# Sorghum Segmentation by Skeleton Extraction

Mathieu Gaillard[1], Chenyong Miao[2], James Schnable[2], and Bedrich Benes[1(✉)]

[1] Purdue University, West Lafayette, IN, USA
bbenes@purdue.edu
[2] University of Nebraska-Lincoln, Lincoln, NE, USA
schnable@unl.edu
http://hpcg.purdue.edu, https://schnablelab.org/

**Abstract.** Recently, several high-throughput phenotyping facilities have been established that allow for an automated collection of multiple view images of a large number of plants over time. One of the key problems in phenotyping is identifying individual plant organs such as leaves, stems, or roots. We introduced a novel algorithm that uses a 3D segmented plant on its input by using a voxel carving algorithm, and separates the plant into leaves and stems. Our algorithm first uses voxel thinning that generates a first approximation of the plant 3D skeleton. The skeleton is transformed into a mathematical tree by comparing and assessing paths from each leaf or stem tip to the plant root and pruned by using biologically inspired features, fed into a machine learning classifier, leading to a skeleton that corresponds to the input plant. The final skeleton is then used to identify the plant organs and segment voxels. We validated our system on 20 different plants, each represented in a voxel array of a resolution $512^3$, and the segmentation was executed in under one minute, making our algorithm suitable for the processing of large amounts of plants.

**Keywords:** 3D plant reconstruction · Phenotyping · Sorghum · Skeleton extraction · Segmentation

## 1 Introduction

The architecture of plant organs such as leaves, stems, roots, and buds, plays a significant role in determining plant growth, health, and yield. Different individuals of the same species growing in the same environment will exhibit considerable differences in architectural traits due to genetic differences. Mapping and identifying the genes which control variation in plant architecture is a critical

**Electronic supplementary material** The online version of this chapter (https://doi.org/10.1007/978-3-030-65414-6_21) contains supplementary material, which is available to authorized users.


A. Bartoli and A. Fusiello (Eds.): ECCV 2020 Workshops, LNCS 12540, pp. 296–311, 2020.
https://doi.org/10.1007/978-3-030-65414-6_21

step in breeding new crop varieties that produce more food, use resources more efficiently, and are more resilient to changing environments.

Mapping genes controlling within-species variation in plant architectural traits requires quantifying these traits, which requires the identification of plant organs. Different approaches have been pioneered and applied in rosette plants (*e.g.,* arabidopsis, brassicas, etc.) using top-down photos and grain crops using side view photos. Skeletonization converts segmented images into graphs that uniquely map onto the plant organs and quantifies architectural traits [3,4]. The skeletonization of 2D side view images followed by computing on the resulting skeleton is useful in segmenting individual leaves of maize and sorghum plants, but is not robust to leaves that intersect from a single view [4].

Several high throughput phenotyping technology facilities have been built that allow for controlled viewing and photographing of hundreds of plants [8,10, 13]. The plants grow in a greenhouse, and they are regularly and thoroughly automatically transported by using conveyor belts into imaging chambers where they are photographed from several angles. Various methods for 3D reconstruction of plants from these controlled environments have been developed [11,17]. We build upon the work of Gaillard et al. [9] that reconstructs several photographs into a voxel grid that approximates the plant's geometry. While the voxel grid is an excellent plant approximation, it does not carry semantic information, such as the number of leaves, curvature, etc.

We present a novel method for extracting high-level semantic features from discrete volumetric arrays obtained by using the voxel carving algorithm into skeletons. The state of the art skeletonization algorithms perform poorly on plants. We claim a contribution in a machine learning-based algorithm that improves the plant skeletonization, allowing us to keep only the essential parts: stem and leaves. These, in turn, are used for efficient segmentation of the input voxel grid. An example in Fig. 1 shows the input voxel grid, the extracted skeleton, and the segmented output (open in Adobe Acrobat for animation).

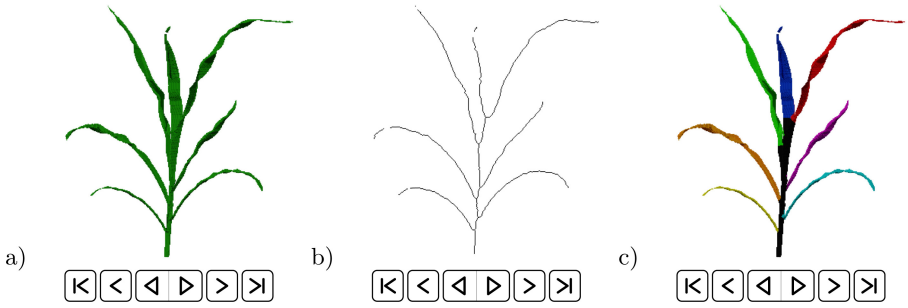


**Fig. 1. Please open in Adobe Acrobat to see the animations.** a) Input voxel grid, b) extracted skeleton, and c) segmented plant organs. (See Supplementary Material)

## 2 Related Work

The space carving algorithm [16] retrieves 3D voxel positions that correspond to an input object represented by a set of images. However, this method requires precise calibration to retrieve the 3D positions. Also, an increasing number of images provides better results. This algorithm is highly suitable for controlled environments. Novel algorithms and extensions add reconstruction of small plant parts [12,14,15] or a hierarchical space enumeration by octrees [19]. Our work builds on top of the work of Gaillard et al. [9] that provides 3D voxel reconstruction of Sorghum plants by using an improved voxel carving algorithm. Overall 336 Sorghum plants grown in the UNL phenotyping facility [10] were reconstructed from only six RGB images (five side and one from the top) per plant.

3D reconstruction algorithms working in voxel space are not suitable for direct measurements of organ-level features such as leaf size [12]. Skeletons [7] provide an intuitive and simplified information about the topology of the shape they represent. Skeletons can significantly help shape segmentation by guiding it. However, while 2D skeletonization is well-understood and studied, the skeletonization of 3D shapes is much more complicated. First, 3D shapes can be represented in different ways: for example, as a triangle mesh or a voxel grid. Various data give rise to multiple types of 3D skeletons: surface skeletons have a good correspondence to the input shape but are slower to compute and more challenging to analyze. Curves skeletons do not strictly follow the mathematical properties of skeletons, but provide better shape analysis capabilities as they decompose shapes into a set of 1D curves. They are of particular interest for plants, which often have a tubular shape. For a detailed survey on 3D skeletonization, we refer the reader to [20], in which Figure 21 is of particular interest to understand the difference between surface and curve skeletons. Existing skeletonization algorithms [1], despite their strong mathematical properties, produce noisy results on plants, and many specialized methods for plant skeletonization have been developed. The work of [14] segments the 3D surface of a voxel grid using the eigenvalues of the second-moments tensor. Also, a database of predefined leaves has been fit on the skeletons extracted from the 2D views of the plant in [21]. Golbach et al. [12] use a flood fill algorithm to identify the stem. They measure the spread of voxels added in each iteration during the graph traversal to detect branches. Scharr et al. [19] find clusters of voxels in horizontal slices of the plant from top to bottom. When merging two clusters, special rules are applied to keep track of leaves. The most similar approach to ours [1] uses a skeletonization algorithm and then filter skeleton branches based on their reprojections. Wu et al. [22,23] use a Laplacian contraction, a generic skeletonization algorithm, to shrink a point cloud and then post-process it to output measurements. Xiang et al. [24] also skeletonize and segment a point cloud to measure traits in Sorghum plants. Our algorithm can be thought of as a post-process of a segmentation from voxels. We attempt to improve the skeleton with a particular focus on the precise branching point used in a follow-up segmentation.

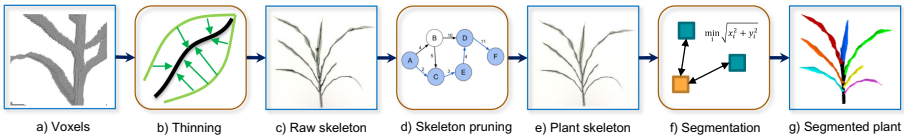


**Fig. 2. System overview:** (round boxes are processes and squared boxes are data): a) the 3D model of the plant represented as a set of voxels is converted into a raw skeleton by using b) the voxel thinning algorithm. c) The raw skeleton is d) filtered to remove noise resulting in e) a smooth plant skeleton. The input voxels are then f) segmented by using the skeleton resulting in g) a segmented set of voxels representing plant organs.

## 3 Overview

Our method works in three steps shown in Fig. 2: 1) thinning 2) skeleton filtering, and 3) segmentation.

Our algorithm's input is a 3D reconstruction of a sorghum plant in the form of a set of voxels. Our algorithm has two outputs: e) is the plant skeleton, and g) is the segmented plant represented as sets of voxels corresponding to plant organs: the stem and individual leaves. Both the plant and the skeleton are represented in a voxel grid at the same resolution. The segmentation assigns a unique identifier to each voxel.

**Thinning:** Removes voxels from the input voxel set until only the raw skeleton remains (Sect. 4.1). The input to this step is a plant represented in a voxel grid. Here we use the output of a voxel carving algorithm [9]. Although a smooth, dense, and connected set of voxels provides good input, we designed our algorithm to be robust against noise to successfully process reconstructed plants that do not strictly adhere to these conditions.

**Skeleton Pruning:** The output of the previous step is a raw skeleton that is equidistant to boundaries. However, the noisy input leads to incorrect results if used directly for leaf counting and plant segmentation. We introduce a novel bio-inspired algorithm in Sect. 4.2 to prune the skeleton removing noisy parts and keeping only the skeleton corresponding to actual leaves.

**Segmentation:** uses the skeleton calculated from the previous step to generate a segmented plant. We post-process the skeleton to identify the stem and each unique leaf (see Sect. 5), and we assign all voxels from the full plant to its nearest segment and represent them by a unique identifier.

**Terminology:** Let $\mathcal{V}$ denote the binary voxel grid that contains the reconstructed plant (we use resolution of $512^3$ voxels). We refer to each voxel as $v_{i,j,k}$ where $0 \leq i, j, k \leq 511$ denote its discrete coordinates. A voxel $v_{i,j,k} = 1$ belongs to the reconstructed plant, zero voxels identify an empty space.

The raw (generic) skeleton is a set of voxels in $\mathcal{V}$ output by a thinning algorithm, is denoted by $\mathcal{S}_{raw}$. The set of endpoints in $\mathcal{S}_{raw}$ is denoted $\mathcal{E} = \{E_i\}$ with

$E_i \in \mathcal{V}$. The endpoint that designates the plant root is denoted $v_0 \in \mathcal{E}$, and $R_{pot}$ is the radius of the pot. The collection of paths $P_i$ in $\mathcal{S}_{raw}$ from endpoints $E_i$ to the root $v_0$ is denoted $\mathcal{P} = \{P_i\}$. During the pruning step, paths $P_i$ are labeled in $\mathcal{K} = \{K_i\}$ as either kept ($K_i = 1$) or discarded ($K_i = 0$). The collection of paths that are retained after pruning is denoted $\mathcal{P}^*$. Finally, the skeleton, which is a set of voxels in $\mathcal{V}$ output by our algorithm, is denoted by $\mathcal{S}$.

## 4 Skeletonization

### 4.1 Thinning

The input to the first step is the voxel grid $\mathcal{V}$ model of the plant. The voxels are converted to a raw skeleton by using a thinning algorithm; in particular, we use the 3D critical kernel thinning algorithm from [6] implemented in the `DGtal` library [5]. We chose the `1isthmus` thinning algorithm because it is the fastest algorithm in the library that outputs a curve skeleton. The algorithm is left to default `dmax`, and the persistence is set to one. In principle, higher persistence values decrease skeleton noise. However, for our dataset, this was not always the case, and our domain-specific pruning algorithm performed better.

Although the critical kernel thinning algorithm is mathematically correct and outputs a skeleton that accurately represents the topology of the plant from the input voxel grid $\mathcal{V}$, it also responds to the inevitable noise in the input leading to a noisy skeleton that cannot be directly used for leaf counting and segmentation. As shown in Fig. 3 column c), the skeleton is noisy, includes small branches and some big lumps, mostly near the stem. Also, it is not guaranteed to have a tree topology; in other words, it may contain loops. Therefore, we further process the raw skeleton by pruning the unnecessary branches and filtering voxels in lumps.

### 4.2 Skeleton Pruning

A desired output of this algorithm would be one skeleton curve for the stem and one per each leaf. Moreover, the branching pattern for sorghum plants should only have one level of hierarchy *i.e.,* the skeleton should not contain T-junctions unless they are located between the stem and a leaf. Based on these biological observations, our pruning method works in four steps: 1) endpoint identification, 2) root identification, 3) branch finding, and 4) branch pruning.

**Endpoints Identification:** We identify all endpoints $\mathcal{E} = \{E_i\}$ in the skeleton obtained by thinning and denoted by $\mathcal{S}_{raw}$. Endpoints are voxels having at most one neighbor:

$$\mathcal{E} = \left\{ v_{i,j,k} \in \mathcal{V} \quad | \sum_{|(i,j,k)-(x,y,z)|_\infty = 1} v_{x,y,z} \leq 1 \right\} \tag{1}$$

and this includes isolated voxels without neighbors and voxels at the end of skeleton branches. The set of endpoints is likely to be located on the leaf ends, but it also includes some false positives.

**Root identification** finds the plant's root $v_0$ that is the starting point of the stem. We define the root as the lowest endpoint that is located within the radius of the pot $R_{pot}$:

$$v_0 = \underset{\substack{v_{i,j,k} \in \mathcal{E} \\ d_{e_z}(v_{i,j,k}) < R_{pot}}}{\arg\min} (k), \tag{2}$$

where $d_{e_z}(v_{i,j,k})$ is the distance from $v_{i,j,k}$ to $e_z$, the $z$-axis. We constrain the root voxel within the pot because we noticed that some leaf tips outside the container could extend below the plant/soil interface level. The lowest endpoint without this restriction could be a low-hanging leaf tip instead of the actual root.

**Branch Finding:** The raw skeleton $\mathcal{S}_{raw}$ resulting from thinning is not guaranteed to have a tree topology: it can contain loops. To convert the graph into a tree and discard some noisy parts, we run a single source shortest path on $\mathcal{S}_{raw}$ starting from the root voxel $v_0$. The output is the shortest path from every voxel of $\mathcal{S}_{raw}$ to the root voxel $v_0$. To accommodate potential disconnections in $\mathcal{S}_{raw}$, we allow the shortest path algorithm to jump between two voxels even if they are not connected. We use the Dijkstra algorithm and add a penalty on the distance if two voxels are not connected.

The distance $d(v_1, v_2)$ between voxels $v_1$ and $v_2$ is computed as follow: if $v_2$ is in the 26-connected neighborhood of $v_1$, then $d(v_1, v_2) = 1$. If $v_2$ is within a cube of length 48 around $v_1$, we still consider them as connected and the penalized distance is computed as a function of their Manhattan distance *i.e.*, $|v_1 - v_2|$:

$$d(v_1, v_2) = \begin{cases} 1 & if \quad \|v_1 - v_2\|_\infty = 1 \\ \frac{|v_1 - v_2|(1 + |v_1 - v_2|)}{2} & if \quad \|v_1 - v_2\|_\infty \leq 24. \end{cases} \tag{3}$$

By still considering $v_1$ and $v_2$ connected but with a distance penalty, we allow the algorithm to connect to the nearest voxel when there is a small discontinuity between two parts of the plant. We empirically chose a maximum distance of 24 in infinity norm for jumping from a voxel to another. It provides a good compromise between the computation time and the distance to which we want to connect two plant parts.

The single-source shortest path algorithm generates a tree of voxels starting from the root $v_0$. As explained above in the *Endpoints Identification* step and in Eq. (1) we may miss some leaf tips if there is a loop in $\mathcal{S}_{raw}$. Therefore, we update the list of endpoints based on the output of the Dijkstra algorithm. Any voxel that has no predecessor and has at most two neighbors in $\mathcal{S}_{raw}$ is added to the list of endpoints $\mathcal{E}$.

Finally, we output a list of shortest paths $\mathcal{P} = \{P_0, P_1, \dots\}$. The path $P_i$ starts from the endpoint $E_i$ and goes all the way down to the root voxel $v_0$, and every path potentially includes a plant leaf. We sort paths in $\mathcal{P}$ by descending length (longest leaf first). We also discard paths with a length of only one voxel because they cannot connect to the root voxel.

**Branch Pruning:** The collection of paths $\mathcal{P}$ represents a plant skeleton that does not include loops nor lumps, and it is less noisy than the raw skeleton $\mathcal{S}_{raw}$.

However, it may still include small spurious branches that do not correspond to any actual leaf. These branches are a likely result of artifacts and irregularities in the 3D reconstruction. They mostly consist of a small spike starting from the middle of a leaf and going to another direction.

This is mitigated by pruning the paths and keep only those that bring the most information. That is achieved by processing each path $P_i$, decreasing order of length, and considering every other shorter path $P_j$ (therefore $j > i$). When comparing two paths, $P_i$ and $P_j$, we compute a set of features and use a machine learning classifier to decide whether we keep or discard the branch $P_j$. We detail the different classifiers and the learning procedure we used in Sect. 6. We repeat the classification for each path in $\mathcal{P}$, making in total $n(n-1)/2$ comparisons, with $n$ being the number of paths in $\mathcal{P}$. The procedure is detailed in Algorithm 1. Paths that are not discarded are kept in a new list $\mathcal{P}^*$

$$\mathcal{P}^* = \{P_i \in \mathcal{P} \mid K_i = 1\}\,. \tag{4}$$

Finally, the union of voxels from all paths in $\mathcal{P}^*$ forms the plant skeleton $\mathcal{S}$. An example of such a skeleton can be seen in Fig. 3 b).

**Data**: $\mathcal{P}$ is a list of paths ordered by decreasing length
**Result**: $\mathcal{K}$ a label for each path stating that a path should be kept or discarded in the output skeleton.

```
/* This algorithm discards paths that are not relevant.                    */
K ← {1}_1^n; /* By default, we keep all paths.                              */
for i ∈ {1...n} do
    if K_i = 1 then
        for j ∈ {i+1...n} do
            if K_j = 1 then
                /* Run the classifier for the paths: P_i and P_j.          */
                if runClassifier(P_i, P_j) == false then
                    /* Let's discard P_j.                                  */
                    K_j ← 0;
                end
            end
        end
    end
end
```

**Algorithm 1:** Creation of the training data set $\mathcal{L}$ from an annotated skeleton

## 5 Segmentation

In this section, we detail how we segment the input set of voxel $\mathcal{V}$ into individual leaves by using the skeleton $\mathcal{S}$. This algorithm proceeds in two steps: we first segment the skeleton and then the full plant.

**Skeleton segmentation** segments each path in $\mathcal{P}^*$ from the skeleton $\mathcal{S}$ into a stem part and a leaf part. The stem is composed of all voxels that are shared between at least two paths. To find voxels that are shared, we compute the histogram of occurrences for each voxel of the skeleton $\mathcal{S}$. If a voxel appears at

least twice in $\mathcal{S}$, it is considered to be a part of the stem. Once the stem has been identified, we remove all stem voxels from all other paths in $\mathcal{P}^*$, and the remaining voxels correspond to the leaves. Although our approach works on well-reconstructed plants, by construction, we do not guarantee that the skeleton has the right topology with poorly reconstructed plants, which have merged leaves. Therefore, it is necessary to check whether the topology, *i.e.,* the stem should not have any T-junctions. We discuss this issue in Sect. 7. Moreover, we also consider the root to be inside the pot.

**Plant Segmentation:** We segment the full plant according to the skeleton $\mathcal{S}$. Each voxel $v_{i,j,k}$ of the plant is assigned to the nearest segment of the segmented skeleton $\mathcal{S}$. Figure 3 c) shows the final segmented plants.

## 6 Branch Classification

Below we explain how we designed and trained machine learning classifiers used in Sect. 4.2 to prune a raw plant skeleton. We trained three different classifiers for this task (linear SVM, SVM with RBF kernels, and a Multi-Layer Perceptron) and compared them.

### 6.1 Data Set

To train our classifiers, we manually annotated 100 plant skeletons by choosing from a pool of 351 sorghum plants. We decided to discard plants that are poorly reconstructed, including merged leaves, missing leaves, or excessive numbers of noise voxels (for example, leaf-like reconstruction artifacts). Since we feed our machine learning classifiers with handcrafted features and are not learning the data representation, we do not use any form of data augmentation.

For each skeleton, we looked at each path $P_i \in \mathcal{P}$ before the pruning step (see Sect. 4.2) and we annotated it in $\mathcal{K}$ by marking whether it should be kept ($K_i = 1$) or discarded ($K_i = 0$) in the final plant skeleton $\mathcal{P}^*$. In our data set, $\mathcal{P}$ contains 108 paths on average before pruning, and $\mathcal{P}^*$ contains only seven paths after pruning. Annotation is a tedious task, and completing annotations for 100 plant skeletons took about a week of effort by a trained expert.

To train the classifiers, we transform the data set into a list $\mathcal{L}$ of triplets $[P_i, P_j, keep]$, with a binary label $keep$ that indicates whether $P_j$ should be kept or discarded when compared to $P_i$. We detail the method to generate triplets in Algorithm 2. The main idea is that a path that is kept should never discard another shorter path that is kept. Moreover, only one longer path is needed to reject a given path that is not supposed to be kept. When a path $P_j$ has to be discarded, we only add the triplet with $P_i$, the shortest longer path that is the most similar to $P_j$. By doing so, we guarantee that $P_i$ is discarded at least once. Any other decision made regarding $P_i$ is not essential.

Once the data set $\mathcal{L}$ of triplets is built, we compute a set of features for each pair of paths that are input into the classifier. After trying different combinations, the set of features we retained includes: 1) the number $s_{ij}$ of voxels that $P_i$

**Data**: $\mathcal{P}$ is a list of paths, ordered by increasing length and $\mathcal{K}$ is a label for each path stating that a path should be kept or discarded in the output skeleton.
**Result**: $\mathcal{L}$ is the set of triplets $[P_i, P_j, keep]$ for training.

```
for j ∈ {1...n} do
    if K_j = 1 then
        for i ∈ {j+1...n} do
            if K_i = 1 then
                L ← L ∪ P_i, P_j, true;
            end
        end
    else
        /* We look for the shortest most similar path longer than P_j
           That is, the first path P_i that maximizes s_ij.                    */
        k ← j;
        mostNbCommonVoxels ← 0;
        for i ∈ {j+1...n} do
            if K_i = 1 then
                s_ij ← computeNumberCommonVoxel(P_i, P_j);
                if s_ij > mostNbCommonVoxels then
                    mostNbCommonVoxels ← s_ij;
                    k ← i;
                end
            end
        end
        if k > j then
            L ← L ∪ {P_k, P_j, true};
        end
    end
end
```

**Algorithm 2:** Creating the training data set $\mathcal{L}$ from an annotated skeleton.

and $P_j$ have in common 2) the number $e_j = length(P_j) - s_{ij}$ of voxels that are included only in $P_j$, and 3) the ratio of the shorter path $P_j$ that is shared with $P_i$: $p_{ij} = s_{ij}/length(P_j)$.

Our interpretation of the role of the features is that: 1) estimates the position of a branch in the plant (bottom or top), 2) estimates the length of a branch (short or long), and 3) estimates the information brought by a branch compared to a longer branch.

Classes in our data set are unbalanced, and $\mathcal{L}$ contains significantly more pairs of branches that should be discarded than couples to keep. Thus, we balance the data set by applying weights on each sample.

We randomly split the 100 annotated skeletons into 64 for training, 16 for validation, and 20 for testing. As a reference, on the one hand, the 80 skeletons in the training and validation set generate 9,877 triplets. On the other hand, the test set generates 2,422 triplets.

### 6.2 Classifiers

The input to our classifier is a set of three features computed on a pair of paths $P_i$ and $P_j$ in the raw skeleton $\mathcal{S}_{raw}$. The output is a binary decision: true means that we keep path $P_j$, false means that we discard it, and we tested three different classifiers. We used machine learning models with handcrafted features because we do not have an extensive data set to allow automatic feature learning. After

all, manual labeling is a very time-consuming process. Moreover, it is easier for a human to interpret the predictions.

We tested a **linear SVM** based on the shared proportion feature $p_{ij}$. This classifier finds the cutoff value of $p_{ij}$ that linearly separates branches that should be kept from those that should be discarded. Although simple, it performs relatively well and can be easily implemented as a single *if* condition in the code. Experimentally, we found the cutoff value to be $p_{ij} = 0.65$, and we used this classifier as a reference for more complicated models.

The second model we tested is a **SVM with RBF kernels** based on the three features. We trained by using grid search and ten-fold cross-validation on the training set.

The third classifier is a **Multi Layer Perceptron** containing an input layer with three units, followed by two eight-units hidden layers, and finally a one-unit output layer. Input features are normalized, and all units use a Sigmoid activation function. For training we use the batch `RPROP` algorithm [18] implemented in the OpenCV [2] library.

### 6.3 Evaluation

We run the branch pruning algorithm (Algorithm 1 in Sect. 4.2) on the test set of 20 skeletons, and we used information retrieval evaluation measures to compare the different classifiers. For each skeleton, we compared the set of branches selected by the branch pruning algorithm to the ground truth $\mathcal{K}$. Let's denote by $tp$ (true positive) the number of successfully reconstructed branches *i.e.,* selected branches that are in $\mathcal{K}$; let's further denote $fp$ (false positive) the number of branches that were selected even if they were not in $\mathcal{K}$; we also denote $fn$ (false negative) the number of branches that were not selected even if they were in $\mathcal{K}$. Finally, $tn$ (true negative) is the number of successfully discarded branches. We compute the precision

$$precision = \frac{tp}{tp + fp}, \tag{5}$$

which is the proportion of selected branches that are relevant, recall

$$recall = \frac{tp}{tp + fn}, \tag{6}$$

is the proportion of relevant branches that are selected, and the F-measure

$$Fmeasure = 2 \cdot \frac{precision \cdot recall}{precision + recall}, \tag{7}$$

is the harmonic mean of precision and recall and gives an overall score for the quality of the branch pruning.

Evaluations of our three different classifiers on the training and validation set are given in Table 1 and on the test set are given in Table 2. While the three classifiers performed comparatively, the RBF SVM classifier provided the best results based on the evaluation metrics used in this paper.

**Table 1.** Evaluation of the classifiers on the Training + Validation set.

| | tp | fp | fn | tn | Precision | Recall | F-measure |
|---|---|---|---|---|---|---|---|
| Linear SVM | 535 | 27 | 29 | 8,057 | 0.952 | 0.949 | 0.950 |
| RBF SVM | 549 | 15 | 15 | 8,069 | 0.973 | 0.973 | 0.973 |
| MLP | 551 | 24 | 13 | 8,060 | 0.958 | 0.977 | 0.968 |

**Table 2.** Evaluation of the classifiers on the Test set.

| | tp | fp | fn | tn | Precision | Recall | F-measure |
|---|---|---|---|---|---|---|---|
| Linear SVM | 131 | 8 | 8 | 1,990 | 0.942 | 0.942 | 0.942 |
| RBF SVM | 135 | 5 | 4 | 1,993 | 0.964 | 0.971 | 0.968 |
| MLP | 136 | 7 | 3 | 1,991 | 0.951 | 0.978 | 0.965 |

## 7 Results

We have implemented our method in `C++` and run it on a workstation equipped with an Intel Xeon W-2145. We did not use any GPU acceleration, and, on average, the thinning step takes 43 s, and the pruning and the segmentation steps take 10 s per plant.

We use the 20 plants in the test set to validate our method. We visually inspected each plant to make sure that the leaves are at the right location and the topology of the skeleton is correct. Table 2 from Sect. 6.3 shows that 97.1% of branches were successfully reconstructed and among reconstructed branches, 96.4% were relevant. Note that the RBF SVM classifier misclassified only 9 branches out of 2,128 (135 true positives and 1,993 true negatives).

We used two measures to automatically get an insight into how well a plant is skeletonized. 1) We compute the maximum distance from any voxel in $\mathcal{V}$ to the skeleton. If this distance is too high (*e.g.,* more than 5 cm), it indicates that one leaf is likely missing in the skeleton. If this distance is small, it could suggest that some spurious branches were not discarded. 2) We look at the skeleton's topology with the assumption that the stem should not include any T-junctions. If these two measures fail, the plant needs to be visually inspected.

One plant (1) from the training set and four plants (2, 3, 4, 5) from the test set are shown in Fig. 3 and additional results are in Fig. 4. Column (b) shows the skeleton output by our method, and segmented plants are in column (c).

**Failure Cases:** Some plants have a wooden stick in their pot, and in some cases, a leaf happens to be reaching under the lowest part of the stem. If such features are reconstructed within the container radius $R_{pot}$, the algorithm will erroneously identify them as the root voxel $v_0$. Although it does not affect counting and measuring leaves, we get the wrong length for the stem.

If the input voxel plant is poorly reconstructed and contains leaves that are merged because they are too close, the algorithm for the shortest path computation can jump between the leaves, because the overall cost is cheaper than

staying on the correct leaf. In this case, the two leaves will have the same starting point but two distinct tips. Depending on the rest of the plant, its topology may even become wrong with a T-junction on the stem. This case is shown in Fig. 3 (2), where the top left leaf is merged to the top leaf. This failure case is not critical when counting leaves, but causes errors in length estimation, by up to a few centimeters.

If the branch classifier makes an erroneous decision during the skeleton pruning step, a leaf can be missed because it has been filtered, as shown in Fig. 3 (3). Conversely, an extra leaf can be left after the pruning step, as can be seen in Fig. 3 (4). It happens especially when voxel carving with a low number of views has been used because it creates leaf-like artifacts.

If a plant is poorly reconstructed and some voxels at the tips are missed, our method, which is only based on input voxels from $\mathcal{V}$, will only output a truncated skeleton because it does not take in account 2D pictures of the plant. This is the reason why skeletons in column (b) of Fig. 3 do not always extend to the end of leaves, because our input voxel plants are not entirely reconstructed. This problem is inherent to voxel carving. To mitigate it, we would need to improve calibration and use multiple cameras instead of a single camera with a turntable.

## 8 Conclusions and Future Work

We have introduced a novel method for generating skeletons and segmentation of voxel data generated by the voxel carving algorithm that is commonly used by phenotyping facilities to create 3D approximations of measured plants. We tested three different classifiers to prune leaves from raw plant skeletons, and our validations using manually labeled data show that the RBF SVM classifier provided the best results. Furthermore, we used the generated skeleton to segment the input voxel data into the plant's leaves and stem. Each voxel is assigned an identifier that defines the plant organ it belongs.

A limitation of our work, and a possible extension as a future work, is the lack of labeled data. The input data is complicated, and an educated person can use only manually label branches. Having an automatic annotation tool would greatly help to expand the training set. It would be interesting to provide a pixel-based performance metric for the segmentation. We do not guarantee that the constructed skeleton's topology is correct because it can have T-junctions if the input plant is poorly reconstructed. Although it covers a vast majority of cases we encountered, we think it is possible to extend the algorithm with backtracking to ensure that the skeleton has a correct topology, even if it features merged leaves. Top and small leaves are hard to identify, as shown in Sect. 7. We could extend the algorithm and use many stages of the same plant over time to discriminate growing leaves from artifacts using a later stage of the plant where the difference is noticeable. Finally, starting from annotated voxels generated by our tool and corrected by experts, one could develop fully automatic machine-learning classifiers that would allow for fast and reliable segmentation at the voxel level.

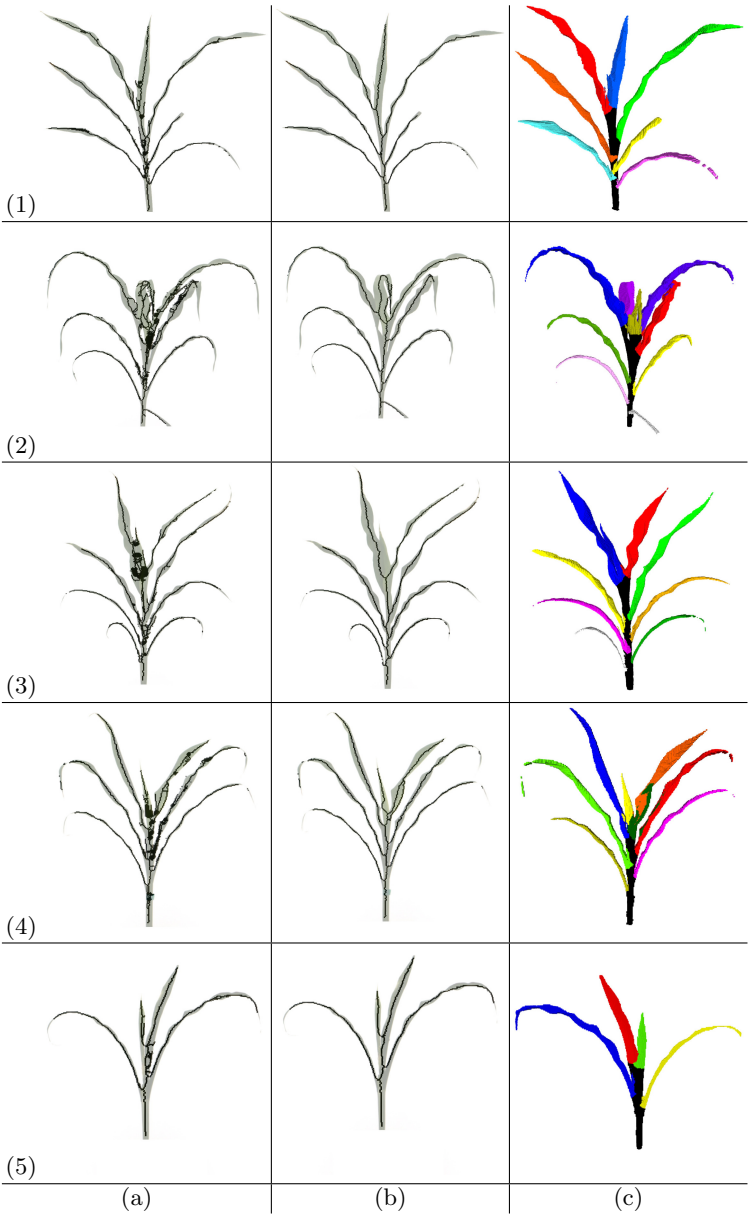


**Fig. 3. Columns:** (a) Raw skeleton resulting from thinning, (b) skeleton after the pruning step, and (c) segmented leaves and stem. (1) a correctly segmented plant, (2) a successfully segmented complex plant from the test set. The blue leaf (the top left one) is off because in the reconstructed plant, leaves are merged. (3) A plant from the test set with the top leaf missing. (4) Another plant from the test set with a spurious missing (see the dark green area between the orange and red leaves). (5) A correctly reconstructed small plant from the test set. (Color figure online)

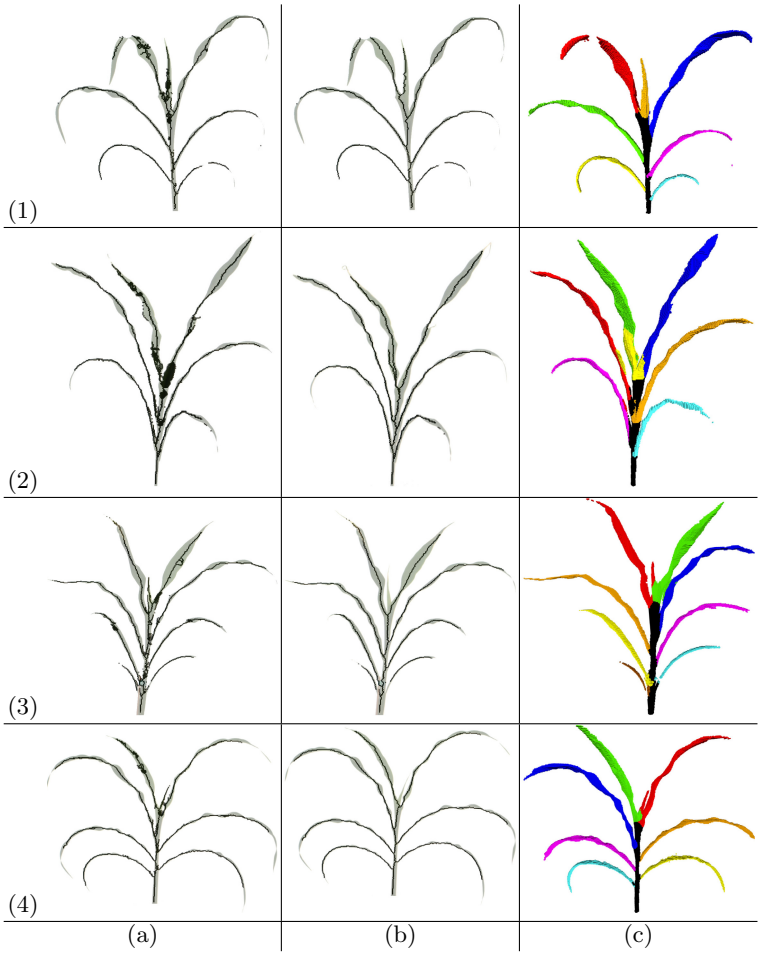


**Fig. 4. Additional examples** (a) Raw skeleton resulting from thinning, (b) skeleton after the pruning step, and (c) segmented leaves and stem.

**Acknowledgements.** This research was supported by the Foundation for Food and Agriculture Research Grant ID: 602757 to Benes and Schnable. The content of this publication is solely the responsibility of the authors and does not necessarily represent the official views of the foundation for Food and Agriculture Research. This research was supported by the Office of Science (BER), U.S. Department of Energy, Grant no. DE-SC0020355 to Schnable. This work was supported by a National Science Foundation Award (OIA-1826781) to Schnable.

## References


1. Artzet, S., et al.: Phenomenal: an automatic open source library for 3D shoot architecture reconstruction and analysis for image-based plant phenotyping. bioRxiv (2019). https://doi.org/10.1101/805739. https://www.biorxiv.org/content/early/2019/10/21/805739
2. Bradski, G.: The OpenCV library. Dr Dobb's J. Softw. Tools **25**, 120–125 (2000)
3. Bucksch, A.: A practical introduction to skeletons for the plant sciences. Appl. Plant Sci. **2**(8), 1400005 (2014)
4. Choudhury, S.D., Bashyam, S., Qiu, Y., Samal, A., Awada, T.: Holistic and component plant phenotyping using temporal image sequence. Plant Methods **14**(1), 35 (2018)
5. Coeurjolly, D., et al.: DGtal-team/dgtal: Release 1.0, March 2019. https://doi.org/10.5281/zenodo.2611275
6. Couprie, M., Bertrand, G.: Asymmetric parallel 3D thinning scheme and algorithms based on isthmuses. Pattern Recogn. Lett. **76**, 22–31 (2016)
7. Du, S., Lindenbergh, R., Ledoux, H., Stoter, J., Nan, L.: AdTree: accurate, detailed, and automatic modelling of laser-scanned trees. Remote Sen. **11**(18), 2074 (2019)
8. Fahlgren, N., et al.: A versatile phenotyping system and analytics platform reveals diverse temporal responses to water availability in Setaria. Mol. Plant **8**(10), 1520–1535 (2015)
9. Gaillard, M., Miao, C., Schnable, J.C., Benes, B.: Voxel carving-based 3D reconstruction of sorghum identifies genetic determinants of light interception efficiency. Plant Direct **4**(10), e00255 (2020). https://doi.org/10.1002/pld3.255
10. Ge, Y., Bai, G., Stoerger, V., Schnable, J.C.: Temporal dynamics of maize plant growth, water use, and leaf water content using automated high throughput RGB and hyperspectral imaging. Comput. Electron. Agric. **127**, 625–632 (2016)
11. Gehan, M.A., et al.: Plantcv v2: image analysis software for high-throughput plant phenotyping. PeerJ **5**, e4088 (2017)
12. Golbach, F., Kootstra, G., Damjanovic, S., Otten, G., Zedde, R.: Validation of plant part measurements using a 3D reconstruction method suitable for high-throughput seedling phenotyping. Mach. Vis. Appl. **27**(5), 663–680 (2016)
13. Junker, A., et al.: Optimizing experimental procedures for quantitative evaluation of crop plant performance in high throughput phenotyping systems. Front. Plant Sci. **5**, 770 (2015)
14. Klodt, M., Cremers, D.: High-resolution plant shape measurements from multi-view stereo reconstruction. In: Agapito, L., Bronstein, M.M., Rother, C. (eds.) ECCV 2014. LNCS, vol. 8928, pp. 174–184. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16220-1_13
15. Koenderink, N., Wigham, M., Golbach, F., Otten, G., Gerlich, R., van de Zedde, H.: Marvin: high speed 3d imaging for seedling classification. In: van Henten, E., Goense, D., Lokhorst, C. (eds.) Precision Agriculture 2009: Papers Presented at the 7th European Conference on Precision Agriculture, Wageningen, The Netherlands, 6–8 July 2009, pp. 279–286. Wageningen Academic Publishers (2009)
16. Kutulakos, K.N., Seitz, S.M.: A theory of shape by space carving. Int. J. Comput. Vision **38**(3), 199–218 (2000)
17. Lobet, G.: Image analysis in plant sciences: publish then perish. Trends Plant Sci. **22**(7), 559–566 (2017)
18. Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In: IEEE International Conference on Neural Networks, pp. 586–591. IEEE (1993)

19. Scharr, H., Briese, C., Embgenbroich, P., Fischbach, A., Fiorani, F., Müller-Linow, M.: Fast high resolution volume carving for 3D plant shoot reconstruction. Front. Plant Sci. **8**, 1680 (2017). https://doi.org/10.3389/fpls.2017.01680. https://www.frontiersin.org/article/10.3389/fpls.2017.01680
20. Tagliasacchi, A., Delame, T., Spagnuolo, M., Amenta, N., Telea, A.: 3D skeletons: a state-of-the-art report. In: Computer Graphics Forum, vol. 35, pp. 573–597. Wiley Online Library (2016)
21. Ward, B., et al..: A model-based approach to recovering the structure of a plant from images (2015). http://search.proquest.com/docview/2081688123/
22. Wu, S., et al.: MVS-pheno: a portable and low-cost phenotyping platform for maize shoots using multiview stereo 3D reconstruction. Plant Phenomics 2020 (2020). https://doaj.org/article/bd4ae8082b0c45c2a1d4c6ff935d9ff1
23. Wu, S., et al.: An accurate skeleton extraction approach from 3D point clouds of maize plants. Front. Plant Sci. **10** (2019)
24. Xiang, L., Bao, Y., Tang, L., Ortiz, D., Salas-Fernandez, M.G.: Automated morphological traits extraction for sorghum plants via 3D point cloud data analysis. Comput. Electron. Agric. **162**, 951–961 (2019)