Leaf Segmentation By Functional Modeling

Yuhao Chen Sriram Baireddy Enyu Cai Changye Yang Edward J. Delp
Video and Image Processing Laboratory (VIPER)
School of Electrical and Computer Engineering
Purdue University
West Lafayette, Indiana, USA

Abstract

The use of Unmanned Aerial Vehicles (UAVs) is a recent trend in field based plant phenotyping data collection. However, UAVs often provide low spatial resolution images when flying at high altitudes. This can be an issue when extracting individual leaves from these images. Leaf segmentation is even more challenging because of densely overlapping leaves. Segmentation of leaf instances in the UAV images can be used to measure various phenotypic traits such as leaf length, maximum leaf width, and leaf area index. Successful leaf segmentation accurately detects leaf edges. Popular deep neural network approaches have loss functions that do not consider the spatial accuracy of the segmentation near an object's edge. This paper proposes a shape-based leaf segmentation method that segments leaves using continuous functions and produces precise contours for the leaf edges. Experimental results prove the feasibility of the method and demonstrate better performance than the Mask R-CNN.

1. Introduction and Related Work

Plant phenotyping is a set of methodologies used to observe, measure, and analyze the relationship and impact of genetic and environmental factors on the growth, yield, and physical traits of the plant. Phenotyping is used by plant scientists to select plant breeds based on desired traits and environments. Phenotypic traits include, but are not limited to, leaf count, leaf angle, maximum leaf width, leaf area index [1], canopy closure, plant and leaf morphology, plant and leaf color, leaf appearance rate, ear heights, and plant dry weight. Traditional phenotyping is labor intensive and destructive. Measuring and collecting phenotypic traits involves extracting plants from the field, laying them on a table and recording measurements using various devices [2]. In contrast, modern phenotyping systems using various sensors, including imaging sensors, provide the capability of high-throughput phenotyping [3, 4, 5, 6], which drastically reduces labor cost.

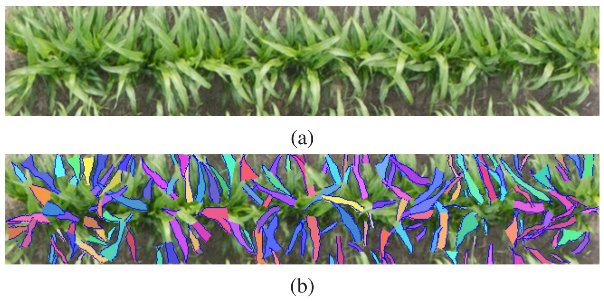


Figure 1: a) An image of a single row plot of crops acquired from an altitude of 50 meters. b) Leaf segmentation result generated by our method, where leaves are colored randomly.

Among the different types of modern phenotyping systems, the use of Unmanned Aerial Vehicles (UAVs) carrying various sensors is attractive because they are portable, cost-effective, non-invasive, and relatively easy to use [4]. One issue with collecting images using UAVs is that to be able to measure properties of the plants on the ground one must geometrically rectify the images and form them into mosaics [7]. Another issue is that due to the camera's field of view, the higher the UAV flies, the more ground area is covered per image and hence, the images have low spatial resolution. Lower altitude flights provide higher resolution images. Flying at a higher altitude reduces the number of flights needed because more of the crop field can be "seen" by the UAV.

Leaves serve as the main site for photosynthesis within a plant. They contribute to many phenotypic traits, such as leaf area, leaf length, maximum leaf width, and leaf area index. In commercial or research fields, plants are grown densely, which causes frequent leaf overlaps. As a result, the profile or edge of each leaf may not be completely visible. Being able to segment each leaf instance is essential to the estimation of leaf traits.

In summary, the problem we are addressing in this paper is instance segmentation for densely overlapped leaves in low resolution UAV images.

Recent work using Deep Neural Networks (DNN) has produced very promising results for image segmentation [8, 9]. In Faster R-CNN [10], Ren *et al.* use a Region Proposal Network (RPN) to propose Regions of Interest (RoI) on DNN generated feature maps. They then use Fully Connected (FC) layers to generate object bounding boxes and corresponding classes. In Mask R-CNN [8], He *et al.* introduce additional convolution layers parallel to the FC layers of Faster R-CNN. These additional layers generate object masks based on the feature maps and RoIs from the Faster R-CNN. In [9], Chen *et al.* modify Faster R-CNN to use semantic and direction features for instance segmentation, where direction features are the orientation of a pixel towards the center of its object. For leaf segmentation in [11], Chen *et al.* use plant centers to transform each plant image into a polar coordinate system and to segment each leaf with the assumption of triangular leaf shapes. In [12], Ward *et al.* generate synthetic leaves of rosette plants with different shapes, sizes and textures. The authors train a Mask R-CNN on both the real and synthetic data, and use the trained model to segment real leaves. In [13], Morris uses a Fully-Convolutional Pyramid Network to estimate tree leaf boundaries. Connected Components[14] and Watershed [15] are then applied to the output edge map to group leaf boundaries and obtain leaf segments.

From the overview of instance segmentation methods above DNNs play an important role in image segmentation. All DNN segmentation approaches are based on convolutional filters which extract features from images. Feature responses are propagated through the network to produce the final results. This design allows the spatial relation between features to be captured by the filters in a discrete manner. It requires a significant amount of labeled data as well as a large network to examine all the combinations of leaf size, shape, orientation, and curvatures. Synthetic data generation such as in [12, 16] overcomes the lack of labeled data, but well defined object models are necessary to create realistic images. In our case, the interaction between leaves causes them to bend, shift, and deform. Such interactions are difficult to model, especially with the growing patterns of plants. In addition, popular DNN loss functions for segmentation including binary loss [8], cross entropy loss [17], and dice loss [18] do not consider the spatial accuracy of the segmented object's edges. In our application where leaf shapes are crucial for measuring phenotypic traits (such as maximum width), DNNs are likely to produce segmented leaves with inaccurate and noisy edges because they do not have a spatial objective. Unlike convolutional filtering, modeling the leaf edges using a continuous function and then estimating parameters of the function from a set of leaf feature points will provide a better spatial representation of the leaf. The estimated functions can be used to search for pixels with weak feature responses and to predict missing feature points.

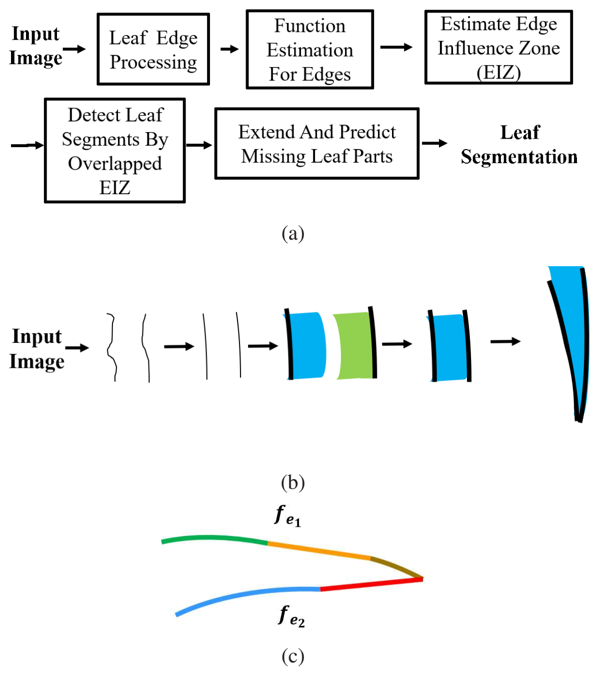

Input Image
Leaf Edge Processing
Function Estimation For Edges
Estimate Edge Influence Zone (EIZ)
Detect Leaf Segments By Overlapped EIZ
Extend And Predict Missing Leaf Parts
Leaf Segmentation
(a)
Input Image
(b)
f_{e_1}
f_{e_2}
(c)

Figure 2: a) Block diagram of proposed approach. b) Graphical block diagram. c) Piece-wise edge function f_{e_1} and f_{e_2}. Each color represents an estimated polynomial.

In this paper, we present a new approach to instance leaf segmentation by modeling leaf edges with continuous functions. Our method detects leaf segments by using semi-parallel features of leaf edges. It generates a leaf shape for each segment and uses the shape model to search for edges with weak responses and predict missing leaf parts. Each leaf is described by two edge functions, which can also be used to measure various phenotypic traits such as leaf area, leaf length, maximum leaf width, and leaf area index. Our method is compared to Mask R-CNN and achieves better results. Figure 1 shows an example result of our method. The plant used in this study is sorghum [Sorghum bicolor (L.) Moench] [19, 20].

2. Processing Structure and Assumptions

Our proposed method uses edge features and the semi-parallel relationship between the two leaf edges to detect, model, and predict leaf parts. Figure 2 shows the block diagrams of our proposed approach.

The following definitions and assumptions are used throughout the paper.

Each leaf has a tip and a tail. The tail is where the leaf meets the stem. The tip is where the two leaf

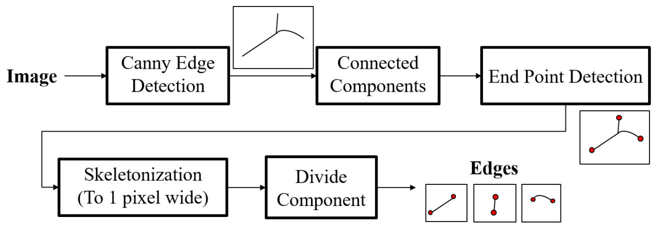


Figure 3: Block diagram for edge processing

edges/boundaries converge. A leaf can be represented by two piece-wise functions that describe the two leaf edges. The edge functions are estimated from edge pixels and are used for detecting leaf segments and extending leaf segment edges.

In computer graphics, a curve in 2D space can be represented by many functions and forms [21]. Among the representations, a polynomial is easy to estimate and model. In this paper, we use second-order polynomials for the leaf edge functions. Second-order polynomials in 2D are either convex or concave, but leaf edges have different orientations. Hence, we rotate the edge pixels before estimating the function. The rotation can be done in various ways. We use Principal Component Analysis (PCA) [22] due to ease of implementation.

A leaf edge can have different local curvatures along the edge. Describing an edge with only one polynomial function is sub-optimal. Therefore, each edge function is modeled as a piece-wise function (as shown in Figure 2c). The estimated function is the fundamental structure we use to represent a leaf edge. We shall define a “2D discrete function” as a set of pixels that describes (or estimates) a 2D function with errors caused only by spatial quantization of the pixels where the pixels in the set belong to the same leaf edge. We assume any location in an image is quantized to the center of a pixel. Thus, the maximum quantization error for each pixel is the distance from the pixel center to a pixel corner which is $\frac{\sqrt{2}}{2}$.

3. Leaf Edge Processing

The goal of this section is to process an image to extract 2D discrete functions. Figure 3 shows a block diagram of edge processing.

Given an RGB image, we first use the Canny Edge operator [23] to obtain an edge mask M_e. This edge mask can be refined by removing unwanted features from the background. We create a plant material mask M_p by thresholding the plant image in the Hue channel of the HSV color space. This exploits the "greenness" of the plant. This mask is used with the edge mask to form a refined edge mask M_r, where

$$M_r = M_e \wedge M_p \,. \tag{1}$$

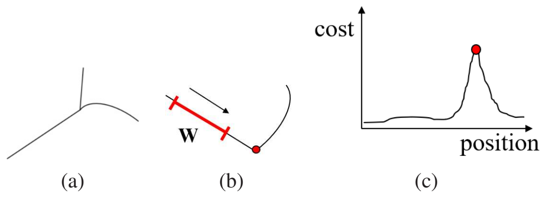


Figure 4: a) An example skeleton with branching structure. b) An example skeleton with edges from different leaves forming a single line. W is a sliding window for function estimation. The red dot is the ideal break point between the two piecewise functions. c) Function estimation cost vs. window position.

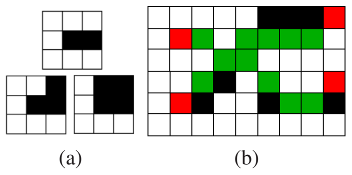


Figure 5: a) End point cases. b) Shortest path (red: end points, green: shortest paths, black: unvisited pixels).

We use Connected Components [24] on the refined edge mask M_r to group the connected edge pixels into sets, and we call each set an edge component. An edge component can be viewed as a graph and each pixel can be viewed as a node. Ideally, each detected edge pixel represents a leaf edge, but detected edge pixels may come from different leaf edges. There are two cases we consider. First, detected edge pixels from different leaves forming a branching structure, as shown in Figure 4a. Second, detected edge pixels from different leaves forming a single line (as shown in Figure 4b).

Our approach to these issues is to use Depth First Search (DFS) [25] on the edge components and to separate the branches during the search. Edges from different leaves in each branch are disjoined. This approach requires each edge component to be a one pixel wide skeleton. Skeletons wider than one pixel create cycles and ambiguous paths. We propose a skeletonization method in Section 3.1 to thin the edge components into one pixel wide.

3.1. Skeletonization

The edge components created by Connected Components are usually not one pixel wide. We need to thin the edge components to form skeleton structures. We use a shortest path [25] approach between end points of a skeleton to produce a map of node visiting frequency. We prioritize selecting a path through frequently visited nodes.

We first define the end points of a skeleton. An end point is a pixel that has one, two, or three adjacent pixels, where the adjacent pixels form a clique [25]. Figure 5a shows an example of an end point's neighborhood. We use Minimum Cost Path (MCP) [26] on all the end point pairs. The cost map C_s for MCP is updated after computing the path for each pair. This step prioritizes selecting an already explored route.

Let $E = \{e_1, e_2 ... e_{N_s}\}$ be the set of all end points in a skeleton, and N_s be the number of end points in the skeleton. Let M_s be the skeleton mask. The cost map C_s is initialized to:

$$C_s = \begin{cases} N_s^2 + 1 & \text{if } M_s > 0 \\ \text{inf} & \text{otherwise.} \end{cases} \tag{2}$$

The detailed implementation of the steps is shown in Procedure 1. Figure 5b shows an example of the skeletonization. Unvisited pixels are removed from the skeleton after all end point pairs have been examined.

```
Procedure 1: Skeletonization
Input : End point set: $E = \{e_1, e_2 ... e_{N_s}\}$ ,
        Number of end points: $N_s$ ,
        Skeleton mask: $M_s$
1  Initialize $C_s$ with $M_s$ and $N_s$
2  $S_{new} \leftarrow \{\}$
3  for $i = 1$ to $N_s$ do
4      for $j = i + 1$ to $N_s$ do
5          $P \leftarrow MCP(e_i, e_j, C_s)$ # compute shortest
            path $P$
6          for all location $x$ in $P$ do
7              $C_s(x) \leftarrow C_s(x) - 1$
8              $S_{new} \leftarrow S_{new} \cup \{x\}$
9          end
10     end
11 end
12 Return $S_{new}$
```

3.2. Skeleton Separation

First, we address the case of edges forming a branching structure, as shown in Figure 4a. We define an intersection to be a pixel in the skeleton with more than two neighbors. A two-neighbor pixel is a pixel that has two neighboring pixels. A branch is a set of connected pixels whose head and tail are either end points or intersections and the rest of its pixels are two-neighbor pixels. Branches are extracted by using a modified DFS on each skeleton (as shown in Procedure 2). As a result, we obtain a set of connected pixels organized in the pixel visit order of DFS. Each pixel set can also be viewed as a set of 2D discrete functions connected together.

We next address the case of edges from different leaves forming a single line, as shown in Figure 4b. We slide a window of size W_w across a branch and estimate a polynomial for the pixels in the window using a Least Square (LS) approach [27]. The cost for LS estimation is recorded for each valid window position along the branch. The polynomial's leading coefficient (which controls the curvature) is set to be bounded by τ_c above and $-\tau_c$ below. Positions where two 2D discrete functions connect will have a higher cost. Branches with sizes smaller than the window size are discarded. Figure 4b shows an example of a break point between two piecewise functions. Figure 4c shows the function estimation cost vs. window position graph associated with a skeleton. The position with the maximum cost is a candidate for the break point. The maximum spatial quantizaton error for a window is $\frac{\sqrt{2}}{2}$ multiplied by the window size. If the square root of the LS cost of a candidate exceeds τ_1 percent of this error value, we break the branch into two at the window center. Procedure 3 shows the detailed steps. At this point, we obtain sets of connected 2D discrete functions that are ready to be combined into leaf segments.

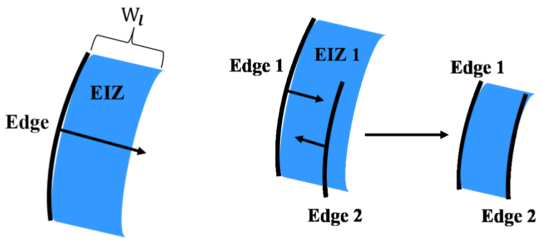
```
Procedure 2: Depth First Search
Input : Skeleton pixel set: $S$,
        End points: $E = \{e_1, e_2 ... e_{N_s}\}$, where $N_s$
        is the number of end points,
        Intersections: $I = \{i_1, i_2 ... i_{N_i}\}$, where $N_i$
        is the number of intersections
1  Initialize visited map $V$
2  Initialize completed branches $B \leftarrow \{\}$
3  Initialize stack $S$
4  $V(e_1) \leftarrow visited$
5  Initialize current branch $b \leftarrow \{e_1\}$
6  Push the neighbor of $e_1$ to $S$
7  while $S$ is not empty do
8      $p \leftarrow pop(S)$
9      $b \leftarrow b \cup \{p\}$
10     $V(p) \leftarrow visited$
11     if $p$ is in $E$ or in $I$ then
12         $B \leftarrow B \cup \{b\}$
13         $b \leftarrow \{\}$
14     end
15     Push unvisited neighbors of $p$ to $S$
16 end
17 Return $B$
```

4. Leaf Segment Detection

Leaf edges have gradient angles pointing towards the leaf surface. Leaves are thin and long, and their edges are semi-parallel. As a result, the two leaf edges have opposite gradient angles. This characteristic allows us to detect segments of a leaf. In UAV images, leaves appear symmetric with respect to their midribs. However, midribs are difficult to

Procedure 3: Divide

```
    Input : Branch: $b = \{a_1, a_2, ...\}$,
            Window size: $W$
            Threshold: $\tau$
1  Initialize stack $S \leftarrow \{b\}$
2  Initialize output list $O \leftarrow \{\}$
3  while $S$ is not empty do
4      $X \leftarrow pop(S)$
5      Initialize cost map $C$
6      for $i = 1$ to $sizeof(b) - W + 1$ do
7          $C(i) \leftarrow cost(b_i, b_{i+1}, ..., b_{i+W-1})$
8      end
9      $m \leftarrow max(C)$
10     $j \leftarrow argmax(C)$
11     if $m > \frac{\sqrt{2}}{2} * W * \frac{\tau}{100}$ then
12         $b_1 \leftarrow \{x_1, ...x_{j+\frac{W}{2}}\}$
13         $b_2 \leftarrow \{x_{j+\frac{W}{2}+1}, ...x_{sizeof(X)}\}$
14         Push $b_1$ and $b_2$ to $S$
15     else
16         $O \leftarrow O \cup \{X\}$
17     end
18 end
19 Return $O$
```



(a) (b)

Figure 6: a) Smearing an edge in the direction of its average gradient angle to create EIZ. b) Detecting candidate edges to form a leaf segment.

detect in low resolution images. We estimate a midrib function for each leaf, where the shortest distances from any point on the midrib to the two edges are equal. The midrib function is essentially a medial axis. An orthogonal line to the midrib function intersects the two leaf edges. The gradient angles at the two intersection points are opposite to each other within a margin of τ_2. The average gradient angle is computed for each edge. We assume if two edges have opposite average gradient angles, they form some parts of a leaf and are considered an edge pair.

To find edge pairs, we smear each leaf edge in the direction of its average gradient angle with a maximum leaf width W_l. We call the area generated by the smearing an Edge Influence Zone (EIZ), as shown in Figure 6a. Any edge inside an EIZ is a candidate for an opposite leaf edge. Candidate edges are further checked for the opposite gradient angles. Successfully paired edges create a leaf segment which is constructed by the overlapping area of two edges' EIZ. Non-overlapped regions and edges are discarded. We call the edges in the segment base edges. This procedure is shown in Figure 6b. Figure 7b shows an example of a detected leaf segment.

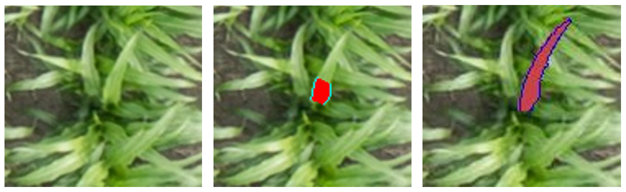

(a) (b) (c)

Figure 7: Example segmentation of a leaf, a) Original image. b) Detected leaf segment. c) Full segmentation of the leaf.

5. Leaf Segment Completion

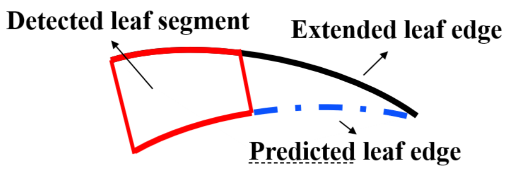
At this stage, a leaf edge can be categorized into three types: a base edge in a leaf segment, an edge discovered by Canny edge detection, and an undetected edge due to its weak response to the edge operator. We use the edges from leaf segments to extend and predict leaf edges. Figure 8a shows the three types of edges.

5.1. Edge Walking

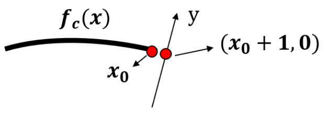
We introduce a method we call Edge Walking as a technique to extend edges by iteratively adding points to the base edge using a statistical model. This technique is inspired by the Kalman filter [28].

Let $f_c(x)$ be an edge function and x_0 be the end of the edge we are considering. We define a slice S_x to be a line that is orthogonal to the function at a position x. A slice profile $g_x(y)$ is generated by sampling the pixel intensities at a set of locations along the slice (as shown in Figure 8b). We take two samples per pixel along the slice. The slice width is set to be 3 which is the maximum width in an 8-connectivity neighborhood. Bi-linear interpolation [29] is used to obtain the sample value at each location. The derivative $g'_x(y)$ of the slice profile $g_x(y)$ is estimated by the difference between the pixel values and smoothed by a 1D Gaussian filter with variance 1.

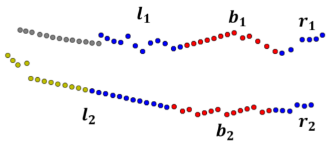
The function $f_c(x)$ is used to predict the location next to x_0 as $(x_0 + 1, f_c(x_0 + 1))$ and we obtain the slice S_{x_0+1}. The slice profile associated is $g_{x_0+1}(y)$, and its derivative is $g'_{x_0+1}(y)$. Our goal is to find the true edge location y in S_{x_0+1}. We model the probability distribution $P_p(y)$ for the



(a)



(b)



(c)

Figure 8: a) Leaf parts that are detected at each stage. b) A slice for a point on the edge. c) Example detected or predicted edge points (red: detected leaf segment edges, blue: paired extended edges, yellow: unpaired extended edges, gray: predicted edges).

prediction y as a zero mean Gaussian with variance 1.

There are two types of observations for the true edge location in slice S_{x_0+1}. The first observation is edges obtained by Canny edge detection. Let $L_{e,i}$ be the observed edge location in the slice S_{x_0+1}, where $i = 1, 2...N_e$ and N_e is the number of observed edges. We model the probability distribution $P_e(y)$ for the Canny edge point y as a Gaussian Mixture Model (GMM) [30]:

$$P_e(y) = \sum_{i=1}^{N_e} w_{e,i}\mathcal{N}(L_{e,i}, \sigma), \tag{3}$$

where weight $w_{e,i} = \frac{1}{N_e}$, and $\mathcal{N}(L_{e,i}, \sigma)$ is a Gaussian with mean $L_{e,i}$ and variance σ.

The second observation is the slice profile derivatives. In the slice profile, an edge can be anywhere along a slope. For a Canny edge, only the peak of absolute derivatives is selected to be an edge. We consider all derivative locations where the absolute value is above a threshold τ_3 as our observations. Let $L_{d,i}$ be the observed derivative location in a slice profile $g_{x_0+1}(y)$, where $i = 1, 2...N_d$ and N_d is the number of observed derivatives. We denote $P_d(y)$ as the probability distribution for a derivative observation y. $P_d(y)$ is also modeled as a GMM:

$$P_d(y) = \sum_{i=1}^{N_d} w_{d,i}\mathcal{N}(L_{d,i}, \sigma)\,, \tag{4}$$

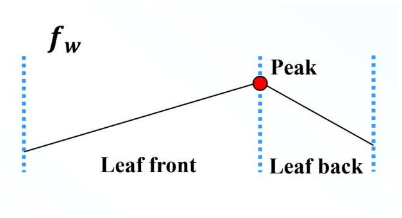


Figure 9: Leaf width function f_w.

where $\mathcal{N}(L_{d,i}, \sigma)$ is a Gaussian with mean $L_{d,i}$ and variance σ, and $w_{d,i}$ is weight, defined as the following:

$$w_{d,i} = \frac{|g'_{x_0+1}(L_{d,i})|}{\sum_{i=1}^{N_d} w_{d,i}|g'_{x_0+1}(L_{d,i})|}\,. \tag{5}$$

Canny edges are obtained from image derivatives, which makes them superior to the derivatives. We use derivative observations only if no Canny edges are available. In the case that no Canny edge or derivative is available, we assume the edge has ended. Now, we define the observation probability $P_o(y)$:

$$P_o(y) = \begin{cases} P_e(y) & \text{if } N_e > 0 \\ P_d(y) & \text{if } N_e = 0 \text{ and } N_d > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

The probability distribution $P(y)$ for edge location y is the joint probability distribution of observation and prediction (assuming independence):

$$P(y) = P_p(y)P_o(y)\,. \tag{7}$$

The Maximum Likelihood Estimate of y is:

$$\hat{y} = \underset{y}{\operatorname{argmax}} P(y)\,. \tag{8}$$

The new edge location $\hat{y}$ is added to the edge we are considering and the edge function is re-evaluated. We keep iterating the process until there are no more Canny edges or derivatives.

5.2. Leaf Modeling

After Edge Walking, we have an extended leaf edge structure that may still contain missing points. Missing edge points are predicted using a leaf shape model. The shape of each leaf is modeled as two functions, a midrib function f_m and a width function f_w. In Section 4, we obtain a matching base edge pair. Let b_1 and b_2 denote the two base edges. In Section 5.1, we obtain a left (l) and right (r) extension for each base edge. We denote l_1, r_1 and l_2, r_2 as the extensions for b_1 and b_2, respectively. These definitions are shown in Figure 8c. Consider the left extensions of the two base edges. The points in l_1 and l_2 are paired based on their order moving away from their respective base edge. The paired extensions are combined with the base edges to form new segment edges b'_1 and b'_2, and $f_{b'_1}$ and $f_{b'_2}$ are their estimated edge functions, respectively. Let b'_1 be the longer edge with N_1 edge points in total. We denote the set of unpaired points on the left as u_l and the set of unpaired points on the right as u_r.

To estimate the midrib function f_m, we project b'_2 onto the PCA space of b'_1. N_1 points are sampled from each edge function. Sampled points from the two functions are paired according to their orientation and order. For each pair, we compute its middle point. Since a leaf may twist, modeling its midrib function with one polynomial is sub-optimal. We model the midrib function as a piece-wise function with three polynomials. The left and right polynomials are estimated by W_w amount of points, where W_w is the sliding window size used in Section 3.2. The middle polynomial is estimated by the rest of the points. Combining the three polynomials, we obtain the midrib function f_m.

To estimate the width function f_w, we take one sample per pixel on the midrib function f_m. At each pixel, we construct an orthogonal line to the midrib function. The line intersects the two edges at i_1 and i_2. The distance between i_1 and i_2 is the width at the pixel. The process is repeated for all pixels to obtain a width profile. We model the width function f_w as a concave piece-wise function consisting of two lines (as shown in Figure 9). The function has only one peak. Let p be the location of the peak. As shown in Figure 9, let the leaf front represent widths from p to the leaf tip, and let the leaf back represent widths from p to the leaf tail. We estimate one function for the leaf front and one for the leaf back. Combining the two functions, we obtain the width function f_w.

Finally, we predict matching edge points for the unpaired point sets u_l and u_r. For each point z in an unpaired point set, we predict the local leaf width w_z using f_w and obtain its orthogonal line to the edge e_j, where j is the edge index. If the width is zero or less, the point is discarded. A matching edge point is estimated to be w_z distance away from the location of z along its orthogonal line. Figure 8c shows an example of detected edge points, extended edge points, and predicted edge points. Figure 7c shows an example of leaf segmentation after edge extension and prediction.

A missing leaf tip can be predicted using the midrib function f_m and width function f_w. However, the appearance of a leaf tip in a UAV image varies across plant breeds. For example, a plant breed with thin long leaves often has visible tips, while a plant breed with wide leaves may have its tips pointing away from the image sensor plane. This means leaf tips from wider leaves may not be visible. In this study, leaf tips are not predicted if no edge is detected at the tip, because we do not consider prior knowledge of plant breeds.

Table 1: Results reported in Symmetric Best Dice (SBD), Foreground-Background Dice (FBD) and absolute Difference in Count (|DiC|)

Metric	**Mask R-CNN**	**Proposed Method**
SBD (higher is better)	34%	**37%**
FBD (higher is better)	**73%**	67%
|DiC| (lower is better)	120	**81**

5.3. Post-processing

We may detect multiple edge pairs from the same leaf. Our proposed method generates a leaf segmentation mask for the entire leaf using only one of its detected edge pairs. Thus, there can be redundant leaf segmentation as similar masks can be developed from different edge pairs of the same leaf. As a solution, we reject an individual leaf mask if it has more than 50% overlap with the leaf foreground mask generated by collating all previous leaf masks. A leaf segmentation mask may also be constructed erroneously using parallel edges from different leaves. We remove contours from a segmentation mask using morphological erosion [31] and a surface mask is obtained. If the surface mask contains K pixels, the leaf segmentation is discarded. K is designed to be half the length of the shortest edge in the leaf segmentation.

6. Experimental Results

We typically estimate phenotypic traits using images acquired by a UAV flying at an altitude of 50 meters. The result of our method on an image from 50 meters altitude is shown in Figure 1. However, to examine the performance of our proposed method, we need manual ground truth. UAV images acquired at 50 meters are difficult to ground truth for leaf segmentation due to the low spatial resolution. In our experiments for this paper, we use images taken from 20 meters altitude to generate ground truth manually. Figure 10b shows an example of the ground truth. The images are then downsampled to the same resolution as images from 50 meters altitude. A 2D Gaussian filter with 0.5 variance is used to emulate the expected blur in images taken from a higher altitude. We use this downsampled data to quantitatively evaluate our proposed method. Our data was obtained from a sorghum field on June 20, 2018 and June 27, 2018. The dataset consists of 88 sorghum images with each image containing about 200 leaves. After downsampling, the image dimensions become roughly 120x580 pixels with a spatial resolution of 0.63 cm per pixel.

The thresholds for the Canny edge detection are set dif-

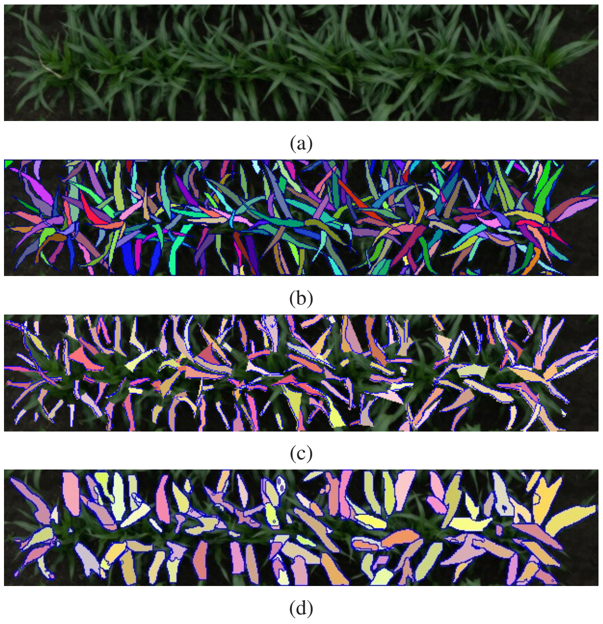


Figure 10: a) A downsampled sorghum image taken from 20 meters altitude on June 27th, 2018. b) Manual ground truth, where leaves are colored randomly. c) Result image generated by our method, where leaves are colored randomly. d) Result image generated by Mask R-CNN, where leaves are colored randomly.

ferently for each date due to the change in illumination. The images from June 27, 2018 appear much darker than the images from June 20, 2018. For June 20, 2018 data, the strong edge threshold is set to 1500 and the weak edge threshold is set to 500. For June 27, 2018 data, the strong edge threshold is set to 750 and the weak edge threshold is set to 500. The maximum percentage τ_1 allowed for the function estimation error is set to be 25%. The gradient angle bias τ_2 is set to be $\frac{\pi}{4}$. The minimum threshold τ_3 for derivative observation is set to be 5 to prevent noise. The curvature coefficient bound τ_c is set to be 0.05 empirically. The sliding window size W_w is set to be 10 empirically. The maximum leaf width W_l is measured to be 15 pixels. The variance σ for all assumed Gaussian distributions is set to be 1.

We compared our method to Mask-RCNN. The training and testing procedure are similar to the work in [8]. Our dataset is split into three, 72 for training, 8 for validation, and 8 for testing. We used Resnet-50 Feature Pyramid Network as the backbone instead of Resnet-101 because of our small dataset. The Region Proposal Network aspect ratios were set to {1:5, 1:2, 1:1, 2:1, 5:1} to better fit the elongated shapes of the sorghum leaves. Leaf bounding boxes may significantly intersect because of the densely overlapping leaves. During testing, we raised the Non-Maximum Suppression (NMS) threshold to 0.8 to account for this. A higher NMS threshold keeps highly overlapped bounding boxes.

Mask R-CNN and our proposed method are evaluated on the 8 test images. We report the results using Symmetric Best Dice (SBD), Foreground-Background Dice (FBD) and absolute Difference in Count (|DiC|) [32]. These metrics are used in the CVPPP segmentation dataset [33].

Both methods under-counted the leaves. Table 1 gives a detailed comparison of the method. Figure 10 shows example results from both methods.

7. Discussion and Conclusion

Compared to Mask R-CNN, our method delivers better results with respect to SBD, which is a primary metric for leaf segmentation. We also out-perform Mask R-CNN for DiC. By visual comparison of the results, our method provides better leaf contours. The leaf edges in our results are well defined, whereas Mask R-CNN easily includes adjacent leaves and ground structures in its segmentation. Therefore, the larger segmentation contours unintentionally generate a better foreground mask. This explains the better results for FBD for Mask R-CNN.

This paper presented a new approach to segment leaf instances in UAV images by modeling leaf shapes. The method detects leaf segments by using semi-parallel features of leaf edges. It generates shape models for the segments and uses those shapes to extend and predict leaf edges. The method is compared to Mask R-CNN with our approach achieving better results. The midrib function f_m and the width function f_w can be used to predict phenotypic traits such as leaf area, leaf length, and maximum leaf width. In addition, the semi parallel edge features are very common in plant structures, such as leaves, stems, and branches. This method has the potential to segment these plant materials. Future work includes adapting the method to account for different leaf shapes that do not have semi-parallel features, such as the leaves in the CVPPP leaf segmentation dataset [33]. We will also combine surface features into the leaf model.

Acknowledgment

We thank Professor Ayman Habib and the Digital Photogrammetry Research Group (DPRG) from the School of Civil Engineering at Purdue University for providing the images used in this paper.

The information, data, or work presented herein was funded in part by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Award Number DE-AR0000593. The views and opinions of the authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof. Address all correspondence to Edward J. Delp, ace@ecn.purdue.edu

References

[1] D. J. Watson, "Comparative physiological studies on the growth of field crops: I. variation in net assimilation rate and leaf area between species and varieties, and within and between years," *Annals of Botany*, vol. 11, pp. 41–76, January 1947. 1

[2] D. Kelly, A. Vatsa, W. Mayham, L. Ngô, A. Thompson, and T. KazicDerek, "An opinion on imaging challenges in phenotyping field crops," *Machine Vision and Applications*, pp. 1–14, December 2015. 1

[3] R. T. Furbank and M. Tester, "Phenomics-technologies to relieve the phenotyping bottleneck," *Trends in Plant Science*, vol. 16, no. 12, pp. 635–644, November 2011. 1

[4] S. C. Chapman, T. Merz, A. Chan, P. Jackway, S. Hrabar, M. F. Dreccer, E. Holland, B. Zheng, T. J. Ling, and J. Jimenez-Berni, "Pheno-copter: A low-altitude, autonomous remote-sensing robotic helicopter for high-throughput field-based phenotyping," *Agronomy*, vol. 4, no. 2, pp. 279–301, June 2014. 1

[5] R. Makanza, M. Zaman-Allah, J. Cairns, C. Magorokosho, A. Tarekegne, M. Olsen, and B. Prasanna, "High-throughput phenotyping of canopy cover and senescence in maize field trials using aerial digital canopy imaging," *Remote Sensing*, vol. 10, no. 2, pp. 330, February 2018. 1

[6] A. Singh, B. Ganapathysubramanian, A.K. Singh, and S. Sarkar, "Machine learning for high-throughput stress phenotyping in plants," *Trends in Plant Science*, vol. 21, no. 2, pp. 110–124, February 2016. 1

[7] A. Habib, W. Xiong, F. He, H. L. Yang, and M. Crawford, "Improving orthorectification of UAV-Based push-broom scanner imagery using derived orthophotos from frame cameras," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, pp. 262–276, January 2017. 1

[8] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2980–2988, October 2017, Venice, Italy. 2, 8

[9] L. Chen, A. Hermans, G. Papandreou, F. Schroff, P. Wang, and H. Adam, "Masklab: Instance segmentation by refining object detection with semantic and direction features," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4013–4022, June 2018, Salt Lake City, UT. 2

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 6, pp. 1137–1149, June 2016. 2

[11] Y. Chen, J. Ribera, C. Boomsma, and E. J. Delp, "Plant leaf segmentation for estimating phenotypic traits," *Proceedings of the IEEE International Conference on Image Processing*, September 2017, Beijing, China. 2

[12] D. Ward, P. Moghadam, and N. Hudson, "Deep leaf segmentation using synthetic data," *arXiv preprint arXiv:1807.10931*, August 2018. 2

[13] D. D. Morris, "A pyramid cnn for dense-leaves segmentation," *Proceedings of the Conference on Computer and Robot Vision*, pp. 238–245, May 2018, Toronto, Canada. 2

[14] J. Hopcroft and R. Tarjan, "Algorithm 447: Efficient algorithms for graph manipulation," *Communications of the ACM*, pp. 372–378, June 1973. 2

[15] S. Beucher, "Watersheds of functions and picture segmentation," *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing*, pp. 1928–1931, May 1982, Paris, France. 2

[16] C. Fu, S. Lee, D. J. Ho, S. Han, P. Salama, K. W. Dunn, and E. J. Delp, "Three dimensional fluorescence microscopy image synthesis and segmentation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop*, June 2018, Salt Lake City, UT. 2

[17] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *arXiv preprint arXiv:1505.04597*, May 2015. 2

[18] F. Milletari, N. Navab, and S. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," *arXiv preprint arXiv:1606.04797*, June 2016. 2

[19] S. K. Panguluri and A. A. Kumar, "Phenotyping in sorghum," *Phenotyping for Plant Breeding: Applications of Phenotyping Methods for Crop Improvement*, vol. 1, pp. 73–110. Springer New York, New York, NY, 2013. 2

[20] National Research Council, "Sorghum," *Lost Crops of Africa. Volume I: Grains*, vol. 1, pp. 127–144. The National Academies Press, Washington, DC, 1996. 2

[21] Alan Watt, *3D Computer Graphics*, 3rd ed., Addison Wesley, Boston, MA, 1999. 3

[22] I.T. Jolliffe, *Principal Component Analysis*, 2nd ed., Springer, New York, NY, 2002. 3

[23] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, November 1986. 3

[24] E. R. Davies, *Computer and Machine Vision*, 4th ed., Elsevier, Amsterdam, Netherlands, 2012. 3

[25] T. H. Cormen, C. E. Leiserson, R. L. Riveest, and C. Stein, *Introduction to Algorithms, 3rd Edition*, The MIT Press, Cambridge, Massachusetts, 2009. 3

[26] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, pp. 100–107, July 1968. 4

[27] S. M. Stigler, "Gauss and the invention of least squares," *The Annals of Statistics*, pp. 465–474, 1981. 4

[28] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960. 5

[29] P. Thévenaz, T. Blu, and M. Unser, "Interpolation revisited," *IEEE Transactions on medical imaging*, pp. 739–758, July 2000. 5

[30] G.J. McLachlan, S.X. Lee, and S.I. Rathnayake, "Finite mixture models," *Annual Review of Statistics and Its Application*, vol. 6, no. 1, pp. 355–378, 2019. 6

[31] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, Inc., Orlando, FL, 1983. 7

[32] H. Scharr, M. Minervini, A.P. French, C. Klukas, D.M. Kramer, X. Liu, I. Luengo, J. Pape, G. Polder, D. Vukadinovic, X. Yin, and S.A. Tsaftaris, "Leaf segmentation in plant phenotyping: a collation study," *Machine Vision and Applications*, vol. 27, no. 4, pp. 585–606, May 2016. 8

[33] M. Minervini, A. Fischbach, H.Scharr, and S.A. Tsaftaris, "Finely-grained annotated datasets for image-based plant phenotypinp," *Pattern Recognition Letters*, vol. 81, no. 1, pp. 80–89, October 2016. 8