Skeletonization of Plant Point Cloud Data in Stochastic Optimization Framework

Ayan Chaudhury and Christophe Godin

***Abstract*—Skeleton extraction from 3D plant point cloud data is an essential prior for myriads of phenotyping studies. Although skeleton extraction from 3D shapes have been studied extensively in the computer vision and graphics literature, handling the case of plants is still an open problem. Drawbacks of the existing approaches include the *zigzag* structure of the skeleton, nonuniform density of skeleton points, lack of points in the areas having complex geometry structure, and most importantly the lack of biological relevance. With the aim to "correct" an existing coarse skeleton structure, we propose a stochastic framework which is supported by the biological structure of the original plant. Initially we estimate the branching structure of the plant by the notion of β-splines to form a *curve tree* defined as a finite set of curves joined in a tree topology with certain level of smoothness. In the next phase, we *force* the discrete points in the curve tree to *move* towards the original point cloud by treating each point in the cloud as a center of Gaussian, and points in the curve tree as observations from the Gaussians. The task is to find the correct Gaussian centroid for each point. The optimization technique is iterative and is based on the Expectation Maximization (EM) algorithm. The E-step estimates which Gaussian the observed point cloud was sampled from, and the M-step maximizes the negative log-likelihood that the observed points were sampled from the Gaussian Mixture Model (GMM) with respect to the model parameters. We experiment with several real world and synthetic datasets and demonstrate the robustness of the approach over the state-of-the-art.**

***Index Terms*—Skeletonization, Point Cloud, Curve Tree, Spline, Stochastic Optimization, Gaussian Mixture Model, Expectation Maximization.**

I. Introduction

AUTOMATED analysis of phenotyping traits of plants using imaging techniques are becoming off-the-shelf tool for botanical analyses these days ([1], [2]). Although high throughput 2D imaging based phenotyping systems have shown promising results on the accuracy and precision for many cases ([3], [4], [5]), but these systems suffer from the inherent limitations of 2D image based analysis techniques. In recent years, 3D point cloud based analysis is getting extremely popular in phenotyping and agricultural applications [6]. Typical applications of point cloud based phenotyping include plant organ segmentation [7], robotic branch pruning [8], automated growth analysis [9], etc. Many of these applications require skeleton structure of the input point cloud data as a *prior* for further processing. Also, as a skeleton is a compact representation of the original object, dealing with skeleton requires less computational overhead than dealing with the original point cloud data. That is why skeletons are widely used for varieties of shape analysis tasks in different application areas [10].

In general, a skeleton is a thin structure obtained from an object, which encodes the topology and basic geometry of the object. Ideally, the skeleton should follow the exact *centerline* of the object. That means within a local neighbourhood, the distance from each skeleton point to the enclosing shape boundaries should be the same. Although skeleton extraction from 2D images is a well studied area, 3D point cloud skeletonization is still an open problem [11]. For the case of plants, skeleton extraction is extremely challenging due to their complex geometry, thin structure, missing data due to self-occlusion, etc. State-of-the-art algorithms ([12], [13], [14]) which are proposed as general skeletonization techniques for regular 3D objects, often fail to perform for the case of plants. Skeletonization of 3D plant point cloud data is a little worked problem, and only a handful of techniques have been proposed in recent years. These techniques can be broadly classified into two categories: i) initial skeleton construction, and ii) skeleton refinement. The first type of algorithm refers to skeleton construction from the input point cloud data. A notable work in this category is the method proposed by Xu *et al.* [15]. Initially, a *Riemannian graph* is constructed from the point cloud by connecting the points using nearest neighbour strategy. Then the whole point cloud is *clustered* based on graph adjacency information and quantized shortest path lengths from the root to all points in the cloud. The center of each cluster is assigned as a skeleton point and the edges are defined as the connection of cluster centers based on their spatial locations. One problem of the technique is that, the skeleton does not maintain the *centerdness* criteria, and results in a *zigzag* structure near the junction point where two or more branches meet. Also, the computed skeleton points sometimes end up being outside the boundary of the point cloud (we discuss about these issues in the next section). Bucksch *et al.* [16] proposed a solution of these type of problems by subdividing the point cloud into octree cells and used some local heuristics to form a skeleton graph from the cells. Zhang *et al.* [17] later applied this technique locally to individual branches. However, these methods rely on several heuristic assumptions and require many parameters to be tuned to obtain the desired result. On a different type of approach, *particle flow* based techniques ([18], [19]) are built on the motivation on the process of transport and exchange of energy and water between root, branches and leaves of a plant. However, the skeleton is not guaranteed to follow the actual geometry of the input point cloud data, and thus suffers from

The Authors are with INRIA Grenoble Rhône-Alpes team MOSAIC, Laboratoire Reproduction et Développement des Plantes, Univ Lyon, ENS de Lyon, UCB Lyon 1, CNRS, INRA. Email: {*ayan.chaudhury,christophe.godin*}*@inria.fr*

biological irrelevance of the results. This technique is mainly used in computer graphics applications. *Space colonization* algorithm ([20], [21]) extracts skeleton from input data by an iterative technique to *"eat-up"* the points in the cloud starting from the root node at the bottom. The overall technique is a local optimization based strategy. Although the technique can produce visually pleasing skeleton structure, the algorithm might result in creating branching structure in the wrong directions, and there is no mechanism to perform *backtracking* to *correct* the biological irrelevance of the branches.

The second type of approach of skeletonization is based on the motivation to *improve* the initial skeleton. Livny *et al.* [22] proposed a series of optimization techniques to *smooth* the branches of a skeleton in a realistic manner. However, the optimization strategy does not involve prior botanical knowledge of plants. Branch smoothing is performed independently without taking into consideration the original data, and can produce over-smoothing or under-smoothing of branches which suffer from botanical inconsistency of the results. In a similar line of work, Wang *et al.* [23] proposed a strategy to refine an existing skeleton to handle the case of occlusion and missing data. The approach is based on a combined local and global optimization strategy, where the coarse skeleton is pushed to move towards the original point cloud, and the original point cloud is forced to contract towards the skeleton points. The method is explicitly built to handle the occlusion cases and overlooks factors like *zigzag* problem and biological irrelevance of the skeleton. In a recent work, Wu *et al.* [24] proposed a skeleton refinement technique for Maize plants. The issue of centerdness criteria and *zigzag* problem is explicitly addressed. The refinement technique is based on a local neighbourhood based approximation strategy and some heurisitcs are used. The method is tested only on a single plant species. Other types of recent skeletonization techniques are shown to be successful for some applications [25], [26], but they assume that the point cloud is voxelized along with voxel connectivity information, instead of considering raw point cloud data without any connectivity information.

We propose a skeletonization technique that belongs to the second category. The aim is to *"correct"* an existing skeleton to maintain the biological relevance with the input point cloud data. Initially we represent a *coarse* skeleton as a tree structure which consists of superposition of parametric curves having uniform density of discrete points. Then we propose a stochastic optimization technique to transform the skeleton points so that the transformed points maintain the centerdness criteria as well as the biological relevance with the input data. The main strength of the approach is the exploitation of the input point cloud data during the optimization process (unlike the previous approaches which optimize the skeleton without taking into consideration the input data), which explicitly helps the skeleton to retain biologically supported structure.

II. Problem Statement

The motivation of this work is to extract the skeleton from plant point cloud data. Before we discuss our model in the next section, we first demonstrate the limitations of the existing approaches on real datasets. We identify the following 3 problems associated with the existing techniques, as shown visually in Figure 1.

A. The Zigzag Structure Problem

One of the typical problems associated with skeletonizing plant point cloud data is the deviation of skeleton points from the centerline of the branch towards the junction point where two or more branches are joined [15], thus creating a *"zigzag"* type skeleton as shown in the left of Figure 1. Ideally the skeleton points should follow the exact *centerline* of the branch representing the actual geometry of the plant. However, the *zigzag* skeleton structure is a wrong approximation of the plant geometry, which results in wrong phenotyping parameter estimation in quantitative measurement of phenotypes such as internode distance, branch length, etc. Although this problem is handled locally in [24], the method is based on many heuristics to skeletonize a handful of Maize plant point cloud data.

B. Invalid and Inaccurate Geometry Estimation Problem

As shown in the middle of Figure 1, sometimes skeleton points do not lie within the enclosing boundary of the original point cloud data [27], [7]. This results in invalid geometry estimation of the input data. For example, in the upper part of the figure we can see the red dots which are located outside the boundary of the point cloud. Also for the case of curved branches, insufficient number of skeleton points are not able to represent the actual geometry of the branch. This is shown at the bottom of the figure where the curved branches are not represented by the skeleton points. One way to handle this problem might be to generate more skeleton points in the curved areas using interpolation or similar point approximation strategies. However, the generated points will be independent of the original point cloud data, and will fail to reconstruct the original geometry of the branch.

C. Inability to Handle Tiny Structures

Tiny branches are often treated as noise in the point cloud data, and the extracted skeleton fails to represent these branches. This is shown at the right of Figure 1. Two tiny branches are completely *overlooked* in terms of skeleton representation of the input data [15].

III. The Model

A. Curve Tree

Initially we obtain a *coarse* skeleton graph of the input point cloud $\mathbb{P} = \{\mathbf{p}_1, \cdots, \mathbf{p}_n\} \in \mathbb{R}^3$ by the method of Xu *et al.* [15]. Note that other skeletonization algorithms can also be used to obtain the coarse skeleton, but we have used this method because the algorithm is simple, fast, works reasonably well, and the open source implementation is available[1]. Next we remove the cycles in the graph by the method of Yan *et al.* [28]. The resultant skeleton point set $\mathbb{S} = \{\mathbf{s}_1, \cdots, \mathbf{s}_k\} \in$

[1] https://github.com/fredboudon/plantscan3d

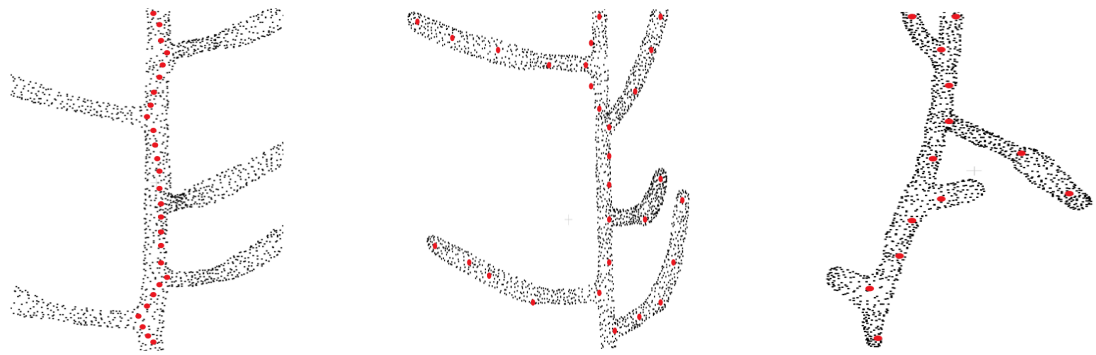

Fig. 1. Problems with the existing approaches on skeletonizing plant point cloud data. Original point cloud is shown as black point cloud and the skeleton points are shown as red points. Left: The problem of *"zigzag"* structure, where the skeleton does not follow the centerline of the stem and tends to deviate towards the branching point [15] (only the main stem skeleton is shown in the figure). Middle: The problem of biologically irrelevant skeleton points which falls beyond the boundary of the input data (shown at the top part), and inability to capture the geometric details for some branches [27], [7]. Right: Overlooking tiny geometrical structures [15].

$\mathbb{R}^3$ is a rooted tree graph embedded in $\mathbb{R}^3$. Formally, let $\mathcal{G} = (V, E)$ be a graph where $V = \{v_i\}_{1 \leq i \leq I}$ (where I is an integer ≥ 1) is the set of vertices, and $E = (v_i, v_j)$ is the set of edges connecting to the ordered pairs of vertices v_i and v_j. The vertex v_j is the child of v_i denoted as, $\text{child}(v_i) = \{v_j \in V | (v_i, v_j) \in E\}$, and the vertex v_i is the parent of v_j denoted as $v_i = \text{parent}(v_j)$. A rooted tree graph is a graph $\mathcal{G}$ that contains no cycle, only one connected component, and there exists a unique vertex in V, called the *root* that has no parent. The *descendants* of a vertex $v_i \in V$ is defined as the set of vertices that belong to the branching system starting from v_i excluding itself. Now we define the following terminologies before introducing the notion of *curve tree*.

Definition 1: Axial Tree, adapted from [29], [30]: An *Axial Tree* $\mathcal{T}_A$ is defined as a tree graph along with a *successor* function $succ$ associated with each vertex in the graph defined as,

$$\begin{aligned} \text{succ} : V &\rightarrow V, \\ v_i &\mapsto v_{i'}, \text{where } v_{i'} \in \text{child}(v_i), \end{aligned}$$

where $v_{i'}$ is called the successor of v_i.

From the definition, we see that all the vertices $v_i \in V$ in an axial tree, there exists at most one successor $v_{i'} \in V$. Note that there may be vertices that do not have a successor in V (such as leaf nodes in particular). Maximal set of vertices connected by successor relationship are paths in $\mathcal{G}$ (called the *axes* of $\mathcal{G}$). For example in Figure 2(a), the axes are denoted as $a1, \cdots, a9$. In this paper, a skeleton is thus represented by an axial tree $\mathcal{T}_A$ for which the vertices correspond to the points in the skeleton, and the edges as the ordering of these points within their axis. The 3D coordinates $\mathbf{s}_i$ of each point is attached to the corresponding vertex v_i.

In a given axis $a = \{v_1, \cdots, v_k\}$ (with slight abuse of notation), we define the *bearing vertex* w as, $w = b(a) \Leftrightarrow w = \text{parent}(v_1)$, if it exists. The *order* relationship between the vertices derived from their order on the path in the axis is defined as, $v_i < v_j \Leftrightarrow v_i$ is an antecedent of v_j. Two vertices v_i and v_j are in *equivalence relationship* (denoted as $v_i \equiv v_j$) if and only if v_i and v_j belong to the same axis of an axial tree. This allows us to define the notion of quotient tree graph as follows.

Definition 2: Quotient Tree Graph, adapted from [30]: Given an axial tree graph $\mathcal{G} = (V, E)$, a *Quotient Tree Graph* $\overline{\mathcal{G}} = (\overline{V}, \overline{E})$ is the quotient graph of the axial tree graph $\mathcal{G}$, which is made of the axes of $\mathcal{G}$, and is also a tree graph.

The vertices in an axial tree having equivalence relationship with each other, are collapsed to a single node in the quotient graph (Figure 2(b)).

Now consider a family of curves $\mathscr{C} \in \mathbb{R}^3$. With each vertex $v_a \in \overline{V}$ in the quotient graph $\overline{\mathcal{G}}$, we associate a curve $\mathcal{C}_a \in \mathscr{C}$ defined as,

$$\begin{aligned} \mathcal{C}_a : [0, 1] &\rightarrow \mathbb{R}^3, \\ u &\mapsto \mathcal{C}_a(u), \end{aligned}$$

where $u \in [0, 1]$ is the parameter of the curve. Curves on the axes must respect the following attachment conditions to one another. For any two vertices a_i, $a_j \in \overline{V}$ such that $a_j = \text{parent}(a_i)$, then the following condition holds:

$$\exists u'_0 \in [0, 1], \text{ s.t. } \mathcal{C}_{a_i}(0) = \mathcal{C}_{a_j}(u'_0). \tag{1}$$

Ideally, we want to represent the skeleton as superposition of curves, where the order of the branching points in the skeleton is the order of vertices in the quotient graph for the attachment of the curves. In order to achieve this, the following condition must hold true in $\overline{\mathcal{G}}$. Consider the axes a_i, a_j, a_k, where $a_i = \text{parent}(a_j)$ and $a_i = \text{parent}(a_k)$. Let w_j and w_k be the respective bearing vertices for the axes a_j and a_k. Then the following condition holds true:

$$\left.\begin{aligned} &w_j \leq w_k \\ &\mathcal{C}_{a_j}(0) = \mathcal{C}_{a_i}(u'_0) \\ &\mathcal{C}_{a_k}(0) = \mathcal{C}_{a_i}(u''_0) \end{aligned}\right\} \Rightarrow u'_0 \leq u''_0. \tag{2}$$

Definition 3: Curve Tree: A *curve tree* $\mathcal{T}_C = (\mathcal{T}_A, g)$ is an axial tree $\mathcal{T}_A$ augmented with an application g that attach parametric curves to each vertex verifying the attachment conditions in Equation 1 and 2 on its quotient tree graph. The application g is defined as,

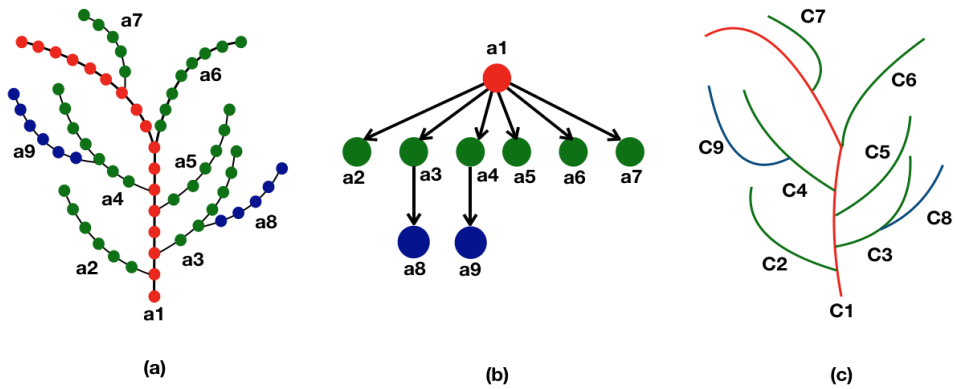

Fig. 2. Representation of skeleton tree graph as a curve tree via quotient graph. Left: each branch $(a1, \cdots, a9)$ in the skeleton tree represents an axis where the discrete points are shown as coloured dots using a hierarchical colour convention. The main stem is represented as red dots, next level branches as green dots, and further level branches as blue dots. The successor of a node of certain colour is coloured with the same colour. Middle: each node in the quotient graph represents each branch (or axis) of the skeleton graph, and the directed edges represent their hierarchical relationship in the skeleton structure. Right: corresponding curve tree where each branch in the skeleton graph is approximated as a parametric curve $(C1, \cdots, C9)$.

$$g : \overline{V} \rightarrow \mathscr{C}$$
$$v_a \mapsto \mathcal{C}_a,$$

where the curve $\mathcal{C}_a$ is associated with each vertex $v_a \in \mathcal{T}_A$. In order to instantiate the curve associated with each axis a, we choose a family of curves that may be parameterized using the 3D points $\{v_1, \cdots, v_k\}$ of the axis a. At this stage, the curve tree basically represents the skeleton graph as a superposition of a set of curves. However, the point density in the curve tree may not be uniform. Insufficient number of points might lead to wrong geometry reconstruction of the branches, as discussed earlier. We propose a *resampling* strategy to uniformly upsample the number of points by approximating the curves as splines as follows.

B. Resampling Strategy

We aim at resampling each branch of the curve tree, where the number of points in a branch can be modelled to the desired number. We have used the following strategy.

First we approximate each branch of the curve tree as a spline. Splines are powerful mathematical tools to approximate curves and surfaces. In the general case, a spline curve $\mathcal{S}(u)$ defined by $(n+1)$ control points[2] $P_0, P_1, \cdots, P_n$ can be defined as,

$$\mathcal{S}(u) = \sum_{i=\lfloor nu-2 \rfloor}^{\lceil nu+2 \rceil} \mathcal{B}(nu - i) P_i, \tag{3}$$

where $\mathcal{B}(\cdot)$ is the blending function of the spline, and $u \in [0, 1]$ [31]. For a cubic B-spline, at most four nearest control points are used to compute the blending function for one spline point. Let the Euclidean distance (we are not taking into account the branch bending in the distance computation) between the endpoints of a branch is d (in mm). Then we approximate the number of spline points as $\lceil d/d_0 \rceil$, where we use $d_0 = 1.0$. Depending on the application, higher or lower density of points can be achieved easily by changing the value of k to a lower or higher value, respectively. Number of spline points can be greater than, equal to or less than the number of control points. Any curve thus can be represented as the polyline joining continuous set of spline points. Although cubic B-splines are widely used in modelling curves, we model the branch curves as β-splines, which provide an intermediate representation between approximation and interpolation and thus captures the local geometry better than cubic B-spline. Given the fact that β splines are 1st and 2nd order continuous [32], we can approximate the branches as *smooth enough*. The blending function of a β-spline denoted as $\beta(v)$, parameterized by v is given by,

$$\beta(v) = \begin{cases} \frac{2}{\delta}(2+v)^3, & -2 \leq v \leq -1 \\ \frac{1}{\delta}((t+4s+4s^2) - 6(1-s^2)v - 3v^2(2+t+2s) \\ -2v^3(1+t+s+s^2)), & -1 \leq v \leq 0 \\ \frac{1}{\delta}((t+4s+4s^2) - 6v(s-s^3) - 3v^2(t+2s^2+ \\ 2s^3) + 2v^3(t+s+s^2+s^3)), & 0 \leq v \leq 1 \\ \frac{2}{\delta}s^3(2-v)^3, & 1 \leq v \leq 2, \end{cases} \tag{4}$$

where $v = nu - i$, t is the tension parameter and s is the skew parameter ([32], [33]), which are kept constant. We used $t = 10$ and $s = 1$ for all of our experiments. While the skew parameter is always set to 1 to ensure the smoothness of the curve, we have tested different values of t on several datasets. From empirical observation, we notice that keeping $t \in [5, 15]$ produces best approximation of the curve (very low error value of mean square distance between the original points and generated points), while other values of t outside this range yields high error value (the error corresponding to $t = 10$ was the minimum). δ is given by $\delta = t+2s^3+4s^2+4s+2$. Basically the blending function is defined for 4 discrete intervals (2 in

[2]By *control points*, we mean the given set of discrete input points, whereas by *spline points* we mean the generated points on the spline curve.

each side) around each control point. While computing the spline points near the endpoints, we use the trick of truncating the generated spline points to the endpoints of the curve in order to prevent the generated points to go *beyond* the boundary of the curve.

At this stage, the curve tree represents a coarse skeleton of the plant having uniform distribution of discrete points. However, the skeleton still has the *zigzag* structure and non-centerdness problem. We introduce a stochastic optimization approach to *"correct"* the current skeleton to follow the geometry of the original point cloud in order to obtain a biologically relevant skeleton of the input point cloud data.

C. Stochastic Modelling

Ideally, the skeleton points should follow the exact centerline of the input point cloud data. Considering the branches of the plant as cylindrical, the skeleton should follow the central axis of the cylinder in each branch. The skeleton in the curve tree suffers from the problem of not maintaining the centerline with respect to the cylinder axis, thus resulting in having points which are *deviated* from the central axis of the cylinder. Now the goal is to *correct* the deviated skeleton by moving the points towards the centerline by maintaining the branching structure of the plant.

We frame the problem of skeleton refinement as a transformation estimation problem as follows. We aim at *moving* (or transforming) the discrete points of the skeleton towards the original input point cloud data, so that the skeleton points get aligned with the centerline of the original point cloud data. However, the problem is highly non-rigid in nature. Global transformation of points is not sufficient to achieve the optimal solution, and we aim at estimating transformation for every point in the skeleton. Exploiting the recent advancements of Gaussian Mixture Model based approaches ([34], [35], [36]), we formulate the problem in a probabilistic framework.

We denote the original point cloud $\mathbb{P}$ as the *fixed* point set, and the skeleton point cloud $\mathbb{Q} = \{\mathbf{q}_1, \cdots, \mathbf{q}_m\} \in \mathbb{R}^d$ of the spline tree as the *moving* point set, where *d* is the dimension of the point cloud (which is 3 in our case). In a typical case of point set registration, $m \approx n$, and there is a correspondence between most of the point pairs $(\mathbf{q}_i, \mathbf{p}_j)$. However in our case $m \ll n$, and instead of one-to-one correspondence, a group of points in the original point set corresponds to a single point in the skeleton point set. We consider the input point cloud as the observation of a Gaussian Mixture Model (GMM), and the centroids of the Gaussians are considered as the skeleton point set. Now in order to estimate the point correspondence between the two sets, we aim at finding which Gaussian the points in the skeleton point set are sampled from. Let $\mathbf{\Theta}$ be the set of unknown model parameters (we define $\mathbf{\Theta}$ later). We estimate the centroid locations $\mathbb{Q}^*$ from the optimal set of model parameters $\mathbf{\Theta}^*$, which is obtained by minimizing the following negative log-likelihood,

$$\mathbf{\Theta}^* = \arg\min_{\mathbf{\Theta}} \; -ln(p(\mathbb{Q}|\mathbb{P}, \mathbf{\Theta}). \tag{5}$$

This type of problem can be solved by Expectation Maximization (EM) framework, where the E-step estimates which Gaussian the observed point cloud was sampled from, and the M-step maximizes the negative log-likelihood that the observed points were sampled from the GMM with respect to the model parameters. Each $\mathbf{p}_j$ is assumed to be the centroid of a Gaussian, and the corresponding $\mathbf{q}_i$ is assumed to be normally distributed around $\mathbf{p}_j$. The probability of each $\mathbf{q}_i$ in the skeleton point set given a point $\mathbf{p}_j$ in the original point cloud is therefore,

$$p(\mathbf{q}_i|\mathbf{p}_j) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{1}{2} \left\| \frac{\mathbf{q}_i - \mathbf{p}_j}{\sigma} \right\|^2 \right), \tag{6}$$

where σ is the covariance of the Gaussian, whose initial value $\sigma_{(0)}$ is set as,

$$\sigma^2_{(0)} = \frac{1}{dmn} \sum_{i=1}^{n} \sum_{j=1}^{m} \|\mathbf{q}_i - \mathbf{p}_j\|^2 . \tag{7}$$

Now we introduce outliers in the model, assuming the outlier probability to have uniform distribution for all data points. Let $\epsilon \in [0, 1]$ be the weight of the outlier distribution in the original point set (that means, each point has the probability ϵ/n), and $(1 - \epsilon)$ be the weight in the skeleton point set. Then the mixture model takes the form,

$$p(\mathbf{q}_i|\mathbf{\Theta}) = \frac{\epsilon}{n} + (1 - \epsilon) \sum_{j=1}^{m} p(\mathbf{q}_i|\mathbf{p}_j), \tag{8}$$

where $\mathbf{\Theta} = \{\sigma^2, \epsilon\}$ is the set of model parameters, which we wish to estimate in EM framework in order to minimize the negative log-likelihood energy of Equation 5. Next, we propose a membership probability $p(\mathbb{P})$ for the GMM components. Instead of assigning equal probability for all the points, a *"confidence score"* is used to indicate the probability of point correspondence between the two point sets. The probability that the observed point $\mathbf{q}_i$ is an observation of the Gaussian of point $\mathbf{p}_j$ is defined by α_{ij} as the membership probability. We use a local Principal Component Analysis (PCA) based technique to estimate the membership probability as follows. Considering a local neighborhood $\mathcal{N}_i$ around the *i*-th point defined as the points within a ball of radious *r*, the 3×3 covariance matrix $\mathcal{C}_i$ is computed as,

$$\mathcal{C}_i = \frac{1}{N} \sum_{k=1}^{N} (\mathbf{v}_k - \bar{\mathbf{v}}_k)(\mathbf{v}_k - \bar{\mathbf{v}}_k)^T, \tag{9}$$

where v_k are the points in the local neighbourhood of *i*-th point, N is the number of points in the neighbourhood, and $\bar{v}_i = \frac{1}{N} \sum_{k \in N} v_k$. If the eigenvalues of the matrix $\mathcal{C}_i$ are $\lambda_0, \lambda_1, \lambda_2$ (where $\lambda_0 \leq \lambda_1 \leq \lambda_2$), then we compute the normalized eigenvalueof the neighbourhood as,

$$\mu_i = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}. \tag{10}$$

Note that instead of normalizing the lowest eigenvalue, other eigenvalues can also be normalized to compute μ_i. We use μ_i as an estimate of the local structure of the neighbourhood of *i*-th point. Similarly we compute μ_j for the other point set. Large difference of $\mu = |\mu_i - \mu_j|$ between the local

neighbourhood is penalized by assigning low probability, whereas small difference contributes to high probability. The membership probability that i-th point corresponds to j-th point is computed as an exponentially decaying function as,

$$\alpha_{ij} = \exp\left(-\alpha|\mu_i - \mu_j|\right), \tag{11}$$

where α is a tuning parameter (set to 1 for all of our experiments).

Along with the membership probability in the negative log-likelihood energy of Equation 5, we minimize the following energy by EM algorithm,

$$\mathcal{L} = -\sum_{j=1}^{n} log \sum_{i=1}^{m+1} p(\mathbf{p}_j) p(\mathbf{q}_i|\mathbf{p}_j). \tag{12}$$

E-step: In the E-step, the current parameter values (the *"old"* set of parameters) are used to estimate the posterior distribution $p^{\text{old}}(\mathbf{p}_j|\mathbf{q}_i)$ using Bayes' theorem as,

$$p^{\text{old}}(\mathbf{p}_j|\mathbf{q}_i) = \frac{p(\mathbf{p}_j)\ p(\mathbf{q}_i|\mathbf{p}_j)}{p(\mathbf{q}_i)}. \tag{13}$$

In order to evaluate the probability p^{old} in the above equation, we use the expressions from Equation 6, 11, and 8. The first term in the numerator is the probability of correspondence of $\mathbf{p}_j$ with $\mathbf{q}_i$. This term is estimated as α_{ij} as in Equation 11. Incorporating the outliers ϵ in the model, the second term in the numerator is estimated as $(1-\epsilon)p(\mathbf{q}_i|\mathbf{p}_j)$. The denominator is obtained from Equation 8. Hence the expression for p^{old} is written as,

$$p^{\text{old}}(\mathbf{p}_j|\mathbf{q}_i) = \frac{\alpha_{ij}\ \exp\left(-\frac{\|\mathbf{q}_i-\mathbf{p}_j\|^2}{2\sigma^2}\right)}{\frac{\epsilon(2\pi\sigma^2)^{d/2}}{n(1-\epsilon)} + \sum_{k=1}^{m} \alpha_{ik} \exp\left(-\frac{\|\mathbf{q}_i-\mathbf{p}_k\|^2}{2\sigma^2}\right)}. \tag{14}$$

M-step: In the M-step, *"new"* parameter set is obtained by minimizing the expectation of the complete negative log-likelihood function of Equation 12 as,

$$\mathcal{L} = -\sum_{j=1}^{n}\sum_{i=1}^{m+1} p^{\text{old}}(\mathbf{p}_j|\mathbf{q}_i) \log(p^{\text{new}}(\mathbf{p}_j) p^{\text{new}}(\mathbf{q}_i|\mathbf{p}_j)). \tag{15}$$

Ignoring the constants independent of $\mathbf{\Theta}$, we can write the expression for complete likelihood $\mathcal{L}_c$ as [36],

$$\mathcal{L}_c = \frac{1}{2\sigma^2}\sum_{j=1}^{n}\sum_{i=1}^{m+1} p^{\text{old}}(\mathbf{p}_j|\mathbf{q}_i)\,\|\mathbf{q}_i - \mathbf{p}_j\|^2 + \frac{\xi d}{2}\log(\sigma^2) - \xi\log(1-\epsilon) - (m-\xi)\log(\epsilon) + \lambda\ ||\mathbb{G}||, \tag{16}$$

where $\xi = \sum_{j=0}^{n}\sum_{i=0}^{m} p^{\text{old}}(\mathbf{p}_j|\mathbf{q}_i)$, and the last term is a regularizer weighted by a factor λ to restrict the *"spread"* of the data by the transformation. For a skeleton point $\mathbf{q}_i$ and the corresponding transformed point $\mathbf{q}'_i$ is regularized as the Gaussian kernel, $\mathbb{G}(\mathbf{q}_i, \mathbf{q}'_i) = \exp -||\mathbf{q}_i - \mathbf{q}'_i||^2/2\beta^2$, where the variable β controls the *spread* of the data. If the value of β is too small, the transformation will have negligible effect on the data, while large value of β will result in badly scattered points. Minimizing $\mathcal{L}_c$ decreases the negative log-likelihood of the energy function.

The EM algorithm iterates until there is not sufficient update of the parameters (we keep the tolerance at 10^{-4}). The updated location of skeleton point set $\mathbb{Q}$ is the optimal transformed point set $\mathbb{Q}^*$.

IV. Experiments and Analyses

We performed several experiments with real world and synthetic datasets in both qualitative and quantitative manners. More specifically, we have experimented with the data from 3 datasets. The first dataset we have used is the PlantScan3D dataset [37], which contains different varieties of plants including Cherry, and Apple tree point cloud data. Next, we used the dataset from Tabb *et al.* [26]. The dataset contains synthetic data in voxel format. We have extracted the raw point cloud from the data, and performed experiments without using any connectivity information from the voxel structure. Finally, we have generated point cloud data by scanning real Arabidopsis thaliana plants. For all the datasets, we obtained the initial coarse skeleton using the method of Xu *et al.* [15], and performed the proposed optimization technique. Although other types of skeletonization methods can be used to obtain the initial skeleton, the result of the optimization algorithm depends on the quality of the initial skeleton. We have performed the optimization by using Space Colonization algorithm ([20], [21]) as the initial skeleton, where we selected the set of parameters which produced best results. We noticed that in many cases, the results suffer from problems where the skeleton points are too far from the centerline. Especially the branch junction points are located out of the boundary of the point cloud in many cases. The optimization performs poorly in these cases than using the method of Xu *et al.* [15] as the initial skeleton. We used Python 2.7 version along with PlantScan3D library for the implementation. The experiments were performed in a MacBook Pro with 2.2 GHz Intel Core i7 processor and 32GB DDR4 RAM. For a point cloud with about 20k points, it takes less than 1 minute to obtain the optimized skeleton.

First we demonstrate some qualitative results in Figure 3, where we show the skeletonization results on Arabidopsis plant and Cherry tree point cloud data. For visual clarity, we show part of the original data for the Arabidopsis plant. Similarly for the Cherry tree data, we removed the noise and truncated part of a branch to demonstrate the quality of the skeleton. In Figure 4, we show the full skeleton structures along with the original point cloud data for a synthetic tree [26], cherry tree, and arabidopsis plant, respectively.

Next, we demonstrate the quantitative results of different plant phenotypes obtained from the computed skeleton. We have generated ground truth of junction points of the branches (branching point) for 3 cases: synthetic data, Arabidopsis plant data, and cherry tree data. We notice that the quality of the skeleton explicitly affects the location of the branching point and the relative length of branch segment between the branching points with respect to the average branch length. In order

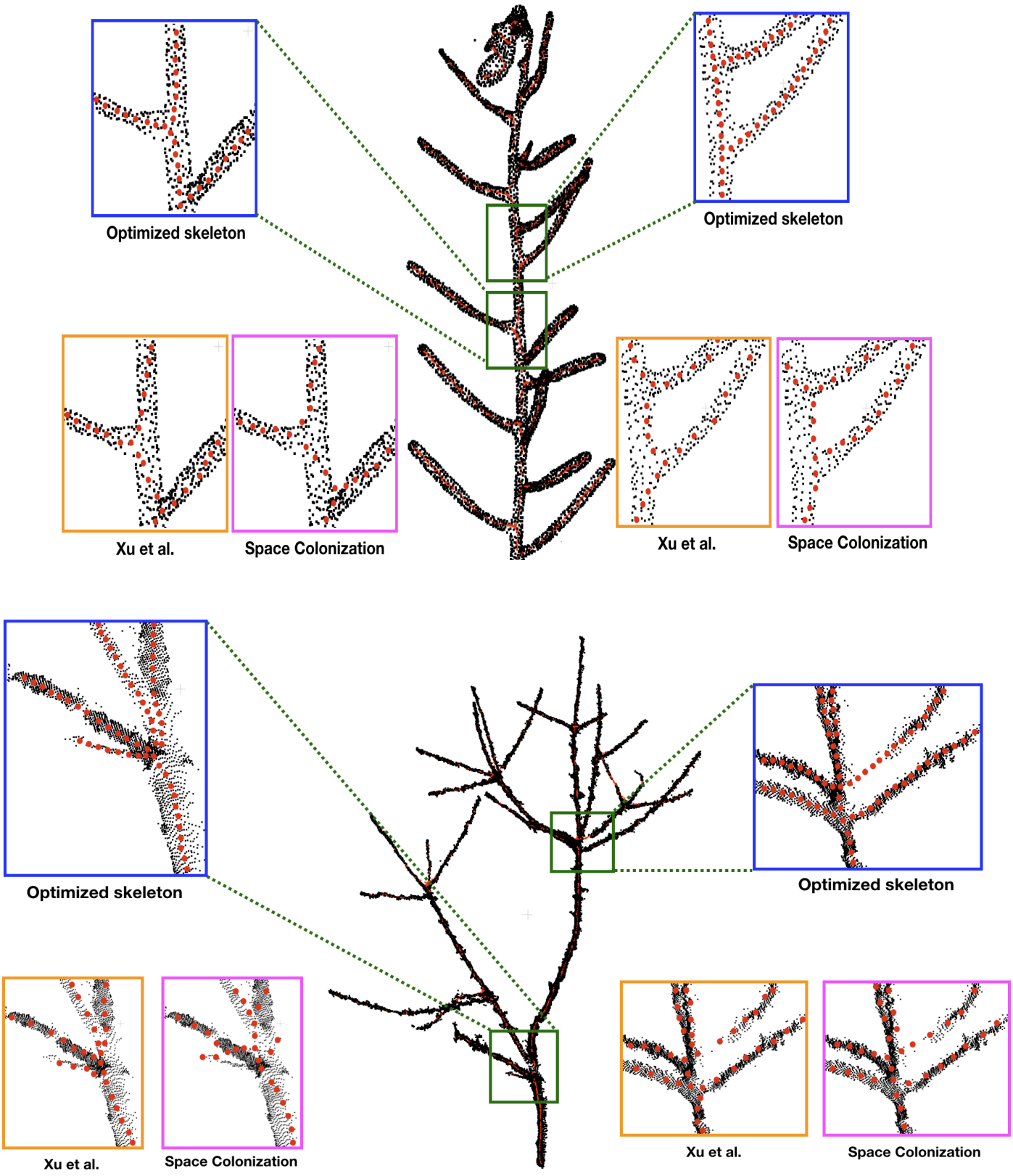

Fig. 3. Skeletonization results on 2 datasets: Arabidopsis plant (top) and Cherry tree (bottom) point cloud data. Note that the red skeleton points lie *"inside"* the plant stem, and that is why the visibility is not always clear due to the occlusion by the original (black) point cloud. We show some parts of the data by zooming-in and showing the skeleton points at the *surface* of the plant by taking the points *"out"* of the plant.

Fig. 4. Results of skeletonization on the whole plant point cloud data. Left: dataset from Tabb *et al.* [26], middle: apple tree dataset from INRIA Plantscan3d dataset, right: arabidopsis data.

to evaluate the quality of the skeleton, we use two metrics based on the above mentioned factors to quantify the errors. The first metric is the deviation of the computed junction point from the ground truth location, and the second metric is the difference between the ground truth branch segment length and the computed branch segment length with respect to the average branch length. We compare our results with the state-of-the-art skeletonization algorithms which are built specifically for plants, consider raw point cloud data as input to the algorithm, and the implementation is publicly available. More specifically, we compare our algorithm with Xu *et al.* [15] and Space Colonization algorithm [20], [21] implemented in PlantScan3D library. In Figure 5, we show the error bar plots for junction point location errors and relative branch segment length errors in top and bottom row respectively. In each case, we have computed the error with respect to the groundtruth for all the algorithms. The first column shows the results on synthetic data having 12 branching points, the second column on Arabidopsis data having 13 branching points, and the third column on Cherry tree data having 12 branching points. In general, our technique outperforms the other algorithms on most of the cases, as can be seen from the bar graphs for individual cases.

We also show the statistical distribution of the corresponding error values in the box plots of Figure 6. Similar to Figure 5, the top row shows the results of junction point location errors and the bottom row shows the results of relative branch segment length errors. As can be seen from the figure, the span of error values for the proposed optimization algorithm is always less than the other methods.

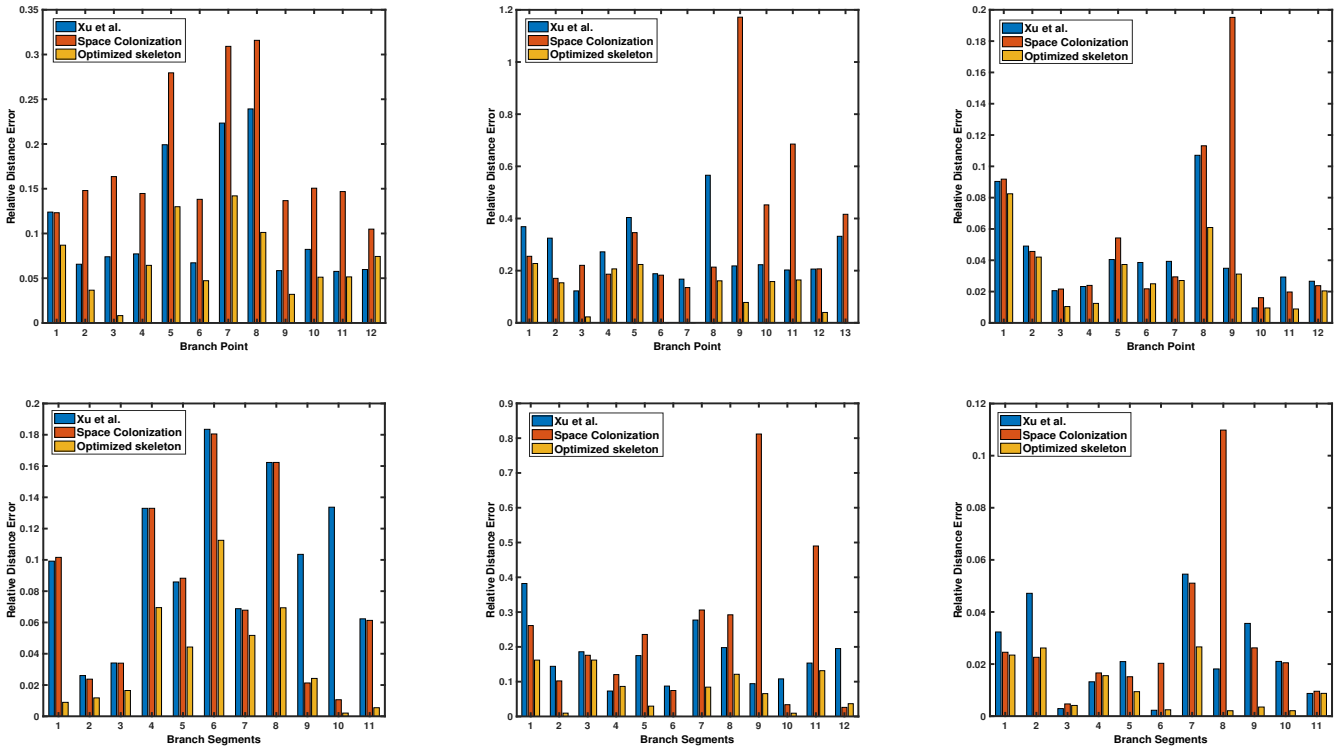

Fig. 5. Error bar graphs for junction point location (top row) and relative branch segment length (bottom row) for 3 datasets: a synthetic data, Arabidopsis plant data (shown in each column), and Cherry tree data. In each graph, the vertical bar represents the absolute value of error with respect to the groundtruth. Performance of each algorithm is shown in different colours, as defined by the legend in each graph.

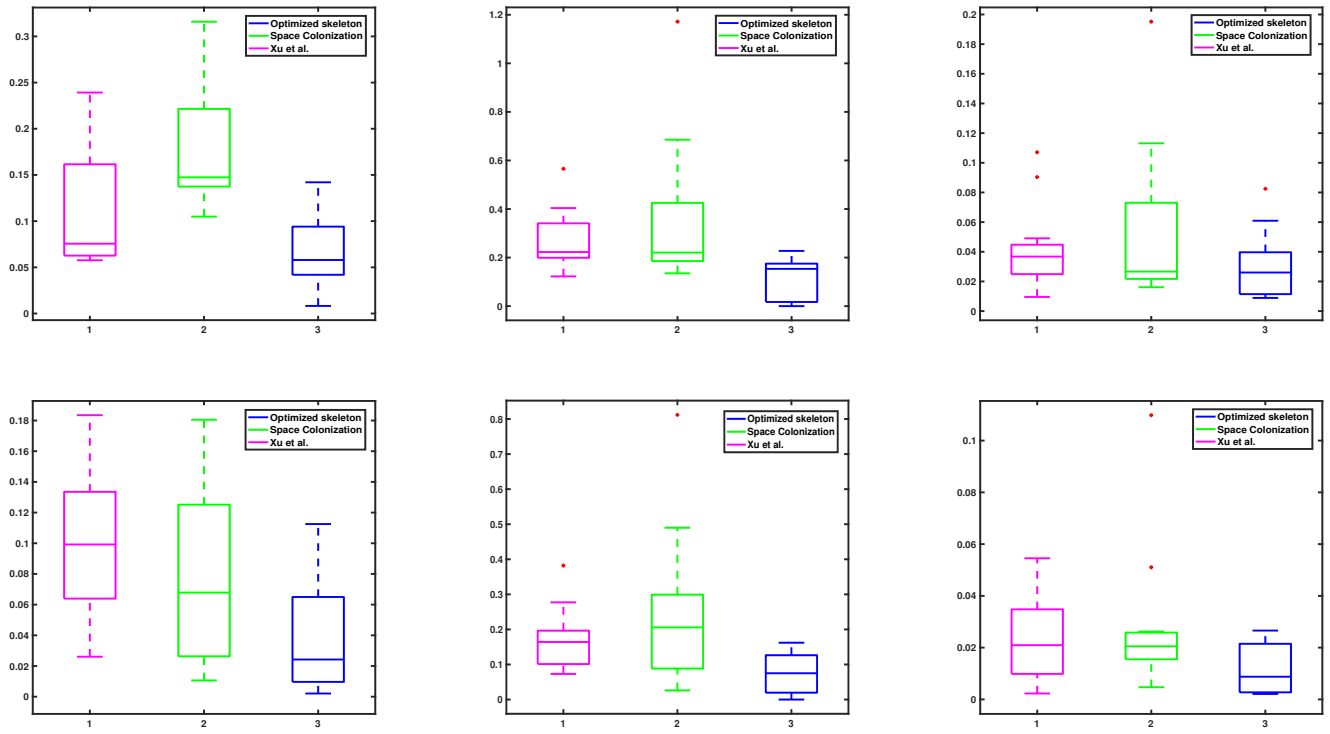

Fig. 6. Box plots corresponding to the data values in Figure 5. The top row shows the results of junction point location errors, and the bottom row is the result of relative branch segment length errors with respect to the groundtruth. In each graph, the central mark of each box indicates the median, while the bottom and top edges of the box indicate the 25-th and 75-th percentiles of the data, respectively. The whiskers extend to the most extreme data points which are not considered as outliers. The red '+' symbols plot the outliers in the data, if there is any.

V. Discussion

In this work, we have proposed a technique to *improve* an existing skeleton by a stochastic optimization technique. The idea of exploiting the original point cloud data in the optimization process allows the skeleton to retain biological relevance with respect to the input point cloud data. The method is tested on different types of data, and improvement over the existing technique is demonstrated.

Two directions of future work might be interesting to study. We have not considered the case where there is lot of noise in the data. Also, explicit handling of occlusion is not modelled in the framework. While the optimization works well in general, we notice that for thick branches, the skeleton curve tends to have discontinuous derivatives or high curvature near the branching points. In order to reconstruct the geometry of the branching structure by the notion of generalized cylinders, it will be problematic with the high curvature areas. Handling these types of cases are not considered here.

With the goal of reconstructing the geometry of the original plant point cloud data, the skeleton can play an important role. In this regard, constructing a graph from the discrete skeleton points will be a challenging problem. Similar type of problem is handled by some heuristic approach by Wu *et al.* [24], but a generic solution to the problem can be a major part of plant geometry reconstruction. The proposed method considers plants having no leaves. Skeletonization by considering plant leaves will be worth studying in order to reconstruct the geometry of leafy plants.

Acknowledgement

This work is supported by Robotics for Microfarms (ROMI) European project. We would like to thank Frédéric Boudon for useful discussions and making the implementation of PlantScan3D library available for public use. We would also like to thank Julie Charlaix for collecting and sharing the data, and other members of the ROMI project for the (not yet published) pipeline to reconstruct the point cloud from images.

References

[1] E. P. Spalding and N. D. Miller, "Image analysis is driving a renaissance in growth measurement," *Current Opinion in Plant Biology*, vol. 16, no. 1, pp. 100–104, 2013.

[2] F. Tardieu, L. Cabrera-Bosquet, T. Pridmore, and M. Bennett, "Plant phenomics, from sensors to knowledge," *Current Biology*, vol. 27, no. 15, pp. R770–R783, 2017.

[3] N. Brichet, C. Fournier, O. Turc, O. Strauss, S. Artzet, C. Pradal, C. Welcker, F. Tardieu, and L. Cabrera-Bosquet, "A robot-assisted imaging pipeline for tracking the growths of maize ear and silks in a high-throughput phenotyping platform," *Plant Methods*, vol. 13, no. 1, p. 96, 2017.

[4] X. Zhang, C. Huang, D. Wu, F. Qiao, W. Li, L. Duan, K. Wang, Y. Xiao, G. Chen, Q. Liu, L. Xiong, W. Yang, and J. Yan, "High-throughput phenotyping and qtl mapping reveals the genetic architecture of maize plant growth," *Plant physiology*, vol. 173, no. 3, pp. 1554–1564, 2017.

[5] S. D. Choudhury, A. Samal, and T. Awada, "Leveraging image analysis for high-throughput plant phenotyping," *Frontiers in plant science*, vol. 10, 2019.

[6] M. Vázquez-Arellano, H. W. Griepentrog, D. Reiser, and D. S. Paraforos, "3-d imaging systems for agricultural applications—a review," *Sensors*, vol. 16, no. 5, 2016.

[7] I. Ziamtsov and S. Navlakha, "Machine learning approaches to improve three basic plant phenotyping tasks using three-dimensional point clouds," *Plant physiology*, vol. 181, no. 4, pp. 1425–1440, 2019.

[8] S. Chattopadhyay, S. A. Akbar, N. M. Elfiky, H. Medeiros, and A. Kak, "Measuring and modeling apple trees using time-of-flight data for automation of dormant pruning applications," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016, pp. 1–9.

[9] A. Chaudhury, C. Ward, A. Talasaz, A. G. Ivanov, M. Brophy, B. Grodzinski, N. P. A. Hüner, R. V. Patel, and J. L. Barron, "Machine vision system for 3d plant phenotyping," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 16, no. 6, pp. 2009–2022, 2019.

[10] N. D. Cornea, D. Silver, and P. Min, "Curve-skeleton properties, applications, and algorithms," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 3, pp. 530–548, 2007.

[11] A. Tagliasacchi, T. Delame, M. Spagnuolo, N. Amenta, and A. Telea, "3d skeletons: A state-of-the-art report," in *Proceedings of the 37th Annual Conference of the European Association for Computer Graphics: State of the Art Reports*, 2016, pp. 573–597.

[12] A. Tagliasacchi, H. Zhang, and D. Cohen-Or, "Curve skeleton extraction from incomplete point cloud," *ACM Transactions on Graphics*, vol. 28, no. 3, pp. 71:1–71:9, 2009.

[13] J. Cao, A. Tagliasacchi, M. Olson, H. Zhang, and Z. Su, "Point cloud skeletons via laplacian based contraction," in *Proceedings of Shape Modeling International (SMI)*, 2010, pp. 187–197.

[14] H. Huang, S. Wu, D. Cohen-Or, M. Gong, H. Zhang, G. Li, and B. Chen, "L1-medial skeleton of point cloud," *ACM Transactions on Graphics*, vol. 32, no. 4, pp. 65:1–65:8, 2013.

[15] H. Xu, N. Gossett, and B. Chen, "Knowledge and heuristic-based modeling of laser-scanned trees," *ACM Transactions on Graphics*, vol. 26, no. 4, 2007.

[16] A. Bucksch, R. C. Lindenbergh, and M. Menenti, "Skeltre - fast skeletonisation for imperfect point cloud data of botanic trees," in *Proceedings of Eurographics Conference on 3D Object Retrieval*, 2009, pp. 13–20.

[17] D. Zhang, N. Xie, S. Liang, and J. Jia, "3d tree skeletonization from multiple images based on pyrlk optical flow," *Pattern Recognition Letters*, vol. 76, 12 2015.

[18] Y. Rodkaew, P. Chongstitvatana, S. Siripant, and C. Lursinsap, "Particle systems for plant modeling," in *Proceedings of Plant Growth Modeling and Applications*, 2003, pp. 10–17.

[19] G. Zeng, J. Wang, S. B. Kang, and L. Quan, "Image-based tree modeling," *ACM Transactions on Graphics*, vol. 26, no. 3, 2007.

[20] A. Runions, B. Lane, and P. Prusinkiewicz, "Modeling trees with a space colonization algorithm," in *Proceedings of Eurographics Workshop on Natural Phenomena*, 2007, pp. 63–70.

[21] C. Preuksakarn, F. Boudon, P. Ferraro, J. B. Durand, E. Nikinmaa, and C. Godin, "Reconstructing plant architecture from 3d laser scanner data," in *Proceedings of 6th International Workshop on Functional Structural Plant Models*, 2010.

[22] Y. Livny, F. Yan, M. Olson, B. Chen, H. Zhang, and J. El-Sana, "Automatic reconstruction of tree skeletal structures from point clouds," *ACM Transactions on Graphics*, vol. 29, no. 6, pp. 151:1–151:8, 2010.

[23] W. Zhen, L. Zhang, T. Fang, X. Tong, P. T. Mathiopoulos, L. Zhang, and J. Mei, "A local structure and direction-aware optimization approach for three-dimensional tree modeling," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 8, pp. 4749–4757, 2016.

[24] S. wu, W. Wen, B. Xiao, X. Guo, J. Du, and C. Wang, "An accurate skeleton extraction approach from 3d point clouds of maize plants," *Frontiers in Plant Science*, vol. 10, 2019.

[25] D. Jin, K. S. Iyer, C. Chen, E. A. Hoffman, and P. K. Saha, "A robust and efficient curve skeletonization algorithm for tree-like objects using minimum cost paths," *Pattern Recognition Letters*, vol. 76, pp. 32–40, 2016.

[26] A. Tabb and H. Medeiros, "Fast and robust curve skeletonization for real-world elongated objects," in *IEEE Winter Conference on Applications of Computer Vision*, 2018.

[27] S. Delagrange, C. Jauvin, and P. Rochon, "Pypetree: a tool for reconstructing tree perennial tissues from point clouds," *Sensors*, vol. 14, no. 3, pp. 4271–4289, 2014.

[28] D. Yan, J, Wintz, B. Mourrain, W. Wang, F. Boudon, and C. Godin, "Efficient and robust reconstruction of botanical branching structure from laser scanned points," in *IEEE International Conference on Computer Aided Design and Computer Graphics (CAD)*, 2009.

[29] P. Prusinkiewicz and A. Lindenmayer, *The algorithmic beauty of plants*. Springer-Verlag, 1990.

[30] C. Godin and Y. Caraglio, "A multiscale model of plant topological structures," *Journal of theoretical biology*, vol. 191, no. 1, pp. 1–46, 1998.

[31] P. Burger and D. Gillies, *Interactive Computer Graphics*. Addison-Wesley Publishing Company, 1989.

[32] T. N. T. Goodman, "Properties of beta splines," *Journal of Approximation Theory*, vol. 44, no. 2, pp. 132–153, 1985.

[33] T. N. T. Goodman and K. Unsworth, "Manipulating shape and producing geometuic contnuity in beta spline curves," *IEEE Computer Graphics and Applications*, vol. 6, no. 2, pp. 50–56, 1986.

[34] B. Jian and B. C. Vemuri, "Robust point set registration using gaussian mixture models," *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1633–1645, 2011.

[35] A. Myronenko and X. B. Song, "Point set registration: Coherent point drift," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2262–2275, 2010.

[36] J. Ma, J. Zhao, and A. L. Yuille, "Non-rigid point set registration by preserving global and local structures," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 53–64, 2016.

[37] "PlantScan3D," https://gforge.inria.fr/frs/?group_id=2512, [Online; accessed January 2020].