# Automatic Segmentation of Tree Structure From Point Cloud Data

Sundara Tejaswi Digumarti, Juan Nieto, Cesar Cadena, Roland Siegwart, and Paul Beardsley

***Abstract*—Methods for capturing and modeling vegetation, such as trees or plants, typically distinguish between two components—branch skeleton and foliage. Current methods do not provide quantitatively accurate tree structure and foliage density needed for applications such as visualization, inspection, or to estimate vegetation parameters. This letter describes an automatic method for segmenting three-dimensional point cloud data of vegetation, acquired from commodity scanners, into its two main components: *branches and leaves*, by using geometric features computed directly on the point cloud. In this letter, the specific type of vegetation considered is broadleaf trees. We present a data-driven approach, where a Random forest classifier is used for segmentation. In contrast to state-of-the-art methods, the point cloud is not reduced to a set of primitives such as cylinders. Instead, the algorithm works at the level of the input point cloud itself, preserving quantitative accuracy in the resulting model. Computation of typical vegetation metrics follows naturally from this model. We achieve an average classification accuracy of 91% on simulated data across three different species of broadleaf trees. Qualitative results on real data are also presented.**

***Index Terms*—Robotics in agriculture and forestry, object detection, segmentation and categorization, RGB-D perception.**

## I. Introduction

FINE-SCALE spatial and temporal sensing and modelling of trees underpins plantation management in orchards [1], [2] and forestry [3]. For example, one of the key measurements for characterizing trees, in ecology as well as in agriculture, is Leaf Area Index (LAI). LAI is defined as the one-sided green leaf area per unit ground surface area for broadleaf trees [4] (with alternative definitions for conifers, not relevant in this work) and typically varies from around 1 to around 10. The measurement of LAI supports the estimation of other fundamental metrics for vegetation such as biomass and photosynthesis.

However, accurate estimation of these metrics is challenging, because it is difficult to capture the geometry of a tree - the branching structure plus the foliage density throughout the tree. For plantation management, measurements are currently done by sparse manual sampling, or using hand-held devices that make relatively limited measurements of tree geometry. Aerial imagery is also used for estimating leaf coverage over large geographical scales but with the obvious limitation that the viewpoint and captured data is constrained to the top view of the canopy.

Advances in sensing and motion planning are enabling new mobile robotics technology to capture high resolution data from the environment [5]. However, there is a research gap between acquisition and automatic processing of this data, particularly for unstructured environments such as vegetation.

This motivates our work on the capture of digital models of vegetation that are as quantitatively accurate as possible. Beyond applications in plantation management, quantitative accuracy of vegetation models is also relevant to other areas such as ecosystem and climate modelling, digital content for animated movies and games, and virtual environments for scientific study and education.

While technology for the reconstruction of urban and human-made structures is increasingly mature [6], [7], the same algorithms do not readily translate to the reconstruction of vegetation for a combination of reasons - vegetation has complex non-parametric geometry, repeated structure, frequent occlusion, limited variation of colour as a usable cue, and non-rigidity. Vegetation also has a multi-scale nature with characteristics that are relevant for applications all the way from global branch skeleton down to shape of individual fruits.

To address these challenges, this letter proposes an algorithm to segment point clouds of trees or tree-like vegetation into two main components - branch structure and foliage. We consider broadleaf trees, in which branches can be treated as closed surfaces that enclose space (like cylinders), while leaves are curved surfaces that do not enclose space [1]. In this letter we shall refer to such objects as those that have a *complex geometry*. This contrasts with coniferous trees with needle-like leaves, which are not considered in this letter.

The input to the proposed method is a 3D point cloud of a tree (or a plant). Each point is associated with a feature vector consisting of surface curvatures extracted at multiple local neighbourhoods of different sizes. The neighbourhood sizes vary for each point and are determined as a factor of the distance of the point from its skeleton, thus taking into account the changes in

Manuscript received February 24, 2018; accepted June 12, 2018. Date of publication June 21, 2018; date of current version July 9, 2018. This letter was recommended for publication by Associate Editor D. Milutinovic and Editor W. K. Chung upon evaluation of the reviewers' comments. *(Corresponding author: Sundara Tejaswi Digumarti.)*

S. T. Digumarti is with the Autonomous Systems Lab, ETH Zürich, Zürich 8092, Switzerland, and also with the Disney Research, Zürich 8006, Switzerland (e-mail: dtejaswi@mavt.ethz.ch).

J. Nieto, C. Cadena, and R. Siegwart are with the Autonomous Systems Lab, ETH Zürich, Zürich 8092, Switzerland (e-mail: jnieto@ethz.ch; cesarcadena.lerma@gmail.com; rsiegwart@ethz.ch).

P. Beardsley is with the Disney Research, Zürich 8006, Switzerland (e-mail: pab@disneyresearch.com).

Digital Object Identifier 10.1109/LRA.2018.2849499

[1]The approach may be generalizable to handle other (typically organic) classes of object as well, which are composed of several geometrically distinctive and non-parametric types of 3D structure, for example corals.

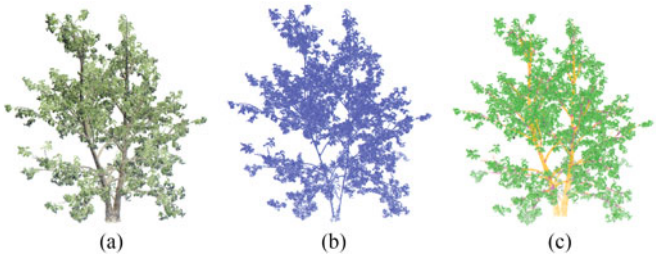


Fig. 1. Figure shows (a) the input point cloud, (b) its extracted meso-skeleton and (c) the segmentation result of the proposed algorithm.

scale across the point cloud. A reliable skeleton is extracted using the Deep Points Consolidation algorithm [8], with suitable modifications to handle complex-geometry. We use the Deep Points Consolidation algorithm as it computes a reliable skeleton in the present of noise and supports parallelization. This is advantageous when dealing with point clouds with a large number of points. A random forest classifier trained on labelled tree data in simulation is used to classify points into branches and leaves, as shown in Fig. 1. Finally as an application, quantitatively accurate metrics such as Leaf Area Index (LAI) are computed from the segmented foliage points. The method supports parallelization and is computationally efficient.

The main contributions of this letter are:

- An extension of the Deep Points Consolidation algorithm to handle complex geometry
- An automatic algorithm for segmenting branch structure and foliage

The proposed algorithm is evaluated quantitatively on simulated ground truth data.

The letter is organized as follows. A review of related work is presented in Section II. Section III presents the algorithm in detail. Results in simulation and on real data are presented in Section IV.

## II. Related Work

Procedural models were some of the early approaches used to model vegetation. These include rule based methods like L-systems [9]–[11] and statistical methods [12], [13], following a set of biologically inspired rules inferred from nature. While the output is of high quality, these models are primarily generative and do not represent a real reconstruction. A few recent approaches by [13] and [14] present ways by which this generation may be guided to resemble a real structure.

Image based approaches form another class of vegetation reconstruction techniques. Reche-Martinez *et al.* [15] use images to extract a volumetric representation of trees. Shlyakther *et al.* [16] construct a visual hull extracted from silhouettes of images of a tree and use a medial axis approximation for the branching structure. Neubert *et al.* [17] use images of trees and particle flow within an approximate volume representation to generate branches.

Three dimensional point cloud based approaches have, more recently, gained popularity due to the availability of affordable laser scanners, especially LIDAR sensors. Most of these techniques focus on extracting a branching structure from the point cloud. As this is relevant to our work, we review the approaches in detail. Xu *et al.* [18] present a semi-automatic approach relying on heuristics from tree allometry to model trees from range based scans. Wang *et al.* [19] cluster points of a tree based on distance from the root. A tree skeleton is then generated by fitting a circle to points in individual clusters projected onto a cross-sectional plane. The centres of such circles constitute nodes of the skeleton. Fitting cylinders to point clouds [1], [20]–[22] is another common method to extract branches. Livny *et al.* [23] construct a spanning tree to automatically extract the branching structure from point clouds. Though the results are promising, their approach doesn't scale when the input point cloud is dense. Aiteanu *et al.* [24] use principal curvatures to infer the skeleton representation. Mei *et al.* [25] use L1-median to extract a coarse skeleton and then propose the L1-minimum spanning tree algorithm to further refine the skeleton. These two approaches are able to handle both dense and sparse point clouds and reconstruct finer branches. Most of these approaches do not reconstruct the canopy, but generate it using procedural methods because the motivation is plausible appearance not quantitative accuracy. For example, [14] propose the idea of texture lobes to generate a visibly similar canopy.

Foliage reconstruction approaches have also been proposed where individual leaves are modelled explicitly. Quan *et al.* [26] employ a structure from motion based approach to reconstruct leaves from images. Nguyen *et al.* [27] use a custom structured light setup in their approach. Bradley *et al.* [28] and Chaurasia *et al.* [29] fit parametric leaf models to point clouds of foliage.

With recent advances in mobile robotics, there has been an increasing amount of work that uses optical devices to estimate vegetation metrics like LAI. These include both land based [30], [31] and airborne [32]. The voxel-base canopy profiling method proposed by Hosoi and Omasa [30] is gaining popularity [33], [34]. A recent work that is similar to the work presented in this letter is that of Li *et al.* [34] who also propose a segmentation of tree point cloud into branches and leaves and compute LAI on the leaf points. The segmentation is based on differences in normal orientations in local neighbourhoods. However, their segmentation requires manual post-processing. In this letter, we propose a fully automatic approach for segmentation that does not require manual input. We use a data-driven approach to train a segmentation classifier that performs well across multiple species of broadleaf trees.

## III. Algorithm

An overview of algorithm is presented in Fig. 2. Different modules of the pipeline are described in the following sections.

### A. Input

The input to the algorithm is a point cloud of the entire tree. The point cloud can be acquired by scanning using a regular camera, laser scanner or RGB-D camera, although associated operations like camera pose estimation or point cloud registration are more challenging with vegetation than with man-made structure.

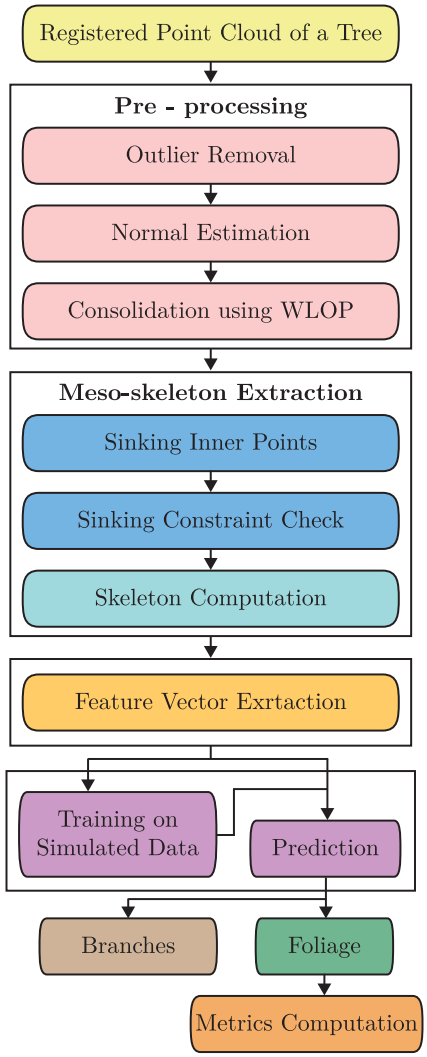


Fig. 2. An overview of the proposed algorithm.

Another challenge when working with vegetation data is that using manual thresholds or heuristics based on tree allometry do not scale well across different species of trees and plants. Hence there is a need for data-driven approaches. But these approaches require large quantities of labelled data. This is difficult to obtain from real world capture, because manual labelling would be difficult and time-consuming for even a single tree, and simulation data is the only practical option to provide training data. We modelled trees in SpeedTree [35] and imported them into a simulation environment built using the Unreal Engine [36]. Microsoft's AirSim [37] framework was used to model an RGB-D camera attached to a drone. Point cloud data, with ground truth labels for branches and foliage, was collected for each tree individually with the drone flying a helical path around the trees.

### *B. Point Cloud Noise Reduction and Outliers Removal*

As a first step, noisy isolated points are removed from the input cloud. A distance based threshold is used to detect them. Points whose mean distance to $k$-nearest neighbours is larger than a threshold are removed. If the input point cloud does not already contain information about the normals at each point (from the original scanning method), the normals are computed using Principal Component Analysis (PCA). A neighbourhood containing $n$ closest points is used to compute normals. The $180^{\circ}$ ambiguity in the direction of each computed normal is fixed so that the normal faces towards the viewpoint from which the corresponding point was observed.

Following this a point consolidation algorithm, the Weighted Locally Optimal Projection (WLOP) [38], is run on the filtered point cloud. This step involves downsampling the original cloud by random sub-sampling of the points of the original cloud and re-distributing them uniformly over the surface implicitly defined by the original point cloud. This step can be seen as a noise reduction step and helps in making the rest of algorithm robust to noisy input.

### *C. Meso-Skeleton Extraction*

The skeleton of the consolidated cloud is then computed using the Deep Points algorithm [8]. The output of this algorithm is a *meso-skeleton*, which is a set of two dimensional surfaces and one dimensional curves that represents the skeleton of a three dimensional object. In this approach, each point of the cloud is associated with an inner point that eventually forms the skeleton. The pair of surface point and its corresponding inner point are collectively referred to as a *Deep Point*. Positions of the inner points are first initialized to those of the corresponding surface points. The inner points are then moved iteratively in the direction away from the normal (*sinking*) until termination, to get a set of candidate skeleton points. Termination for an inner point is achieved if its local neighbourhood at any iteration step consists of inner points whose normal orientations differ greatly from its own normal orientation. Once all the inner points reach termination, or if a chosen maximum number of iterations is exceeded, a joint optimization further refines the positions of the inner points. The objective of this refinement is to move each inner point such that it is close to the L1 median of its neighbouring inner points and that the set of inner points is uniformly distributed. Finally using the points of the meso-skeleton as anchor points, another joint optimization is performed over the surface points, with similar objectives as the previous step along with an additional constraint that the deep points orientation is along the surface normal.

The above algorithm works well for extracting the meso-skeleton of a wide variety of closed 3D objects. A skeleton extracted from a branching coral like structure is also shown in [8]. However, The original algorithm applies only to closed surfaces, not to open surfaces like leaves or branches observed from one side only, and it will produce incorrect skeletons as shown in Fig. 3(a). This is fairly common in regions surrounding leaves, as they tend to occlude the finer branches like twigs.

We extend the Deep Points algorithm, to retain its advantages while also supporting open surfaces. A check is performed on the termination status of inner points after a sufficiently large number of sinking iterations. As described earlier, branches and the trunk enclose space in their interior and hence inner points must reach termination after a fixed number of iterations. Any

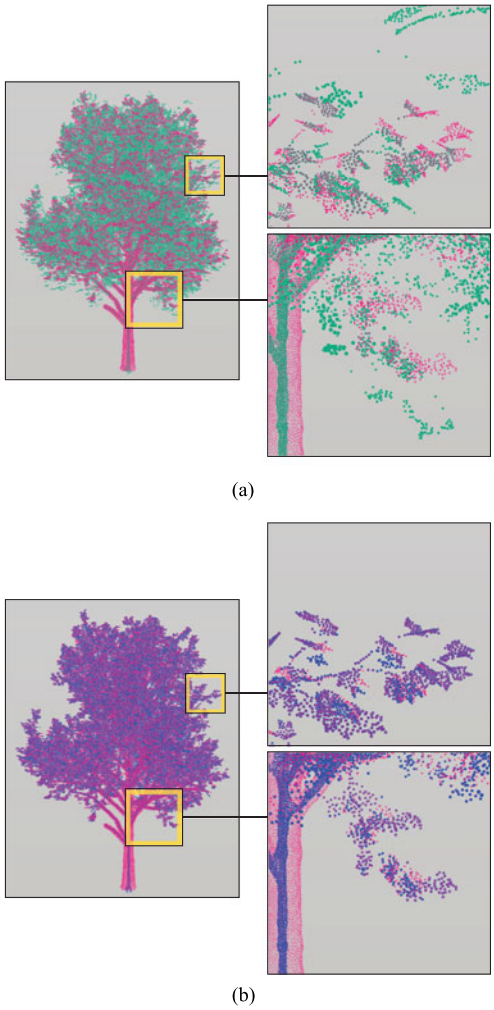


Fig. 3. Figure shows (a) the issue with sinking of inner points computed by the unmodified Deep Points algorithm. The inner points (in green), move far away from the surface points (in red), in leafy regions; (b) the inner points (in blue) after applying the sinking criteria and repositioning inner points, that are still moving, to their corresponding surface points (in red).

**Algorithm 1:** Feature Vector Computation.

**Initialize:**
Choose number of radii $n$
Choose number of relevant radii $n_p$
Choose radius step $r_s$
Construct radii vector $\mathbf{r}$ of size $n$ in steps of size $r_s$ in some chosen range $[r_{\min}, r_{\max}]$, $\mathbf{r} \leftarrow [r_{\min}, r_{\min} + r_s, \ldots, r_{\max}]$, where $r_{\max} = r_{\min} + (n-1) \cdot r_s$

**for all** $p$ in the cloud **do**
  $\kappa_{1,p} \leftarrow$ zero vector of size $n$
  $\kappa_{2,p} \leftarrow$ zero vector of size $n$
  Compute $d_p \leftarrow$ distance to skeleton
  Compute $r_{d,p} \leftarrow \arg\min_{r \in \mathbf{r}} \|d_p - r\|_2$
  $r_{d,p} \leftarrow \min(\max(r_{\min} + n_p \cdot r_s, r_{d,p}, r_{\max} - n_p \cdot r_s)$
  Compute vector of relevant radii,
    $\mathbf{r}_p \leftarrow [r_{d,p} - n_p \cdot r_s, \ldots, r_{d,p}, \ldots, r_{d,p} + n_p \cdot r_s]$
  Initialize:
  $c_1 \leftarrow$ zero vector of length same as that of $\mathbf{r}_p$
  $c_2 \leftarrow$ zero vector of length same as that of $\mathbf{r}_p$
  **for all** $r \in \mathbf{r}_p$ **do**
    Compute $c_1[r] \leftarrow$ curvature at $p$ in a ngbd. of size r
    Compute $c_2[r] \leftarrow$ curvature at $p$ in a ngbd. of size r
  **end for**

  **for all** $r \in \mathbf{r}$ **do**
    **if** $r \in \mathbf{r}_p$ **then**
      $\kappa_{1,p}[r] \leftarrow c_1[r]$
      $\kappa_{2,p}[r] \leftarrow c_2[r]$
    **else**
      continue
    **end if**
  **end for**
  Concatenate $\mathbf{f}_p \leftarrow [\kappa_{1,p}, \kappa_{2,p}]$
  **Output:** $\mathbf{f}_p$
**end for**

points that are still moving after these number of iterations are either leaves or occluded twigs. Such inner points are identified and re-positioned to their respective surface points. They are further also omitted from the skeleton extraction step, i.e the joint optimization step. In other words, the skeleton for these surface points are the points themselves. The reason for omitting the non-terminating inner points from the optimization step is because we are not looking for the mid-rib of a leaf when its skeleton is computed but the surface of the leaf itself. This might seem counter intuitive, but since our primary goal of extracting the skeleton is to compute the distance of surface points from the skeleton, such a construction makes sense.

### D. Feature Extraction

Surface curvatures are used as features to describe every point of the point cloud. Curvatures are local features which depend on the neighbourhood in which they are computed. As point clouds of vegetation typically contain objects of multiple scales, choosing an appropriate neighbourhood is important.

We build on the output of the modified Deep Points algorithm, and use spherical neighbourhoods whose radii relate to the distance between the point at which the curvature has to be estimated and its associated skeleton point. For a point on a branch, the skeleton distance corresponds to the radius of the branch at that point, which is a natural way to define the neighbourhood for that structure. On the other hand, since individual leaves are small in size, a relatively smaller neighbourhood should be chosen to compute a meaningful curvature value. This is realized because, as described in the previous section, the skeleton for a leaf is the surface of the leaf itself.

Computing curvatures in different neighbourhood sizes for every point, as above, poses another challenge i.e. of comparing the curvatures between two points. In order to address this, we propose the following feature vector. The feature vector for every point $p$ is of fixed length $n$, and initialized to a zero

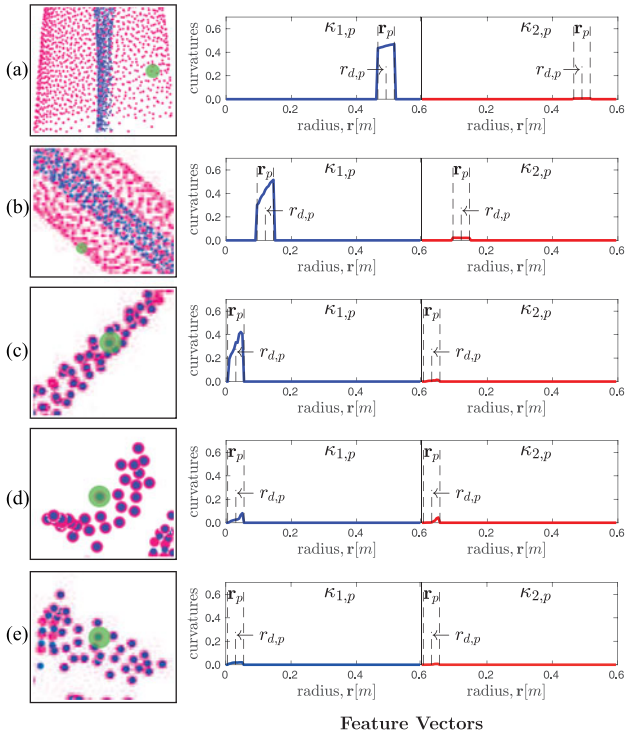


Fig. 4. Figure shows different regions of the tree point cloud on the left; (a) trunk, (b) branch, (c) twig, (d) curved leaf, (e) flat leaf. Surface points are in red and extracted meso-skeleton points are in blue. Images on the right show the feature vector computed at the highlighted point (green). Points in the trunk, branch and twigs are cylindrical and hence have a strong $\kappa_1$ but almost zero $\kappa_2$. On the other hand, for leaves $\kappa_1$ and $\kappa_2$ are both small or close to zero. $r_{d,p}$ is the quantized distance to the skeleton point and $\mathbf{r}_p$ are the valid radii in which curvatures are computed.

vector. Each element of the vector corresponds to the curvature extracted in a neighbourhood size (radius) in increasing order of radii. The set of radii $\mathbf{r}$ are chosen in some range $[r_{\min}, r_{\max}]$. Computing curvatures at all the radii is computationally expensive and at the same time larger neighbourhoods result in incorrect values at leaf points if multiple leaves fall within the neighbourhood. Hence, curvatures are computed only for a relevant subset of radii $\mathbf{r}_p$ for point $p$. This subset is a window of size $2n_p$ around the skeleton distance $d_p$ of the point $p$. We compute the curvature vectors along both the Principal directions in the neighbourhood and concatenate the two vectors to form a single feature vector. This is summarized in Algorithm 1.

Fig. 4 shows the feature vectors for points belonging to different regions of the point cloud. These points are all real data and are selected as typical examples of their corresponding class. It can be seen that for branch points, there is a strong curvature only along one of the principal directions. On the other hand for leaf points, there is either close to zero curvature along both the directions for flat leaves or a small but prominent curvature along both directions for curved leaves.

### E. Segmentation

Segmentation was done using a random forest classifier with 200 decision trees. This value was chosen by cross validation. A higher number of decision trees did not improve accuracy. The

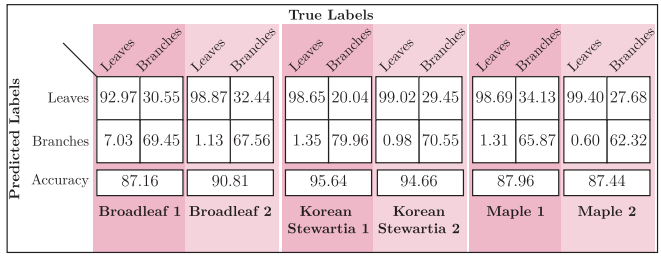


Fig. 5. Confusion matrices and classification accuracies (in percentages) for each tree species. The classifier for each species was trained on a separate tree belonging to the same species.

classifier was trained, on the above mentioned feature vectors, with ground truth labels for each point obtained from simulation.

## IV. Results

Segmentation results are shown (a) on simulated tree data, in order to allow evaluation against ground truth and (b) on real data, in which case it is not feasible to obtain ground truth but an assessment of the algorithm is possible by visual inspection.

### A. Implementation Details

A feature vector of length 240 was chosen, with $\kappa_1$ and $\kappa_2$ of length 120 each. The neighbourhood sizes ranged from 5 mm ($r_{\min}$) to 60 cm ($r_{\max}$) with increments of 5 mm ($r_s$). These values were chosen to accommodate all tree structures ranging from small leaves to large trunks and kept constant across all species. The number of relevant radii ($n_p$) was also empirically chosen to be 11 corresponding to a spread of 5 cm around the computed distance to the skeleton.

A simulated Kinect One sensor, following a spiral trajectory around the target tree and maintaining an approximate distance of 2 m from it, was used to collect data. Depth images (70° FOV, 512 × 424 resolution) were collected at intervals of 10° and the corresponding point clouds were aggregated to get the final cloud. On average there were around 40 points per leaf.

The proposed algorithm was implemented in C++ but the code is not optimized. On average, for a point cloud with a million points, the entire pipeline takes around 30 minutes on a 3.20 GHz Intel Core$^{\text{TM}}$ i7-3930K CPU. We present these values only to give the reader an idea of the runtime. An offline algorithm is suitable for our applications, as mentioned in the Introduction.

### B. Trees of Different Species

Experiments in simulation were performed using three different species of broadleaf trees Korean Stewartia, Maple and a generic Broadleaf tree. For each of the three species, three trees were generated using Speedtree. One tree was used for training the segmentation classifier specific to that species and two trees were used to evaluate the classifier. Classification accuracies, along with their confusion matrices are shown in Fig. 5. Fig. 6 shows a visualization of these results for one tree of each species. Points in orange in the second and third rows are

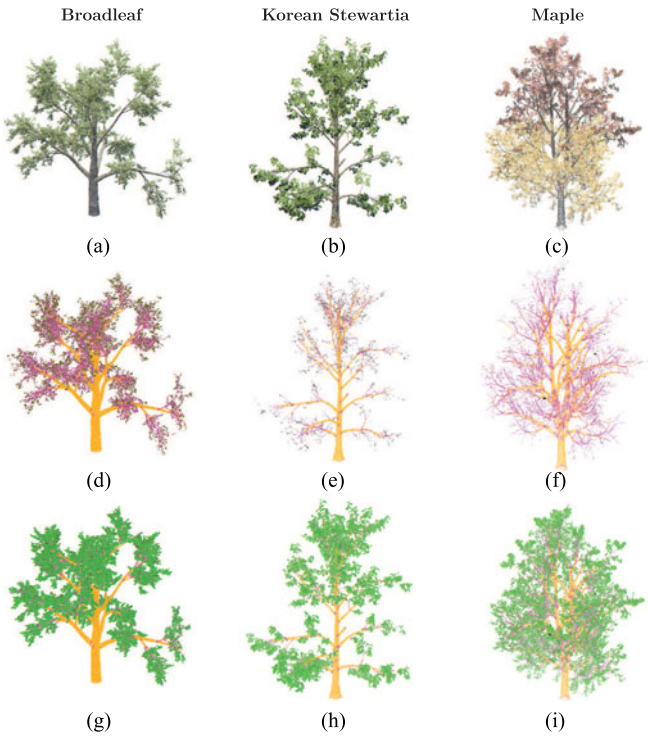


Fig. 6. Figure showing classification results on trees of different species. The input clouds are shown in Figures (a, b, c). Figures (d, e, f) show the correctly labelled branch points in orange, incorrectly labelled branch points in magenta and incorrectly labelled leaf points in black. Figures (g, h, i) show the correctly labelled leaf points in green.

| Classifier Used | Broadleaf 1 | Broadleaf 2 | Korean Stewartia 1 | Korean Stewartia 2 | Maple 1 | Maple 2 |
|---|---|---|---|---|---|---|
| Broadleaf | 87.16 | 90.81 | 92.08 | 89.97 | 83.29 | 88.29 |
| Korean Stewartia | 70.21 | 84.20 | 95.64 | 94.66 | 64.92 | 86.49 |
| Maple | 85.58 | 85.394 | 90.72 | 88.33 | 87.96 | 87.44 |
| Combined | 89.29 | 90.09 | 95.22 | 94.32 | 87.51 | 89.37 |

Accuracy

Fig. 7. Table shows percentage accuracy of segmentation using classifiers across different species. The combined classifier was trained with one tree from each species.

branch points correctly classified as branch points. Green points are leaf points correctly classified as leaf points, which the classifier predicts with very high (sometimes near perfect) accuracy. Points in magenta are branch points incorrectly classified as leaf points. This often happens in the twig regions close to leaves where the branches are thin and there is a lot of occlusion. The final cluster of black points are leaf points incorrectly predicted as branches. These are almost negligible and generally occur at the tips of leaves which are incorrectly identified as ends of twigs.

Cross-species prediction accuracies were also calculated to evaluate differences across different species of trees. These are presented in Fig. 7. Furthermore, a combined segmentation classifier was trained, using one generated tree from each species, and then evaluated on the remaining six trees.


Fig. 8. Figure shows (a) the input broadleaf tree with no leaves, (b) the segmentation output. Here the magenta points are incorrectly classified as leaves.

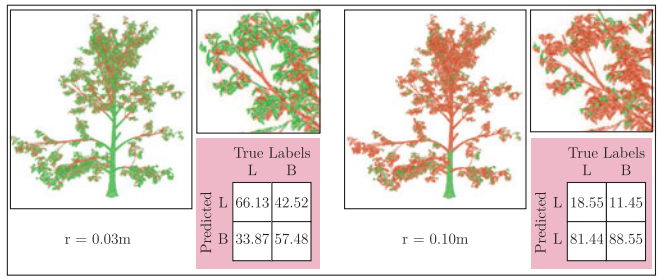


Fig. 9. Figure shows the output of the DiffN approach from [34] for two different neighbourhood sizes. The threshold $\theta$ was fixed at 0.5. Green points are predicted leaf points and brown are predicted branch points.

One can see that the combined classifier performs well despite the individual trees having different geometric characteristics like branch thicknesses, leaf spread, etc. For any given species, its accuracy is on par with the species specific classifiers for that species.

Additionally for the generic Broadleaf species, we also evaluated the performance on a tree with no leaves. Classification accuracy was 73% which is slightly better than the branch class accuracy achieved for trees with leaves. One can notice in Fig. 8 that it is the twigs that are misclassified.

### C. Comparison With State-of-the-art

We compare our results with the Difference of Normal Orientations (DiffN) method presented in [34]. For every point $p$, DiffN computes average difference of unit normals $\hat{n}$, in *fixed* spherical neighbourhoods $\mathcal{N}_{r,p}$, of size $r$, around the point. If the absolute value of this difference is smaller than a chosen threshold $\theta$ (implying flat surfaces) the point is classified as a leaf. Otherwise it is a branch. This is summarized in the following equation. For every point $p$

$$\text{label}(p) = \begin{cases} \text{leaf}, & \frac{1}{|\mathcal{N}_{r,p}|} \left| \sum_{\forall q \in \mathcal{N}_{r,p}} (\hat{n}(p) - \hat{n}(q)) \right| < \theta \\ \text{branch}, & \text{otherwise} \end{cases} \tag{1}$$

It is immediately evident that $r$ plays a key role. A small $r$ captures the differences between leaves and thin branches well but labels the trunk incorrectly as it would appear flat. A large $r$ characterizes the trunk well but is not meaningful for leaves as the neighbourhood may contain multiple leaves (Fig. 9). For a fair comparison to our approach, for each species, we calculate

| | | Broadleaf 1 r = 0.054m | | Broadleaf 2 r = 0.059m | | Korean Stewartia 1 r = 0.040m | | Korean Stewartia 2 r = 0.034m | | Maple 1 r = 0.045m | | Maple 2 r = 0.040m | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | True Labels | | | | | | | | | | | |
| | | L | B | L | B | L | B | L | B | L | B | L | B |
| Li et al. [34] Predicted | L | 42.75 | 63.06 | 55.21 | 70.24 | 77.39 | 77.79 | 62.01 | 52.11 | 59.48 | 58.43 | 60.70 | 65.19 |
| | B | 57.25 | 36.94 | 44.79 | 29.76 | 22.61 | 22.21 | 37.99 | 47.89 | 40.52 | 41.57 | 39.30 | 34.81 |
| Accuracy | | | | | | | | | | | | | |
| Li et al. [34] | | 41.58 | | 48.99 | | 76.53 | | 59.74 | | 56.20 | | 54.92 | |
| Ours | | 87.16 | | 90.81 | | 95.64 | | 94.66 | | 87.96 | | 87.44 | |

Fig. 10. Figure shows segmentation results of the DiffN approach on the same trees as in Fig. 5. We see that our proposed algorithm outperforms DiffN for all the species.


Fig. 11. Figure shows the segmentation results on real tree data collected using a LIDAR [24]. (a) input point cloud, (b) segmented branch points, (c) leaf points in green and branch points in black.

the mean of the radius ($r_{d,p}$) of all points, computed during feature extraction and use that as the fixed neighbourhood size $r$. The threshold $\theta$, was fixed at 0.5. The results are summarized in Fig. 10. A point to note is that because the number of leaf points for every tree is much larger than the number of branch points, the average radius computed above is biased towards leaves. Hence the per class accuracy for branches is very poor. However, our approach outperforms this method as it adaptively computes the neighbourhood size for every point.

### D. Noisy Input

To test robustness to noisy input, Gaussian noise was added to both the sensor itself and also to the pose of the camera during capture. The RGBD sensor was modelled as a Kinect One sensor with a Gaussian noise of zero mean and standard deviation of 3 mm along the axial direction. This noise was added to every depth frame acquired from the sensor. Further Gaussian noise of zero mean and 0.5 cm was added to each direction of the translation vector to simulate misalignments during registration that occur during real world capture. This evaluation was done on the Korean Stewartia 1 tree. The species specific classifier gave an accuracy of $85.41\%$ and the combined classifier gave an accuracy of $85.61\%$. The decrease in accuracy was largely due to misclassification of the branch points as leaves.

### E. Testing on Real Data

We tested the segmentation framework on real data of a Beech tree acquired from multiple Lidar scans [24]. Fig. 11 shows the results of segmentation on this cloud. The combined classifier trained on all the three species in simulation was used to segment the data.

| Leaf Area Index | Broadleaf 1 | Broadleaf 2 | Korean Stewartia 1 | Korean Stewartia 2 | Maple 1 | Maple 2 |
|---|---|---|---|---|---|---|
| True | 1.98 | 1.50 | 4.32 | 1.35 | 3.11 | 3.73 |
| Predicted | 2.02 | 1.59 | 4.33 | 1.40 | 3.35 | 4.10 |

Fig. 12. Figure shows a comparison between LAI computed from ground truth versus the LAI computed from the segmented leaf points.

### F. Leaf Area Index

As an application we show the computation of one commonly used vegetation metric, the Leaf Area Index which can be estimated directly using the segmented leaves. We calculate LAI using the Voxel-based Canopy Profiling method (VCP) from [30]. The point cloud of leaves is voxelized using a voxel grid of resolution 5mm. This size is chosen as described in [34]. These voxels are projected vertically onto the ground plane and the convex hull of this projection is computed. This hull represents the canopy coverage area of the tree. This hull is projected back into the voxel grid vertically and represents the total volume of the tree. This volume is split horizontally into layers and leaf area density (LAD) is computed in each horizontal layer as follows.

$$\mathrm{LAD}(h, \Delta h) = \alpha \frac{1}{\Delta h} \sum_{i \in [h, h+\Delta h)} \frac{n_{l,i}}{n_{l,i} + n_{e,i}} \quad , \qquad (2)$$

where $h$ is the height at which a horizontal layer starts, $\Delta h$ is the layer thickness, $i$ is the index of the voxel layer, $n_{l,i}$ are the number of voxels in layer $i$ that are occupied by leaf points and $n_{e,i}$ are the number of voxels that are empty in the same layer. We drop the correction factor from [30] since in all our experiments, the camera was fairly orthogonal to the plane of the leaves. Leaf Area Index is then calculated by just summing up the individual LAD of each layer times the layer height.

On simulated data we compared the LAI calculated from the predicted leaf point cloud with the *true* LAI of the tree, which was computed by applying VCP directly on the leaf mesh used to generate the tree model. Fig. 12 presents this comparison. We see that the predicted LAI is a bit higher than the true LAI due to overestimating leaf points.

## V. Conclusion

This letter has described a method to segment a 3D point cloud of vegetation such as a tree or woody shrub to create a hybrid model comprised of branch skeleton and segmented foliage. Avoiding the introduction of parametric models for organic structure found in previous work, which are inevitably imprecise for complex geometry like vegetation. Improved accuracy is valuable for human interaction with the data such as visualization, for automatic analysis like computation of the LAI, and for robotic deployment.

While the accuracy in segmentation is high, finer branches and twigs are still incorrectly classified as leaves. A possible way to address this issue could be to perform an additional processing step that would extract linear structures in the leaf clusters and re-classify them as twigs. More sophisticated techniques that infer about the present of twigs close to leaves may also be employed.

## REFERENCES


[1] H. Medeiros *et al.*, "Modeling dormant fruit trees for agricultural automation," *J. Field Robot.*, vol. 34, pp. 1203–1224, 2017.

[2] S. Bargoti, J. P. Underwood, J. I. Nieto, and S. Sukkarieh, "A pipeline for trunk detection in trellis structured apple orchards," *J. Field Robot.*, vol. 32, no. 8, pp. 1075–1094, 2015.

[3] V. Kankare *et al.*, "Accuracy in estimation of timber assortments and stem distribution–a comparison of airborne and terrestrial laser scanning techniques," *ISPRS J. Photogramm. Remote Sens.*, vol. 97, pp. 89–97, 2014.

[4] Wikipedia, "Leaf area index," 2008. [Online]. Available: https://en.wikipedia.org/wiki/Leaf_area_index. Accessed on: Jun. 18, 2018.

[5] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *Int. J. Robot. Res*, vol. 32, pp. 1231–1237, 2013.

[6] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, "ElasticFusion: Real-time dense SLAM and light source estimation." *Int. J. Robot. Res.*, vol. 35, no. 14, pp. 1697–1716, 2016.

[7] R. A. Newcombe, D. Fox, and S. M. Seitz, "Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 343–352.

[8] S. Wu, H. Huang, M. Gong, M. Zwicker, and D. Cohen-Or, "Deep points consolidation," *ACM Trans. Graph.*, vol. 34, no. 6, p. 176, 2015.

[9] A. Lindenmayer, "Mathematical models for cellular interactions in development i. filaments with one-sided inputs," *J. Theoretical Biol.*, vol. 18, no. 3, pp. 280–299, 1968.

[10] P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*. New York, NY, USA: Springer, 1990.

[11] P. Prusinkiewicz, M. James, and R. Měch, "Synthetic topiary," in *Proc. 21st Annu. Conf. Comput. Graph. Interactive Tech.*, 1994, pp. 351–358.

[12] O. Deussen and B. Lintermann, *Digital Design of Nature: Computer Generated Plants and Organics*. New York, NY, USA: Springer, 2006.

[13] J. O. Talton, Y. Lou, S. Lesser, J. Duke, R. Měch, and V. Koltun, "Metropolis procedural modeling," *ACM Trans. Graph.,* vol. 30, no. 2, p. 11, 2011.

[14] Y. Livny *et al.*, "Texture-lobes for Tree Modelling," *J. ACM Trans. Graph.*, vol. 30, no. 4, 2011.

[15] A. Reche-Martinez, I. Martin, and G. Drettakis, "Volumetric reconstruction and interactive rendering of trees from photographs," *J. ACM Trans. Graph*, vol. 23, no. 3, pp. 720–727, 2004.

[16] I. Shlyakhter, M. Rozenoer, J. Dorsey, and S. Teller, "Reconstructing 3d tree models from instrumented photographs," *IEEE Comput. Graph. Appl.*, vol. 21, no. 3, pp. 53–61, May/Jun. 2001.

[17] B. Neubert, T. Franken, and O. Deussen, "Approximate image-based tree-modeling using particle flows," *ACM Trans. Graph.*, vol. 26, no. 3, p. 88, 2007.

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

[19] Y. Wang *et al.*, "Tree branching reconstruction from unilateral point clouds," in *Transactions on. Edutainment VIII*. New York, NY, USA: Springer, 2012, pp. 250–263.

[20] N. Pfeifer *et al.*, "Automatic reconstruction of single trees from terrestrial laser scanner data," in *Proc. 20th Int. Soc. Photogramm. Remote Sensing Congress*, 2004, pp. 114–119.

[21] J. Binney and G. S. Sukhatme, "3d tree reconstruction from laser range data," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2009, pp. 1321–1326.

[22] D.-M. 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 *Proc. IEEE 11th Int. Conf. Comput.-Aided Des. Comput. Graph.,*, 2009, pp. 572–575.

[23] Y. Livny, F. Yan, M. Olson, B. Chen, H. Zhang, and J. El-Sana, "Automatic reconstruction of tree skeletal structures from point clouds," *ACM Trans. Graph.*, vol. 29, no. 6, p. 151, 2010.

[24] F. Aiteanu and R. Klein, "Hybrid tree reconstruction from inhomogeneous point clouds," *Vis. Comput.*, vol. 30, no. 6–8, pp. 763–771, 2014.

[25] J. Mei, L. Zhang, S. Wu, Z. Wang, and L. Zhang, "3d tree modeling from incomplete point clouds via optimization and l 1-MST," *Int. J. Geograph. Inf. Sci.*, vol. 31, no. 5, pp. 999–1021, 2017.

[26] L. Quan, P. Tan, G. Zeng, L. Yuan, J. Wang, and S. B. Kang, "Image-based plant modeling," *J. ACM Trans. Graph.*, vol. 25, no. 3, pp. 599–604, 2006.

[27] T. T. Nguyen, D. C. Slaughter, N. Max, J. N. Maloof, and N. Sinha, "Structured light-based 3d reconstruction system for plants," *Sensors*, vol. 15, no. 8, pp. 18 587–18 612, 2015.

[28] D. Bradley, D. Nowrouzezahrai, and P. Beardsley, "Image-based reconstruction and synthesis of dense foliage," *ACM Trans. Graph.*, vol. 32, no. 4, p. 74, 2013.

[29] G. Chaurasia and P. Beardsley, "Editable parametric dense foliage from 3d capture," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5305–5314.

[30] F. Hosoi and K. Omasa, "Estimating vertical leaf area density profiles of tree canopies using three-dimensional portable lidar imaging," in *Proc. ISPRS Workshop Laser-Scanning*, 2009, vol. 9, pp. 152–157.

[31] M. Béland, J.-L. Widlowski, and R. A. Fournier, "A model for deriving voxel-level tree leaf area density estimates from ground-based lidar," *Environ. Model. Softw.*, vol. 51, pp. 184–189, 2014.

[32] W. Li *et al.*, "Generating pseudo large footprint waveforms from small footprint full-waveform airborne lidar data for the layered retrieval of lai in orchards," *Opt. Express*, vol. 24, no. 9, pp. 10 142–10 156, 2016.

[33] F. Hosoi, Y. Nakai, and K. Omasa, "3-D voxel-based solid modeling of a broad-leaved tree for accurate volume estimation using portable scanning lidar," *ISPRS J. Photogramm. Remote Sens.*, vol. 82, pp. 41–48, 2013.

[34] S. Li, L. Dai, H. Wang, Y. Wang, Z. He, and S. Lin, "Estimating leaf area density of individual trees using the point cloud segmentation of terrestrial lidar data and a voxel-based model," *Remote Sens.*, vol. 9, no. 11, p. 1202, 2017.

[35] "Speedtree," 2008. [Online]. Available: https://store.speedtree.com/. Accessed on: Jun. 18, 2018.

[36] E. Games, "Unreal engine 4," 2018. [Online]. Available: https://www.unrealengine.com/en-US/what-is-unreal-engine-4. Accessed on : Jun. 18, 2018.

[37] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Proc. Field Service Robot.*, 2017. [Online]. Available: https://arxiv.org/abs/1705.05065

[38] H. Huang, D. Li, H. Zhang, U. Ascher, and D. Cohen-Or, "Consolidation of unorganized point clouds for surface reconstruction,"*J. ACM Trans. Graph.*, vol. 28, no. 5, p. 176, 2009.