S.: "Branch-Pipe: Improving graph skeletonization around branch points in 3D point clouds", *Remote Sensing 2021*, (under review)

ABSTRACT OF THE DISSERTATION

An algorithmic toolbox for plant phenotyping with 3D point clouds

by

Illia Ziamtsov

Doctor of Philosophy in Computer Science

University of California San Diego, 2022

Professor Saket Navlakha, Chair
Professor Hao Su, Co-Chair

Developing automated methods to efficiently process large volumes of point cloud data remains a grand challenge for 3D plant phenotyping applications. Phenotyping is one of the key pre-cursors towards developing more efficient breeding strategies and improvement of crop yields. Using 3D point clouds for plant phenotyping helps to eliminate problems that are usually encountered by performing phenotyping with images. However, point clouds come with their own set of problems, which include registration errors, missing data, sensor noise, and scalability. Point clouds of plants also typically have problems related to "natural" noise caused by small features such as trichomes or natural shape irregularities; further, handling the diversity of organic

structures observed across plant species is challenging. All these nuances need to be considered to ensure quality extraction of phenotyping traits.

Here, we develop a collection of graph-theoretic and machine learning methods to tackle three primary challenges in plant phenotyping: leaf/branch classification, leaf counting, and branch skeletonization. For classification, we assess and validate the accuracy of our methods on a dataset of 54 3D shoot architectures representing multiple species, growth conditions, and developmental time-points. Using deep learning, we achieved 97.8% accuracy on classifying leaves versus branches; critically, we also demonstrate the robustness of our method to growth conditions and species that have not been trained on, which is important in practical applications but is often untested. For leaf counting, we developed an enhanced region growing algorithm to reduce over-segmentation; this method achieved 86.6% accuracy, outperforming prior methods developed for this problem. Finally, for branch skeletonization, we developed an enhanced tip detection technique that ran an order of magnitude faster and generated more precise skeleton architectures compared to prior methods. In addition, we improved the quality of angles produced by skeletonization. Overall, our improvements enable higher throughput and accurate extraction of phenotypic properties from 3D point cloud data. Finally, all the algorithmic methods presented in this thesis were packaged in a stand alone GUI application, called Plant3D (P3D).

Chapter 1

Introduction

The world's demand for food is rapidly increasing along with the population of our planet. It is expected that the world population will reach 9 billion people by the year 2050 [GBC+10]. According to a 2015 report by the United Nations Food and Agriculture Organization [RCG+15], our global demand for cereals is expected to double by 2050. In recent decades, there has been a great deal of improvement in crop yields, mostly attributed to progress in breeding and genetics. However, there is evidence that indicates that these improvements are insufficient to meet future food demands [BRA+14]. If this is true, then the amount of agricultural land will have to be expanded. The expansion of agricultural land has large negative consequences. In the past, it has been shown that expansion of agricultural land has a significant effect on climate change, pollution, and biodiversity loss. In addition, recent and abrupt changes in climate have had a negative effect on crop yields. Thus, agricultural expansion is both affected by, and contributes to, climate change [AC08]. The current situation is further complicated by the fact that there is a limited amount of soil which can be used for agriculture and a limited amount of fresh water for irrigation. Agriculture already consumes about 70% of fresh water withdrawals [CRT10].

To limit further expansion of agricultural land and mitigate the negative effects of global warming, crops will not only need to produce greater yields, but also become more resistant to

harsher environments. Recently, there has been a large investment in research to breed plants that are more stress-tolerant [ZWR+15]. ***Genotyping*** is the process of determining an organism's DNA sequence, especially to assess variation with respect to a population. ***Phenotyping*** is the process of identifying and quantifying an organism's observable traits. In plants, common phenotyping features of interest include the number, size, and shape of leaves [WZC+16, HWQ+18]; plant height and growth rates [MBdS+17]; and branch lengths, diameters, and angles [BABA+17]; amongst others. Extracting plant features is a first step towards quantifying biomass and yield [MBR16], assessing flowering time and drought tolerance [MFST16], mapping genotypes to phenotypes [BBB+17, Set12], and studying morphological properties of plant architectures [CPC+17, CPCN17, BABA+17, LFC+18a, PL96]. In theory, genotyping and phenotyping work together and complement one another. For example, once stress-resistant phenotypes are observed, researchers can then search for the regions of the plant's genome responsible for those particular traits. This process is known as ***forward genetics***. However, there is a significant knowledge gap between them, with phenotyping currently being a bottleneck [ZWR+15]. Emerging technologies for high-throughput plant architecture capture have further increased the demand for automated phenotyping methods [FT11, FS13]. Clearly, new and improved approaches for rapidly identifying and propagating favorable plant traits are needed.

There are many techniques used for non-invasive plant phenotyping, ranging from camera-based imaging [HHK+15, NSM+15] to 3D laser scanning [PDMK13, PBM+14], with advantages and disadvantages to each in terms of cost, accuracy, and throughput. For example, 2D imaging methods remain a cheap, high-throughput solution [PSNEC17, US17, GES+18, ZZR18] and can be used in conjunction with structure from motion techniques [Sze10] to generate point clouds [SKBR15, GDHB17, DMS12] or mesh [dMFK10, PSB+12] representations of plants, the latter providing additional areal and volumetric information. This approach, however, faces several issues including camera calibration, image registration, object occlusion, and inconsistencies in scaling and illumination. Images that contain an extra depth component (called

RGB-D) have also been used in the context of plant phenotyping that helps overcome some of these challenges [CRL+12, XWCL15]. Further, some platform-specific solutions have been proposed [NFL+16, GPF+18], though general purpose solutions remain elusive.

On the other hand, light detection and ranging (LiDAR) and 3D laser scanning devices are considerably more expensive but generate point cloud representations of plant geometry with up to micron-level precision. Compared to most 2D methods, 3D can provide a more comprehensive and detailed view of the entire architecture. Prior works have used 3D scanning to study, for example, growth and development [LFM+13] and light interception of leaves [GSA+12b], and these technologies are now being deployed in the field [SLP17, ARAM+13]. However, these techniques also face challenges such as non-uniform sampling of points, outliers and missing data [LFM+13], scalability, and automation [CWT+18]. Thus, developing robust computational methods to study plant shapes from 3D point cloud data has become increasingly important.

This thesis concentrates on developing efficient algorithms for capturing and evaluating meaningful traits from 3D point clouds of plants. In particular, we study three basic computational problems that are essential for downstream high-throughput phenotyping: classification of point cloud data into sets of branch and leaf points, leaf counting, and branch skeletonization.

Classification: The ability to divide a point cloud of a plant according to the appropriate plant's organs is essential for further phenotyping efforts. Without this step, only general phenotyping measurements regarding the entire plant can be made such as overall size and volume. Often times researches are interested in one particular part of a plant such as a head of a wheat plant [GWC+21] or a branching network of a cotton plant [PSB+12]. Moreover, selection of useful traits for optimization of food production is the key in plant breeding. Accurate selection of traits depends on robust quantification of various plant organs. Thus, accurate plant organ classification is crucial. To meet the demand of high-throughput phenotyping, fast and accurate classification still remains challenging.

Leaf counting: For many plant species, leaves comprise the majority of the surface morphology and often are the biggest structural elements [LCS+19]. Monitoring of the shape and physical appearance of leaves allows farmers to check on a plant's overall health and growth status [LCT+18]. Also the morphology of leaves along with texture and color can signal the type of stress a plant is under so appropriate measures can be taken. In order to analyze morphology of single leaflets, leaves need to be separated into individual leaves. Accurate segmentation of leaves still remains a challenge mostly due to tendency of leaves to overlap in large canopies.

Skeletonization: Branching skeleton is a one dimensional line that ideally passes through the center of all branches. The skeleton is a concise representation of the plant's architecture. An accurate skeleton of a plant can be used for automatic analysis of plant's growth [CWT+18] as well as robotic pruning of branches [CAE+16]. However, an accurate extraction of a skeleton still remains a difficult problem due to noise, holes, and large diversity of plant data.

In an attempt to be robust to the abundance of natural variation found in biology, the work presented in this thesis was tested on multiple species and across various development time points and growth conditions. Namely, the dataset we used consists of 3D laser-scanned shoot architectures from two dicot species (tomato, tobacco), each grown across three developmental time-points, and five growth conditions (ambient, high-heat, shade, drought, high-light).

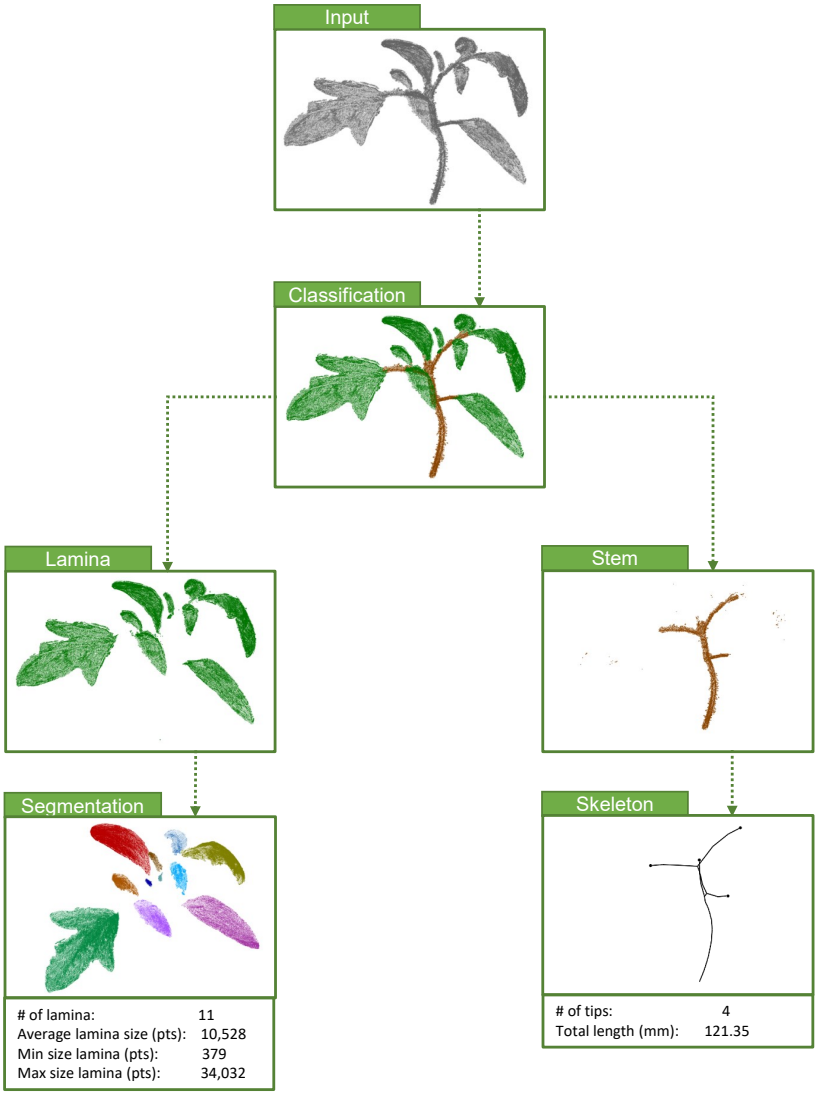

Figure 1.1: **Overview.** A summary of the three problems tackled: classification of leaf versus branch points, leaf counting/segmentation, and branch skeletonization.

1.1 Thesis outline

This section covers a brief overview of the content that follows in chapters below. The remaining of chapter 1 describes the dataset used to test all the methods in this thesis and establishes some common mathematical notations. Chapter 2 focuses on classification of points

that belong to different plant organs. In particular, we classified points into two classes: leaf (lamina) points and branch points. Classification is the first and essential step, the results of which allow us to apply different algorithms to different classes of points. In chapter 3 we focus on leaf counting. We extended a region growing algorithm to improve the robustness of leaf segmentation to natural variation, including ripples, wrinkles, and hairs. Chapter 4 talks about extraction of a skeleton graph from 3D points that were classified as branch points. In this chapter, we present a method that improves tip and root detection to generate more precise skeletons. Chapter 5 continues with skeleton graph improvements paying special attention to the fork area of a plant. The method described in this chapter builds on results from chapter 4 and helps to improve the quality of estimated angles. Finally, chapter 6 presents the *Plant3D* (P3D) toolbox, which is a stand-alone and self containing application that contains all the methods listed in this thesis and more. The application comes with an intuitive graphical user interface and a few trained models based on our dataset as a starting point. Our aim for P3D was to design a tool that would be easy to install and use for non-CS communities, such as plant and agriculture biologists. The precedence of methods in this thesis and their relationships are illustrated in Figure 1.1.

1.2 Dataset of 3D plant shoot architectures

We used 3D laser scanning to generate point cloud representations of 54 full shoot architectures. We used the FaroArm EDGE Model 14000, which is a non-invasive laser scanner and provides micron-level resolution. We scanned plants from two species: 36 tomato plants (*Solanum lycopersicum cv m82D*) and 18 tobacco plants (*Nicotiana benthamiana*). Tomato plants were grown in 5 conditions (control/ambient light, high-heat, high-light, shade, and drought), and tobacco plants were grown in 3 conditions (control, high-heat, and shade). There were two replicates per species-conditions pair. These conditions were selected because they represent a range of realistic environments regularly faced by many plants, and for which there is phenotypic

plasticity that challenges accurate phenotyping. Each plant was scanned on developmental days 5, 12, and 20. Each of the 54 architectures was manually classified into sets of branch and leaf points. Full experimental details was previously described [CPC+17, CPCN17].

This dataset encapsulates a diverse set of architectures, with variation in many phenotypic features, including plant size (biomass), number and structure of leaves, and branch lengths and angles. For example, the plant volume, measured by the convex hull of the plant's cloud points, of day 20 tomato plants varied across conditions by up to 50-fold: from 13.78 cm^3 in high-heat to 668.68 cm^3 in high-light [CPCN17]. The number of leaves for tobacco plants varied from 4 to 12 from the young to old plants [CPC+17]. Over all 36 tomato plants, the range in number of cloud points was from 22,411 to 958,991, with an average of 233,917 cloud points per scanned architecture. Similarly, over all 18 tobacco plants, the range was from 3,709 to 806,269 cloud points. For tomato, on average 84.7% of the cloud points of an architecture belonged to leaves and 15.3% for branches. For tobacco, on average 90.1% of the cloud points belonged to leaves and 9.9% to branches. Thus, this dataset represents a diverse set of phenotypes to test the generality of our methods.

1.3 Common notations

This section contains common mathematical symbols. These symbols are referenced throughout this thesis. 3D space can be assumed for most of the concepts and methods listed in the thesis unless specified otherwise.

p_i	a point in 3D with $\{x_i, y_i, z_i\}$ coordinates
$\vec{n}_i$	a normal vector $\{n_{x_i}, n_{y_i}, n_{z_i}\}$ that corresponds to point p_i
$\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$	a point cloud (set of 3D points) of size n
$N = \{\vec{n}_1, \vec{n}_2, \ldots, \vec{n}_m\}$	a set of corresponding normal vectors of size m
r	a radius of a sphere around a point
$\mathcal{P}_i^r$	a set of points $\{p_j \mid \ell_2(p_j, p_i) \leqq r\}$ of size k
$\overline{p}$	a centroid point of a point cloud P^k

$$\overline{p} = \frac{1}{k} \sum_{i=1}^{k} p_i$$

$\overline{N_i}$	an average normal vector of a p_i computed from normals of its neighbors $\mathcal{P}^k$

$$\overline{N_i} = \frac{\sum_j N_j}{|\sum_j N_j|},$$

$\vec{a} \otimes \vec{b}$	tensor product (outer product) between vectors $\vec{a}$ and $\vec{b}$
κ	radial basis kernel function (RBF)

$$\kappa(x, x') = e^{-\frac{||x - x'||^2}{2\sigma^2}}$$

Chapter 2

Classification of branch & lamina points

In this chapter we cover classification of points according to plant organs they represent in a point cloud. Namely, we classify all the points in $\mathcal{P}$ into two classes: leaf (lamina) and branch points. Classification is the first and an essential step for extracting traits. Before traits can be measured and analyzed we need to know which points correspond to what organs of a plant. The rest of the algorithms described in the chapters to follow use results obtained from classification.

Our classification is built on the idea of creating a feature vector f_i for every point $p_i \in \mathcal{P}$. The purpose of a feature vector f_i is to capture local geometry around point p_i. An ideal feature vector for plant phenotyping is fast to compute, does not require a lot of memory, and has enough descriptive power to distinguish different plant organs. After a feature vector has been computed for every point, a classifier (supervised trained model) is used to assign a class (plant organ) to every point. Some of the challenges that make such classification difficult include natural biological variety of shapes for the same organs and imbalance in terms of number of points that correspond to different organs, especially in older plants.

2.1 Classification framework and 3D shape descriptors

As input we are given a point cloud $\mathcal{P}$ of n points. Each point $p_i \in \mathcal{P}$ represents a 3-dimensional location on the surface of the shoot architecture. Each point is also associated with a normal vector, $\vec{n}_i$, indicating the orientation of each point in 3D space. Normal vectors were provided by the 3D scanner, and thus did not need to be determined computationally [RC11]. We associate each point p_i with a local neighborhood of points $\mathcal{P}_i^r$, defined by a sphere of radius r. This neighborhood consists of all p_j such that $\|p_i - p_j\|_2 \leq r$.

Our first goal is to learn a classifier that maps each point p_i to one of two labels: leaf or branch (including stem, petioles, etc). Each point p_i is represented by a feature vector f_i describing the local geometry around the point. In principle, a good descriptor should capture general properties of the two classes (leaves or branches) and the variability within a class, without over-specifying shapes to a particular species or environmental growth condition.

Several geometric feature descriptors have been proposed, both within the plant phenotyping community [PDMK13, WPKM15] and in the computer vision community [AFB+12, GBS+16] more broadly. Below, we summarize the descriptors compared here (Table 2.1).

Table 2.1: **List of 3D shape feature descriptors.** The summary of features used with our dataset. The size column represents the dimensionality of the feature vector per point in the point cloud.

Descriptor	Size	Ref.
Point Feature Histogram (PFH)	125	[RMBB08]
Fast Point Feature Histogram (FPFH)	33	[RBB09]
Radius-Based Surface Descriptor (RSD)	289	[MPB+10]
Signatures of Histograms of Orientations (SHOT)	352	[TSDS10]
Spin Images (SI)	153	[JH99]
Principle Curvatures (PC)	2	[RC11]

Principal curvature (PC [RC11]). Intuitively, the principal curvature of a point p_i is the speed with which a surface around p_i "moves away" from a tangent plane defined at p_i [VMA86]. Thus,

the speed of moving away on a flat surface (e.g., a leaf) should be relatively small, and the speed of moving away on a surface of a cylinder (branch) should be relatively fast, at least along some axis.

To estimate the principal curvature of a point p_i, first, every normal vector $\vec{n}_j$ of $p_j \in \mathcal{P}_i$ is projected to a tangent plane formed by $\vec{n}_i$ and point p_i:

$$\vec{m}_j = (I - \vec{n}_i \otimes \vec{n}_i) \cdot \vec{n}_j, \tag{2.1}$$

where I is the 3x3 identity matrix.

Next we calculate the 3x3 covariance matrix C_i of the $\vec{m}_j$ projections:

$$C_i = \frac{1}{k} \sum_{j=1}^{k} (\vec{m}_j - \vec{\bar{m}}) \otimes (\vec{m}_j - \vec{\bar{m}}), \tag{2.2}$$

where k is the number of points in the neighborhood of p_i and $\vec{\bar{m}}$ is the mean of all $\vec{m}_j$.

We then calculate principal curvatures from the non-zero eigenvalues of C_i. These eigenvalues represent how much variation there is along each direction. Assuming $0 \leq \lambda_1 \leq \lambda_2 \leq \lambda_3$, then for point p_i, the feature $f_i = [\lambda_3, \lambda_2]$ corresponds to the maximum and minimum curvatures, respectively.

Radius-based surface descriptor (RSD [MPB$^+$10]). This method was originally developed to estimate the radius of an object to determine if the object can be grasped by a robotic arm.

Given two points and their normal vectors, we can compute the radius of an assumed circle the two points lie on. The radius between p_i and p_j is approximated by first computing the angle α_{ij} between the normals $\vec{n}_i$ and $\vec{n}_j$, and second, computing the distance between p_i and p_j, which is equal to $\sqrt{2r} \cdot \sqrt{1 - \cos(\alpha_{ij})}$. Solving for r gives us the radius between p_i and p_j.

The final RSD feature can come in two versions. In the first version, f_i will contain two values, the minimum and maximum radii computed between p_i and each point $p_j \in \mathcal{P}_i$. In the

second version, f_i will contain histograms of counts of the angles and distances between p_i and each point in $\mathcal{P}_i$. The second version performed best and we report these results below.

Point feature histograms (PFH [RMBB08]). Rusu et al. [RMBB08] derived a method that generalizes the mean curvature around a point that is more robust to noise. For each point p_i, the method computes four features between all pairs of points $p_s, p_t \in \mathcal{P}_i$. These features provide higher descriptive power by capturing all possible normal interactions between the points in the local surface.

For each pair of points $p_s, p_t \in \mathcal{P}_i$, we first compute three vectors corresponding to the *Darboux frame* [RMBB08]:

$$\text{Darboux}(p_s, p_t) = \begin{cases} \vec{u} = \vec{n}_s \\ \vec{v} = \vec{u} \times \dfrac{(p_t - p_s)}{||p_t - p_s||_2} \\ \vec{w} = \vec{u} \times \vec{v} \end{cases}$$

We then use the Darboux frame to calculate four features for the pair: $\alpha_{st}, \phi_{st}, \theta_{st}, d_{st}$.

$$\begin{aligned} \alpha_{st} &= \vec{v} \cdot \vec{n}_t \\ \phi_{st} &= \vec{u} \cdot \frac{(p_t - p_s)}{||p_t - p_s||_2} \\ \theta_{st} &= \arctan(\vec{w} \cdot \vec{n}_t, \vec{u} \cdot \vec{n}_t) \\ d_{st} &= ||p_t - p_s||_2 \end{aligned}$$

The final feature f_i for point p_i is based off the histograms of these four features.

Fast point feature histograms (FPFH [RBB09]). PFH features are computationally expensive to compute because the four features are calculated between all pairs of points in $\mathcal{P}_i$, which takes

time $O(nk^2)$, where k is the size of the neighborhood. FPFH reduces this complexity to $O(nk)$ by only considering pairs that include p_i. FPFH also adds a distance-based weighing step that reduces the contribution of a feature based on how far the corresponding point is from p_i. The two steps are:

1. Compute features $\alpha_{ij}, \phi_{ij}, \theta_{ij}$ between p_i and p_j, where $p_j \in \mathcal{P}_i$.

2. Scale each feature via a distance-based weighting. For example, for α:

$$\alpha_{ij} = \alpha_{ij} + \frac{1}{k} \sum_{j=1}^{k} \frac{1}{d_{ij}} \cdot \alpha_{ij},$$

and similarly for the other two features. Here, d_{ij} equals the Euclidean distance between p_i and p_j. The result is a feature vector f_i that is based off the histograms for each feature.

Signature of histogram of orientations (SHOT [TSDS10]). This descriptor captures local topology around a point by computing a histogram of the orientations of points in $\mathcal{P}_i$. The first stage of the descriptor is to establish a local reference frame, M_i, around point p_i. The goal of this frame is to make the feature locally invariant to geometric transformations. The frame is computed using the eigenvalue decomposition of the weighted covariance matrix:

$$M_i = \frac{1}{\sum_{p_j \in \mathcal{P}_i} (r - d_{ij})} \sum_{p_j \in \mathcal{P}_i} (r - d_{ij})(p_j - p_i)(p_j - p_i)^T, \tag{2.3}$$

where $d_{ij} = ||p_j - p_i||_2$. The directions of the reference frame are aligned to the directions where the majority of the points in the local neighborhood lie.

Next, we divide the neighborhood into multiple sub-volumes of a sphere. The sub-volumes are divided along its three spherical axes and a local histogram is constructed for every sub-volume. Each histogram contains bins of normalized point counts of points that lie within the angle between $\vec{n}_i$ and every normal vector for each point in the sub-volume. The final feature f_i is

a concatenation of all sub-volume histograms, with a quadrilinear interpolation applied to remove abrupt changes in histogram boundaries.

Spin image (SI [JH99]). Unlike SHOT, where the neighborhood is aligned with a local reference frame, the neighborhood in the spin image descriptor is aligned with a local reference axis. The local reference axis is established by using the normal vector of p_i.

Intuitively, imagine a cylinder with a cylindrical axis defined by $\vec{n}_i$. The cylinder is sub-divided into bins along its height and radius. Dividing along the height produces sub-volumes of smaller cylinders. Dividing along the radius poduces sub-volumes of rings. The final feature f_i consists of counts of points in each sub-volume (after interpolation and distance-based weighting are applied, as above).

2.2 Testing and validation framework

The ground-truth classification was created by manually labeling each point in each architecture as either a leaf or branch point using the PolyWorks software tool. Because our data does not have a color component, there was some ambiguity in this labeling; for example, at locations near a branch where a leaf emerges, it was difficult, even for humans, to precisely delineate where the leaf begins and where the branch ends. We discuss this further in Classification results.

We tested performance using 6-fold cross-validation. We chose 6-fold because our labeled tomato dataset consists of 36 plants, thus each fold contains exactly 6 plants. We report the mean accuracy and standard deviation over all folds. No pre-processing steps were applied to the point cloud data prior to computation of features and classification. This was by design to test robustness to noisy and imperfect data that may be expected in real-time scanning applications.

There are many classifiers and performing cross-validation using all feature $\times$ classifier combinations is laborious. Thus, we first used cross-validation with a deep learning classifier [LBH15] to determine the best performing feature (FPFH; Classification results, Figure 2.1), and we then tried many different classifiers using this feature. Specifically, we compared four classifiers: deep learning, linear SVM, K-nearest neighbors (KNN), and random forest. Prior work has used FPFH features for plant classification in conjunction with SVMs [PDMK13] and k-means [WPKM15].

For each classifier, we performed a grid search to find the best model parameters, evaluated using 6-fold cross-validation. This search included different values of the neighborhood radius, $r \in [1.0, 2.0, \ldots, 12.0]$. For linear SVM, we tested values for C, where $C = \log_2(i), i \in [-8, -7, -6, \ldots, 7]$. For deep learning, we varied the number of training steps and the architecture of the neural network. The steps tested were 2000, 20000, and 200000. We tried six different architectures with as few as one hidden layer, as well as deeper networks with up to 10 hidden layers with various number of nodes in each layer [GBC16]. The best architecture contained an input layer, three hidden layers with 33, 66, and 33 nodes, respectively, followed by the output layer. For KNN, we tested K values of 7, 77, 277, and 777. Finally, for the random forest classifier, we used 5, 25, 50, or 100 trees.

2.3 Classification results

Here we ask two questions: What combination of features and classifiers leads to robust classification results? How do these models perform when tested on plants grown in never-before-seen conditions or on plants from a different species?

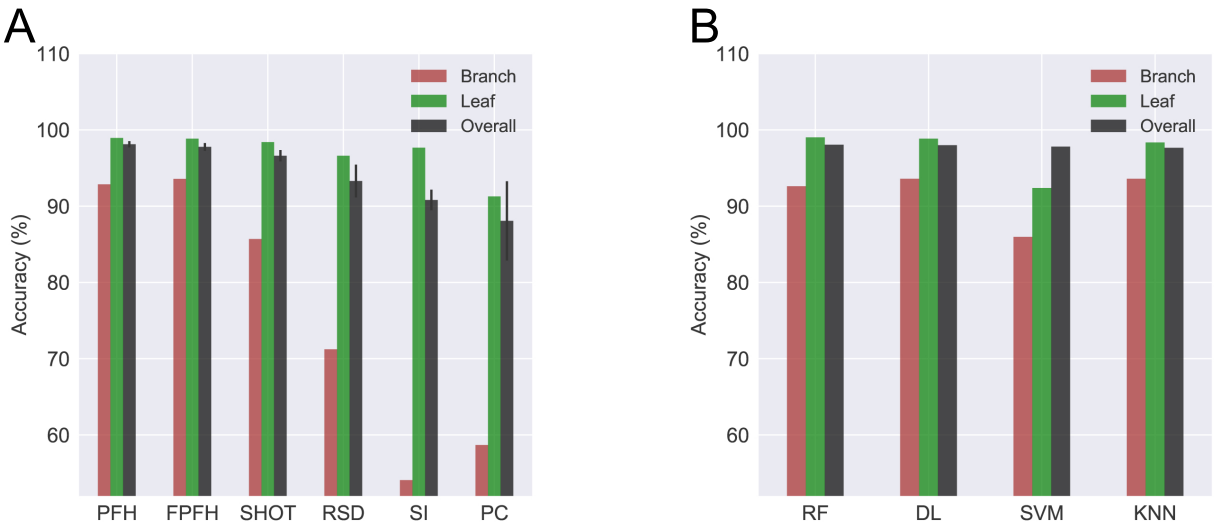

Figure 2.1: **Impact of different classifiers and descriptor features on leaf versus branch classification accuracy.** A) Results of 6-fold cross-validation for each descriptor using an underlying deep learning classifier. Error bars indicate standard deviation across folds. PFH and FPFH features perform best. B) Results of 6-fold cross-validation for different classifiers using FPFH descriptors. The three bars for each descriptor/classifier show accuracy by class (branch, leaf, and overall for both classes).

2.3.1 Features

Using 6-fold cross-validation on the 36 tomato plants, we found that point feature histograms (PFH, and their faster version, FPFH) achieved the highest accuracy in classifying between leaf and branch points: $98.12\% \pm 0.39\%$ and $97.77\% \pm 0.50\%$, respectively (Figure 2.1A). While PFH was slightly more accurate than FPFH (by 0.35%), PFH's complexity is $O(nk^2)$ [Rus09] as opposed $O(nk)$ for FPFH. This difference corresponds to days to perform cross-validation versus a few hours for FPFH, and thus FPFH achieves a better trade-off between scalability and accuracy.

Because our dataset is not balanced between classes — $\approx 85\%$ of points are leaves, $\approx 15\%$ are branches — we computed the accuracy for each feature by class (Figure 2.1A). Here again, PFH and FPFH perform similarly and dominate the other features. For leaf points, the accuracies per class was 98.94% and 98.85%, respectively for PFH and FPFH; for branch

points, the accuracies were 92.86% and 93.60%, respectively. The next best feature, SHOT, performs comparable for leaves, but significantly worse (85.69%) for branch points. We see the largest variation in performance for branch point classification, ranging from 93.60% (FPFH) to 54.1% (spin image features). We note that an improvement in even 1% accuracy corresponds to potentially thousands of cloud points, and thus can be significant.

It might be reasonable to say that the structure of a shoot architecture is roughly composed of two types of primitive shapes: planes (leaves) and cylinders (branches). Thus, simple and fast geometric descriptors, such as normal vectors and curvature values, should in principle be sufficient to accurately segment leaves and branches. However, our empirical results above show that these features do not in practice achieve high segmentation accuracy, likely due to irregularities in these basic shapes and noise. For example, principal curvatures achieved an overall accuracy of only 88.1% $\pm$ 5.2. Different stressors, such as temperature and drought, can also affect leaf shape, which makes these primitives somewhat simplistic. Thus, in the results described below, we only consider FPFH.

2.3.2 Classifiers

The classification results above were achieved using a deep learning classifier. Prior work has used a variety of classifiers for segmentation. For example, Paulus et al. [PDMK13] used FPFH in conjunction with SVMs, and Wahabzada et al. [WPKM15] used FPFH with k-means. Here, we study the effect of using different classifiers in conjunction with FPFH features for classification accuracy. We tested the following classifiers: K-nearest neighbors (KNN), random forest (RF), support vector machines (SVM), and deep learning (DL).

Several of these classifiers performed comparably (Figure 2.1B); however, when comparing the nature of the errors of each method, we found that DL made the most "reasonable" mistakes. Specifically, we visualized the errors made by each method and noticed qualitatively that most of the errors made by DL were in ambiguous regions of the plant where leaves emerge

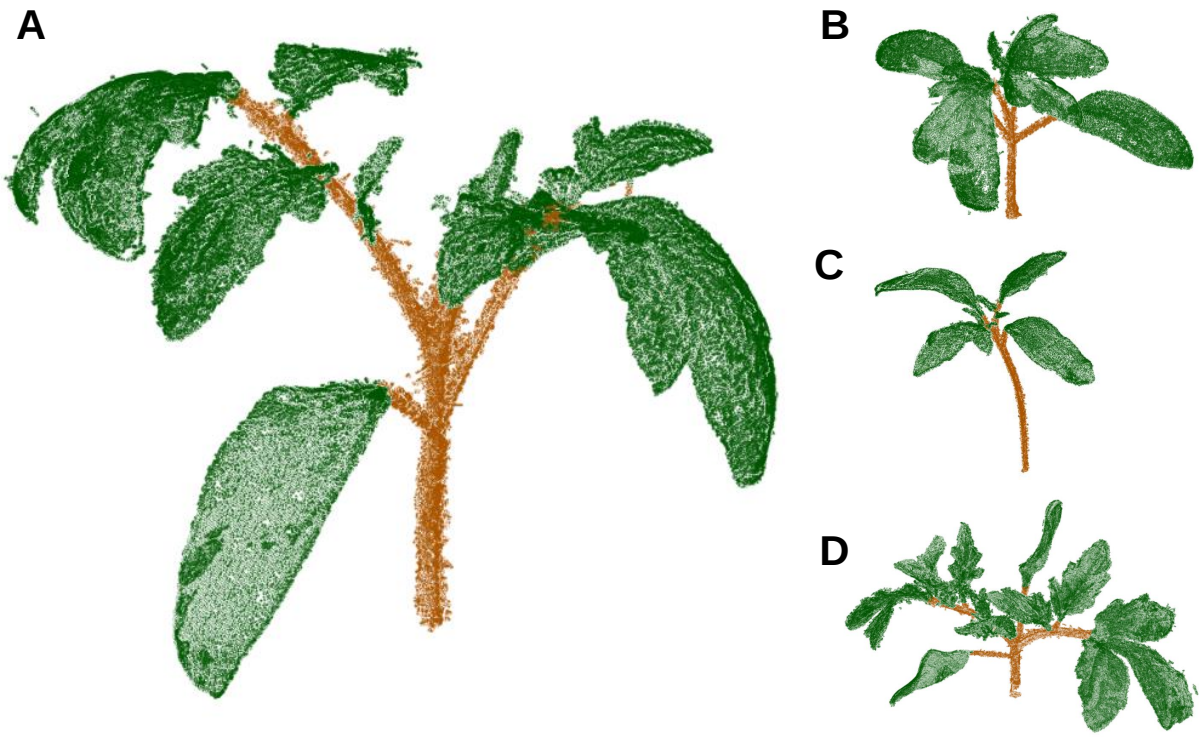

Figure 2.2: **Point classification results.** A–D) Classification of lamina (green) versus branch (brown) points on four plants.

from a branch. In these regions, it is difficult, even for humans, to agree on the precise boundary between a branch and a leaf. The other methods (RF, KNN) made more mistakes in less ambiguous locations (e.g., the middle of a leaf or branch). Thus, for the remaining experiments, we used the DL classifier. Figure 2.2 shows results of the point classification on four plants using a combination of FPFH feature vectors with the DL classifier.

2.3.3 Generalization across different environments

Plants architectures are highly plastic and often modify their structure — including leaf shape, stem thickness, and branch angles — in response to the environment. In practice, training data may not be available for all possible conditions, and thus in practical applications, it is important that classifiers can accurately classify leaves and branches when applied to architectures

grown in never-seen-before conditions.

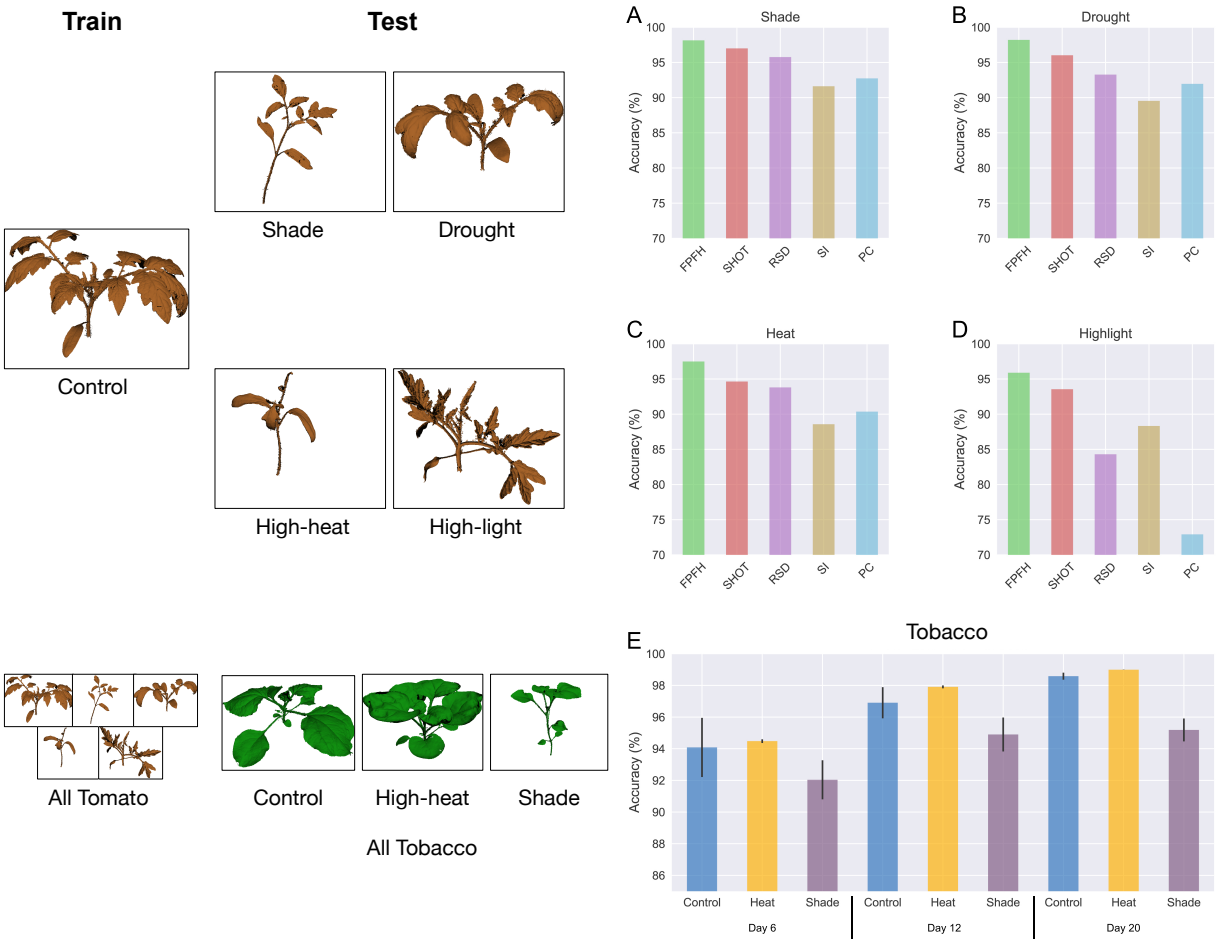

Figure 2.3: **Generalization of classification accuracy across different environments and species.** A–D) Results of training a classifier on tomato control plants, and testing on tomato plants for four other conditions — shade, drought, high-heat, and high-light. Over all descriptors, FPHF generalizes the best across conditions. E) Results of training a classifier on tomato data from all conditions and testing it on tobacco data, using FPFH descriptors. Performance on the tobacco data is shown separately per condition and time-point.

To test this, we trained a DL model using a set of 12 control tomato plants (ranging from developmental days 5–20), and tested the model on plants grown in four other conditions: shade, drought, high-heat, and high-light [CPCN17]. These conditions produced many variations in architectural features (Figure 2.3, left). For example, in high-light, we observed significant thickening of the leaves and stem compared to all other conditions, and many more irregularities

in leaf shape. In drought, we observed a weaker plant with fewer and thinner leaves and branches.

Compared to training a DL classifier with other features, we found that FPFH features again performed best, achieving over 95% accuracy on all new conditions (Figure 2.3A–D). For example, in high-light, FPFH achieved an overall accuracy of 95.88%, compared to 93.55% for SHOT and 72.91% for PC. In high-heat, FPFH achieved 97.48% accuracy, compared to 94.64% for SHOT and 90.35% for PC. In drought, FPFH achieved 98.2% accuracy compared to 96.01% for SHOT and 91.96% for PC. Principal curvatures has the largest variance across conditions, perhaps because this feature only consists of two values. On the other hand, SHOT has over 10-times as many feature values as FPFH (352 versus 33), but it still performed worse than FPFH.

While prior results have also proposed using FPFH features for leaf/branch classification [WPKM15, PDMK13], this to our knowledge is the first quantitative demonstration of its accuracy, coupled with deep learning, across new conditions.

2.3.4 Generalization across different species

A related challenge is to build classifiers that can accurately classify leaves and branches on species that have not been trained on. Indeed, there are thousands of plant species, some with very different branching and leaf patterns, and it is unreasonable to assume that training data will be available for all of them. Here, we tested how well a DL classifier trained with FPFH features from tomato plants generalized to another dicot plant: tobacco. The tobacco plants were also collected under three growth environments (control, shade, and high heat) and across three developmental time-points (days 6, 12, and 20) [CPCN17].

Overall, we find $> 95.89\%$ accuracy, averaged over all tobacco plants (Figure 2.3E). Older plants are classified more accurately regardless of the environment (e.g., 98.6% control day 20 versus 94.1% control day 6), whereas plants grown in shade had a slightly lower accuracy. Nonetheless, these results suggest that some cross-species generalization is possible within our

framework, and sets a benchmark for future studies to improve upon.

2.4 Acknowledgments

Materials used in this chapter appear in Plant Physiology 181(47). Ziamtsov, Illia; Navlakha, Saket., 1425-1440 2019. The dissertation author was the primary investigator and author of this paper.

Chapter 3

Leaf counting

The result of the classification yields two sets of points: leaf points and branch points. The leaf points by themselves are "free flowing" islands of points in space. Leaf points can consist of a single leaf or many different leaves. The next challenge is to segment or cluster these leaf points into subsets, where each subset represents an individual leaf. Segmented leaves allow one to calculate traits, such as flowering time and drought tolerance [MFST16]; different stresses can also affect leaf size [JMD74], which can determine photosynthetic response [VQM16].

In this section, we present a region growing algorithm for separation of individual leaves from all the points that were classified as leaf points. In particular, we use four metrics to decide whether a point belongs to a current cluster (leaf). One primary challenge for leaf segmentation is leaf touching or overlapping of other leaves, which becomes worse when the volume of foliage grows. Our algorithm is developed to mitigate errors of these particular cases.

3.1 Leaf segmentation algorithms

Leaf segmentation is a difficult problem because leaves can have many different shapes, poses, and orientations, including variation in wrinkles and ripples [XWCL15, MTM16]. Further, the data itself can be noisy due to occlusion of certain parts of the leaf during scanning, issues

in registration multiple scans, identification of small newly emerging leaves [LFM+13], and close proximity and perhaps touching of two different leaves [MTM16]. While some methods have been proposed that solve these problems for specific types of leaf structure (e.g., sunflower leaves [GDHB17] or heuristics to deal with some types of touching leaves [MTM16]), here we seek a more general solution that is agnostic to any such prior.

Below, we describe two prior methods for segmentation, followed by our new proposed method. All methods use an iterative, region-growing approach to cluster leaf points into subsets, but each method uses a different criteria to determine whether to admit a candidate point into a cluster.

3.1.1 Euclidean clustering

The first method clusters leaf points into subsets based on Euclidean distances [Rus09]. A point is picked randomly to start a cluster, and then that cluster is grown by admitting points that are less than a threshold distance away from at least one point from the cluster. This procedure is repeated until all leaf points have a cluster assignment. Ideally, the distance threshold should be larger than the average distance between two points on the same leaf, yet smaller than the distance between points on two distinct leaves. It is difficult to set this parameter exactly due to variation in leaf structure; however, in our case, we performed a grid search to optimize performance and set it to 0.3. In addition, we filtered out clusters with less than 300 cloud points, as these were typically noise.

3.1.2 Segmentation using smoothness curvature

The second method clusters leaf points into subsets using normal vectors and estimated curvature values (called smoothness) [RvV06]. The objective is to cluster sets of points that have a smooth surface. The method starts by computing and sorting the estimated curvature for all points. A cluster is initialized with the point p_i with the smallest curvature (i.e., the surrounding region around p_i has a flat surface). This cluster is grown by considering p_i's neighbors; for each neighbor p_j, if the angle between normal vectors of the p_i and p_j is less than a threshold, p_j is admitted to the cluster. To generate new seed points to grow the cluster, the curvature of p_j is checked to see if it is less than another threshold; if so, p_j is also added to a queue of seed points. If not, p_j remains in the cluster with p_i, but its neighbors are no longer candidates to further expand the cluster. Once all points from the seed queue are exhausted, a new cluster is started with the next smallest curvature value (that has not already been assigned to a cluster) and the process repeats. The parameters for the normal calculation and curvature threshold values were selected using a grid search on the entire dataset.

3.2 Improved leaf segmentation

The third method, which we developed, clusters leaf points into subsets using distances, normal vectors, curvature, and FPFH features (Algorithm 1). The curvature around point p_i is typically calculated as: $(\lambda_3)/(\lambda_1+\lambda_2+\lambda_3)$, where $\lambda_3 \leq \lambda_2 \leq \lambda_1$, and where the eigenvalues are calculated using PCA on the neighborhood of p_i. The drawback of this approach is that this measure sometimes produces sharp changes at the edges and in areas that are flat but have lower density. This can lead to over-segmentation.

Instead, we propose an anisotropic measure of smoothness: $(\lambda_1-\lambda_3)/\lambda_1$. This measure provides better clustering of points located on edges and in poorly scanned areas (lower point

density or holes). Because this method does not consider the second eigenvalue, it produces a smoother curvature in flat areas, and the abrupt changes are more likely to happen where two leaves are in contact.

Our method also uses an additional check to determine whether a point should be admitted into a cluster based on the similarity of FPFH features. This approach is inspired by a similar method used for creating supervoxels for point clouds [PASW13]. Intuitively, using FPFH features will help capture the essence of the surface, despite possible hairs, wrinkles, or tears on the leaf. The complexity of FPFH allows this technique to scale to plants with even a million points.

Algorithm 1: Our leaf segmentation method based on region growing

```
Data: Set of leaf points ℒ
Result: Set of leaf clusters C_ℒ
Function conditioned_RG(Δ_d, Δ_n, Δ_c, Δ_f):
    C_ℒ ← {}   // Empty list of leaf clusters
    Q ← {}   // Queue of points to be checked
    T ← createKD_Tree(ℒ)   // Used to find candidate nearest-neighbors.

    for p_i ∈ ℒ : do
        append(Q, p_i)
        for p_i ∈ Q : do
            // Get point closest to p_i not yet assigned to a cluster.
            p_y ← getClosestNonProcessed(p_i, T, C_ℒ)

            // Difference in Euclidean distance
            if distance(p_i, p_y) > Δ_d then
                continue;
            // Difference in normal orientation (angle)
            if degrees(n⃗_i, n⃗_y) < Δ_n then
                append(Q, p_y);
            // Difference in curvature (smoothness anisotropy)
            if distance(Anisotropy(p_i), Anisotropy(p_y)) < Δ_c then
                append(Q, p_y);
            // Difference in FPFH distance
            if distance(FPFH(p_i), FPFH(p_y)) < Δ_f then
                append(Q, p_y);
        append(C_ℒ, Q)
        Q ← {}
    return C_ℒ
```

In summary, our clustering method starts from a random seed point and the cluster is grown if any of three conditions are satisfied (Algorithm 1). Once the cluster is grown to exhaustion, a next random seed point is drawn from the points that have not been clustered yet, and this

process is repeated until all points belong to some cluster.

3.3 Testing and validation framework

To test the accuracy of each leaf segmentation method, we only count a leaf as correctly segmented if all its points are correctly placed into the same subset. If a leaf's points are split into two subsets, we call this over-segmentation; if two leaves are merged, we call this under-segmentation. Each ground-truth leaf is counted independently and is not weighted by its size.

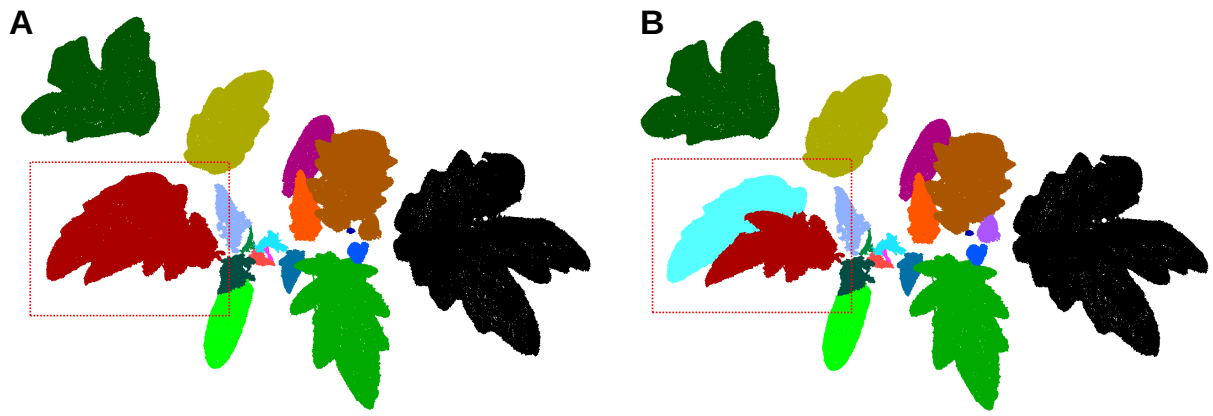

Figure 3.1: **Leaf segmentation results.** A) Results of segmentation with smoothness curvature constraint. B) Results of our improved leaf segmentation. The leaves in the red box are overlapping, however our segmentation was able to detect them as separate leaves.

3.4 Leaf counting results

The leaf points identified after classification will be "free flowing" islands of points in 3D space. The next task is to cluster these points, where each cluster corresponds to an individual leaf.

When analyzed on 291 total ground-truth leaves from the tomato plants, our enhanced

Table 3.1: **Leaf segmentation and counting results.** Comparison of three methods: Euclidean Clustering, Segmentation with Smoothed Curvature, and Our method. 'Under' and 'Over' indicates the number of under- and over- segmented leaves, respectively. Summary on the bottom row shows overall performance over all plants.

		Euclidean Clustering			Segmentation using SC			Our method		
Plant	**Day**	**Under**	**Correct**	**Over**	**Under**	**Correct**	**Over**	**Under**	**Correct**	**Over**
Control-A	5	0	4/4	0	0	4/4	0	0	4/4	0
Control-A	12	1	10/12	0	0	11/12	1	0	11/12	2
Control-A	20	6	11/21	0	0	20/21	1	1	18/21	2
Control-B	5	0	4/4	0	0	4/4	0	0	4/4	0
Control-B	12	2	5/10	1	1	6/10	2	0	9/10	1
Control-B	20	5	12/18	1	1	14/18	4	1	12/18	13
Control-C	5	0	4/4	0	1	3/4	0	0	4/4	0
Control-C	12	1	8/11	0	1	9/11	0	0	8/11	3
Control-C	20	6	10/18	0	1	15/18	3	1	16/18	5
Control-D	5	0	3/3	0	0	3/3	0	0	3/3	0
Control-D	12	2	7/10	0	0	10/10	0	0	10/10	0
Control-D	20	6	12/18	0	1	16/18	1	0	16/18	7
Heat-A	5	1	1/3	0	1	2/3	0	0	3/3	0
Heat-A	12	1	3/6	0	1	4/6	0	1	4/6	0
Heat-A	20	0	6/6	0	0	6/6	0	0	6/6	0
Heat-B	5	0	2/2	0	0	2/2	0	0	2/2	0
Heat-B	12	0	3/3	0	0	3/3	0	0	3/3	0
Heat-B	20	1	2/4	0	1	2/4	1	0	4/4	0
Shade-A	5	0	4/4	0	0	4/4	0	0	4/4	0
Shade-A	12	0	6/8	0	1	7/8	0	1	7/8	0
Shade-A	20	0	9/12	0	2	10/12	0	2	10/12	0
Shade-B	5	0	4/4	0	0	4/4	0	0	4/4	0
Shade-B	12	1	6/8	0	1	7/8	0	0	7/8	1
Shade-B	20	2	12/16	0	2	13/16	0	3	11/16	0
Drought-A	5	0	4/4	0	1	3/4	0	0	4/4	0
Drought-A	12	1	7/9	0	2	6/9	0	1	8/9	0
Drought-A	20	2	12/15	0	0	15/15	0	1	14/15	0
Drought-B	5	2	1/4	0	1	3/4	0	0	4/4	0
Drought-B	12	1	5/7	0	1	4/7	1	0	7/7	0
Drought-B	20	2	8/12	0	1	9/12	1	1	9/12	1
Hightlight-A	5	1	2/4	0	1	2/4	0	0	4/4	0
Hightlight-A	12	7	3/11	0	1	8/11	2	0	10/11	2
Hightlight-B	5	0	4/4	0	0	4/4	0	0	4/4	0
Hightlight-B	12	4	8/13	0	0	9/13	6	0	8/13	8
Summary		55	202/291 **69.42%**	2	23	242/291 **83.16%**	23	13	252/291 **86.60%**	45

leaf counting method achieved an overall accuracy of 86.60%, with 13 under-segmented and 45 over-segmented leaves (Table 3.1). In contrast, the Euclidean clustering method achieved 69.42% accuracy, with 55 under-segmentations and 2 over-segmentations. The performance of Euclidean clustering declined significantly as plants grew and more leaves started to become in contact with each other. Finally, segmentation with the smoothness constraint performed better than Euclidean clustering, achieving 83.16% accuracy, with 23 under-segmentations and 23 over-segmentations, though still slightly worse than our method. Thus, adding FPFH features to the growing criteria helped the algorithm deal better with natural variations in leaf structure, including hair, ripples, wrinkles, and veination. Although not reported here, we also tested other

eigenvalue-based curvature metrics, such as planarity and surface variation [HWS16], which did not perform competitively.

Across all methods, the bulk of over-segmentation errors occurred for leaves with scanning holes or with poor registration from multiple scans. All of these methods include parameters that can be tuned to optimize under- and over- segmentation, and future work may try to introduce a learning procedure to set these parameters automatically given data. Figure 3.1 shows an example of a plant, where our segmentation separates an overlapping pair of leaves into two separate leaves.

3.5 Acknowledgments

Materials used in this chapter appear in Plant Physiology 181(47). Ziamtsov, Illia; Navlakha, Saket., 1425-1440 2019. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Skeletonization: small branches & root

The next challenge is to transform the branch points into an underlying skeleton (a graph-theoretic tree) that captures the essential shape of the plant. The skeleton of a 3D object is a thinned 1D representation (often in the form of a graph) that captures the basic geometry and shape of the object. Skeletons have broad applications in computer vision, such as for object matching, surface reconstruction, feature tracking, and computer animation [CSM07]. Accordingly, there are a variety of approaches for extracting skeleton graphs from structures represented as 3D meshes or point clouds [TZC09, CTO+10, ATC+08, DJR14, VL00, HWC+13, SLSK07].

More recently, skeletonization has become critical to modern plant phenotyping [CPCN17, BABA+17, PL12]. The advent of high-throughput 3D imaging platforms, such as light detection and ranging (LiDAR), has generated large datasets of point clouds of plant shoot architectures [PNE17, PS19]. While advancements in data acquisition have made it possible to rapidly scan plants, methods for analyzing them in real-time remain a bottleneck. In addition to plant science, skeletonization is also important in forestry [HSC+15], ecology [MMK+04], urban planning [ZLD+14], and engineering [QZN14]. All of these applications require methods for accurately skeletonizing shapes from 3D data.

In this thesis, we focus on skeletonization of organic shapes. Compared to man-made

structures, skeletonizing natural, organic structures has many challenges, including dealing with the curvature of branches (i.e., no straight lines), the tapering of branches (i.e., unequal radii over the length of the branch), the non-uniformity of branch lengths, and other nuisances, such as trichomes (the fine hairs on plant stems), which disrupt fitting with simple primitives. These issues lie alongside the usual difficulties associated with processing point clouds, such as missing data, non-uniform density of points, and registration errors [HJW+17, PMG04].

4.1 Families of skeletonization methods

Several methods for extracting skeleton graphs from 3D data are designed to function on closed polygonal meshes. However, LiDAR scanners do not reliably produce watertight shapes due to challenges posed by self-occlusion and light reflection by the target surface [TZC09]. Surface reconstruction is often unreliable, since errors introduced can lead to inaccurate skeletons [CTO+10]. Therefore, we focused on approaches that can be applied directly to unorganized 3D point clouds.

Algorithms for skeleton extraction from unorganized 3D point clouds broadly fall into four categories: topological thinning, distance field-based, general field-based, or geometric [CSM07]. Some approaches, such as topological thinning and distance fields, typically work on voxelized data. However, these methods are sensitive to the discretization resolution used, and they can have poor results when the point cloud is noisy, non-uniform, or incomplete. In practice, some methods combine elements from multiple categories, so the distinctions between categories are not strict.

Topological thinning or contraction algorithms start at the surface of an object and shrink it to a 1D skeletal representation [CSM07]. This is done by iteratively removing voxels from the shape's boundary while preserving its basic topology. Because each voxel can be evaluated locally, thinning algorithms are generally computationally efficient. However, they are susceptible

to the excessive removal of branch endpoints, which can lead to truncated skeletons [CSM07]. Au et al. [ATC+08] developed a mesh contraction algorithm for non-volumetrized data that uses the Laplacian operator to smoothly collapse an object into a skeletal shape. Subsequently, Cao et al. [CTO+10] adapted this Laplacian thinning approach to work directly on raw point clouds. However, this method requires careful tuning of contraction parameters to remain faithful to the object's topology and is sensitive to sampling density. Because the Laplacian operator requires the calculation of higher-order derivatives, it is also computationally expensive and not guaranteed to be numerically stable.

Distance field-based methods apply the distance transform, which identifies the shortest distance from each computed interior point to the boundary of the object [HF05]. Any locally-centered voxels are putative members of the shape's skeleton, which are subsequently pruned by thinning or clustering. The remaining voxels are re-connected to produce a final skeleton graph. Although these methods are efficient, they are sensitive to boundary noise and suffer from poor centeredness due to errors at the re-connecting step [CSM07]. Huang et al. [HWC+13] apply a spatially-localized version of the L_1-median to local neighborhoods of points, and then apply contraction and re-centering to generate a final skeleton. However, this method is sensitive to sampling density and uniformity.

General field-based approaches apply functions other than the distance transform to point sets, such as potential fields or Newtonian repulsion [CSM07]. For example, Sharf et al. [SLSK07] perform surface reconstruction by growing a deformable blob that captures the point cloud's shape, and then extract a skeleton by finding its centerline. In general, these methods tend to be robust to noise at object boundaries. However, if the data is incomplete (e.g., if there are holes or sparsely sampled areas), the blob can leak outside of the underlying shape. In addition, some of these methods require calculating higher-order derivatives, and thus may be inefficient for large point clouds.

Geometric methods encompass approaches that work directly on raw point cloud data.

For example, Voronoi tesselation can produce an approximation of a shape's medial surface [OI92, DS06], which can then be pruned into a skeleton. However, this process is highly sensitive to noise [CSM07]. Alternatively, Reeb graphs are more robust to outliers, but can be sensitive to the object's orientation [CSM07]. PypeTree [DJR14], which builds off the method by Verroust and Lazarus [VL00], is designed specifically for point clouds of plant architectures. It calculates geodesic distances from the plant's root to the rest of the point cloud. Skeleton points are then extracted from the centroids of the points in each level set. However, if a level set spans a fork, points that belong to multiple branches will not be differentiated. Therefore, PypeTree can produce geometric errors and poor angle measurements at forks. ROSA [TZC09] is another method that uses normal information and rotational symmetry to extract skeletons from point clouds that have an underlying cylindrical shape. ROSA performs well on incomplete point clouds; however, it is generally less robust to noise and outliers, requiring preprocessing and denoising before it can be applied to raw data [TZC09].

4.2 PypeTree algorithm

To extract a skeleton graph, we extended upon the PypeTree method [DJR14], which improves upon the Verroust and Lazarus method [VL00] and other prior work [Buc14]. The Verroust and Lazarus method assumed that cloud points are sampled uniformly or near uniformly; PypeTree removed this assumption to better deal with point clouds with inconsistent density and outliers. Prior work has also studied skeletonization of botanical trees [DJR14, BLM09, BLM10]; one challenge with our dataset, however, are the numerous small branches, where a more scale-invariant method would be desired.

Our proposed method builds upon PypeTree and is designed to better capture skeleton curvature by adding two new features: 1) More accurate tip detection; and 2) Enhanced root selection.

First, we briefly outline the five main steps of the PypeTree algorithm.

1. A point cloud is turned into a neighborhood graph $G = (V, E)$ by connecting each point with its neighbors. Specifically, each cloud point p_i corresponds to a node $v_i \in V$, and there exists an edge between v_i and v_j if v_j is within radius r from v_i.

2. The neighborhood graph G is converted into a geodesic graph, $F = (V, E' \subset E)$ with respect to some root point, $v_{\text{root}} \in V$. The edges E' of the geodesic graph are found by computing the shortest path from v_{root} to every vertex $v_i \in V$ using Dijkstra's algorithm, and then taking the union of all the edges in these paths.

3. The nodes of the geodesic graph are divided into "levels" based on the geodesic distance from the root. In other words, all nodes within distance 1 from the root are placed into the first level; all nodes with distance between 1 and 2 are placed into the second level; etc. Figure 4.1 illustrates nodes assigned to different levels with different colors. Each level will ultimately represent one or more nodes in the final skeleton graph (see Step 5 below).

4. A subgraph H_i is created for each level i. The nodes of H_i are all the points in level i. Each node in H_i is connected to its two nearest-neighbors in H_i.

5. A subgraph H_i may contain multiple connected components; for example, after a branch point (Y-junction) in the preceding level, there may be two connected components in the same level, representing the left and right sides of the branch. Thus, we create one node in the final skeleton graph per connected component, and this node is created at the centroid of the connected component. The final step is to create edges between centroid nodes in level i and level $i+1$. A centroid in level $i+1$ is connected to a centroid in level i if a (random) point in the former's connected component goes through some point in the latter's connected component along the way to the root in the geodesic graph. Intuitively, this will avoid branches from "crossing over" across levels.

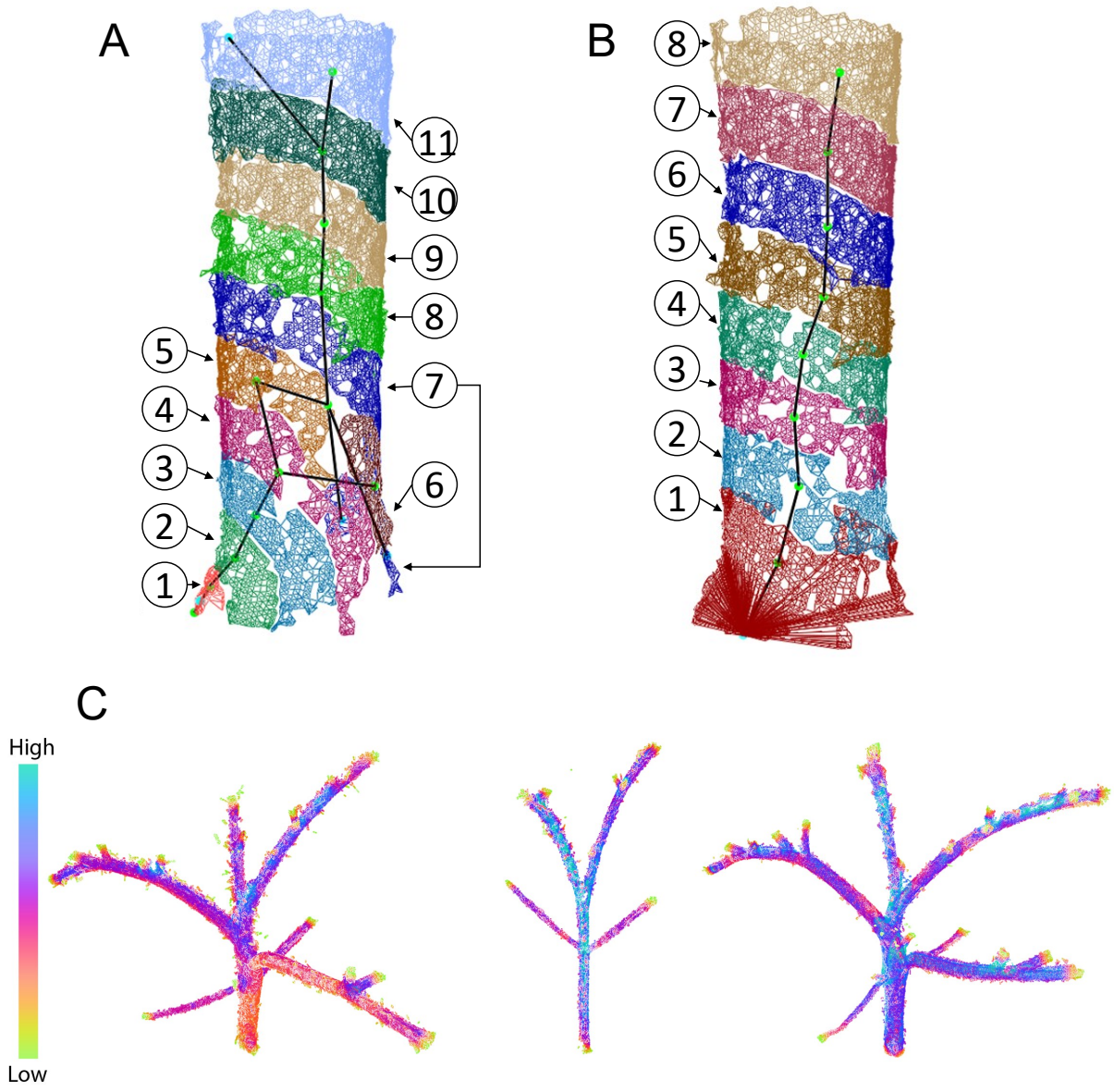

Figure 4.1: **Improvements in branch skeletonization.** A) Visualization of the levels (colored bands) outputted by the PypeTree method. The default root point shown in Level 1 can be off-center when selected as the point with the lowest y-value. This can cause tilting of the architecture and other deformities in the resulting skeletonization (black line), especially when holes are present in the point cloud data. B) Visualization of the levels and root point output by our method. The balanced levels and centered root point improves the quality of the skeleton graph. C) Visualization of 1st (most dominant) eigenvalues for each point. Points with low eigenvalues, as indicated by the scale bar, are predominantly located at tips (the root and regions where leaves emerge).

PypeTree includes additional filtering steps to reduce noise and other implementation optimizations to reduce running time, which we do not describe here and instead refer the reader to the original publication [DJR14].

We next describe two additional features we added to improve skeletonization.

4.3 Improved detection of tips.

A critical component of skeletonization is finding tips, which are degree-one nodes of the skeleton where leaves emerge. Accurately identifying tips is challenging due to impartial capture, holes, and small features, such as hairs on the stem. The tips that PypeTree identifies are highly sensitive to the size of the levels: if the level size is set very large, then tips connected to small branches and petioles will be missed. On the other hand, if the level size is set too small, then false tips may emerge and the resulting skeleton may be too sensitive to variation.

Our approach to finding tips is independent of connected components or level size, thus making it independent of the resolution of the skeleton graph. Tips are located at the boundary (edges) of a point cloud, which have a natural discontinuity compared to other points. Thus, we can detect edge points by using principal components analysis. Specifically, for every point in the point cloud, we compute its first eigenvalue based on its neighboring points within some radius r. The first eigenvalue of points close to the boundary will be smaller than that of points far from a boundary because the neighbors of boundary points will not be spread around the point but rather will be biased to some side (Figure 4.1C). We then perform Euclidean clustering on all points whose first eigenvalue is among the smallest 15% of all points, such that points that belong to the same tip should reside in the same cluster. Clusters that contain less than 50 points are filtered out as noise. The centroids of each cluster become the tip nodes of the skeleton graph. Finally, a tip is attached to the skeleton graph by picking a random point from its cluster, finding which connected component it falls in (Step 5 above), and adding an edge from the centroid of

this cluster to the tip. Finding tips in this manner allows for the detection of even small branches and petioles emerging during development.

4.4 Improved detection of the root.

The stem base (hereafter simply called the "root") is a special case of a tip and is critical for grounding the skeleton in the correct location. The goal is to find a root point that lies at the base and center of the stem. This location of the plant, however, is often difficult to scan because of its proximity to the soil and because of holes caused by scanning errors. The default root point of PypeTree is the point with the lowest y-value. This point can be highly skewed to one side of the stem (Figure 4.1A) which can cause artificial curvature in the skeleton.

Our solution starts by noticing that the level closest to the default root — i.e., all the points within distance 1 to the root — often contains points that do not wrap evenly around the base of the stem (Level 1, Figure 4.1A). This is because the distance needed to wrap the base of the stem is greater than distance that demarcates a single layer. In addition, these distances are highly sensitive to noise, holes, or other small variations around the root. However, as the level number increases, the points appear more evenly balanced around the stem, forming even rings (Level 11, Figure 4.1A).

To form even rings around the root, then, our idea is to build the geodesic graph starting from the tip that is furthest away from the root and trace *backwards* (Level 1, Figure 4.1B). In this way, the level closest to the root now will be less sensitive to the choice of a root point. Specifically, let r_1 be PypeTree's default root, and let r_2 be the point that is furthest away from r_1. Then we find all points p_i such that:

$$d(r_1, r_2) = d(r_2, p_i) \pm \beta,$$

where β is a parameter that adjusts amount of points to be sampled in the vicinity of the

root. Then, for each p_i we compute:

$$z(p_i) = d(r_1, r_2) - (d(r_1, p_i) + d(r_2, p_i)).$$

Intuitively, all points with $z_p \approx 0$ will represent an evenly balanced ring around the true root (Level 1, Figure 4.1B). We reset the root v_{root} to be the centroid of these points, and then add edges from v_{root} to all such points (with equal weight) to the neighborhood graph G. We then proceed with Step 2 of the PypeTree algorithm above.

4.5 Testing and validation framework

We tested the accuracy of our method to detect tips on the 12 oldest tomato plants, which had the most branching. We manually identified and counted the tips for each plant and compared these counts with those predicted by each method based on precision and recall.

4.6 Improving small branches & root results

Using only the branch points as input, we next compared the quality of our algorithm versus PypeTree in forming a skeleton graph of the branch points.

First, we find that our algorithm generated skeletons that are visually better aligned with the underlying cloud points (Figure 4.2). PypeTree consistently struggled identifying small branches (Figure 4.2A), corresponding to where new leaves are emerging. PypeTree also often off-centered the root point, which caused the underlying skeleton to incorrectly lean. While PypeTree does include parameters to adjust the level size that can improve these errors in some cases, it was difficult to set this parameter in such a way that it generalized across species, plant size, and branching patterns. In contrast, our method, which is independent of level size

parameters, defined a root point that was much more centered along the main axis of the stem and better dealt with more ambiguous regions of the point cloud (Figure 4.2B).

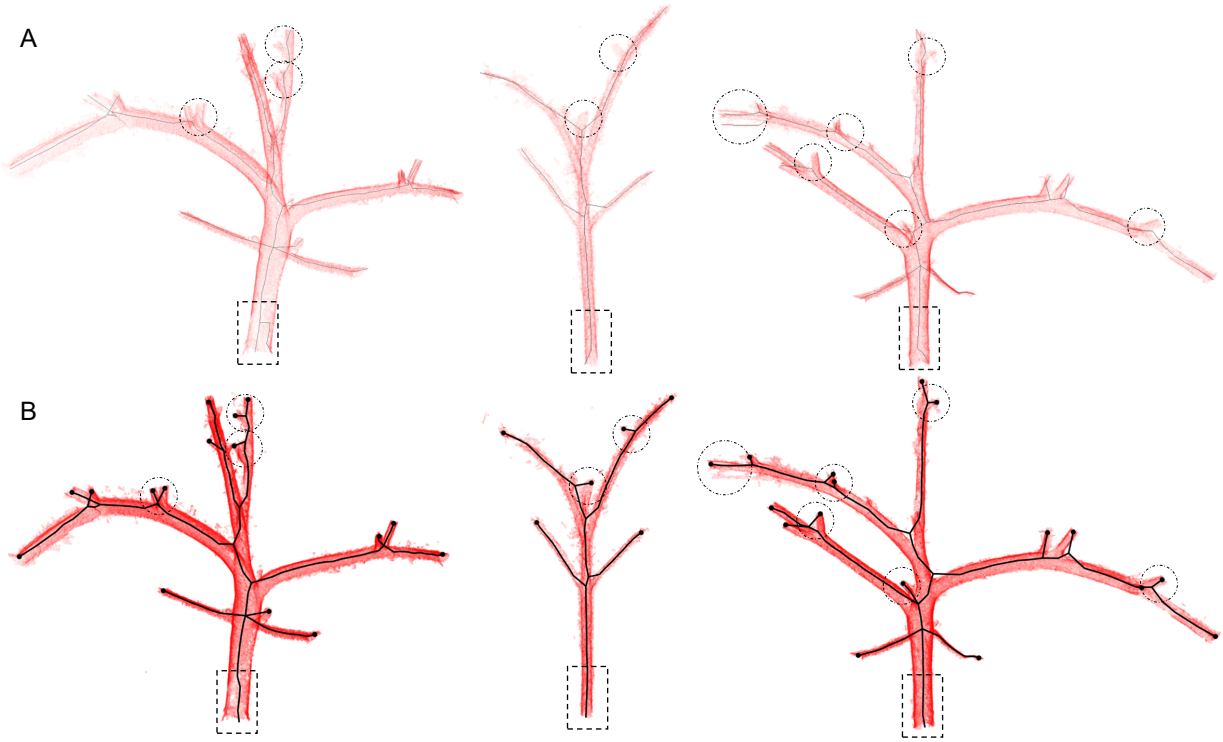

Figure 4.2: **Branch skeletonization comparison for three plants.** The branch skeletonization produced by (A) PypeTree and (B) our method, on three tomato plants. The red points are the branch cloud points, and the black line is the resulting skeleton graph. Dotted circles highlight areas of improved tip detection for our method. Dotted rectangles highlight instances where the root tip was off-center according to PypeTree and better centered by our method.

Second, we quantified how accurately each method identified tips on a set of 12 tomato plants from day 20 across the five environmental conditions (Table 4.1). Day 20 plants were selected because these plants have the largest size and most branching. Our method achieved higher or equivalent precision on 11 of the 12 plants, and higher or equivalent recall on all 12 plants, compared to PypeTree. Averaged over all 12 plants, our method achieved a precision of 95.0% $\pm$ 7.0% compared to 89.0% $\pm$ 12.0% for PypeTree. The reduction in false positive and false negative tips is important, as such errors can alter the underlying shape of the skeleton,

which can affect downstream morphological analyses.

Third, we evaluated the run-time of each method and found 1–2 orders of magnitude improvement in our algorithm versus the provided PypeTree implementation (Table 4.1). For example, on the Highlight-B plant, PypeTree took 919 seconds, whereas our method only took 17.51 seconds. This improvement is significant, especially when dealing with large collections of plants scanned in real-time applications.

Table 4.1: **Improvement in skeleton tips identification.** Precision vs. recall of PypeTree versus our tip identification method on 12 tomato plants. Overall, our method achieves higher precision and recall and is over an order of magnitude faster.

		PypeTree			Our method		
Plant	**Day**	**Precision**	**Recall**	**Time (s)**	**Precision**	**Recall**	**Time (s)**
Control-A	20	0.87	0.93	677	0.88	1.00	23.56
Control-B	20	0.83	0.83	764	0.80	1.00	25.92
Control-C	20	0.67	0.80	427	0.83	1.00	16.89
Control-D	20	0.64	0.82	596	1.00	1.00	14.94
Drought-A	20	0.86	0.75	163	0.89	0.89	7.66
Drought-B	20	1.00	0.57	570	1.00	0.86	8.03
Heat-A	20	1.00	1.00	25	1.00	1.00	4.14
Heat-B	20	1.00	1.00	12	1.00	1.00	2.27
Highlight-A	20	0.86	0.75	1132	1.00	1.00	2.29
Highlight-B	20	0.93	0.72	919	1.00	0.94	17.51
Shade-A	20	1.00	1.00	137	1.00	1.00	10.40
Shade-B	20	1.00	0.75	217	1.00	1.00	14.85
Average		0.89	0.83	469.92	**0.95**	**0.97**	**12.37**

4.7 Acknowledgments

Materials used in this chapter appear in Plant Physiology 181(47). Ziamtsov, Illia; Navlakha, Saket., 1425-1440 2019. The dissertation author was the primary investigator and author of this paper.

Chapter 5

Skeletonization: angle extraction

In this chapter, we continue on the topic of skeletonization of plant architectures from 3D point clouds which is critical for many downstream tasks including analyses of plant shape, morphology, and branching angles. Specifically, we aimed to improve skeletonization at branch points (forks) by leveraging geometric properties of cylinders around branch points. In other words, our goal was to improve the quality of angles in the skeleton graph.

Fast and accurate determination of branch angles is important for many downstream tasks, including the study of plant growth and development [LFC$^+$18b], measuring light interception by leaves [GSA$^+$12a], and for inferring genotype-to-phenotype relationships [XB20], both in the lab and in the field [SLP$^+$18, ARAM$^+$13]. In addition, assessing and quantifying changes in plant morphology is often used for measuring agricultural yields [MBR16], analyzing stress responses and plant-environment interactions [GMGB17], and facilitating functional genomics studies [YDC$^+$13].

In this chapter, we describe a fast skeletonization method that can, in seconds, generate accurate branch angles of noisy and heterogeneous 3D point clouds. The method builds upon our prior work from the previous chapter [ZN19] by analyzing the geometry of normal vectors around branch points, projecting these vectors onto a Gaussian sphere, and then robustly clustering them

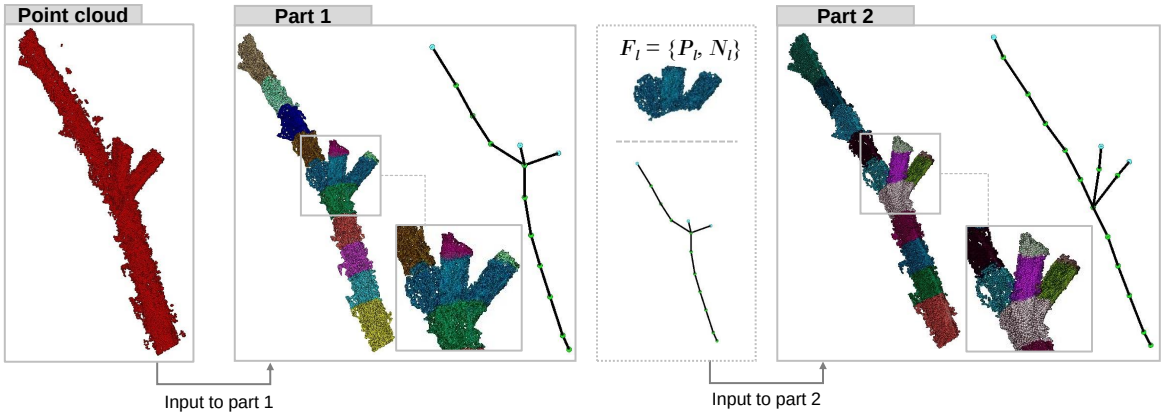

Figure 5.1: **Overview.** As input, we are provided a 3D point cloud of a plant architecture. A partial point cloud of the region around a branch point (fork) is shown. In Part 1, we compute its skeleton. Each color represents points and normal vectors that belong to a node in the graph. In Part 2, the geometry and branch angles around a fork are refined to better match the underlying plant shape.

into unique individual branches.

5.1 Method overview

As input, we are given a point cloud $\mathcal{P}$ of n points, where each three-dimensional point $p_i \in \mathcal{P}$ describes a location on the surface of the plant. Each p_i has an associated normal vector $\overrightarrow{n_i}$ representing its 3D orientation; these vectors form a set of normals N. In our dataset, normal information was provided by the 3D scanner.

Our method can be divided into two parts (Figure 5.1): computation of the initial skeleton, and refinement of the skeleton at branch points (forks). The first part is done using the approach described in previous chapter [ZN19], which builds off of PypeTree [DJR14]. We refer to the improved version as PypeTree* to distinguish it from the original method [DJR14].

Briefly, PypeTree* divides the points $\mathcal{P}$ into level sets, where each level l consists of points $\mathcal{P}_l \subset \mathcal{P}$ that lie within a similar geodesic distance from the root of the tree. For each level, a node of the skeleton graph is created at the centroid C_l of that level's points. For example, Figure 5.1

(Part 1) shows nodes (i.e., levels) in different colors, with their associated points. Forks are detected by running a connected components algorithm on $\mathcal{P}_l$; if $\mathcal{P}_l$ consists of two disconnected components — points on the left and right of the fork, each with a similar geodesic distance to the root — then the level is split into two, and two nodes are created, one at the centroid of each component. The output is a skeleton graph $G = \{V, E\}$, where the nodes $V = \{C_l\}$ over all l; each level has an associated set of points $\mathcal{P}_l$ and normal vectors N_l; and the edges E connect nodes across successive levels.

This approach is fairly fast on large point clouds; however, it often suffers from geometric errors at forks. Even if there is a fork at level l, in practice, $\mathcal{P}_l$ often consists of only one component, and it isn't until level $l+1$ where the two branches of the fork are sufficiently separated from each other that their corresponding points no longer belong to the same component. In terms of the skeleton, this means that $\mathcal{P}_l$ generates a single node in the graph, and P_{l+1} generates two nodes, each connected to the centroid of $\mathcal{P}_l$. This severely compromises the angle estimation at the fork since the branch point is detected late. Furthermore, the location of the centroid C_l can be biased (due to averaging of points that belong to different branches), leading to inaccurately-placed nodes within forks, and again, incorrect angles.

Thus, the second part of our method refines the skeleton geometry around each fork, using as input the points $\mathcal{P}_l$ and normal vectors N_l associated with each fork (Figure 1 (Part 2)). If normals are unavailable, methods exist for normal estimation [MN03, ZCL$^+$13], which can be applied before using our method.

5.2 Gaussian sphere mapping

The points $\mathcal{P}_l$ (Figure 5.2A) and normal vectors N_l for each fork node l are processed as follows. For simplicity, we drop the subscript l from the notation below, since the same logic is applied to each fork node/level l. We do not require that forks consist of exactly two children

(branches).

First, for each normal vector $N_i = (n_i^x, n_i^y, n_i^z)$ associated with a point p_i in the level, we compute its local neighborhood of points j where $\ell_2(p_i, p_j) < r$, where r defines the radius of the neighborhood. We then compute a smoothed version of the normal vector by averaging the normal vectors around its local neighborhood:

$$\overline{N_i} = \frac{\sum_j N_j}{|\sum_j N_j|}.$$

This reduces normal vector noise around the fork.

Second, we map each $\overline{N_i}$ to a Gaussian sphere. Every normal is a unit vector, so if we imagine it as a position, it will lay on the surface of a unit sphere. Mapping all normals onto a unit sphere will produce a Gaussian sphere. Comparing Figure 5.2B (unsmoothened) to Figure 5.2C (smoothened) shows how smoothening tightens the spread of normal vectors and generates a clear pattern of rings. All normal vectors that map onto the same ring belong to the same cylinder, which represents a branch of the fork. Assuming all branches have an approximately cylindrical shape, the goal is to identify the normal vectors that belong to different cylinders/branches. The number of rings to identify is set to the number of children of the fork node in the skeleton graph from Part 1.

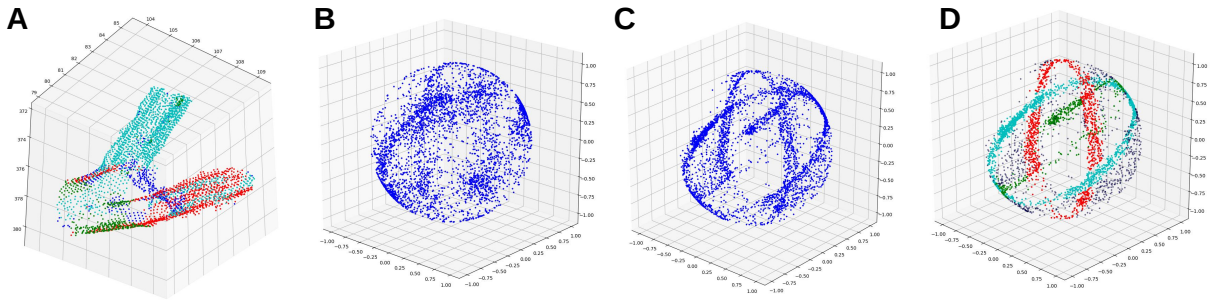

Figure 5.2: **Gaussian spheres.** (A) Points that correspond to an example fork node. (B) Mapping of normal vectors to a Gaussian sphere. (C) Mapping of smoothed normals to a Gaussian sphere. (D) Correspondence between colored rings and original branches in panel A.

For example, the fork in Figure 5.2A has three cylinders (a mother branch and two daughter branches), which matches the three rings we observe in Figure 5.2C; these rings are color-coded in Figure 5.2D to match their original branches.

Finding rings in a Gaussian sphere can be challenging when there is missing data, such as for the mother branch (green) in Figure 5.2A. As a result, the ring of green points is incomplete in Figure 5.2C–D. Next, we discuss how to extract rings that correspond to different cylinders/branches from a Gaussian sphere.

5.3 Sampling of cylinder axes

We extract rings from a Gaussian sphere (Figure 5.3A) by investigating the cross product between normal vectors:

$$\mathcal{S}_i = \overline{N_i} \times \overline{N_j}.$$

If the cross product is taken between two normal vectors that belong to the same ring, then $\mathcal{S}_i$ is a perpendicular vector that points along the axis of the cylinder/branch. If we continue taking cross products between normal vectors that belong to the same ring and mapping them onto the Gaussian sphere, clear clusters will form on the Gaussian sphere. However, the problem is we do not know *a priori* which normal vectors belong to the same ring.

Previously, a method called Pipe-run [QZN14] was proposed to extract pipes from noisy point clouds using Gaussian spheres. Pipe-run draws random pairs of normal vectors and computes their cross products, but this method has several shortcomings. First, it assumes that most of the randomly drawn normal vectors come from the same ring, which works well when the input contains a single cylinder. However, in our case, there are multiple rings, each with a different orientation. Second, each cylinder may not have the same number of points, since branches could vary in thickness and size. Thus, sampling an equal number of points per cylinder could produce spurious axes. Third, cylinders with missing data would be poorly represented by

random sampling. Indeed, Figure 5.3B shows that Pipe-run sampling does not effectively cluster the three cylinders. Other methods, such as RANSAC [FB81], are also highly susceptible to noise and missing data.

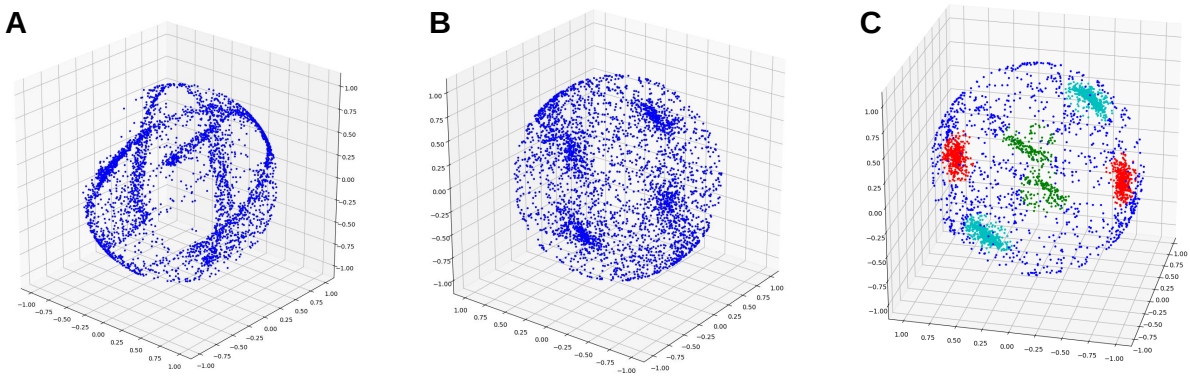

Figure 5.3: **Sampling of cylinder axes.** (A) Mapping of smoothed normals to a Gaussian sphere for an example fork. (B) Cylinder axes identified using Pipe-run (random sampling). (C) Cylinder axes identified using our sampling approach. All rings are represented by well-separated clusters, including the ring with missing data.

Our sampling approach is based on two key observations. First, pairs of normal vectors that belong to the same ring likely have corresponding points that lie close to each other in space. So, we sample pairs of points that lie within a local neighborhood, defined by a radius r. Second, pairs of normal vectors that belong to the same ring will lie on a circle with approximately the same radius. To compute the approximate radius between a point and every point in its neighborhood, we used a method called the radius-based surface descriptor (RSD) [MPB+10]:

$$\ell_2(p_i, p_j) = \sqrt{2r'} \cdot \sqrt{1 - \cos(\angle(\overline{N_i}, \overline{N_j}))}.$$

The RSD takes two points p_i and p_j and their normal vectors as input and computes the approximate radius r' of an assumed circle, if the points were lying on the circle. For example, if the points represented a perfect cylinder, r' would be constant. In reality, the data is noisy and the approximated radii tend to vary.

Given these observations, our sampling method is as follows. Given a point p_i, we compute its approximate radius with all points p_j that lie within a distance r from p_i. We then sort all the approximate radii and choose the point p_{j*} associated with the median radius. Finally, we compute the cross product between the smoothened normal vectors of points p_i and p_{j*}. This results in a single sample per point, and a robust cylinder axis $\mathcal{S}_i$ supported by p_i's neighborhood.

Figure 5.3C shows the results of our sampling method, which reduces noise. There are six clusters corresponding to the three rings because the sign of the cross product depends on the order of operation; thus, we generally expect two opposite vectors for each ring.

5.4 Clustering cylinder axes

The previous step generates a set $\mathcal{S}$ consisting of an $\mathcal{S}_i$ for each point i. Our next goal is to cluster the values in $\mathcal{S}$ such that each cluster corresponds to a different cylinder axis. We performed unsupervised clustering using the mean-shift algorithm [CM02] because it deals well with outliers [QZN14]. The algorithm identifies clusters as locations in space with high sampling density. Formally, the algorithm iterates as follows:

$$a_{\text{next}} = \frac{\sum_{\mathcal{S}_i \in \mathcal{S}} \kappa(\left\| a_{\text{prev}} - \mathcal{S}_i \right\|_2) \cdot \mathcal{S}_i}{\sum_{\mathcal{S}_i \in \mathcal{S}} \kappa(\left\| a_{\text{prev}} - \mathcal{S}_i \right\|_2)},$$

where a_{next} is the current point of highest density, a_{prev} is the previous point, and κ is the radial basis kernel function (RBF). Initially, a_{prev} is selected randomly from $\mathcal{S}$. After every iteration, a_{next} is re-projected back to the surface of the sphere. The process stops when the distance between the current and previous point is very small (e.g., $< 10^{-8}$). The number of clusters is known from the skeleton graph (i.e., the number of branches at the fork).

Once the process converges, we obtain a vector a_i, which represents the cylinder axis of one branch i. Before searching for the next axis, we perform two steps to find additional points that belong to the same axis. First, we find all normal vectors that are within 10 de-

grees of being perpendicular to a_i. The resulting normal vectors and their corresponding points p_i' constitute a ring/cylinder. Second, we remove from set $\mathcal{S}$ all $\mathcal{S}_i$ that are within 10 degrees of a_i. We then repeat the clustering procedure on a new random point to find the next cylinder axis.

5.4.1 A modified clustering algorithm

We modified this algorithm in two ways for better performance for our application.

First, the RBF computes $||a_{\text{prev}} - \mathcal{S}_i||_2$, but as mentioned above, the sign of the cross-product can be inverted based on the order of the points. Thus, vectors that lie on opposite sides belong to the same axis. To account for this, we make the following adjustment:

$$a_{\text{next}} = \frac{\sum_{\mathcal{S}_i \in \mathcal{S}} \kappa(\min(\left\|a_{\text{prev}} - \mathcal{S}_i\right\|_2, \left\|a_{\text{prev}} - (-\mathcal{S}_i)\right\|_2)) \cdot \mathcal{S}_i}{\sum_{\mathcal{S}_i \in \mathcal{S}} \kappa(\min(\left\|a_{\text{prev}} - \mathcal{S}_i\right\|_2, \left\|a_{\text{prev}} - (-\mathcal{S}_i)\right\|_2))}.$$

This allows for more precise computation of the direction of higher density.

Second, we reduce σ (a kernel parameter) as the number of iterations increases. A large σ prevents the algorithm from getting stuck at a local high density location initially; then, as the algorithm settles, we reduce σ to hone in on the highest density location more precisely. This is akin to reducing the learning rate over time during gradient descent optimization. Specifically, when the distance between the current and previous step falls below a threshold, we reduce σ by an order of magnitude. If the distance becomes larger than the threshold, then we continue iterating with the new σ. Otherwise (if the distance remains below the threshold), we say that the algorithm has converged and return. This dynamic adjustment of σ becomes more important in forks with higher numbers of branches.

5.4.2 Dealing with ambiguous points

One final issue remains: each ring (cylinder) of the Gaussian sphere intersects with at least one other ring in two places, and it is unclear to which ring the points at these intersections should be assigned. We call such points *ambiguous*. In other words, when we extract normal vectors that are with 10 degrees of being perpendicular to cylinder axis a_i, there are multiple axes that satisfy this rule. The more branches a fork has, the more intersections there are and the more ambiguity there will be. Figure 5.4A shows another example fork, and Figure 5.4B highlights its ambiguous points in red.

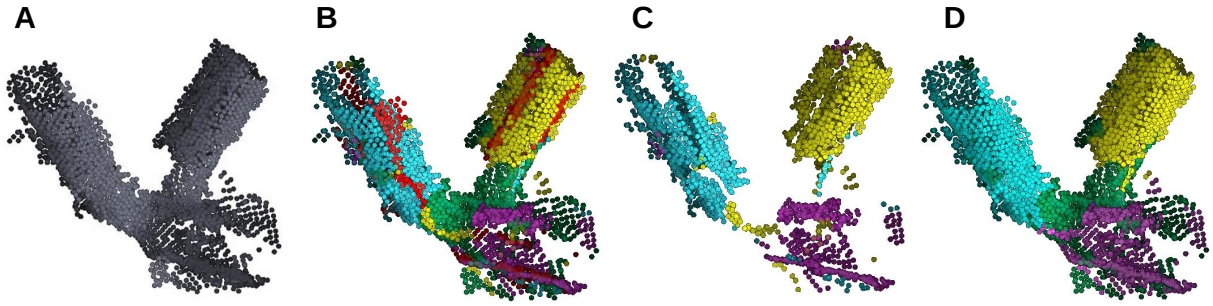

Figure 5.4: **Clustering of ambiguous points.** (A) Initial fork points. (B) Results after initial clustering. Cyan, yellow, purple: non-ambiguous points (i.e., points assigned to a single cylinder axis). Red: ambiguous points. Green: noise (i.e., points not assigned to any ring). (C) Non-ambiguous points only. (D) Final clustering after resolving the assignment of ambiguous points to a single axis/branch. Cylinder accuracy is improved.

To resolve ambiguous points into their correct branches, we perform the following steps. First, we compute the centroid c_i' over all non-ambiguous points p_i' in ring i. Non-ambiguous points are those that were clustered to only one cylinder axis. These points are shown in Figure 5.4B (cyan, yellow, and purple points). Points that were not within 10 degrees of being perpendicular to any a_i are marked as noise and shown in green. Second, viewing each i as a plane with corresponding axis a_i and point c_i', we project each ambiguous point onto each plane i. We then compute the Manhattan distance between each projected point and each c_i'. Ambiguous points are assigned to the ring (axis) with the shortest Manhattan distance.

Figure 5.4D shows the final custering after resolving ambiguous points. Resolution of ambiguous points not only improves cylinder accuracy, but could also be useful for estimating branch volumes.

5.5 Final skeleton graph

The clustering step above produces an axis for each cylinder ring found in the Gaussian sphere. Each of these axes represents a direction of a splitting branch. To generate the final skeleton graph, we need to identify the new position of the branch point, and then connect the new branches to this branch point.

The identified axes (a_i) are 3D vectors that will generally not intersect. We find points of intersection between all pairs of axis vectors by using the skew lines intersection algorithm. The idea is to find the shortest line segments that connect each pair of initial axes. For each pair, the point of intersection is defined as the midpoint of the line segment.

For example, for axes i and $i+1$, we turned the cylinder axes a_i, a_{i+1} and centroids c'_i, c'_{i+1} into 3D line equations:

$$\ell_i := c'_i + \lambda a_i$$

$$\ell_{i+1} := c'_{i+1} + \mu a_{i+1}$$

and solved the linear system of equations $Au = v$ for u where

$$A = \begin{pmatrix} a_i^2 & -a_{i+1} \cdot a_i \\ -a_{i+1} \cdot a_i & a_{i+1}^2 \end{pmatrix} \quad u = \begin{pmatrix} \lambda \\ \mu \end{pmatrix} \quad v = \begin{pmatrix} (c'_{i+1} - c'_i) \cdot a_i \\ -(c'_{i+1} - c'_i) \cdot a_{i+1} \end{pmatrix}$$

Inserting λ and μ back into the linear equations yields the two closest points on those lines, which form a line segment. The center of this line segment is the intersection point of ℓ_i and ℓ_{i+1}. The average of the intersection points, across all pairs of lines, is the new position of

the fork node. Lastly, we create a new node for each branch axis vector i at its centroid c_i', and insert these nodes in the old skeleton graph. We then add an edge between the intersection point and the new nodes for each branch.

For example, in Figure 5.1 (Part 2), we show how a single fork node is divided into three nodes, corresponding to the mother branch and two daughter branches. The graph structure outside the fork node remains the same.

This procedure enhances the geometry of the skeleton graph at fork areas, thus making angle measurements more accurate, as we evaluate next.

5.6 Testing and validation framework

We tested our method on point clouds generated from 3D scans of tomato (*Solanum lycopersicum cv m82D*) and tobacco (*Nicotiana benthamiana*) plants [CPCN17, CPC$^+$17, ZN19]. To capture a wide range of shoot architectures and branch angles, plants were grown under several conditions: an ambient light control, shade, high-temperature, high-light, and drought. These conditions promote diverse phenotypes and are representative of realistic growth environments.

We analyzed 31 forks (18 tomato, 13 tobacco) from 12 plants (8 tomato, 4 tobacco), comprising a total of 76 angles; many forks split two ways, but some split > 2 ways. Lamina were removed from the point clouds via deep learning classification from chapter 2. The remaining points (representing the branching structure of the plant) were then analyzed.

5.7 Angle extraction results

We compared the accuracy and run-time of our method to several popular approaches for skeleton extraction from 3D point clouds: PypeTree* [DJR14, ZN19], L_1-medial skeletonization [HWC$^+$13], and Laplacian-based contraction [CTO$^+$10]. Briefly, PypeTree* [DJR14, ZN19] is

a geometric method specifically tailored for plant biology applications. The L_1-medial skeleton [HWC$^+$13] is a distance field-based algorithm designed to operate on general point clouds. Laplacian-based contraction [CTO$^+$10] is a thinning-based algorithm, and the successor to the well-known ROSA method [TZC09]. Each of these methods represents a different family of skeleton extraction algorithms (see chapter 4 work for details). For each method, we used a grid search to identify the best set of parameters yielding the most visually accurate skeletons. We used the publicly-available code and implementation for each method.

We measured the angles produced at each fork and compared them to ground-truth angles, which were measured manually via cylindrical fitting in third party point cloud visualization software. To test our method's robustness, we selected a range of plants with varying sizes, amounts of missing data, and levels of registration noise. Figure 5.5 shows a subset of 7 of the 12 plants evaluated, consisting of tomato plants spanning four growth conditions.

5.7.1 Accuracy of branch angles

Our method consistently generated accurate skeletons at forks, and remained more robust to noise and missing data compared to other methods. Figure 5.5 shows example forks from tomato plants (columns) and the ground-truth angle calculation for each fork (top row), followed by the skeletons and estimated angles for each of the four methods (subsequent rows). For example, on the Highlight (A20) plant, the ground-truth angle was 120°. The skeleton extracted by our method estimated the angle to be 118°, whereas the estimates of the other methods were much worse: 111°(PypeTree*), 98°(L_1-medial), and 92°(Laplacian).

These seven examples are illustrated because they represent a range of challenges faced by skeletonization algorithms. For example, Highlight (A20), Control (B5), and Shade (A20') all show noisy forks with hairs (trichomes) on the stem and with registration errors; Highlight (A20) and Highlight (A20') show examples of missing data at a fork; and Shade (A20) and Heat (A20) show "webbed" forks.

Each method struggled with overcoming these challenges. For example, L_1-medial often missed small branches due to stochastic sampling (Heat A20, missed two branches). PypeTree* mainly struggled with averaging points from different branches at forks, which tended to pull the branch point away from the main axis, in turn reducing the quality of angles (e.g., Shade (A20') and Heat (A20)). Laplacian was the least resilient to noise (e.g., Shade (A20'), Control (B5)), and closely packed branches often produced loops, spurious branches, and erroneous skeleton shapes (Shade (A20'), Control (B5)). While our method did better at almost every fork, it sometimes struggled with missing data (e.g., Control (B5)).

Over all 18 tomato forks analyzed (from 8 plants in 5 conditions), in terms of absolute errors, our method was on average 1.96 times more accurate than PypeTree*; 2.25 times more accurate than L_1-medial; and 3.15 times more accurate than Laplacian. In terms of percent errors, our method was on average 1.90 times more accurate than PypeTree*; 2.21 times more accurate than L_1-medial; and 2.92 times more accurate than Laplacian.

Importantly, our method demonstrated similar gains when tested on forks from tobacco plants. Over 13 forks from 4 tobacco plants across 2 conditions, in terms of absolute errors, our method was on average 1.39 times more accurate than PypeTree*; 2.13 times more accurate than L_1; and 2.16 times more accurate than Laplacian. In terms of percent errors, our method was on average 1.26 times more accurate than PypeTree*; 1.99 times more accurate than L_1-medial; and 1.99 times more accurate than Laplacian. Thus, our method was able to generalize well across two distinct species.

Results from all 31 forks are quantified in Figure 5.6, demonstrating our method's improved estimation of branch angles in terms of absolute errors (Figure 5.6A) and percent errors (Figure 5.6B).

Table 5.1: **Time comparison for all methods.** All measurements shown are in seconds.

Species	Point cloud	Our method	PypeTree	Laplacian	L_1-medial
Tomato	Control (B5)	1.40	1.26	41.77	3.76
Tomato	Control (B20)	6.13	5.25	494.49	5.69
Tomato	Heat (A20)	2.51	2.06	53.91	4.05
Tomato	Highlight (A20)	1.68	1.34	47.44	3.06
Tomato	Highlight (A20')	1.89	1.62	77.95	5.22
Tomato	Shade (A20)	2.68	2.19	68.12	3.05
Tomato	Shade (A20')	2.27	2.03	91.01	2.98
Tomato	Drought (A12)	1.27	1.01	73.22	4.49
Tomato	Highlight (B4)	0.75	0.67	20.56	3.64
Tomato	Shade (B20)	9.73	7.98	491.51	9.99
Tobacco	Control (B6)	0.27	0.23	13.77	4.34
Tobacco	Control (B12)	0.82	0.64	21.66	4.20
Tobacco	Heat (B6)	0.25	0.19	14.98	2.42
Tobacco	Heat (B20)	7.37	5.49	360.03	5.77

5.7.2 Comparison of run-time

For each fork, we compared the amount of time each method took to run. Measurements were made on a Windows 10 Pro 64-bit machine with an Intel i7-2600 CPU @ 3.4 GHz, and 16GB RAM. Because the L_1-medial method is stochastic (i.e., it requires selection of random seeds during initialization), we averaged its run-time over four runs.

Table 5.1 shows that PypeTree* and our method are the two fastest algorithms in our comparison. On average, our method is only 0.17 times slower than PypeTree*, suggesting that the error correction process we applied on top of the original PypeTree algorithm only leads to a small loss in timing efficiency. The other two methods (L_1-medial and Laplacian) are significantly slower. Specifically, our method is 2.81 times faster than L_1-medial and 41 times faster than Laplacian. L_1-medial was fairly competitive on larger plants with more cloud points, but showed significantly slower times on smaller plants compared to PypeTree* and our method.

In summary, our method can compute branch angles for large plants in seconds, and generally produces more accurate branch angles than existing methods.

5.8 Acknowledgments

Materials from this chapter has been submitted for publication as it may appear in Remote Sensing 2021, Ziamtsov, Illia; Faizi, Kian; Navlakha, Saket. The dissertation author was the primary investigator and author of this paper.

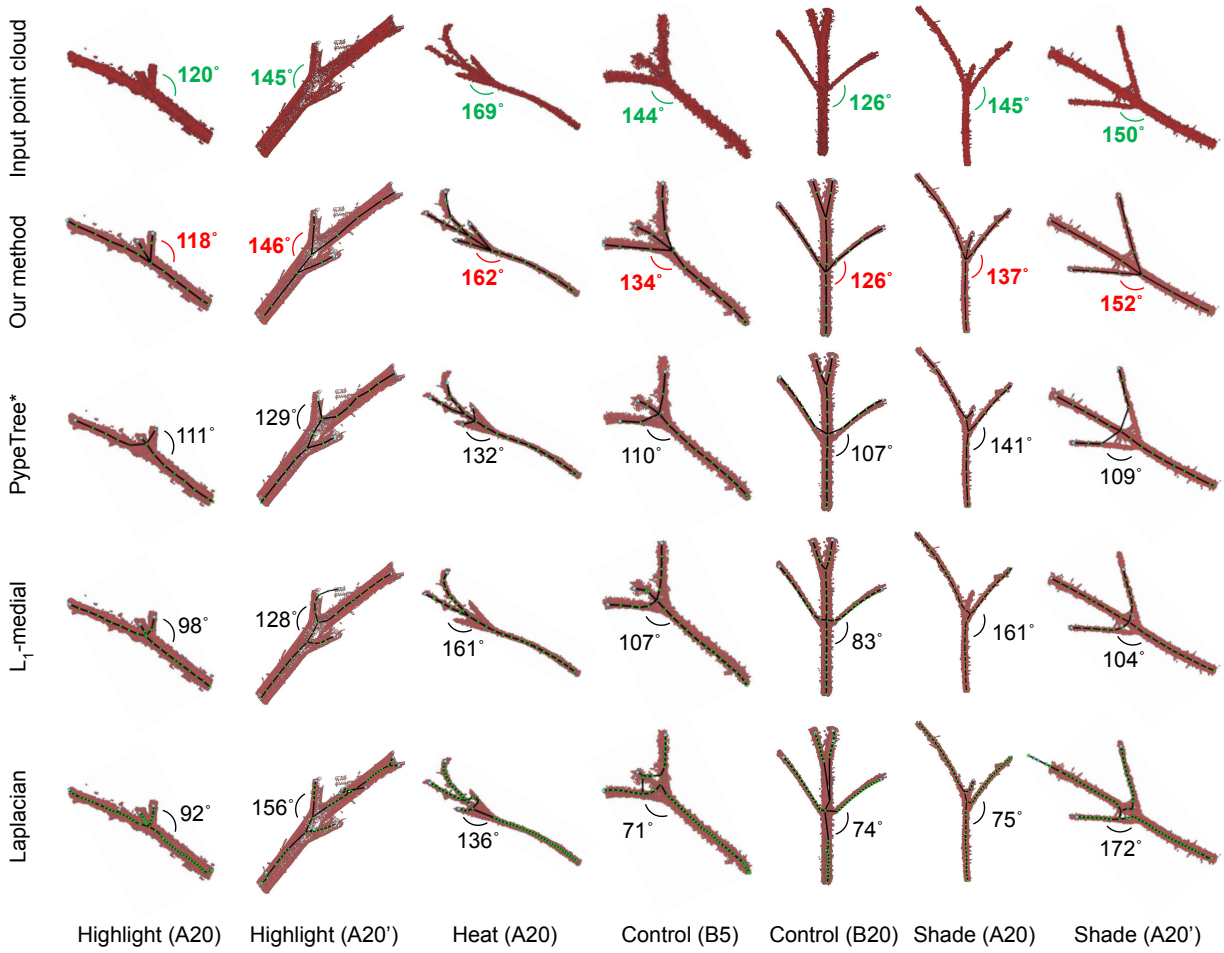

Figure 5.5: **Branch angle prediction for four methods.** Seven forks are shown (columns), each from a tomato plant across different conditions. The name "Highlight (A20)" means the fork comes from a plant grown in highlight conditions scanned on developmental day 20. Plants are given names "A" or "B" for replicates. The apostrophe symbol denotes a different fork that belongs to the same plant. Each row shows the skeletons and estimated angles for each of the four methods; the top row shows the ground-truth angle, computed manually. Only one angle is highlighted in each fork.

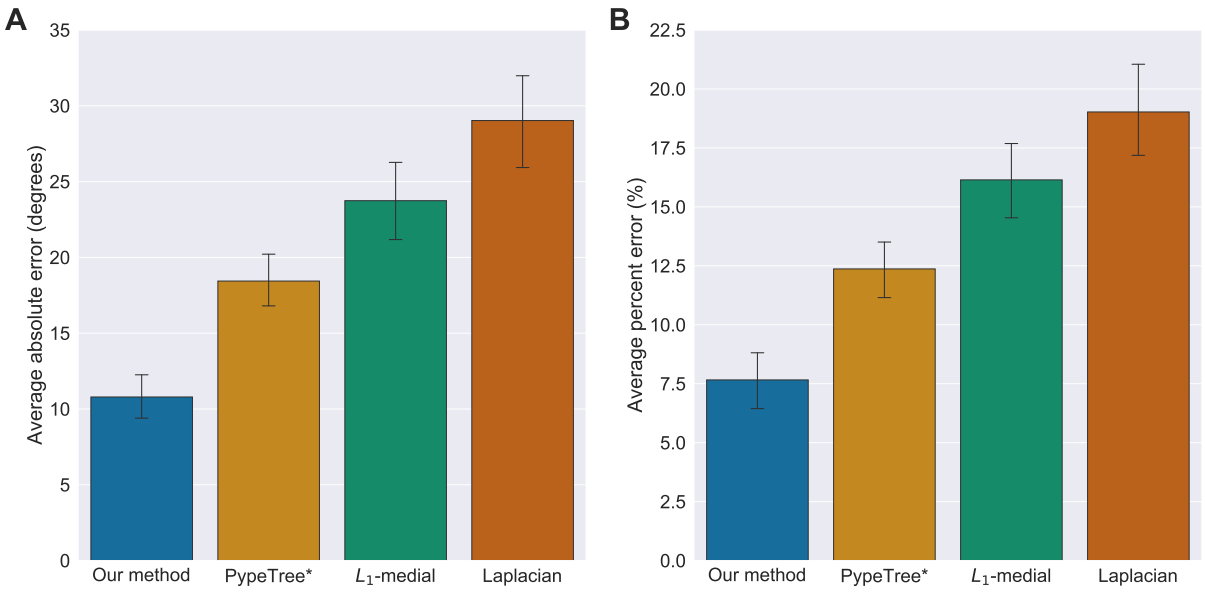

Figure 5.6: **Accuracy of predicted angles for all methods.** (A) Average absolute errors for each method, defined as the absolute value of the ground-truth angle minus the predicted angle. Error bars represent the standard error of the mean. (B) Average percent errors for each method, defined as the ratio of the absolute error to the ground-truth angle. Error bars represent the standard error of the mean.

Chapter 6

P3D: Phenotyping toolbox

Developing methods to efficiently analyze 3D point cloud data of plant architectures remains challenging for many phenotyping applications. New technological advances in plant phenotyping are now being used to generate large volumes of data detailing the 3D architectures of various plant species, grown in different environments and conditions, in both the lab and in the field. This has raised the challenge of automatically extracting phenotyping features of interest, such as leaf size, shape, and quantity [WZC+16, HWQ+18], branch lengths and angles [BABA+17], and growth rates [MBdS+17], amongst others. These features are important for numerous tasks, including quantifying plant biomass and yield [MBR16], understanding plant responses to stressful conditions [MFST16], mapping genotypes to phenotypes, and building predictive structural-functional models of plant growth [BBB+17, VEBS+10, SGDN14, GS05].

In this chapter, we present a toolbox that integrates the algorithmic methods developed in this thesis. The four core phenotyping tasks it addresses are: classification of cloud points into stem and lamina points; graph skeletonization of the stem points; segmentation of individual lamina; and whole leaf labeling. The Plant3D (P3D) tool provides an intuitive graphical user interface, a fast 3D rendering engine for visualizing plants with millions of cloud points, and several graph-theoretic and machine learning algorithms for 3D architecture analyses. The P3D

tool is an aggregation of the methods described in this thesis into one application. As 3D point clouds become a standard data type for digitizing plant architectures in the lab and in the field, we hope the P3D tool can help accelerate next-generation plant phenotyping.

6.1 Motivation

While many tools currently exist for plant phenotyping, these tools have mostly been developed to analyze images of plant architectures [HHK+15, NSM+15, US17]. While camera-based imaging offers a cheap and high-throughput data capture solution, subsequent analysis faces numerous challenges, including camera calibration, image registration, object occlusion, and inconsistencies in illumination — especially as these factors vary across labs and experimental setups [PSNEC17]. More recently, light detection and ranging (LiDAR) and 3D laser scanning have become attractive alternatives to imaging. These technologies generate point cloud representations of plants with micron-level precision that can capture minute details, including trichomes and leaf venation patterns [Pau19, LFM+13, GSA+12b, CWT+18], and do not require background subtraction nor extensive calibration across setups.

Our goal was to publish an application that is capable to perform four basic phenotyping tasks using 3D point clouds. In the effort to make P3D more accessible for plant community, we equipped it with an easy to navigate graphical user interface, the ability to import different file types for seamless usage, and an installation via Windows installer executable file. The latter allows to avoid compiling and building from source code, which could be a major hurdle for people unfamiliar with nuances of C++.

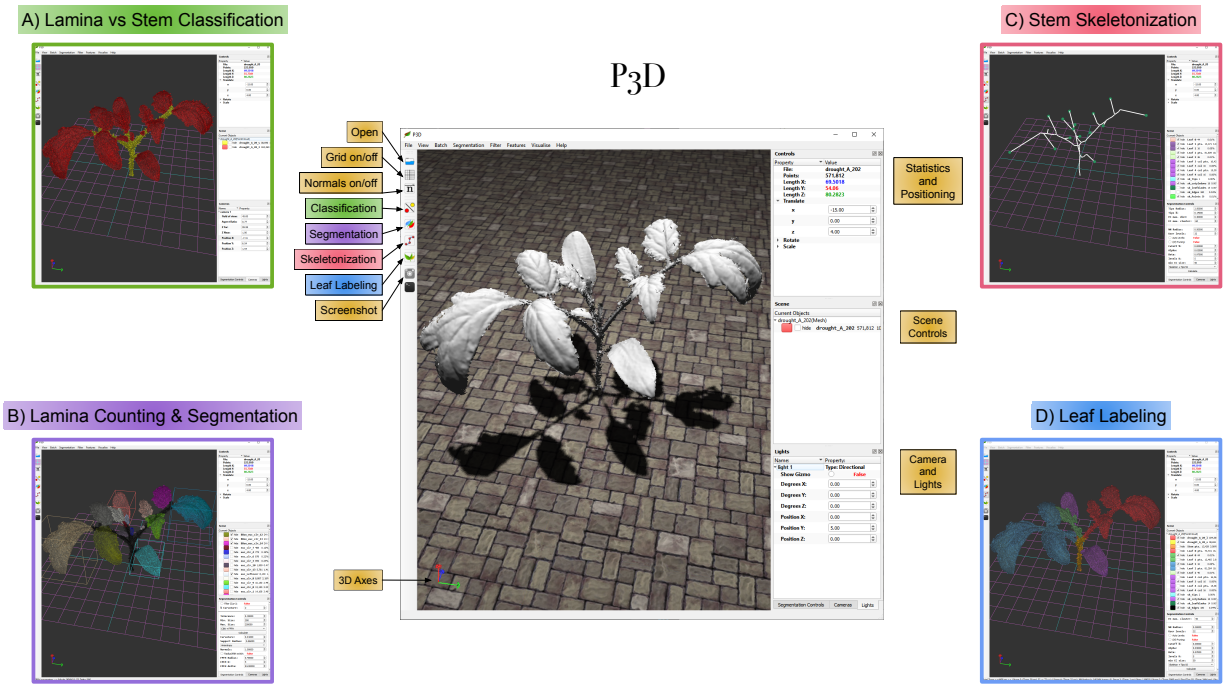

Figure 6.1: **Overview of the P3D tool.** Center panel shows the overall GUI and an example scan of a tomato plant on growth day 20. Examples of the four phenotyping tasks are shown in panels A–D. A) Classification of the point cloud into lamina points (red) and stem points (yellow) using a deep learning classifier. B) Segmentation of the lamina points into individual lamina (shown in different colors). C) Skeletonization of the stem points into a 3D graph-theoretic tree. White lines show the tree, and green dots signify the 3D locations of lamina found in the previous step. D) Clustering of individual lamina into whole leaves. Three leaves are shown (red, green, and blue), as well as two cotyledons (purple). Orange points signify stem points.

6.2 The Plant 3D (P3D) software package

We developed a tool called Plant 3D (P3D) to automatically perform common phenotyping tasks using high-resolution 3D scans of plant architectures. P3D is open-source and is bundled with a stand-alone Windows application (Figure 6.1). P3D is written in C++ using OpenGL, QT, TensorFlow, and the point cloud library (PCL) [RC11]. P3D can visualize and process data imported as a 3D point cloud (`pcd` or `txt` formats) or a mesh (`obj` format).

The tool focuses on four phenotyping tasks described below. The algorithms developed for these tasks are novel and were recently shown to improve accuracy and/or run-time compared to existing methods on a large dataset of 3D plant architectures, including two species (tomato, tobacco), each grown in multiple environments, and through roughly 20 days of development [ZN19].

6.2.1 Lamina vs. stem classification

The first task is to classify each point in the point cloud as belonging to either a lamina structure (e.g., leaf, cotyledon) or a stem structure (e.g., petiole, branch) (Figure 6.1A). A point cloud consists of a list of (x, y, z) coordinates of the object of interest (a plant architecture). P3D computes fast point feature histogram features [RBB09] for every point in the cloud, and then feeds these features into a deep neural network for binary classification. This classification is the first step towards computing leaf-related features (using lamina points) and morphology/shape-related features (using stem points).

Because plants show remarkable plasticity in their form, P3D provides a comprehensive classification model pre-trained on diverse architectures grown in different environments (ambient, high-light, high-heat, drought, shade) and at different developmental time-points (from day 5 to day 20). In addition, P3D allows users to provide a path to their own pre-trained network.

P3D's rendering engine allows the classification results to be visualized quickly, with

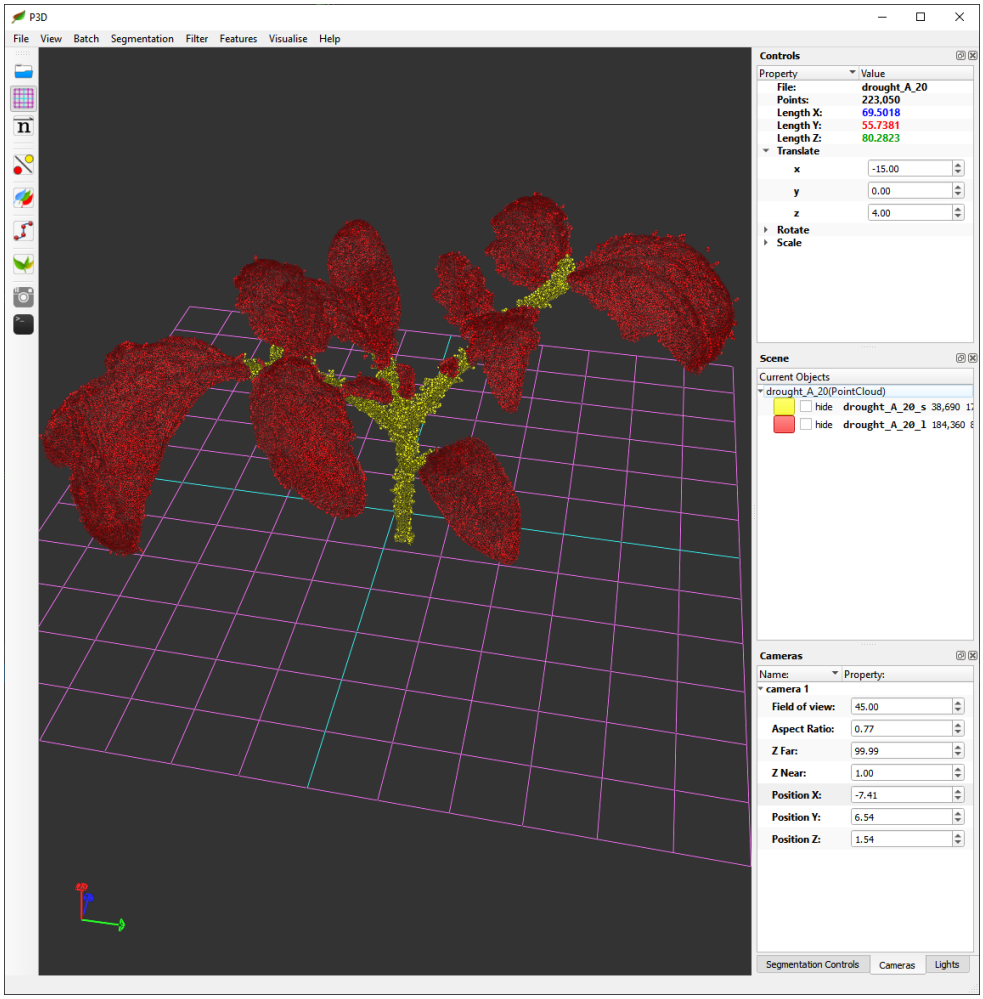

Figure 6.2: **P3D: Classification of branch and leaf points.** Classification of points within P3D.

zooming and panning capabilities for intuitive evaluation of performance.

6.2.2 Lamina counting and segmentation

The second task is to segment or cluster all of the lamina points into subsets, where each subset represents an individual lamina (Figure 6.1B). P3D uses a conditional region growing method that clusters lamina points based on similarity of their curvature (anisotropy), normal vectors, and fast point feature histogram features. These features together improve the identification of transition points between lamina blades, where the continuity of the architecture shifts.

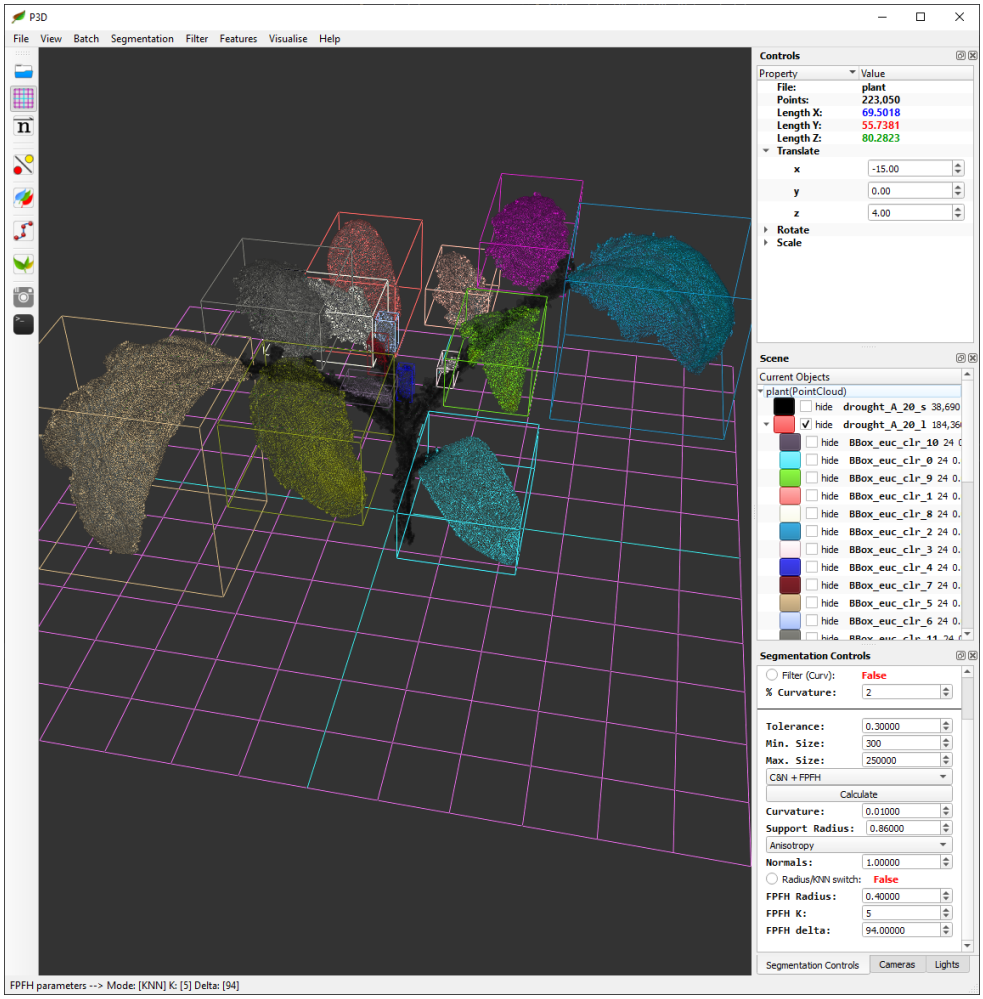

Figure 6.3: **P3D: Leaf segmentation.** Leaf segmentation within P3D.

This step produces a set of lamina — which are individually colored — and can then be used for analyses of lamina size, shape [WZC+16], arrangement [RK02, Kuh17], and quantity [GDT18].

6.2.3 Stem skeletonization

The third task is to generate a skeleton of the stem points (Figure 6.1C). P3D provides a skeletonization algorithm, which outputs a graph-theoretic tree, with nodes corresponding to the root, lamina points, or branch points, and with edges corresponding to the underlying stem structure connecting the nodes. Morphological analyses of plant architectures provides

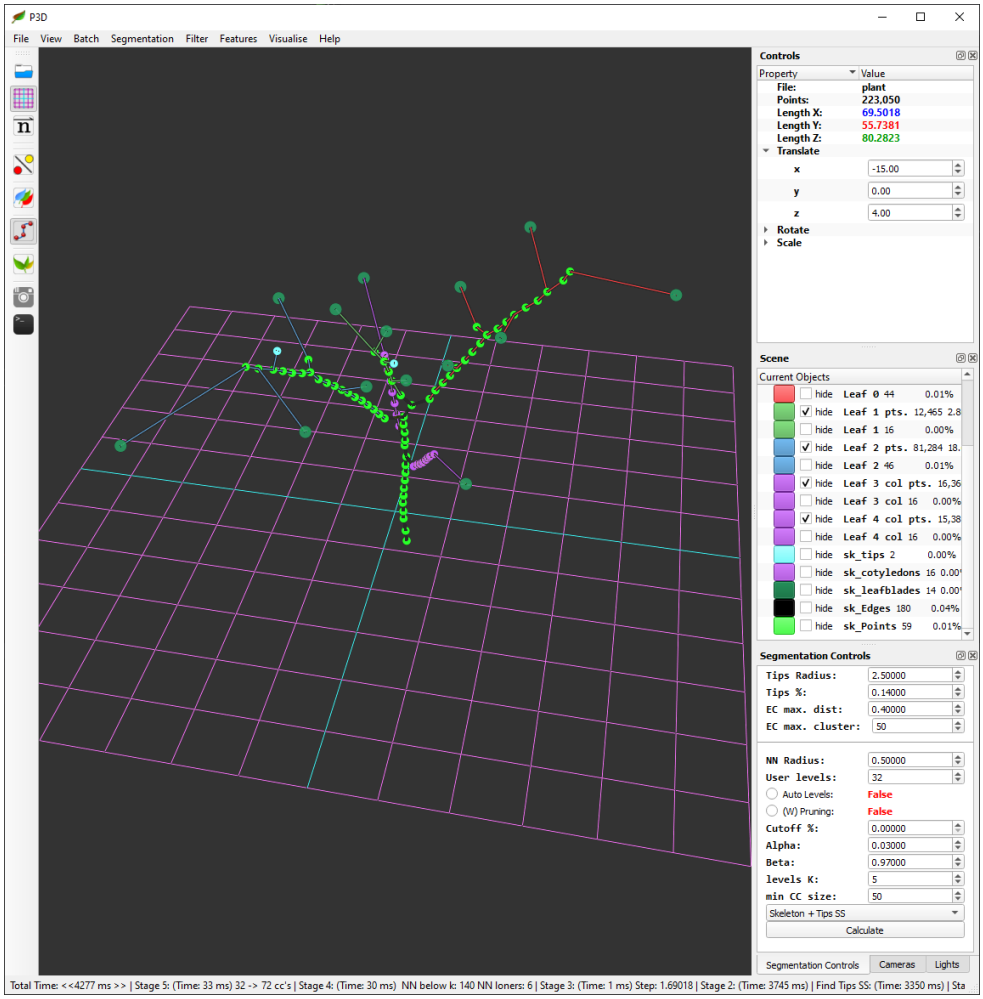

Figure 6.4: **P3D: Skeletanization.** Skeletanization within P3D.

numerous informative traits for understanding plant geometry [BABA+17], nutrient transport efficiency [CPCN17], spatial distribution of branches [Con], and branching morphogenesis [PL96], amongst others.

6.2.4 Whole leaf labeling

The fourth task builds upon the first three tasks and identifies whole leaves (Figure 6.1D). Biologically, a single "leaf" of a tomato plant consists of all the stem tissue (petioles and petioules) and all the individual lamina that are downstream of a single branch-point from the primary

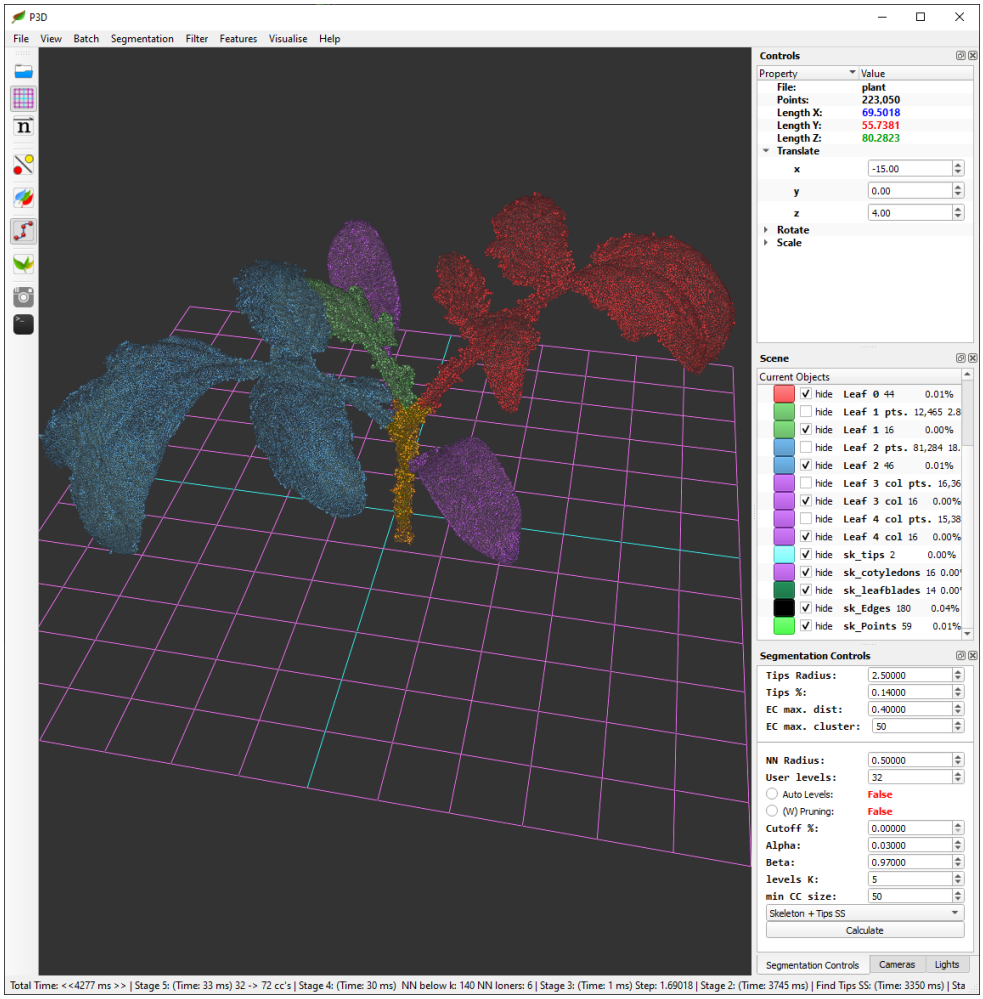

Figure 6.5: **P3D: Leaf labeling.** An example of leaf labeling within P3D.

stem. These components are grouped together based on the outputs of tasks two and three. This definition of a "leaf" is more commonly used in some communities, and similar analyses of leaf size, shape, and counting can be then performed.

6.2.5 Additional features

P3D includes additional features useful in practice, including methods to downsample very large point clouds, methods to remove outliers and smoothen the data, and a shading editor to model light capture by plants. There is, of course, significant diversity in plant structures

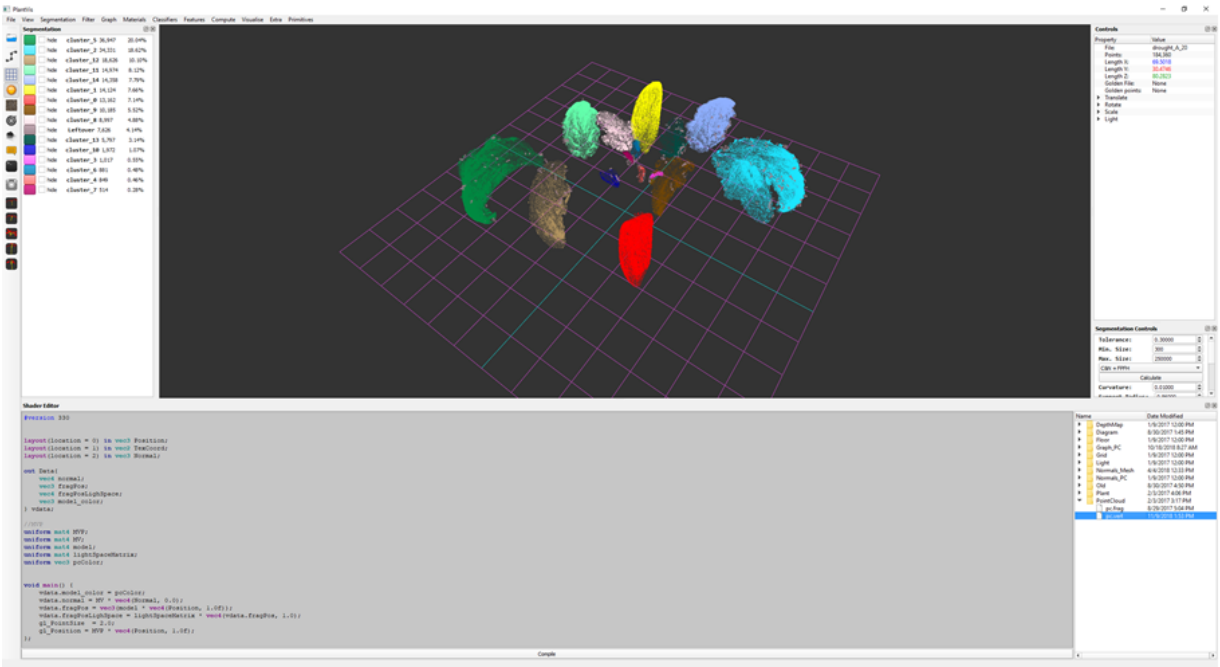

Figure 6.6: **P3D: GLSL live shader editor.** Bottom half of the screen shows a color coded syntax of GLSL editor. User can edit any of the shader files and compile them to see the result without quitting the application.

across the entire plant kingdom, and our methods currently have only been tested on Solanaceous species. However, P3D has a modular structure, which can be extended to handle other leaf and branching structures in the future.

6.3 Acknowledgments

This chapter uses materials that were published in Bioinformatics 36(12). Ziamtsov, Illia; Navlakha, Saket., 3949-3950 2020. The dissertation author was the primary investigator and author of this material. Instructions and Windows installer are freely available at: *https://github.com/iziamtso/P3D/*.

Chapter 7

Conclusion

In this work, we presented a number of algorithmic solutions aimed at four basic phenotyping tasks for 3D point clouds of plant shoot architectures. These four tasks are critical for numerous downstream phenotyping goals, such as quantifying plant biomass, performing morphological analyses of plant shapes, and uncovering genotype to phenotype relationships. We tested our algorithms on high-resolution 3D point cloud data from two species of plants (tomato and tobacco) and showed an improvement in the accuracy and timing compared to competing approaches. Compared to prior approaches that only addressed these problems on a small number of plants or plants from only a single species [GDHB17, PDMK13, WPKM15, MTM16], our dataset encompassed 3 time-points, 3–5 growth environments, and 2 species, which allowed us to test some generalization ability of our algorithms. Our algorithms have the potential to be deployed in large-scale phenotyping applications with noisy point clouds, and is designed to be used with minimal pre-processing.

There are several lessons we learned from applying these methods to noisy plant architectures. *For classification*, carefully chosen feature vectors along with standard deep learning classifier yielded good results. New approaches exist that can omit the necessity for hand-curated feature vectors. While our approach was satisfactory for the phenotyping problems faced here,

other problems may benefit from more automated methods for feature extraction. We showed how a deep learning classifier using fast point feature histogram features can achieve over 95% accuracy in classifying leaf versus branch points, outperforming many other features and classifier types. We also tested how well this approach generalized to instances where training data is not available for a test condition of interest; e.g., testing on a new species or new growth condition. This is an important yet at times understudied component of validating phenotyping algorithms, especially since many different architectures can be formed within the same genotype. Overall we achieved good results in terms of quality of classification. The speed of classification was contingent on the initial model and feature computation. While the model had to be computed only once, the feature vectors for every point had to be computed not only for the model computation but also for every inference. This suggests that the feature approach is better suited for small to mid size datasets. *For leaf segmentation*, we extended a classic region growing algorithm to improve segmentation and counting of individual leaves. We were able to improve leaf segmentation in the areas where leaves overlap at the later stages of a plant's maturity. The larger the number of accurately segmented leaves the more accurate the morphological analysis of their surfaces and plant's growth status will be. In order to push the leaf segmentation accuracy further, a more globally aware approach will probably be needed. For example, we could augment the segmentation algorithm with the locations of petioles derived from the skeleton graph. *For skeletonization*, we developed an enhanced branch skeletonization algorithm with more visually aligned skeletons, improved tip identification, faster run-time, and more accurate fork angles. An accurate angle estimation benefits from considering normal vector directions at forks. Spawning branches have small angles and can be very close to each other, which causes points to cluster together; using normal vectors to distinguish each branch thus becomes critical. Taking advantage of some plant-specific geometric characteristics — such as the assumption that branches are cylindrical, the absence of loops in the skeleton, and the tapering of branch radii — can lead to improvements in overall skeleton quality and performance. This improvement in accuracy of

overall quality of skeletons will benefit down the line analysis of branching structure and help with tasks such as robotic branch pruning.

There are several avenues for future work. First, advances in point cloud analysis using deep learning without feature extraction have been recently proposed [CSKG17, QYSG17]; however, there remain challenges in using this technique when plant size varies. Second, we tested our branch skeletonization algorithm on shoot architectures, but it is possible that a very similar method can also be applied to skeletonizing root system architectures to study foraging behavior [MGG$^+$17, APBW19, SA17, DKVH$^+$09, FGRY17, ZIBE11]. Third, extracting leaf veins from individually identified leaves may provide another useful leaf characteristic to aid segmentation, and may enable study of vasculature [DK01, ND97]. Fourth, the identification of forks requires that each level set (colored points in Figure 5.1) is sampled with enough points from each branch. PypeTree* generates each level based on distances from the root, but sometimes this approach places a border between two adjacent levels that lies right in the middle of a fork. While uncommon, this can compromise the quality of predicted angles. Thus, designing automated methods that improve the identification of levels such that the borders of adjacent levels do not cut through a fork remains an open challenge. Fifth, skeleton quality can be improved by better calculating the positions of nodes within levels. Currently, our method defines a node in the skeleton graph to be the centroid of the points in the level. Developing a method that considers normals when calculating the centroid could improve not only angle node predictions, but also the quality of other nodes in the skeleton graph. Sixth, when we insert new nodes into the skeleton after clustering, we assume that all branches emit from the same node; while this is certainly true in many cases, there are times when branches can be offset relative to each other. Adding a more clever node-merging mechanism would further improve the quality of skeletons. Finally, while we designed our method to be species-agnostic, we only tested it on two Solanaceous species: tomato (*Solanum lycopersicum cv m82D*) and tobacco (*Nicotiana benthamiana*). Evaluating it on broader datasets with more diverse phenotypes could identify additional challenges.

Bibliography

[AC08] Cumhur Aydinalp and Malcolm S Cresser. The effects of global climate change on agriculture. *American-Eurasian Journal of Agricultural & Environmental Sciences*, 3(5):672–676, 2008.

[AFB+12] G. Arbeiter, S. Fuchs, R. Bormann, J. Fischer, and A. Verl. Evaluation of 3d feature descriptors for classification of surface geometries in point clouds. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1644–1650, Oct 2012.

[APBW19] J. A. Atkinson, M. P. Pound, M. J. Bennett, and D. M. Wells. Uncovering the hidden half of plants using new advances in root phenotyping. *Curr. Opin. Biotechnol.*, 55:1–8, Feb 2019.

[ARAM+13] Dionisio Andújar, Victor Rueda-Ayala, Hugo Moreno, Joan Ramón Rosell-Polo, Alexandre Escolá, Constantino Valero, Roland Gerhards, César Fernández-Quintanilla, José Dorado, and Hans-Werner Griepentrog. Discriminating crop, weeds and soil surface with a terrestrial lidar sensor. *Sensors*, 13(11):14662–14675, 2013.

[ATC+08] Oscar Kin-Chung Au, Chiew-Lan Tai, Hung-Kuo Chu, Daniel Cohen-Or, and Tong-Yee Lee. Skeleton extraction by mesh contraction. *ACM Transactions on Graphics*, 27(3):1–10, August 2008.

[BABA+17] Alexander Bucksch, Acheampong Atta-Boateng, Akomian F. Azihou, Dorjsuren Battogtokh, Aly Baumgartner, Brad M. Binder, Siobhan A. Braybrook, Cynthia Chang, Viktoirya Coneva, Thomas J. DeWitt, Alexander G. Fletcher, Malia A. Gehan, Diego Hernan Diaz-Martinez, Lilan Hong, Anjali S. Iyer-Pascuzzi, Laura L. Klein, Samuel Leiboff, Mao Li, Jonathan P. Lynch, Alexis Maizel, Julin N. Maloof, R. J. Cody Markelz, Ciera C. Martinez, Laura A. Miller, Washington Mio, Wojtek Palubicki, Hendrik Poorter, Christophe Pradal, Charles A. Price, Eetu Puttonen, John B. Reese, Rubén Rellán-Álvarez, Edgar P. Spalding, Erin E. Sparks, Christopher N. Topp, Joseph H. Williams, and Daniel H. Chitwood. Morphological plant modeling: Unleashing geometric and topological potential within the plant sciences. *Frontiers in Plant Science*, 8:900, 2017.

[BBB+17] M. Balduzzi, B. M. Binder, A. Bucksch, C. Chang, L. Hong, A. S. Iyer-Pascuzzi, C. Pradal, and E. E. Sparks. Reshaping Plant Biology: Qualitative and Quantitative Descriptors for Plant Morphology. *Front Plant Sci*, 8:117, 2017.

[BLM09] A. Bucksch, R. C. Lindenbergh, and M. Menenti. Skeltre - fast skeletonisation for imperfect point cloud data of botanic trees. In *Proceedings of the 2Nd Eurographics Conference on 3D Object Retrieval*, 3DOR '09, pages 13–20, Aire-la-Ville, Switzerland, Switzerland, 2009. Eurographics Association.

[BLM10] Alexander Bucksch, Roderik Lindenbergh, and Massimo Menenti. Skeltre: Robust skeleton extraction from imperfect point clouds. *VIS. COMPUT*, 26:1283–1300, 2010.

[BRA+14] Bojana Bajželj, Keith S Richards, Julian M Allwood, Pete Smith, John S Dennis, Elizabeth Curmi, and Christopher A Gilligan. Importance of food-demand management for climate mitigation. *Nature Climate Change*, 4(10):924–929, 2014.

[Buc14] Alexander Bucksch. A practical introduction to skeletons for the plant sciences. *Applications in Plant Sciences*, 2(8), 2014.

[CAE+16] Somrita Chattopadhyay, Shayan A Akbar, Noha M Elfiky, Henry Medeiros, and Avinash Kak. Measuring and modeling apple trees using time-of-flight data for automation of dormant pruning applications. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9. IEEE, 2016.

[CM02] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, May 2002.

[Con]

[CPC+17] A. Conn, U. V. Pedmale, J. Chory, C. F. Stevens, and S. Navlakha. A Statistical Description of Plant Shoot Architecture. *Curr. Biol.*, 27(14):2078–2088, Jul 2017.

[CPCN17] A. Conn, U. V. Pedmale, J. Chory, and S. Navlakha. High-Resolution Laser Scanning Reveals Plant Architectures that Reflect Universal Network Design Principles. *Cell Syst*, 5(1):53–62, Jul 2017.

[CRL+12] Yann Chene, David Rousseau, Philippe Lucidarme, Jessica Bertheloot, Valérie Caffier, Philippe Morel, Étienne Belin, and François Chapeau-Blondeau. On the use of depth camera for 3d phenotyping of entire plants. *Computers and Electronics in Agriculture*, 82:122 – 127, 2012.

[CRT10] Alvaro Calzadilla, Katrin Rehdanz, and Richard S.J. Tol. The economic impact of more sustainable water use in agriculture: A computable general equilibrium analysis. *Journal of Hydrology*, 384(3):292–305, 2010. Green-Blue Water Initiative (GBI).

[CSKG17] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, July 2017.

[CSM07] Nicu D. Cornea, Deborah Silver, and Patrick Min. Curve-Skeleton Properties, Applications, and Algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):530–548, May 2007.

[CTO+10] Junjie Cao, Andrea Tagliasacchi, Matt Olson, Hao Zhang, and Zhinxun Su. Point Cloud Skeletons via Laplacian Based Contraction. In *2010 Shape Modeling International Conference*, pages 187–197, Aix-en-Provence, France, June 2010. IEEE.

[CWT+18] Ayan Chaudhury, Christopher Ward, Ali Talasaz, Alexander G Ivanov, Mark Brophy, Bernard Grodzinski, Norman PA Hüner, Rajnikant V Patel, and John L Barron. Machine vision system for 3d plant phenotyping. *IEEE/ACM transactions on computational biology and bioinformatics*, 16(6):2009–2022, 2018.

[DJR14] Sylvain Delagrange, Christian Jauvin, and Pascal Rochon. Pypetree: A tool for reconstructing tree perennial tissues from point clouds. *Sensors (Basel, Switzerland)*, 14:4271–89, 03 2014.

[DK01] Nancy Dengler and Julie Kang. Vascular patterning and leaf shape. *Current opinion in plant biology*, 4(1):50–56, 2001.

[DKVH+09] H. De Kroon, E. J. Visser, H. Huber, L. Mommer, and M. J. Hutchings. A modular concept of plant foraging behaviour: the interplay between local responses and systemic control. *Plant Cell Environ.*, 32(6):704–712, Jun 2009.

[dMFK10] Renato Prata de Moraes Frasson and Witold F. Krajewski. Three-dimensional digital model of a maize plant. *Agricultural and Forest Meteorology*, 150(3):478 – 488, 2010.

[DMS12] D. Dey, L. Mummert, and R. Sukthankar. Classification of plant structures from uncalibrated image sequences. In *2012 IEEE Workshop on the Applications of Computer Vision (WACV)*, pages 329–336, Jan 2012.

[DS06] Tamal K Dey and Jian Sun. Defining and computing curve-skeletons with medial geodesic function. In *Symposium on Geometry Processing*, volume 6, pages 143–152, 2006.

[FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[FGRY17] R. J. Flavel, C. N. Guppy, S. M. R. Rabbi, and I. M. Young. An image processing and analysis tool for identifying and analysing complex plant root systems in 3D soil using non-destructive analysis: Root1. *PLoS ONE*, 12(5):e0176433, 2017.

[FS13] Fabio Fiorani and Ulrich Schurr. Future scenarios for plant phenotyping. *Annual Review of Plant Biology*, 64(1):267–291, 2013. PMID: 23451789.

[FT11] Robert T. Furbank and Mark Tester. Phenomics – technologies to relieve the phenotyping bottleneck. *Trends in Plant Science*, 16(12):635 – 644, 2011.

[GBC+10] H Charles J Godfray, John R Beddington, Ian R Crute, Lawrence Haddad, David Lawrence, James F Muir, Jules Pretty, Sherman Robinson, Sandy M Thomas, and Camilla Toulmin. Food security: the challenge of feeding 9 billion people. *science*, 327(5967):812–818, 2010.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016.

[GBS+16] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, Jianwei Wan, and Ngai Ming Kwok. A comprehensive performance evaluation of 3d local feature descriptors. *International Journal of Computer Vision*, 116(1):66–89, Jan 2016.

[GDHB17] William Gélard, Michel Devy, Ariane Herbulot, and Philippe Burger. Model-based Segmentation of 3D Point Clouds for Phenotyping Sunflower Plants. In *12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP 2017)*, volume 4, pages 459–467, Porto, Portugal, February 2017.

[GDT18] M. V. Giuffrida, P. Doerner, and S. A. Tsaftaris. Pheno-Deep Counter: a unified and versatile deep learning architecture for leaf counting. *Plant J.*, 96(4):880–890, 11 2018.

[GES+18] Tianshuang Gao, Hamid Emadi, Homagni Saha, Jiaoping Zhang, Alec Lofquist, Arti Singh, Baskar Ganapathysubramanian, Soumik Sarkar, Asheesh K. Singh, and Sourabh Bhattacharya. A novel multirobot system for plant phenotyping. *Robotics*, 7(4), 2018.

[GMGB17] Marco Giovannetti, Anna Małolepszy, Christian Göschl, and Wolfgang Busch. Large-Scale Phenotyping of Root Traits in the Model Legume Lotus japonicus. In *Plant Genomics*, volume 1610, pages 155–167. Springer New York, New York, NY, 2017.

[GPF+18] Jonathon Ashley Gibbs, Michael P Pound, Andrew P French, Darren M Wells, Erik H. Murchie, and Tony P Pridmore. Plant phenotyping: An active vision cell for three-dimensional plant shoot reconstruction. *Plant Physiology*, 2018.

[GS05] C. Godin and H. Sinoquet. Functional-structural plant modelling. *New Phytol.*, 166(3):705–708, Jun 2005.

[GSA+12a] Louarn Gaetan, Carre Serge, Eprinchard Annie, Combes Didier, and Boudon Frederic. Characterization of whole plant leaf area properties using laser scanner point clouds. In *2012 IEEE 4th International Symposium on Plant Growth Modeling, Simulation, Visualization and Applications*, pages 250–253, Shanghai, China, October 2012. IEEE.

[GSA+12b] L. Gaëtan, C. Serge, E. Annie, C. Didier, and B. Frédéric. Characterization of whole plant leaf area properties using laser scanner point clouds. In *2012 IEEE 4th International Symposium on Plant Growth Modeling, Simulation, Visualization and Applications*, pages 250–253, Oct 2012.

[GWC+21] Morteza Ghahremani, Kevin Williams, Fiona M. K. Corke, Bernard Tiddeman, Yonghuai Liu, and John H. Doonan. Deep segmentation of point clouds of wheat. *Frontiers in Plant Science*, 12:429, 2021.

[HF05] M Sabry Hassouna and Aly A Farag. Robust centerline extraction framework using level sets. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 458–465. IEEE, 2005.

[HHK+15] S. Heckwolf, M. Heckwolf, S. M. Kaeppler, N. de Leon, and E. P. Spalding. Image analysis of anatomical traits in stalk transections of maize and other grasses. *Plant Methods*, 11:26, 2015.

[HJW+17] Xian-Feng Han, Jesse S Jin, Ming-Jie Wang, Wei Jiang, Lei Gao, and Liping Xiao. A review of algorithms for filtering the 3d point cloud. *Signal Processing: Image Communication*, 57:103–112, 2017.

[HSC+15] Jan Hackenberg, Heinrich Spiecker, Kim Calders, Mathias Disney, and Pasi Raumonen. Simpletree—an efficient open source tool to build tree models from tls clouds. *Forests*, 6(11):4245–4294, 2015.

[HWC+13] Hui Huang, Shihao Wu, Daniel Cohen-Or, Minglun Gong, Hao Zhang, Guiqing Li, and Baoquan Chen. L_1-medial skeleton of point cloud. *ACM Transactions on Graphics*, 32(4):1–8, July 2013.

[HWQ+18] C. Huang, Z. Wang, D. Quinn, S. Suresh, and K. J. Hsia. Differential growth and shape formation in plant organs. *Proc. Natl. Acad. Sci. U.S.A.*, 115(49):12359–12364, Dec 2018.

[HWS16] Timo Hackel, Jan D. Wegner, and Konrad Schindler. Contour detection in unstructured 3d point clouds. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[JH99] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, May 1999.

[JMD74] K J. McCree and Stephen Davis. Effect of water stress and temperature on leaf size and on size and number of epidermal cells in grain sorghum1. *Crop Science - CROP SCI*, 14, 01 1974.

[Kuh17] C. Kuhlemeier. Phyllotaxis. *Curr. Biol.*, 27(17):R882–R887, Sep 2017.

[LBH15] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[LCS+19] Dawei Li, Yan Cao, Guoliang Shi, Xin Cai, Yang Chen, Sifan Wang, and Siyuan Yan. An overlapping-free leaf segmentation method for plant point clouds, 2019.

[LCT+18] Dawei Li, Yan Cao, Xue-song Tang, Siyuan Yan, and Xin Cai. Leaf Segmentation on Dense Plant Point Clouds with Facet Region Growing. *Sensors*, 18(11):3625, October 2018.

[LFC+18a] M. Li, M. H. Frank, V. Coneva, W. Mio, D. H. Chitwood, and C. N. Topp. The Persistent Homology Mathematical Framework Provides Enhanced Genotype-to-Phenotype Associations for Plant Morphology. *Plant Physiol.*, 177(4):1382–1395, Aug 2018.

[LFC+18b] Mao Li, Margaret H. Frank, Viktoriya Coneva, Washington Mio, Daniel H. Chitwood, and Christopher N. Topp. The Persistent Homology Mathematical Framework Provides Enhanced Genotype-to-Phenotype Associations for Plant Morphology. *Plant Physiology*, 177(4):1382–1395, August 2018.

[LFM+13] Yangyan Li, Xiaochen Fan, Niloy J. Mitra, Daniel Chamovitz, Daniel Cohen-Or, and Baoquan Chen. Analyzing growing plants from 4d point cloud data. *ACM Trans. Graph.*, 32(6):157:1–157:10, November 2013.

[MBdS+17] S. Madec, F. Baret, B. de Solan, S. Thomas, D. Dutartre, S. Jezequel, M. Hemmerle, G. Colombeau, and A. Comar. High-Throughput Phenotyping of Plant Height: Comparing Unmanned Aerial Vehicles and Ground LiDAR Estimates. *Front Plant Sci*, 8:2002, 2017.

[MBR16] Jyotirmaya Mathan, Juhi Bhattacharya, and Aashish Ranjan. Enhancing crop yield by optimizing plant developmental features. *Development*, 143(18):3283–3294, 2016.

[MFST16] Massimo Minervini, Andreas Fischbach, Hanno Scharr, and Sotirios A. Tsaftaris. Finely-grained annotated datasets for image-based plant phenotyping. *Pattern Recognition Letters*, 81:80 – 89, 2016.

[MGG+17] E. C. Morris, M. Griffiths, A. Golebiowska, S. Mairhofer, J. Burr-Hersey, T. Goh, D. von Wangenheim, B. Atkinson, C. J. Sturrock, J. P. Lynch, K. Vissenberg, K. Ritz, D. M. Wells, S. J. Mooney, and M. J. Bennett. Shaping 3D Root System Architecture. *Curr. Biol.*, 27(17):R919–R930, Sep 2017.

[MMK+04] Felix Morsdorf, Erich Meier, Benjamin Kötz, Klaus I Itten, Matthias Dobbertin, and Britta Allgöwer. Lidar-based geometric reconstruction of boreal type forest stands at single tree level for forest and wildland fire management. *Remote Sensing of Environment*, 92(3):353–362, 2004.

[MN03] Niloy J Mitra and An Nguyen. Estimating surface normals in noisy point cloud data. In *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, pages 322–328, 2003.

[MPB+10] Z. C. Marton, D. Pangercic, N. Blodow, J. Kleinehellefort, and M. Beetz. General 3d modelling of novel objects from a single view. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3700–3705, Oct 2010.

[MTM16] Ryan F. McCormick, Sandra K. Truong, and John E. Mullet. 3d sorghum reconstructions from depth images identify qtl regulating shoot architecture. *Plant Physiology*, 172(2):823–834, 2016.

[ND97] Timothy Nelson and Nancy Dengler. Leaf vascular pattern formation. *The Plant Cell*, 9(7):1121, 1997.

[NFL+16] C. V. Nguyen, J. Fripp, D. R. Lovell, R. Furbank, P. Kuffner, H. Daily, and X. Sirault. 3d scanning system for automatic high-resolution plant phenotyping. In *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8, Nov 2016.

[NSM+15] T. T. Nguyen, D. C. Slaughter, N. Max, J. N. Maloof, and N. Sinha. Structured Light-Based 3D Reconstruction System for Plants. *Sensors (Basel)*, 15(8):18587–18612, 2015.

[OI92] R. Ogniewicz and M. Ilg. Voronoi skeletons: Theory and applications. In *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 63–69, Champaign, IL, USA, 1992. IEEE Comput. Soc. Press.

[PASW13] Jeremie Papon, Alexey Abramov, Markus Schoeler, and Florentin Worgotter. Voxel cloud connectivity segmentation - supervoxels for point clouds. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[Pau19] Stefan Paulus. Measuring crops in 3d: using geometry for plant phenotyping. In *Plant Methods*, 2019.

[PBM+14] Stefan Paulus, Jan Behmann, Anne-Katrin Mahlein, Lutz Plümer, and Heiner Kuhlmann. Low-cost 3d systems: Suitable tools for plant phenotyping. *Sensors*, 14(2):3001–3018, 2014.

[PDMK13] Stefan Paulus, Jan Dupuis, Anne-Katrin Mahlein, and Heiner Kuhlmann. Surface feature based classification of plant organs from 3d laserscanned point clouds for plant phenotyping. *BMC Bioinformatics*, 14(1):238, Jul 2013.

[PL96] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag New York, Inc., New York, NY, USA, 1996.

[PL12] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The Algorithmic Beauty of Plants*. Springer Science & Business Media, 2012.

[PMG04] Mark Pauly, Niloy J. Mitra, and Leonidas J. Guibas. Uncertainty and variability in point cloud surface data. In *Proceedings of the First Eurographics Conference on Point-Based Graphics*, SPBG'04, page 77–84, Goslar, DEU, 2004. Eurographics Association.

[PNE17] Fernando Perez-Sanz, Pedro J Navarro, and Marcos Egea-Cortines. Plant phenomics: An overview of image acquisition technologies and image data analysis algorithms. *GigaScience*, 6(11), November 2017.

[PS19] Roland Pieruschka and Uli Schurr. Plant phenotyping: past, present, and future. *Plant Phenomics*, 2019, 2019.

[PSB+12] Anthony Paproki, Xavier Sirault, Scott Berry, Robert Furbank, and Jurgen Fripp. A novel mesh processing based technique for 3d plant analysis. *BMC Plant Biology*, 12(1):63, May 2012.

[PSNEC17] Fernando Perez-Sanz, Pedro J. Navarro, and Marcos Egea-Cortines. Plant phenomics: an overview of image acquisition technologies and image data analysis algorithms. In *GigaScience*, 2017.

[QYSG17] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5099–5108. Curran Associates, Inc., 2017.

[QZN14] Rongqi Qiu, Qian-Yi Zhou, and Ulrich Neumann. Pipe-run extraction and reconstruction from point clouds. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 17–30, Cham, 2014. Springer International Publishing.

[RBB09] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217, May 2009.

[RC11] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4, May 2011.

[RCG+15] Md. Matiur Rahaman, Dijun Chen, Zeeshan Gillani, Christian Klukas, and Ming Chen. Advanced phenotyping and phenotype data analysis for the study of plant growth and development. *Frontiers in Plant Science*, 6:619, 2015.

[RK02] D. Reinhardt and C. Kuhlemeier. Plant architecture. *EMBO Rep.*, 3(9):846–851, Sep 2002.

[RMBB08] Radu Rusu, Zoltan Marton, Nico Blodow, and Michael Beetz. Persistent point feature histograms for 3d point clouds. 16, 01 2008.

[Rus09] Radu Bogdan Rusu. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Technische Universität München, 2009.

[RvV06] T. Rabbani, F.A. van den Heuvel, and G. Vosselman. Segmentation of point clouds using smoothness constraints. In H.G. Maas and D. Schneider, editors, *ISPRS 2006 : Proceedings of the ISPRS commission V symposium Vol. 35, part 6 : image engineering and vision metrology, Dresden, Germany 25-27 September 2006*, volume 35, pages 248–253. International Society for Photogrammetry and Remote Sensing (ISPRS), 2006.

[SA17] Z. Shahzad and A. Amtmann. Food for thought: how nutrients regulate root system architecture. *Curr. Opin. Plant Biol.*, 39:80–87, 10 2017.

[Set12] Tim L Setter. Analysis of constituents for phenotyping drought tolerance in crop improvement. In *Front. Physio.*, 2012.

[SGDN14] R. Sievanen, C. Godin, T. M. DeJong, and E. Nikinmaa. Functional-structural plant models: a growing paradigm for plant studies. *Ann. Bot.*, 114(4):599–603, Sep 2014.

[SKBR15] Thiago Teixeira Santos, Luciano Vieira Koenigkan, Jayme Garcia Arnal Barbedo, and Gustavo Costa Rodrigues. 3d plant modeling: Localization, mapping and segmentation for plant phenotyping using a single hand-held camera. In Lourdes Agapito, Michael M. Bronstein, and Carsten Rother, editors, *Computer Vision - ECCV 2014 Workshops*, pages 247–263, Cham, 2015. Springer International Publishing.

[SLP17] Shangpeng Sun, Changying Li, and Andrew H. Paterson. In-field high-throughput phenotyping of cotton plant height using lidar. *Remote Sensing*, 9(4), 2017.

[SLP+18] Shangpeng Sun, Changying Li, Andrew H. Paterson, Yu Jiang, Rui Xu, Jon S. Robertson, John L. Snider, and Peng W. Chee. In-field High Throughput Phenotyping and Cotton Plant Growth Analysis Using LiDAR. *Frontiers in Plant Science*, 9:16, January 2018.

[SLSK07] Andrei Sharf, Thomas Lewiner, Ariel Shamir, and Leif Kobbelt. On-the-fly curve-skeleton computation for 3d shapes. In *Computer Graphics Forum*, volume 26, pages 323–328. Wiley Online Library, 2007.

[Sze10] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[TSDS10] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 356–369, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[TZC09] Andrea Tagliasacchi, Hao Zhang, and Daniel Cohen-Or. Curve Skeleton Extraction from Incomplete Point Cloud. In *ACM SIGGRAPH 2009 papers*, pages 1–9. 2009.

[US17] Jordan R. Ubbens and Ian K Stavness. Deep plant phenomics: A deep learning platform for complex plant phenotyping tasks. In *Front. Plant Sci.*, 2017.

[VEBS+10] J. Vos, J. B. Evers, G. H. Buck-Sorlin, B. Andrieu, M. Chelle, and P. H. de Visser. Functional-structural plant modelling: a new versatile tool in crop science. *J. Exp. Bot.*, 61(8):2101–2115, May 2010.

[VL00] Anne Verroust and Francis Lazarus. Extracting skeletal curves from 3d scattered data. *Vis. Comput.*, 16(1):15–25, February 2000.

[VMA86] BC Vemuri, A Mitiche, and JK Aggarwal. Curvature-based representation of objects from range data. *Image and Vision Computing*, 4(2):107 – 114, 1986.

[VQM16] Enrique Valencia, José L. Quero, and Fernando T. Maestre. Functional leaf and size traits determine the photosynthetic response of 10 dryland species to warming. *Journal of Plant Ecology*, 9(6):773–783, 01 2016.

[WPKM15] Mirwaes Wahabzada, Stefan Paulus, Kristian Kersting, and Anne-Katrin Mahlein. Automated interpretation of 3d laserscanned point clouds for plant organ segmentation. *BMC Bioinformatics*, 16(1):248, Aug 2015.

[WZC+16] P. Wilf, S. Zhang, S. Chikkerur, S. A. Little, S. L. Wing, and T. Serre. Computer vision cracks the leaf code. *Proc. Natl. Acad. Sci. U.S.A.*, 113(12):3305–3310, Mar 2016.

[XB20] Hao Xu and George W Bassel. Linking genes to shape in plants using morphometrics. *Annual Review of Genetics*, 54:417–437, 2020.

[XWCL15] Chunlei Xia, Longtan Wang, Bu-Keun Chung, and Jang-Myung Lee. In situ 3d segmentation of individual plant leaves using a rgb-d camera for agricultural automation. 15:20463–79, 08 2015.

[YDC+13] Wanneng Yang, Lingfeng Duan, Guoxing Chen, Lizhong Xiong, and Qian Liu. Plant phenomics and high-throughput phenotyping: accelerating rice functional genomics using multidisciplinary technologies. *Current Opinion in Plant Biology*, 16(2):180–187, 2013.

[ZCL+13] Jie Zhang, Junjie Cao, Xiuping Liu, Jun Wang, Jian Liu, and Xiquan Shi. Point cloud normal estimation via low-rank subspace clustering. *Computers & Graphics*, 37(6):697–706, 2013.

[ZIBE11] J. Zhu, P. A. Ingram, P. N. Benfey, and T. Elich. From lab to field, new approaches to phenotyping root system architecture. *Curr. Opin. Plant Biol.*, 14(3):310–317, Jun 2011.

[ZLD+14] Xiaopeng Zhang, Hongjun Li, Mingrui Dai, Wei Ma, and Long Quan. Data-driven synthetic modeling of trees. *IEEE Transactions on Visualization and Computer Graphics*, 20(9):1214–1226, 2014.

[ZN19] I. Ziamtsov and S. Navlakha. Machine Learning Approaches to Improve Three Basic Plant Phenotyping Tasks Using Three-Dimensional Point Clouds. *Plant Physiol.*, 181(4):1425–1440, Dec 2019.

[ZWR+15] Michel Zivy, Stefanie Wienkoop, Jenny Renaut, Carla Pinheiro, Estelle Goulas, and Sebastien Carpentier. The quest for tolerant varieties: the importance of integrating "omics" techniques to phenotyping. *Frontiers in Plant Science*, 6:448, 2015.

[ZZR18] Xiaolong Zhu, Meng Zhu, and Honge Ren. Method of plant leaf recognition based on improved deep convolutional neural network. *Cognitive Systems Research*, 52:223 – 233, 2018.