Modeling Dormant Fruit Trees for Agricultural Automation

Henry Medeiros
Department of Electrical and Computer Engineering
Marquette University
Milwaukee, WI 53201, USA
henry.medeiros@marquette.edu

Donghun Kim
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47906, USA
zava@purdue.edu

Jianxin Sun
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47906, USA
sun287@purdue.edu

Hariharan Seshadri
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47906, USA
hseshadr@purdue.edu

Shayan Ali Akbar
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47906, USA
sakbar@purdue.edu

Noha M. Elfiky
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47906, USA
nelfiky@purdue.edu

Johnny Park
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47906, USA
jpark@purdue.edu

Abstract

Dormant pruning of fruit trees is one of the most costly and labor-intensive activities in specialty crop production. We present a system which solves the first step in the process of automated pruning: accurately measuring and modeling the fruit trees. Our system employs a laser sensor to collect observations of fruit trees from multiple perspectives and uses these observations to measure parameters needed for pruning. A split-and-merge clustering algorithm

divides the collected data into three sets of points: trunk candidates, junction point candidates, and branches. The trunk candidates and junction point candidates are then further refined by a robust fitting algorithm that models as cylinders each segment of the trunk and primary branches. In this work, we focus on measuring the diameters of the primary branches and the trunk, which are important factors in dormant pruning and can be obtained directly from the cylindrical models. We show that the results are qualitatively satisfactory using synthetic and real data. Our experiments with three synthetic and three real apple trees of two different varieties showed that the system is able to identify the primary branches with an average accuracy of 98% and estimate their diameters with an average error of 0.6cm. Although the current implementation of the system is too slow for large scale practical applications (it can measure approximately two trees per hour), our study shows that the proposed approach may serve as a fundamental building block of robotic pruners in the near future.

1 Introduction

One of the most expensive and labor-intensive operations in perennial specialty crop production is dormant pruning. For fruit trees, dormant pruning accounts for approximately 20% of total labor costs—the second largest labor input after harvesting (College of Agricultural Science, Penn State Cooperative Extension, 2009). Traditionally, pruning is carried out during the winter while the plants are leafless (hence the name dormant pruning). Besides the physiological advantages, this timing allows for better visibility of the branches to be pruned. Without pruning, fruit quality and the efficacy of pest and disease control decline precipitously, which ultimately results in significantly reduced orchard profitability. To mitigate the need for a large, skilled seasonal workforce to complete the necessary and labor-intensive pruning operations, perennial specialty crop producers have begun to investigate more mechanized and automated pruning techniques. Such early attempts have entailed creating two-dimensional fruit tree architectures that lend themselves to mechanical hedging. This, however, is a nonselective pruning practice that may reduce fruit yield and quality (Ferree and Rhodus, 1993).

The Automation of Dormant Pruning of Specialty Crops Project is a groundbreaking undertaking funded by the USDA Specialty Crop Research Initiative (SCRI), which intends to develop a comprehensive and effective solution to the dormant pruning problem. The project involves researchers from five different institutions in academia, government, and industry and is investigating the problem from the perspectives of engineering, horticulture, and agricultural economy and sociology.[1] The ultimate goal of the project is to devise technologies for automating pruning operations. From an engineering point of view, the very first step to develop these technologies is to find a method that allows us to determine the locations of potential pruning points and to measure relevant morphological parameters of the tree to

[1] http://pruningautomation.com/

decide whether each pruning point candidate should indeed be pruned. In order to determine which parameters are most important in this process, we have worked in close collaboration with the horticulturists in the project and decided that the first such parameters should be the trunk diameter and the diameters of the primary branches, i.e., the branches connected directly to the trunk. The horticulturists are studying to what extent these simple parameters impact the entire pruning process, and their preliminary results seem to indicate that pruning based on diameters only may simplify the entire process without noticeable impact on the overall productivity of the fruit trees.

Based on the results obtained by the horticulturists, our goal in this work is to measure fruit trees in the field and estimate the locations and the diameters of the trunk and the primary branches. As we will see, the complexity of trees within a modern high density commercial orchard makes this a formidable robotic vision research problem. Hence, the objective of this work is to design a robust robotic vision system to generate three-dimensional models of apple trees in the field and estimate the aforementioned tree parameters which are relevant for pruning. The proposed data collection system uses time-of-flight laser scanning technology in order to be robust to the drastically unpredictable lighting conditions in the field and to the severe background cluttering caused by neighboring trees in an orchard. The proposed algorithms use robust segmentation and model fitting methods to accurately determine the parameters.

The rest of this paper is organized as follows. In the next section, we discuss the relevant literature. Section 3 shows the data acquisition system we developed to measure the trees, which is based on a laser scanner. In Section 4 we describe the algorithms we designed to measure the pruning parameters. These algorithms are divided into a trunk detection algorithm, which finds the locations of the primary branches, and a trunk and branch modeling algorithm, which fits cylinders to the trunk and branches and then allows us to measure their diameters. In Section 5 we show experimental results on synthetic and real data. Section 6 discusses the main characteristics and current limitations of the proposed system. Finally, Section 7 presents our concluding remarks and summarizes what we have learned in this project so far.

2 Related Work

Much of the literature on modeling trees based on real-world measurements focuses on generating photorealistic models for applications such as computer graphics or virtual reality (Tan et al., 2007; Teng and Chen, 2009; Sun et al., 2009; Long and Jones, 2012; Neubert et al., 2007). These works attempt to generate tree models that resemble their real counterparts, but give little consideration to the exact branching structure of the tree. In fact, in many cases, the branching structure is estimated based on the arrangement of the leafs. Such approaches are, therefore, not suitable for the purpose of pruning automation. While several of the photorealistic modeling methods utilize cameras to collect images of the trees, many of the works that attempt to generate models that can be used to accurately measure relevant parameters of the tree, such as the diameters of individual branches, rely on laser scanners

to collect the information (Livny et al., 2010; Schöler and Steinhage, 2012; Yan et al., 2009; Binney and Sukhatme, 2009; Xu et al., 2007; Pfeifer et al., 2004; Dassot et al., 2012; Côté et al., 2011; Rosell et al., 2009; Cheng et al., 2007; Watt and Donoghue, 2005; Delagrange and Rochon, 2011). That can be explained, to a large extent, by the overwhelming amount of background clutter found in images of trees collected in the field. Oftentimes, the branches from trees in the background are virtually indistinguishable from those of the tree of interest. Laser scanners alleviate this problem by collecting three-dimensional measurements of the trees, thereby allowing background clutter to be removed by simply defining a bounding region. Robustness to illumination variation is also a practical problem that must be carefully considered for automated pruning systems, as they must be able to operate uninterruptedly under a variety of conditions.

Although time-of-flight (van der Heijden et al., 2012) and stereo-vision systems (Bac et al., 2014a) have been used to estimate parameters of plants, these methods were applied to analyze pepper plants in greenhouses. In these scenarios, the rows of plants form a dense and relatively constant background and the range of distances that must be estimated is relatively small. Hence, the aforementioned problems with background clutter and illumination (particularly back-light) that afflict stereo-based systems can be mitigated with artificial illumination systems and heuristic rules based on the expected geometry of prominent foreground features. In addition, the dependency of accuracy and robustness of depth estimation on the distance to the target object, which are an issue for both stereo vision and time-of-flight cameras, are less significant. Laser scanners can easily resolve most of these problems, and hence are widely used in outdoor robotic applications (e.g., Lalonde et al., 2006). Furthermore, there is evidence that laser-based systems can provide more accurate measurements of tree parameters than systems based on video cameras (Dassot et al., 2012; Delagrange and Rochon, 2011; Weiss and Biber, 2011).

One approach to generate accurate representations of leafless trees is to employ a branching structure generative model to produce hypotheses as to where branching points might be located and use the measured data to confirm or reject these hypotheses. Schöler and Steinhage (2012) use Relational Growth Grammars (Kniemeyer et al., 2008) along with Reversible Jump Markov Chain Monte Carlo sampling to generate such hypotheses. Binney and Sukhatme (2009) as well as Friedman and Stamos (2013) use models based on L-Systems (Prusinkiewicz and Runions, 2012) and Preuksakarn et al. (2010) use space colonization algorithms (Palubicki et al., 2009) to accomplish the same goal. These methods, however, require the generation of a large number of hypotheses in order to appropriately model highly complex trees. In addition, in generative models in general, robustness to outliers, which are fairly common especially in high density orchards, depends on a carefully designed likelihood function that may require taking into consideration relationships among data points rather than considering each measurement individually.

An alternative to the use of generative models is to define edges connecting the 3D points and then find a minimum spanning tree (Livny et al., 2010) or a shortest path tree from the root (Xu et al., 2007; Cheng et al., 2007) to identify the overall structure of the tree. In (Livny et al., 2010), points belonging to a common branch are identified by connecting them with

edges weighted by their Euclidean distances and finding a minimum spanning tree on that graph. A global optimization procedure is then applied to construct a smooth graph that minimizes the edge orientation variation. Xu et al. (2007) initially find the skeleton of the main branching structure by computing the shortest paths from the root of the tree to each data point in the point cloud that is within a certain distance from its neighbors. Points are then clustered according to their distances to the root and the centroids of the clusters are connected to form the skeleton of the tree. The main problem with these approaches is that when the branching structure is intricate, with several thin branches overlapping, identifying the global structure without imposing local constraints on the relationship among points is very difficult. Hence, the results of minimum spanning tree or shortest path tree algorithms may be unpredictable.

Instead of searching for a tree structure directly on the point cloud, a more robust alternative is to first cluster nearby point clouds according to their local organization and then use these clusters to identify the overall structure of the tree. Cheng et al. (2007) segregates different branches according to jump discontinuities in the point cloud. The main limitation of this approach is that, due to floating pixel problems (Tuley et al., 2005; Sotoodeh, 2006; Tang et al., 2007), jump edge detection is somewhat challenging when there are many nearby branches at different depths. Pfeifer et al. (Pfeifer et al., 2004) first applies a set of morphological operations to the point clouds as proposed in (Gorte and Pfeifer, 2004) in order to generate a skeleton of the tree. In an approach similar to that proposed by Xu et al. (2007), they segment the skeleton according to the distance to the root. Trunks and branches are modeled by fitting the segmented data points to cylinders and then merging cylinders whose axes are well aligned. Although generally more robust than the methods that attempt to overlay a tree directly on the point cloud, none of the aforementioned methods accounts for the possibility of incorrect segmentation.

Most closely related to our approach is the work by Yan et al. (2009). In their method, neighboring points are clustered based on their Euclidean distances using k-means. For each cluster, a bounding cylinder is found and if the bound is not tight enough, the cluster is partitioned into two. The cylinders are used to construct a graph that represents the branches of the tree. To guarantee that no loops are formed, a minimum spanning tree based on the Euclidean distances between the centers of the clusters is computed. However, this method does not account for the possibility of imperfect segmentation either, nor does it consider the presence of outliers in the cylinder fitting process. The proposed approach addresses these limitations by employing a split-and-merge segmentation method in conjunction with robust fitting algorithms.

3 Data Acquisition System

In this work, we focus on modern high density orchards based on the tall spindle tree architecture, which is shown in Figure 1. The figure shows a picture of one of the orchards at the Penn State University (PSU) Fruit Research and Extension Center (FREC) in Bigglerville, PA. As the figure demonstrates, these trees are very thin and tall (3 to 4 meters high),

Figure 1: Modern high-density apple orchard.

Table 1: SICK LMS111 specifications.

Scan angles	270°
Resolution	0.25°
Scan rate	$50Hz$
Measurement range	$20m$
Communication	Ethernet
Environmental rating	IP67
Typical systematic error	±30mm

with approximately horizontal branches, and very densely spaced (approximately one meter apart). Because this architecture is relatively fragile, the trees are supported by a trellis structure. As the figure shows, there is significant overlap between branches from neighboring trees, and identifying the primary branches is relatively difficult even for a human observer.

Figure 2 shows a picture of our data acquisition system. The central element to the system is the SICK LMS111[2] laser measurement system (Figure 3). The LMS111 is a relatively inexpensive compact laser scanner designed for outdoors applications. Table 1 lists the specifications of the sensor. We mounted the laser scanner to an Aerotech ART310[3] rotary stage, which, by its turn, was mounted to an Aerotech ATS02060 linear slide. We used the rotary stage to rotate the laser sensor along its radial direction in order to build an entire range image at a single position. The linear slide allowed us to translate the sensor in order to collect multiple range images from slightly different viewpoints without having to move the entire data acquisition system.

[2]http://www.sick.com/
[3]http://www.aerotech.com/

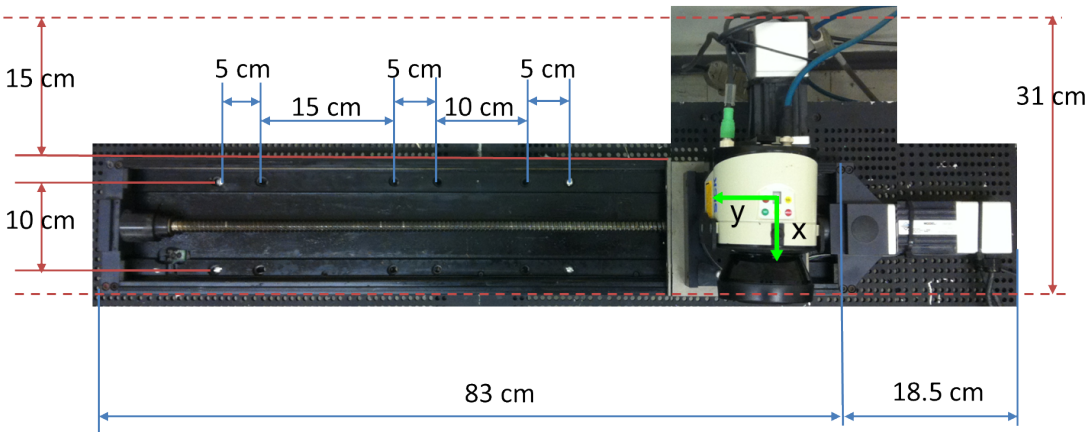


Figure 2: Data acquisition system. The green arrows represent the x and y directions of the coordinate system. The z axis is orthogonal to the xy plane and points toward the reader.

Figure 3: SICK LMS111 laser measurement system.

In order to collect data in the field, we mounted the data acquisition system to an N.BLOSI apple picking platform[4] and used a lead-acid battery connected to a power inverter to provide power to the system. In addition to allowing us to move the data acquisition system to the field and collect data from multiple viewpoints, the N.BLOSI platform is also capable of moving vertically, which allowed us to capture data at different heights. Figure 4 shows the complete data acquisition platform in operation in an orchard at the PSU FREC.

Consider a right-handed coordinate system with the y axis along the direction of the linear slide and the x axis along the axis of the laser sensor (i.e., perpendicular to the linear slide, as shown in Figure 2). We define a spherical coordinate system in which θ is the polar angle (i.e., on the xy plane), ϕ is the azimuth angle, and ρ is the radial distance. In order to obtain one range image of a tree, we collect one scan line at every 0.45° between $-45° \leq \theta < 45°$. Since each scan line corresponds to one point at every 0.25° between $-45° \leq \phi < 225°$ (see Table 1), our range image $\rho(\theta, \phi)$ has a resolution of 200×1080 points. Figure 5 shows one example of a single range image acquired by the system side-by-side with a photograph of the corresponding tree. The color map in the range image indicates the values of $\rho(\theta, \phi)$ for each

[4]http://www.nblosi.com/

Figure 4: Data acquisition platform.

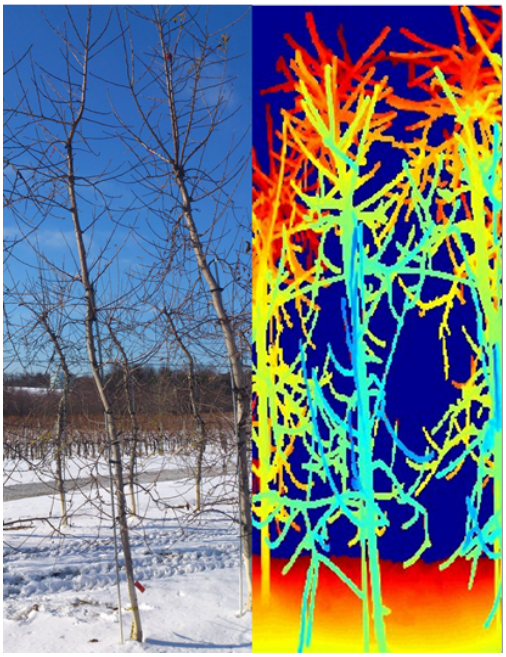


Figure 5: Sample range image collected by the sensing system. Best viewed in color.

value of θ and ϕ. Note that since the data is collected in spherical coordinates, the range image shown in Figure 5 presents some distortions with respect to the actual image. This problem is not present once the images are transformed into point clouds to be processed by our algorithms as we explain in detail below.

In our current implementation, we collect range images of each tree from 40 different view-

points. 20 images are collected on one side of the tree (which we arbitrarily call the front side to facilitate our explanation) and 20 additional images are acquired from the other side (i.e., the back). Out of the 20 images acquired at each side, 10 are measured with the data acquisition system at a height of approximately 1.5m and the other 10 at a height of approximately 4m. The first five images collected at each height are obtained by translating the linear slide 10cm between each image. The last five images are obtained by moving the N.BLOSI platform approximately 1m along the y axis and again moving the linear slide 10cm between each image.

3.1 Region of Interest Selection and Registration

Before the raw range images collected by the laser scanner can be processed by the measurement algorithms described in Section 4, we first need to convert them into three-dimensional models of the individuals trees observed by the system. This section describes this process.

The range image shown in Figure 5 was previously cropped to facilitate the comparison with the target tree. In reality, however, each range image contains a significant amount of irrelevant information, as shown in Figure 6. The figure shows that in addition to the target tree (within the red box), the image contains neighboring trees and objects behind the sensor (on the top half of the image). This happens because the sensor has a scanning angle of 270° and the tree of interest is fully contained in the region between $0° \leq \phi < 180°$. To remove the irrelevant information, we manually crop the area that corresponds to the target tree and then apply a bounding region to eliminate points that belong to trees within the cropped area that are behind the tree of interest. Figure 6 shows the results of these two steps.

After the tree of interest is isolated from each range image, we triangulate each view individually using the method described in (Park and N. DeSouza, 2005). During triangulation, we apply a simple floating pixel (Tuley et al., 2005; Sotoodeh, 2006; Tang et al., 2007) removal filter, which eliminates any point which has no immediate neighbors within a certain distance threshold ρ_T. That is, let $\rho_{i,j} = \rho(\theta_i, \phi_j)$ be the point under consideration and $\mathcal{N}_{i,j} = \{\rho_{k,l} | i-1 \leq k \leq i+1, j-1 \leq l \leq j+1\} - \rho_{i,j}$ be the immediate neighborhood of $\rho_{i,j}$, we remove any points that satisfy

$$\min_{\rho_{k,l} \in \mathcal{N}_{i,j}} (\rho_{i,j} - \rho_{k,l}) > \rho_T. \tag{1}$$

In addition, we apply a Laplacian surface smoothing to the generated mesh to reduce sensor measurement noise. Finally, we remove small isolated components. The output of the triangulation step is a set of 3D points $p_i = (x_i, y_i, z_i)$, where x_i, y_i, and z_i are the Cartesian coordinates of the points, and triangles $t_j = (p_{j1}, p_{j2}, p_{j3})$. From this point on, the triangles are used solely for visualization purposes and all the remaining algorithms operate only on the point cloud, i.e., the set of points $P = \{p_i\}_{i=1}^{N_p}$, where N_p is the total number of points. Figure 7 shows one sample view after triangulation. The surfaces and the corresponding shadows significantly facilitate the visualization of the results. Note that figure shows a

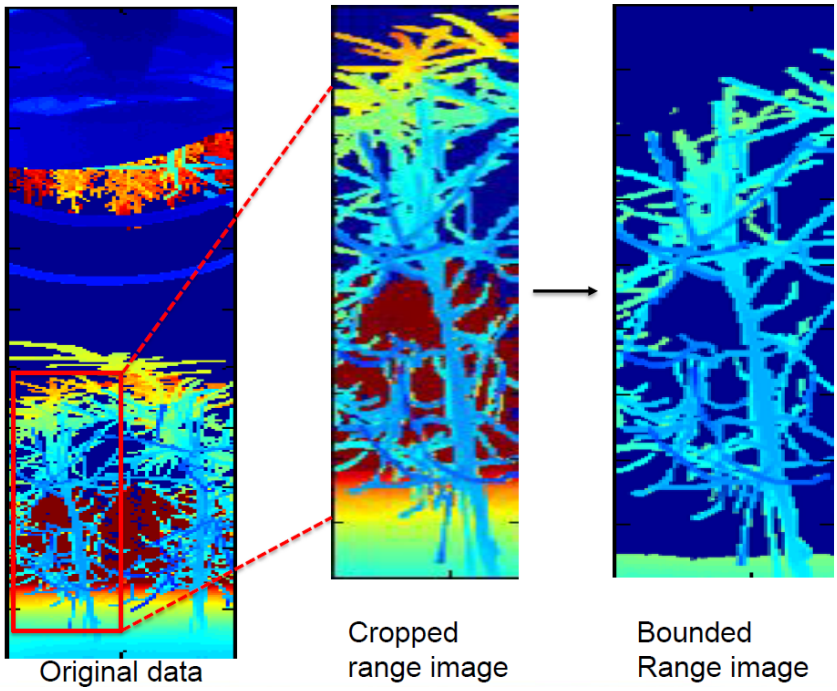


Figure 6: Manual segmentation of the region of interest. Best viewed in color.

Figure 7: Example of one view after triangulation.

number of gaps in the data, which are mostly caused by self-occlusions and hence justify the need to collect data from multiple viewpoints.

In order to generate a single consistent model, the point clouds from all 40 different view-

points need to be registered into a single model. That is, we need to compute

$$P^G = \bigcup_{k=1}^{N_v} T_k(P_k), \tag{2}$$

where N_v is the number of viewpoints (40 in the current implementation), P_k is the point cloud corresponding to the $k-th$ view, T_k is the transformation relating the $k-th$ point cloud to the global coordinate system (which in our case is simply the coordinate system of P_1) and P^G is the global point cloud, which combines all the views. Registration is performed using the well-known iterative closest point algorithm (ICP) (Besl and McKay, 1992).[5]

Algorithm 1 summarizes the main steps of the data acquisition procedure.

Algorithm 1 Data acquisition procedure.

1: Collect one scan line of the range image for $-45° \leq \phi < 225°$.
2: Rotate the laser sensor by $0.45°$ and collect the next scan line (step 1). Repeat for $-45° \leq \theta < 45°$.
3: Move the linear slide by $10cm$ and collect another range image (step 2). Repeat 5 times.
4: Move the platform to 2 positions at a height of 1.5m and 2 positions at 4m in the front of the tree. At each position, collect 5 range images (step 3). Move platform to the back of the tree and repeat the procedure.
5: Manually segment the relevant target from each range image.
6: Filter each range image according to Eq. 1. Triangulate the image to generate the point cloud $P_k = \{p_{ki}\}_{i=1}^{N_{kp}}$, where p_{ki} is the $i-th$ point in the $k-th$ viewpoint. Apply Laplacian smoothing and small component removal to P_k.
7: Register all the point clouds into a single global point cloud P^G according to Eq. 2.

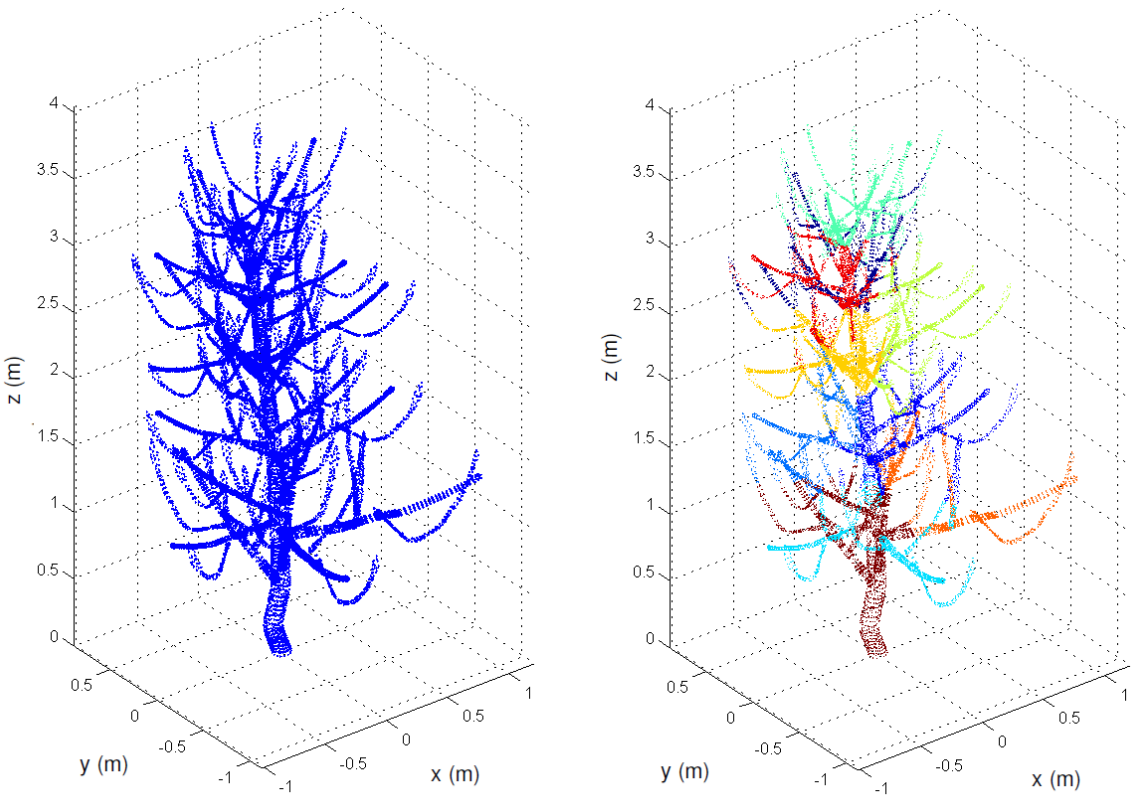
4 Pruning Parameters Measurement Algorithms

This section describes the algorithms we developed to process the data collected by the data acquisition system in order to locate the trunk and the primary branches and measure their respective diameters. As previously mentioned, these algorithms are divided into two main steps: the trunk detection algorithm, and the trunk and branch modeling algorithm.

4.1 Trunk Detection Algorithm

The objective of the trunk detection algorithm is to separate the point cloud into three groups: the points that belong to the trunk, the points that belong to the initial section of the primary branches, and the points that belong to the remaining branches. The first step in this procedure is to partition the point cloud into clusters of more manageable size. The goal of this step is to generate clusters that contain a few branch segments and possibly

[5]Registration was performed using Meshlab (http://meshlab.sourceforge.net/)



(a) (b)

Figure 8: Initial clustering of the point cloud. (a) Original synthetic data, (b) clustered points. Best viewed in color.

part of the trunk so that we can explore the structure of the data within each cluster and identify the relevant elements. We use the k-means algorithm to partition the point cloud into N_c initial clusters, that is, $P = \bigcup_{i=1}^{N_c} P_i$, where P_i are the initial clusters.[6] In our current implementation, we chose N_c so that each cluster consists of approximately $3,000$ points. Figure 8(a) shows one example of a synthetically generated[7] tree and Figure 8(b) shows the initial clusters obtained from this tree.

4.1.1 Split-and-merge Cluster Partitioning

After partitioning the original point cloud into smaller clusters, we employ a split-and-merge approach to further divide these clusters into approximately cylindrical segments. The split step goes over each of the clusters and breaks them down until all the clusters can be approximated by a cylinder. The merge step then goes over all the clusters merging the ones that were partitioned more than necessary.

[6]From now on, since we will be working only with global models, we drop the superscript G from our notation and represent P^G simply as P.

[7]The synthetic trees were generated using the xfrog software (http://www.xfrog.com)

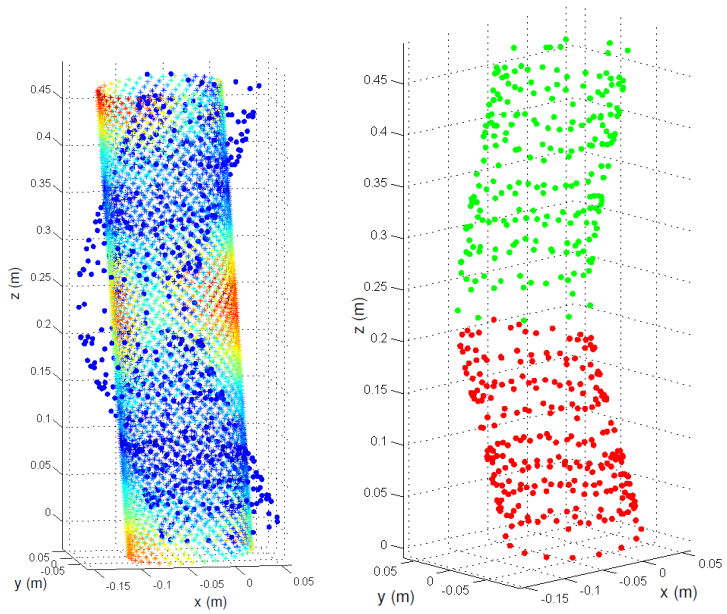
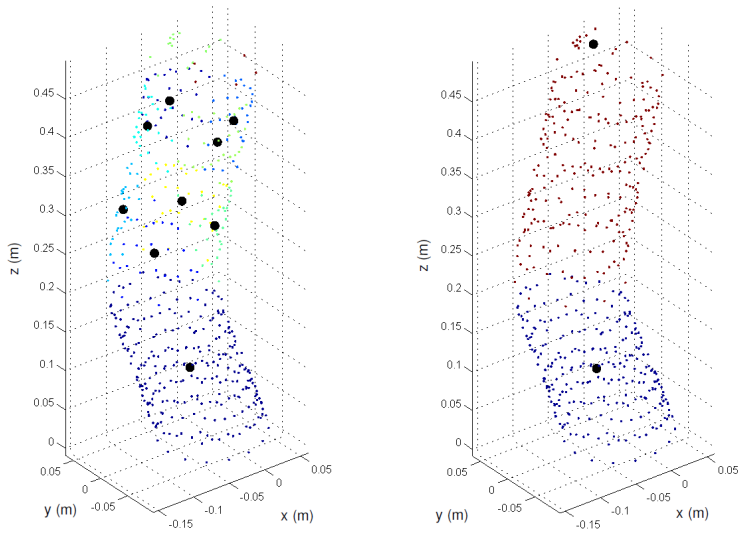


Figure 9: Cluster splitting process. (a) Original cluster consisting of two cylindrical segments. (b) Clusters after the split step. Best viewed in color.

In the split step, for each initial cluster P_i, we use a least squares approach to fit the points in the cluster to a cylinder $l_i = \{s, a, d, h\}$, where $s = [x, y, z]$ is the center of the axis of the cylinder, $a = [a_x, a_y, a_z]$ is a unit vector indicating the direction of the axis of the cylinder, d is the cylinder diameter, and h is its height. We then generate a new point cloud PC_i of cylindrical shape according to the parameters of l_i and compare this point cloud to P_i. Figure 9(a) shows a color-coded error map overlaid on PC_i for an example in which P_i consists of two cylindrical segments. If the total error between the points in PC_i and P_i is above a threshold e_t (in our implementation $e_t = 0.4\,\text{cm}$), we partition P_i into two new clusters using k-means and increment the total number of clusters N_c. Figure 9(b) shows the results of splitting the cluster in Figure 9(a). We repeat this process until no more clusters fail the cylinder fitting test.

Although the split step attempts to generate approximately cylindrical clusters, partitioning clusters using k-means may sometimes generate results such as the one shown in Figure 10(a). The figure shows that, although the cylinder on the bottom part of the figure was correctly segmented, the top part of the image was clustered into eight separate clusters covering different segments of the surface of the cylinder. Since in these scenarios, the resulting clusters correspond to points on different sides of the surface of the same cylinder, their centers tend to be in close proximity. Therefore, in the merge step, for each cluster P_i, we compute the distance between its center and the centers of its neighboring clusters P_k, where $k \in \mathcal{N}_i$ and $\mathcal{N}_i$ is the set of neighbors of P_i. Let c_i be the center of the cluster P_i, then if the distance between any c_k and c_i is smaller than a distance threshold d_t, i.e. $d(c_i, c_k) < d_t$, where $d(.)$ represents the Euclidean distance, then we replace P_i by a new



(a) (b)

Figure 10: Cluster merging process. (a) Original clusters in which the top cylindrical segment was split into several small clusters. (b) Clusters after the merge step. Best viewed in color.

cluster $P_i' = P_i \bigcup P_k$ and decrement the number of clusters N_c. We repeat this process until no more clusters can be merged (in our current implementation $d_t = 5\,\text{cm}$). Figure 10(b) shows the result of applying the merge step to 10(a).

Figure 11 shows the results of the split-and-merge cluster partitioning algorithm. The large black dots represent the cluster centers, and the points of different colors correspond to different clusters. As the figure shows, these clusters are approximately cylindrical, which is the desired result.

4.1.2 Trunk Localization

After the point cloud has been partitioned into approximately cylindrical clusters, we form an undirected graph $G(C, E)$, where $C = \{c_i\}_{i=1}^{N_c}$ is the set of cluster centers and $E = \{(c_i, c_k)\}$ is the set of edges between connected clusters, which is determined as follows. Consider the points corresponding to a branch segment which was partitioned into a pair of clusters shown in Figure 12. To facilitate our discussion, let us call the cluster on the left side of the figure (in red points) P_i and the cluster on the right side (in black) P_k. Our first step is to find the point p_i in P_i that is closest to any point in P_k (the green point in Figure 12). Let $\mathcal{N}_{p_i}^N$ be the set of N closest neighbors of p_i, which is represented by the shaded circle in the figure. If p_i is indeed a genuine point in the first cluster which sits in the interface with the second cluster (rather than a noisy observation or some artifact of the previous processing steps), then approximately half of its neighbors should belong to its own cluster and the other half should belong to the neighboring cluster. That is, let $\mathcal{N}_{p_i,i}^N = P_i \cap \mathcal{N}_{p_i}^N$ be the set of nearest neighbors of p_i that belong to P_i and $\mathcal{N}_{p_i,k}^N = P_k \cap \mathcal{N}_{p_i}^N$ be the set of neighbors that belong to P_k, we expect $n_{i,k} = \left|\mathcal{N}_{p_i,k}^N\right| / \left|\mathcal{N}_{p_i}^N\right| \approx \left|\mathcal{N}_{p_i,i}^N\right| / \left|\mathcal{N}_{p_i}^N\right| \approx 0.5$. Therefore, in order to connect

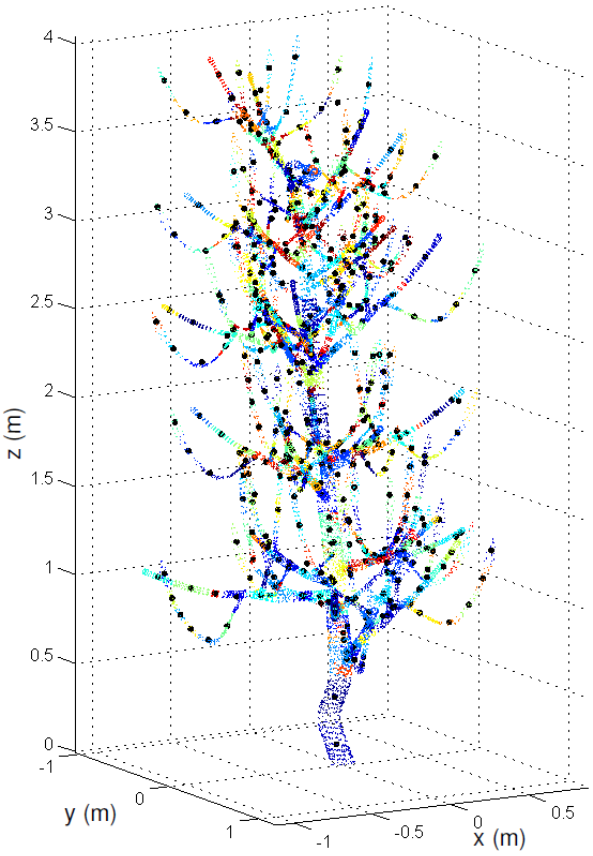


Figure 11: Results of the split-and-merge clustering algorithm. Best viewed in color.

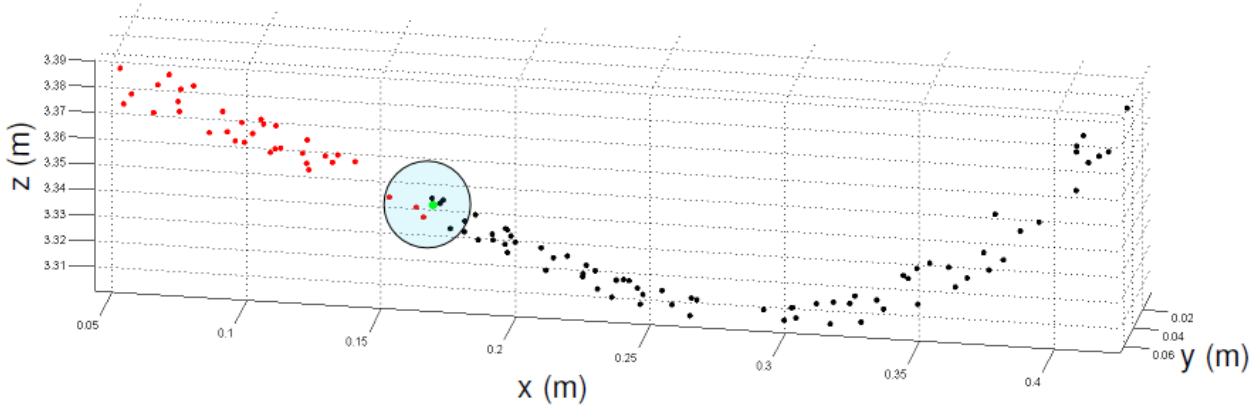


Figure 12: Cluster connection procedure. Best viewed in color.

the two clusters we check whether $n_L \leq n_{i,k} \leq n_H$, where n_L is a low percentage threshold and n_H is a high percentage threshold (in our implementation $n_L = 0.3$ and $n_H = 0.7$). For increased robustness, we also verify that $n_{k,i}$ satisfies the same constraint (which amounts to checking whether the edge is bidirectional).

Once we have the graph of connected clusters $G(C, E)$, we need to identify the subgraph $G_T(C_T, E_T)$ that corresponds to the centers of the clusters that belong to the trunk. Since we know that the trunk of the tree begins close to the ground and follows an approximately vertical trajectory, we identify G_T according to the following heuristics. Assume that initially $C_T = \emptyset$ and $E_T = \emptyset$. Starting from the cluster center c_i with the smallest z coordinate, we compute the angle between the unit vector $u = [\ 0 \quad 0 \quad 1\ ]^T$ and each of the edges in E that include c_i. That is, let $\mathcal{S}_i = \{(a, b)\,|a = c_i \,\text{and}\, (a, b) \in E\}$ be the set of edges in E that include c_i as one of its vertices. Then, we find the vertex

$$e^* = \underset{e_k \in \mathcal{S}_i}{\operatorname{argmin}} \left[\psi(e_k)\right], \tag{3}$$

where $\psi(e_k) = \cos^{-1}\left(\frac{e_k^T u}{\|e_k\|\|u\|}\right)$, and e^* corresponds to the vertex that shows the smallest angle with respect to the vertical direction. We include $e^* = (c_i, c_k)$ into E_T and both c_i and c_k to C_T. We restart the procedure from c_k and terminate it when no transition is possible in which $\psi(e_k)$ is smaller than a threshold ψ_T (in our implementation $\psi_T = 72^o$). This approach guarantees that we do not continue our search along branches at the top part of the tree. The green edges in Figure 13(a) show the result of our trunk detection algorithm. Figure 13(b) shows the clusters corresponding to the trunk in blue and the primary branch clusters in green. The primary branch clusters are given by the clusters whose centers are directly connected to the centers of the clusters in the trunk but are not in C_T. From now on, whenever we use the expression primary branches we will be referring to these clusters.

Algorithm 2 summarizes the trunk detection procedure.

Algorithm 2 Trunk detection.

1: Partition the point cloud into N_c clusters.
2: For each cluster, try to fit a cylinder to its points and partition the cluster into two if the fit is poor. Repeat until no more clusters can be partitioned.
3: For each cluster, merge those whose centers are closer than a distance threshold d_T. Repeat until no more clusters can be merged.
4: Create the graph G of connected clusters centers.
5: Create the graph G_T of cluster centers that belong to the trunk by following the clusters that satisfy Eq. 3 starting from the lowest cluster.

4.2 Modeling Algorithm

As previously mentioned, the objective of this work is to locate the trunk and the primary branches of a given tree and then measure their diameters. The algorithm described in Section 4.1 provides us the approximate locations of the trunk and primary branches, and hence solves the first part of our problem. In this section, we describe the algorithm that takes as input the clusters corresponding to the trunk and primary branches and then models them as cylinders so that we can measure their diameters.

Let us first consider the process of modeling the trunk as a set of cylinders of potentially

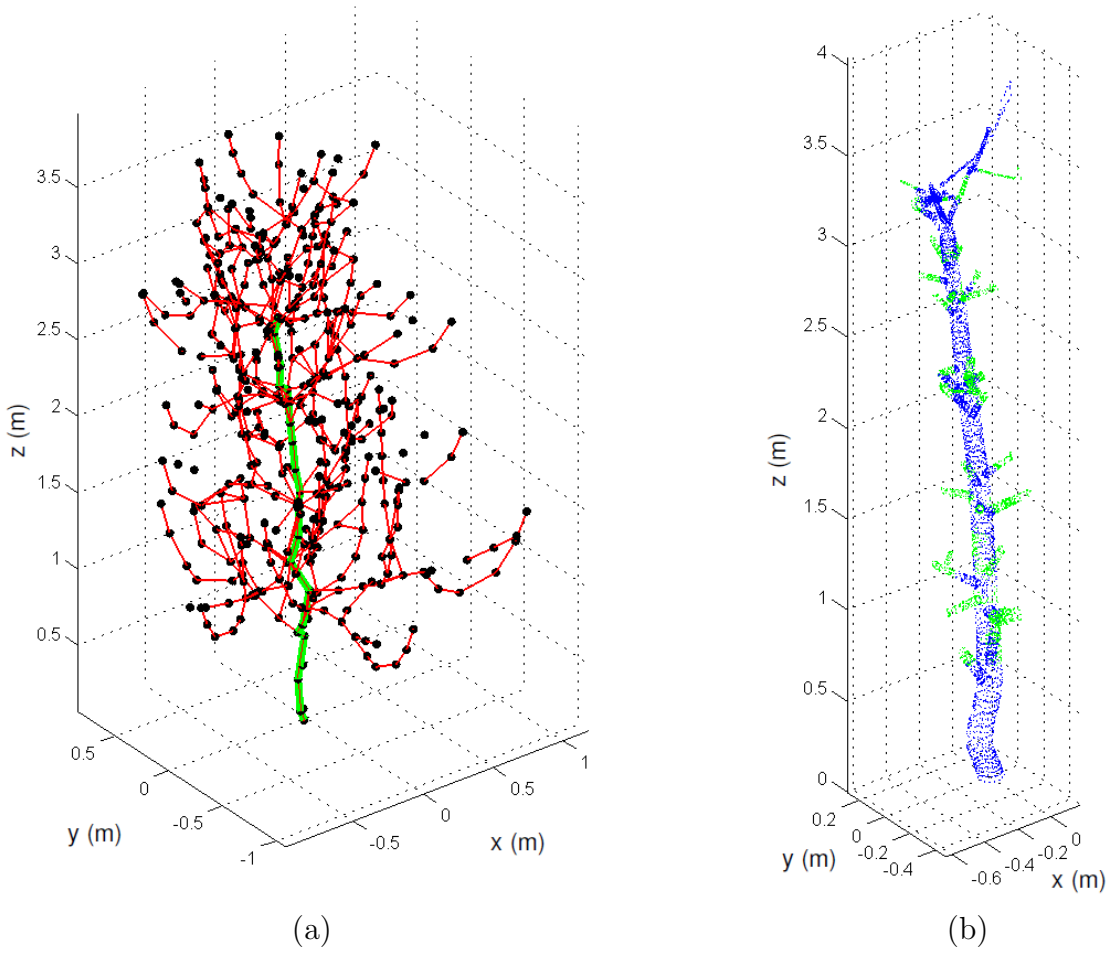
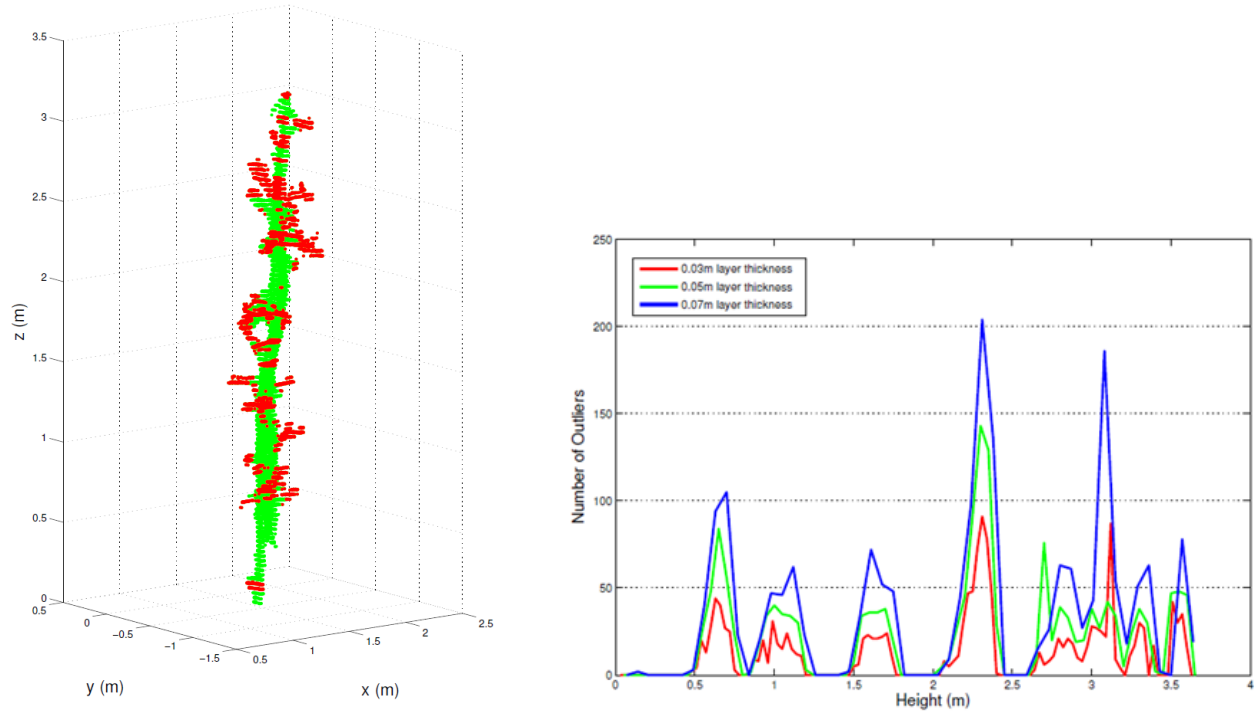


Figure 13: Results of the trunk detection algorithm. (a) Cluster centers and their respective neighbors. (b) Trunk points (blue) and primary branch points (green). Best viewed in color.

different diameters so that diameter variations along the trunk can be taken into consideration. Clearly, any relatively straight portion of the trunk that does not include branches can be modeled as a cylinder with reasonable accuracy. Therefore, we must partition the trunk so that every segment contains at least some portion that does not include any branches – we will consider the case in which a branchless segment includes a curve momentarily. This is illustrated in Figure 14(a), which shows the portion of a point cloud corresponding to the trunk and primary branches as determined by the algorithm in Section 4.1. In the figure, branchless trunk regions are shown in green and the regions that are connected to branches are shown in red. We will now describe how we can partition the trunk into segments that contain more points in branchless regions (i.e., green points in the figure) than branch points (red). Once this is done, a robust fitting approach such as the RANSAC algorithm (Fischler and Bolles, 1981) can be used to model each segment as a cylinder.

In order to identify regions along the trunk that include branches, we first divide the trunk



(a) (b)

Figure 14: Segmentation of the trunk. (a) Regions where branches connect to the trunk (red points) generate the disturbances shown in (b). Best viewed in color.

into thin slices

$$S_n = \{(x, y, z) \,|(x, y, z) \in P_T \cup P_P \text{ and } z_n \leq z < z_{n+1}\}, \tag{4}$$

where P_T and P_P are the clusters that belong to the trunk and primary branches, $z_n = n\Delta_Z$, $0 \leq n < \left\lceil \frac{Z}{\Delta_Z} \right\rceil$ is the number of slices, Z is the height of the trunk, and Δ_Z is the height of each slice (3 to 7cm in our experiments). We then project the points in each slice S_n onto the plane orthogonal to the direction of the trunk at the height of the slice under consideration (i.e., z_n). We try to fit a circle to the projected points using RANSAC and we keep track of the number of outliers for each slice. These outliers are used to generate the function $\eta(z_n)$, which consists of the number of outliers along the direction of the trunk. Figure 14(b) shows a plot of $\eta(z_n)$ for different slice heights. As the figure shows, regardless of the height of the slices, the results are very consistent and the areas where branches are connected to the trunk show a substantial number of outliers. We call these areas disturbance regions.

One method to ensure that each trunk segment contains at least one cylindrical portion is to partition the trunk at the beginning of each disturbance region. The top graph in Figure 15 shows the curve $\eta(z_n)$ for $\Delta z = 3$cm. The second graph in the figure shows the derivative of $\eta(z_n)$. As the graph indicates, at the beginning of the disturbance regions the derivative of $\eta(z_n)$ is positive. The points along the trunk where the derivative of $\eta(z_n)$ are positive are shown in the third plot in the figure. As the plot indicates, within a single disturbance

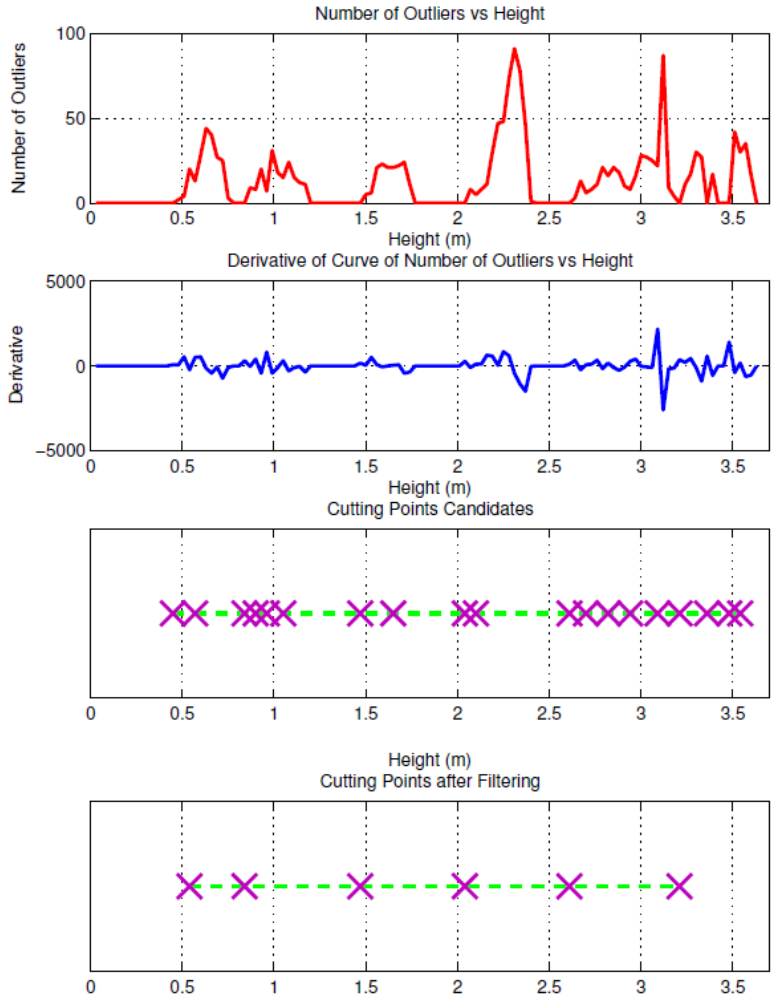


Figure 15: Partitioning point candidates. From top to bottom: number of outliers vs. tree height curve, derivative of the outliers curve, points where the derivative is positive, first point in each disturbance region.

region, there are several positive derivatives. Hence, we need to make sure we split the trunk only at the first point in each disturbance region, i.e., points where a positive derivative is preceded by a segment with zero derivative. These are the points shown at the bottom plot of Figure 15, and they correspond to the points where we partition the trunk.

The limitation of using disturbance regions to locate the partitioning points is that this approach cannot identify curvatures in the trunk. Suppose, for the sake of illustration, that we need to model the trunk segment shown on the left side of Figure 16. Clearly, this segment cannot be represented using a single cylinder and must be partitioned at the point of curvature. As illustrated on the top right side of Figure 16, since we are using a robust method to fit the cylinder to the points, in this specific case only points below the curve are

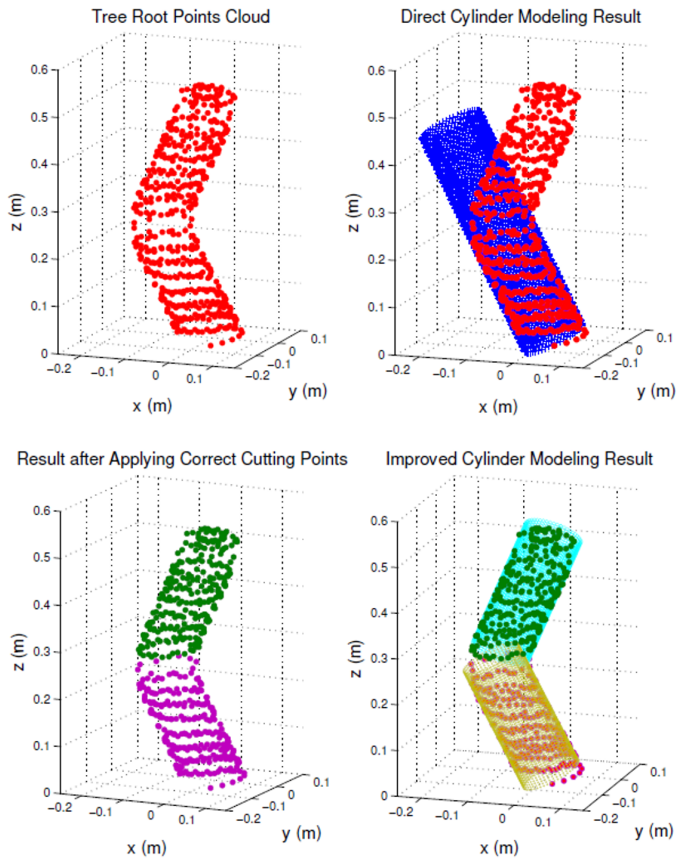


Figure 16: Result of fitting a cylinder to a high curvature point. Best viewed in color.

properly modeled and the remaining points are considered outliers. The point of curvature can then be immediately identified by looking at the outlier curve along the axis of the fitted cylinder, which is shown in Figure 17. Whenever there is a high curvature point in a segment, the number of outliers increases drastically at that point, as the figure illustrates. We can then add additional partitioning points accordingly and obtain the results shown at the bottom of Figure 16. Figure 18 shows the final partitioning points for the trunk shown in Figure 14(a) after accounting for both disturbance regions and curvature points. For each of the M partitioning points, we create a set A_m, where $0 \leq m < M-1$, which includes the points in $P_T \cup P_P$ whose z coordinate lies between a pair of partitioning points. We then fit a cylinder to each A_m using RANSAC. These cylinders correspond to the trunk model.

Once we have all our trunk segments properly modeled, we obtain the models of the primary branches according to the following procedure. The leftmost graph in Figure 19 shows one of the trunk segments A_m which includes seven primary branch segments. When we fit a cylinder to the points in A_m using RANSAC, the points corresponding to the primary branches are considered outliers. The center graph in Figure 19 shows the same segment when only the outliers are retained. Clearly, the primary branches consist of several reasonably separated small clusters of points. We apply an Euclidean clustering algorithm (Rusu, 2009) to these points in order to separate them into distinct sets, each of which corresponds to a

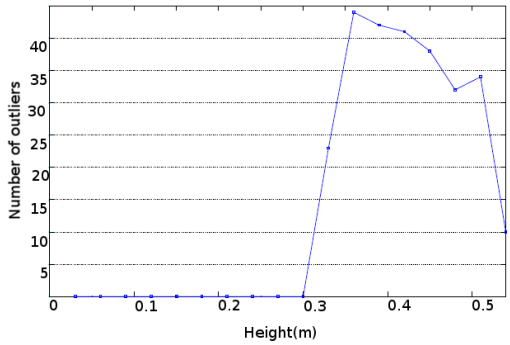


Figure 17: Outlier distribution along high curvature region.

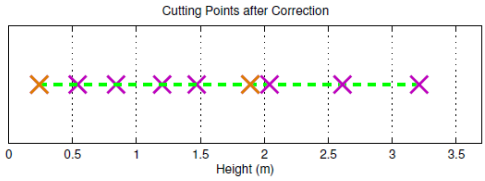


Figure 18: Final partitioning points for the trunk shown in Figure 14(a). Purple crosses represent disturbance regions and golden crosses correspond to high curvature points. Best viewed in color.

different primary branch.[8] The left side of Figure 19 shows each of the sets in a different color. As we did for each A_m during trunk modeling, we simply fit a cylinder to each primary branch using RANSAC. Figures 20(a) and (b) show the modeled cylinders for one trunk segment in synthetic and real data, respectively.

Algorithm 3 summarizes the trunk and branch modeling procedure.

Algorithm 3 Trunk and branch modeling.

1: Locate the disturbance regions and regions of high curvature to define the positions of the partitioning points.
2: Partition the trunk into M segments A_m, each of which has at least some cylindrical portion.
3: Fit a cylinder to each A_m. These are the cylinders that represent the trunk.
4: For each A_m, separate the points that do not belong to the cylinder (i.e., outliers). These are the points that correspond to the primary branch segments.
5: Fit a cylinder to each set of points corresponding to the branches. These cylinders represent the primary branch segments.
6: Use the cylinder models to compute the diameter of the trunk and primary branches.

[8] All the algorithms in this section were implemented using the Point Cloud Library (http://www.pointclouds.org)

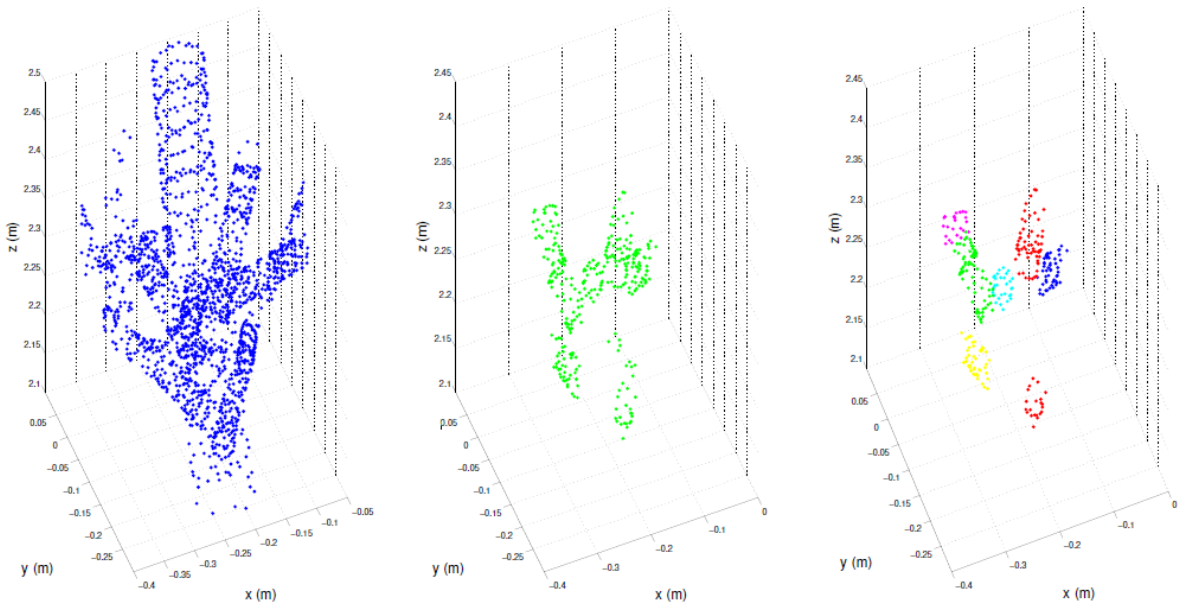


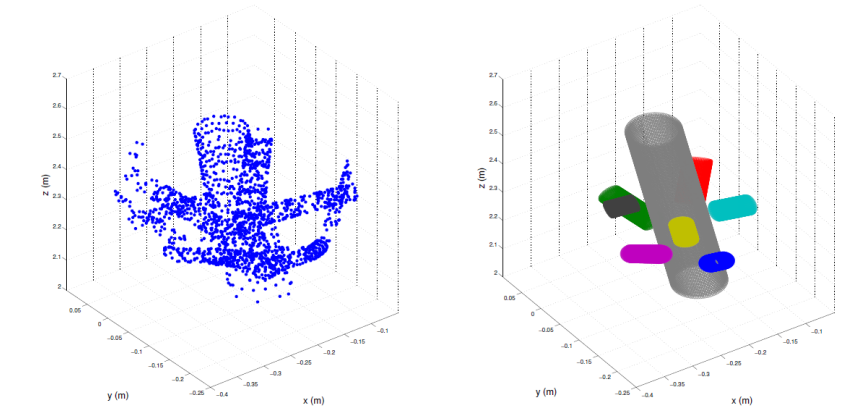
Figure 19: Clustering of branch points in a trunk segment. Best viewed in color.

5 Experimental Evaluation

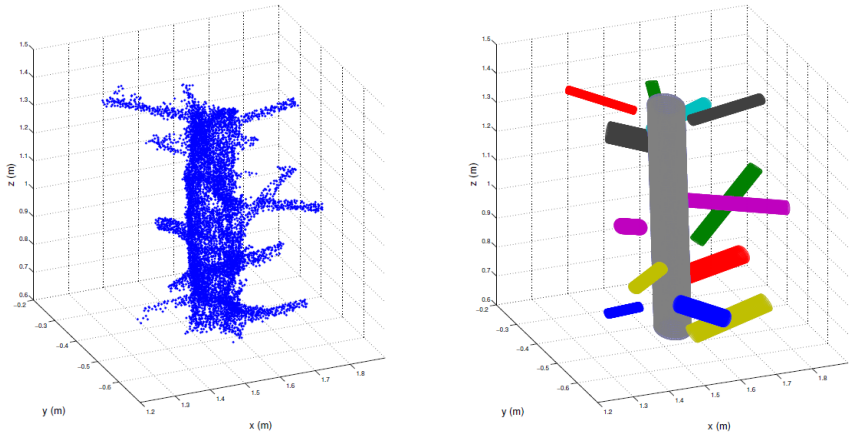
In this section we show experimental results that qualitatively illustrate the performance of our data acquisition system and of our algorithms to localize the trunk and primary branches and measure their diameters. The experiments were performed on both synthetically generated data as well as on data from real trees acquired by the system in a laboratory environment and in the field.

5.1 Data Acquisition

Figure 21 shows a model generated by our data acquisition system. To facilitate the experimentation, this tree was removed from the Purdue orchard and scanned inside the laboratory. Additional experiments with trees in an orchard setting are presented in Section 5.3. We used a standard measuring tape to manually measure the actual diameters of the trunk and primary branches at the points marked I to VII in Figure 21 and then computed the absolute error between the actual measurements and the corresponding diameters in the model as measured by Meshlab. The accuracy of the ground truth measurements, which includes the accuracy of the tape as well as human error, is approximately $\pm$1mm. Table 2 shows the measurement results and the corresponding error.



(a)



(b)

Figure 20: Results of branch fitting in one trunk segment of (a) synthetic data and (b) real data. Best viewed in color.

Table 2: Measurement errors in the model shown in Figure 21 (GT = Ground Truth).

Point	**Meas.**	**GT**	**Error (cm)**
I	22.10	22.30	0.20
II	10.41	10.57	0.16
III	17.65	18.62	0.97
IV	8.41	8.94	0.53
V	10.41	10.69	0.28
VI	15.62	16.56	0.94
VII	14.35	15.49	1.14
Mean			**0.60**

5.2 Trunk Detection

Figure 22 shows the results of applying our trunk detection algorithm to the tree shown in Figure 21. From left to right, the figure shows the clusters, the connected cluster centers

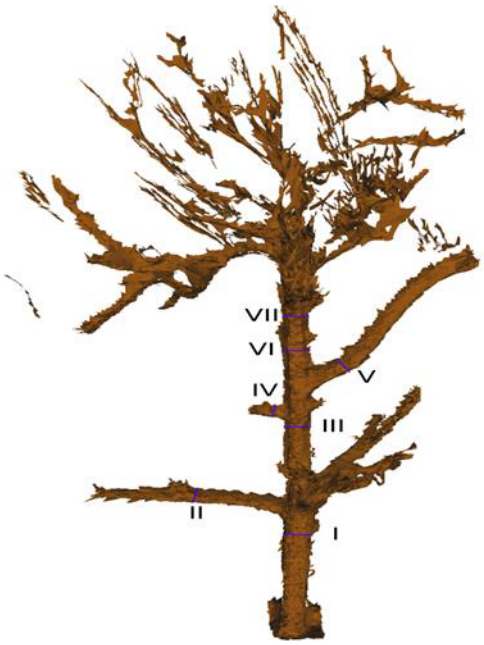


Figure 21: Results of the data acquisition system.

with the edges corresponding to the trunk highlighted, and the final detection results. In the rightmost figure, the brown region corresponds to the trunk points, the green regions are the primary branch clusters, and the blue regions are the remaining parts of the primary branches as well as the other branches. We can see that the algorithm localizes the trunk fairly well. The small portions of the primary branches that are detected as part of the trunk are properly removed in the next step, when we model them as cylinders. As for the top part of the tree, as is generally the case for most trees, even for a human observer it is difficult to define what corresponds to the trunk in that region. Our algorithm simply selected the most horizontal branches.

5.3 Modeling

Figure 23 shows the final results of modeling the entire trunk and the primary branch segments on synthetic data, and Figure 24 shows the results on real data (the results on the real trees shown in this section are limited to tall spindle apple trees). On both figures, the leftmost column shows a rendering of the tree model, the center column shows the detected trunk and primary branches and the rightmost column shows the cylindrical models generated to represent the segments of the trunk and primary branches. As we can see, the trunk cylinders provide a fairly accurate representation of the trunk and virtually all the primary branch segments were also properly modeled in these examples. Table 3 summarizes the number of primary branches detected by our algorithm in comparison with the ground truth.

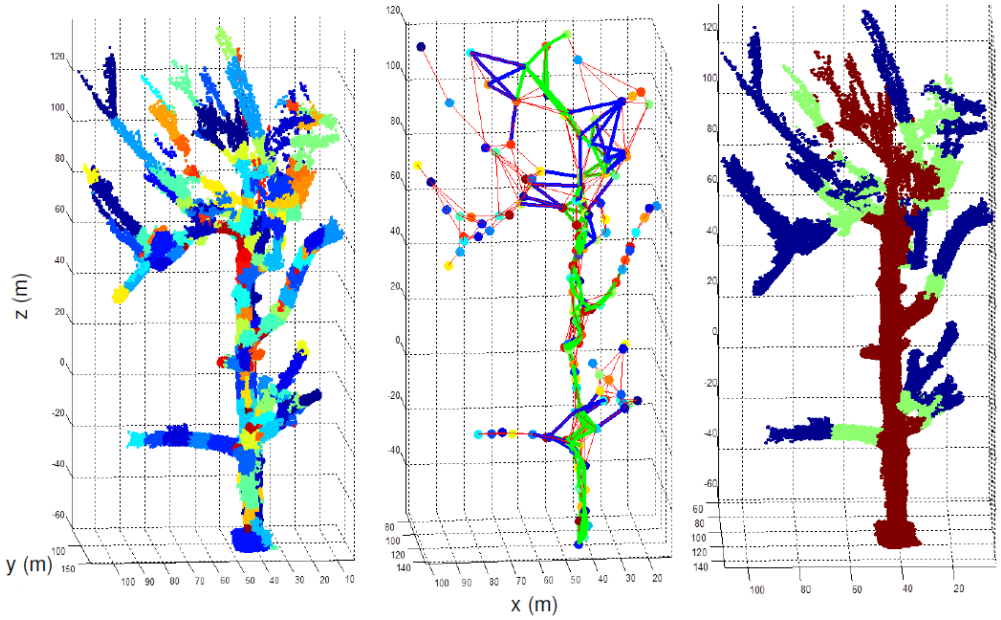


Figure 22: Results of the trunk detection algorithm on the tree model shown in Figure 21. (left) Results of the clustering algorithm in which each cluster is represented by a different color. (center) Results of the trunk finding algorithm wherein the centers of the clusters that belong to the trunk are connected by a green line. (left) Final classification into trunk (brown), primary branches (green), and remaining (blue) points. The origin of the coordinate system is at the center of the laser scanner. Best viewed in color.

Table 3: Summary of the primary branch segment classification results shown in Figures 23 and 24 (PB = Primary Branches, IPB = Identified Primary Branches).

Tree	**PB**	**IPB**	**False Positive Rate (%)**	**False Negative Rate (%)**	**F1 Score Rate (%)**
Synthetic Tree (a)	24	24	0.00	0.00	100.00
Synthetic Tree (b)	13	13	0.00	0.00	100.00
Synthetic Tree (c)	8	9	12.50	0.00	94.11
Real Tree (d)	25	26	4.00	0.00	98.04
Real Tree (e)	36	35	0.00	2.78	98.59
Average			3.30	0.56	98.15

6 Discussion

One of the greatest limitations to the practical applicability of the current system is the necessity of manual intervention in parts of the registration process. The ICP algorithm, which is used to register multiple views, needs a reasonably accurate initial estimate of the transformation between a pair of images in order to perform accurate registration. If the algorithm is initialized with a poor initial guess, it may stop at a local minimum and produce grossly inaccurate results. However, except for the sets of images collected by translating

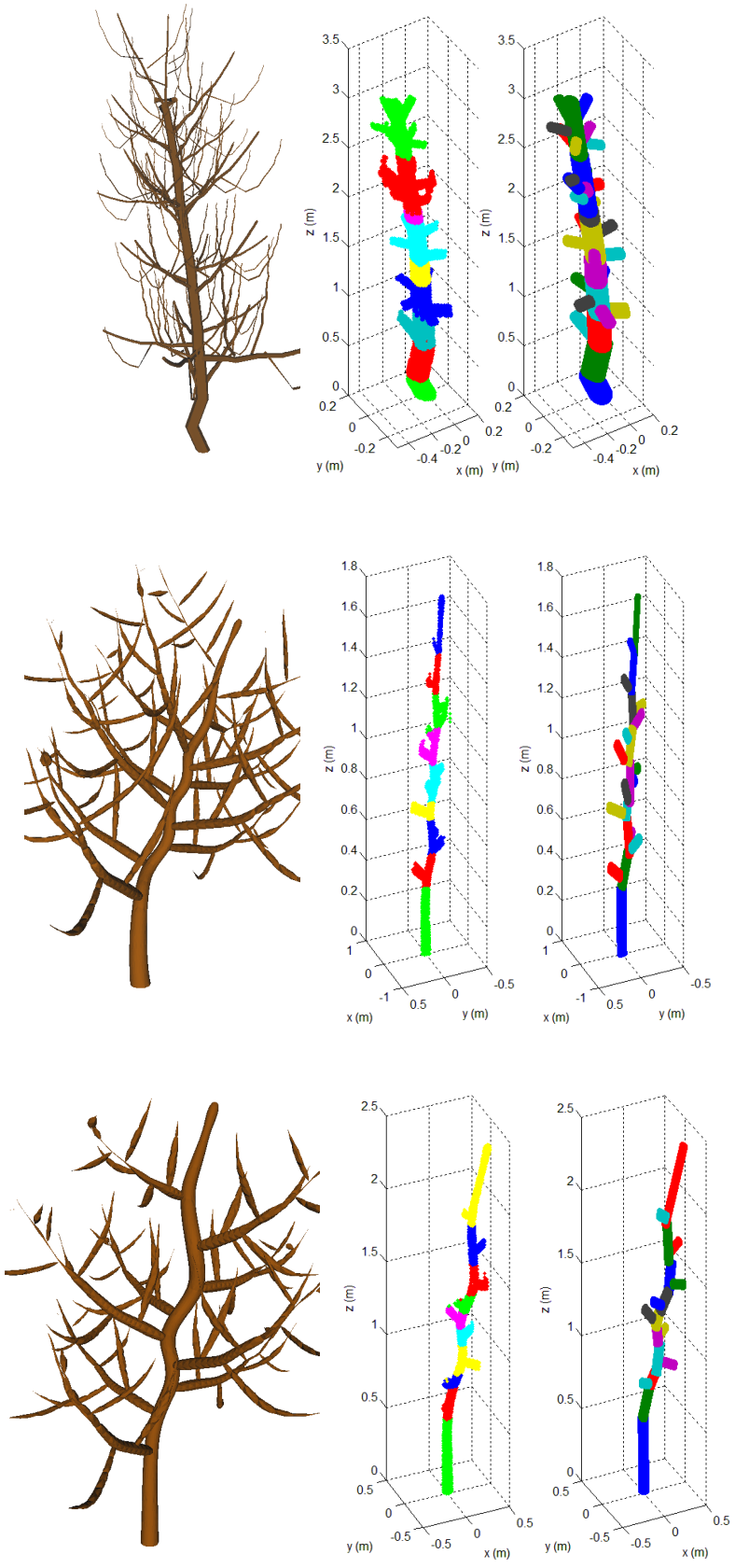


Figure 23: Results of the trunk and branch modeling algorithm on synthetic data. Synthetically generated trees are shown on the leftmost column. The center column shows the points corresponding to the trunk and primary branch segments. The rightmost column shows the cylinders corresponding to the trunk and branch segments. Best viewed in color.

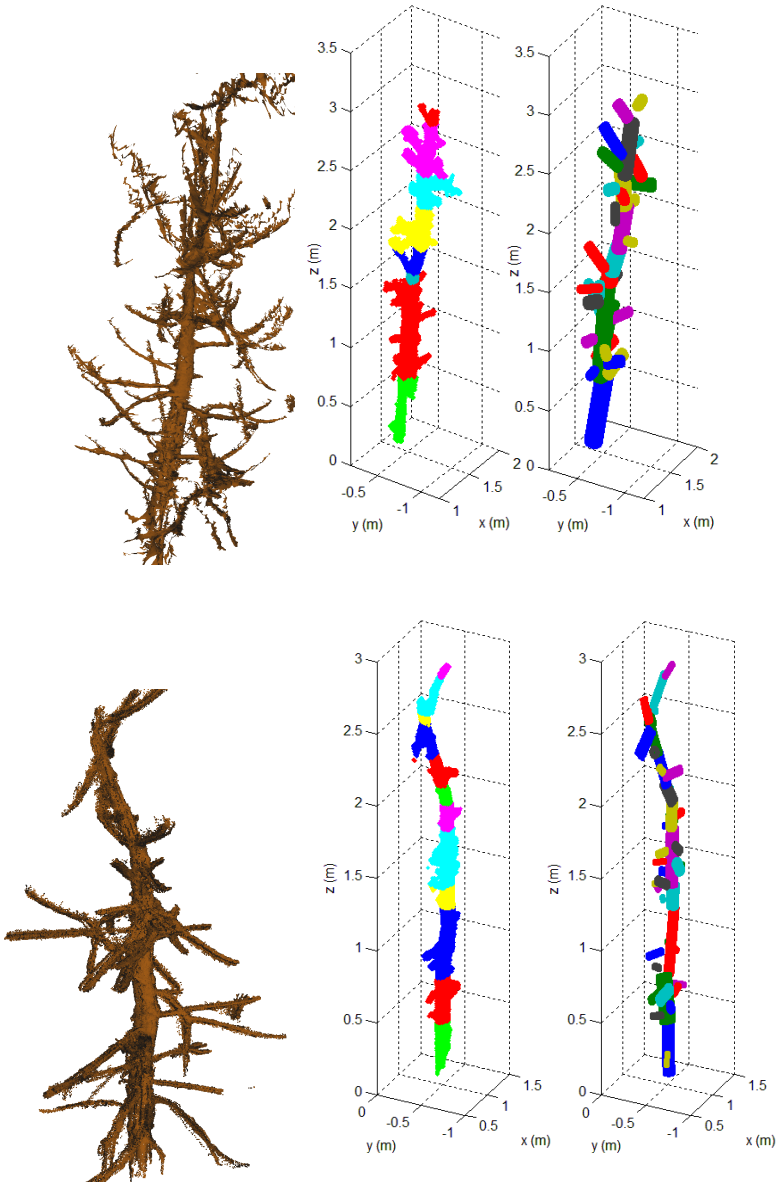


Figure 24: Results of the trunk and branch modeling algorithm on real data. Data from real trees collected in the field is shown on the leftmost column. The center column shows the points corresponding to the trunk and primary branch segments. The rightmost column shows the cylinders corresponding to the trunk and branch segments. Best viewed in color.

the linear slide, obtaining a good initial estimate of the transformation between pairs of images is not straightforward. When the platform is moved from one point to another, only a very rough estimate of the transformation can be obtained. This is particularly true for the transformation between images acquired on the front side of the trees and the ones acquired on the back side. Currently, registration is performed automatically between pairs of images acquired by moving the linear slide, and human intervention is necessary to perform registration of the other views in order to provide an appropriate initial guess to the

system. We are currently investigating more robust frameworks to perform fully automated registration (Elfiky et al., 2015; Chattopadhyay et al., 2016). Data segmentation is also a somewhat tedious task that currently must be done manually. Automating that process should be relatively straightforward, however, based on the assumption that the roots of neighboring trees are relatively well separated.

Another practical limitation of the current system is the data acquisition time. It currently needs approximately 30 minutes to scan one tree from all 40 viewpoints. Improvements in the implementation of the system should allow a significant reduction in data collection time. The average time to collect each view is currently 45s. However, the laser scanner could collect that information in 4s. Hence, there is significant room for improvement. Platform repositioning is also a major factor in data acquisition time. It takes approximately 10 minutes to position the platform in all the eight positions required by the system. The utilization of multiple lower cost laser scanners such as the SICK TiM5xx series would allow the repositioning time to be reduced in half. Furthermore, although the 40 different viewpoints currently used to model each tree seem to provide enough information to handle the significant self-occlusions present in the trees and allow us to generate sufficiently accurate models, an investigation to determine the optimal number of viewpoints is subject of future work. The determination of the minimum number of viewpoints needed to generate accurate models would allow us to avoid spending time obtaining unnecessary information. Finally, we should keep in mind that as a row of trees is scanned in an orchard, multiple neighboring trees tend to be visualized by the system (see Figure 7, for example). Hence, the acquisition time for each tree is amortized as the overlapping information is used for multiple trees.

As described in previous sections, the proposed algorithms rely on a number of parameters. Although the parameter values presented in this work proved general enough to handle two different types of real trees (from the Purdue and the PSU orchards) as well as a number of synthetic trees with different characteristics, we recognize that these parameters might need to be adjusted for different datasets. Hence, we also intend to automate the selection of several such parameters, particularly for the split-and-merge algorithm. For example, the number of initial clusters N_c could potentially be defined based on the sensor resolution and the approximate average size of the target trees. The error threshold e_t and the split threshold d_t of the split and merge steps, respectively, could be determined according to the expected ratio between the length and the diameter of the model cylinders for a particular scenario. The angular thresholds ψ_T that determines the end of the search for trunk segments could also be adjusted based on the expected curvature of the trunk at the top of the tree. In addition, in order to further increase the robustness of the system, we might impose additional constraints on our search such as a limit on the radius change between two consecutive trunk segments. Trunk detection at the top part of the tree, for example, could be improved if, rather than simply selecting the most horizontal branches, we used the diameter of the trunk cylinders to avoid incorrectly fitting smaller vertical branches at the top of the tree.

In addition, as the results shown in the previous section demonstrate, the proposed system is able to accurately model the trunk and primary branches of apple trees in an orchard

setting. However, given the substantial amount of clutter present in the orchard, sometimes the cylinder fitting mechanisms fail because of the presence of an overwhelming number of outliers, which causes the entire algorithm to collapse. We are working on increasing the robustness of our algorithms by restricting the search space used by RANSAC to determine the best fit. For example, when fitting a trunk segment, there is no reason to search for cylinders whose axes are not approximately vertical or whose diameters are drastically different from those of previously modeled segments.

Furthermore, although the proposed system based on a laser scanner showed promising results, it is important to keep in mind that alternative technologies such as time-of-flight and stereo cameras continue to make progress. Because these technologies have the ability to quickly scan two-dimensional regions, as technological improvements resolve current issues with illumination robustness and spatial resolution, we should continue to consider them as viable options to generate models of trees as well as other complex and large objects in the field.

Finally, despite the satisfactory results obtained in our qualitative evaluation, as Bac et al. (2014b) pointed out, a rigorous and comprehensive quantitative evaluation is necessary in order to fully understand the potential of the system and allow for its comparison with alternative approaches. Ideally, such evaluation should be carried out in conjunction with horticulturists who specialize in apple tree pruning and can help design experimental procedures which determine the variables that must be accounted for and how to properly measure them. Such evaluation is one of the ultimate goals of the Automation of Dormant Pruning of Specialty Crops Project, and we intend to perform it in the near future.

7 Conclusions

In this work, we presented a system that collects measurements of fruit trees in the field and uses this information to estimate parameters relevant for automated pruning, such as the locations and diameters of the trunk and primary branches. Our system consists of a laser scanner attached to a high precision positioning system mounted to an agricultural mobile platform, which are used to generate three-dimensional models of the fruit trees from multiple perspectives and integrate them into a global model. Our algorithms partition the models generated by the system into small clusters that can then be modeled as cylinders from which the parameters of interest can be immediately computed. We evaluated our system using synthetic and real data in the laboratory and in the field.

Dormant fruit trees are extremely complex structures, especially in a modern high density orchard setting where there is substantial overlap among neighboring trees. Furthermore, despite being notoriously robust to illumination changes and to background clutter, the measurements collected by laser scanners are significantly noisy and cannot be processed by naive modeling algorithms. Hence, we proposed a method that integrates robust fitting algorithms with simple but reasonable heuristics based on the expected physical characteristics of the trees to create accurate models of the trunk and the primary branches of the trees. These

models allowed us to successfully measure the parameters of interest and could be employed to measure additional parameters such as the angles between the primary branches and the trunk, if necessary.

The fast pace at which new vision sensing technologies are being developed and their robustness in specific scenarios may lead to the impression that it would be a simple matter to apply these devices to agricultural robotic system. However, as the seemingly disproportionately slow advance of agricultural robotic technologies shows, the complexity of real agricultural applications requires additional levels of sophistication. To the best of our knowledge, this is the first system that can measure the parameters of dormant fruit trees in the field. While more extensive field evaluations of the proposed system will provide additional insights into its capabilities, it corresponds nonetheless to the first step in the design of an automated fruit tree pruner based on robotic vision technologies. Although the design and implementation of such autonomous pruner for fruit trees is still years in the future, it would not be possible without a foundational robotic vision system as the one presented in this work.

Acknowledgments

This project was supported by the USDA Specialty Crop Research Initiative (SCRI). We would like to thank Edwin Winzeler and Jim Schupp for their assistance with the data collection and field experiments at PSU. We would also like to thank Peter Hirst for assisting us with the field experiments at Purdue.

References

Bac, C., Hemming, J., and van Henten, E. (2014a). Stem localization of sweet-pepper plants using the support wire as a visual cue. *Computers and Electronics in Agriculture*, 105:111 – 120.

Bac, C. W., van Henten, E. J., Hemming, J., and Edan, Y. (2014b). Harvesting robots for high-value crops: State-of-the-art review and challenges ahead. *Journal of Field Robotics*, 31(6):888–911.

Besl, P. and McKay, N. D. (1992). A method for registration of 3-D shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239–256.

Binney, J. and Sukhatme, G. (2009). 3D tree reconstruction from laser range data. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 1321–1326.

Chattopadhyay, S., Akbar, S. A., Elfiky, N. M., Medeiros, H., and Kak, A. (2016). 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.

Cheng, Z.-L., Zhang, X.-P., and Chen, B.-Q. (2007). Simple reconstruction of tree branches from a single range image. *Journal of Computer Science and Technology*, 22(6):846–858.

College of Agricultural Science, Penn State Cooperative Extension (2009). *2008-2009 Pennsylvania Tree Fruit Production Guide*. Penn State University.

Côté, J.-F., Fournier, R. A., and Egli, R. (2011). An architectural model of trees to estimate forest structural attributes using terrestrial lidar. *Environmental Modelling & Software*, 26(6):761 – 777.

Dassot, M., Colin, A., Santenoise, P., Fournier, M., and Constant, T. (2012). Terrestrial laser scanning for measuring the solid wood volume, including branches, of adult standing trees in the forest environment. *Computers and Electronics in Agriculture*, 89(0):86 – 93.

Delagrange, S. and Rochon, P. (2011). Reconstruction and analysis of a deciduous sapling using digital photographs or terrestrial-lidar technology. *Annals of botany*, 108(6):991–1000.

Elfiky, N. M., Akbar, S. A., Sun, J., Park, J., and Kak, A. (2015). Automation of dormant pruning in specialty crop production: An adaptive framework for automatic reconstruction and modeling of apple trees. In *Computer Vision and Pattern Recognition Workshops*.

Ferree, D. C. and Rhodus, W. T. (1993). Apple tree performance with mechanical hedging or root pruning in intensive orchards. *Journal of the American Society for Horticultural Science*, 118(6):707–713.

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

Friedman, S. and Stamos, I. (2013). Automatic procedural modeling of tree structures in point clouds using wavelets. In *3D Vision - 3DV 2013, 2013 International Conference on*, pages 215–222.

Gorte, B. and Pfeifer, N. (2004). Structuring laser-scanned trees using 3D mathematical morphology. In *International Archives of Photogrammetry and Remote Sensing, Vol. XXXV*, Instanbul, Turkey.

Kniemeyer, O., Barczik, G., Hemmerling, R., and Kurth, W. (2008). Relational growth grammars - a parallel graph transformation approach with applications in biology and architecture. In Schürr, A., Nagl, M., and Zündorf, A., editors, *Applications of Graph Transformations with Industrial Relevance*, volume 5088 of *Lecture Notes in Computer Science*, pages 152–167. Springer Berlin Heidelberg.

Lalonde, J.-F., Vandapel, N., Huber, D. F., and Hebert, M. (2006). Natural terrain classification using three-dimensional ladar data for ground robot mobility. *Journal of Field Robotics*, 23(10):839–861.

Livny, Y., Yan, F., Olson, M., Chen, B., Zhang, H., and El-Sana, J. (2010). Automatic reconstruction of tree skeletal structures from point clouds. *ACM Trans. Graph.*, 29(6):151:1–151:8.

Long, J. and Jones, M. (2012). 3D tree modeling using motion capture. In *Plant Growth Modeling, Simulation, Visualization and Applications (PMA), 2012 IEEE Fourth International Symposium on*, pages 242–249.

Neubert, B., Franken, T., and Deussen, O. (2007). Approximate image-based tree-modeling using particle flows. *ACM Trans. Graph.*, 26(3):Article No. 88.

Palubicki, W., Horel, K., Longay, S., Runions, A., Lane, B., Měch, R., and Prusinkiewicz, P. (2009). Self-organizing tree models for image synthesis. *ACM Trans. Graph.*, 28(3):58:1–58:10.

Park, J. and N. DeSouza, G. (2005). 3-D modeling of real-world objects using range and intensity images. In Apolloni, B., Ghosh, A., Alpaslan, F., C. Jain, L., and Patnaik, S., editors, *Machine Learning and Robot Perception*, volume 7 of *Studies in Computational Intelligence*, pages 203–264. Springer Berlin Heidelberg.

Pfeifer, N., Gorte, B., and Winterhalder, D. (2004). Automatic reconstruction of single trees from terrestrial laser scanner data. In *Proceedings of the 20th ISPRS Congress*, pages 114–119.

Preuksakarn, C., Boudon, F., Ferraro, P., Durand, J.-B., Nikinmaa, E., and Godin, C. (2010). Reconstructing plant architecture from 3D laser scanner data. In *6th International Workshop on Functional-Structural Plant Models*.

Prusinkiewicz, P. and Runions, A. (2012). Computational models of plant development and form. *New Phytologist*, 193(3):549–569.

Rosell, J. R., Llorens, J., Sanz, R., Arnó, J., Ribes-Dasi, M., Escolà, A., Masip, J., Camp, F., Solanelles, F., Gràcia, F., Gil, E., Val, L., Planas, S., and Palacín, J. (2009). Obtaining the three-dimensional structure of tree orchards from remote 2D terrestrial LIDAR scanning. *Agricultural and Forest Meteorology*, 149(9):1505 – 1515.

Rusu, R. B. (2009). *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Computer Science department, Technische Universitaet Muenchen, Germany.

Schöler, F. and Steinhage, V. (2012). Towards an automated 3D reconstruction of plant architecture. In Schürr, A., Varró, D., and Varró, G., editors, *Applications of Graph Transformations with Industrial Relevance*, volume 7233 of *Lecture Notes in Computer Science*, pages 51–64. Springer Berlin Heidelberg.

Sotoodeh, S. (2006). Outlier detection in laser scanner point clouds. In *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences XXXVI-5*, pages 297–302.

Sun, R., Jia, J., Li, H., and Jaeger, M. (2009). Image-based lightweight tree modeling. In *Proceedings of the 8th International Conference on Virtual Reality Continuum and Its Applications in Industry*, VRCAI '09, pages 17–22, New York, NY, USA. ACM.

Tan, P., Zeng, G., Wang, J., Kang, S. B., and Quan, L. (2007). Image-based tree modeling. *ACM Trans. Graph.*, 26(3):Article No. 87.

Tang, P., Huber, D., and Akinci, B. (2007). A comparative analysis of depth-discontinuity and mixed-pixel detection algorithms. In *Proceedings of the International Conference on 3-D Digital Imaging and Modeling (3DIM)*, pages 29–38.

Teng, C.-H. and Chen, Y.-S. (2009). Image-based tree modeling from a few images with very narrow viewing range. *The Visual Computer*, 25(4):297–307.

Tuley, J., Vandapel, N., and Hebert, M. (2005). Analysis and removal of artifacts in 3-D LADAR data. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2203–2210.

van der Heijden, G., Song, Y., Horgan, G., Polder, G., Dieleman, A., Bink, M., Palloix, A., van Eeuwijk, F., and Glasbey, C. (2012). SPICY: towards automated phenotyping of large pepper plants in the greenhouse. *Functional Plant Biology*, 39:870–877.

Watt, P. J. and Donoghue, D. N. M. (2005). Measuring forest structure with terrestrial laser scanning. *International Journal of Remote Sensing*, 26(7):1437–1446.

Weiss, U. and Biber, P. (2011). Plant detection and mapping for agricultural robots using a 3D LIDAR sensor. *Robotics and Autonomous Systems*, 59(5):265 – 273. Special Issue ECMR 2009.

Xu, H., Gossett, N., and Chen, B. (2007). Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Trans. Graph.*, 26(4):Article No. 19.

Yan, D.-M., Wintz, J., Mourrain, B., Wang, W., Boudon, F., and Godin, C. (2009). Efficient and robust reconstruction of botanical branching structure from laser scanned points. In *Computer-Aided Design and Computer Graphics, 2009. CAD/Graphics '09. 11th IEEE International Conference on*, pages 572–575.