Machine Learning Approaches to Improve Three Basic Plant Phenotyping Tasks Using Three-Dimensional Point Clouds[1][OPEN]

Illia Ziamtsov and Saket Navlakha[2,3]

The Salk Institute for Biological Studies, Integrative Biology Laboratory, La Jolla, California 92037

ORCID ID: 0000-0002-5505-9718 (S.N.).

Developing automated methods to efficiently process large volumes of point cloud data remains a challenge for three-dimensional (3D) plant phenotyping applications. Here, we describe the development of machine learning methods to tackle three primary challenges in plant phenotyping: lamina/stem classification, lamina counting, and stem skeletonization. For classification, we assessed and validated the accuracy of our methods on a dataset of 54 3D shoot architectures, representing multiple growth conditions and developmental time points for two Solanaceous species, tomato (*Solanum lycopersicum cv 75 m82D*) and *Nicotiana benthamiana*. Using deep learning, we classified lamina versus stems with 97.8% accuracy. Critically, we also demonstrated the robustness of our method to growth conditions and species that have not been trained on, which is important in practical applications but is often untested. For lamina counting, we developed an enhanced region-growing algorithm to reduce oversegmentation; this method achieved 86.6% accuracy, outperforming prior methods developed for this problem. Finally, for stem skeletonization, we developed an enhanced tip detection technique, which ran an order of magnitude faster and generated more precise skeleton architectures than prior methods. Overall, our improvements enable higher throughput and accurate extraction of phenotypic properties from 3D point cloud data.

Emerging technologies for high-throughput plant architecture capture has increased the demand for automated phenotyping methods (Furbank and Tester, 2011; Fiorani and Schurr, 2013). Extracting plant features is a first step toward quantifying biomass and yield (Mathan et al., 2016), assessing flowering time and drought tolerance (Minervini et al., 2016), mapping genotypes to phenotypes (Balduzzi et al., 2017; Setter, 2012), and studying morphological properties of plant architectures (Conn et al., 2017a; Conn et al., 2017b; Bucksch et al., 2017; Li et al., 2018; Prusinkiewicz and Lindenmayer, 1996). Common phenotyping features of interest include the number, size, and shape of leaves (Wilf et al., 2016; Huang et al., 2018); plant height and growth rates (Madec et al., 2017); and branch lengths, diameters, and angles (Bucksch et al., 2017); among others.

There are many techniques used for noninvasive plant phenotyping, ranging from camera-based imaging (Heckwolf et al., 2015; Nguyen et al., 2015) to three-dimensional (3D) laser scanning (Paulus et al., 2013; Paulus et al., 2014), with advantages and disadvantages to each in terms of cost, accuracy, and throughput. For example, two-dimensional (2D) imaging methods remain a cheap, high-throughput solution (Perez-Sanz et al., 2017; Ubbens and Stavness, 2017; Gao et al., 2018; Zhu et al., 2018) and can be used in conjunction with structure from motion techniques (Szeliski, 2010) to generate point clouds (Dey et al., 2012; Santos et al., 2015; Gélard et al., 2017b) or mesh (Moraes Frasson and Krajewski, 2010; Paproki et al., 2012) representations of plants, the latter providing additional areal and volumetric information. This approach, however, faces several issues, including camera calibration, image registration, object occlusion, and inconsistencies in scaling and illumination. Images that contain an extra depth component (called RGB-D) have also been used in the context of plant phenotyping that helps overcome some of these challenges (Chene et al., 2012; Xia et al., 2015). Further, some platform-specific solutions have been proposed (Nguyen et al., 2016; Gibbs et al., 2018), although general-purpose solutions remain elusive.

On the other hand, light detection and ranging (LiDAR) and 3D laser scanning devices are considerably more expensive but generate point cloud representations of plant geometry with up to micron-level precision. When compared with most 2D methods, 3D can provide a more comprehensive and detailed view of the entire architecture. Prior works have used 3D scanning

[1]This work was supported by the Pew Charitable Trusts, the National Science Foundation under award CAREER DBI-1846554, and the National Institutes of Health National Institute on Deafness and Other Communication Disorders under award number 1R01DC017695 (to S.N.).

[2]Author for contact: navlakha@salk.edu.

[3]Senior author.

The author responsible for distribution of materials integral to the findings presented in this article in accordance with the policy described in the Instructions for Authors (www.plantphysiol.org) is: Saket Navlakha (navlakha@salk.edu).

I.Z. and S.N. conceived the study and wrote the paper; I.Z. performed the research.

[OPEN]Articles can be viewed without a subscription.
www.plantphysiol.org/cgi/doi/10.1104/pp.19.00524

to study, for example, growth and development (Li et al., 2013) and light interception of leaves (Gaëtan et al., 2012), and these technologies are now being deployed in the field (Sun et al., 2018; Andújar et al., 2013). However, these techniques also face challenges, such as nonuniform sampling of points, outliers and missing data (Li et al., 2013), scalability, and automation (Chaudhury et al., 2018). Thus, developing robust computational methods to study plant shapes from 3D point cloud data has become increasingly important.

Here, we study three basic computational problems that are essential for downstream high-throughput phenotyping: classification of point cloud data into sets of stem and lamina points, lamina counting, and stem skeletonization (Fig. 1). We test our methods on a dataset of 54 3D laser-scanned shoot architectures from two dicot species (tomato [*Solanum lycopersicum*], *Nicotiana benthamiana*), each grown across three developmental time points, and five growth conditions (ambient, high-heat, shade, drought, high-light). Prior approaches to point cloud classification in context of phenotyping included both supervised (Paulus et al., 2013) and unsupervised (Wahabzada et al., 2015) machine learning methods, and hand-crafted methods specialized toward the geometry of particular species (Dey et al., 2012; Gélard et al., 2017b). For lamina counting, we extended a region growing algorithm to improve the robustness of lamina segmentation to natural variation, including ripples, wrinkles, and hairs. For stem skeletonization, our method improves

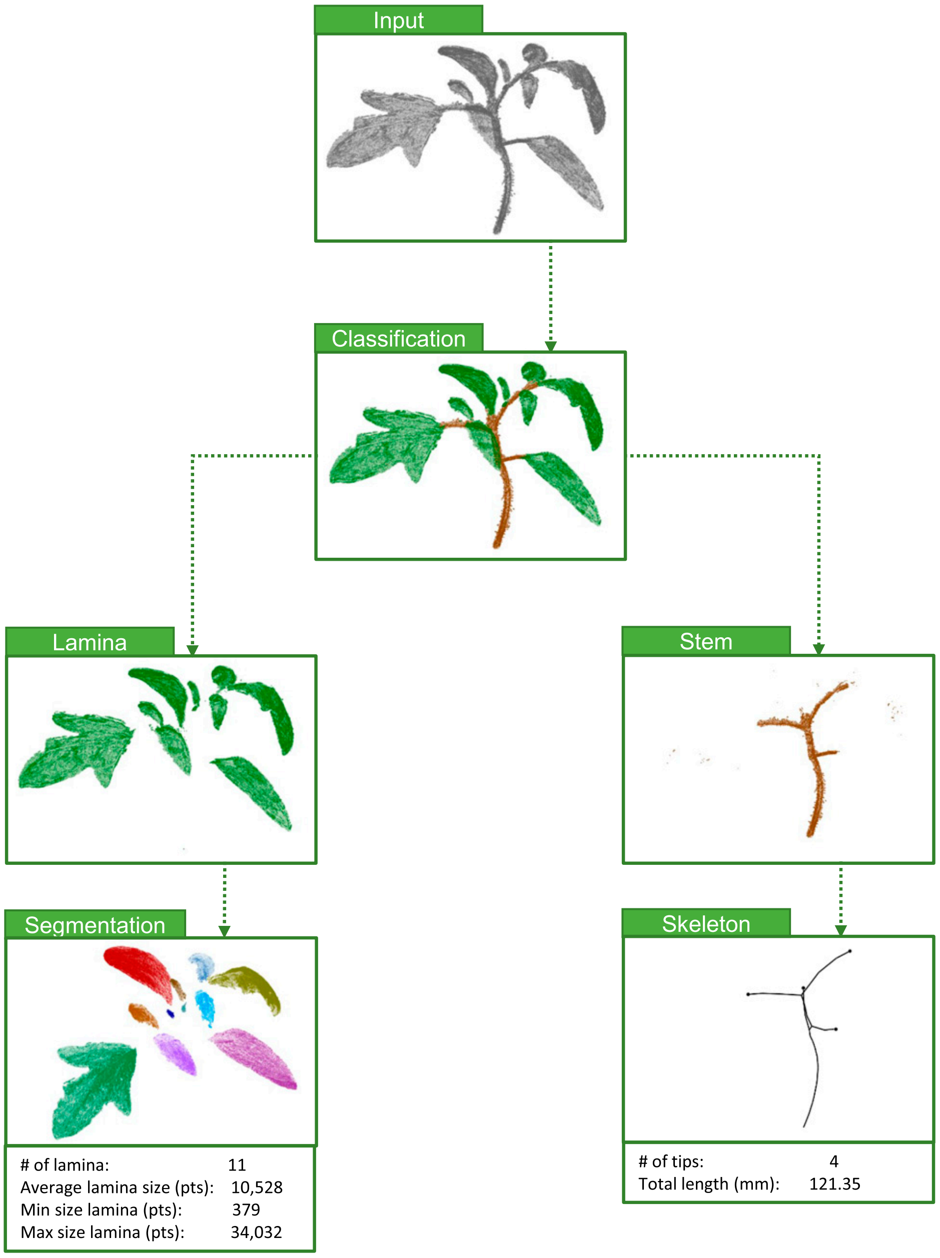

Input
Classification
Lamina
Stem
Segmentation
Skeleton
# of lamina: 11
Average lamina size (pts): 10,528
Min size lamina (pts): 379
Max size lamina (pts): 34,032
# of tips: 4
Total length (mm): 121.35

Figure 1. Overview. A summary of the three problems tackled: classification of lamina versus stem points, lamina counting/segmentation, and stem skeletonization.

tip and root detection to generate more precise skeletons, and we improve run-time by an order of magnitude. Finally, we combine these three pipelines and group lamina, petioles, and petiolules into individual tomato leaves. Our algorithms are open-source and freely available for the community.

RATIONALE

Dataset of 3D Plant Shoot Architectures

We used 3D laser scanning to generate point cloud representations of 54 full shoot architectures. We used the FaroArm EDGE model 14000, which is a noninvasive laser scanner and provides micron-level resolution. We scanned plants from two species: 36 tomato plants (*Solanum lycopersicum cv m82D*) and 18 *Nicotiana benthamiana* plants. Tomato plants were grown in five conditions (control/ambient light, high-heat, high-light, shade, and drought), and *Nicotiana benthamiana* plants were grown in three conditions (control, high-heat, and shade). There were two replicates per species-conditions pair. These conditions were selected because they represent a range of realistic environments regularly faced by many plants, and for which there is phenotypic plasticity that challenges accurate phenotyping. Each plant was scanned on developmental days 5, 12, and 20. Each of the 54 architectures was manually classified into sets of stem and lamina points. Full experimental details were previously described (Conn et al., 2017a; Conn et al., 2017b).

This dataset encapsulates a diverse set of architectures, with variation in many phenotypic features, including plant size (biomass), number and structure of lamina, and stem lengths and angles. For example, the plant volume, measured by the convex hull of the plant's cloud points, of day 20 tomato plants varied across conditions by up to 50-fold: from 13.78 cm^3 in high-heat to 668.68 cm^3 in high-light (Conn et al., 2017b). The number of lamina for *Nicotiana benthamiana* plants varied from 4 to 12 from the young to old plants (Conn et al., 2017a). Over all 36 tomato plants, the range in number of cloud points was from 22,411 to 958,991, with an average of 233,917 cloud points per scanned architecture. Similarly, over all 18 *Nicotiana benthamiana* plants, the range was from 3,709 to 806,269 cloud points. For tomato, on average 84.7% of the cloud points of an architecture belonged to lamina, and 15.3% for stems. For *Nicotiana benthamiana*, on average 90.1% of the cloud points belonged to lamina, and 9.9% to stems. Thus, this dataset represents a diverse set of phenotypes to test the generality of our methods across two Solanaceous species.

Classification Framework and 3D Shape Descriptors

As input we are given a point cloud $P=\{p_1, p_2, \ldots, p_n\}$ of n points. Each point $p_i=(x_i, y_i, z_i)$ represents a 3-dimensional location on the surface of the shoot architecture. Each point is also associated with a normal vector, $\overrightarrow{n_i}=(n_i^x, n_i^y, n_i^z)$, indicating the orientation of each point in 3D space. Normal vectors were provided by the 3D scanner, and thus did not need to be determined computationally (Rusu and Cousins, 2011). We associate each point p_i with a local neighborhood of points P_i, defined by a sphere of radius r. This neighborhood consists of all p_j such that $||p_i\text{-}p_j||_2 \leq r$.

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

Several geometric feature descriptors have been proposed, both within the plant phenotyping community (Paulus et al., 2013; Wahabzada et al., 2015) and in the computer vision community (Arbeiter et al., 2012; Guo et al., 2016) more broadly. Below, we summarize the descriptors compared here (Table 1).

Principal curvature

Intuitively, the principal curvature of a point p_i is the speed with which a surface around p_i "moves away" from a tangent plane defined at p_i (Vemuri et al., 1986; Rusu and Cousins, 2011). Thus, the speed of moving away on a flat surface (e.g. a lamina) should be relatively small, and the speed of moving away on a surface of a cylinder (stem) should be relatively fast, at least along some axis.

To estimate the principal curvature of a point p_i, first, every normal vector $\overrightarrow{n_j}$ of $p_j \in P_i$ is projected to a tangent plane formed by $\overrightarrow{n_i}$ and point p_i:

$$\overrightarrow{m_j} = \left(I - \overrightarrow{n_i} \otimes \overrightarrow{n_i}\right) \cdot \overrightarrow{n_j} \tag{1}$$

where I is the 3x3 identity matrix.

Next we calculate the 3×3 covariance matrix C_i of the $\overrightarrow{m_j}$ projections:

$$C_i = \frac{1}{k} \sum_{j=1}^{k} \left(\overrightarrow{m_j} - \overrightarrow{\overline{m}}\right) \otimes \left(\overrightarrow{m_j} - \overrightarrow{\overline{m}}\right) \tag{2}$$

where k is the number of points in the neighborhood of p_i and $\overrightarrow{\overline{m}}$ is the mean of all $\overrightarrow{m_j}$.

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

Table 1. *List of 3D shape feature descriptors*

The summary of features used with our dataset is shown. The size column represents the dimensionality of the feature vector per point in the point cloud.

Descriptor	Size	Reference
PFH	125	Rusu et al., 2008
FPFH	33	Rusu et al., 2009
Radius-based surface descriptor	289	Marton et al., 2010
SHOT	352	Tombari et al., 2010
Spin images	153	Johnson and Hebert, 1999
PC	2	Rusu and Cousins, 2011

Radius-based surface descriptor

This method was originally developed to estimate the radius of an object to determine if the object can be grasped by a robotic arm (Marton et al., 2010).

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

The final radius-based surface descriptor feature can come in two versions. In the first version, f_i will contain two values, the minimum and maximum radii computed between p_i and each point $p_j \in P_i$. In the second version, f_i will contain histograms of counts of the angles and distances between p_i and each point in P_i. The second version performed best, and we report these results below.

Point feature histograms

Rusu et al. (2008) derived a method that generalizes the mean curvature around a point that is more robust to noise. For each point p_i, the method computes four features between all pairs of points $p_s, p_t \in P_i$. These features provide higher descriptive power by capturing all possible normal interactions between the points in the local surface.

For each pair of points $p_s, p_t \in P_i$, we first compute three vectors corresponding to the Darboux frame (Rusu et al., 2008):

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

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

$$\alpha_{st}=\overrightarrow{v}\cdot\overrightarrow{n_t}$$

$$\phi_{st}=\overrightarrow{u}\cdot\frac{(p_t\text{-}p_s)}{||p_t\text{-}p_s||_2}$$

$$\theta_{st}=\ arctan\left(\overrightarrow{w}\cdot\overrightarrow{n_t},\ \overrightarrow{u}\cdot\overrightarrow{n_t}\right)$$

$$d_{st}=||p_t\text{-}p_s||_2$$

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

Fast point feature histograms

Point feature histograms (PFH) features are computationally expensive to compute because the four features are calculated between all pairs of points in P_i, which takes time $O(nk^2)$, where k is the size of the neighborhood. Fast PFH (FPFH) reduces this complexity to $O(nk)$ by only considering pairs that include p_i (Rusu et al., 2009). FPFH also adds a distance-based weighing step that reduces the contribution of a feature based on how far the corresponding point is from p_i. The two steps are:

(1) Compute features $\alpha_{i,j}, \phi_{i,j}, \theta_{i,j}$ between p_i and p_j, where $p_j \in P_i$ and call this operation $S(p_i)$.
(2) Scale each resulting histogram via distance-based weighting.

$$FPFH(p_i)=S(p_i)+\frac{1}{k}\sum_{j=1}^{k}\frac{1}{d_{ij}}\cdot\ S(p_j)$$

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

Signature of histogram of orientations

This descriptor captures local topology around a point by computing a histogram of the orientations of points in P_i. The first stage of the descriptor is to establish a local reference frame, M_i, around point p_i. The goal of this frame is to make the feature locally invariant to geometric transformations (Tombari et al., 2010). The frame is computed using the eigenvalue decomposition of the weighted covariance matrix:

$$M_i = \frac{1}{\sum_{p_j \in P_i} \left(r - d_{ij}\right)} \sum_{p_j \in P_i} \left(r - d_{ij}\right) \left(p_j - p_i\right) \left(p_j - p_i\right)^T, \tag{3}$$

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

Next, we divide the neighborhood into multiple subvolumes of a sphere. The subvolumes are divided along its three spherical axes, and a local histogram is constructed for every subvolume. Each histogram contains bins of normalized point counts of points that lie within the angle between $\overrightarrow{n_i}$ and every normal vector for each point in the subvolume. The final feature f_i is a concatenation of all subvolume histograms, with a quadrilinear interpolation applied to remove abrupt changes in histogram boundaries.

Spin image

Unlike signature of histogram of orientations (SHOT), where the neighborhood is aligned with a local reference frame, the neighborhood in the spin image descriptor is aligned with a local reference axis (Johnson and Hebert, 1999). The local reference axis is established by using the normal vector of p_i.

Intuitively, imagine a cylinder with a cylindrical axis defined by $\overrightarrow{n_i}$. The cylinder is subdivided into bins along its height and radius. Dividing along the height produces subvolumes of smaller cylinders. Dividing along the radius produces subvolumes of rings. The final feature f_i consists of counts of points in each subvolume (after interpolation and distance-based weighting are applied, as above).

Testing and validation framework

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

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

There are many classifiers, and performing cross-validation using all feature × classifier combinations is laborious. Thus, we first used cross-validation with a deep learning classifier (LeCun et al., 2015) to determine the best performing feature (FPFH; "Results"; Fig. 2), and we then tried many different classifiers using this feature. Specifically, we compared four classifiers: deep learning, linear support vector machines (SVM), K-nearest neighbors (KNN), and random forest. Prior work has used FPFH features for plant classification in conjunction with SVMs (Paulus et al., 2013) and k-means (Wahabzada et al., 2015). We also tested our how well classification accuracy generalized to environments that were not trained on ("Results"; Fig. 3).

For each classifier, we performed a grid search to find the best model parameters, evaluated using 6-fold cross-validation. This search included different values of the neighborhood radius, $r \in [1.0,\ 2.0,\ \ldots,\ 12.0]$. For linear SVM, we tested values for C, where $C = \log_2(i),\quad i \in [-8,\ -7,\ -6,\ \ldots,\ 7]$. For deep learning, we varied the number of training steps and the architecture of the neural network. The steps tested were 2000, 20000, and 200000. We tried six different architectures with as few as one hidden layer, as well as deeper networks with up to 10 hidden layers with

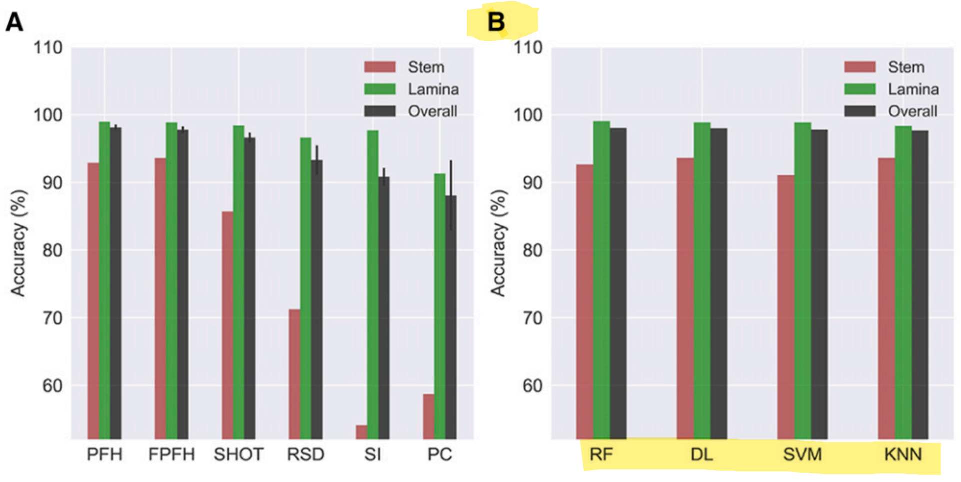
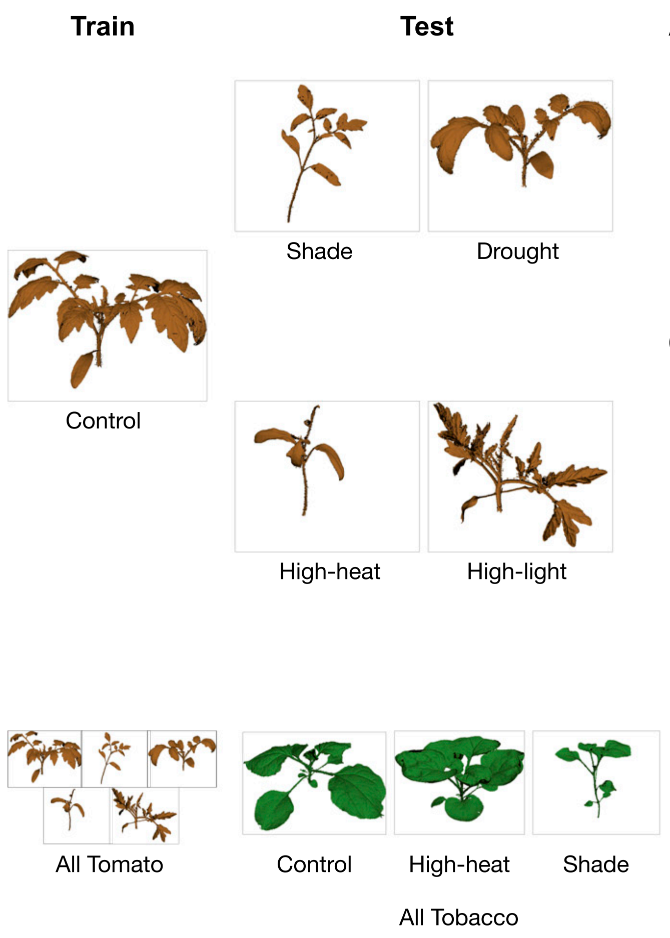


Figure 2. Impact of different classifiers and descriptor features on lamina versus stem classification accuracy. A, Results of 6-fold cross-validation for each descriptor using an underlying deep learning classifier. Error bars indicate SD across folds. PFH and FPFH features perform best. B, Results of 6-fold cross-validation for different classifiers using FPFH descriptors. The three bars for each descriptor/classifier show accuracy by class (stem, lamina, and overall for both classes). RSD, radius-based surface descriptor; SI, spin image; RF, random forest; n = 36 tomato plants.



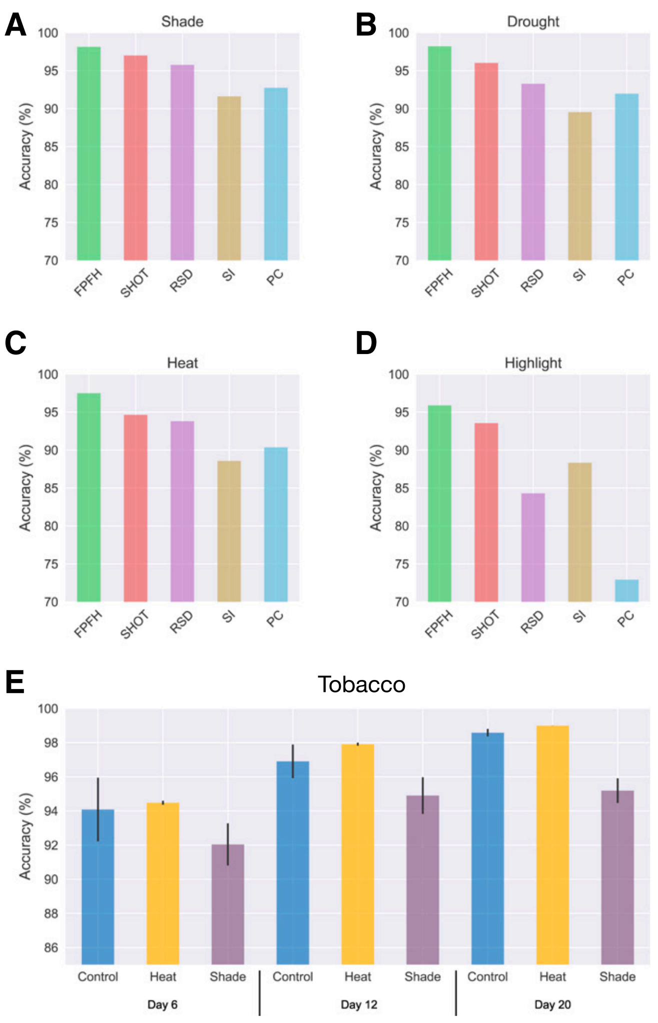


Figure 3. Generalization of classification accuracy across different environments and species. A to D, Results of training a classifier on tomato control plants, and testing on tomato plants for four other conditions (shade, drought, high-heat, and high-light). Over all descriptors, FPFH generalizes the best across conditions. E, Results of training a classifier on tomato data from all conditions and testing it on *Nicotiana benthamiana* data, using FPFH descriptors. Error bars indicate SD over replicates. Performance on the *N. benthamiana* data are shown separately per condition and time point. RSD, radius-based surface descriptor; SI, spin image; $n = 54$ tomato and *Nicotiana benthamiana* plants.

various number of nodes in each layer (Goodfellow et al., 2016). The best architecture contained an input layer, three hidden layers with 33, 66, and 33 nodes, respectively, followed by the output layer. For KNN, we tested K values of 7, 77, 277, and 777. Finally, for the random forest classifier, we used 5, 25, 50, or 100 trees.

Lamina Segmentation Algorithms

The result of the classification yields two sets of points: lamina points and stem points. The lamina points by themselves are "free flowing" islands of points in space. The next challenge is to segment or cluster these lamina points into subsets, where each subset represents an individual lamina. Segmented lamina allow one to calculate traits, such as flowering time and drought tolerance (Minervini et al., 2016); different stresses can also affect lamina size (McCree and Davis, 1974), which can determine photosynthetic response (Valencia et al., 2016).

Lamina segmentation is a difficult problem because lamina can have many different shapes, poses, and orientations, including variation in wrinkles and ripples (Xia et al., 2015; McCormick et al., 2016). Further, the data itself can be noisy due to occlusion of certain parts of the lamina during scanning, issues in registration multiple scans, identification of small newly emerging lamina (Li et al., 2013), and close proximity and perhaps touching of two different lamina (McCormick et al., 2016). Although some methods have been proposed that solve these problems for specific types of lamina structure (e.g. sunflower lamina [Gélard et al., 2017b] or heuristics to deal with some types of touching lamina [McCormick et al., 2016]), here we seek a more general solution that is agnostic to any such prior.

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

Euclidean clustering

The first method clusters lamina points into subsets based on Euclidean distances (Rusu, 2010). A point is picked randomly to start a cluster, and then that cluster is grown by admitting points that are less than a threshold distance away from at least one point from the cluster. This procedure is repeated until all lamina points have a cluster assignment. Ideally, the distance threshold should be larger than the average distance between two points on the same lamina, yet smaller than the distance between points on two distinct lamina. We set the threshold to 0.3 after performing a computational grid search to optimize lamina counting performance. In practice, the threshold is data-set sensitive, both with respect to the scanning hardware/resolution used and to variation in lamina density and structure of the species analyzed. It may be possible to learn the threshold from the dataset, for example, by manually classifying lamina for a few plants of interest, plotting distances between lamina points, and selecting a threshold that well separates different lamina. In addition, we filtered out clusters with less than 300 cloud points, as these were typically noise.

Segmentation using smoothness curvature

The second method clusters lamina points into subsets using normal vectors and estimated curvature values (called smoothness; Rabbani et al., 2006). The objective is to cluster sets of points that have a smooth surface. The method starts by computing and sorting the estimated curvature for all points. A cluster is initialized with the point p_i with the smallest curvature (i.e. the surrounding region around p_i has a flat surface). This cluster is grown by considering p_i's neighbors; for each neighbor p_j, if the angle between p_i and p_j is less than a threshold, p_j is admitted to the cluster. To generate new seed points to grow the cluster, the curvature of p_j is checked to see if it is less than another threshold; if so, p_j is also added to a queue of seed points. If not, p_j remains in the cluster with p_i, but its neighbors are no longer candidates to further expand the cluster. Once all points from the seed queue are exhausted, a new cluster is started with next smallest curvature value (that has not already been assigned to a cluster) and the process repeats. The parameters for the normal calculation and curvature threshold values were selected using grid search on the entire dataset.

Our method

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

Instead, we propose an anisotropic measure of smoothness: $(\lambda_1 - \lambda_3)/\lambda_1$. This measure provides better clustering of points located on edges and in poorly scanned areas (lower point density or holes). Because this method does not consider the second eigenvalue, it produces a smoother curvature in flat areas, and the abrupt changes are more likely to happen where two lamina are in contact.

Our method also uses an additional check to determine whether a point should be admitted into a cluster based on the similarity of FPFH features. This approach is inspired by a similar method used for creating supervoxels for point clouds (Papon et al., 2013). Intuitively, using FPFH features will help capture the essence of the surface, despite possible hairs, wrinkles, or tears on the lamina.

The complexity of FPFH allows this technique to scale to plants with even a million points. In summary, our clustering method starts from a random seed point and the cluster is grown if any of three conditions are satisfied (Algorithm 1). Once the cluster is grown to exhaustion, a next random seed point is drawn from the points that have not been clustered yet, and this process is repeated until all points belong to some cluster.

Testing and validation framework

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

Stem Skeletonization Algorithms

The next challenge is to transform the stem points into an underlying skeleton (a graph-theoretic tree) that captures the essential shape of the plant. Deriving skeleton graphs of plants is commonly used to perform morphological analysis of plant architectures (Bucksch et al., 2017; Conn et al., 2017b; Prusinkiewicz and Lindenmayer, 1996).

To extract a skeleton graph, we extended upon the PypeTree method (Delagrange et al., 2014), which

Algorithm 1: Our lamina segmentation method based on region growing

```
Data: Set of lamina points $L$
Result: Set of lamina clusters $C_L$
Function conditioned_RG($\Delta_d, \Delta_n, \Delta_c, \Delta_f$):
    $C_L$ ← {}   // Empty list of lamina clusters
    $Q$ ← {}   // Queue of points to be checked
    $T$ ← createKD_Tree($L$)   // Used to find candidate nearest-neighbors.

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

            // Difference in Euclidean distance
            if distance($p_i, p_y$) > $\Delta_d$ then
                continue;
            // Difference in normal orientation (angle)
            if degrees($\overrightarrow{n_i}, \overrightarrow{n_y}$) < $\Delta_n$ then
                append($Q, p_y$);
            // Difference in curvature (smoothness anisotropy)
            if distance(Anisotropy($p_i$), Anisotropy($p_y$)) < $\Delta_c$ then
                append($Q, p_y$);
            // Difference in FPFH distance
            if distance(FPFH($p_i$), FPFH($p_y$)) < $\Delta_f$ then
                append($Q, p_y$);
        append($C_L, Q$)
        $Q$ ← {}
    return $C_L$
```

improves upon the Verroust and Lazarus method (Verroust and Lazarus, 2000) and other prior work (Bucksch, 2014). The Verroust and Lazarus method assumed that cloud points are sampled uniformly or near uniformly; PypeTree removed this assumption to better deal with point clouds with inconsistent density and outliers. Prior work has also studied skeletonization of botanical trees (Bucksch et al., 2009; Bucksch et al., 2010; Delagrange et al., 2014); one challenge with our dataset, however, are the numerous small stems, where a more scale-invariant method would be desired.

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

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

(1) A point cloud is turned into a neighborhood graph $G=(V,E)$ by connecting each point with its neighbors. Specifically, each cloud point p_i corresponds to a node $v_i \in V$, and there exists an edge between v_i and v_j if v_j is within radius r from v_i.
(2) The neighborhood graph G is converted into a geodesic graph, $F=(V, E' \subset E)$ with respect to some root point, $v_{root} \in V$. The edges E' of the geodesic graph are found by computing the shortest path from v_{root} to every vertex $v_i \in V$ using Dijkstra's algorithm, and then taking the union of all the edges in these paths.
(3) The nodes of the geodesic graph are divided into "levels" based on the geodesic distance from the root. In other words, all nodes within distance 1 from the root are placed into the first level; all nodes with distance between 1 and 2 are placed into the second level; etc. Figure 4, A and B, illustrates nodes assigned to different levels with different colors. Each level will ultimately represent one or more nodes in the final skeleton graph (see Step 5 below).
(4) A subgraph H_i is created for each level i. The nodes of H_i are all the points in level i. Each node in H_i is connected to its two nearest neighbors in H_i.
(5) A subgraph H_i may contain multiple connected components; for example, after a branch point (Y-junction) in the preceding level, there may be two connected components in the same level, representing the left and right sides of the stem. Thus, we create one node in the final skeleton graph per connected component, and this node is created at the centroid of the connected component. The final step is to create edges between centroid nodes in level i and level $i + 1$. A centroid in level $i + 1$ is connected to a centroid in level i if a (random) point in the former's connected component goes through some point in the latter's connected

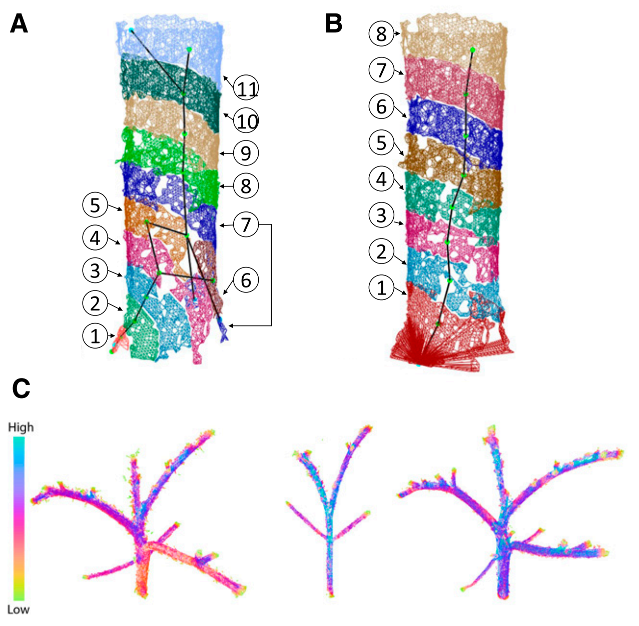


Figure 4. Improvements in stem skeletonization. A, Visualization of the levels (colored bands) outputted by the PypeTree method. The default root point shown in Level 1 can be off-center when selected as the point with the lowest y-value. This can cause tilting of the architecture and other deformities in the resulting skeletonization (black line), especially when holes are present in the point cloud data. B, Visualization of the levels and root point outputted by our method. The balanced levels and centered root point improve the quality of the skeleton graph. C, Visualization of first (most dominant) eigenvalues for each point. Points with low eigenvalues, as indicated by the scale bar, are predominantly located at tips (the root and regions where lamina emerge).

component along the way to the root in the geodesic graph. Intuitively, this will avoid stems from "crossing over" across levels.

PypeTree includes additional filtering steps to reduce noise and other implementation optimizations to reduce running time, which we do not describe here and instead refer the reader to the original publication (Delagrange et al., 2014).

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

Improved detection of tips

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

Our approach to finding tips is independent of connected components or level size, thus making it independent of the resolution of the skeleton graph. Tips are located at the boundary (edges) of a point cloud, which have a natural discontinuity compared with other points. Thus, we can detect edge points by using principal components analysis. Specifically, for every point in the point cloud, we compute its first eigenvalue based on its neighboring points within some radius r. The first eigenvalue of points close to the boundary will be smaller than that of points far from a boundary because the neighbors of boundary points will not be spread around the point but rather will be biased to some side (Fig. 4C). We then perform Euclidean clustering on all points whose first eigenvalue is among the smallest 15% of all points, such that points that belong to the same tip should reside in the same cluster. Clusters that contain less than 50 points are filtered out as noise. The centroids of each cluster become the tip nodes of the skeleton graph. Finally, a tip is attached to the skeleton graph by picking a random point from its cluster, finding which connected component it falls in (Step 5 above), and adding an edge from the centroid of this cluster to the tip. Finding tips in this manner allows for the detection of even small stems emerging during development.

Improved detection of the root

The stem base (hereafter simply called the "root") is a special case of a tip and is critical for grounding the skeleton in the correct location. The goal is to find a root point that lies at the base and center of the stem. This location of the plant, however, is often difficult to scan because of its proximity to the soil and because of holes caused by scanning errors. The default root point of PypeTree is the point with the lowest y-value. This point can be highly skewed to one side of the stem (Fig. 4A), which can cause artificial curvature in the skeleton.

Our solution starts by noticing that the level closest to the default root—i.e. all the points within distance 1 to the root—often contains points that do not wrap evenly around the base of the stem (Level 1, Fig. 4A). This is because these distances are highly sensitive to noise, holes, or other small variations around the root. However, as the level number increases, the points appear more evenly balanced around the stem, forming even rings (Level 11, Fig. 4A).

To form even rings around the root, then, our idea is to build the geodesic graph starting from the tip that is furthest away from the root and trace *backward* (Level 1, Fig. 4B). In this way, the level closest to the root now will be less sensitive to variability. Specifically, let r_1 be PypeTree's default root, and let r_2 be the point that is furthest away from r_1. Then we find all points p_i such that:

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

where β is a parameter that adjusts amount of points to be sampled in the vicinity of the root. Then, for each p_i we compute:

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

Intuitively, all points with $z(p_i) \approx 0$ will represent an evenly balanced ring around the true root (Level 1, Fig. 4B). We reset the root v_{root} to be the centroid of these points, and then add edges from v_{root} to all such points (with equal weight) to the neighborhood graph G. We then proceed with Step 2 of the PypeTree algorithm above.

Testing and validation framework

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

Application and Validation

First, we describe results of classifying between lamina and stem points on our dataset of 54 shoot architectures, including generalization tests to assess how classifiers perform on an additional Solanaceous species and on growth conditions for which no training data are used. Second, we use the lamina points as input to compare our lamina segmentation/counting approach versus classic methods. Third, we use the stem points as input to compare our algorithm to generate skeleton graphs versus prior methods. Finally, we combine these three tasks to isolate individual tomato leaves and their connecting stem tissue.

Improved Classification of Lamina and Stem Points in 3D Shoot Architectures

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

Features

Using 6-fold cross-validation on the 36 tomato plants, we found that point feature histograms (PFH, and their faster version, FPFH) achieved the highest accuracy in classifying between lamina and stem points: 98.12% ± 0.39% and 97.77% ± 0.50%, respectively (Fig. 2A).

Although PFH was slightly more accurate than FPFH (by 0.35%), PFH's complexity is $O(nk^2)$ (Rusu, 2010) as opposed $O(nk)$ for FPFH. This difference corresponds to days to perform cross-validation versus a few hours for FPFH, and thus FPFH achieves a better trade-off between scalability and accuracy.

Because our dataset is not balanced between classes (≈ 85% of points are lamina, ≈ 15% are stems) we computed the accuracy for each feature by class (Fig. 2A). Here again, PFH and FPFH perform similarly and dominate the other features. For lamina points, the accuracies per class was 98.94% and 98.85%, respectively for PFH and FPFH; for stem points, the accuracies were 92.86% and 93.60%, respectively. The next best feature, SHOT, performs comparable for lamina, but substantially worse (85.69%) for stem points. We see the largest variation in performance for stem point classification, ranging from 93.60% (FPFH) to 54.1% (spin image features). We note that an improvement in even 1% accuracy corresponds to potentially thousands of cloud points, and thus can be substantial.

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

Classifiers

The classification results above were achieved using a deep learning classifier. Prior work has used a variety of classifiers for segmentation. For example, Paulus et al. (2013) used FPFH in conjunction with SVMs, and Wahabzada et al. (2015) used FPFH with k-means. Here, we study the effect of using different classifiers in conjunction with FPFH features for classification accuracy. We tested the following classifiers: KNN, random forest, SVM, and deep learning (DL).

Several of these classifiers performed comparably (Fig. 2B); however, when comparing the nature of the errors of each method, we found that DL made the most "reasonable" mistakes. Specifically, we visualized the errors made by each method and noticed qualitatively that most of the errors made by DL were in ambiguous regions of the plant where lamina emerge from a stem. In these regions, it is difficult, even for humans, to agree on the precise boundary between a stem and a lamina. The other methods (RF, KNN) made more mistakes in less ambiguous locations (e.g. the middle of a lamina or stem). Thus, for the remaining experiments, we used the DL classifier.

Generalization across different environments

Plants architectures are highly plastic and often modify their structure (including lamina shape, stem thickness, and stem angles) in response to the environment. In practice, training data may not be available for all possible conditions, and thus in practical applications, it is important that classifiers can accurately classify lamina and stems when applied to architectures grown in never-seen-before conditions.

To test this, we trained a DL model using a set of 12 control tomato plants (ranging from developmental days 5–20) and tested the model on plants grown in four other conditions: shade, drought, high-heat, and high-light (Conn et al., 2017b). This condition produced many variations in architectural features (Fig. 3, left). For example, in high-light, we observed thickening of the lamina and stem compared with all other conditions, and many more irregularities in lamina shape. In drought, we observed a weaker plant with fewer and thinner lamina and stems.

When compared with training a DL classifier with other features, we found that FPFH features again performed best, achieving over 95% accuracy on all new conditions (Fig. 3, A–D). For example, in high-light, FPFH achieved an overall accuracy of 95.88%, compared with 93.55% for SHOT and 72.91% for principal curvature (PC). In high-heat, FPFH achieved 97.48% accuracy, compared with 94.64% for SHOT and 90.35% for PC. In drought, FPFH achieved 98.2% accuracy compared with 96.01% for SHOT and 91.96% for PC. Principal curvatures has the largest variance across conditions, perhaps because this feature only consists of two values. On the other hand, SHOT has over 10 times as many feature values as FPFH (352 versus 33), but it still performed worse than FPFH.

While prior results have also proposed using FPFH features for lamina/stem classification (Paulus et al., 2013; Wahabzada et al., 2015), here we quantitatively demonstrate its accuracy, coupled with deep learning, across new conditions.

Table 2. *Lamina segmentation and counting results*

Comparison of three methods: Euclidean Clustering, Segmentation with Smoothed Curvature, and our method. 'Under' and 'Over' indicate the number of under- and oversegmented lamina, respectively. Summary on the bottom row shows overall performance over all plants.

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

Generalization from tomato to Nicotiana benthamiana *plants*

A related challenge is to build classifiers that can accurately classify lamina and stems on species that have not been trained on. Indeed, there are thousands of plant species, some with very different branching and lamina patterns, and it is unreasonable to assume that training data will be available for all of them. Here, we tested how well a DL classifier trained with FPFH features from tomato plants generalized to another dicot plant: *Nicotiana benthamiana*. The *Nicotiana benthamiana* plants were also collected under three growth environments (control, shade, and high heat) and across three developmental time points (days 6, 12, and 20; Conn et al., 2017b).

Overall, we find >95.89% accuracy, averaged over all *Nicotiana benthamiana* plants (Fig. 3E). Older plants are classified more accurately regardless of the environment (e.g. 98.6% control day 20 versus 94.1% control day 6), whereas plants grown in shade had a slightly lower accuracy. Nonetheless, these results suggest that some cross-species generalization from tomato to *Nicotiana benthamiana* is possible with our framework and sets a benchmark for future studies to improve upon.

Improvement in Lamina Counting

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

When analyzed on 291 total ground-truth lamina from the tomato plants, our enhanced lamina counting method achieved an overall accuracy of 86.60%, with 13 undersegmented and 45 oversegmented lamina (Table 2). In contrast, the Euclidean clustering method achieved 69.42% accuracy, with 55 undersegmentations and 2 oversegmentations. The performance of Euclidean clustering decline substantially as plants grow and more lamina started to become in contact with each other. Finally, segmentation with the smoothness constraint performed better than Euclidean clustering, achieving 83.16% accuracy, with 23 undersegmentations and 23 oversegmentations, although still slightly worse than our method. Thus, adding FPFH features to the growing criteria helped the algorithm deal better with natural variations in lamina structure, including hair, ripples, wrinkles, and veination. Although not reported here, we also tested other eigenvalue-based curvature metrics, such as planarity and surface variation (Hackel et al., 2016), which did not perform competitively.

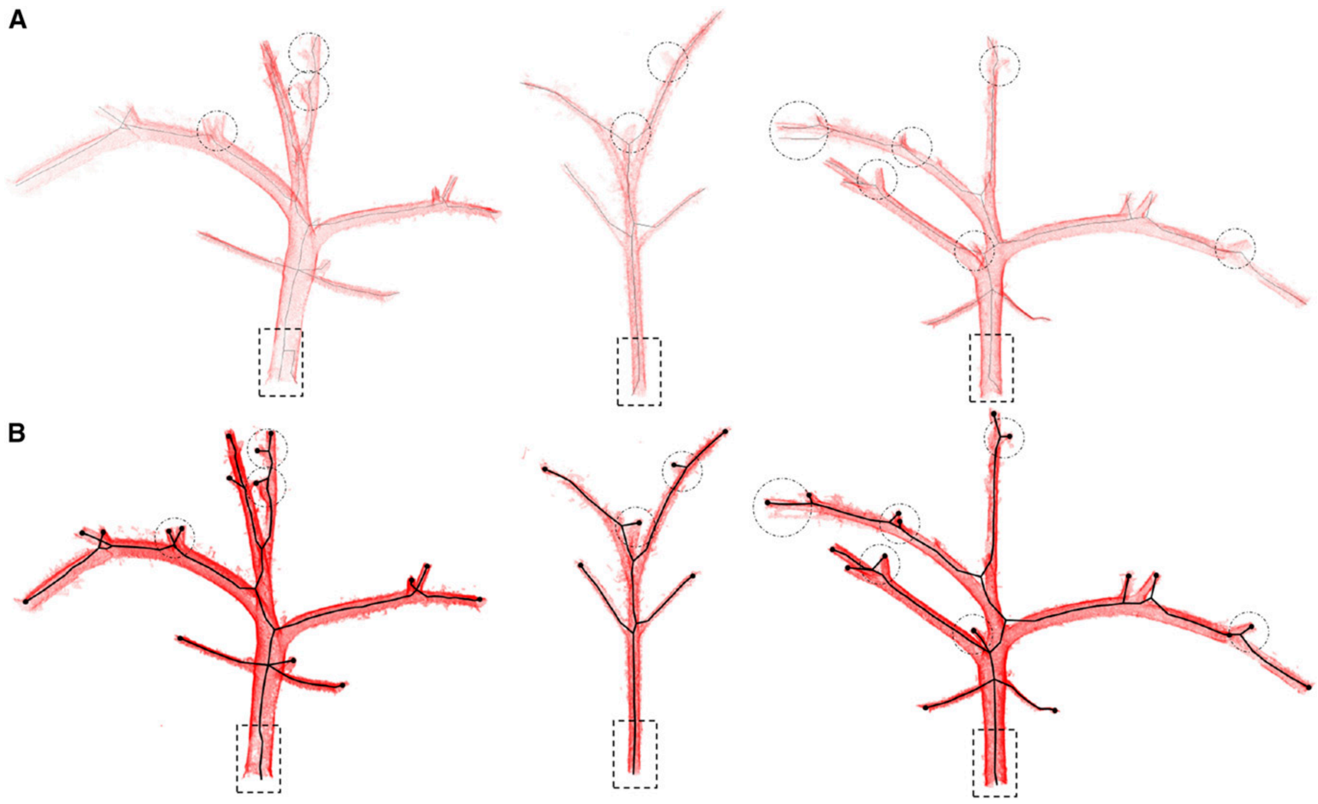


Figure 5. Stem skeletonization comparison for three plants. The stem skeletonization produced by PypeTree (A) and our method (B), on three tomato plants. The red points are the stem cloud points, and the black line is the resulting skeleton graph. Dotted circles highlight areas of improved tip detection for our method; dotted rectangles highlight instances where the root tip was off-center according to PypeTree and better centered by our method.

Across all methods, the bulk of oversegmentation errors occurred for lamina with scanning holes or with poor registration from multiple scans. All of these methods include parameters that can be tuned to optimize under- and oversegmentation, and future work may try to introduce a learning procedure to set these parameters automatically given data.

Improvements in Stem Skeletonization

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

First, we find that our algorithm generated skeletons that are visually better aligned with the underlying cloud points (Fig. 5). PypeTree consistently struggled identifying small stems (Fig. 5A), corresponding to where new lamina are emerging. PypeTree also often off-centered the root point, which caused the underlying skeleton to incorrectly lean. Although PypeTree does include parameters to adjust the level size that can improve these errors in some cases, it was difficult to set this parameter in such a way that it generalized across species, plant size, and branching patterns. In contrast, our method, which is independent of level size parameters, defined a root point that was much more centered along the main axis of the stem and better dealt with more ambiguous regions of the point cloud (Fig. 5B).

Second, we quantified how accurately each method identified tips on a set of 12 tomato plants from day 20 across the five environmental conditions (Table 3). Day 20 plants were selected because these plants have the largest size and most branch points. Our method achieved higher or equivalent precision on 11 of the 12 plants, and higher or equivalent recall on all 12 plants, compared with PypeTree. Averaged over all 12 plants, our method achieved a precision of 95.0% ± 7.0% compared with 89.0% ± 12.0% for PypeTree. The reduction in false positive and false negative tips is important, as such errors can alter the underlying shape of the skeleton, which can affect downstream morphological analyses.

Third, we evaluated the run-time of each method and found 1 to 2 orders of magnitude improvement in our algorithm versus the provided PypeTree implementation (Table 3). For example, on the Highlight-B plant, PypeTree took 919 s, whereas our method only took 17.51 s. This improvement is subsantial, especially when dealing with large collections of plants scanned in real-time applications.

Isolating True Leaves of Tomato Plants

Here, we show how to isolate true leaves of tomato plants by building off the three pipelines developed above. This requires solving two tasks. First, the lamina classification and counting steps together identified a set of individual lamina, which need to be grouped into individual leaves. Second, the stem classification step identified petioles and petiolules, which also need to be associated with individual leaves based on their branching patterns (as identified by the skeletonization algorithm).

Algorithmically, we first assign each lamina to the closest tip node on the skeleton graph. This gives us a full skeleton graph with nodes corresponding to the root, graph-theoretic branch points, or lamina. Finding the cotyledon blades requires iterating through each level starting at level 0 (root) and finding the two closest lamina nodes that have similar geodesic distance from the root and that are oriented in nearly the opposite direction. Specifically, we mark each node u with the number of tips in the subtree rooted at u. Nodes marked with a value of 1 and satisfying the conditions above indicate a cotyledon blade. We then group the cotyledon petioles with the cotyledon blades by finding the first and second branch point off the main stem,

Table 3. *Improvement in skeleton tips identification*

Precision vs. recall of PypeTree versus our tip identification method on 12 tomato plants at day 20. Overall, our method achieves higher precision and recall and is over an order of magnitude faster.

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

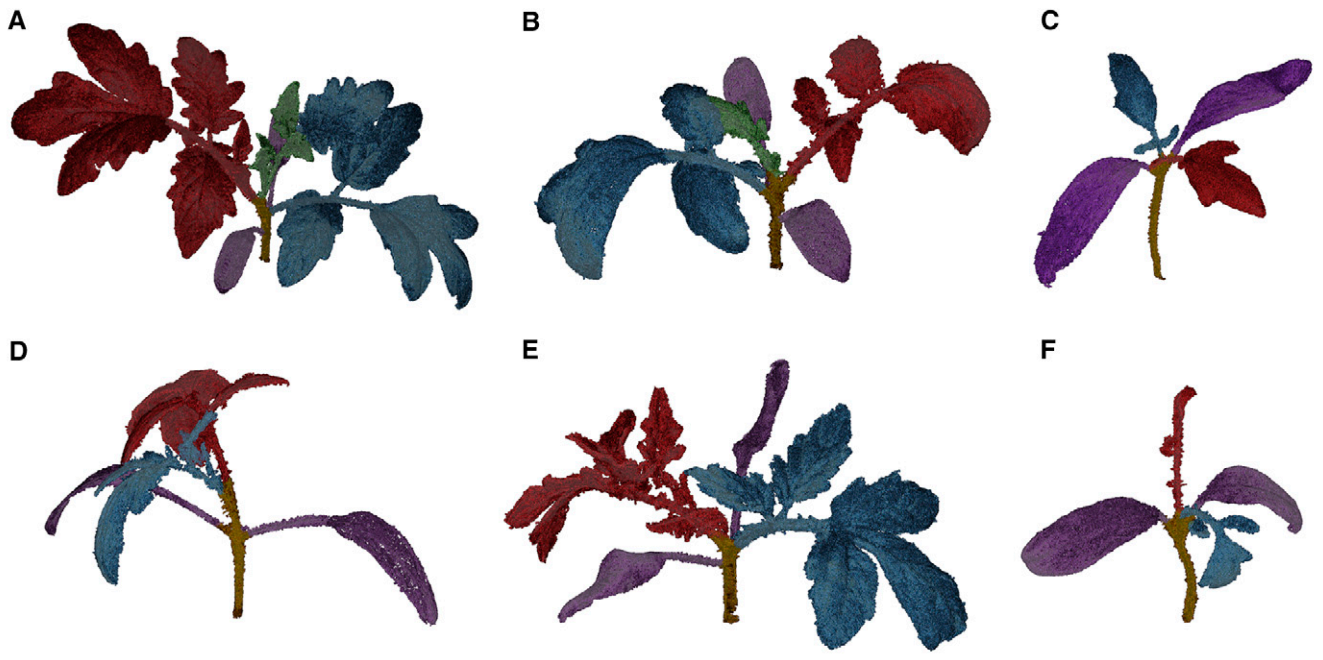


Figure 6. Isolating true leaves of tomato plants. Combining the three pipelines to segment individual tomato leaves. The main stem is shown in brown; the cotyledons are shown in purple; and each individual leaf, including associated lamina, petioles, and petiolules, are shown in a different color (red, purple, green). A, Control. B, Drought. C, Shade. D, Control. E, Highlight. F, Heat.

respectively. A similar procedure is used to identify individual leaves, by grouping together individual lamina and their connecting petioles and petiolules. Each node in the full skeleton graph is associated with a set of points from the original point cloud when it was sectioned into levels. Thus, each individual cloud point can be assigned to either the stem or to an individual leaf.

Figure 6 shows examples of six tomato plants with the main stem and individual leaves high- lighted. The stem is shown in brown; the cotyledons are shown in purple; and each individual leaf (including associated lamina, petioles, and petiolules) are shown in a different color. The six plants were selected from different days and different conditions: highlight, heat, drought, shade, and control. Overall, we find that our pipelines can be used to accurately segment individual leaves, which is an important component of downstream leaf analyses. This procedure to isolate leaves takes specific advantage of the structure of tomato plants; other logic may be needed to isolate the leaves of other, especially non-Solanaceous, species.

CONCLUSIONS

High-throughput phenotyping requires fast and accurate analysis methods that can handle large, noisy, and heterogeneous point cloud data. Here, we tackled three basic problems of point cloud analysis: lamina versus stem classification, lamina counting/segmentation, and stem skeletonization.

First, we showed how a deep learning classifier using fast point feature histogram features can achieve over 95% accuracy in classifying lamina versus stem points, outperforming many other features and classifier types. We also tested how well this approach generalized to instances where training data are not available for a test condition of interest, for example, testing on a new species or new growth condition. This is an important yet at times understudied component of validating phenotyping algorithms, especially because many different architectures can be formed within the same genotype. Second, we extended a classic region growing algorithm to improve segmentation and counting of individual lamina. Third, we developed an enhanced stem skeletonization algorithm with more visually aligned skeletons, improved tip identification, and faster run-time.

When compared with prior approaches that addressed these problems only on a small number of plants or plants from only a single species (Paulus et al., 2013; Wahabzada et al., 2015; McCormick et al., 2016; Gélard et al., 2017), our dataset encompassed 3 time points, 3–5 growth environments, and 2 species, which allowed us to test some generalization ability of our algorithms. Of course, the two Solanaceous species we selected (tomato and *Nicotiana benthamiana*) both demonstrate phyllotaxis and other similar growth habits, and it remains to be seen how well our algorithm generalizes to the many thousands of other plant species. Further, unlike some prior work (McCormick et al., 2016; Gélard et al., 2017a, 2017b), we purposefully did not perform any filtering, manual clean up, or other such preprocessing steps to the point cloud data to mimick the type of data expected in real-time scanning applications. Overall, these techniques can help improve downstream analyses, such as those based on analyzing lamina shape and size, self-shading, developmental growth, and morphological properties of stem patterning. Moreover, the accuracy of our methods may help uncover new phenotypes that have not yet been quantified, and may be used to perform quantitative genetic analyses to identify genes involved in regulating shoot architectures (Zhou et al., 2019).

These are several avenues for future work. First, advances in point cloud analysis using deep learning without feature extraction have been recently proposed (Charles et al., 2017; Qi et al., 2017); however, there remain challenges in using this technique when plant size varies. Second, we tested our stem skeletonization algorithm on shoot architectures, but it is possible that a very similar method can also be applied to skeletonizing root system architectures to study foraging behavior (De Kroon et al., 2009; Zhu et al., 2011; Flavel et al., 2017; Morris et al., 2017; Shahzad and Amtmann, 2017; Atkinson et al., 2019). Third, because lamina structure can vary across environments, prior methods (Dey et al., 2012) have proposed using conditional random field spatial smoothing to reduce noise and improve lamina/stem classification, although this may impose additional computational overhead. Fourth, extracting lamina veins from individually identified lamina may provide another useful lamina characteristic to aid segmentation, and may enable study of vasculature (Dengler and Kang, 2001; Nelson and Dengler, 1997). Fifth, while architecture data captured using 3D point clouds may be expected to depict fewer overlapping lamina compared to using 2D imaging, segmenting lamina that are touching remains an outstanding challenge in leaf analysis. Prior work has proposed methods tailored to particular leaf structures (e.g. sunflowers; Gélard et al., 2017) or particular shapes of blades (e.g. unlobed; McCormick et al., 2016); however more general approaches are currently lacking. Finally, although our approach can isolate individual tomato leaves, a next step would be to classify each tomato leaf into subsequent categories: primary, intercalary, secondary, and tertiary.

ACKNOWLEDGMENTS

The authors thank Adam Conn, Matthieu Platre, and Sam Sultan for helpful comments on the manuscript.

Received April 30, 2019; accepted September 15, 2019; published October 7, 2019.

LITERATURE CITED

Andújar D, Rueda-Ayala V, Moreno H, Rosell-Polo JR, Escolá A, Valero C, Gerhards R, Fernández-Quintanilla C, Dorado J, Griepentrog H-W (2013) Discriminating crop, weeds and soil surface with a terrestrial LIDAR sensor. Sensors (Basel) **13:** 14662–14675

Arbeiter G, Fuchs S, Bormann R, Fischer J, Verl A (2012) Evaluation of 3D feature descriptors for classification of surface geometries in point clouds. 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1644–1650

Atkinson JA, Pound MP, Bennett MJ, Wells DM (2019) Uncovering the hidden half of plants using new advances in root phenotyping. Curr Opin Biotechnol **55:** 1–8

Balduzzi M, Binder BM, Bucksch A, Chang C, Hong L, Iyer-Pascuzzi AS, Pradal C, Sparks EE (2017) Reshaping plant biology: Qualitative and quantitative descriptors for plant morphology. Front Plant Sci **8:** 117

Bucksch A (2014) A practical introduction to skeletons for the plant sciences. App Plant Sci **2:** 1400005

Bucksch A, Atta-Boateng A, Azihou AF, Battogtokh D, Baumgartner A, Binder BM, Braybrook SA, Chang C, Coneva V, DeWitt TJ, et al (2017) Morphological plant modeling: Unleashing geometric and topological potential within the plant sciences. Front Plant Sci **8:** 900

Bucksch A, Lindenbergh R, Menenti M (2010) SkelTre: Robust skeleton extraction from imperfect point clouds. Vis Comput **26:** 1283–1300

Bucksch A, Lindenbergh RC, Menenti M (2009) SkelTre - fast skeletonisation for imperfect point cloud data of botanic trees. Proceedings of the 2nd Eurographics Conference on 3D Object Retrieval. 3DOR '09. Munich, Germany: Eurographics Association, pp. 13–20

Charles RQ, Su H, Kaichun M, Guibas LJ (2017) PointNet: Deep learning on point sets for 3D classification and segmentation. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 77–85

Chaudhury A, Ward C, Talasaz A, Ivanov A G, Brophy M, Grodzinski B, Huner NPA, Patel R V Barron JL (2018) Machine vision system for 3D plant phenotyping. IEEE/ACM transactions on computational biology and bioinformatics, doi:10.1109/TCBB.2018.2824814

Chene Y, Rousseau D, Lucidarme P, Bertheloot J, Morel P, Belin E, Chapeau-Blondeau F (2012) On the use of depth camera for 3D phenotyping of entire plants. Comput Electron Agric **82:** 122–127

Conn A, Pedmale UV, Chory J, Navlakha S (2017b) High-resolution laser scanning reveals plant architectures that reflect universal network design principles. Cell Syst **5:** 53–62.e3

Conn A, Pedmale UV, Chory J, Stevens CF, Navlakha S (2017a) A statistical description of plant shoot architecture. Curr Biol **27:** 2078–2088.e3

De Kroon H, Visser EJ, Huber H, Mommer L, Hutchings MJ (2009) A modular concept of plant foraging behaviour: The interplay between local responses and systemic control. Plant Cell Environ **32:** 704–712

Delagrange S, Jauvin C, Rochon P (2014) PypeTree: A tool for reconstructing tree perennial tissues from point clouds. Sensors (Basel) **14:** 4271–4289

Dengler N, Kang J (2001) Vascular patterning and leaf shape. Curr Opin Plant Biol **4:** 50–56

Dey D, Mummert L, Sukthankar R (2012) Classification of plant structures from uncalibrated image sequences. 2012 IEEE Workshop on the Applications of Computer Vision (WACV), pp. 329–336

Fiorani F, Schurr U (2013) Future scenarios for plant phenotyping. Annu Rev Plant Biol **64:** 267–291

Flavel RJ, Guppy CN, Rabbi SMR, Young IM (2017) An image processing and analysis tool for identifying and analysing complex plant root systems in 3D soil using non-destructive analysis: Root1. PLoS One **12:** e0176433

Furbank RT, Tester M (2011) Phenomics--technologies to relieve the phenotyping bottleneck. Trends Plant Sci **16:** 635–644

Gaëtan L, Serge C, Annie E, Didier C, Frédéric B (2012) Characterization of whole plant leaf area properties using laser scanner point clouds. 2012 IEEE 4th International Symposium on Plant Growth Modeling, Simulation, Visualization and Applications, pp. 250–253

Gao T, Emadi H, Saha H, Zhang J, Lofquist A, Singh A, Ganapathysubramanian B, Sarkar S, Singh AK, Bhattacharya S (2018) A novel multirobot system for plant phenotyping. Robotics **7:** 61

Gélard W, Herbulot A, Devy M, Debaeke P, McCormick RF, Truong SK, Mullet J (2017a) Leaves Segmentation in 3D Point Cloud. In J Blanc-Talon, R Penne, W Philips, D Popescu, and P Scheunders, eds, Advanced Concepts for Intelligent Vision Systems. Springer International Publishing, Cham, pp 664–674

Gélard W, Devy M, Herbulot A, Burger P (2017b) Model-based Segmentation of 3D Point Clouds for Phenotyping Sunflower Plants. 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP 2017). Vol. 4. Porto, Portugal, pp. 459–467

Gibbs JA, Pound MP, French AP, Wells DM, Murchie EH, Pridmore TP (2018) Plant phenotyping: An active vision cell for three-dimensional plant shoot reconstruction. Plant Physiol **178:** 524–534

Goodfellow I, Bengio Y, Courville A (2016) Deep Learning. The MIT Press

Guo Y, Bennamoun M, Sohel F, Lu M, Wan J, Kwok NM (2016) A comprehensive performance evaluation of 3d local feature descriptors. Int J Comput Vis **116:** 66–89

Hackel T, Wegner JD, Schindler K (2016) Contour Detection in Unstructured 3D Point Clouds. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1610–1618

Heckwolf S, Heckwolf M, Kaeppler SM, de Leon N, Spalding EP (2015) Image analysis of anatomical traits in stalk transections of maize and other grasses. Plant Methods **11:** 26

Huang C, Wang Z, Quinn D, Suresh S, Hsia KJ (2018) Differential growth and shape formation in plant organs. Proc Natl Acad Sci USA **115:** 12359–12364

McCree KJ, Davis SD (1974) Effect of water stress and temperature on leaf size and on size and number of epidermal cells in grain sorghum1. Crop Sci **14:** 751–755

Johnson AE, Hebert M (1999) Using spin images for efficient object recognition in cluttered 3D scenes. IEEE Trans Pattern Anal Mach Intell **21:** 433–449

LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature **521:** 436–444

Li M, Frank MH, Coneva V, Mio W, Chitwood DH, Topp CN (2018) The persistent homology mathematical framework provides enhanced genotype-to-phenotype associations for plant morphology. Plant Physiol **177:** 1382–1395

Li Y, Fan X, Mitra NJ, Chamovitz D, Cohen-Or D, Chen B (2013) Analyzing growing plants from 4D point cloud data. ACM Trans Graph **32:** 157:1-157:10

Madec S, Baret F, de Solan B, Thomas S, Dutartre D, Jezequel S, Hemmerlé M, Colombeau G, Comar A (2017) High-throughput phenotyping of plant height: Comparing unmanned aerial vehicles and ground LiDAR estimates. Front Plant Sci **8:** 2002

Marton ZC, Pangercic D, Blodow N, Kleinehellefort J, Beetz M (2010) General 3D modelling of novel objects from a single view. 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3700–3705

Mathan J, Bhattacharya J, Ranjan A (2016) Enhancing crop yield by optimizing plant developmental features. Development **143:** 3283–3294

McCormick RF, Truong SK, Mullet JE (2016) 3D sorghum reconstructions from depth images identify QTL regulating shoot architecture. Plant Physiol **172:** 823–834

Minervini M, Fischbach A, Scharr H, Tsaftaris SA (2016) Finely-grained annotated datasets for image-based plant phenotyping. Pattern Recognit Lett **81:** 80–89

Moraes Frasson RP d, Krajewski WF (2010) Three-dimensional digital model of a maize plant. Agric For Meteorol **150:** 478–488

Morris EC, Griffiths M, Golebiowska A, Mairhofer S, Burr-Hersey J, Goh T, von Wangenheim D, Atkinson B, Sturrock CJ, Lynch JP, et al (2017) Shaping 3D root system architecture. Curr Biol **27:** R919–R930

Nelson T, Dengler N (1997) Leaf vascular pattern formation. Plant Cell **9:** 1121–1135

Nguyen CV, Fripp J, Lovell DR, Furbank R, Kuffner P, Daily H, Sirault X (2016) 3D scanning system for automatic high-resolution plant phenotyping. IEEE Conference on Digital Image Computing: Techniques and Applications (DICTA), pp. 1–8

Nguyen TT, Slaughter DC, Max N, Maloof JN, Sinha N (2015) Structured light-based 3D reconstruction system for plants. Sensors (Basel) **15:** 18587–18612

Papon J, Abramov A, Schoeler M, Worgotter F (2013) Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2027-2034

Paproki A, Sirault X, Berry S, Furbank R, Fripp J (2012) A novel mesh processing based technique for 3D plant analysis. BMC Plant Biol **12:** 63

Paulus S, Dupuis J, Mahlein A-K, Kuhlmann H (2013) Surface feature based classification of plant organs from 3D laserscanned point clouds for plant phenotyping. BMC Bioinformatics **14:** 238

Paulus S, Behmann J, Mahlein A-K, Plümer L, Kuhlmann H (2014) Low-cost 3D systems: Suitable tools for plant phenotyping. Sensors (Basel) **14:** 3001–3018

Perez-Sanz F, Navarro PJ, Egea-Cortines M (2017) Plant phenomics: An overview of image acquisition technologies and image data analysis algorithms. Gigascience **6:** 1–18

Prusinkiewicz P, Lindenmayer A (1996) The Algorithmic Beauty of Plants. Springer-Verlag New York, Inc., New York, NY

Qi CR, Yi L, Su H, Guibas LJ (2017) Advances in neural information processing systems. In I Guyon, UV Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, eds, PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space, Vol **Vol 30**. Curran Associates, Inc., pp 5099–5108

Rabbani T, van den Heuvel F, Vosselman G (2006) Segmentation of point clouds using smoothness constraints. English. ISPRS 2006: Proceedings of the ISPRS commission V symposium Vol. 35, part 6: image engineering and vision metrology, Dresden, Germany 25-September 27, 2006. Ed. by Maas, H. and Schneider, D. Vol. 35. International Society for Pho- togrammetry and Remote Sensing (ISPRS), pp. 248–253

Rusu RB (2010) Semantic 3D object maps for everyday manipulation in human living environments. KI-Künstliche Intelligenz **24:** 345–348

Rusu RB, Blodow N, Beetz M (2009) Fast Point Feature Histograms (FPFH) for 3D registration. 2009 IEEE International Conference on Robotics and Automation, pp. 3212–3217

Rusu RB, Cousins S (2011) 3D is here: Point Cloud Library (PCL). 2011 IEEE International Conference on Robotics and Automation, pp. 1–4

Rusu R, Marton R, Blodow N, Beetz M (2008) Persistent point feature histograms for 3D point clouds. In W Burgard, R Dillamn, and C Plagemann, eds, Proceedings of the 10th International Conference on Intelligent Autonomous Systems. IOS Press, Baden, Germany, pp 119–128

Santos TT, Koenigkan LV, Barbedo JGA, Rodrigues GC (2015) L Agapito, MM Bronstein, and C Rother, eds, 3D Plant Modeling: Localization, Mapping and Segmentation for Plant Phenotyping Using a Single Hand-held Camera Computer Vision - ECCV 2014 Workshops. Springer International Publishing, Cham, pp 247–263

Setter TL (2012) Analysis of constituents for phenotyping drought tolerance in crop im provement. Front Physiol **3:** 180

Shahzad Z, Amtmann A (2017) Food for thought: How nutrients regulate root system architecture. Curr Opin Plant Biol **39:** 80–87

Sun S, Li C, Paterson AH, Jiang Y, Xu R, Robertson JS, Snider JL, Chee PW (2018) In-field high-throughput phenotyping of cotton plant height using LiDAR. Front Plant Sci **9:** 16

Szeliski R (2010) Computer Vision: Algorithms and Applications. Springer Science & Business Media

Tombari F, Salti S, Di Stefano L (2010) Unique Signatures of Histograms for Local Surface Description. In K Daniilidis, P Maragos, and N Paragios, eds, Computer Vision – ECCV 2010. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 356–369

Ubbens JR, Stavness IK (2017) Deep plant phenomics: A deep learning platform for complex plant phenotyping tasks. Front Plant Sci **8:** 1190

Valencia E, Quero JL, Maestre FT (2016) Functional leaf and size traits determine the photosynthetic response of 10 dryland species to warming. J Plant Ecology **9:** 773–783

Vemuri B, Mitiche A, Aggarwal J (1986) Curvature-based representation of objects from range data. Image Vis Comput **4:** 107–114

Verroust A, Lazarus F (2000) Extracting skeletal curves from 3D scattered data. Vis Comput **16:** 15–25

Wahabzada M, Paulus S, Kersting K, Mahlein A-K (2015) Automated interpretation of 3D laserscanned point clouds for plant organ segmentation. BMC Bioinformatics **16:** 248

Wilf P, Zhang S, Chikkerur S, Little SA, Wing SL, Serre T (2016) Computer vision cracks the leaf code. Proc Natl Acad Sci USA **113:** 3305–3310

Xia C, Wang L, Chung B-K, Lee J-M (Aug. 2015) In situ 3D segmentation of individual plant leaves using a RGB-D camera for agricultural automation. Sensors (Basel) **15:** 20463–20479

Zhou Y, Srinivasan S, Mirnezami SV, Kusmec A, Fu Q, Attigala L, Salas Fernandez MG, Ganapathysubramanian B, Schnable PS (2019) Semiautomated feature extraction from RGB images for sorghum panicle architecture GWAS. Plant Physiol **179:** 24–37

Zhu J, Ingram PA, Benfey PN, Elich T (2011) From lab to field, new approaches to phenotyping root system architecture. Curr Opin Plant Biol **14:** 310–317

Zhu X, Zhu M, Ren H (2018) Method of plant leaf recognition based on improved deep convolutional neural network. Cogn Syst Res **52:** 223–233