

ESTIMATING PLANT PHENOTYPIC TRAITS FROM RGB IMAGERY

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Yuhao Chen

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Edward J. Delp, Chair

School of Electrical and Computer Engineering

Dr. Melba M. Crawford

Department of Agronomy

Dr. Amy R. Reibman

School of Electrical and Computer Engineering

Dr. Mary Comer

School of Electrical and Computer Engineering

Approved by:

Dr. Dimitrios Peroulis

Head of the School of Electrical and Computer Engineering

ACKNOWLEDGMENTS

First and most importantly, I would like to thank Professor Edward J. Delp for his guidance and support for my PhD and academic career. He pushes me to learn new things and to understand every detail of them. His high standard for documentation, presentations, and publications has crushed me, pushed me, guided me, and inspired me. At the same time, he offers me opportunities to attend conferences, meet with other experts, and share my ideas. He teaches me how to research, how to communicate, and how to manage projects. Without him, I would not become the person I am.

I would like to thank Professor Melba M. Crawford for her guidance, suggestions, criticisms, and compliments. Her countless long emails and critical questions have prepared me for better writing and communication skills and deeper project understandings.

I would like to thank my other committee members, Professor Amy R. Reibman and Professor Mary Comer for their valuable insights, suggestions, discussions, and supports.

I would like to thank Professor Melba M. Crawford again, and her student Zhou Zhang, Ali Masjedi, Karoll Jessenia Quijano Escalante, Ruya Xu, Taojun Wang, Azam Karami, and their team from the School of Civil Engineering for coordinating data collections, giving suggestions on data calibrations, providing great discussions on machine learning topics, and helping with ground truth labeling.

I would like to thank Professor Ayman Habib, and his student Fangning He, Tian Zhou and Seyyed Meghdad Hasheminasab, and the Digital Photogrammetry Research Group (DPRG) from the School of Civil Engineering for providing the orthophotos, images, various datasets, and tools used in this thesis.

I would like to thank Professor Mitch Tuinstra, his student Kai-wei Yang, and their team from the College of Agriculture for providing ground reference data and educating us on plant phenotyping and plant breeding in fascinating ways.

I would like to thank Professor David Ebert and his student Jieqiong Zhao, and the Visual Analytics for Command, Control, and Interoperability Environments (VACCINE) Center for providing visualization tools and helpful discussions.

I would like to thank my colleague Dr. Javi Ribera for being the best teammate for four years. I would like to thank David Gera Cobo, Sriram Baireddy, Enyu Cai, and Changye Yang for being great teammates and providing wonderful and inspirational discussions. I would also like to thank the rest of the VIPER lab colleagues Dr. Neeraj Gadgil, Dr. Khalid Tahboub, Di Chen, Qingchaung (Cici) Chen, Dr. Dahjung Chung, Dr. Jeehyun Choe, Dr. Shaobo Fang, Dr. Chichen Fu, Shuo Han, Hanxiang (Hans) Hao, Dr. David Ho, Jnos Horvth, Dr. Joonsoo Kim, Dr. Soonam Lee, He Li, Chang Liu, Daniel Mas, Ruiting Shao, Zeman Shao, Dr. Yu Wang, Sri Kalyan Yarlagadda, Jiaju Yue, and Yifan Zhao.

I would like to thank my parents and my brother for their love, understanding, and support.

Last but not least, I would like to thank my wife, Yuhan Huang, for her unconditional love and support. I thank her for bringing me happiness, giving me inspirations, being my light, and sharing her life with me.

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.

TABLE OF CONTENTS

	Page
LIST OF TABLES	viii
LIST OF FIGURES	ix
NOMENCLATURE	xiii
ABSTRACT	xiv
1 INTRODUCTION	1
1.1 Plant Phenotyping	1
1.2 Sorghum	2
1.3 Plant Phenotyping With Outdoor RGB Imagery	4
1.4 Field Description and Terminologies	10
1.5 List of Phenotypic Traits	14
1.6 Plant Phenotyping with Machine Learning	15
1.7 Contributions of This Thesis	15
1.8 Publications Resulting From This Work	16
2 PLANT PHENOTYPING SYSTEM	18
2.1 Image Phenotyping System	18
2.2 Tools	25
2.2.1 Orthophoto Rotation Estimation	25
2.2.2 Inverse Mapping	28
2.3 Distributed Image-Based Phenotyping System	30
3 PLANT MATERIAL SEGMENTATION	37
3.1 HSV Color Segmentation	37
3.1.1 Motivation	37
3.1.2 Proposed Method	37
3.1.3 Experimental Results	38

	Page
4 PLANT LOCATION AND COUNTING	41
4.1 Introduction	41
4.2 Overview of Deep Learning Methods	41
4.3 Plant Location Related Works	51
4.4 Finding Plant Location Using Multiple Instance Learning	52
4.4.1 Motivation	52
4.4.2 Multiple Instance Learning	52
4.4.3 Features	56
4.4.4 Weak Classifier	57
4.4.5 Post-Processing	60
4.4.6 Experiment and Results	63
4.5 Finding Plant Location Using Deep Binary Classifier	67
4.5.1 Motivation	67
4.5.2 Proposed Method	67
4.5.3 Experimental Results	70
5 LEAF SEGMENTATION	75
5.1 Leaf Segmentation Introduction	75
5.2 Related Works	75
5.3 Slice Based Leaf Segmentation	84
5.3.1 Motivation	84
5.3.2 Proposed Method	84
5.3.3 Experimental Results	86
5.4 Leaf Segmentation In Polar Coordinates	88
5.4.1 Motivation	88
5.4.2 Leaf Segmentation	88
5.4.3 Leaf modeling	93
5.4.4 Experimental Results	95
5.5 Leaf Segmentation by Functional Modeling	99

	Page
5.5.1 Motivation	99
5.5.2 Processing Structure and Assumptions	100
5.5.3 Leaf Edge Processing	102
5.5.4 Leaf Segment Detection	107
5.5.5 Leaf Segment Completion	111
5.5.6 Experimental Results	119
5.5.7 Discussions	122
5.6 Midrib Detection for Leaf Length and Width Estimation	124
5.6.1 Preprocessing	126
5.6.2 Midrib Curve Estimation	129
5.6.3 Leaf Length and Width Estimation	129
5.7 Midrib Detection for Leaf Angle Estimation	132
6 DEEP LEARNING EXPERIMENTS	136
6.1 Introduction	136
6.2 Training by Comparing	136
6.2.1 Experimental Results	138
6.2.2 Conclusion, and Discussion	143
6.3 Logical AND Operations with Deep Learning	144
6.3.1 Conclusion and Discussion	145
7 SUMMARY AND FUTURE WORKS	147
7.1 Summary	147
7.2 Future Works	149
7.3 Publications Resulting From This Work	152
VITA	153
REFERENCES	154

LIST OF TABLES

Table	Page
4.1 Orientations of the Gabor wavelets	58
4.2 Plant location detection results	65
4.3 Results compared to Multiple Instance Learning	74
5.1 Segmentation results of the proposed method and Mask R-CNN	119
5.2 Segmentation results of the proposed method and the slice-based method	121

LIST OF FIGURES

Figure	Page
1.1 Examples of traditional phenotyping	2
1.2 Sorghum anatomy	3
1.3 Example of high-throughput indoor phenotyping platform	5
1.4 Example of a ground data collection platform	5
1.5 Example of a UAV data collection platform	7
1.6 Image of a subrow of a sorghum field	9
1.7 Image of a section of a sorghum field	9
1.8 Field Definition and Examples	10
1.9 Panel Definition	11
1.10 Plot Definition	12
1.11 Row Segment Definition	13
2.1 Phenotyping system simple blockdiagram	19
2.2 Phenotyping system detail flowchart	20
2.3 Plot Extraction	21
2.4 File structure for storage	23
2.5 File structure example	24
2.6 A Vertical Projection Profile Example	26
2.7 An Example of Rotation Estimation	27
2.8 Image Selection for Inverse-mapping	29
2.9 An example of inverse-mapping	29
2.10 Login Page	30
2.11 Home Page	31
2.12 Data Upload Page	31
2.13 Image Phenotyping Tools Page	32

Figure	Page
2.14 Available Image Phenotyping Tools	33
2.15 Region of Interest Select	34
2.16 Results Page	35
2.17 Documentation Page	36
3.1 Segmentation results	39
3.2 Detail segmentation results	40
4.1 AlexNet architecture	42
4.2 CNN layer visualization	43
4.3 GoogleNet Inception Module	44
4.4 GoogleNet architecture	45
4.5 ResNet architecture example	45
4.6 ResNet Residual Learning Block	46
4.7 DenseNet Dense Block	46
4.8 DenseNet architecture	46
4.9 Faster R-CNN architecture	47
4.10 Mask R-CNN architecture	48
4.11 YOLO object detection	49
4.12 SSD and YOLO architecture comparison	50
4.13 MIL explanation	56
4.14 Features by sections	59
4.15 Post-processing	61
4.16 Dataset	62
4.17 Plant location results	64
4.18 Block diagram	68
4.19 Example of a classification output mask	68
4.20 Post-processing explained	69
4.21 Block diagram of post-processing	70
4.22 Results on young plants	71

Figure	Page
4.23 Results on mature plants	72
4.24 Precision and recall	73
4.25 Results compared to Multiple Instance Learning	74
5.1 Semantic Segmentation	76
5.2 Instance Segmentation	78
5.3 Panoptic Segmentation	79
5.4 Graph Neural Network Application	81
5.5 Data Augmentation Example	82
5.6 Ground truth tool	83
5.7 A ground truth example	83
5.8 Leaf slice definition	84
5.9 Connecting leaf segments	85
5.10 Leaf segmentation results	87
5.11 Polar representation	89
5.12 Leaf slice in hierarchical representation	91
5.13 Segmentation of individual leaves	91
5.14 Corner case: holes in the segmentation mask	92
5.15 Modeled leaf shape	95
5.16 Segmentation of a leaf	96
5.17 Leaf segmentation results	98
5.18 A Leaf Segmentation Example	99
5.19 Overall Block Diagrams and Assumptions	101
5.20 Edge Processing Block Diagram	102
5.21 Leaf Edge Preprocessing	103
5.22 Leaf Edge Skeletonization	105
5.23 Leaf Segment Detection	110
5.24 A Segmentation Example of A Leaf	111
5.25 Leaf Segment Completion	112

Figure	Page
5.26 Leaf Width Function	115
5.27 Segmentation Results	118
5.28 An example image taken at ground reference data collection shed	124
5.29 Blockdiagram for estimation leaf length and width	125
5.30 Blockdiagram for segmenting the white table in image	127
5.31 Blockdiagram for segmenting each leaf on the table	128
5.32 Blockdiagram for estimating the midrib of a leaf	130
5.33 Examples of leaf width and length estimation	131
5.34 Examples of Midrib Detection	133
5.35 Leaf Angle Estimation	134
5.36 Examples of Leaf Angle Estimation	135
6.1 Comparison Network Architecture	137
6.2 Results for 2 Classes	140
6.3 Results for 7 Classes	141
6.4 Results for 10 Classes	142
6.5 Results for 2 Classes with 2 Dimensional Relationship	142
6.6 Comparison Network for Plant Location Estimation	143

NOMENCLATURE

DL	Deep Learning
ML	Machine Learning
HSV	Hue, Saturation, and Value
RGB	Red, Green, and Blue
ROI	Region Of Interest
UAV	Unmanned Aerial Vehicle
LAI	Lear Area Index
MIL	Multiple Instance Learning
CNN	Convolutional Neural Network
NDVI	Normalized Difference Vegetation Index
CRF	Conditional Random Field
DNN	Deep Neural Network
GNN	Graph Neural Network
GCN	Graph Convolutional Network
FCN	Fully Convolutional Network
GAN	Generative Adversarial Nets
FPN	Feature Pyramid Network
VAE	Variational Autoencoder
IMU	Inertial Measurement Unit

ABSTRACT

Chen, Yuhao Ph.D., Purdue University, December 2019. Estimating Plant Phenotypic Traits From RGB Imagery. Major Professor: Edward J. Delp.

Plant Phenotyping is a set of methodologies for measuring and analyzing characteristic traits of a plant. While traditional plant phenotyping techniques are labor-intensive and destructive, modern imaging technologies have provided faster, non-invasive, and more cost-effective capabilities for plant phenotyping. Among different image-based phenotyping platforms, I focus on phenotyping with image data captured by Unmanned Aerial Vehicle (UAV) and ground vehicles. The crop plant used in my study is sorghum [*Sorghum bicolor* (L.) Moench]. In this thesis, I present multiple methods to estimate plot-level and plant-level plant traits from data collected by various platforms, including UAV and ground vehicles. I propose an image plant phenotyping system that provides end-to-end RGB data analysis for plant scientists. I describe a plant segmentation method using HSV color information. I introduce two methods to locate the center of the plants using Multiple Instance Learning (MIL) and Convolutional Neural Networks (CNN). I present three methods to segment individual leaves by shape-based approaches in both Cartesian coordinates and Polar coordinates. I propose a method to estimate leaf length and width for overhead leaf images. I describe a method to estimate leaf angle from data collected by a modified wheel-based sprayer with a sensor boom vehicle, Phenorover. Methods are tested and verified on image data collected by UAV and ground vehicle platforms in sorghum fields in West Lafayette, Indiana, USA. Estimated phenotypic traits include plant locations, the number of plants per plot, leaf area, canopy cover, Leaf Area Index (LAI), leaf count, leaf angle, leaf length, and leaf width.

1. INTRODUCTION

1.1 Plant Phenotyping

The process used by crop scientists to measure the physical and chemical properties of a plant is known as phenotyping [1–4]. It is a set of methodologies for recording, measuring, and analyzing characteristic traits of a plant resulting from genetic and environmental factors.

Phenotyping provides a key solution to the challenge of agricultural production posed by the increasing human population. According to [5], crop production needs to double to meet the demand by 2050. To meet demand for future food and fuel needs, agricultural crops need to be improved to produce higher crop yield and to tolerate abiotic stresses [3]. The rapidly developing genome sequencing technologies have provided genetic information across the various types of agricultural crops, enabling plant scientists to link genes to phenotypes of interests [6–11]. The linkages of genotype to phenotype help plant breeders and researchers to select high yielding and stress-tolerant plants efficiently [12]. Low cost, high-throughput phenotyping becomes the key in finding these linkages with precision, accuracy, and rapidness [4, 13].

Traditional field phenotyping requires a great deal of human labor to routinely obtain traits measurement on a plant by plant basis, making data acquisition labor-intensive and time-consuming. For example, workers first need to identify an individual plant from neighbors as plants in the field are typically planted densely to maximize yield. Then, several traits are recorded with various devices. Many data collection methods are destructive where plants are damaged or removed from the field. Figure 1.1 shows some examples of traditional phenotyping. Plants are cut and laid out on the tables, where researchers take measurements, including plant height, leaf counts, leaf angle, leaf area, and the number of tillers. Since measurements are

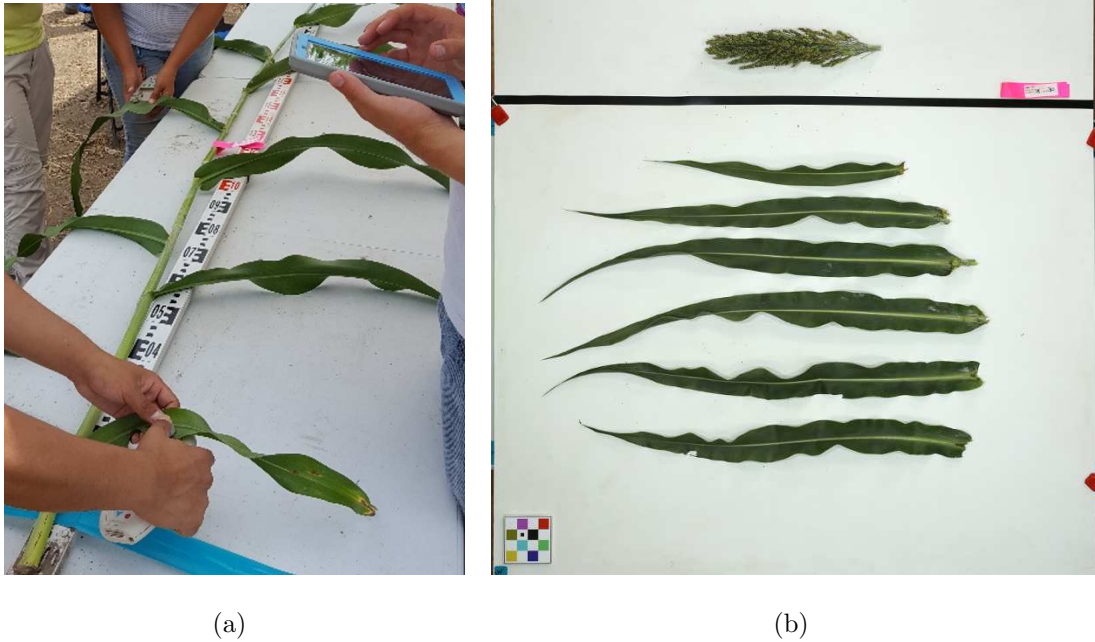


Fig. 1.1. Traditional phenotyping: a) researchers are taking measurements of a sorghum plant. b) Leaves and panicle are cut and laid out on the table for measuring area.

made plant by plant, field size directly relates to the time need for collecting data. Thus, labor required for collecting and analyzing data becomes the bottleneck of high-throughput plant phenotyping.

1.2 Sorghum

I focus my analysis on the crop sorghum [*Sorghum bicolor* (L.) Moench] [1, 14]. Sorghum is the fifth most important cereal crop in the world [1]. It can be grown in both temperate and tropical zones and is one of the most photosynthetically efficient plants [14]. Sorghum is mostly used as food and can also be used as feed, fodder, fiber [1], and energy [15–17]. Figure 1.2 shows the anatomy of sorghum plants. In one growing circle, sorghum has nine growth stages [19]. In the first 4 stages, sorghum develops its leaves and tillers. A tiller is an extra branch that grows out of the stem.

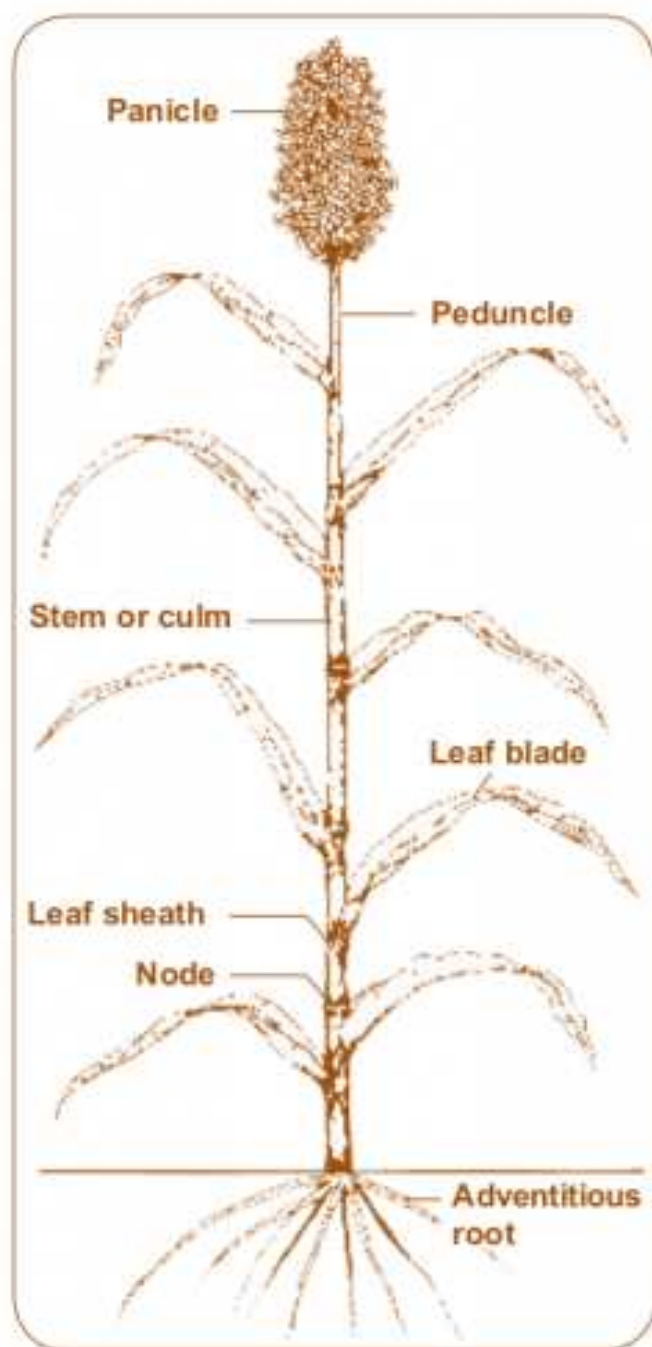


Fig. 1.2. Sorghum anatomy [18]

At stage 4, sorghum develops all its leaves. At stage 5, sorghum fully expands its leaves and starts developing the seed panicle. In stages 6-9, sorghum begins to flower and fill its grain.

1.3 Plant Phenotyping With Outdoor RGB Imagery

While traditional phenotyping techniques are labor intensive and destructive, modern imaging technologies have provided faster, non-invasive and more cost-effective plant phenotyping [3, 12, 20–22].

Previous work in image-based phenotyping can be categorized into two approaches. In one approach, individual plants are phenotyped in environments where the imaging conditions and the distribution of plants are controlled. Plants are under consistent treatments, so experiments conducted in a controlled environment can be easily reproduced. Figure 1.3 shows an example of indoor high-throughput phenotyping platform layout. Plants in the platform can be freely moved around to be watered, weighed, and taken image measurements. Controlled environment studies have successfully provided high precision traits estimation and predicted plant growth. In [24, 25], leaves are destructively collected to obtain the leaf area measurement by scaling the number of pixels in the segmentation mask. In [26], plants in an automated indoor facility are detected using color thresholding, then located using k-means, and finally segmented using active contours. In [27], 3D individual plants are constructed from depth images acquired from multiple viewpoints. In [28], Gibbs *et al.* reconstruct a 3D plant shoot surface from 2D RGB images. In [29], a light-field camera is used under near-infrared lighting to monitor spatiotemporal plant growth of plants with rosette leaf forms. Phenotyping in a controlled environment shows promising capabilities, but cannot be employed in large scale field studies [12]. In addition, this approach is limited in reflecting real growing conditions in the field, such as extreme weather, intense plant-plant competition, high solar radiation, fast evaporation rate, soil conditions for proper root development, and drainage conditions.

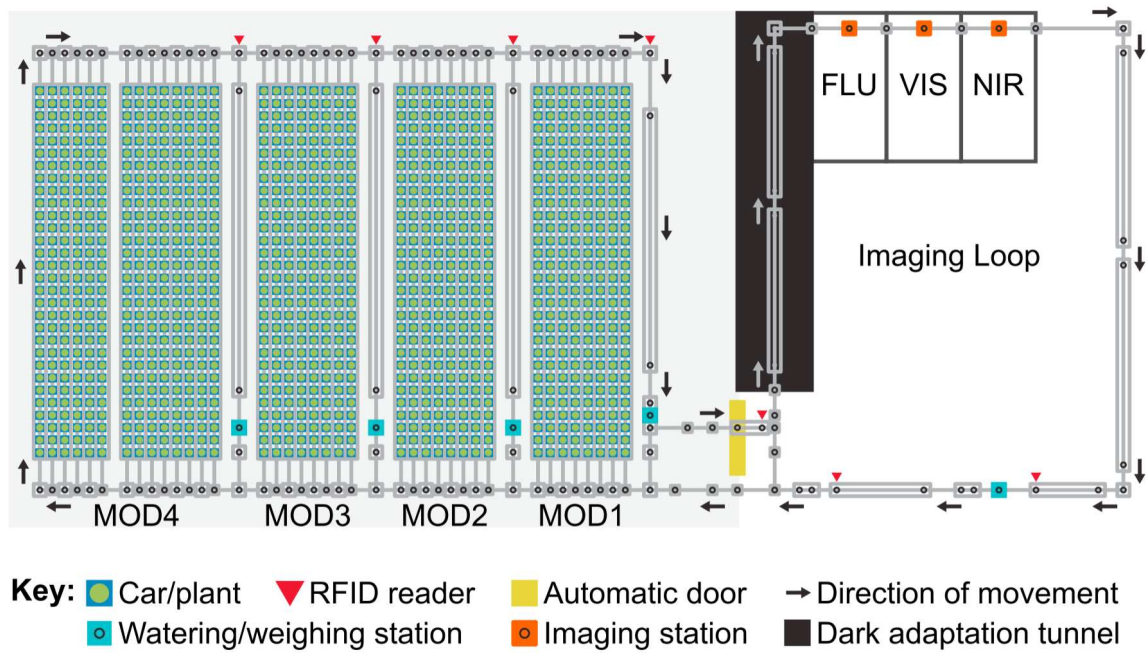


Fig. 1.3. Example of high-throughput indoor phenotyping platform [23]



Fig. 1.4. Example of a ground data collection platform

The other approach to plant phenotyping estimates plant traits from image data collected in an outdoor environment (field-based). Figure 1.4 shows an example of a ground data collection vehicle. In [12], a vehicle carrying multiple sets of sensors is used as a field-based phenotyping platform. With the platform, the study aims to use commercial software to estimate plant traits, including plant height, Leaf Area Index, extinction coefficient, transpiration, leaf number, and area of individual leaves. Similarly, in [30] and [31], sensor mounted small tractors are used to collect image data for traits estimation. In [32], an orthomosaic of the field is created from UAV images, and pixels are classified into crop, soil, and shadow using maximum likelihood estimates. The mask of the crop is used to estimate the Normalized Difference Vegetation Index (NDVI). In [33], field images from a hand-held device are acquired and segmented using histogram-based thresholding. The segmentation masks are then used to estimate LAI, where LAI is defined as the total leaf surface per unit area of land [34]. In [35], Hu *et al.* estimate plant height from RGB images taken from UAV. In [36], the canopy temperature of sugarcane is predicted pixel-wise. In [37], plants are segmented using a decision tree based method, and the canopy cover is computed. Field-based approaches can significantly increase the efficiency of large scale phenotyping studies with relatively lower cost [12, 13, 32, 36, 38, 39].

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 [36]. Figure 1.5 shows an example UAV carrying multiple sensors. Large scale data collection using UAVs is often finished within a short period time compared to traditional data collection. Figure 1.6 and 1.7 show some data collected by the UAV. In [40], Hu *et al.* study how the image capture altitude and the spatial resolution of image pixels relate to the canopy cover computation error. They suggest the UAV flying altitude to be 20m-30m to minimize this error.

Most current studies using UAV imagery [32, 36, 38, 39, 41–43] focus on average traits for small regions of the crop field or plots, such as height, NDVI, LAI, canopy



(a)



(b)

Fig. 1.5. Example of a UAV data collection platform carrying multiple sensors

coverage without estimating traits of each plant. Estimation of plot averages often balance factors that may affect plant growth, such as the uneven distribution of natural resources and plant competition. For example, plants near borders have more space and sunlight coverage, which results in larger plants. In our plant experiments, border plants are excluded. Crop fields for research mostly contain multiple plant varieties in neighboring plots [44]. Within each plot, competition for natural resources can also increase plant variability. When the plant variability within each plot is high, average values may not be sufficient in estimating and analyzing phenotypic traits. One solution is to only consider plants in the middle of a row segment, but accurately setting the boundary within a row segment is difficult due to the unknown location of the border plants in the row segment. Therefore, it is useful to introduce methods that detect the center of plants, which can increase the precision of phenotypic trait estimation with UAV imagery and account for individual plant traits.

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 mosaic them [45]. 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 a 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 and reduces problems with matching for orthorectification. This strategy limits the possible changes in illumination, weather conditions, and perspective distortions within the dataset, and thus, it improves the data consistency.



Fig. 1.6. Image of a subrow of a sorghum field acquired from a UAV at an altitude of 40m.

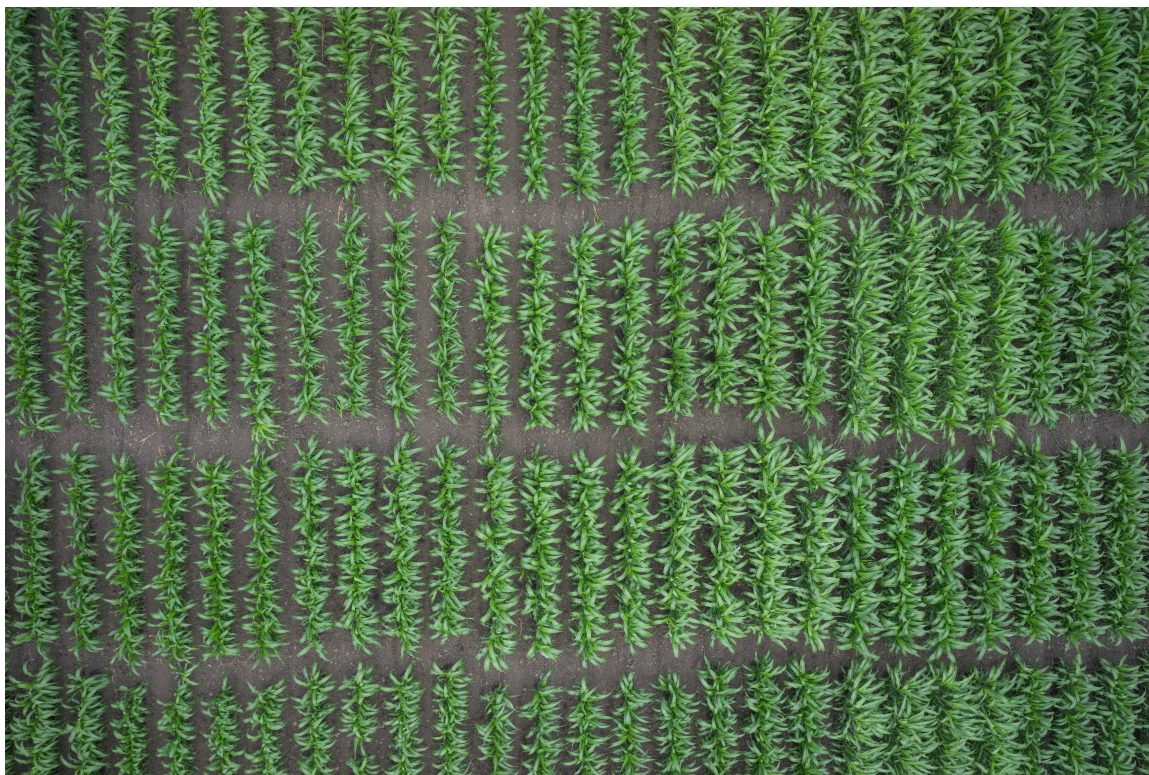


Fig. 1.7. Image of a section of a sorghum field (field 54 calibration panel) in West Lafayette, Indiana acquired from a UAV on 06/20/2018 at an altitude of 20m.

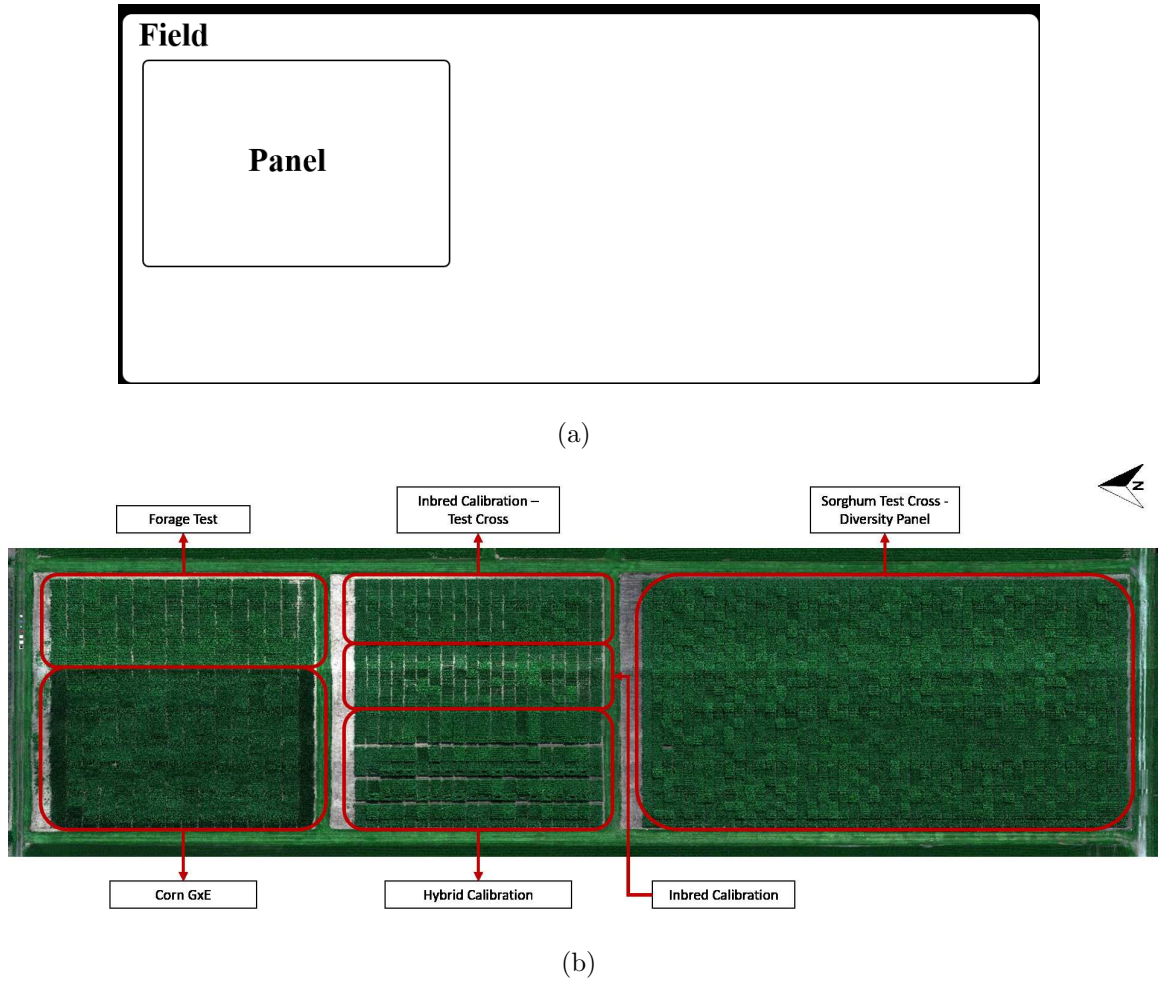


Fig. 1.8. a) Field definition. b) An example of a field containing multiple experiment panels. The image is taken in 2018 at Field 54 in Purdue ACRE.

1.4 Field Description and Terminologies

A field contains multiple experiment panels (as shown in Figure 1.8). A calibration panel contains plants that are used as a reference for observing the changes in plants in other experiments. Inbred calibration contains plants that are bred from pure gene lines, while hybrid calibration contains plants that are bred from two pure genetic lines. A diversity panel contains hundreds of inbred plants. Hybrid plants are bred from two inbred plants.

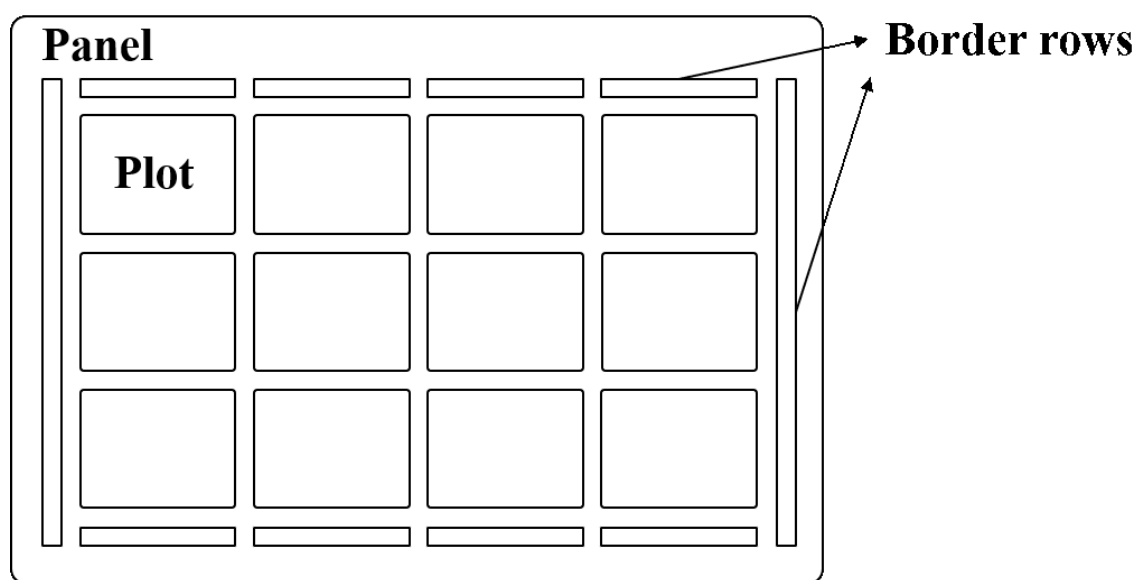


Fig. 1.9. Panel definition

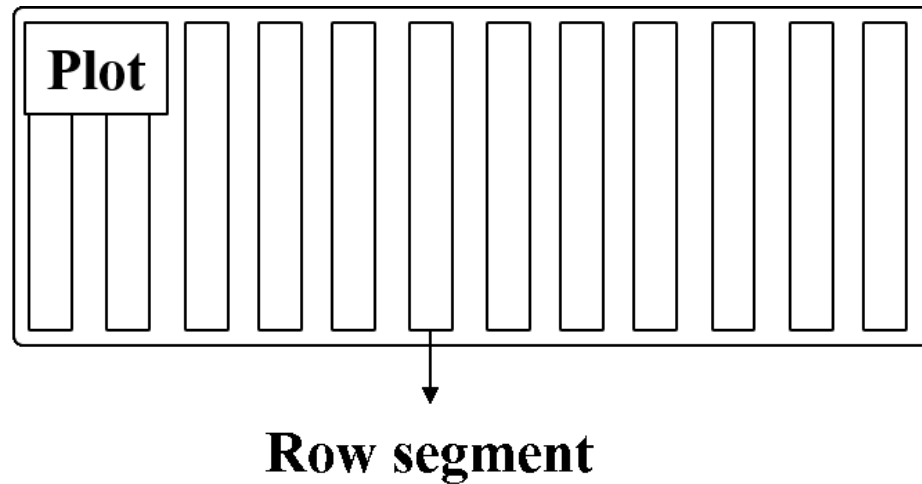


Fig. 1.10. Plot definition

Each panel contains different plots and border rows (as shown in Figure 1.9). Border rows are to separate panels and reduce border effects on the experiment. All plants in a plot share the same genetic line. Different plots typically have plants with different genes.

Each plot contains multiple row segments (as shown in Figure 1.10). A typical plot has 1 row, 4 rows, or 12 rows. Due to the border effect, the left and right most rows are not used in data analysis. Some of the rows are destructively sampled for data collection. Once the plants are removed, they can no longer provide data in the later growing stage. To solve the issue, the plot size is increased. A plot with 12 rows can have multiple destructive data collections on multiple dates. Rows that are not destructively sampled are harvested, and biomass and yield data are recorded.

A row segment is a basic unit within the plot (as shown in Figure 1.11). Row segment is often reduced at both ends to reduce the inter-plot border effect.

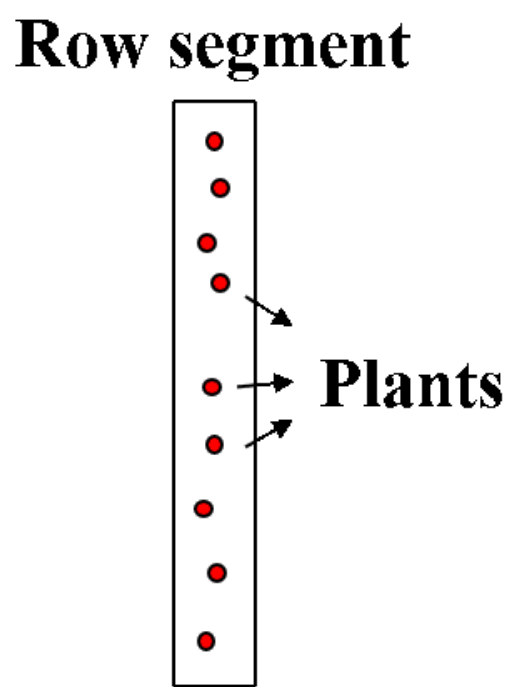


Fig. 1.11. Row segment definition

1.5 List of Phenotypic Traits

Example plot level traits:

- Flowering time
- Normalized Difference Vegetation Index (NDVI)

$$NDVI = \frac{NIR - RED}{NIR + RED}, \quad (1.1)$$

where NIR stands for near-infrared regions of the spectrum.

- Canopy cover Canopy cover is defined by the percentage of green area over the plant's occupying space.

Example row segment level traits:

- NDVI and other spectra indices
- Canopy cover
- Plant locations
- Plant count
- Leaf count
- Panicle count

Example plant level traits:

- Leaf angle
- Leaf area
- Leaf area index (LAI)

LAI is defined as the one-sided green leaf area per unit ground surface area (LAI = leaf area / ground area, m^2 / m^2) in broadleaf canopies.

- Leaf length
- Leaf width
- Leaf count
- Leaf angle
- Collar leaf height
- Leaf appearance rate
- Ear height

1.6 Plant Phenotyping with Machine Learning

With the rapid development of modern machine learning, scientists are able to apply it in the field of plant phenotyping. In [46], Zhou *et al.* proposed using a Random Forest [47] over color and texture features to segment plants and compute the canopy coverage. In [48, 49], Singh *et al.* give reviews on existing Deep Learning and machine learning methods studying plant stresses such as water stress, brown spot on leaf, mold, and disease. In [50], Ubbens *et al.* used Deep Learning for leaf counting, mutant classification, age regression. In [51], Alkhudaydi *et al.* segments wheat spike regions using Fully Convolutional Network. In [52], Pound *et al.* uses a U-Net [53] shaped Deep Learning architecture for wheat spikes and spikelets detection. In [54], Toda *et al.* study how learned CNN features relate to plant diseases. In [55], Pound *et al.* use CNN to localize and classify plant shoots. In [56], Masjedi *et al.* use Support Vector Regression and Multi-layer Perception for biomass from Light Detection And Ranging (LiDAR) point clouds and hyperspectral data.

In addition, many open-sourced tools and libraries are developed for plant scientists to use. PlantCV [57] is a computer vision based image phenotyping library that is implemented in python. The website <https://plant-image-analysis.org>, described in [58], is an online database for plant phenotyping tools.

1.7 Contributions of This Thesis

In this thesis, I propose multiple methods to estimate high precision phenotypic traits from UAV, Phenorover, and ground reference images including plant locations, the number of plants per plot, leaf area, leaf count, canopy cover, LAI, leaf length, leaf width, and leaf angle. Methods are tested and verified on image data collected by various platforms in fields of West Lafayette, Indiana, USA.

The main contributions of this work are:

- I describe an image phenotyping system that takes data collected from UAV and estimates phenotypic traits per-field, per-plot, per-row-segment, and per-plant.

- I create several tools, including orthophoto rotation estimator and inverse-mapping to facilitate data flows in the system.
- I develop a distributed computing system with a web interface for plant scientists to use our tools and visualize the results
- I propose a pixel-wise plant segmentation method using thresholds in HSV space.
- I propose a leaf location method using MIL.
- I propose a leaf location method using CNN.
- I present a leaf segmentation method by a shape-based approach.
- I propose a leaf segmentation method by segmenting and modeling individual leaves in polar coordinates.
- I propose a leaf segmentation method by modeling leaf shapes with polynomial functions.
- I propose a method to estimate the length and width of leaves that are laying on the table.
- I describe a method to detect the leaf angle for images taken from a ground vehicle.
- I experiment training a DNN by the relationship between input samples.
- I explore the use of logical “AND” operation in DNN.

1.8 Publications Resulting From This Work

1. **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. **Y. Chen**, J. Ribera, C. Boomsma, and E. J. Delp, “Locating Crop Plant Centers From UAV-Based RGB Imagery“, *Proceedings of the IEEE International Conference on Computer Vision, Workshop on Computer Vision Problems in Plant Phenotyping*, October 2017, Venice, Italy.

3. **Y. Chen**, J. Ribera, C. Boomsma, and E. J. Delp, “Estimating Plant Centers Using A Deep Binary Classifier“, *Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation*, April 2018, Las Vegas, Nevada
4. **Y. Chen**, S. Baireddy, E. Cai, C. Yang, and E. J. Delp, “Leaf Segmentation by Functional Modeling“, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, June 2019, Long Beach, CA
5. J. Ribera, D. Guera, **Y. Chen**, and E. J. Delp, “Locating Objects Without Bounding Boxes“, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2019, Long Beach, CA
6. J. Ribera, **Y. Chen**, C. Boomsma, and E. J. Delp, “Counting Plants Using Deep Learning“, *Proceedings of the IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, November 2017, Montreal, Canada
7. J. Ribera, F. He, **Y. Chen**, A. F. Habib, and E. J. Delp, “Estimating Phenotypic Traits From UAV Based RGB Imagery”, *ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Workshop on Data Science for Food, Energy, and Water*, August 2016, San Francisco, CA.

2. PLANT PHENOTYPING SYSTEM

2.1 Image Phenotyping System

Collecting plant-level data produces accurate data correspondence, but the process is time-consuming, labor-intensive, and sometimes difficult to carry out. In an outdoor environment, crops are grown densely. Accessing plants can be problematic and destructive. Therefore, collecting data per plant is less favorable for high throughput plant phenotyping. Modern imaging platforms such as UAVs have enabled collective data acquisition. These platforms can capture image data of a field in less than an hour, but establishing accurate data correspondence is not as straightforward. Manually sorting the captured images and labeling them are time-consuming and labor-intensive tasks. Orthorectification techniques described in [45] projects the spectral information captured from UAV imageries to a flat surface with geocoordinates. With the geocoordinates for each plot, we can assign the plot with its pixel-level sensor data, which is used for traits estimation.

We describe an image-based plant phenotyping system for RGB images that computes the data correspondences and estimates traits at per plot or per plant basis. In addition, this system includes a distributed computing platform with a web interface that makes our tools easy to access and use for the plant scientists.

We show a simple block diagram of the system in Figure 2.1, and a detail version of the system flowchart in Figure 2.2.

The system takes the RGB, Global Positioning System (GPS), and Inertial Measurement Unit (IMU) data generated by the UAV platform as input, and outputs the estimated traits for selected plots. The components of the system are described below.

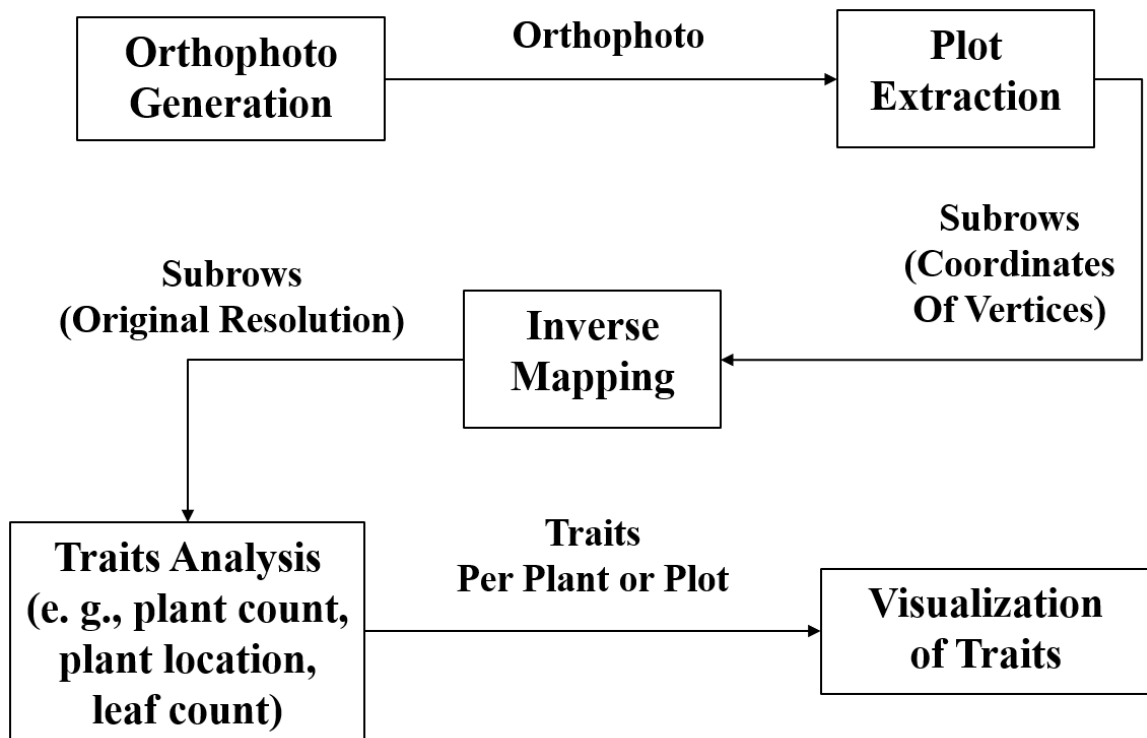


Fig. 2.1. Phenotyping system blockdiagram

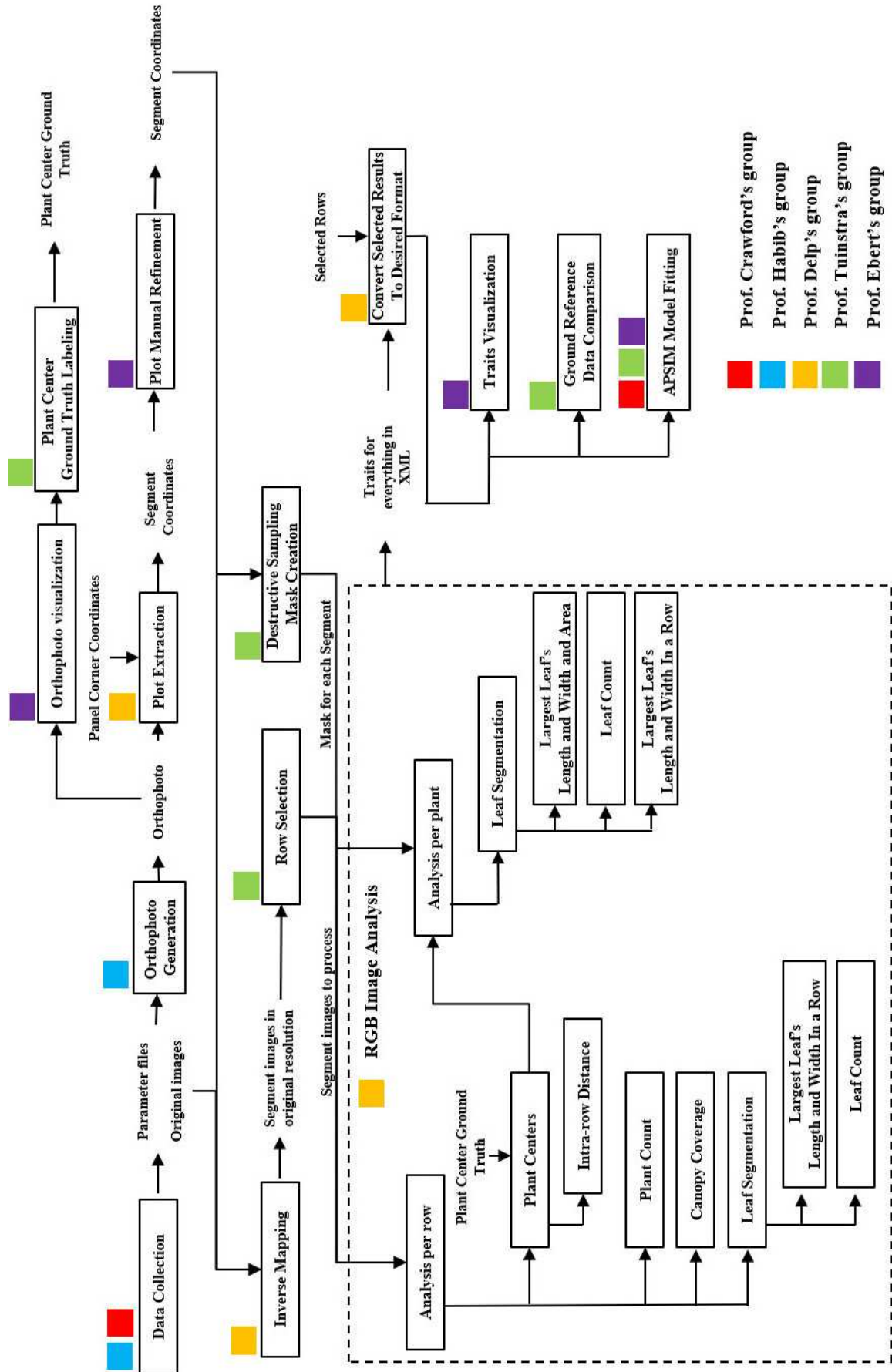


Fig. 2.2. Phenotyping system detail flowchart

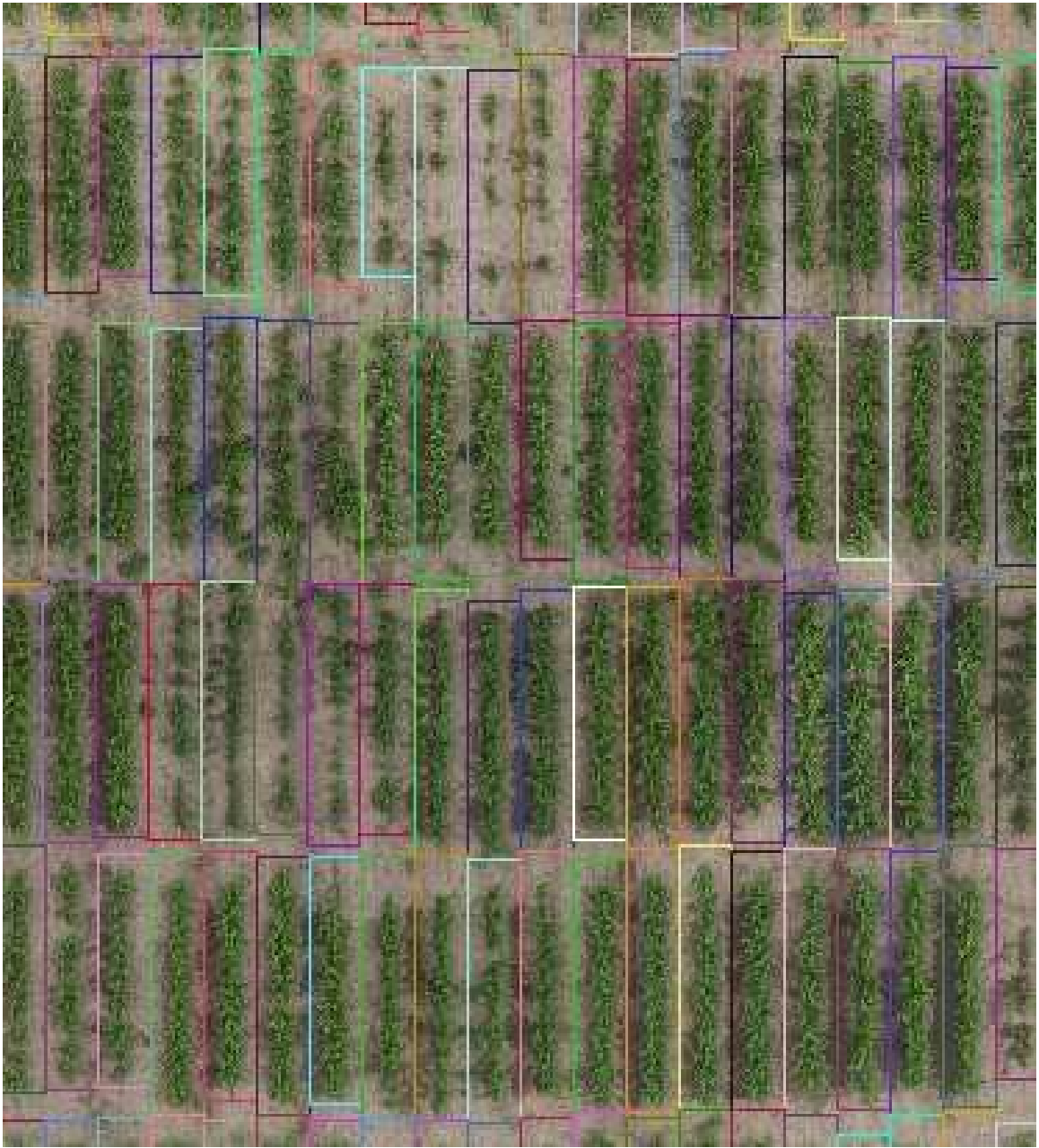


Fig. 2.3. An example of plot extraction, each row segment is marked in a colored rectangle.

- Orthophoto generation. With images and GPS/IMU data captured from UAV, system level, and camera parameters, orthophoto is created. This is accomplished by other members of the team, and this process is described in [45].
- Plot extraction (as shown in Figure 2.3). Plant experiments are planted with precision planters which use Global Navigation Satellite System (GNSS), resulting in well characterized row/plots. Plot extraction, described in [59], is a tool that automatically estimates a grid to divide the field into row segments. Each grid cell is represented by a bounding box, and the corners of the bounding box are represented in geocoordinates. The grid allows us to map RGB images to corresponding experimental plots in the field. This grid is essential for precise image-based plant phenotyping, since any estimated phenotypic traits from the cropped row segment image can be registered with its plot number and used by plant scientists. In a commercial setting, row segments in a field may not align due to planting conditions, weather, and ground conditions. For this kind of field, plot extraction can be applied multiple times on different parts of the field to obtain a plot grid.
- Inverse-mapping. Inverse-mapping crops row segments in their original resolution using the bounding boxes generated by the plot extraction.
- Trait analyses. The cropped row segments are then used for traits analysis. Row-segment-level traits analysis includes canopy cover, leaf count, plant count, and plant spacing. Once plant centers are detected in the row segment, traits can be estimated at the plant-level.
- Visualization. Estimated traits are visualized for better understanding. This is accomplished by other members of the team.

In addition, a storage unit Data Depot is shared between the teams. We propose a protocol for storing data and naming convention for the files. Figure 2.4 shows the proposed file structure. Figure 2.5 shows an example of the file structure. We propose the following rules for file and directory names:

- All lowercase

- No spaces
- Use underscore instead of dash
- Dates as year month and day, such as 20180418
- If there is a number in the name, pad the numbers with many zeros, such as 0041

Example: image_0041_f41_calibration_panel_20180714.png

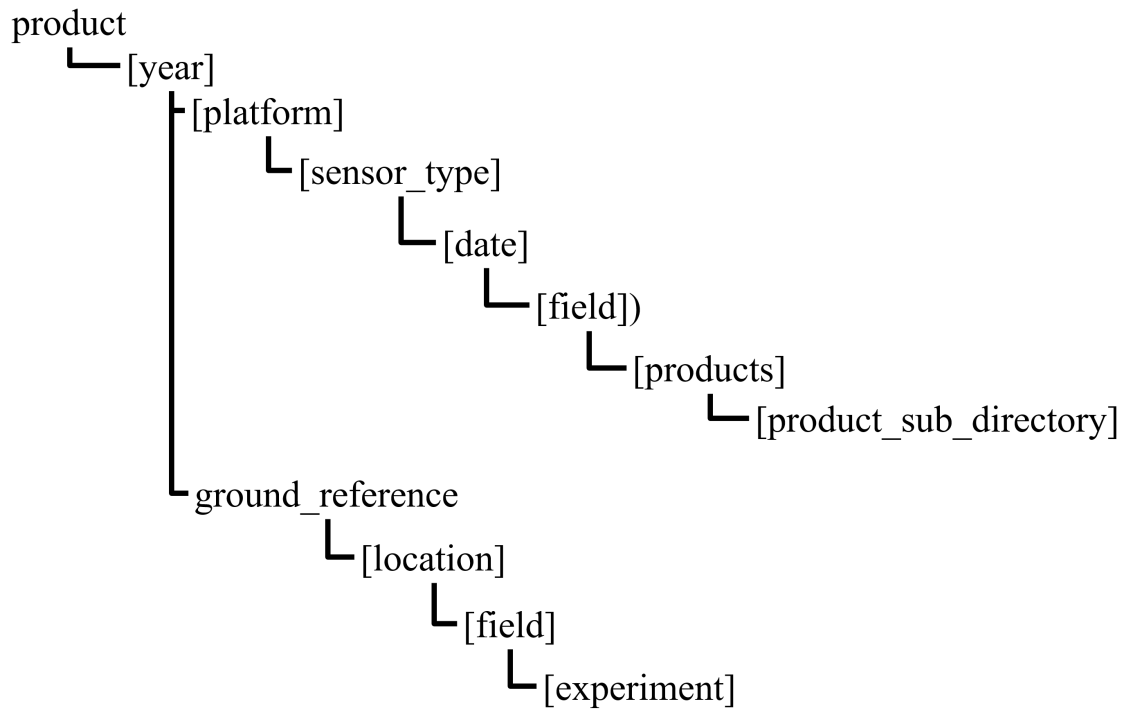


Fig. 2.4. File structure for storage

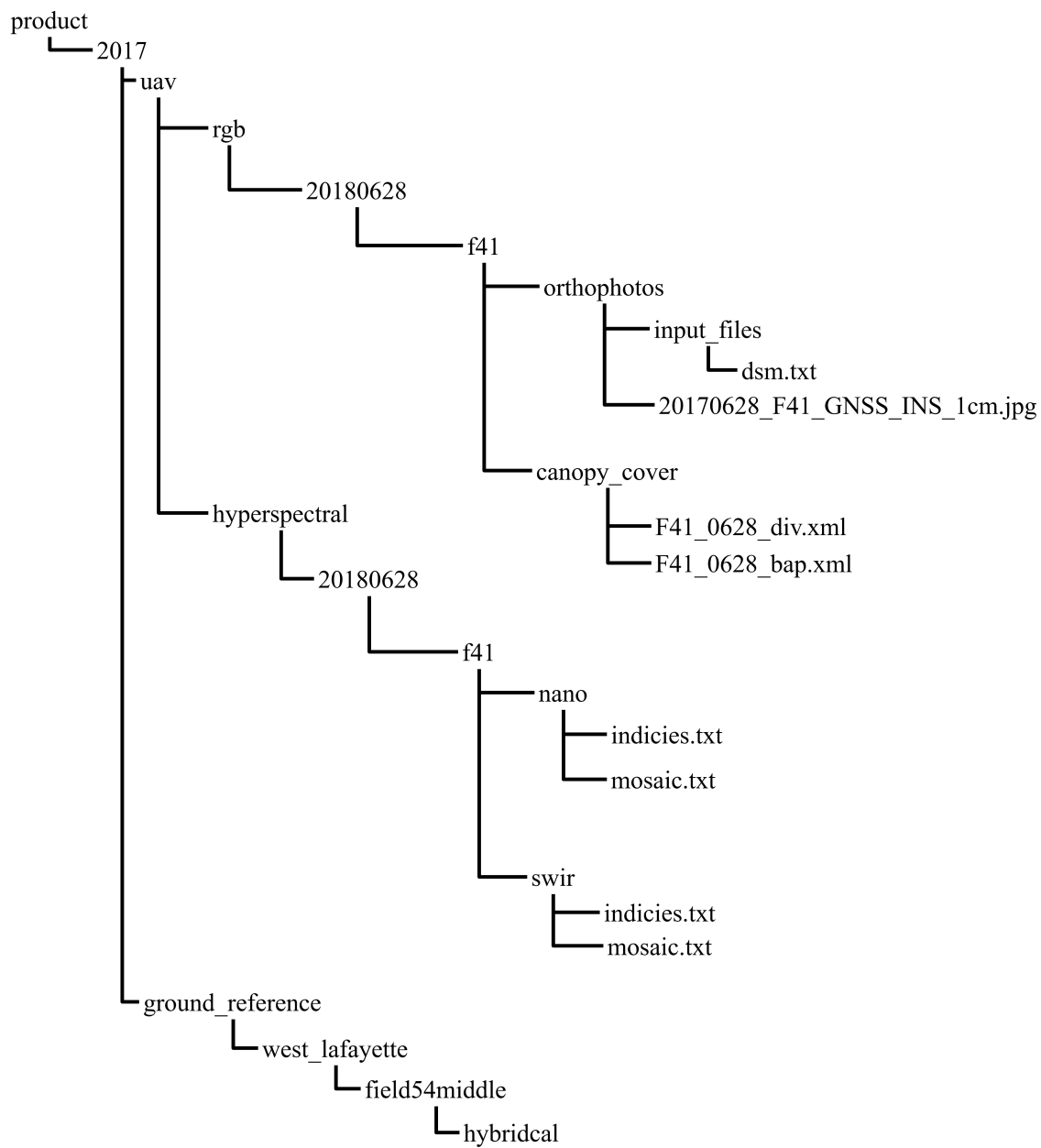


Fig. 2.5. File structure example

2.2 Tools

2.2.1 Orthophoto Rotation Estimation

The plot extraction tool we developed is based on the assumption that row segments are aligned with a North-South axis. A rotation of a field results in an inaccurate grid generation, which is crucial for image-based phenotypic trait estimation. In commercial settings, fields often are not aligned with the North-South axis due to various reasons such as maximizing space usage, managing erosion, and avoiding wet spots. Having the field data aligned with orthogonal rows and columns simplifies image processing, since there is no need to consider the rotation angle. The offset rotation angle that offsets the field is not available during the orthophoto generation. To find the rotation angle, we propose the following method.

First, we transpose the image if the row segments are horizontal. Let $I(x, y)$ to be the pixel intensity of an orthophoto at coordinate (x, y) . Let h to be the height of the image and w to be the width of the image. We define M to be the plant mask, and

$$M(x, y) = \begin{cases} 1 & \text{if } I(x, y) \text{ contains plant material} \\ 0 & \text{otherwise.} \end{cases} \quad (2.1)$$

Let M_θ to be the plant mask with a rotation of θ degree, where

$$M_\theta(x, y) = \begin{cases} M(x \cos \theta, y \sin \theta) & 0 \leq x < w, 0 \leq y < h \\ 0 & \text{otherwise} \end{cases} \quad (2.2)$$

We find the vertical projection profile $P_{x,\theta}$ by

$$P_{x,\theta} = \sum_{y=1}^H M_\theta(x, y) \quad (2.3)$$

Figure 2.6 shows an example of a vertical projection profile.

Let τ to be the maximum rotation angle of the orthophoto. Then the optimal rotation angle $\hat{\theta}$ is estimated by

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}(\operatorname{Var}(P_{x,\theta})), \quad (2.4)$$

where $Var(\cdot)$ is the variance function.

Once the rotation angle is estimated, the rotated orthophoto is computed by

$$I_{rot}(x, y) = \begin{cases} I(x\cos\theta, y\sin\theta) & 0 \leq x < w, 0 \leq y < h \\ 0 & otherwise \end{cases} \quad (2.5)$$

The image coordinates (x, y) associated with the original orthophoto I is now $(x\cos\theta, y\sin\theta)$.

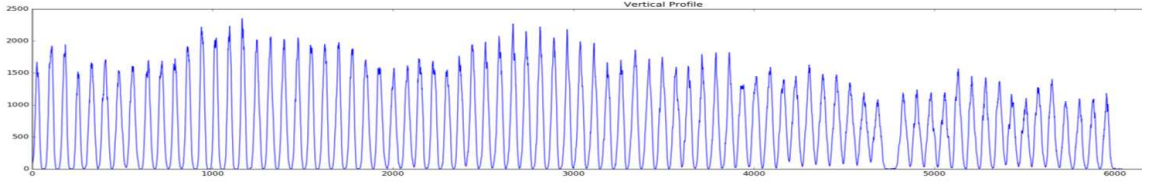
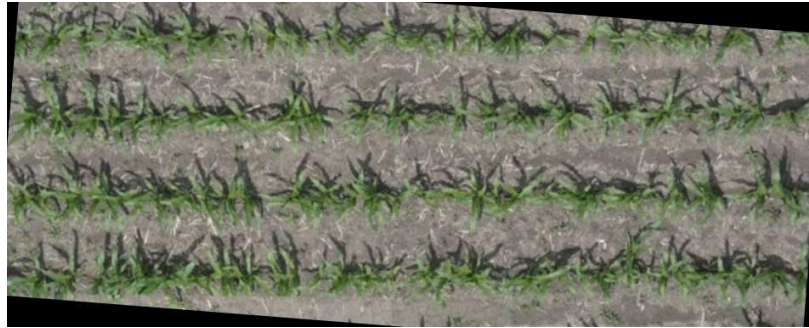


Fig. 2.6. A vertical projection profile example



(a)



(b)

Fig. 2.7. a) an example input image. b) The rotated input image.

2.2.2 Inverse Mapping

Images captured from a UAV may have different spatial resolutions due to physical conditions such as flying altitude, and pitch or roll angle. These raw images are used to generate orthophotos with uniform spatial resolutions. The resolutions are selected based on the application. For example, some application requires maximizing the processing speed so the orthophotos can be visualized within hours. Estimating traits directly on the orthophoto can be challenging due to the low resolution and artifacts during the generation process. In addition, orthophoto generated pixels do not guarantee the spatial continuity of the image values, which means the image contains pixels from different source images. This causes problems in image analysis that requires spatial continuity such as leaf shapes. We developed an inverse mapping tool that retrieves image crops in original resolution with the given plot coordinates.

This tool projects the four points of each row segment back into the geo-coordinates, and find their closest points. Each point belongs to one source image. We find a list of source images that satisfy the condition where all 4 corner points are in the same image. We then find the center of the row segment by averaging all 4 coordinates. We select the image with a center closest to the row segment center, since the center of each image has less perspective distortion. Figure 2.8 shows the image selection based on row segment corner coordinates. Figure 2.9(a) shows an example of a row segment with orthophoto resolution. Figure 2.9(b) shows an example of the inverse-mapped row segment in original image resolution.

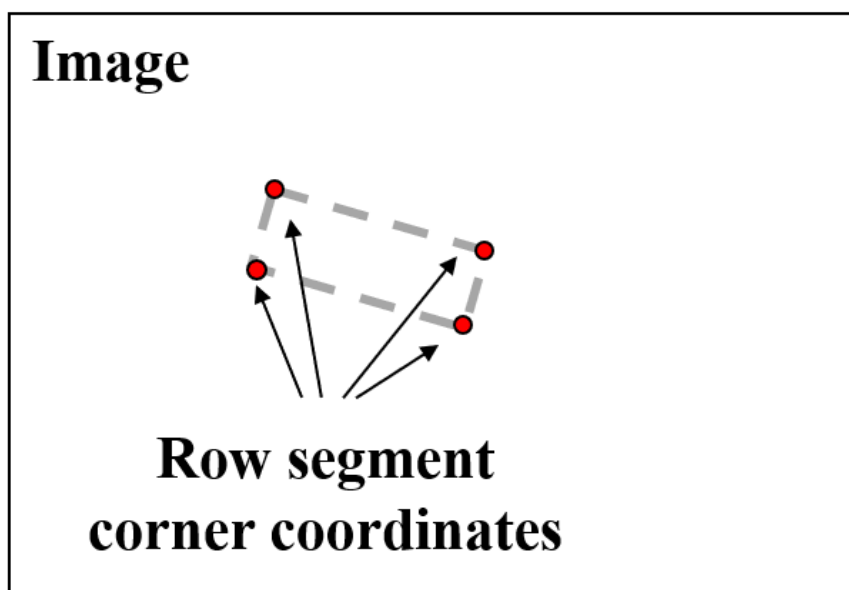
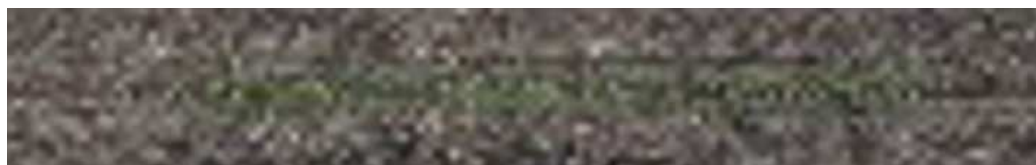
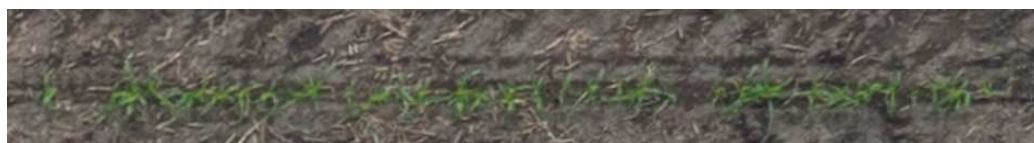


Fig. 2.8. Image selection for inverse mapping based on the corners of the row segment.



(a)



(b)

Fig. 2.9. An example of inverse-mapping: a) a row segment cropped from orthophoto with resolution 4cm/pixel. b) an inverse-mapped row segment with resolution 0.63cm/pixel.

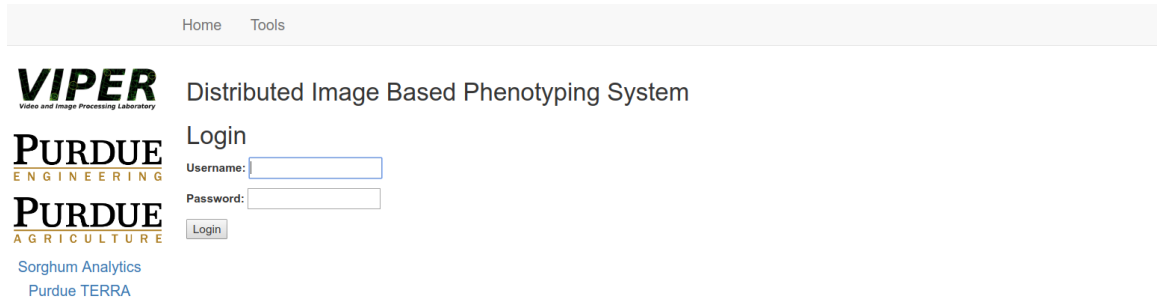


Fig. 2.10. Login page

2.3 Distributed Image-Based Phenotyping System

A distributed RGB image-based phenotyping system is developed to facilitate plant scientists for traits predictions. Among all the web frameworks, we choose Django [60] because its framework is python based and easy to use. The system runs on Linux and can be deployed to Amazon Web Service [61]. Anyone with an account can login and use the provided image analysis tools without the need to have computing resources.

Our web interface has few components.

- Login page (as shown in Figure 2.10). User needs to enter their login credentials before using our tools.
- Home page (as shown in Figure 2.11). Once login is successful, user will be brought to this page. Clicking on "Tools" will direct the user to the image phenotyping tools menu.
- Data upload page (as shown in Figure 2.12). This page is used for uploading input data for processing. Multiple data samples can be uploaded. Clicking on "Basic Upload" in "Menu" will direct you to this page.
- Image Phenotyping Tools page (as shown in Figure 2.13). Image Phenotyping Tools page contains all the provided image analysis tools, including plot extraction, plant location, and leaf segmentation. In addition, plot viz is a vi-

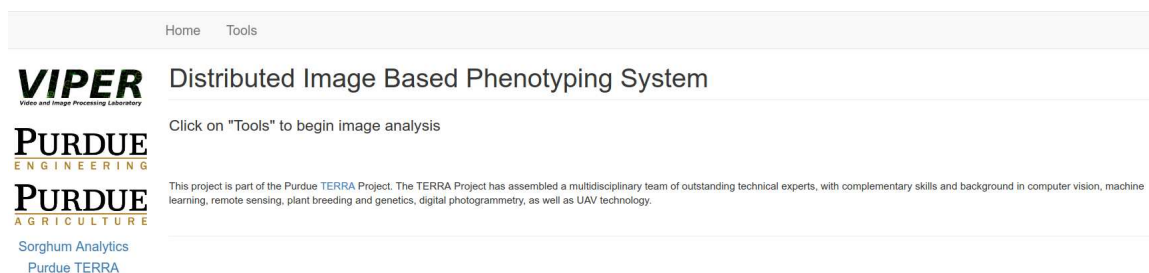


Fig. 2.11. Home page

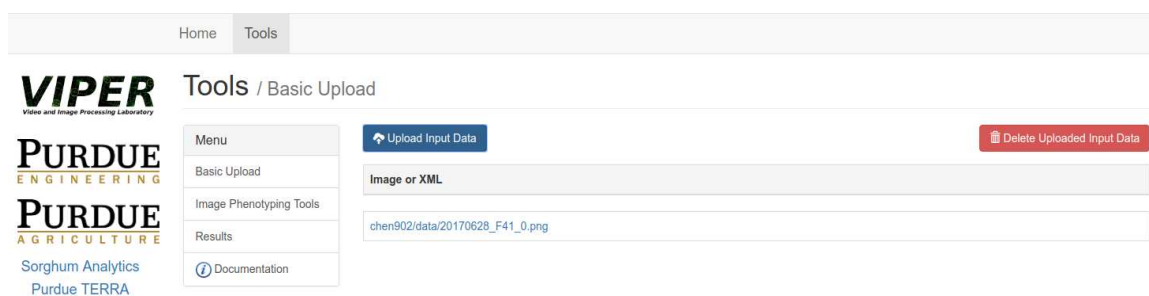


Fig. 2.12. Data upload page

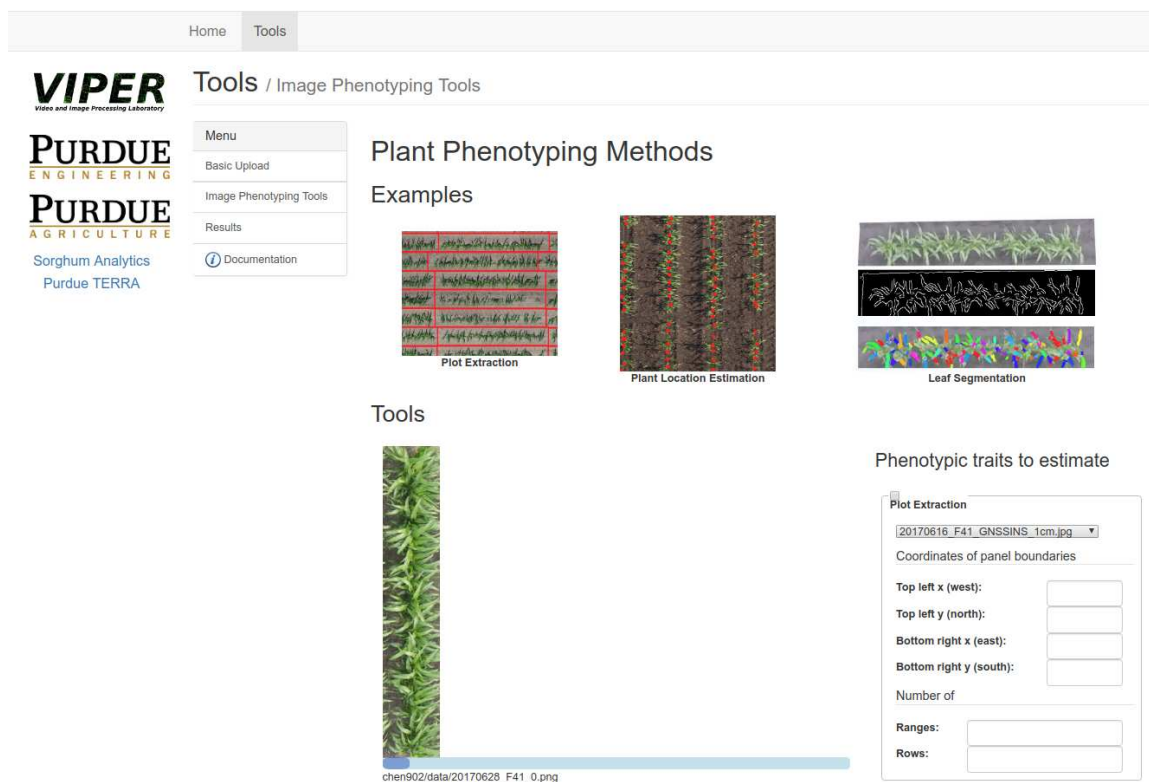


Fig. 2.13. Image phenotyping tools page

sualization tool to view the results of plot extraction. The user needs to input the required parameters, select the tool to run, and click the "Process" button. Once the processing is finished, the user will be redirected to the results page where outputs can be downloaded.

Most of the tools take input from the upload page and produce outputs on the results page without any parameters. Figure 2.14 shows the tool selection box. Plot extraction requires several input parameters, including the coordinates of the region of interest, and the number of ranges and rows. To obtain those parameters is not straightforward. Therefore, we implement a map feature to display the orthophoto. Users can select the ROI by clicking on the rectangle box in the map interface. The coordinates of the ROI will be automatically filled into the parameter box. Figure 2.15 shows this map interface.

Phenotypic traits to estimate

Plot Extraction

20170616_F41_GNSSINS_1cm.jpg ▼

Coordinates of panel boundaries

Top left x (west):

Top left y (north):

Bottom right x (east):

Bottom right y (south):

Number of

Ranges:

Rows:

☐ Process Uploaded

- File 1 Leaf Seg ▼

☐ Plot Viz

20170616_F42_GNSSINS_4cm.jpg ▼

☐ Plant Location

☒ Leaf Segmentation

 Process

Fig. 2.14. Available image phenotyping tools

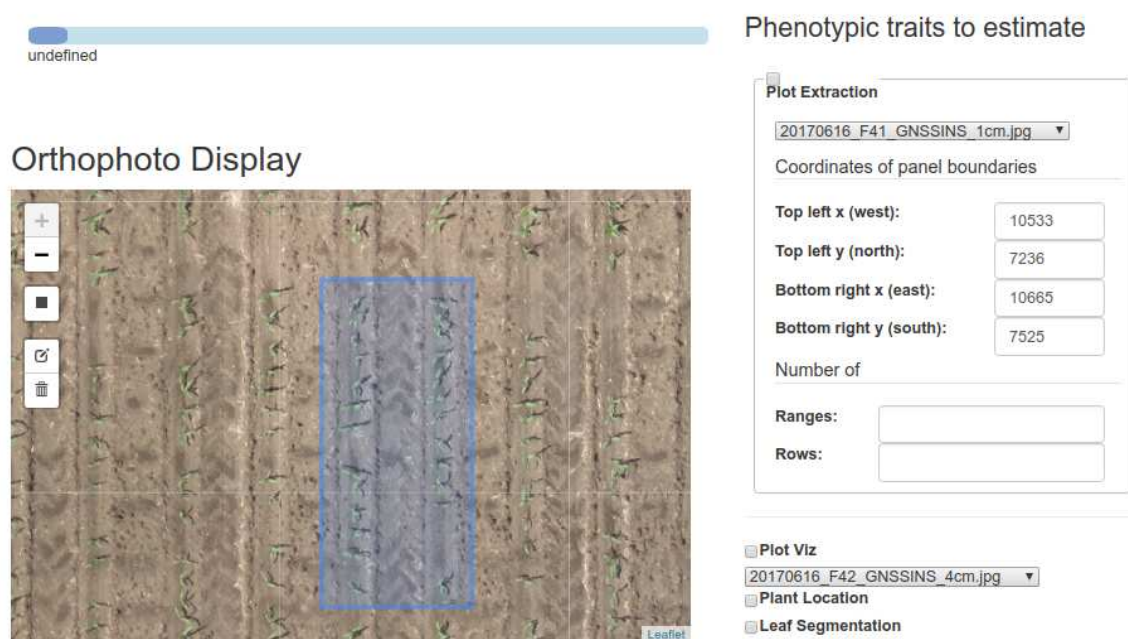


Fig. 2.15. Region of interest selection on the map view of an orthophoto

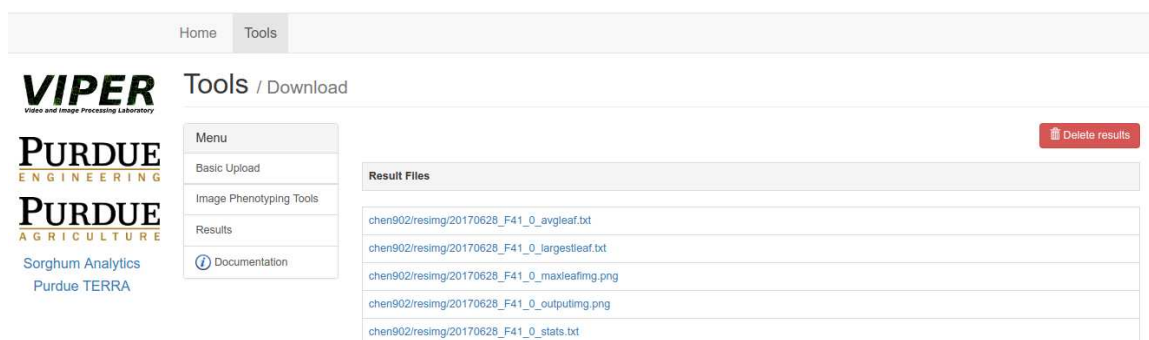





Fig. 2.16. Results page

- The results page (as shown in Figure 2.16). Results page shows the list of outputs corresponding to the input data. The inputs and outputs share the same prefix. Some tools may generate multiple output files for one input sample. All outputs can be downloaded by right click and clicking "save link as".
- Documentation page (as shown in Figure 2.17). Documentation page contains instructions, requirements, and output descriptions for each image phenotyping tool.

Home

Tools

Sorghum Analytics
Purdue TERRA

Tools / Documentation

Menu

Basic Upload

Image Phenotyping Tools

Results

Documentation

Plot Extraction

The Plot Extraction tool obtains the coordinates of each row segment in a field panel from an orthophoto.

Demo Instructions

1. Click on "Image Phenotyping Tools".
2. Check the "Plot Extraction" box.
3. Select an orthophoto from the drop-down list.
4. Provide the pixel coordinates of the panel boundaries manually OR use the button "Draw a rectangle" in the "Orthophoto Display" section and drag the cursor to obtain the coordinates from the rectangle you draw.
5. Provide the number of ranges and rows in the selected square (panel).
6. Click on "Process".
7. Once the process is done, it will take you directly to the results page.
8. You can view/download the result by clicking on the file.

Requirements

- The input must be a very well rectified orthophoto.
- The row segments must be vertical in the orthophoto (aligned top-bottom).
- The analysis is constraint to a constant number of ranges and rows. The most common use case is selecting a single panel.
- The plants must be at an intermediate growing stage (plants must not be too small, but the canopy must not be completely closed).
- The plants must be greenish (panicles should not be present and plants should not be dry, which would make them brownish).

Output

The units in the output XML files are pixels.

Plant Location

The Plant Location tool estimates the center of the stalk of all the plants in an image.

Demo Instructions

1. Upload/select your image.
2. Click on "Image Phenotyping Tools".
3. Check the "Plant Location" box.
4. Click on "Process".
5. Once the process is done, it will take you directly to the results page.
6. You can view/download the result by clicking on the file.

Requirements

- The input must be a single image containing any reasonable number of plants (between 0 and ~100).
- The plants must be similar to the plants used during training (e.g. color, growing stage, canopy closure, or number of row segments per image). If this requirement is not met, the tool will return a low confidence score. The tool is not intended to be used on images of plants that are not similar to the training data.

Fig. 2.17. Documentation page

3. PLANT MATERIAL SEGMENTATION

3.1 HSV Color Segmentation

3.1.1 Motivation

Canopy coverage is an important phenotypic trait that indicates the growth of plants. To compute canopy cover, a plant material mask is required. I propose a pixel level segmentation method to estimate plant material mask since it has low computation cost and requires no training data.

3.1.2 Proposed Method

First, the input image is converted from RGB to HSV color space [62], because HSV color model aligns with how human sees color. From the image in the HSV color space, a segmentation mask Y is generated. Each pixel Y_m in this mask is obtained as:

$$Y_m = \begin{cases} 1 & \text{if } \tau_1 \leq H_m \leq \tau_2 \text{ and } (\tau_3 \leq S_m \text{ or } \tau_4 \leq V_m) \\ 0 & \text{otherwise} \end{cases} \quad (3.1)$$

where H_m , S_m , and V_m are the hue, saturation, and value of the pixel m . The thresholds τ_1, τ_2, τ_3 , and τ_4 are determined experimentally. τ_1 and τ_2 are the upper boundary and lower boundary in hue for selecting the green color of the leaves. Soil pixels within plant's shadows have similar hue as the plants, but they are low in both saturation and value. τ_3 prevents misclassifying soil pixels in shadows as leaves. τ_4 accounts for leaf's highlight color region where pixel's value is high and saturation is low.

This pixelwise segmentation makes use of the strong color difference between the sorghum leaves, which are generally green or yellow, and the soil, which is usually

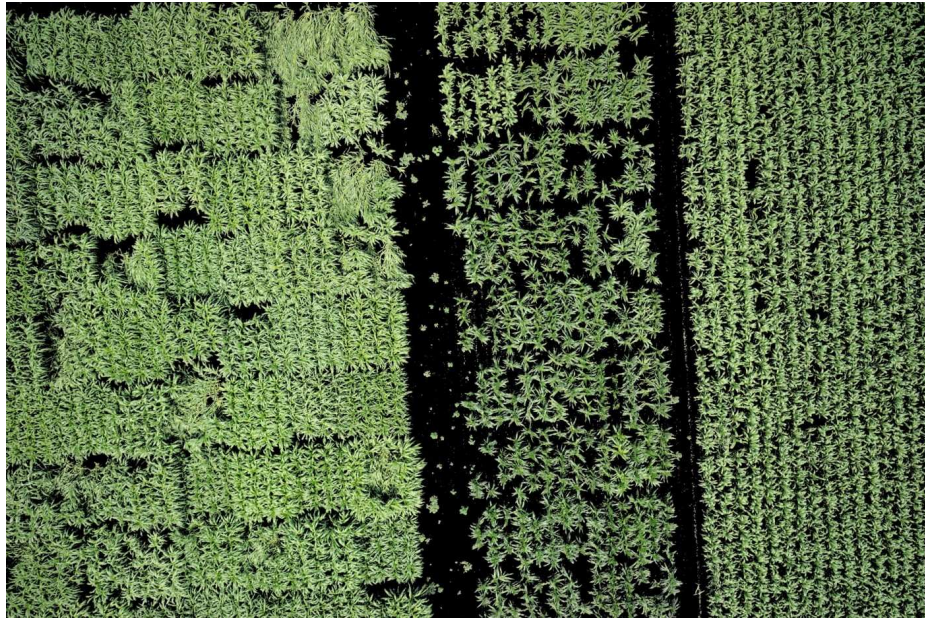
brown. The segmentation also addresses problems from shiny leaves and soil shadow regions where hue is not a reliable information to be used for classifying plant, soil, and panicles.

3.1.3 Experimental Results

An example of the segmentation result is shown in Figure 3.1 and Figure 3.2.

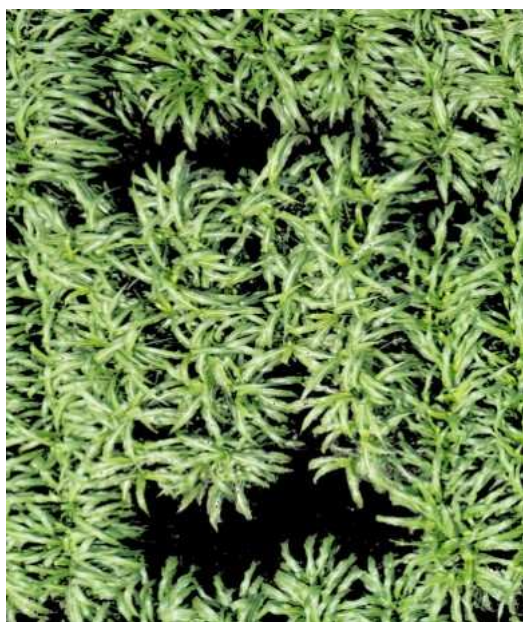


(a)

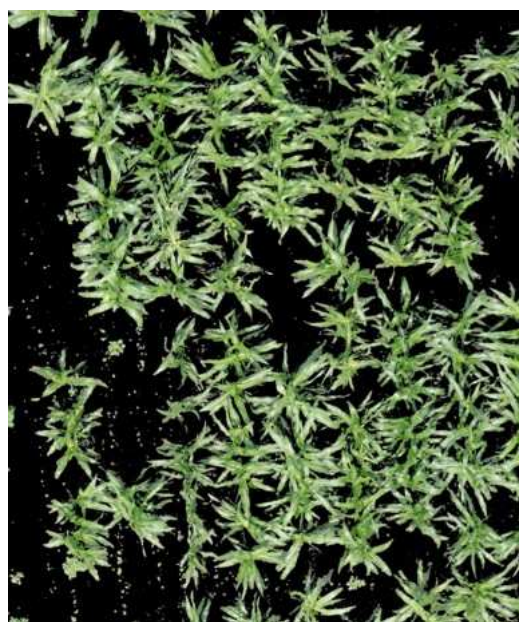


(b)

Fig. 3.1. a) Original image. b) Segmentation results.



(a)



(b)

Fig. 3.2. Details of segmentation results

4. PLANT LOCATION AND COUNTING

4.1 Introduction

Identifying individual plants is a crucial step in analyzing UAV image data in field-based plant-level phenotyping. Since leaves spread out from a plant’s center, the location of the center is important to study the plant structure and distinguish leaves from leaves of neighboring plants. Distinguishing individual plants from their neighbors is essential to track the growth of each plant, and obtain precise estimates of plant traits.

Collecting UAV image data is very weather and time-sensitive. Plant centers are easier to detect and count when the plants are young. However, the time window for capturing young plant data is small. Farmers may miss the window for young plant data collection due to weather or various other reasons. Being able to locate plant centers and count them later in the season can be important. In this thesis, I focus on detecting plant centers for mature plants.

4.2 Overview of Deep Learning Methods

Since 2012, Convolutional Neural Networks (CNN) have been widely used in many image classification problems. CNN is a network that contains multiple layers of weights and convolutional filters. It takes an input image and computes convolutional filter responses. The resulting filter responses are input into a neural network, and the classification result is estimated. With the latest development of DenseNet [63], a top-5 error rates of 6.12% has been achieved on the ImageNet validation set [64]. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [64] is a standard benchmark for large-scale object recognition. It contains 1.2 million training images,

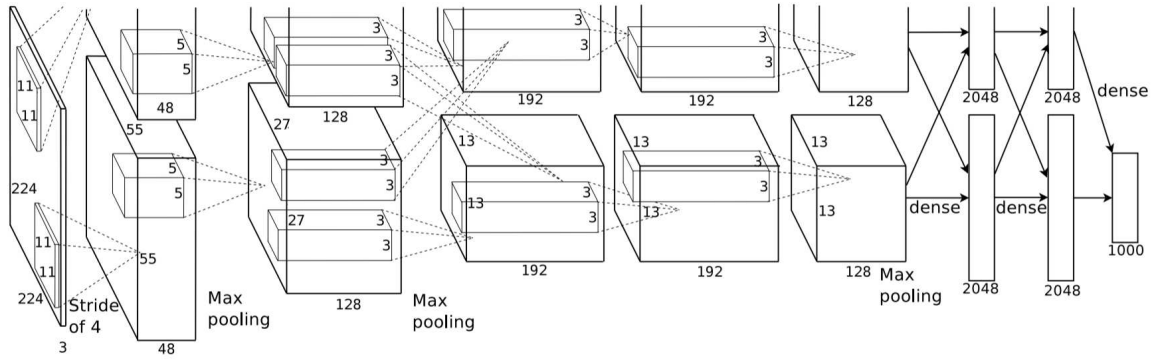


Fig. 4.1. AlexNet architecture [66]

50,000 validation images and 100,000 test images of 1000 object classes [64]. I review some popular CNNs.

AlexNet: In 1998, the CNN LeNet [65] was introduced and was successfully used to classify zip code. In 2012, expanded from LeNet [65], AlexNet [66] was proposed to have a larger and deeper architecture for object classification and won the ILSVRC with a top 5 error rate of 15.3%. AlexNet is designed to have larger capacities for learning complex object features. Figure 4.1 shows the architecture of AlexNet. Instead of using TANH as the activation function as in LeNet, AlexNet uses ReLU,

$$f(x) = \max(0, x). \quad (4.1)$$

ReLU's non-saturating nonlinearity provides a much faster training time with gradient descent than TANH [66]. To fight overfitting, AlexNet uses dropout and data augmentation techniques including patch extraction, horizontal reflections, and color alternation. Dropout is a technique which the network randomly sets the output of hidden neurons to zero, making the neuron not contribute to forward pass and backward propagation [66].

Deconvnet: In 2013, Zeiler and Fergus presented Deconvnet to visualize the activations of CNNs [67]. The network maps each activation back to the input pixel space, so researchers can see the relationship between input patterns and feature

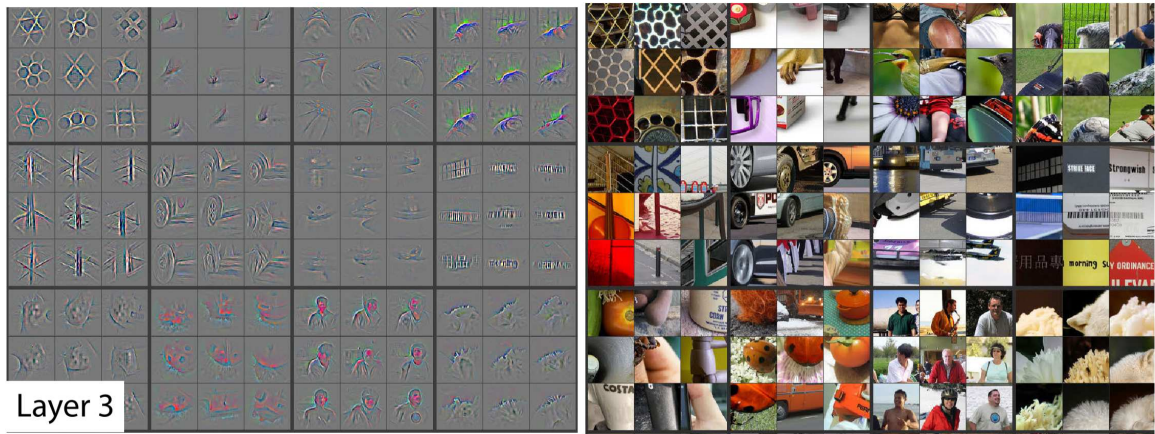


Fig. 4.2. Visualization of a layer in a CNN [67]

activations. Their work shows the process and contents that a CNN learns. Figure 4.2 shows visualizations of features learned by their CNN.

VGG: In 2014, VGGNet [68] was introduced to be a simpler and deeper CNN. One of the main contributions of the paper is its strict usage of a small size convolutional window. The paper argued that instead of using a large size convolutional layer, stacking multiple 3x3 convolutional layers has the same effective receptive field. With the additional activation layers between multiple small convolutional layers, features learned by multiple layers will be more discriminative than the features learned by just one layer. Moreover, the use of multiple small convolutional layers reduces the size significantly. The paper also investigates the effect of adding depth to CNN architectures and concluded that deeper networks achieve better accuracy.

GoogleNet: In 2015, rather than making network simpler, GoogleNet [69] introduced inception modules to make the network wider. An inception module shown in Figure 4.3, consists of pooling and different size convolutional layers in parallel. Figure 4.4 shows its architecture. While networks like AlexNet and VGGNet uses uniform structure across each layer, the inception module is invented to make use of filter-level sparsity to break the symmetry. GoogleNet achieved a top-5 error rate of 6.67% on the ILSVRC 2012 [64] dataset. The inception module is further developed

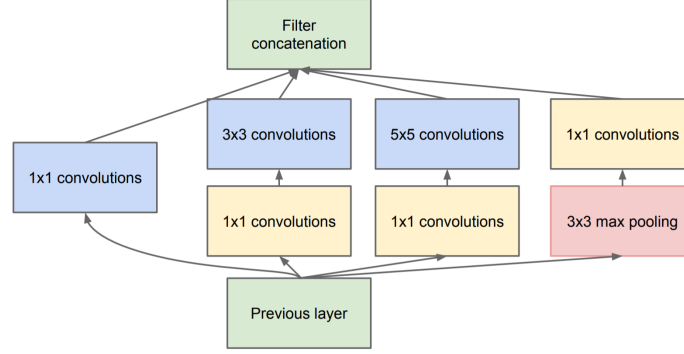


Fig. 4.3. The Inception Module of GoogleNet [69]

in [70] to avoid representational bottlenecks and balance the width and depth of the network. The top-5 error rate is improved to 3.58% on ILSVRC 2012 [64] dataset.

ResNet: Vanishing gradient [72, 73] is a major problem preventing CNN from going deeper. In 2015, Microsoft introduced ResNet, a 152 layer network architecture (Figure 4.5), using a residual learning block [71]. Let the mapping of stacked non-linear layers to be $F(x)$ for input x . The residual learning block (Figure 4.6) adds an identity connection from x to $F(x)$, and I denote the overall mapping $H(x)$ to be

$$H(x) = F(x) + x. \quad (4.2)$$

The paper argued that it is easier to optimize input referenced mapping, and this identity connection allows the gradient to skip intermediate layers.

Densenet: While ResNet add shortcuts to the network, DenseNet, introduced by G. Huang et al. in 2017, connects all layers directly with each other (shown in Figure 4.7, 4.8). ResNet combines features through summation where information could be potentially lost in the process. DenseNet combines features through concatenation, allowing the gradient to flow through the network freely.

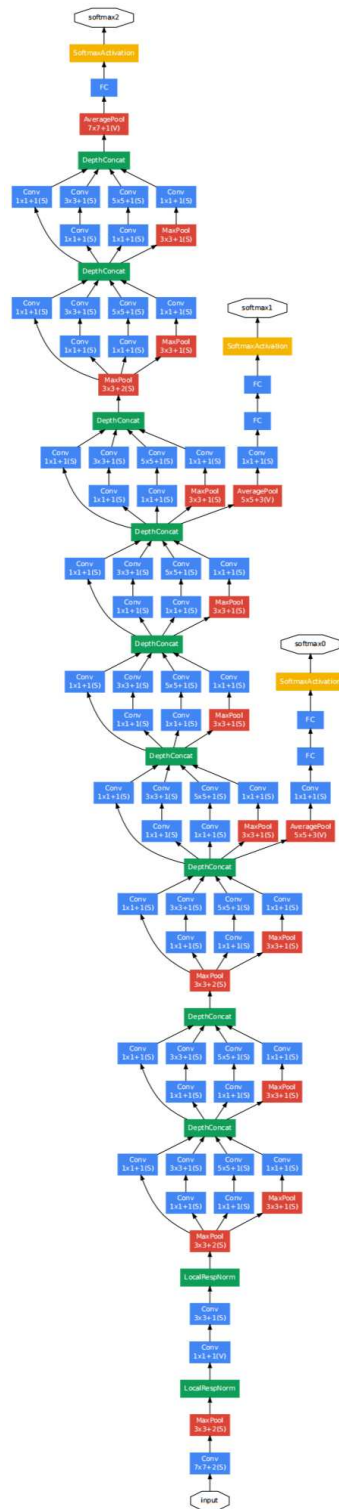


Fig. 4.4. GoogleNet architecture [69]

34-layer residual

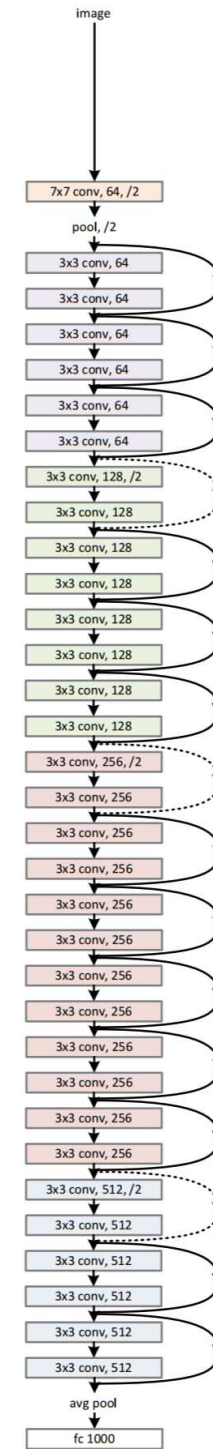


Fig. 4.5. An example 34 layer ResNet architecture [71]

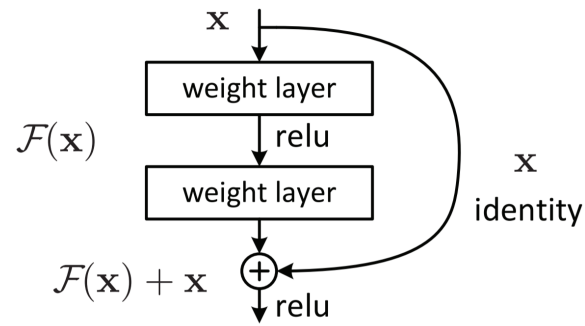


Fig. 4.6. The Residual Learning Block of ResNet [71]

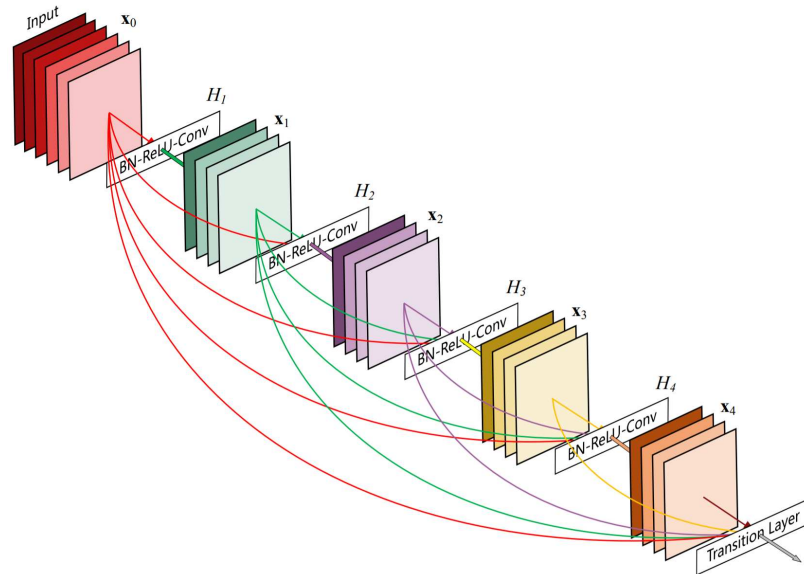


Fig. 4.7. The 5 layer Dense Block of DenseNet [63]

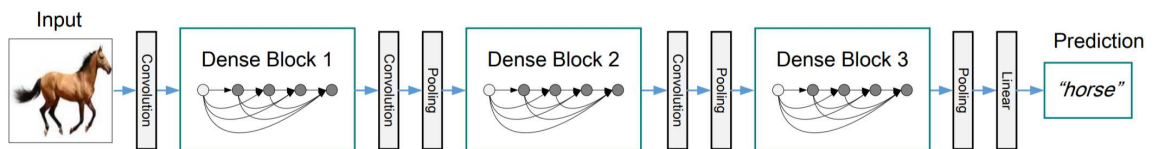


Fig. 4.8. DenseNet architecture [63]

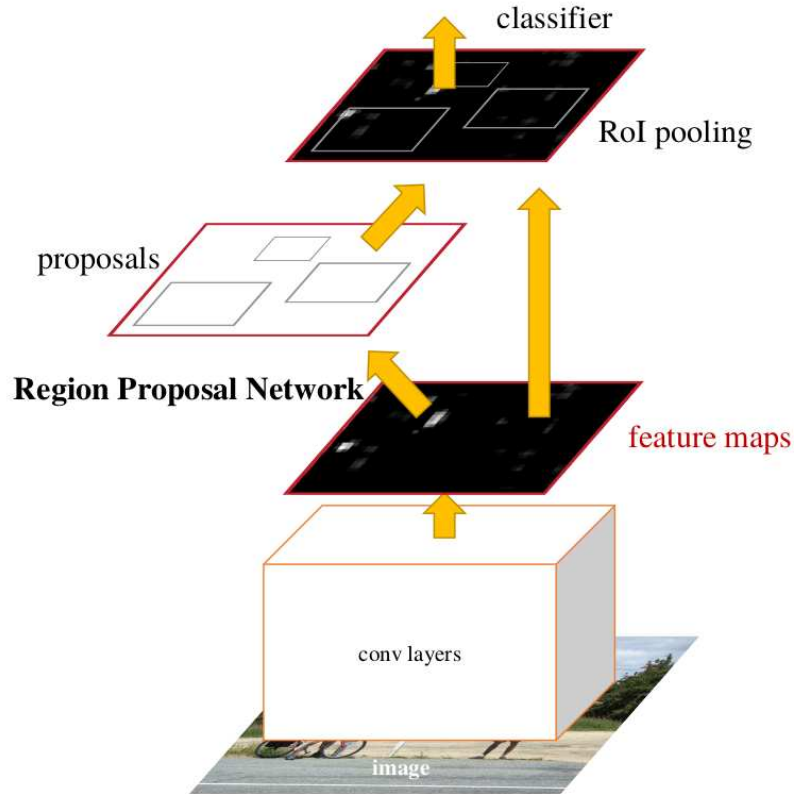


Fig. 4.9. Faster R-CNN architecture [74]

While object classification-based CNNs are excellent for classifying fixed-size images, object detection networks detect the location of the object and predict the label.

In R-CNN [77], candidate regions containing objects are searched using selective search [78]. CNN is used to generate feature vectors for selected regions. A Support Vector Machine classifier is used to classify these feature vectors. Fast R-CNN [79] is proposed to improve R-CNN for faster computation time. In Fast R-CNN, a feature map is computed first. Each proposed region pools a fixed-length feature vector from the feature map. The feature vector is input into a CNN to predict object label and refine bounding box coordinates. Faster R-CNN [74] improves the method further by replacing generic region proposal methods with a Region Proposal Network. Faster

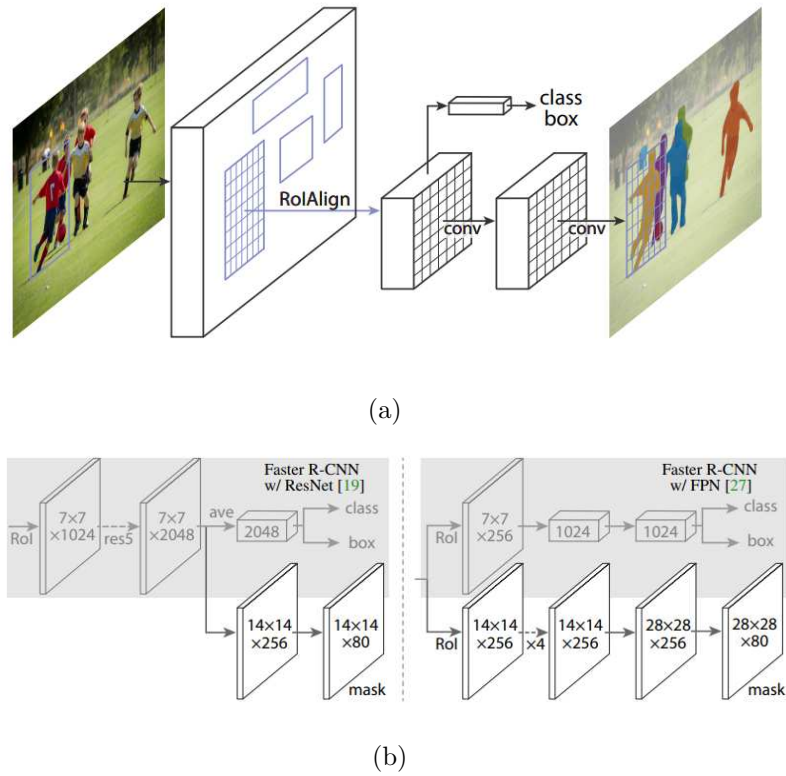


Fig. 4.10. Mask R-CNN architecture

R-CNN has been successfully used in [80] to locate fruits in images of trees. Figure 4.9 shows the architecture used by Faster R-CNN. Mask R-CNN [81] adds a branch to the Faster R-CNN to predict the segmentation mask for the detected bounding box. Figure 4.10 shows the architecture of Mask R-CNN. You Only Look Once (YOLO) [75, 82, 83] divides the input image into grids. Each grid predicts a number of bounding boxes it belongs to and their confidence scores. Objects are detected by thresholding the confidence scores. Figure 4.11 shows the detection block diagram of YOLO. In Single Shot Detection [76], Liu *et al.* added extra feature layers to a bounding box prediction network to detect objects at multiple scales. Figure 4.12 shows the architecture comparison between SSD and YOLO. In [84], Liu *et al.* improve the Non-Maximum Suppression in object detection networks by using Multiple

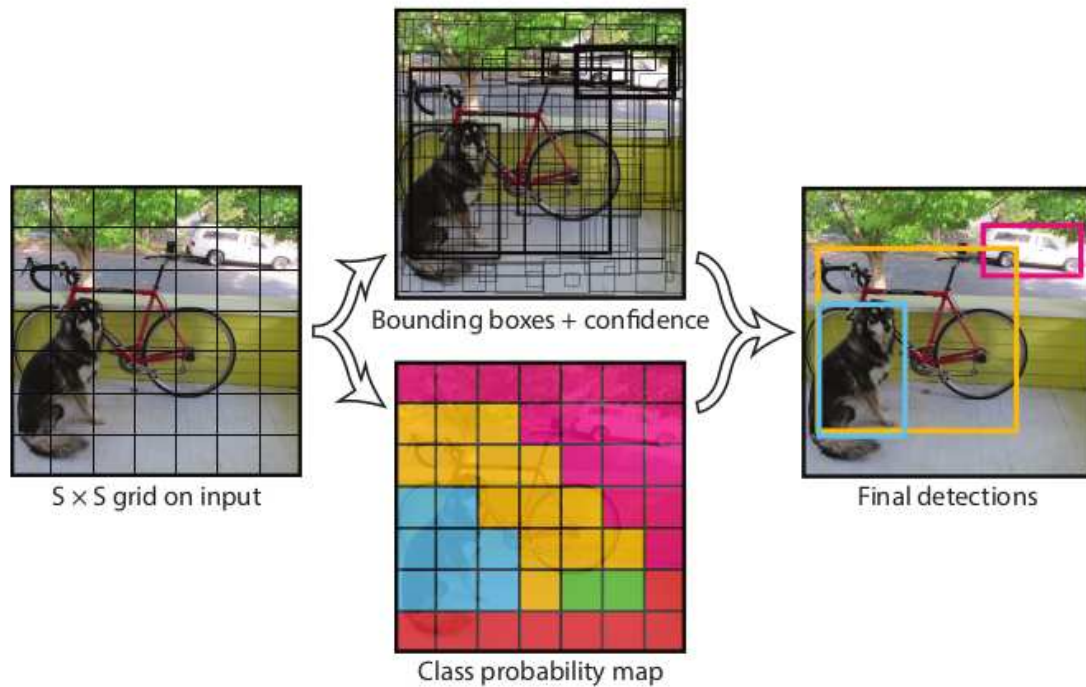


Fig. 4.11. YOLO object detection [75]

Instance Learning on the similar bounding boxes. Object detection methods can also be applied in tracking problems. In [85], a CNN is used to detect the location of objects in consecutive video frames. All pixels are classified as either being the object center or not. The input to the CNN is a bounding box around each pixel location. The classification score for each location forms a probability map, and the location of the object is estimated as by the pixel with the highest probability. Instead of detecting bounding boxes, some studies detect the keypoints of an object and reconstruct the bounding box or boundaries of the object. In CenterNet [86], the center point of an object is detected, and the bounding box is created based on the detection. In CornerNet [87], Law *et al.* detect the two corners of object bounding box. In ExtremeNet [88], Zhou *et al.* detect the extreme points of an object, and group

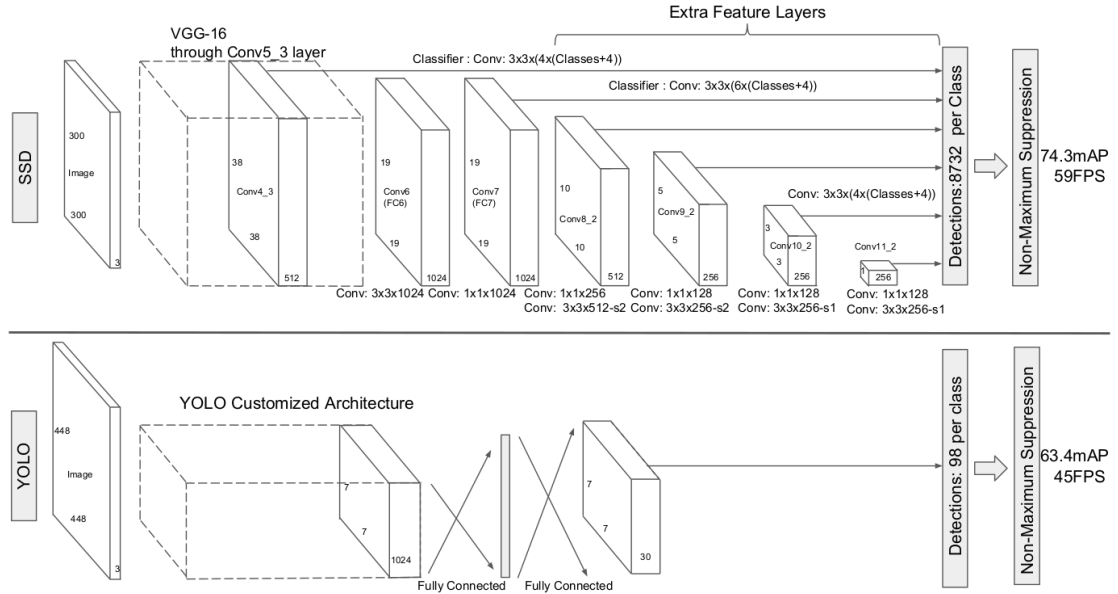


Fig. 4.12. SSD [76] and YOLO [75] architecture comparison

them to create a bounding polygon. Another study recreates bounding boxes from the detected center point [89].

Aside from deep learning architectures, several loss functions has been proposed to improve the learning in different applications. In [90], Barron proposed a loss function that generalizes regression losses including Cauchy/Lorentzian, Geman-McClure, Welsch/Leclerc, generalized Charbonnier, Charbonnier/pseudo-Huber/L1-L2, and L2 loss functions. In Focal Loss [91], Lin *et al.* proposed a new loss for dense object detection. The loss function increases the weight of difficult samples while reduces the weight of easy samples.

4.3 Plant Location Related Works

In previous studies, plant locations have been estimated using two different types of datasets. The first type of dataset contains high resolution images collected in greenhouse environments. Prior information about the morphology of a particular plant is used to locate plant centers. In [92], a mixture of Gaussian color model of plants is estimated by using expectation maximization. The model is used to segment the image into plants and background. A connected component analysis is used, and each plant is identified by their component mask. In [93], the skeleton of the plant is computed by using the segmentation mask of the plant. The plant center is estimated as the center of the most frequently used path segment when connecting all endpoint pairs. In [94], lines are drawn along the orientation of the leaves. The plant center is estimated as the closest point to all orientation lines. The second type of data contains images obtained from UAVs flying over a crop field. In [95], given the total number of plants, the plant centers are searched iteratively by minimizing a cost function. The cost function takes into account the alignment of the plants in the field. In [96], traditional connected component [97] is used for detecting corn plants. In [98], Ghosal *et al.* uses Resnet with Feature Pyramid Network [99] for sorghum head detection and counting. In [100], Vit *et al.* detect keypoints for each leaf using Mask R-CNN. The detected keypoints are used to compute the leaf length and width.

Related works can also be found in crowd counting where objects cluster and overlap. In [101], Liu *et al.* propose Recurrent Attentive Zooming Network for crowd density estimation. The network detects ambiguous density areas and reprocesses them with higher image resolution. In [102], Shi *et al.* incorporate the correlation between training samples into training by adding more samples from the training set as support images. In [103], Zhao *et al.* fuse the density map L2 loss with crowd semantic segmentation loss, depth prediction loss, and crowd counting regression loss. In [104], Wang *et al.* generate synthetic crowd counting dataset from computer games.

4.4 Finding Plant Location Using Multiple Instance Learning

4.4.1 Motivation

A stem represents the center of a plant, but in overhead images, stems are not visible. Therefore, the center of plants can not be detected directly. To indirectly detect the plant center, I use clues from plant leaves. Sorghum leaves grow around the stem. I define a plant center to be the intersection of sorghum leaves. By utilizing features that capture the orientation of leaves, a plant center can be detected. I use Gabor features, because Gabor features have been widely used in many recognition applications due to its spatial locality and orientation selectivity [105–108].

Since a center is defined indirectly through leaves, the labels for the plant center may not be precise. Leaves are wider than lines, which make the intersection of leaves to be a region instead of a single point. A labeled plant center can be anywhere inside this region. Multiple Instance Learning (MIL) [109,110] is a training framework that can be trained by classifying regions rather than individual locations. I use MIL to train a classifier that labels each pixel as “plant center” or “non-plant center”.

In this section, I describe some initial work in estimating plant centers from UAV images for mature sorghum images. This work makes use of MIL with Gabor features to detect plant centers.

4.4.2 Multiple Instance Learning

In this section, I describe how I train a classifier using MIL to determine if a location is a plant center. Many studies have successfully used MIL with a boosting framework [111] to solve object detection problems [109,112–115]. I implement the MILBoosting presented in [109].

In MIL, each sample is considered an instance. All the instances are put into bags. A bag must contain at least one instance and an instance may exist in multiple bags. I define the label of each bag based on the following criteria. A bag is considered positive

if at least one of the instance in the bag is positive. A bag is considered negative if all of the instances in the bag are negative. Unlike in a common classification problem, the label for each instance can tolerate some error. As long as the label of the bag is correct, the classifier can be properly trained. Therefore, I can train the classifier with an approximate label of the plant location.

For a candidate plant location $P = (P_x, P_y)$ in image, I train classifier C to determine if location P is the center of the plant. The pixel value at location P may not be sufficient to discriminate between “plant center” or “non-plant center”. I design the classifier C to consider the surrounding leaf structures of the plant for classification. Let $W(P)$ be the gray scale image that corresponds to a square window of size w centered at P (Figure 4.13(a)). The classifier is defined as:

$$C(W(P)) = \begin{cases} 1 & P \text{ is plant's center} \\ 0 & P \text{ not plant's center.} \end{cases} \quad (4.3)$$

Let $W(P)$ be an instance. I denote the i -th bag as x_i , and the j -th instance of the i -th bag as x_{ij} . Each bag x_i has N_i instances. I assign y_i to be the label of i -th bag, and assign y_{ij} to be the label of j -th instance in i -th bag.

I train a strong classifier from K weak classifiers. Let $c_k(x_{ij})$ be the classification result of the k -th weak classifier on instance x_{ij} . Then, I define the strong classifier as a linear combination of K weak classifiers:

$$C(x_{ij}) = \sum_{k=1}^K \lambda_k c_k(x_{ij}). \quad (4.4)$$

The label y_{ij} for instance x_{ij} is obtained as

$$y_{ij} = \begin{cases} 1 & \text{if } C(x_{ij}) > 0 \\ 0 & \text{otherwise.} \end{cases} \quad (4.5)$$

Based on the definition of a bag, the label y_i for bag x_i is:

$$y_i = \max_j y_{ij} \quad (4.6)$$

To train the classifier, I model the probabilities that the instances are positive and use them to compute the probability for each bag being positive. Then I compute the likelihood function for all bags. Finally, I construct the strong classifier finding the best K weak classifier that maximize the likelihood function iteratively. Sigmoid function is often used for logistic regression in binary classification problems. I model the probability p_{ij} of the j -th instance in the i -th bag as a sigmoid function of its classification result $C(x_{ij})$.

$$p_{ij} = \sigma(C(x_{ij})) = \frac{1}{1 + e^{-C(x_{ij})}} \quad (4.7)$$

Once I estimate the instance probabilities, I use the noisy-or (NOR) boosting frameworks presented in [109, 113] to compute the bag probability. The probability of a bag being positive is

$$p_i = 1 - \prod_j (1 - p_{ij}). \quad (4.8)$$

After the probabilities of all I bags are computed, I compute the likelihood L of all bags being correctly classified by classifier C as

$$L(C) = \prod_{i=1}^I p_i^{y_i} (1 - p_i)^{1-y_i}, \quad (4.9)$$

and

$$\log(L(C)) = \sum_i (y_i \log p_i + (1 - y_i) \log(1 - p_i)). \quad (4.10)$$

As in the Milboost presented in [109], the weight w_{ij} of each sample is computed using the derivative of the log likelihood function:

$$w_{ij} = \frac{\partial \log(L)}{\partial C(x_{ij})} = \frac{y_i - p_i}{p_i} p_{ij} \quad (4.11)$$

k -th weak classifier is selected to maximize the sum score of all instances $\sum_{i,j} c_k(x_{ij}) w_{ij}$. Then, I use exhaustive search to determine the weight λ_k of each classifier over the range $[0,1]$ that maximizes the updated log likelihood with classifier $C + \lambda_k c_k$. Algorithm 1 shows the pseudocode to train the classifier.

Procedure 1: Training Classifier

Input : Data set: $\{X_1, X_2..X_N\}$, where $X_i = \{x_{i1}, x_{i2}, ...x_{iN_i}\}$,
Label set: $\{y_1, y_2..y_N\}$

- 1 Initialize $w_{ij} \leftarrow 1$ for all positive instances
- 2 Initialize $w_{ij} \leftarrow -1$ for all negative instances
- 3 **for** $k = 1$ *to* K **do**
 - 4 $c_k \leftarrow \underset{c}{\operatorname{argmax}} \sum_{i,j} c(x_{ij})w_{ij}$
 - 5 $\lambda_k \leftarrow \underset{\lambda}{\operatorname{argmax}} \log L \left(\sum_{l=1}^{k-1} \lambda_l c_l + \lambda c_k \right)$
 - 6 **for** *all* i **do**
 - 7 **for** *all* j **do**
 - 8 $p_{ij} \leftarrow \sigma \left(\sum_{l=1}^k \lambda_l c_l(x_{ij}) \right)$
 - 9 **end**
 - 10 $p_i \leftarrow 1 - \prod_j (1 - p_{ij})$
 - 11 **end**
 - 12 Update $w_{ij} \leftarrow \frac{y_i - p_i}{p_i} p_{ij}$ for all i, j
- 13 **end**
- 14 $C \leftarrow \sum_{k=1}^K \lambda_k c_k$

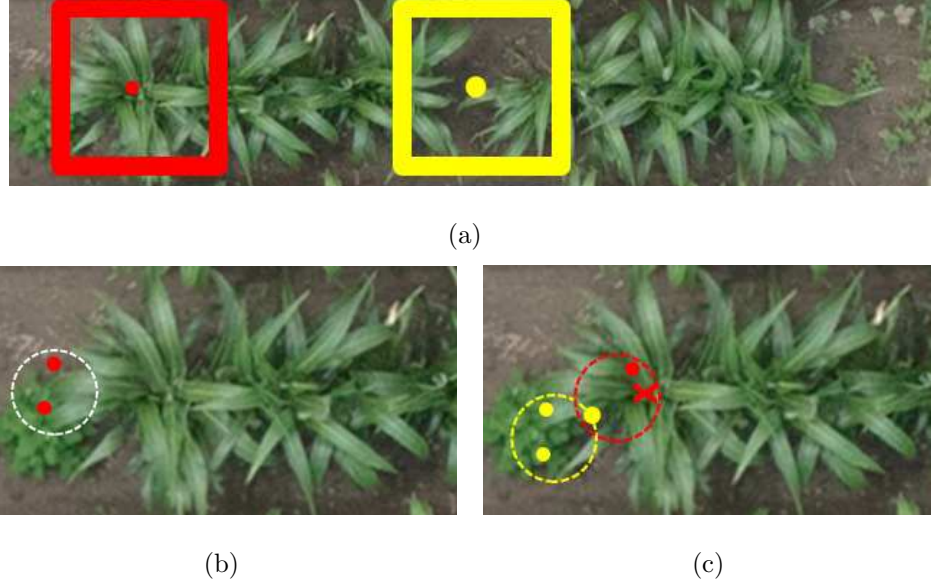


Fig. 4.13. a) A row of plants with positive sample (red dot) and negative sample (yellow dot). The window for classification is marked as a square for each sample. b) A bag (white circle) contains 2 instance (red dot). c) A positive bag (red circle) contains a positive instance (red dot) and a negative instance (yellow dot), and a negative bag contains 3 negative instances. Ground truth location is marked as a red "X".

4.4.3 Features

I use Gabor features [116]. Gabor features have been widely used in many recognition applications because of its spatial locality and orientation selectivity [105–108]. Leaves exhibit thin triangle-like shapes with different orientations that Gabor features will be able to capture while accounting for the illumination changes [105].

I denote the value of the n -th feature computed on instance x_{ij} as $v_{n,ij}$. Gabor wavelets are defined as

$$f(x, y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}} e^{-j\omega(x \cos \theta + y \sin \theta)}, \quad (4.12)$$

where (x, y) are the 2D image coordinates, σ is the variance of the Gaussian, ω is the oscillation frequency, and θ is the orientation angle. Gabor wavelets with orientations

$\theta = \{0, \frac{\pi}{8}, \frac{\pi}{4}, \frac{3}{8}\pi, \frac{\pi}{2}, \frac{5}{8}\pi, \frac{3}{4}\pi, \frac{7}{8}\pi\}$ are generated. The wavelets are also generated with frequencies $\omega = \{\frac{\pi}{\sqrt{2}}, \frac{\pi}{2}, \frac{\pi}{2\sqrt{2}}, \frac{\pi}{4}, \frac{\pi}{4\sqrt{2}}\}$.

In order to account for how leaves relate to the plant center, I divide the classification window into 9 equally spaced sections as shown in Figure 4.14(a) and Figure 4.14(b). In a top-view sorghum image, leaves spread out from the center. I assume leaves in each section follow a specific orientation as shown in Figure 4.14(c). Section 1, 3, 7, and 9 have diagonal leaves. Section 2 and 8 have vertical leaves. Section 4 and 6 have horizontal leaves. Section 5 may have horizontal, vertical and diagonal leaves. Any leaf that does not follow this assumption is from a neighboring plant. With this design, cases where the classifier incorrectly classifies a neighbor plant can be avoided. Table 4.1 shows the orientations of Gabor wavelets I used and were assigned to each window within the image. Gabor features are computed by convolving the window section image with a Gabor wavelet. The result of the convolution at each pixel represents a Gabor feature. I choose these wavelets based on the work in [106, 107]. The variance σ of the Gabor wavelets is selected to be $\frac{w_i}{2}$ to account for small window section size.

4.4.4 Weak Classifier

I use a naive Bayes classifier to replace the Sign classifier used in [109]. The Sign classifier maps the feature values to two discrete values, 1 for a positive result and value 0 for a negative result. I am unable to determine if a classifier with one feature is better than the classifier with another feature when the classification results are both positive. A Sign classifier is selected based on the number of instances it correctly classified. In my application, the data labels can be wrong. Thus, the quantity of correct classification is not a reliable measurement for the performance of a classifier. I need a classifier that can estimate how strong an instance is classified to be positive

Table 4.1.
Orientations of the Gabor wavelets

Section	Orientations	Features / pixel
1, 9	$\frac{5}{8}\pi, \frac{3}{4}\pi, \frac{7}{8}\pi$	15
2, 8	$\frac{7}{8}\pi, 0, \frac{\pi}{8}$	15
3, 7	$\frac{\pi}{8}, \frac{\pi}{4}, \frac{3}{8}\pi$	15
4, 6	$\frac{3}{8}\pi, \frac{\pi}{2}, \frac{5}{8}\pi$	15
5	$0, \frac{\pi}{8}, \frac{\pi}{4}, \frac{3}{8}\pi, \frac{\pi}{2}, \frac{5}{8}\pi, \frac{3}{4}\pi, \frac{7}{8}\pi$	40

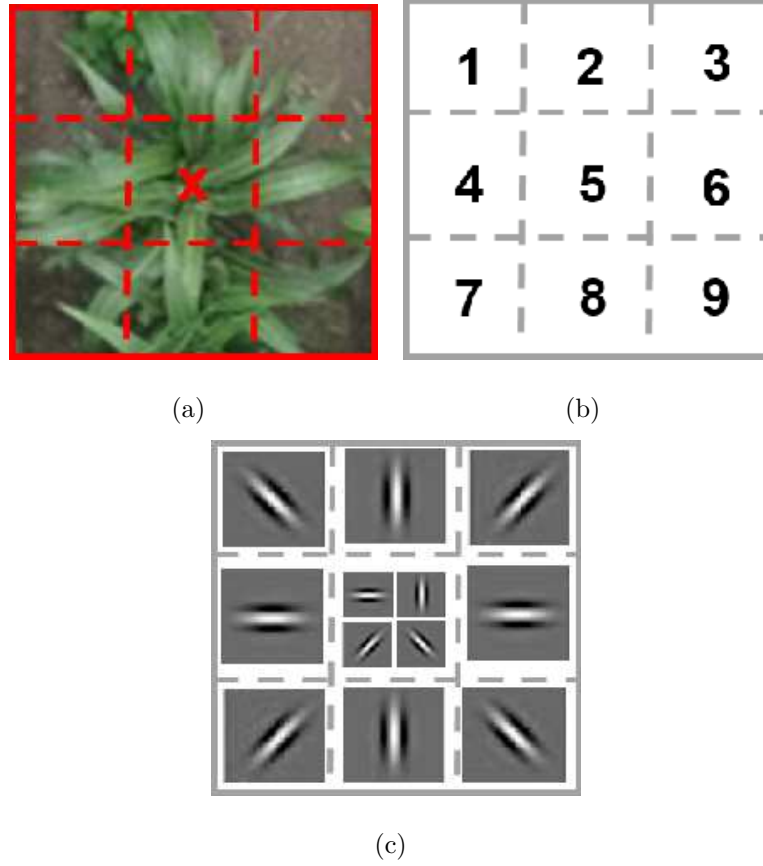


Fig. 4.14. a) The lines that divide the classification window. b) Section numbers of the classification window. c) Orientations of the Gabor feature in each section.

or negative. Naive Bayes classifier is a better choice. A Naive Bayes classifier that uses the n -th feature is given by:

$$c^{(n)}(x_{ij}) = \log \frac{p(y_{ij} = 1|v_{n,ij})}{p(y_{ij} = 0|v_{n,ij})}. \quad (4.13)$$

By using Bayes rule, the above equation becomes

$$c^{(n)}(x_{ij}) = \log \frac{p(v_{n,ij}|y_{ij} = 1)p(y_{ij} = 1)}{p(v_{n,ij}|y_{ij} = 0)p(y_{ij} = 0)}. \quad (4.14)$$

I let $p(y_{ij} = 1) = p(y_{ij} = 0)$, which means I have equal amount of positive instances and negative instances. I model $p(v_{n,ij}|y_{ij} = 1)$ to be Gaussian with mean μ_1 and

variance σ_1 , and $p(v_{n,ij}|y_{ij} = 0)$ to be Gaussian with mean μ_0 and variance σ_0 . The means μ_0 , μ_1 , and the variances σ_0 , σ_1 can be computed as

$$\mu_1 = \frac{1}{N_t} \sum_{i,j} v_{n,ij} y_{ij}, \quad (4.15)$$

$$\mu_0 = \frac{1}{N_t} \sum_{i,j} v_{n,ij} (1 - y_{ij}), \quad (4.16)$$

$$\sigma_1 = \sqrt{\frac{1}{N_t} \sum_{i,j} y_{ij} (v_{n,ij} - \mu_1)^2}, \quad (4.17)$$

$$\sigma_0 = \sqrt{\frac{1}{N_t} \sum_{i,j} (1 - y_{ij}) (v_{n,ij} - \mu_0)^2}, \quad (4.18)$$

where N_t is the total number of instances.

4.4.5 Post-Processing

After the locations in an image are classified, I obtain a map M with classification labels as shown in Figure 4.15(c) and Figure 4.15(d), where $M(P_x, P_y) \in \{0, 1\}$. Because my classifier is trained to classify areas and specific location is not given, the classification label map M shows the clusters of possible plant center location.

I further process the label map M to estimate the plant center. I assume small clusters in M are noise. The connected component analysis [97] is used to the label map M to obtain individual clusters. Small components are removed by thresholding the number of pixels in the components with threshold τ_a . Let the horizontal direction be denoted as \mathcal{X} , and let the vertical direction be denoted as \mathcal{Y} , as shown in 4.15(a). I assume all the plants in a subrow are somewhat aligned in the \mathcal{X} direction, so I only consider results that are within a distance τ_b to the \mathcal{Y} mid line (see Figure 4.15(b)). Some candidate centers may come from the same plant but may be in different components. I merge components whose centroids are within a distance of τ_c of each other in \mathcal{X} direction. After the merging, the centroid of each component is considered as the center of plants for my results.

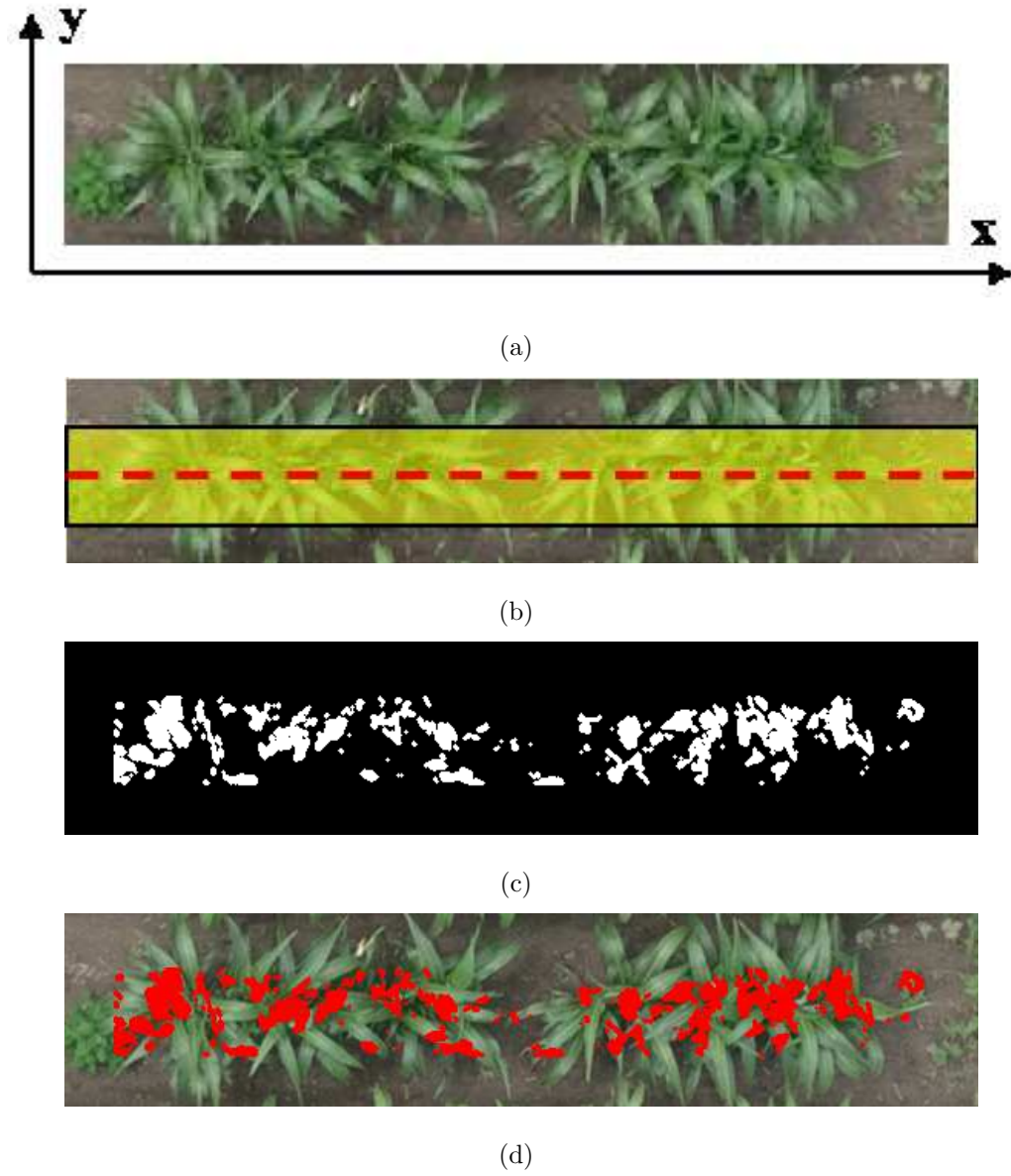


Fig. 4.15. a) Original subrow with coordinates system. b) Plant location region (yellow region) and the \mathcal{Y} direction mid line (red dash line). c) Classification label map M (positive: white, negative: black). d) Classification label marked in red in original image.



Fig. 4.16. Image of a section of a sorghum field acquired from a UAV at an altitude of 40m. The pixel resolution is 0.59 cm/pixel.

4.4.6 Experiment and Results

My dataset was collected from a UAV on July 19th, 2016 in field 54 middle calibration panel. The images were taken at an altitude of 40 meters at UAV velocity of 8m/s. Figure 4.16 shows an example image in my dataset. I cropped 26 subrows from an original image, and converted them into grayscale images. I used 6 subrows for training, and 20 subrows for testing. The cropped subrows were selected to have low lens and perspective distortion.

I ground truthed the center locations for each plant. Each ground truthed location corresponds to one positive bag. I labeled the locations around each ground truth as positive if the location is within 5 pixels of the ground truth. Those positive instances are put into the same bag as the ground truthed location. The locations that are at least 30 pixels away from any of the ground truth are considered as negative instances, and each negative instance is put into one bag. Positive instances and negative instances are randomly selected to reduce the computation cost. I used 800 positive instances and 800 negative instances in the 6 subrows for training. I tested my method with 10 high leaf density subrows and 10 low leaf density subrows.

I set the classification window width w to be the average plant diameter, 99 pixels. The number of weak classifiers K is set to be 10. The threshold τ_a to remove small components is set to be 100 pixels empirically. The threshold τ_b is set to be $\frac{1}{3}w = 33$. The merge distance τ_c is set to be 30 pixels. To reduce the computational cost, I used locations in the plants segmentation mask as candidate locations for classification, and the mask is obtained by applying the color threshold method described in [95].

Precision and recall was used to evaluate my method [117,118]. Precision is defined as

$$\text{Precision} = \frac{TP}{TP + FP}, \quad (4.19)$$

and recall is defined as

$$\text{Recall} = \frac{TP}{TP + FN} \quad (4.20)$$



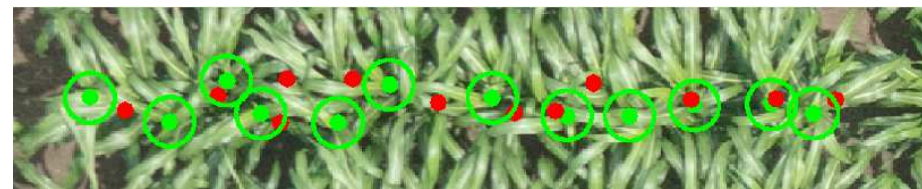
(a)



(b)



(c)



(d)

Fig. 4.17. a) A subrow with low leaf density. b) The center of the red dots are the detected plant centers, the center of the green dots are the ground truth. If a detected plant falls inside the green circumference, I consider it a true positive. c) A subrow with high leaf density. d) Detected centers (red dots) of high leaf density plants.

If a center is detected within 30 pixels of the ground truth, the center will be considered as a true positive (TP). If more than one center is detected within 30 pixels of any ground truth, the extra detection will be counted as false positive (FP). Note that this distance is the same as the distance I use to generate negative samples during

Table 4.2.

Results for plants with different leaf density after training and evaluating 10 times. My dataset contains 121 high density plants and 110 low density plants.

Type	TP	FP	FN	Precision	Recall
High Density	779	506	431	61%	64%
Low Density	701	242	399	74%	64%
Overall	1480	748	830	66%	64%

training. False negatives (FN) occur when no center is detected for a ground truth location. As the instances are randomly selected during training, the precision and recall may vary when evaluated. I report the results after training and evaluating 10 times. Table 5.1 shows the results for high leaf density data and low leaf density data. My result obtained an average overall precision of 66%, and a recall of 64%. Figure 5.16 shows an example output for a low leaf density subrow, and an example output for a high leaf density subrow.

The results show that the the method produces more false positive with high leaf density data. This is due to the fact that the classifier is trained to look for the structures of surrounding leaves. When plants have small amount of leaves, the structures around the plant centers are clear. Leaves from the same plant point toward the plant center. If a pixel location is not a plant center, then the leaves around it will converge to different location. When plants have high leaf density, there is more leaf overlapping. The overlapping makes the leaf structures more complex, and confuses the classifier. As a result, there are more false positives.

The performance of my method can be interpreted as follows. In my application, a plant consists of an indefinite number of leaves. Each leaf has a unique shape including length, width, and leaf curvature. Plants have different number of leaves in various orientations. A combination of simple features may not be enough for

describing heterogeneous sorghum plants. In addition, my negative training data may have false labels due to human error during labeling. Plant centers can easily be missed during labeling, and may accidentally be used as negative training data.

4.5 Finding Plant Location Using Deep Binary Classifier

4.5.1 Motivation

In my previous study, I describe a method that uses MIL to address the ambiguity in plant location labels. The method uses Gabor wavelets to capture the orientation of leaves. However, the overlapping between leaves makes the leaf structures more complex. Gabor wavelets are not designed to capture overlapping leaf structures. Deep learning is a powerful tool for image classification. It has a large capacity to capture various class features including overlapping leaf structures. In this section, I describe a method that uses CNN to detect plant centers.

4.5.2 Proposed Method

I adopt the detection strategy presented in [77, 85, 119]. Each pixel location in the image is classified as plant center or not plant center by considering a rectangular window around the pixel. Figure 4.18 shows the block diagram of the overall method. Plants in the same dataset have similar size because they were planted on the same day under similar environmental conditions. Therefore, I fix the size of the rectangular windows. I generate image patches by extracting rectangular windows centered at all possible pixel locations (x, y) in the image. I then classify each window in the patch with a CNN, and obtain a classification mask, as shown in Figure 4.19. I denote the classification mask as for the coordinate (x, y) as

$$M(x, y) = \begin{cases} 1 & \text{pixel } (x, y) \text{ is a plant center} \\ 0 & \text{otherwise.} \end{cases} \quad (4.21)$$

Ideally, $M(x, y)$ would be 1 only when there is a true plant center at (x, y) . However, I experimentally observed in the classification mask that there is usually more than one positive response for each plant center. The center of a plant is defined as the location where all its leaves intersect. Unfortunately, in my dataset the leaves of a plant may be partially or completely occluded by the leaves of neighboring plants.

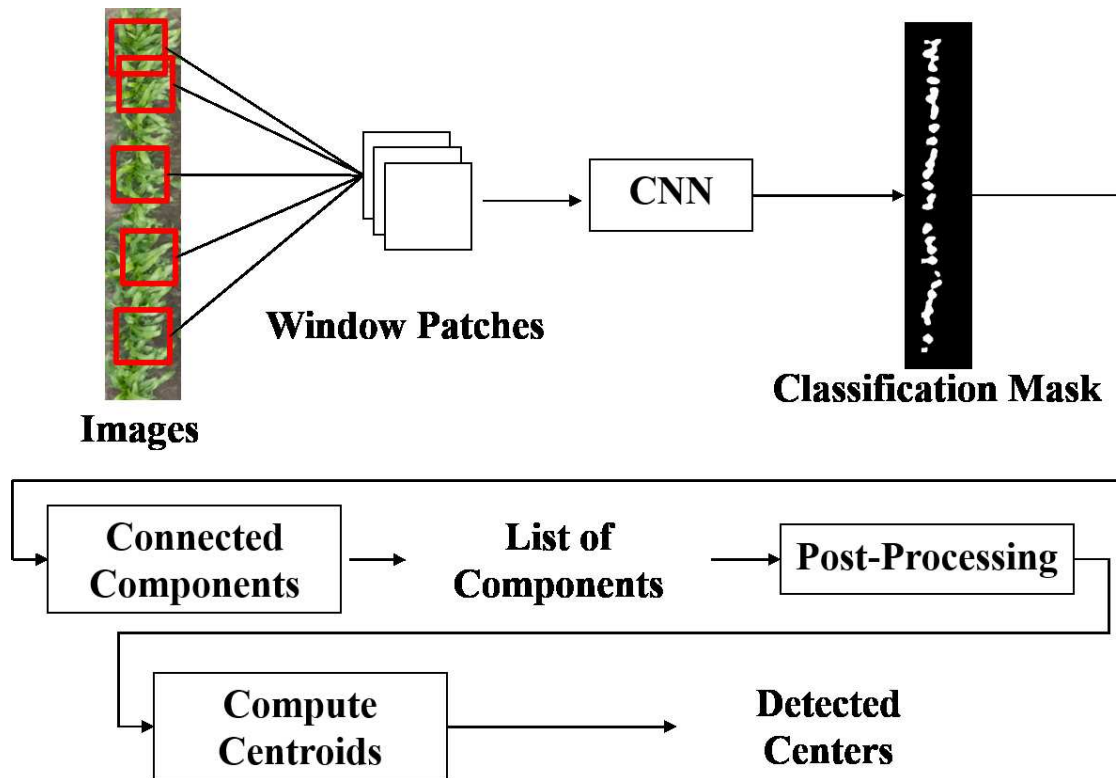


Fig. 4.18. Block diagram.



Fig. 4.19. Example of a classification output mask. Red locations are classified as plant centers.

In addition, the edges of the leaves may be blurred, making leaves hard to recognize. This can be observed in Figure 4.19. This makes the groundtruthing of plant centers inaccurate and inconsistent because plant centers can be labeled anywhere in a region where leaves intersect. I observed that these regions have a shape similar to an ellipse.

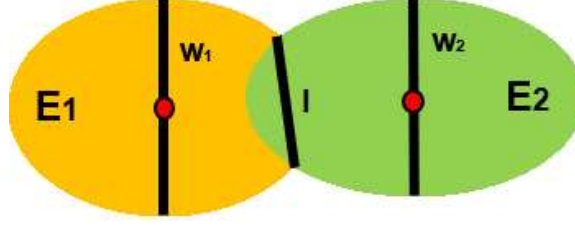


Fig. 4.20. Connected components E_1 and E_2 merge together. Red dot is the center of each component. W_1 and W_2 are the minor axes of ellipse for E_1 and E_2 respectively. l is the maximum length of the intersection between two components.

Once the classification mask $M(x, y)$ has been generated, I post-process it to locate the center of the plants. I apply connected components [97] to the mask $M(x, y)$ and obtain a list of components. For illustration purposes, assume that the generated connected components have an ellipse shape, and that the center of the plant is at the center of the ellipse. When neighboring plants are close, the ellipses merge together as shown in Figure 4.20. I consider two different cases. In the first case, the two components with an ellipse shape, E_1, E_2 , have a small overlapping area. Let the minor axes of E_1 and E_2 be W_1 and W_2 , respectively. In this case, the longest length of the overlapping area, l , satisfies

$$l < W_1 \text{ and } l < W_2. \quad (4.22)$$

Figure 4.20 illustrates this first case. I can apply morphological erosion [120] until the intersection between the two ellipses disappears and none of the two components will disappear. Therefore, in this case this component can be split into two different components. In the second case, the two ellipses have a large overlapping area because they are very close. In this case, successive applications of erosion will not be able to split the merged component into two, so I consider them as the same component.

I use erosion to reduce the size of the components and find the center of each plant, as follows. I initialize a list with all the connected components in $M(x, y)$. Then, I apply morphological erosion to each component until it splits in two or until further

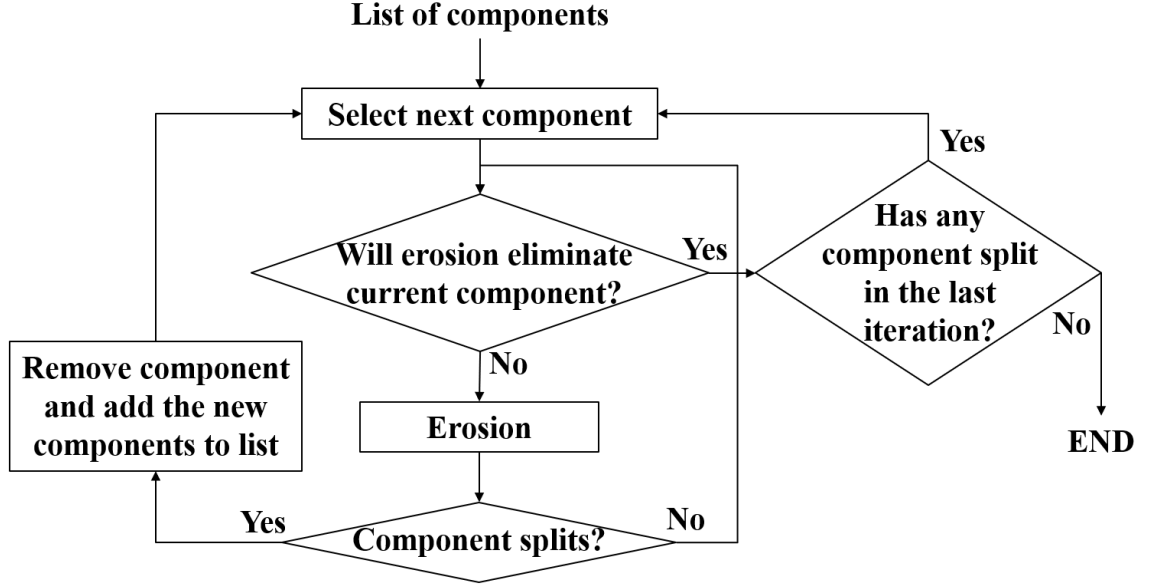


Fig. 4.21. Block diagram of post-processing the plant center clusters.

erosion would completely eliminate the component. When a component splits, I add the new components to the list. I terminate when all components in the list are small enough that applying an additional erosion would remove at least one component. I finally estimate the plant centers by computing the centroid coordinate of each component. Figure 4.21 shows the block-diagram of the post-processing.

4.5.3 Experimental Results

I evaluated my method using two datasets with plants at two different growth stages. Both datasets were acquired by a UAV platform at an altitude of 40m in a sorghum field in West Lafayette, Indiana, USA. The first dataset was collected on June 13, 2016 in field 54 middle calibration panel, and the second dataset was collected on June 28, 2017 in field 41 calibration panel. The plants in the first dataset are young and the level of overlapping of leaves between neighboring plants is low. The plants in the second dataset are more mature, and the overlap of leaves

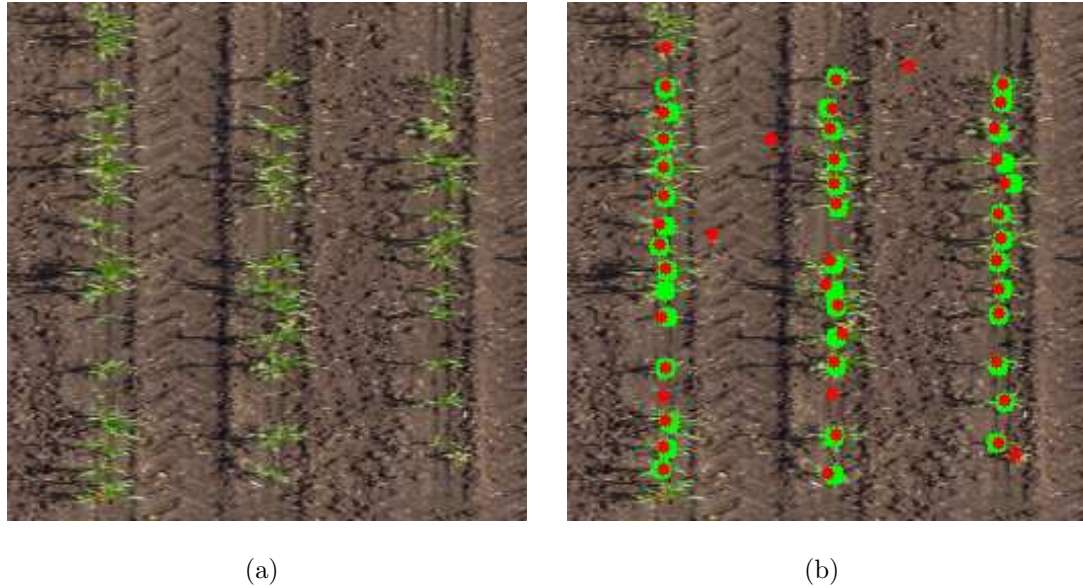


Fig. 4.22. a) An image sample from the young plant dataset. b) Estimated plant centers and ground-truth. The ground-truthed plant centers are at the center of the green regions of radius $r = 5$. Red dots indicate predicted plant centers. Red dots inside a green region are counted as true positives (TP).

between neighboring plants is high. I manually label the center of each plants in both datasets. The first dataset contains 50 images with a total of 2938 plants, and the second dataset contains 90 images with a total of 2001 plants. Each plant center represents a positive sample, and all pixel locations that are not a plant center are negative samples. This results in the datasets containing more negative samples than positive samples. During training, I also consider as positive samples the pixels that are at a distance of 1 or less pixels to the true plant center. This increases the number of positive samples by 5. To balance the number of positive and negative samples, I randomly select as many negative samples as positive samples.

The size of each classification window is set to be approximately the average plant size in each dataset. In the dataset with young plants, the window size is set to be 40×40 pixels. In the dataset with mature plants, the window size is set to be 60×60

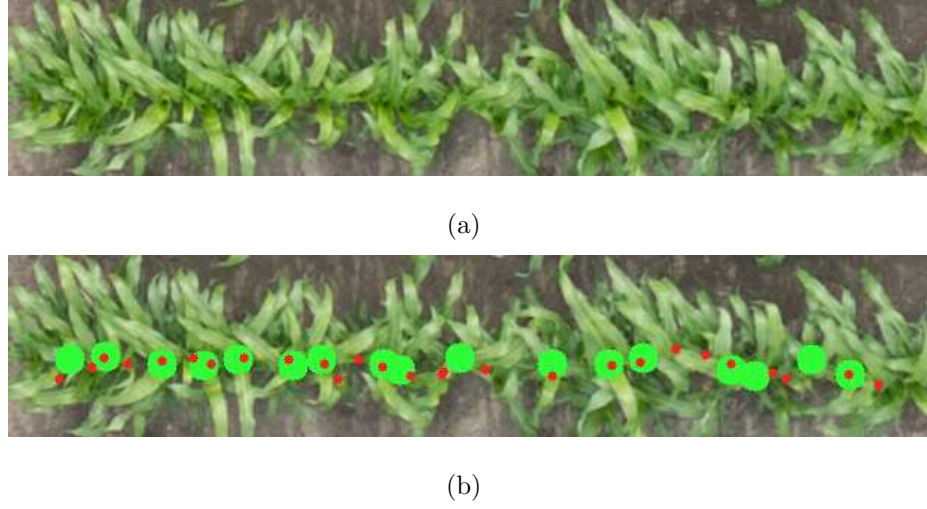


Fig. 4.23. a) An image sample from the mature plant dataset. b) Results of the image in Figure 4.23(a).

pixels. I divide the datasets so that 50% of the images are used for training, and 50% of the images are used for testing. The CNN I use for classification is VGG [68] since it is the basic deep learning architecture. I also tried Resnet [71] architecture, but there is no improvement. The input images are resized to 224×224 before being fed to the CNN.

An estimated plant center is considered as true positive (TP) if it is within r pixels of a ground-truth location. An estimated plant center is considered as false positive (FP) if it is not within r pixels of a ground-truth location. A ground-truthed plant center is considered as false negative (FN) if it is not within r pixels of an estimated plant center. I use precision and recall to evaluate my method [117,118]. Precision is defined as

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad (4.23)$$

and recall is defined as

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (4.24)$$

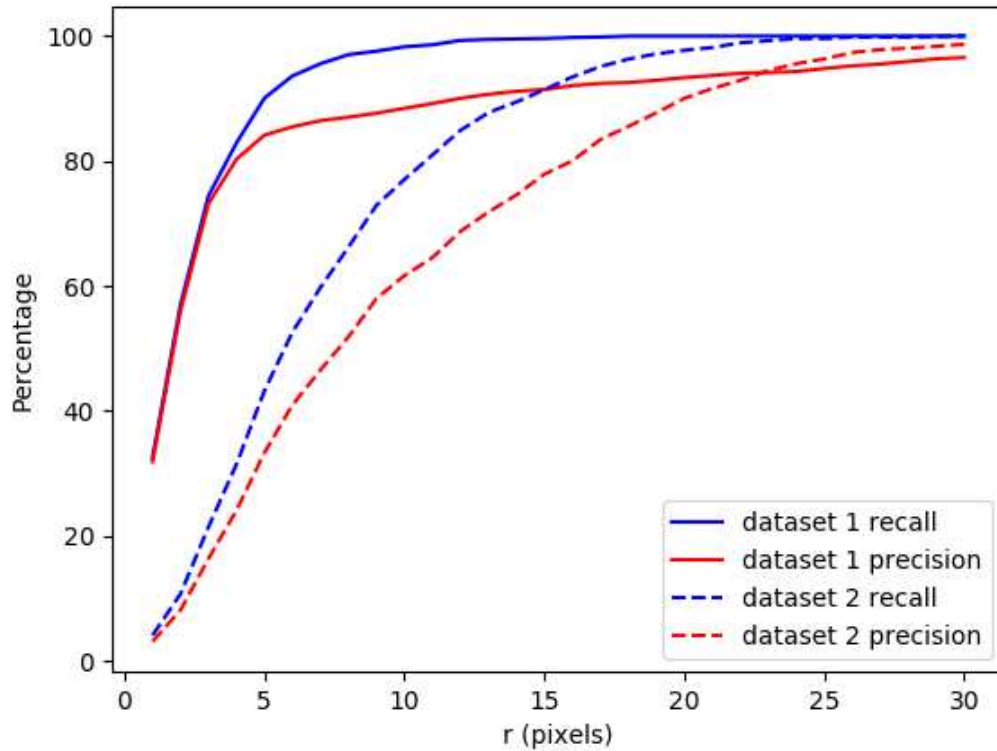
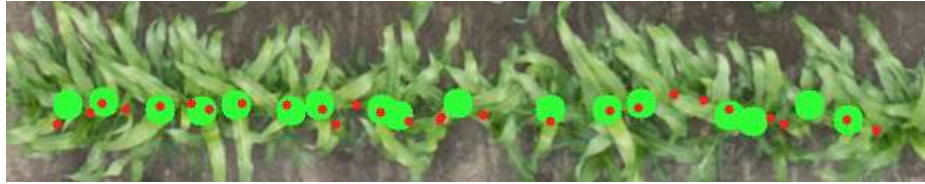


Fig. 4.24. Precision and recall for different r values on the two datasets.

Figure 4.22 and 4.23 show examples of estimated and ground-truthed plant centers in one image of each dataset. In the dataset with young plants, I achieve 84% precision and 90% recall with $r = 5$. In the dataset with mature plants, I achieve 62% precision and 77% recall with $r = 10$. Figure 4.24 shows precision and recall as a function of r for both datasets. In the plot I can see that recall is always higher than precision. This means that I have more false positives than false negatives. This may be because the human selecting ground truth misses some true plant centers due to very high occlusion, creating a bias in the labels. I can also observe in Figure 4.24 that both precision and recall are lower in the second dataset. The CNN does not use the location of neighboring plants to estimate one plant location, while the human

Table 4.3.
Results compared to Multiple Instance Learning

Metric	Multiple Instance Learning	Proposed Method
Precision (higher is better)	99%	62%
Recall (higher is better)	31%	77%



(a)



(b)

Fig. 4.25. a) Results of the proposed method. b) Results of the Multiple Instance Learning method.

performing the labeling often uses this information. This is particularly true for the second dataset, where plants have much more overlap.

In addition, I compare this method with Multiple Instance Learning method described in Section 4.4 on the mature dataset. Table 4.3 gives a detailed comparison of the proposed method and Multiple Instance Learning method. Figure 4.25 shows an example result from both methods. From the results, I can see that MIL method produces less false positive but miss-detects a lot of plants. The proposed method is able to capture more plants but produces a lot of false positives.

5. LEAF SEGMENTATION

5.1 Leaf Segmentation Introduction

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. We extract plant leaves throughout the season to study the growth of plants. In the early season, plants have less overlap, but the leaves are pointing towards the sky, which makes them difficult to observe in overhead images. We do not study these images due to the limitation of overhead images. In late season, plants are grown densely, which causes frequent overlapping of leaves as the canopy develops. 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 addition to the leaf overlapping issue, UAV images often have low spatial resolution as described in Section 1.3. Therefore, the problem I am addressing is instance segmentation for densely overlapped leaves in low resolution UAV images.

5.2 Related Works

In previous studies, leaf segmentation has been primarily applied mostly applied to rosette plants grown in an indoor environment. In [121], the input image of a rosette plant is thresholded to obtain the foreground mask. The contour of the foreground object is computed. Stems of the leaves are found by comparing the distance between pixels. Since each stem connects a leaf to the plant center, by selecting the contour that attaches to the leaf end of the stem, the leaf contour is obtained. The leaf contour is then used for leaf segmentation. In [122], an open-sourced software is used to classify pixels of a rosette plant image into leaf border pixels and inner leaf

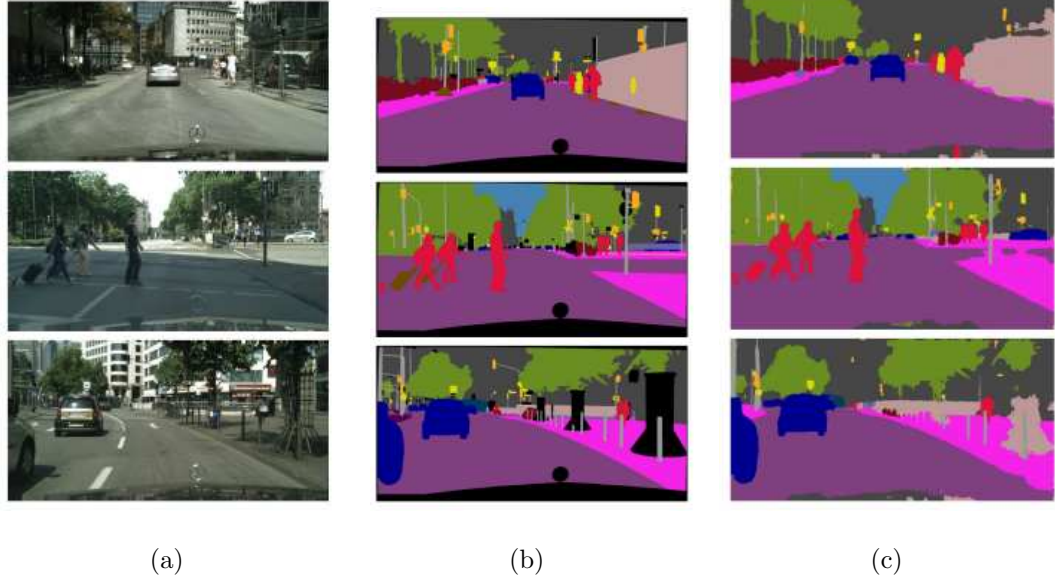


Fig. 5.1. Semantic segmentation example: a) input image. b) ground truth. c) results.

pixels. A distance transform is applied to the inner leaf pixels, and the peaks in the distance transform map are searched. Leaf is labeled by applying flood-filling on the detected peaks. In [29], the input image of a rosette plant image is first segmented into foreground and background using watershed-based segmentation. A distance transform is applied to the foreground object. Local maxima points in the distance transform map are used as a seed for another iteration of watershed-based segmentation, and the results are the individually segmented leaves.

Recent work using Deep Neural Networks (DNN) has produced very promising results for image segmentation [81,123,124]. Deep learning based image segmentation can be categorized into two: semantic segmentation and instance segmentation.

Semantic Segmentation: In semantic segmentation, every pixel of a given image is assigned to a class, and thus it creates a segmentation mask for each class object. Figure 5.1 shows an example of semantic segmentation. In [124], Chen *et al.* change the standard convolution to Atrous Convolution which is convolution with upsam-

pled filters. An Atrous Spatial Pyramid Pooling is proposed to capture objects with multiple scales. Finally, they refine the segmentation results with a fully-connected Conditional Random Field (CRF). In [125], Takikawa *et al.* fuse Resnet and a shape module with Atrous Spatial Pyramid Pooling [124], and produce accurate boundaries for semantic object segmentation. In [126], Wei *et al.* proposed to replace max pooling and average pooling with Polynomial Pooling layers which provide both non-linearity and detail sensitivity. The resulting networks are able to enhance the details of semantic segmentation. Embedding approach for segmentation has also become a recent trend. Typical class labels for deep learning do not consider the similarities between classes. For example, a classification task often has cross entropy as a loss function. The class labels are encoded into one-hot representation. Let the dataset contain only three classes: cat, dog, and car. Then the cat's label is (1,0,0). Dog's label is (0,1,0). Car's label is (0,0,1). Although the cat looks more similar to the dog, the network is trained without considering their similarities. Embedding approaches try to solve this problem by projecting the classes into an embedding space where the similarity between classes is represented by a metric system. In [127], Kulikov *et al.* color an object instance with one color, and its adjacent objects with different colors. Therefore, the learned metric space produces similar color values for the same instance and different color values for different object instances. In [128], Kulikov *et al.* proposed a embedding space consisting of harmonic guide functions. In late plant growing seasons, plants are grown densely, which causes frequent leaf overlaps. As a result, the profile or edge of each leaf may not be completely visible. This creates a problem for applying semantic segmentation, where segmentation masks do not consider overlapping regions. Furthermore, semantic segmentation does not separate instances of the same class, where in my application application, being able to segment each leaf instance is essential to the estimation of leaf traits. Therefore, my work focuses on the approach taken by instance segmentation.

Instance Segmentation: In instance segmentation, a mask is produced for each object instance. Figure 5.2 shows an example of instance segmentation. In [129],

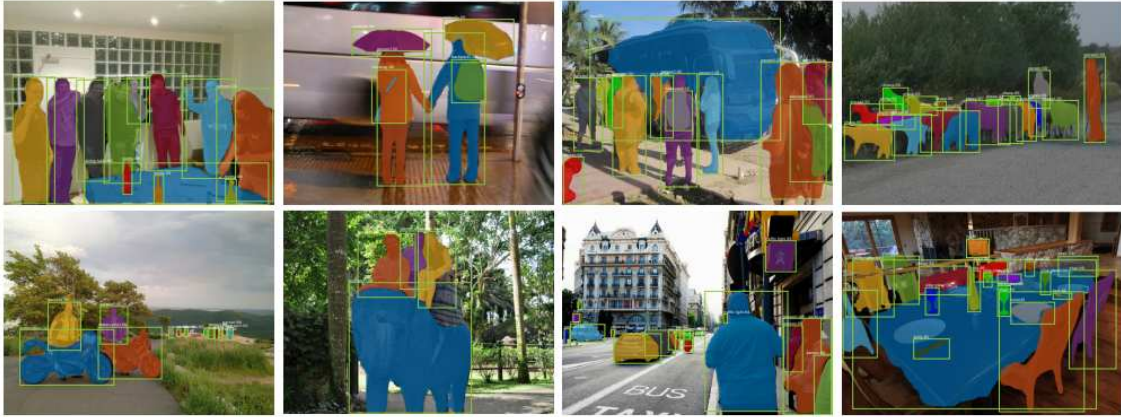


Fig. 5.2. Instance segmentation example [81]

Ahn *et al.* generate an attention map for each class and use the generated maps to divide pixels into sets based on their class labels. They use the divided sets to derive a loss function that considers the instance's centroid and boundaries. In [130], Zhu *et al.* propose a module that fills the instance extent based on the feature layers of a classification network. The extent filling output is combined with a noisy proposed region map to produce an Instance Activation Map which is further processed by a CRF to generate the instance mask. In Faster R-CNN [74], 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 [81], 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 [123], 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. In [131], 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 [132], Morris uses a Fully-Convolutional Pyramid Network to estimate tree leaf

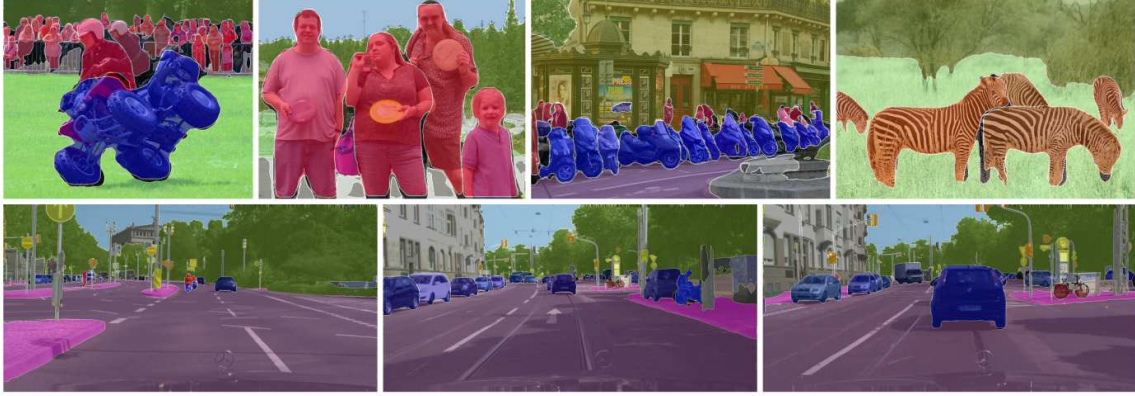


Fig. 5.3. Panoptic segmentation example [134]

boundaries. Connected Components [97] and Watershed [133] are then applied to the output edge map to group leaf boundaries and obtain leaf segments.

Panoptic Segmentation: Recently, in order to close the gap between semantic segmentation and instance segmentation, Kirillov *et al.* [135] proposed a new task called Panoptic Segmentation which unifies semantic and instance segmentation. While semantic segmentation assigns each pixel a class label and instance segmentation assigns each object an instance id, Panoptic Segmentation assigns each pixel a class label and an instance id. Figure 5.3 shows an example of Panoptic Segmentation. In [134], Kirillov *et al.* propose Panoptic Feature Pyramid Networks for this task. The network added a semantic segmentation branch to Mask R-CNN [81] networks. A Feature Pyramid Network (FPN) [99] backbone is shared with both the instance segmentation branch and the semantic segmentation branch. In [136], Xiong *et al.* take the similar approach of adding a semantic segmentation branch to Mask R-CNN. They propose a Panoptic head that merges the results from both segmentation branches to resolve conflicts. In [137], Liu *et al.* propose Spatial Ranking Module to resolve the conflicts between foreground instance segmentation and background semantic segmentation. In [138], Li *et al.* propose Proposal Attention Module and Mask Attention Module to help the network segmenting background objects.

From the overview of segmentation methods, it is clear that CNNs play an important role in image segmentation. All CNN 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 [131, 139] overcomes the lack of labeled data, but well-defined object models are necessary to create realistic images. In my 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 CNN loss functions for segmentation including binary loss [81], cross entropy loss [53], and dice loss [140] do not consider the spatial accuracy of the segmented object’s edges. In my application, where leaf shapes are crucial for measuring phenotypic traits (such as maximum width), popular segmentation CNNs are likely to produce segmented leaves with inaccurate and noisy edges because they do not have a spatial objective.

To overcome this issue, some studies borrow ideas from non-DNN approaches, including Active Contour [141] and Domain Transform [142]. Active Contour approaches find objects’ boundaries according to the shape of the boundary and the gradient of the image. Extended from Active Contour, Geodesic Active Contour [143] finds objects boundary by minimizing geodesic distances on a Level Set [144]. In [145], Acuna *et al.* combine learned features of a DNN with Geodesic Active Contour to refine the edges of ground truth, and use the refined ground truth to train a segmentation network. In [146], Chen *et al.* detect edges by adding an edge prediction network and Domain Transform Filter to Deeplab segmentation network [124].

Recently, the development of the Graph Neural Network (GNN) [147, 148] has delivered promising results for many applications. Figure 5.4 shows an example application of GNN. In [149], Wang *et al.* initialize a graph ellipsoid representation for an object and use Graph Convolutional Network (GCN) to propagate image features

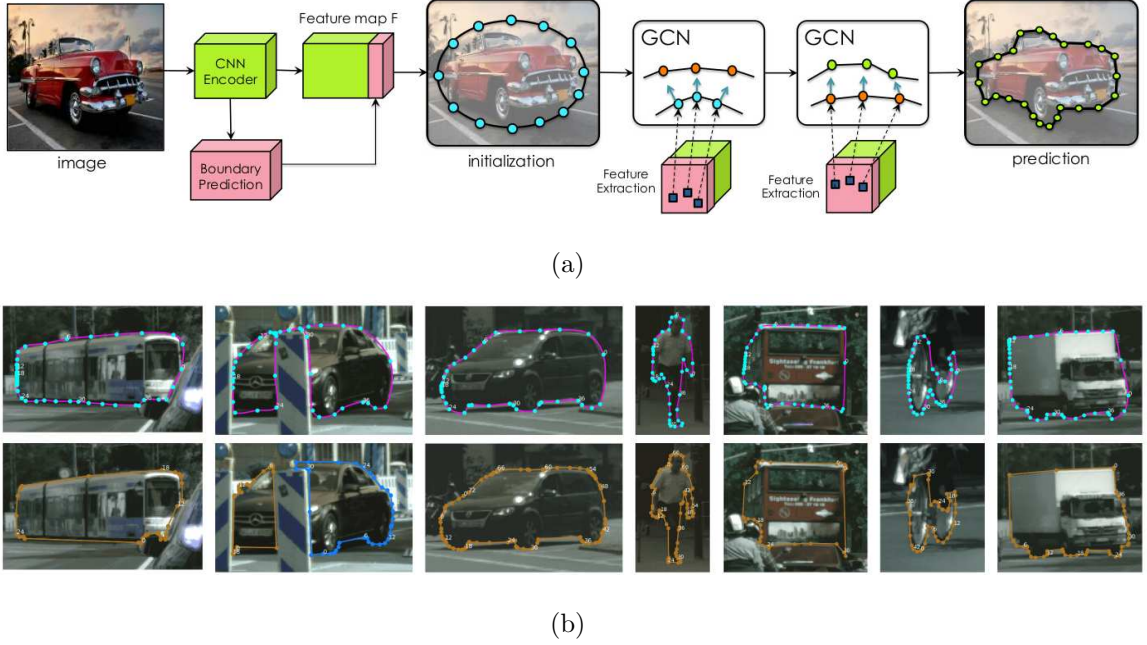


Fig. 5.4. An application for Graph Convolutional Network (GCN). a) the architecture for estimating object's boundary using GCN. b) top: results of the network, bottom: ground truth.

into vertices. The graph gradually deforms into the shape of ground truth as the network gets deeper. In [150], Ling *et al.* use a graph to represent the control points of a spline curve that describe the object boundary. They iteratively update the control points with GCN for propagating feature information. The final graph represents the boundary of the object. Unlike CNNs, GNNs give the network capabilities to capture the relationships between the features such as object boundaries. Therefore, a boundary can be reconstructed from the relationship even with partial occlusions. This is very useful for leaf segmentation where leaves are commonly occluded.

Lack of labeled data can cause model overfitting or unable to converge. Obtaining labeled data is expensive and time-consuming. On the contrary, synthetic data is inexpensive to generate but sometimes does not provide enough details. With the recent development of generative networks, data can be generated with realistic details. With additional training data, many studies have achieved promising results in seg-

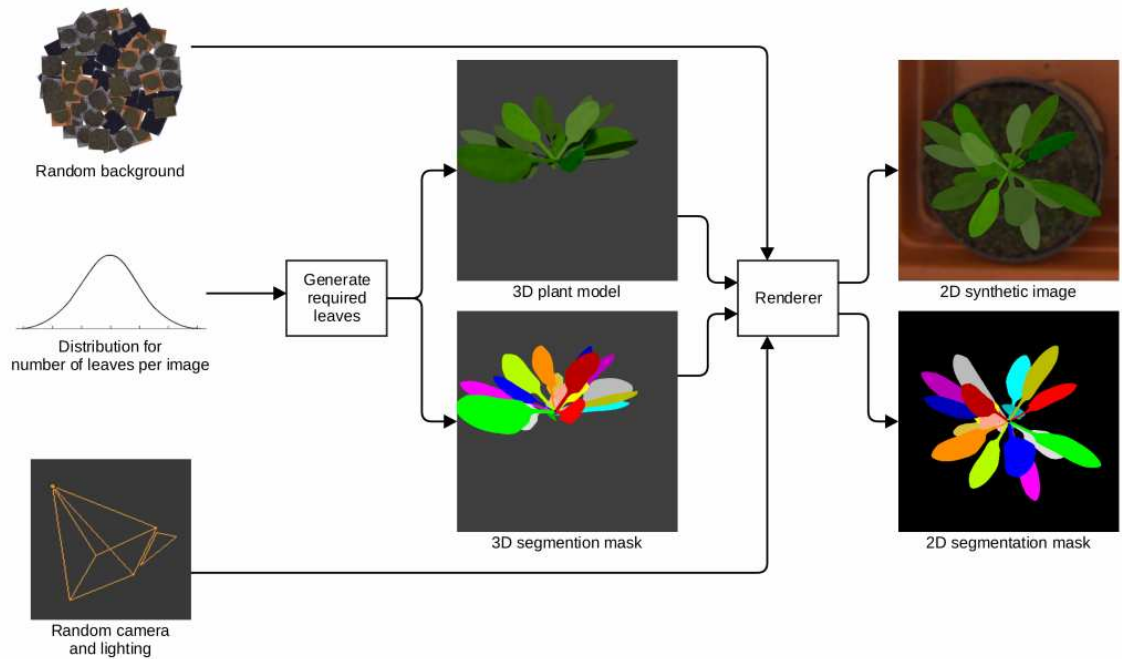


Fig. 5.5. Synthetic data generation for leaf segmentation [131]

mentation. Figure 5.5 shows an example for generating synthetic leaf segmentation data. These data augmentation techniques include the Generative Adversarial Network (GAN) [151], Variational Autoencoder (VAE) [152–154], and data simulation. GAN can be trained with paired inputs using Conditional GAN [155], or without paired inputs using CycleGAN [156]. For example, in [139], synthetic microscopy images are generated using CycleGAN. In plant phenotyping, data simulation is more commonly used to increase the quality of data for training. In [157], individual leaves are cropped and recombined into synthetic data. In [131], Synthetic leaves of rosette plants are generated with different shapes, sizes, and textures.

For simplicity and accessibility, I use labelme [158] to label segmentation masks. Figure 5.6 shows an example of the tool. Figure 5.7 shows an ground truth example.

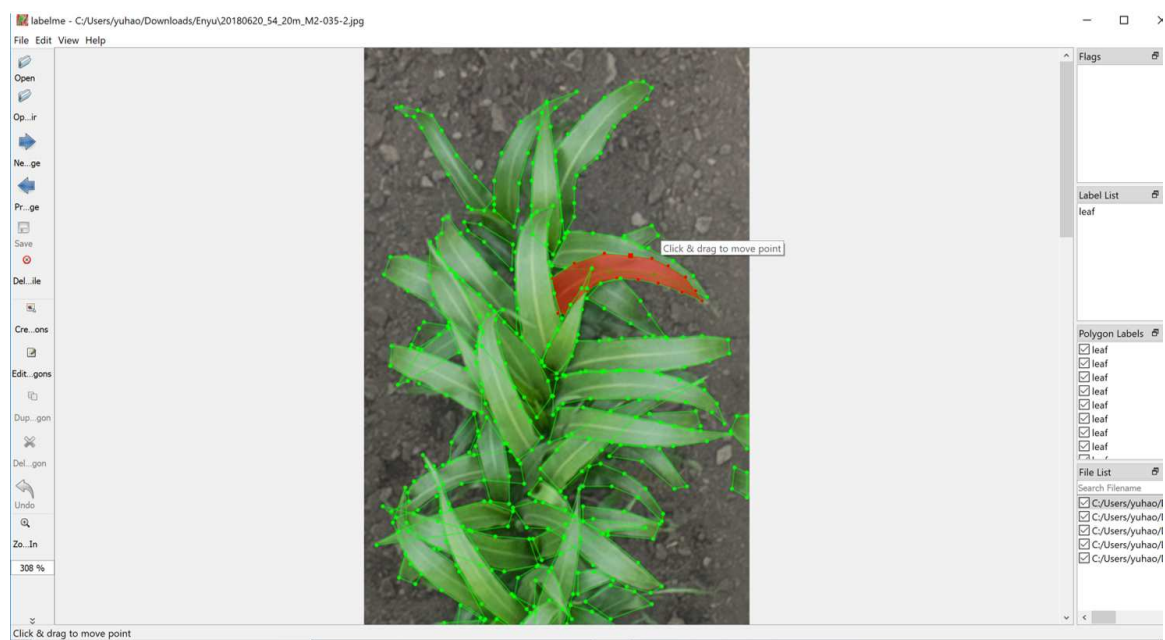


Fig. 5.6. Our ground truth tool (labelme [158])



Fig. 5.7. A ground truth example

5.3 Slice Based Leaf Segmentation

5.3.1 Motivation

Plant leaves have similar morphology but different traits such as leaf length and width. Each leaf has a curved shape due to its weight and its interaction with neighboring leaves. The leaves of a sorghum plant are thin and long. As a result, the two edges of a sorghum leaf are semi-parallel to each other even though the edges are curved. I exploit this feature to segment leaves. From the segmentation of each individual leaf, I estimate leaf traits. In this section, I present a shape-based approach to leaf segmentation.

5.3.2 Proposed Method

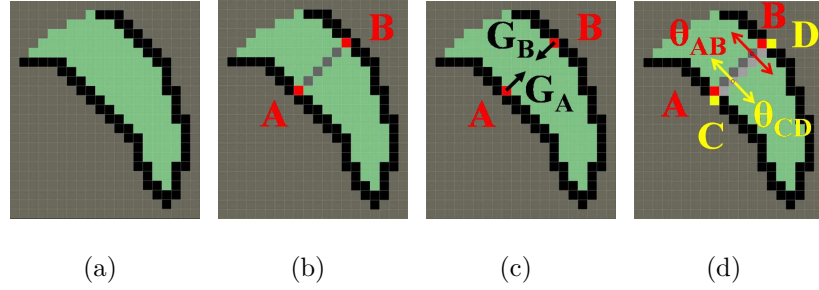


Fig. 5.8. a) A synthetic leaf. b) A leaf slice connecting two pixels, A and B , at opposite edges of the leaf. c) G_A and G_B are the gradient angles of pixels A and B . d) θ_{AB} and θ_{CD} are the angles the of slices S_{AB} and S_{CD} .

A and B are two pixels located at two opposite edges of a leaf. A pixel-wide line connecting A and B is defined as the leaf slice S_{AB} . Figure 5.8 depicts these definitions with a synthetic leaf. The gradient angles at A and B are G_A and G_B , respectively. The angle 0° is defined in the direction of the x axis. G_A and G_B are expected to be opposite to each other with some bias T_a , i.e,

$$|G_A - G_B + \pi| \mod 2\pi < T_a \quad (5.1)$$

Leaf slices are obtained by using the Stroke Width Transform [159]. The slice angle θ_{AB} of S_{AB} is defined as the normal of the slice (in radians) as

$$\theta_{AB} = \frac{G_A + G_B}{2} \mod \pi \quad (5.2)$$

In order to reconstruct leaves from leaf slices, adjacent leaf slices, S_{AB} and S_{CD} , are compared. If their angles θ_{AB} and θ_{CD} differ less than a constant T_b , i.e.,

$$|\theta_{AB} - \theta_{CD}| < T_b \quad (5.3)$$

then slices S_{AB} and S_{CD} are merged.

Plants with high leaf density can cause leaves to overlap with each other. In the case of leaf overlap, there may be a discontinuity between leaf slices as shown in Figure 5.9.

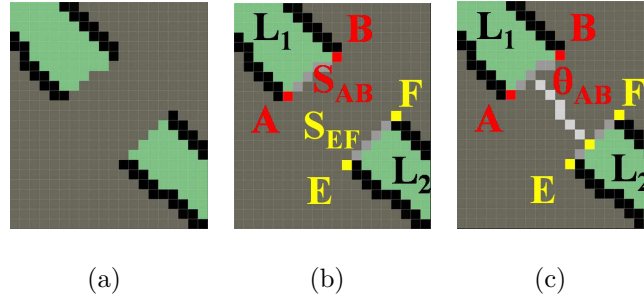


Fig. 5.9. a) A single leaf with a discontinuity that may be due to a leaf crossing. b) Two leaf segments L_1 and L_2 with almost parallel facing slices S_{AB} and S_{EF} . c) From leaf slice S_{AB} , I search in the direction of θ_{AB} until I find the slice S_{EF} of the opposite leaf segment L_2 .

In this case, the two leaf segments will be merged by the following search method. Let a leaf be split into two leaf segments L_1 and L_2 , separated by a discontinuity, as shown in Figure 5.9. Let S_{AB} be the leaf slice with angle θ_{AB} at the end of L_1 .

Let S_{EF} be the leaf slice with angle θ_{EF} at the end of L_2 . Let X_{AB} to be a vector contains all the pixels in S_{AB} . From pixel x_i in X_{AB} , I search in the direction of θ_{AB} . If a leaf slice S_{EF} from another leaf L_2 is found and the difference between the two angles is less than a constant T_c , i.e.,

$$|\theta_{AB} - \theta_{EF}| < T_c, \quad (5.4)$$

then leaves segments L_1 and L_2 are considered the same leaf.

Note that the thresholds T_a, T_b, T_c are determined experimentally. This method addresses leaf discontinuity due to leaf crossing, but requires that leaf edges are clearly distinguishable. Overlap of parallel leaves may lead to undersegmentation.

5.3.3 Experimental Results

A total of 52 Sorghum rows such as the one shown in Figure 5.10 were analyzed. From the segmentation of each leaf, the leaf count was obtained. The thresholds I used for these images were empirically chosen as $T_a = \frac{\pi}{5}$, $T_b = \frac{\pi}{8}$, and $T_c = \frac{\pi}{6}$. T_a controls the number of slices to be detected. A lower value reduces the number of detected slices. A higher value increases false positives. T_b controls the curvature allowed in a segment. A higher value allows more curved segments. T_b controls the curvature allowed in a leaf. A higher value allows more curved leaves.



(a)



(b)

Fig. 5.10. a) A single row sorghum plot. b) Leaf segmentation. Individual leaves are shown in different colors.

5.4 Leaf Segmentation In Polar Coordinates

5.4.1 Motivation

My slice-based leaf segmentation method makes use of the semi-parallel edge feature from sorghum leaves. The detection of leaf segments are noisy and contain holes. In order to have complete leaf masks, I propose the following method to fit triangular shape models to leaves so the leaves can be reconstructed from the shape models.

5.4.2 Leaf Segmentation

As plants grow, their leaves grow around the centers. Analyzing leaves requires considering them at different orientations. To simplify the analysis, I transform the image into polar coordinates and make all the leaves point in the same direction. I denote $I(x, y)$ as the RGB values of image at pixel coordinates (x, y) . I denote (C_x, C_y) as the pixel coordinates of the plant center. I convert the circular region with radius R around the plant center into a polar coordinate image $P(r, \theta)$:

$$P(r, \theta) = I \left(\left\lfloor C_x + r \cos \left(\frac{\pi}{180} \theta \right) \right\rfloor, \left\lfloor C_y + r \sin \left(\frac{\pi}{180} \theta \right) \right\rfloor \right), \quad (5.5)$$

where $\lfloor \cdot \rfloor$ denotes rounding to the closest integer, and $0 \leq r < R$ and $0 \leq \theta < 360$ are integers. Figure 5.11(a) and Figure 5.11(b) show an example of the polar coordinate transformation. In polar coordinates, all leaves point along the positive r axis.

I assume there are only two classes in my dataset: plant material (foreground) and soil (background). I apply the plant segmentation described in chapter 3 to obtain a foreground mask $F(r, \theta)$ such that

$$F(r, \theta) = \begin{cases} 1 & \text{if } P(r, \theta) \text{ contains plant material} \\ 0 & \text{otherwise.} \end{cases} \quad (5.6)$$

I obtain the background mask $B(r, \theta)$ by computing the complement of the foreground mask,

$$B(r, \theta) = 1 - F(r, \theta). \quad (5.7)$$

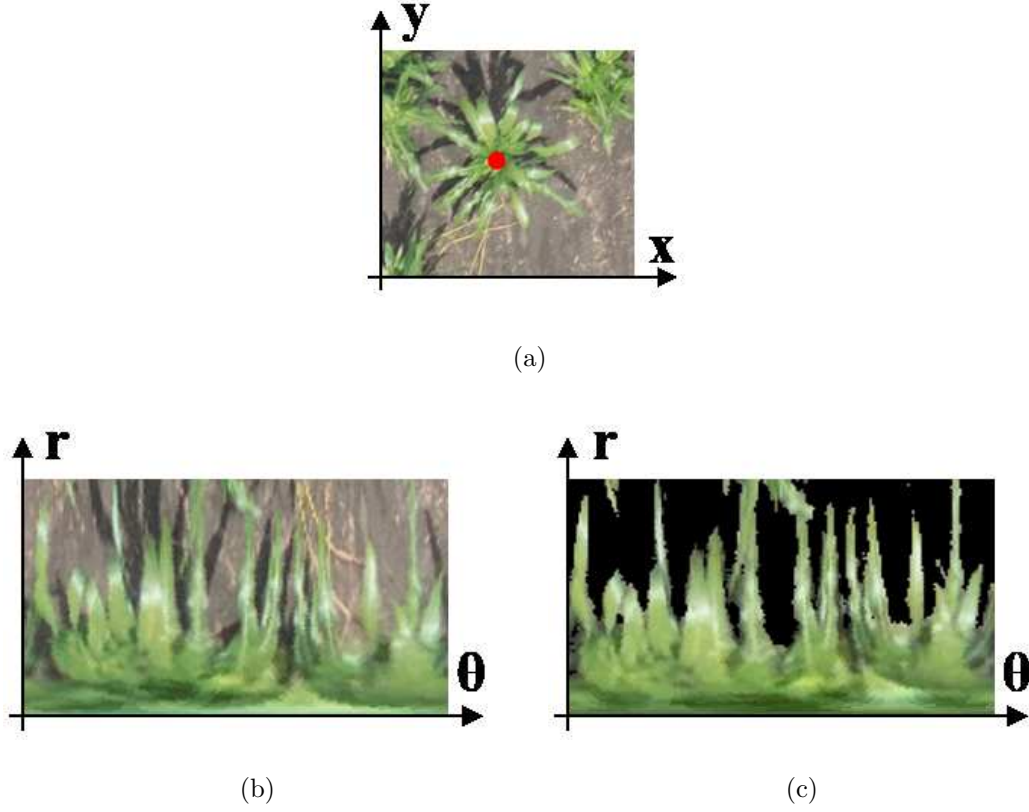


Fig. 5.11. a) A plant with its center labeled in red. b) Polar representation of a circular neighborhood around the plant. c) Segmented plant in polar coordinates.

An example of the segmented foreground is shown in Figure 5.11(c).

Pixel-wise plant segmentation generates noise presented as holes in the foreground mask $F(r, \theta)$. In order to remove holes without distorting the leaf morphology in the segmentation mask, I use connected components [97]. By applying connected components on the background mask $B(r, \theta)$, I obtain the number of components \mathcal{C} and a labeled image $\Omega_B(r, \theta)$ with range $[1, \mathcal{C}]$ indicating the component label of the coordinate (r, θ) . Let \mathcal{H} indicates the set of all labels whose component has less than T_A pixels:

$$\mathcal{H} = \{c \in [1, \mathcal{C}] : \sum_{r=0}^{R-1} \sum_{\theta=0}^{359} \delta(\Omega_B(r, \theta) - c) < T_A\}, \quad (5.8)$$

where $\delta(\cdot)$ is the Kronecker delta function. I fill the holes in the foreground mask $F(r, \theta)$, by moving all components in \mathcal{H} into $F(r, \theta)$ as

$$F(r, \theta) \leftarrow F(r, \theta) + \sum_{c \in \mathcal{H}} \delta(\Omega_B(r, \theta) - c). \quad (5.9)$$

I assume independent components appear in the foreground mask are leaves from neighboring plants. Connected components is applied on the foreground mask $F(r, \theta)$ to retrieve all foreground objects with a labeled image $\Omega_F(r, \theta)$. The component that contains the plant center is the plant I am interested in. The plant center is a single pixel in Cartesian Coordinates. When transforming into polar coordinates, the plant center becomes a line spanning $r = 0$. Therefore, $\Omega_F(0, 0)$ is the label of the connected component that corresponds to the plant. I update the foreground mask to contain only the plant component, and

$$F(r, \theta) \leftarrow \delta(\Omega_F(r, \theta) - \Omega_F(0, 0)). \quad (5.10)$$

I present a basic element, a slice, for analyzing the leaf morphology. A slice is a connected set of coordinates in the foreground mask $F(r, \theta)$ with a constant r . Figure 5.12(b) shows an example of a slice marked in red.

I stack leaf slices vertically, and obtain a leaf shape. The parent-child relationship of two vertically connected slices is defined as the following. The slice at a smaller r value is the parent of the slice at a larger r value. Figure 5.12(c) shows the parent slice in blue and the child slice in green. With this relation, I construct a hierarchical representation of the slices (see Figure 5.12(a) and Figure 5.12(d)). I denote a node as a slice in the hierarchical representation. The root node is the slice at $r = 0$ and corresponds to the plant center. There are three possible hierarchical relations between nodes. The simplest case occurs when a parent node has only one child node and the child also has only one parent. This one-to-one relation usually indicates that both the parent and child node are from a portion of a leaf. A different case occurs when a parent node has multiple child nodes and each child node has only one parent. In the last case, multiple parent nodes share the same child node. This

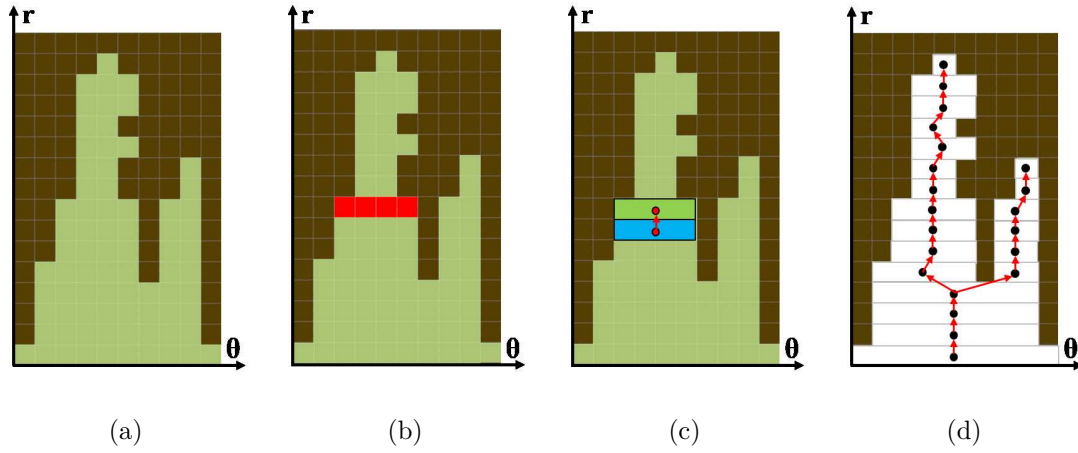


Fig. 5.12. a) Synthetic image with two leaves. b) A slice marked in red. c) Parent slice (blue) and child slice (green). d) Hierarchical slice representation.

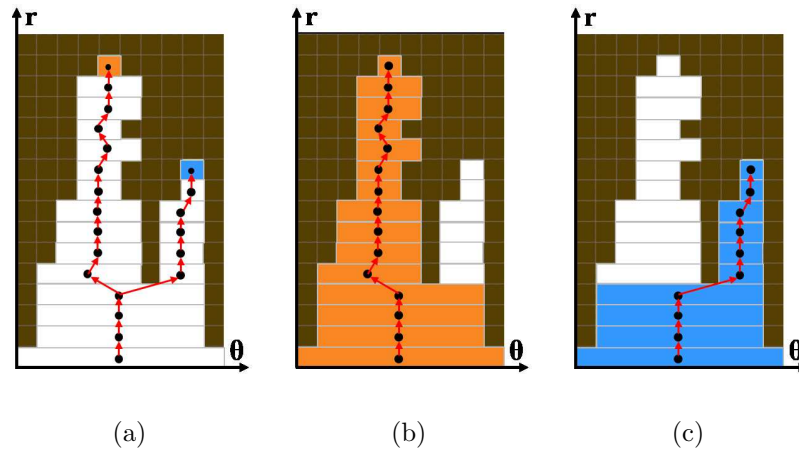


Fig. 5.13. a) Tip slices (orange and blue) in a hierarchical slice representation. b) Leaf segmented with an orange tip slice. c) Leaf segmented with a blue tip slice.

relation is caused by holes in the foreground mask that remained after the noise removal step. When this occurs, I compute the distance between the slice centers of the parent nodes and the slice center of the child node. I assign the child node to its

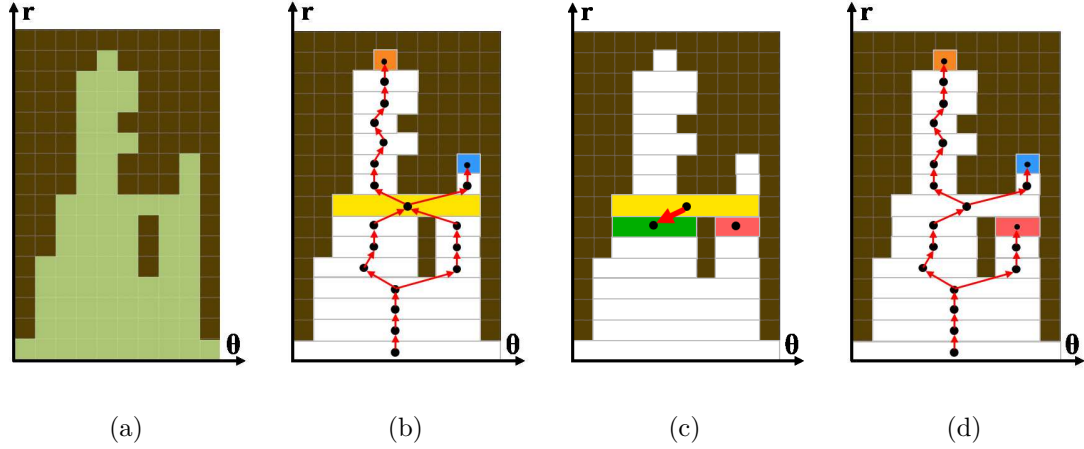


Fig. 5.14. a) Two synthetic leaves and a hole in the foreground mask. b) Hierarchical slice representation. The slice in yellow is where the two slices merge. The orange and blue slices are the two tip slices. c) The yellow slice is assigned to the closest slice (green). d) Hierarchical slice representation after the assignment. The three tip slices found are marked in orange, blue, and red slice.

closest parent node according to these distances. I mark the other parent nodes as end-nodes of the hierarchy (see Figure 5.14).

I define the tip of a leaf as a node with no child. I reconstruct a leaf shape by combining the tip with all its ancestors (see Figure 5.13(a), Figure 5.13(b), and Figure 5.13(c)). I define set

$$\mathcal{A} = \{X_1, X_2, \dots, X_N\}. \quad (5.11)$$

containing all N reconstructed leaf shapes $X_i, i = 1, \dots, N$. I define an observed leaf shape $X \in \mathcal{A}$ as

$$X = \{x_1, x_2, \dots, x_{L_X}\}. \quad (5.12)$$

where L_X is the number of slices in the leaf shape X . As each slice is one pixel thick, L_X is the length of the leaf shape X . Each slice $x_m, m = 1, \dots, L_X$, is $\Theta(x_m)$ degrees wide.

5.4.3 Leaf modeling

I use a shape modeling technique to remove noise and inter-plant occlusion. Figure 5.15 shows an example of the shape modeling of a leaf. Inter-plant occlusion is defined as leaves from neighboring plants occluding the peaks I want to seek as shown in Figure 5.15(a). In addition, I define self-leaf as a leaf that belongs to the plant being analyzed and neighbor-leaf as a leaf belonging to neighboring plants. These two type of leaves vary in the distribution of their leaf widths. The width of the slices in self-leaves tends to decrease as r increases. On the contrary, the width of the slices in neighbor-leaves tends to increase as r increases. Inter-plant occlusion may create hourglass-like shapes (see Figure 5.15(a)). In order to estimate the true shape of each leaf in these scenarios, I define a finite set of K possible shape models:

$$\mathcal{D} = \{S_1, S_2, \dots, S_K\}. \quad (5.13)$$

Each shape model $S \in \mathcal{D}$, consists of a set of slices:

$$S = \{y_1, y_2, \dots, y_{L_S}\}, \quad (5.14)$$

where L_S is the number of slices in the shape model S .

The width $W_m, m = 1, \dots, L_S$ of the m -th slice y_m in shape model S is modeled as random variable with normal distribution

$$p_{W_m}(w) = \mathcal{N}(\mu_m, \sigma_m^2), \quad (5.15)$$

where μ_m and σ_m^2 are the mean and variance, respectively. I call $p_{W_m}(\cdot)$ the slice matching probability. I set the value of σ_m^2 as

$$\sigma_m^2 = \max\{1, \tau\mu_m\}, \quad (5.16)$$

where τ is the tolerance rate. The higher the τ , the more disperse the width of a slice can be. I constrain σ_m^2 to be higher than 1 degree to tolerate a minimum amount of noise.

For an observed leaf shape $X \in \mathcal{A}$ and any possible shape model $S \in \mathcal{D}$, I define a shape model matching score that takes all slice matching probabilities into account:

$$f(X, S) = \sum_{m=1}^{L_S} \omega_{m,S} p_{W_m}(\Theta(x_m)), \quad (5.17)$$

where $\omega_{m,S}$ is the weight for the m -th slice of the shape model S . I assume that all the slices are equally weighted in the model, and

$$\sum_{m=1}^{L_S} \omega_{m,S} = 1. \quad (5.18)$$

I use the matching score to find the best shape model \hat{S} to describe the observed leaf shape X as

$$\hat{S} = \underset{S \in \mathcal{D}}{\operatorname{argmax}} f(X, S). \quad (5.19)$$

For each of all the observed leaf shapes $X_n, n = 1, \dots, N$, I obtain \hat{S}_n . Some X_n may be matched to a shape model even if it is purely noise. To avoid this, I set a threshold T_B below which the match will be rejected, and the observation will not be considered. I use Otsu's method [160] to find the threshold T_B that separates strong and weak matching scores. I remove any observed leaf shape that has a matching score smaller than T_B . I denote the new leaf shape set as \mathcal{A}' and the number of matched shapes as N' .

Finally, I transform the matched shape model back to Cartesian coordinates to estimate phenotypic traits. A horizontal line in polar coordinates is an arc in Cartesian coordinates. The width of the arc is called cord length. For a model shape $S \in \mathcal{D}$, I obtain the cord length c_m of the m -th slice y_m in S as

$$c_m = 2r_m \sin\left(\frac{\mu_{m,r}}{2}\right), \quad (5.20)$$

where r_m is the r coordinate of slice y_m , and $\mu_{m,r}$ is the mean μ_m in radian. I estimate the total area of S as

$$\begin{aligned} A(S) &= \left(\sum_{m=1}^{L_S} c_m \right) + \frac{r_m^2}{2} (\mu_{L_S,r} - \sin(\mu_{L_S,r})) \\ &= \left(\sum_{m=1}^{L_S} 2r_m \sin\left(\frac{\mu_{m,r}}{2}\right) \right) + \frac{r_m^2}{2} (\mu_{L_S,r} - \sin(\mu_{L_S,r})), \end{aligned} \quad (5.21)$$

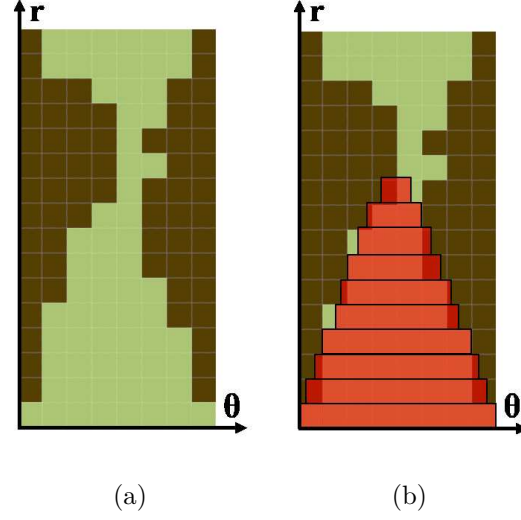


Fig. 5.15. a) Two synthetic leaves from different overlapping plants. b) A shape model (marked in red) to describe the observed leaf shape.

where the second term corresponds to the tip of the leaf, a small circular segment area. LAI is obtained as

$$\text{LAI} = \frac{\sum_{n=1}^{N'} A(\hat{S}_n)}{A_u}, \quad (5.22)$$

where A_u is the unit area the plant occupies.

5.4.4 Experimental Results

I evaluated my method with the image shown in Figure 4.16. This image was acquired from a UAV flying at an altitude of 40 meters with a speed of 8m/s on June 30, 2016 in West Lafayette, Indiana. The centers of 10 plants were manually selected. R , the radius of the circular neighborhood around each plant center, was set to 70 pixels. I set the area threshold T_A for rejecting background holes to 150. Shape models were generated as triangles in polar coordinates. The widths μ of the triangles spanned between 12 degrees and 100 degrees, and the length was between 20 and 70 pixels. A total of 4,539 shape models were constructed. The tolerance rate τ was set to 0.3 empirically. A_u , the unit plant area was set to 19,600 pixels.

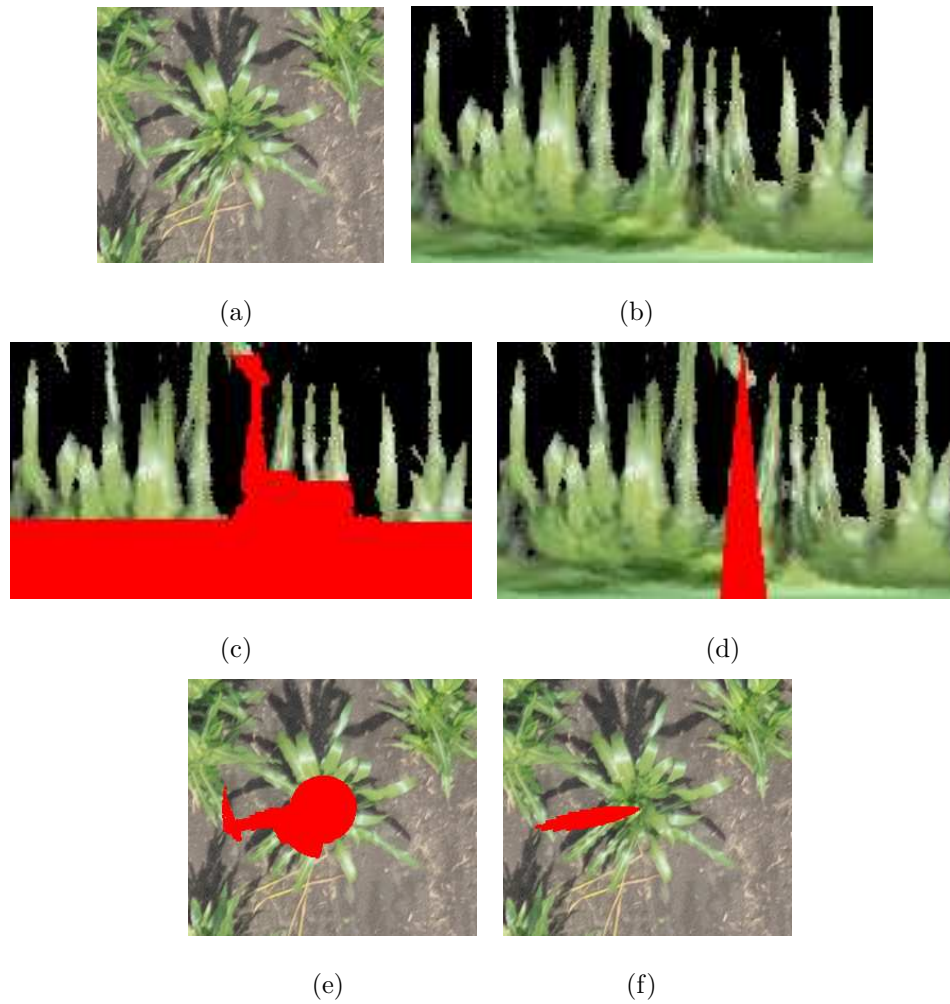


Fig. 5.16. a) The original image. b) Image segmented by color thresholding in polar coordinates. c) Segmentation of a leaf marked in red. d) The best shape model that describes the leaf is marked in red. e) Segmentation of the leaf transformed back into Cartesian coordinates. f) Shape model transformed back into Cartesian coordinates.

The leaf segmentation was evaluated by visually comparing the original images (such as Figure 5.16(a)) with the segmented images (such as Figure 5.17(a)). The method was evaluated using the 10 selected plants. The number of leaves in the selected plants was 82. Out of these 82 leaves, a total of $TP = 57$ true positive leaves were correctly detected. A total of $FP = 18$ false positive leaves were detected, meaning that I incorrectly detected 18 noisy segments of the image as leaves. Out of these 82 leaves, the method produced a total of $FN = 25$ false negative samples, meaning that 25 leaves could not be detected. Precision and recall was used to evaluate the performance of my method [117,118]. I obtained a precision and recall of 69.5% and 76.0%, respectively. Since the number of false positives and the number of false negatives is similar, the precision and recall are similar. There is no bias towards missing leaves or overdetecting leaves. Hence, the average number of detected leaves is very close to the true value. Based on the shape models, phenotypic traits were estimated. Traits included the area (Equation (5.21)) and length (L_S) of each leaf, and the leaf count (N') and LAI (Equation (5.22)) of individual plants. Figure 5.16 shows the segmentation result of a leaf for one plant. The length and area of the leaf shown was estimated to be 98 pixels and 1,000 square pixels, respectively. The final segmentation result of an entire plant is shown in Figure 5.17(a) and 5.17(b). I correctly detected and segmented 8 out of 14 leaves and the LAI of the plant was estimated to be 0.21.

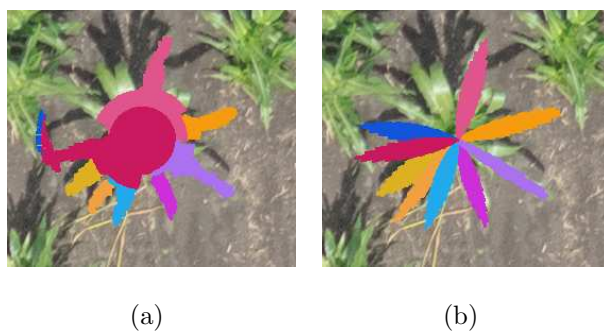


Fig. 5.17. a) Different leaves are segmented in different colors. b) The models describing each leaf are marked in different colors.



(a)



(b)

Fig. 5.18. a) An image of a single row plot of crops acquired from an altitude of 50 meters. b) Leaf segmentation result generated by my method, where leaves are colored randomly.

5.5 Leaf Segmentation by Functional Modeling

5.5.1 Motivation

In my previous “Leaf Segmentation In Polar Coordinates” method, leaf shapes are modeled as triangles, but the triangle models may fit erroneously when a leaf bends and has a curved shape. In addition, the method requires an accurate plant center as input, which can be difficult to obtain. In my application, where leaf shapes are crucial for measuring phenotypic traits (such as maximum width), popular segmentation method CNNs 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 may 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. In addition,

these functions provide easy computation for traits estimation such as leaf width and leaf length.

I present a new approach to instance leaf segmentation by modeling leaf edges with continuous functions. This method combines the advantage of my previous leaf segmentation methods. The 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 shape model is described by two continuous polynomial functions representing the edges. My method is compared to a popular segmentation method Mask R-CNN and achieves better results. Figure 5.18 shows an example result of my method. The plant used in this study is sorghum [*Sorghum bicolor* (L.) Moench] [1, 14].

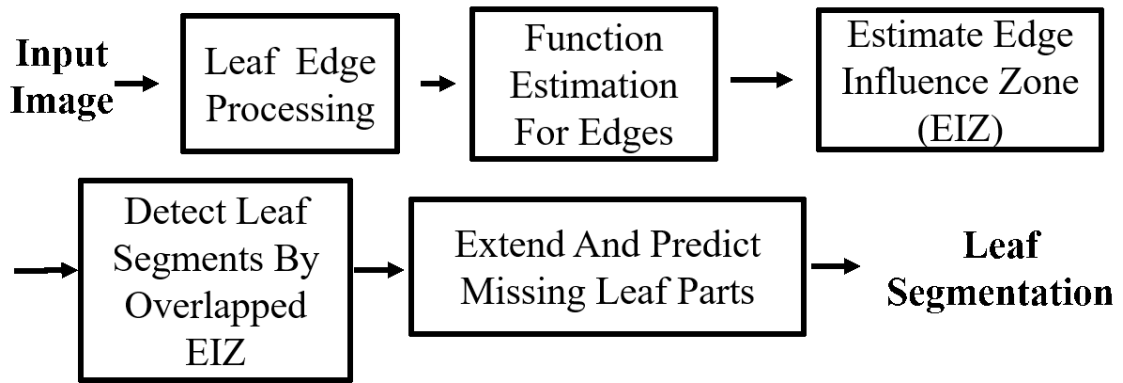
5.5.2 Processing Structure and Assumptions

My proposed method uses edge features and the semi-parallel relationship between the two leaf edges to detect, model, and predict leaf parts. Figure 5.19 shows the block diagrams of my proposed approach.

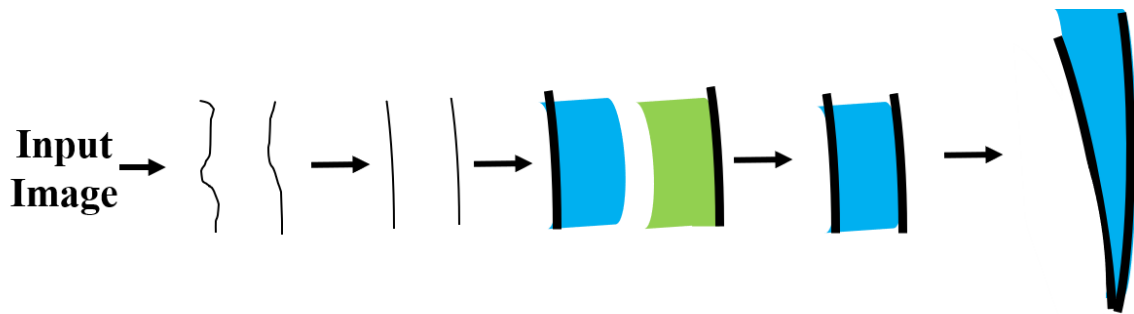
The following definitions and assumptions are used in this method.

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

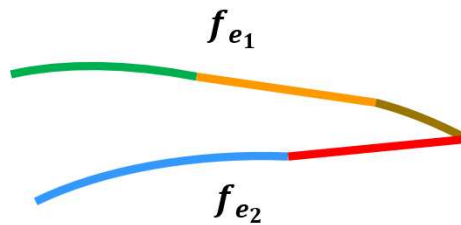
In computer graphics, a curve in 2D space can be represented by many functions and forms [161]. Among the representations, a polynomial is easy to estimate and model. In this paper, I use second-order polynomials for the leaf edge functions. Second-order polynomials in 2D are either convex or concave, but leaf edges have different orientations. Hence, I rotate the edge pixels before estimating the function.



(a)



(b)



(c)

Fig. 5.19. a) Block diagram of proposed approach. b) Graphical block diagram. c) Piece-wise edge function f_{e1} and f_{e2} . Each color represents an estimated polynomial.

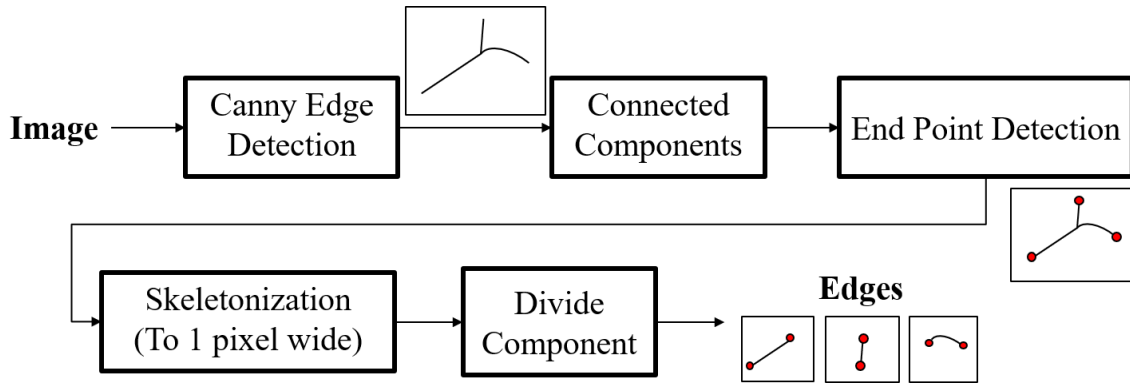


Fig. 5.20. Block diagram for edge processing

The rotation can be done in various ways. I use Principal Component Analysis (PCA) [162] 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 5.19(c)). The estimated function is the fundamental structure I use to represent a leaf edge. I 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. I 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 $\frac{\sqrt{2}}{2}$.

5.5.3 Leaf Edge Processing

The goal of this subsection is to process an image to extract edges. Figure 5.20 shows a block diagram of edge processing.

Given an RGB image, I first use the Canny Edge operator [163] to obtain an edge mask M_e . This edge mask can be refined by removing unwanted features from the background. I create a plant material mask M_p by thresholding the plant image in

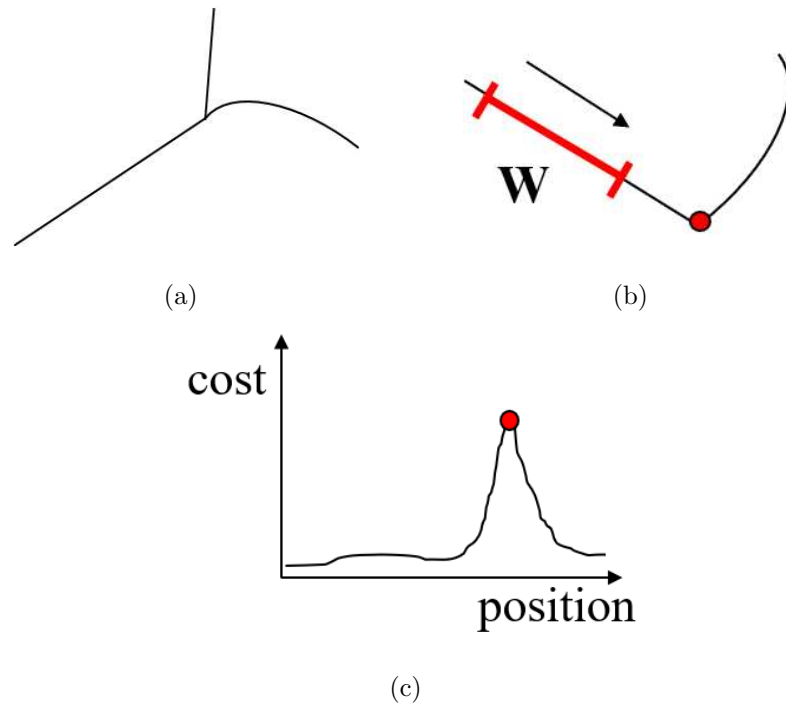


Fig. 5.21. a) An example skeleton with branching structure. b) An example skeleton with edges from different leaves forming a single line. W is a sliding window for function estimation. The red dot is the ideal break point between the two piecewise functions. c) Function estimation cost vs. window position.

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

$$M_r = M_e \wedge M_p. \quad (5.23)$$

I use Connected Components [164] on the refined edge mask M_r to group the connected edge pixels into sets, and I 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 I consider. First, detected edge pixels from different leaves forming a branching structure, as shown in Figure 5.21(a). Second, detected edge pixels from different leaves forming a single line (as shown in Figure 5.21(b)).

My approach to these issues is to use Depth First Search (DFS) [165] on the edge components and to separate the branches during the search. Edges from different leaves in each branch are disjointed. This approach requires each edge component to be a one pixel wide skeleton. Skeletons wider than one pixel create cycles and ambiguous paths. I propose a skeletonization method in Subsection 5.5.3 to thin the edge components into one pixel wide.

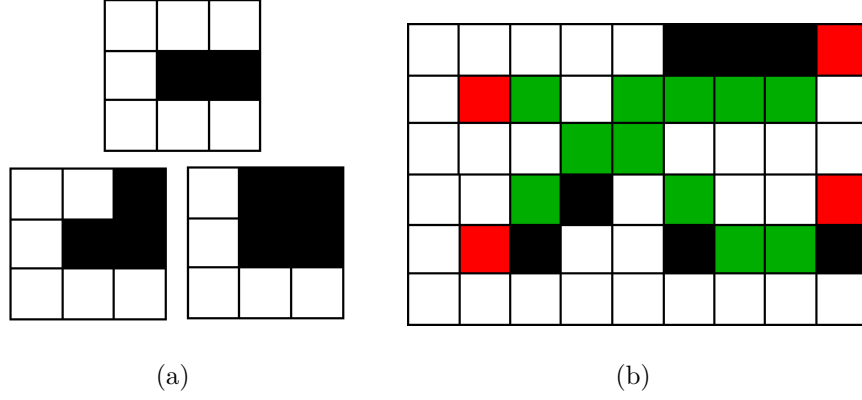


Fig. 5.22. a) End point cases. b) Shortest path (red: end points, green: shortest paths, black: unvisited pixels).

Skeletonization

The edge components created by Connected Components are usually not one pixel wide. I need to thin the edge components to form skeleton structures. I use a shortest path [165] approach between end points of a skeleton to produce a map of node visiting frequency. I prioritize selecting a path through frequently visited nodes.

I 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 [165]. Figure 5.22(a) shows an example of an end point's neighborhood. I use Minimum Cost Path (MCP) [166] 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 $E = \{e_1, e_2 \dots e_{N_s}\}$ 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:

$$C_s = \begin{cases} N_s^2 + 1 & \text{if } M_s > 0 \\ \inf & \text{otherwise.} \end{cases} \quad (5.24)$$

The detailed implementation of the steps is shown in Procedure 2. Figure 5.22(b) shows an example of the skeletonization. Unvisited pixels are removed from the skeleton after all end point pairs have been examined.

Procedure 2: Skeletonization

Input : End point set: $E = \{e_1, e_2 \dots e_{N_s}\}$,
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}

Skeleton Separation

First, I address the case of edges forming a branching structure, as shown in Figure 5.21(a). I 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 3). As a result, I 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.

I next address the case of edges from different leaves forming a single line, as shown in Figure 5.21(b). I 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 [167]. 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 5.21(b) shows an example of a break point between two piecewise functions. Figure 5.21(c) 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 quantization 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, I break the branch into two at the window center. Procedure 4 shows the detailed steps. At this point, I obtain sets of connected 2D discrete functions that are ready to be combined into leaf segments.

5.5.4 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 detect in low resolution images. I 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

Procedure 3: Depth First Search

Input : Skeleton pixel set: S ,

End points: $E = \{e_1, e_2 \dots e_{N_s}\}$, where N_s
is the number of end points,

Intersections: $I = \{i_1, i_2 \dots i_{N_i}\}$, 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

Procedure 4: Divide

Input : Branch: $b = \{a_1, a_2, \dots\}$,
 Window size: W
 Threshold: τ

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

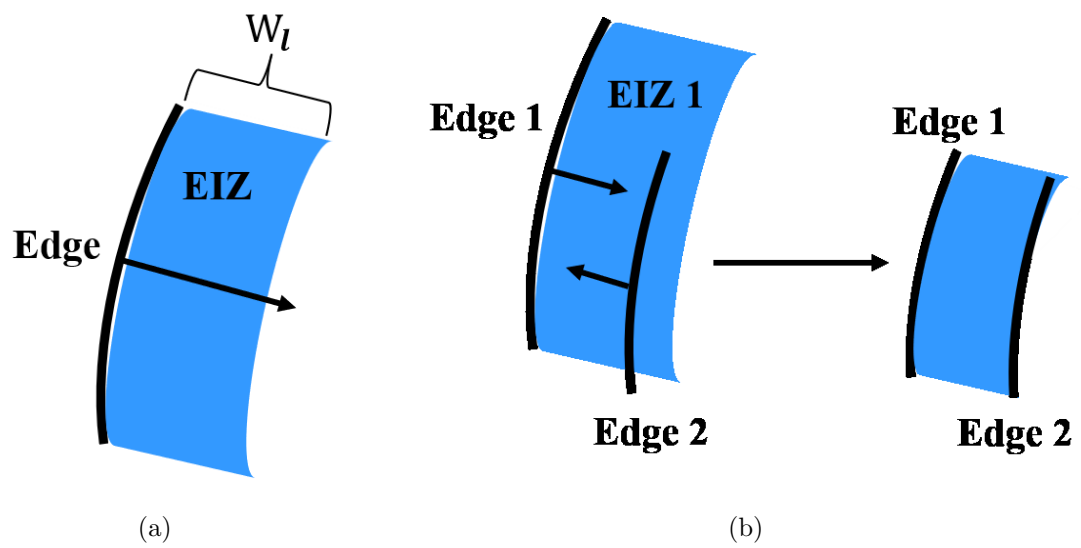


Fig. 5.23. a) Smearing an edge in the direction of its average gradient angle to create EIZ. b) Detecting candidate edges to form a leaf segment.

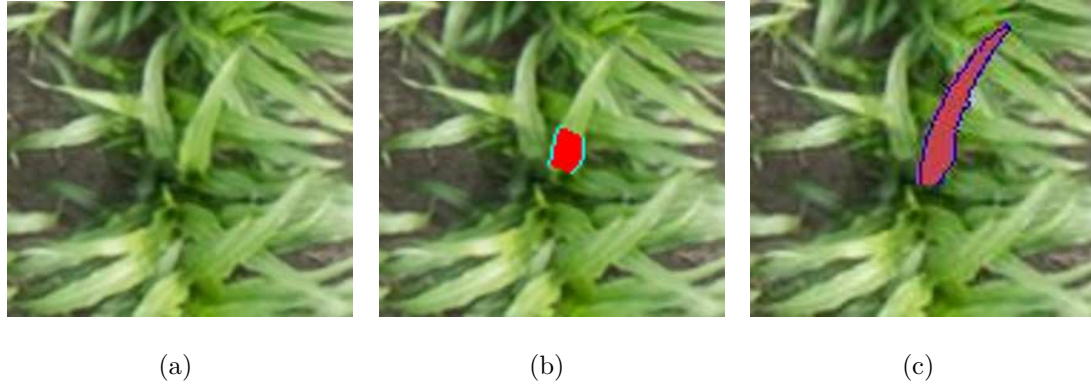


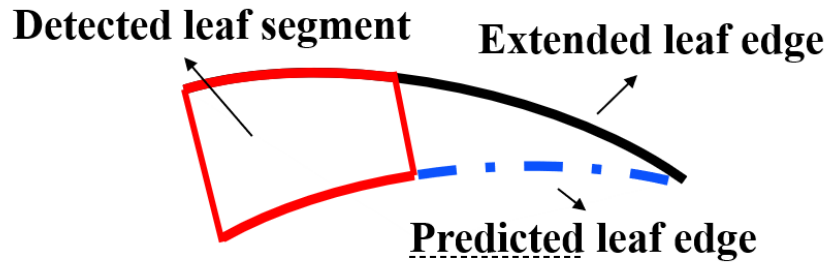
Fig. 5.24. Example segmentation of a leaf, a) Original image. b) Detected leaf segment. c) Full segmentation of the leaf.

gradient angle is computed for each edge. I assume if two edges have opposite average gradient angles, they form some parts of a leaf and are considered an edge pair.

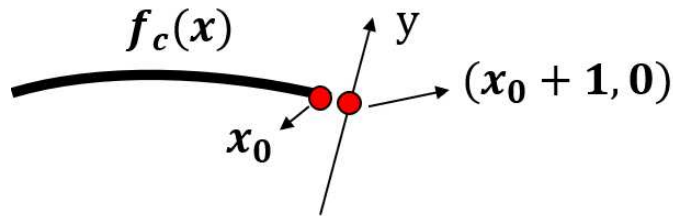
To find edge pairs, I smear each leaf edge in the direction of its average gradient angle with a maximum leaf width W_l . I call the area generated by the smearing an Edge Influence Zone (EIZ), as shown in Figure 5.23(a). Any edge inside an EIZ is a candidate for an opposite leaf edge. Candidate edges 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. I call the edges in the segment base edges. This procedure is shown in Figure 5.23(b). Figure 5.24(b) shows an example of a detected leaf segment.

5.5.5 Leaf Segment Completion

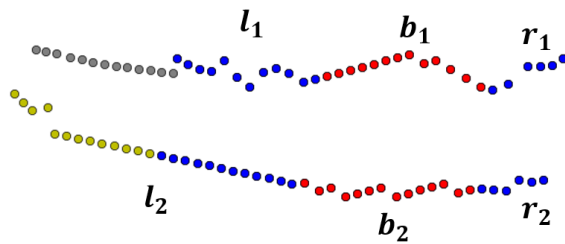
At this stage, a leaf edge can be categorized into three types: a base edge in a leaf segment, an edge discovered by Canny edge detection, and an undetected edge due to its weak response to the edge operator. I use the edges from leaf segments to extend and predict leaf edges. Figure 5.25(a) shows the three types of edges.



(a)



(b)



(c)

Fig. 5.25. 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).

Edge Walking

I introduce a method I 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 [168].

Let $f_c(x)$ be an edge function and x_0 be the end of the edge I am considering. I 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 5.25(b)). I 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 [169] 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 I obtain the slice S_{x_0+1} . The slice profile associated is $g_{x_0+1}(y)$, and its derivative is $g'_{x_0+1}(y)$. My goal is to find the true edge location y in S_{x_0+1} . I model the probability distribution $P_p(y)$ for the 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 $i = 1, 2, \dots, N_e$ and N_e is the number of observed edges. I model the probability distribution $P_e(y)$ for the Canny edge point y as a Gaussian Mixture Model (GMM) [170]:

$$P_e(y) = \sum_{i=1}^{N_e} w_{e,i} \mathcal{N}(L_{e,i}, \sigma), \quad (5.25)$$

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. I consider all derivative locations where the absolute value is above a threshold τ_3 as my observations. Let $L_{d,i}$ be the observed derivative location in a slice profile $g_{x_0+1}(y)$, where $i = 1, 2, \dots, N_d$ and N_d is the number of observed derivatives. I denote $P_d(y)$ as the probability distribution for a derivative observation y . $P_d(y)$ is also modeled as a GMM:

$$P_d(y) = \sum_{i=1}^{N_d} w_{d,i} \mathcal{N}(L_{d,i}, \sigma), \quad (5.26)$$

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:

$$w_{d,i} = \frac{|g'_{x_0+1}(L_{d,i})|}{\sum_{i=1}^{N_d} w_{d,i} |g'_{x_0+1}(L_{d,i})|} . \quad (5.27)$$

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

$$P_o(y) = \begin{cases} P_e(y) & \text{if } N_e > 0 \\ P_d(y) & \text{if } N_e = 0 \text{ and } N_d > 0 \\ 0 & \text{otherwise.} \end{cases} \quad (5.28)$$

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

$$P(y) = P_p(y)P_o(y) . \quad (5.29)$$

The Maximum Likelihood Estimate of y is:

$$\hat{y} = \underset{y}{\operatorname{argmax}} P(y) . \quad (5.30)$$

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

Leaf Modeling

After Edge Walking, I 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 Subsection 5.5.4, I obtain a matching base edge pair. Let b_1 and b_2 denote

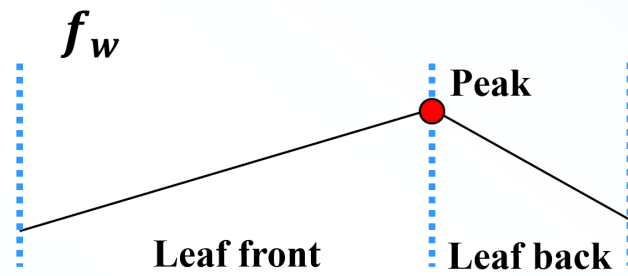


Fig. 5.26. Leaf width function f_w .

the two base edges. In Subsection 5.5.5, I obtain a left (l) and right (r) extension for each base edge. I 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 5.25(c). 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. I 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 , I 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, I compute its middle point. Since a leaf may twist, modeling its midrib function with one polynomial is sub-optimal. I 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 Subsection 12. The middle polynomial is estimated by the rest of the points. Combining the three polynomials, I obtain the midrib function f_m .

To estimate the width function f_w , I take one sample per pixel on the midrib function f_m . At each pixel, I 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. I model the width function f_w as a concave piece-wise function consisting of two lines (as shown in Figure 5.26). The function has only one peak. Let p be the location of the peak. As shown in Figure 5.26, 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. I estimate one function for the leaf front and one for the leaf back. Combining the two functions, I obtain the width function f_w .

Finally, I predict matching edge points for the unpaired point sets u_l and u_r . For each point z in an unpaired point set, I 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 5.25(c) shows an example of detected edge points, extended edge points, and predicted edge points. Figure 5.24(c) shows an example of leaf segmentation after edge extension and prediction.

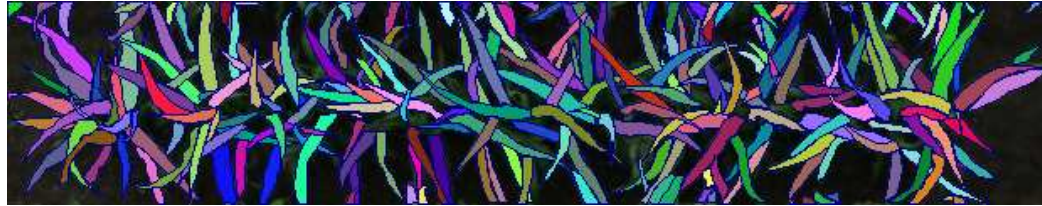
A missing leaf tip can be predicted using the midrib function f_m and width function f_w . However, the appearance of a leaf tip in a UAV image varies across plant breeds. For example, a plant breed with thin long leaves often has visible tips, while a plant breed with wide leaves may have its tips pointing 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 I do not consider prior knowledge of plant breeds.

Post-processing

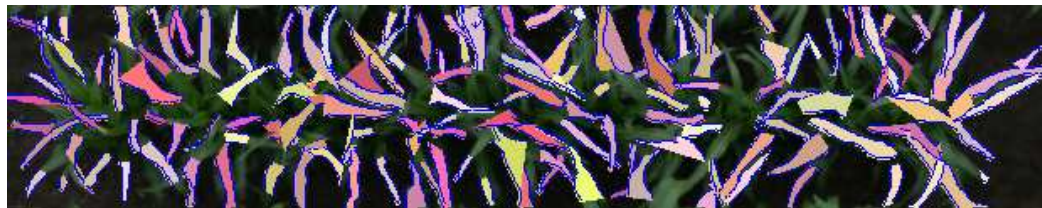
I may detect multiple edge pairs from the same leaf. My proposed method generates a leaf segmentation mask for the entire leaf using only one of its detected edge pairs. Thus, there can be redundant leaf segmentation as similar masks can be developed from different edge pairs of the same leaf. As a solution, I 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. I remove contours from a segmentation mask using morphological erosion [171] 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.



(a)



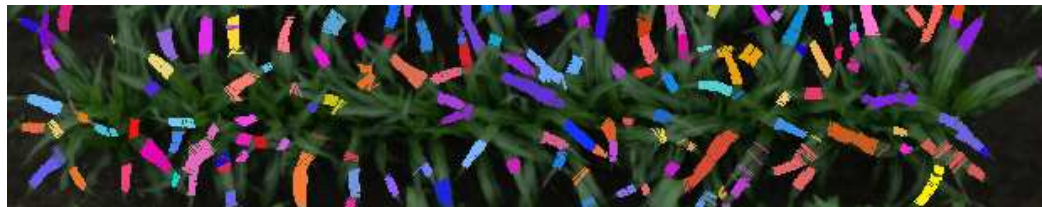
(b)



(c)



(d)



(e)

Fig. 5.27. 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 my method, where leaves are colored randomly. d) Result image generated by Mask R-CNN, where leaves are colored randomly. e) Result image generated by slice-based leaf segmentation, where leaves are colored randomly.

Table 5.1.

Segmentation results of the proposed method and Mask R-CNN 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

5.5.6 Experimental Results

UAV images often have low spatial resolution as described in Section 1.3. I set my problem scope to instance segmentation for densely overlapped leaves in low resolution UAV images as stated in 5.1. In my team, we typically estimate phenotypic traits using images acquired by a UAV flying at an altitude of 50 meters. The result of my method on an image from 50 meters altitude is shown in Figure 5.18. However, to examine the performance of my proposed method, I 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 my experiments for this paper, I use images taken from 20 meters altitude to generate ground truth manually. Figure 5.27(b) 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. I use this downsampled data to quantitatively evaluate my proposed method. My data were 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 differently 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.

I compared my method to Mask-RCNN. The training and testing procedure are similar to the work in [81]. My dataset is split into three, 72 for training, 8 for validation, and 8 for testing. I used Resnet-50 Feature Pyramid Network as the backbone instead of Resnet-101 because of my 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, I 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 my proposed method are evaluated on the 8 test images. I report the results using Symmetric Best Dice (SBD), Foreground-Background Dice (FBD) and absolute Difference in Count ($|DiC|$) [172]. These metrics are used in the CVPPP segmentation dataset [173].

Both methods under-counted the leaves. Table 5.1 gives a detailed comparison of the proposed method and Mask R-CNN. Figure 5.27 shows example results from both methods.

Table 5.2.

Segmentation results of the proposed method and the slice-based method reported in Symmetric Best Dice (SBD), Foreground-Background Dice (FBD) and absolute Difference in Count ($|DiC|$)

Metric	Slice-based leaf segmentation	Proposed Method
SBD (higher is better)	24%	38%
FBD (higher is better)	38%	65%
$ DiC $ (lower is better)	85	73

In addition, I compared this method with the slice-based leaf segmentation method described in Section 5.3. Both methods are evaluated on all 88 images in the dataset. Table 5.2 gives a detailed comparison of the proposed method and Slice-based leaf segmentation method. Figure 5.27(e) shows an result of the slice-based leaf segmentation method. The results show that compared to the slice-based segmentation method, the proposed method is able to detect more leaves and provide a more complete leaf segmentation.

5.5.7 Discussions

Compared to Mask R-CNN, my method delivers better results with respect to SBD, which is a primary metric for leaf segmentation. I also out-perform Mask R-CNN for DiC. By visual comparison of the results, my method provides better leaf contours. The leaf edges in my 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.

When segmenting shape and orientation variant leaves, shape modeling produces better results without the requirement for large amount of training samples. Accurate segmentation ground truth is challenging to produce. The ground truth generated by my team is inconsistent in terms of precision and how the details are handled. Often, objects having blurry or ambiguous boundaries cause problems. The ground truthing of those objects relies on the person’s prior knowledge of the shape of an object, as well as their understanding of its interaction with neighbors and the scene. For example, if a leaf tip is small, blurry and has washed-out color, one may not label it. This is a very common problem in segmentation ground truth such as Microsoft COCO dataset [174], where sometimes smaller objects are not labeled and are considered background. This problem can be solve by labeling high resolution images and downsample them for processing. At each resolution, multiple pixel location can be

referred to the same edge, which creates uncertainty in labeling. Image with higher resolution provides higher precision for labeling the location of an edge. By downsampling the image, high precision labels become more accurate in low resolution images, which eliminates the ambiguity in data labels. Shape models described with continuous functions can be reused in different image resolutions, whereas DNN approaches need a large dataset to cover all the resolutions. Such dataset is very expensive to produce. The time to label each image in my dataset ranges from 45 minutes to 6 hours, and the average time is 1.5 hours. Some people put more polygon points than others. Thus, my method is more favorable for the plant scientists because it does not require labeled data.

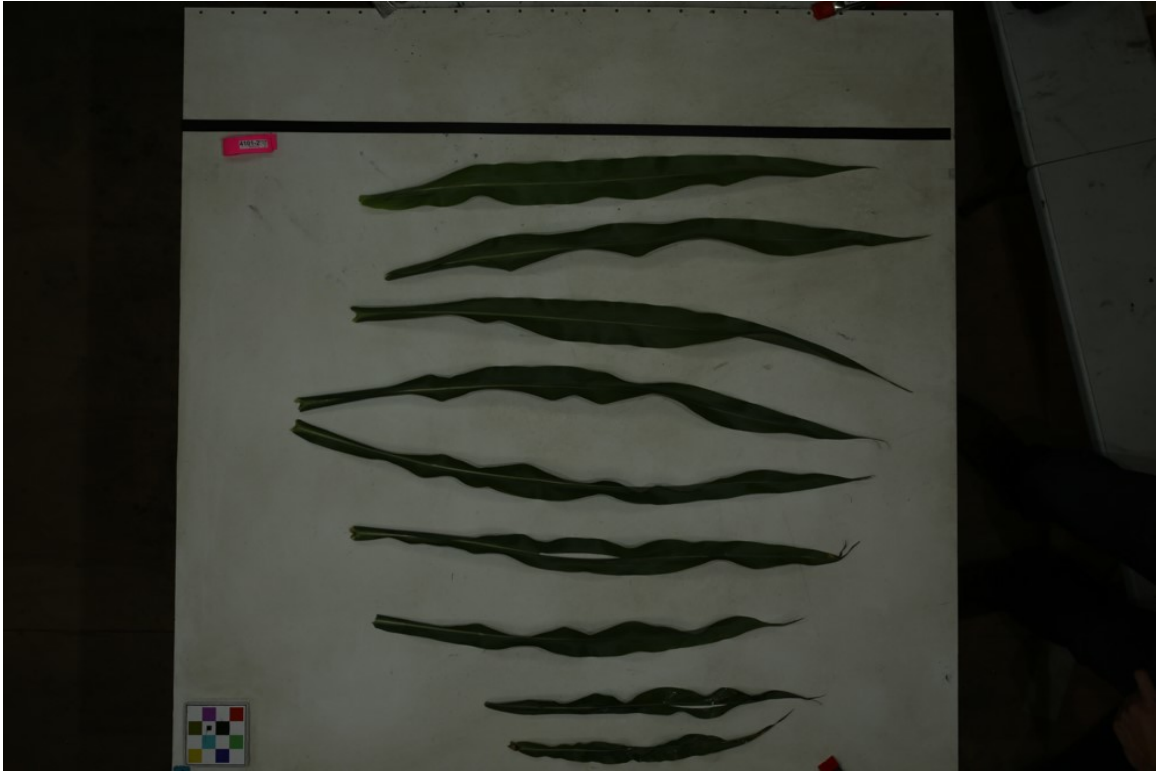


Fig. 5.28. An example taken at ground reference data collection shed

5.6 Midrib Detection for Leaf Length and Width Estimation

During ground reference data collection, plants are manually removed, and leaves are cut from the main stem for taking measurements. Measuring leaf's width and length can be physically challenging as the leaf bends and twists. This process requires a lot of labor work. To speed up this process, imaging tools are used to capture the conditions of plant leaves and analyze them later. Figure 5.28 shows an example of the RGB images taken at the ground reference data collection shed. I present some early work to measure the length and width of a leaf. Figure 5.29 shows the overall blockdiagram for estimating leaf length and width. This tool has been used by plant scientists for trait predictions.

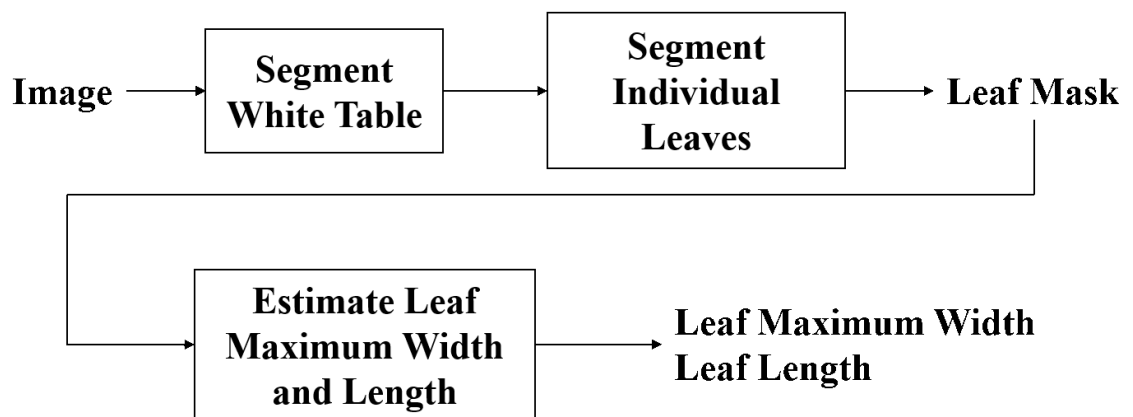


Fig. 5.29. Blockdiagram for estimation leaf length and width

5.6.1 Preprocessing

I preprocess the given image to extract individual leaves so they can be further analyzed for width and length measurements. The leaves are placed on a white table horizontally.

I detect and segment the white table as the region of interest. Figure 5.30 shows the blockdiagram for segmenting the white table in an image. The method includes the following steps:

1. Input image is converted to HSV color space. I apply a threshold of 25 out of the maximum 100 in V channel and creates a table mask. Any pixel location with a V value larger than the threshold is set to value 1 in the mask. Otherwise, the pixel value is set to 0.
2. Connected component [97] is applied to the mask to find sets of connected pixels.
3. The largest component is assumed to be the white table. All other components are discarded.
4. Contours are approximated by using [175].
5. The 4 extreme corner points are retrieved from the contours to be the bounding polygon corners.
6. The white table is segmented using the four polygon coordinates.

Once I have the white table as my region of interest, I segment individual leaves by the following steps. Figure 5.31 shows the blockdiagram for segmenting individual leaves from the table.

1. HSV color segmentation described in Chapter 3 is applied to obtain a mask of leaves.
2. Connected component [97] is applied to the mask to find sets of connected pixels.
3. Small components that have less than 10,000 pixels are removed. These components are noises or false detection in the image.

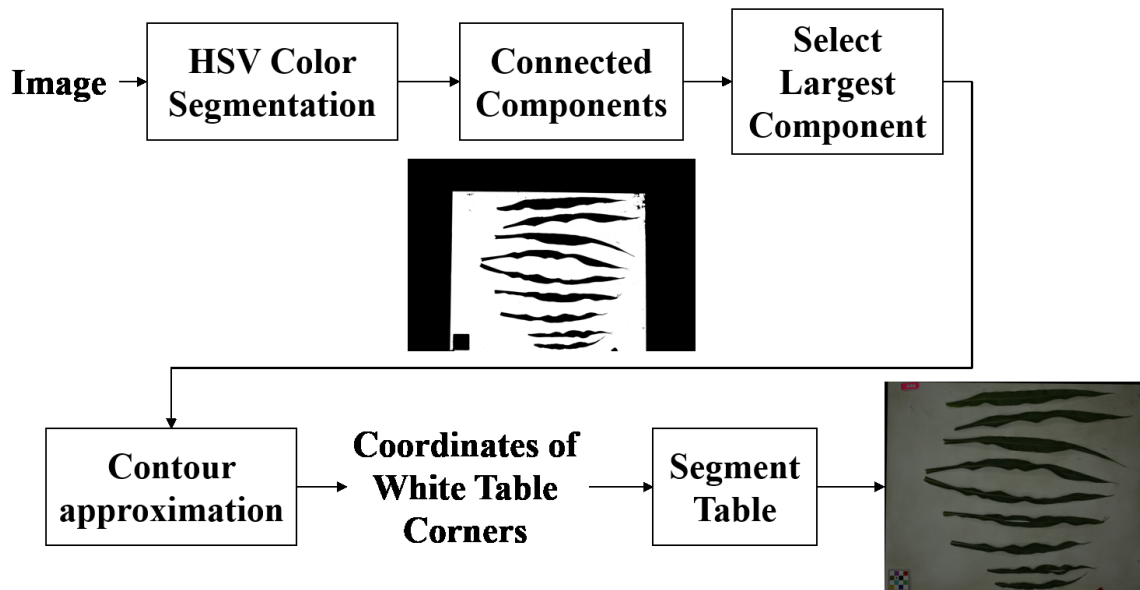


Fig. 5.30. Blockdiagram for segmenting the white table in image

4. Holes in a component that has less than 2,000 pixels are removed.
5. The remaining components are the segmentation of leaves.

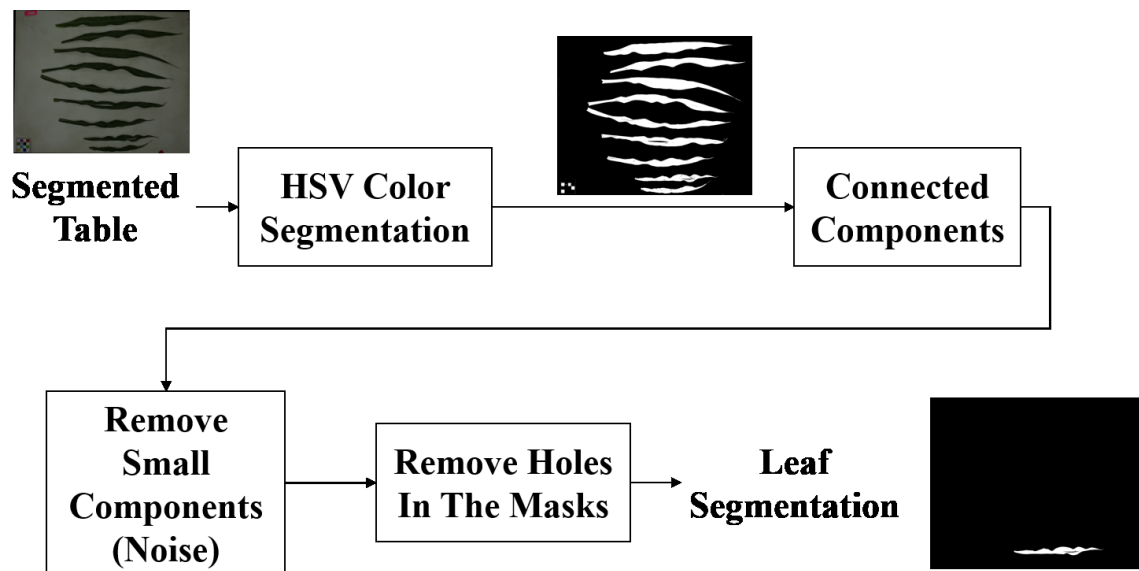


Fig. 5.31. Blockdiagram for segmenting each leaf on the table

5.6.2 Midrib Curve Estimation

I estimate a midrib curve for the segmented leaf. Figure 5.32 shows the block diagram for midrib curve estimation. The procedure contains the following steps.

1. I find contours for the leaf mask by [175].
2. I find the leaf's two end points by finding the extreme horizontal points in the contour.
3. I split the contour point set into two. Each set starts and ends with an end-point. The two sets correspond to the top edge and bottom edge. The set with a higher mean vertical coordinate is the upper edge, and the other set is the lower edge.
4. I estimate the midrib points by averaging upper edge point and lower edge point for each horizontal coordinate x .
5. I apply least square polynomial [167] fit to find a curve representing the midrib points. The polynomial degree is set to be 10. The weights for the midrib points are set to be 1, and the two end points have weights of 10.

5.6.3 Leaf Length and Width Estimation

Let the estimated curve be $y = f(x)$, where (x, y) is the horizontal and vertical coordinate, respectively.

The leaf length L is computed by the following,

$$L = \int \sqrt{1 + \left(\frac{dy}{dx}\right)^2} dx \quad (5.31)$$

To find the width profile of the leaf, I first compute the orthogonal lines of the curve at each midrib point. For each midrib point, I compute the intersections of its orthogonal line with the upper and lower edge. The width of this midrib point is the difference between the intersections. The maximum leaf width is estimated to be the maximum of all the computed leaf width.

Figure 5.33 shows examples of estimated leaf midribs and maximum leaf widths.

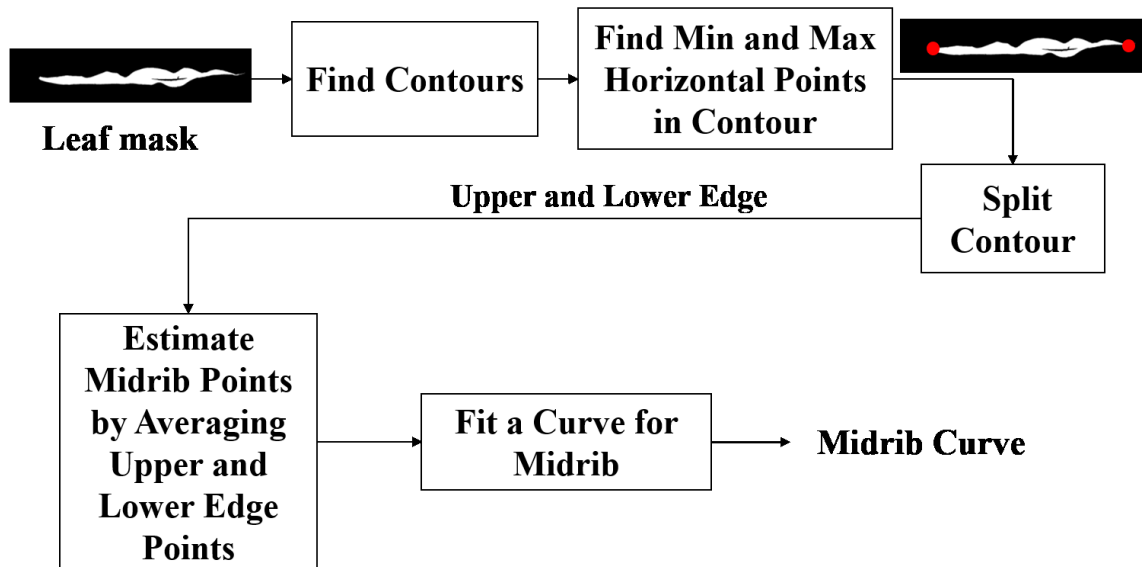
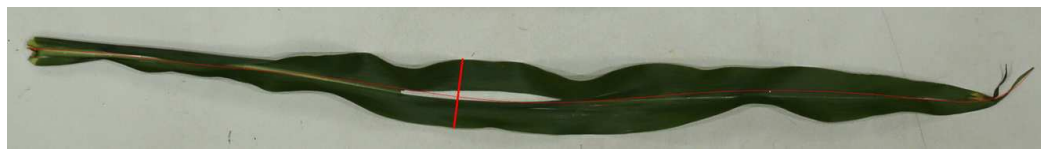


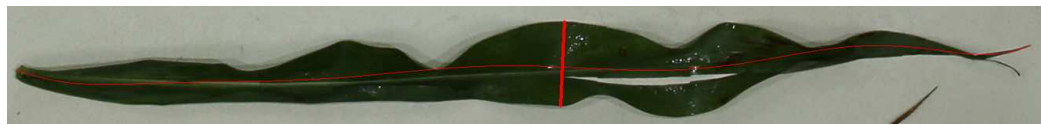
Fig. 5.32. Blockdiagram for estimating the midrib of a leaf



(a)



(b)



(c)

Fig. 5.33. Examples of leaf width and length estimation. Horizontally oriented line indicates the length of the leaf. Vertically oriented line indicates the maximum width of the leaf.

5.7 Midrib Detection for Leaf Angle Estimation

In this work, I describe some initial work to detect the leaf angle for images taken from a vehicle, Phenorover (as shown in Figure 1.4 in Section 1.3. Phenorover is a modified wheel-based sprayer with a sensor boom vehicle. Our data set is collected from one of the high resolution RGB cameras on top of the sensor boom facing forward-down to capture plant height and structures. Figure 5.34(a) shows an example image taken from Phenorover.

Leaf midrib is the backbone of a leaf, and it controls the angle, curvature, and length of the leaf. Each leaf has only one midrib, so counting midribs is effectively the same as counting leaves. In high resolution images, leaf midrib can be described by two parallel curves. This fact satisfies the leaf shape assumption in my "Leaf Segmentation by Functional Modeling", described in Section 5.5. I reuse this method to detect leaf midribs instead of leaves. Figure 5.34(b) shows an example of detect leaf midribs in different color.

The method also outputs two polynomial functions for each midrib. Since midrib is very thin and has constant width, the two polynomial functions are almost identical. I use one of the functions to estimate the leaf angle. I assume plant stems in one row segment in an image aligns with a line. Then the intersection between the line and the midrib polynomial function can be solved analytically. I then compute the gradient angle at the intersection for midrib and the line. The difference in angle is the leaf angle. Figure 5.35 demonstrates the leaf angle estimation. Figure 5.36 shows examples of estimated leaf angle.



(a)



(b)

Fig. 5.34. Examples of Midrib Detection: a) Original image taken from Phenover on 06/18/2018 for Field 57 East in ARCE. b) The detected midribs in different color.

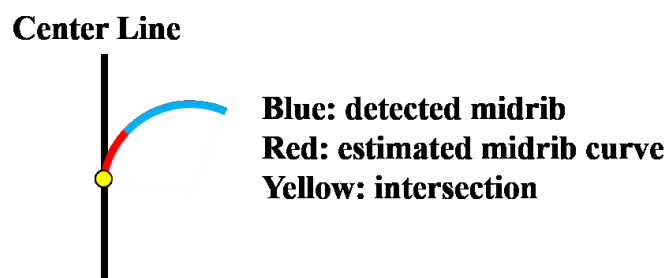
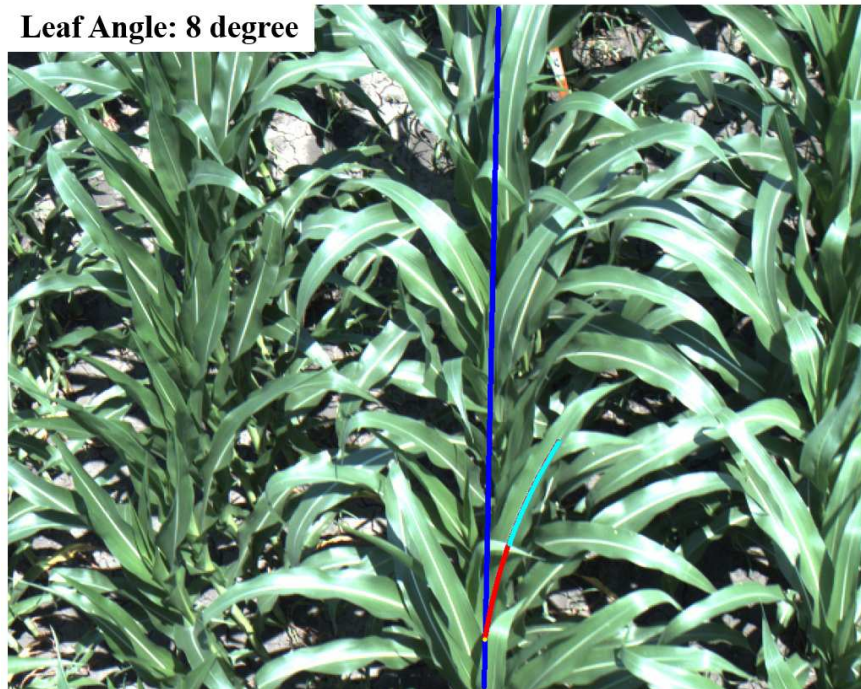
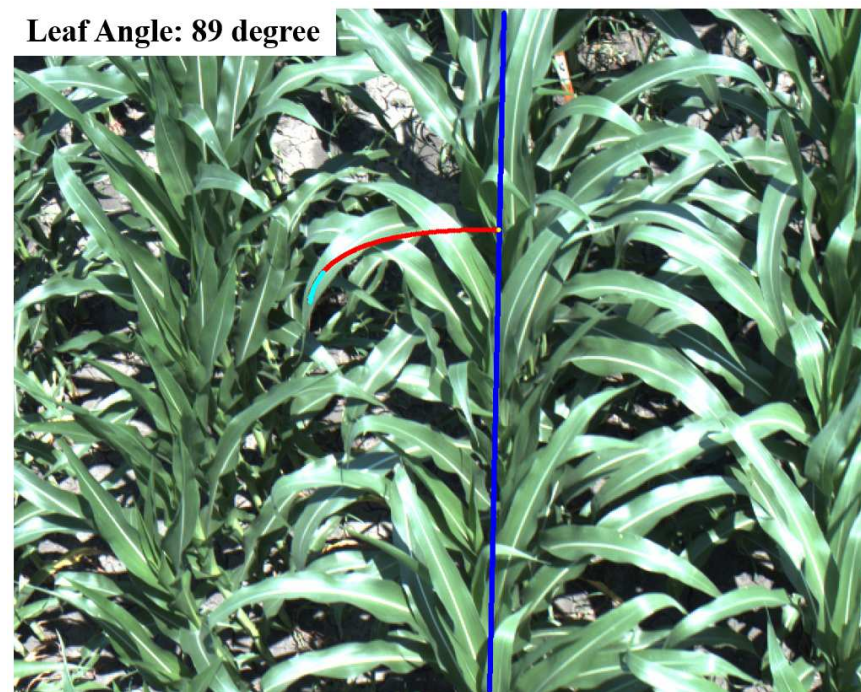


Fig. 5.35. Estimating leaf angle from a detected midrib



(a)



(b)

Fig. 5.36. Examples of estimated leaf angles. Blue: row segment center line. Cyan: detected midrib. Red: extrapolated curve. Yellow: intersection.

6. DEEP LEARNING EXPERIMENTS

6.1 Introduction

Deep learning is a recent trend in object classification, object detection, and segmentation. It out performs traditional computer vision approaches in many areas. In this chapter, I explore and experiment with some modifications to the deep learning framework to address problems in label ambiguity and lack of “AND” operations in deep learning. Experiments failed and potential causes are discussed.

6.2 Training by Comparing

Deep learning is powerful at classifications, but the network does not consider the relationships between samples. Embedding approaches discussed in 5.2 train networks to learn a metric space. In my application, plant centers are difficult to label, and the labeled center may not be precise. Instead of training from the exact location of a plant center, I want to train a network by using the relationship between the plant center and nearby image locations, and train the network to learn a metric space describing the distance to plant centers.

In this work, I explore a framework to learn relationships between samples. The network shares the same architecture concept of Siamese Network [176]. Siamese Network is introduced by Koch for learning with only one sample data per class. The network is used to determine the similarity between samples. I pair-wise train the network on the relationships experimentally, and the results show that the network does not learn the metric space. I discuss the potential reasons for failed results.

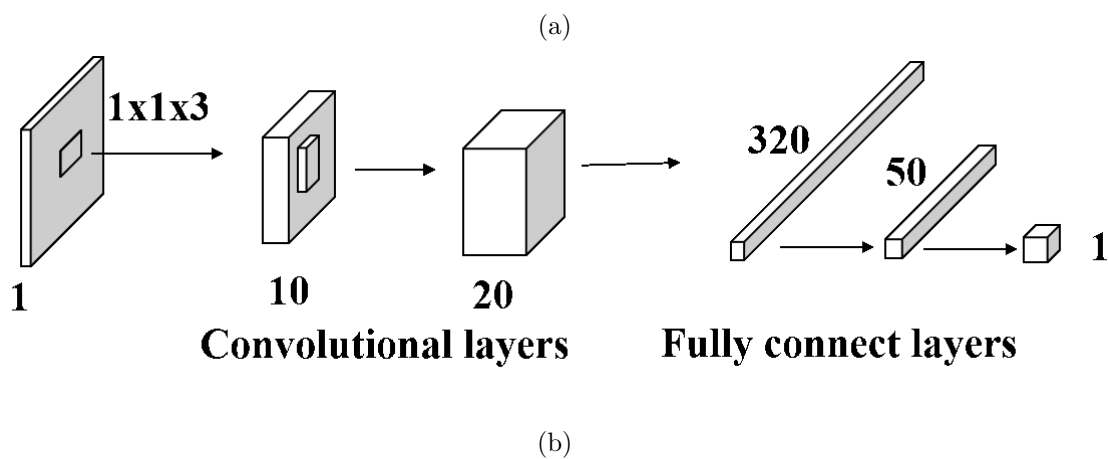
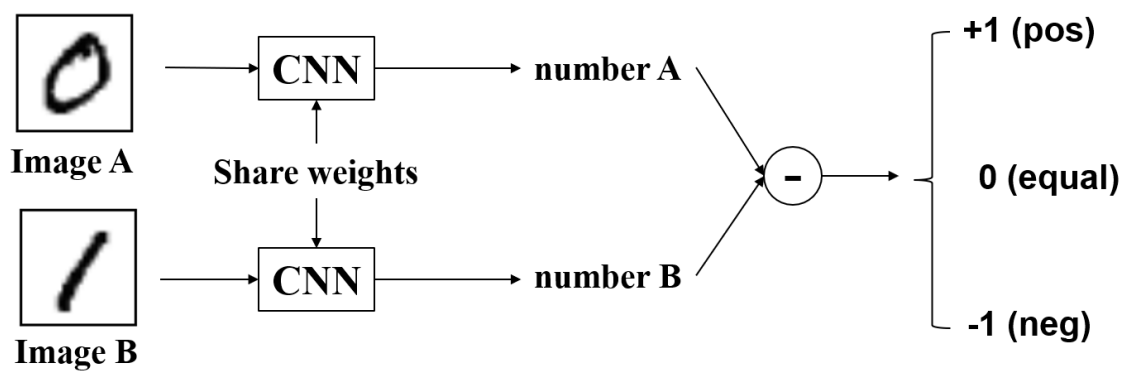


Fig. 6.1. a) Comparison network setup. b) My simple CNN architecture.

6.2.1 Experimental Results

My experiment uses the MNIST dataset [177] which is simple for demonstration. I use a simple CNN architecture that takes an image as input and outputs a number indicating the digit of the image. I simultaneously input two images into two CNN instances. The two CNN share the same weights. My goal is to train only one CNN for classifying digits. The result of the second CNN is subtracted from the first CNN. Finally, the subtracted value is classified into three classes: positive, equal, and negative, with value -1, 0, and 1, respectively. The network is trained with a Mean Square Error loss to capture the relative distance to its class.

I start my experiment with two classes. Figure 6.2 shows the histogram of CNN classification results. Then, I increase the class number to 6 and 10. Figure 6.3 and Figure 6.4 show the histograms of CNN classification results.

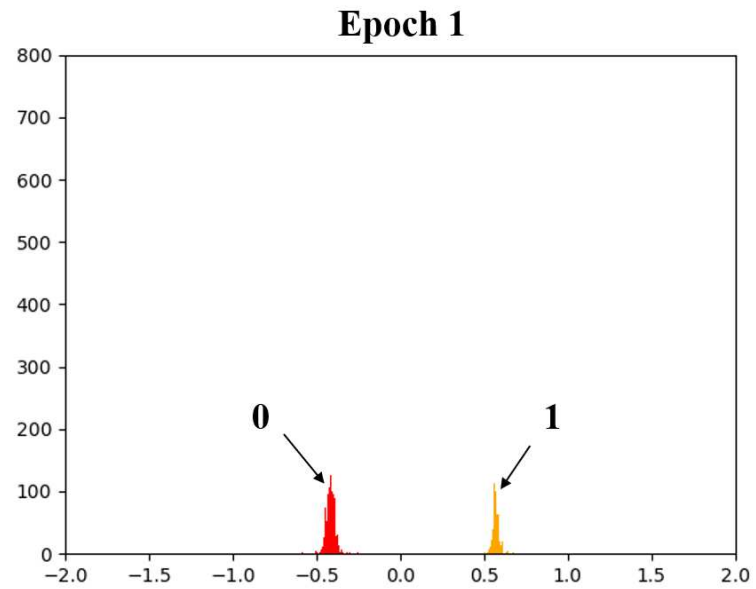
Next, I run my method on 2-dimensional relationship. I define the first dimension to have positive classes for positive subtraction value, and define the second dimension to have negative classes for positive subtraction value. Figure 6.5 shows the scatter plot of results on two digits trained with 2-dimensional relationship.

From the MNIST experiment, I have the following observations:

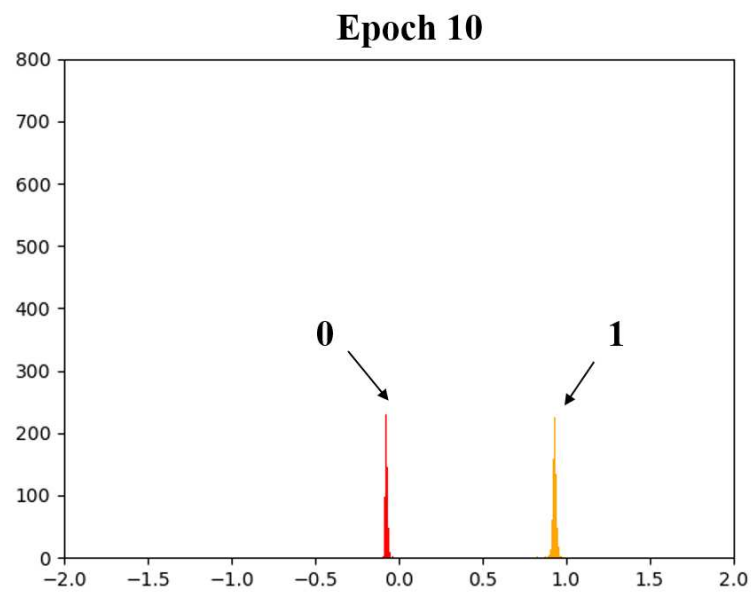
- More training makes the digits evenly distributed, and the distribution of the digits maintains relative distance from each other.
- More training makes the distribution of each digit thinner.
- Experiments with more training classes have difficulties in disentangling the digits.
- The same network is able to produce similar results while I add a dimension to the relationship between samples.
- The distribution for classes with no neighbor on one side is more spread out, since there is no “force” from the neighbor to push the distribution.

Finally, I want to test this idea and observe if the network can find a metric space for plant centers. I assign a smaller class number to locations that are closer to the

plant center. Figure 6.6 shows the class assignments. The classification accuracy for class 0 only achieved 65%. I compared the classification result with the network described in Section 4.5 where the network only has two classes, plant center and not plant center. The two-class network is simpler and achieved 80% classification accuracy, and thus is better than my comparison network.



(a)



(b)

Fig. 6.2. Results for 2 Classes: a) Value distribution at 1st epoch. b) Value distribution at 10th epoch.

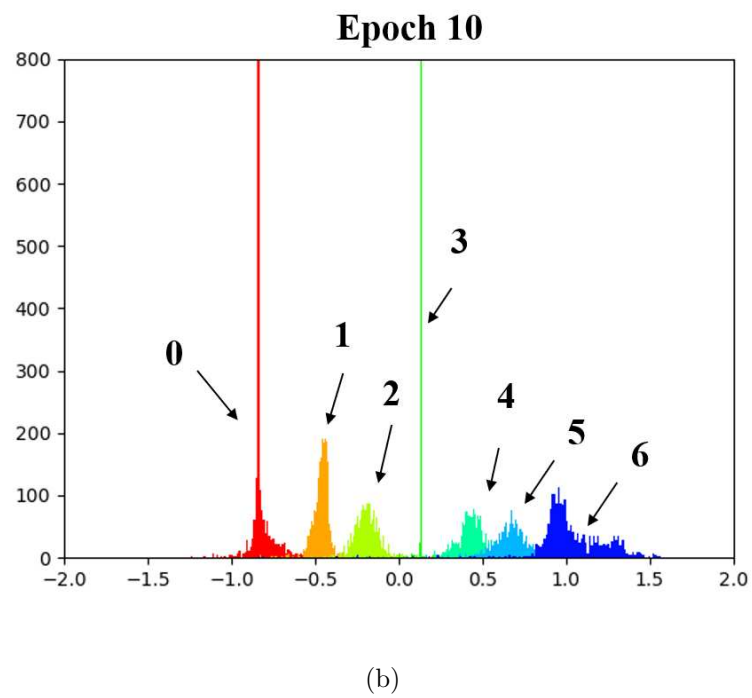
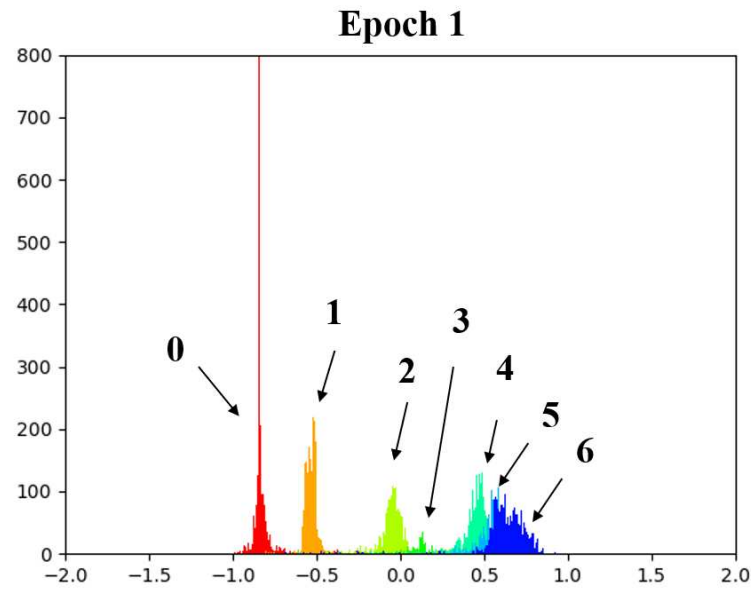


Fig. 6.3. Results for 7 Classes: a) Value distribution at 1st epoch. b) Value distribution at 10th epoch.

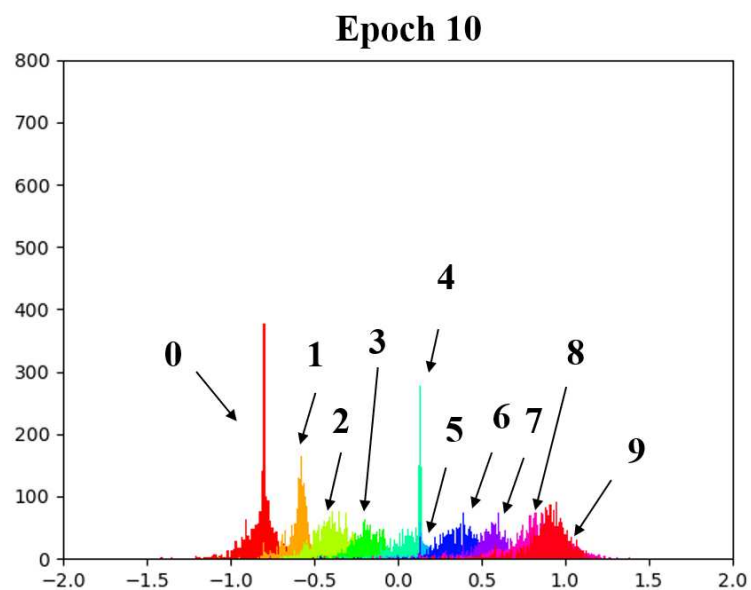


Fig. 6.4. Value distribution at 10th epoch for 10 classes.

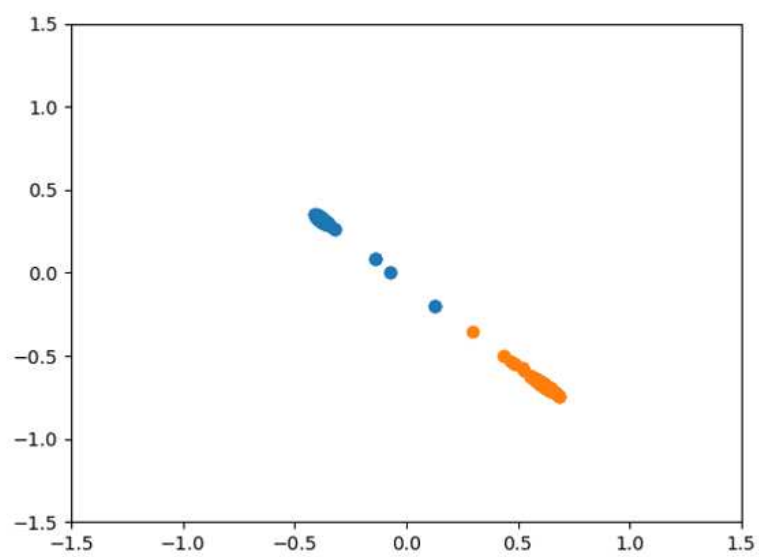


Fig. 6.5. Scatter plot of the two dimension values for 2 classes.

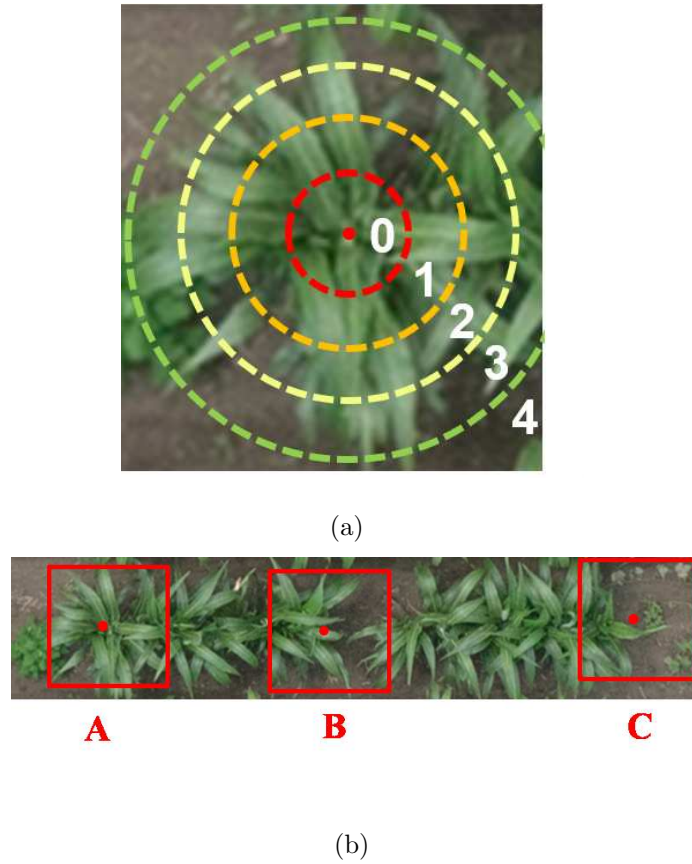


Fig. 6.6. a) Class assignment for training samples. b) An example relationship between samples A, B, and C: $A < B < C$.

6.2.2 Conclusion, and Discussion

From the experiments, we can see that my training framework performs well in two-class case. The more class I add to the network, the more difficult the network can separate them. This is due to my pair-wise training design. More classes mean the training for setting boundaries with classes that are closer in numerical values is diluted, as the network spend more training time on comparing classes that are far away.

When I apply the idea to the plant location dataset, the comparison framework performs much worse than the CNN training with two classes (plant center and not

plant center). This is due to the fact that training with more classes dilutes the training for finding centers, and therefore it is less powerful than the network trained with 2 classes. In addition, CNN architecture does not consider the spatial relationship or spatial shifts between samples, since the network only propagates the filter response while spatial information for these filter responses is discarded during pooling and fully connected layers.

Future work includes learning the boundaries between classes that are closer in numerical values.

6.3 Logical AND Operations with Deep Learning

A typical deep neural network layer has a form of

$$a = f(WX + b), \quad (6.1)$$

where $X = \{x_1, x_2 \dots x_N\}$ is the input containing N samples, W is the network weights, b is the bias, f is the activation function, and a is the activation value. From the equation, we can see that the output depends on the summation of the inputs. This can be interpreted as a logical "OR" gate. Therefore, a neural network can be seen as a giant logical "OR" gate with knobs on each sample to turn the sample on and off. It is natural to think of logical "AND" gate in neural networks. A logical "AND" gate can be computed using "OR" by De Morgan's Laws.

$$Y_1 \text{ and } Y_2 = \text{not} ((\text{not } Y_1) \text{ or } (\text{not } Y_2)) \quad (6.2)$$

Each "NOT" operation can be viewed as the weights of a layer in the network. Then, the simulated "AND" gate requires 2 layers of weights where regular "AND" gate only requires 1 layer of weights. The regular "AND" operation allows the two inputs to be combined without training, while the two-layer "OR" approach requires an additional layer and additional training. Therefore, having "AND" operation is the key to modularization. Moreover, "AND" operation is ideal for reconstruction. For example, if a result is 1, then all inputs must be 1.

In Boolean algebra, summation represents "OR", and multiplication represents "AND". I change all the summation to multiplications and all the multiplication to summations in a DNN. The layer activation now becomes:

$$a = f(b \prod_{i=1}^N (x_i + w_i)) \quad (6.3)$$

$$a = f(e^{\sum_{i=1}^N \log(x_i + w_i) + \log(b)}) \quad (6.4)$$

Since the term $x_i + w_i$ can be negative and the \log of a negative number is undefined, I add an activation function to the term. I also add a constant 1 to offset the minimum value for this term to 0 in \log .

$$a = f(e^{\sum_{i=1}^N \log(\text{Relu}(x_i + w_i) + 1) + \log(b)}) \quad (6.5)$$

I tested the method on MNIST dataset with a simple CNN. The result did not converge. The output values of the network go to either infinity or zero.

6.3.1 Conclusion and Discussion

From the experiment above, I discovered that it is challenging to train a network with logical "AND" for object recognition.

I have the following observations from the experiment.

- If one input is 0, then the output is 0.
- In the case of 10 inputs, if all of the inputs have a probability of 0.9, the output is $0.9^{10} = 0.35$. The result is less than a probability of 0.5 and will be classified as false.

The training of the weights in "AND" operation on one input depends on other inputs. Therefore, training is very unstable, and the results can be easily affected by noises.

For "AND" operations, an output of true value means every input is true. Partial detection indicates the presence of the whole. Therefore, an "AND" operation can be viewed as a model, a template, a framework, or a decoder. It defines structures and boundaries to data representation instead of the flexibility provided "OR" operations.

Finding the model/template/framework/decoder requires considering the texture and spatial accuracy as well as spatial continuity.

As deep learning approaches focus on texture information, I will investigate loss functions that consider spatial accuracy such as active contour approaches. Computers contain millions of connecting “AND” and “OR” operations. In the future, I will also investigate the use of multiple detections (“OR” operations) and reconstruction (“AND” operations) sub-networks in deep learning. Besides deep learning approaches, I will investigate traditional approaches on model fitting that require try and error instead of backpropagation, similar to finding kernels for Support Vector Machine (SVM) [178].

7. SUMMARY AND FUTURE WORKS

7.1 Summary

In this thesis, I developed multiple methods to estimate high precision phenotypic traits from UAV, Phenorover, and ground reference images including plant locations, the number of plants per plot, leaf area, canopy cover, LAI, leaf length, leaf width, leaf count, and leaf angle. The main contributions of this work are:

- I described an image phenotyping system that takes data collected from UAV and estimate phenotypic traits per-field, per-plot, per-row-segment, and per-plant. The estimated traits are used by plant scientists for plant biomass prediction. In addition, several tools are created to facilitate data flows in the system. These tools include orthophoto rotation estimation, plot extraction, and inverse-mapping. A distributed computing system with a web interface is developed. The system contains image phenotyping tools for plant scientists to use and visualize the results. The tools include plot extraction, plant location, and leaf segmentation. The web interface makes my tools accessible and easy to use.
- I presented a method to find the centers of sorghum plants from UAV imagery. Pixel locations are classified as plant center or not plant center. Areas of possible plant centers are estimated by MIL. Then, the areas are merged based on their centroid distance. Finally, I estimate the center of the plants to be the centroids of each merged area.
- I described a method to locate the center of plants from UAV images. Each pixel is classified into a plant center or non-plant-center using a Convolutional Neural Network. The plant centers are finally estimated from the classification mask by using connected components and morphological operations.

- I proposed a method to segment image leaves by a shape-based approach. Leaf slices are detected by Stroke Width Transform [159]. Slices are connected and combined into individually segmented leaves.
- I presented a method to segment leaves of sorghum plants from UAV imagery. Images of plants were converted into polar coordinates, with the origin of coordinates being the center of the plants. Then, a mask indicating plant material was obtained. All segmented leaves were reconstructed and modeled with a set of predefined shape models. By analyzing the shape model matched with each observed leaf, I estimated the phenotypic traits of individual leaves.
- I combined and improved the slice-based approach and the shape model based approach to segment leaf instances in UAV images by modeling leaf shapes with polynomial functions. 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 my approach achieving better results. The estimated leaf shape functions 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.
- I proposed a method to estimate leaf length and width for images that contain leaves lying on the table. The method estimates the leaf midrib and represents it with a polynomial curve. Leaf length and width are measured based on the curve. The results from this method are used by plant scientist for traits estimation.
- I described a method to detect the leaf angle for images taken from a ground vehicle. The method makes use of my "Leaf Segmentation by Functional Modeling" method to segment leaf midrib instead of leaves. The estimated polynomials for midribs are used to find their intersections with the stems. The angle between the midrib and the stem at the intersection is the leaf angle.

- I experimented with training a DNN by comparing samples. The training framework works for MNIST data but failed on classifying plant centers. Results are discussed, and potential causes of failure are analyzed.
- I explored the use of logical “AND” operation in DNN, and I discovered that it is very difficult to train a network with logical “AND” for object recognition. Results are discussed, and potential causes of failure are analyzed.

7.2 Future Works

The experiments done in this work have given us understandings and insights on image-based plant phenotyping, including leaf segmentation and plant location problems. I can improve and extend my work further by the following:

- Plant Phenotyping System
 - I proposed an inverse-mapping tool that maps plot bounding boxes to original images that are used for orthophoto generation. I will improve the tool to map the plot bounding boxes to any images with geo-coordinates and rotation information, so that the tool can be used for precise data analytic on different data resolution and different data types such as hyperspectral data and thermal data.
 - My current plant-level analysis are computed from the plot-level traits. For example, leaves per plant are computed by the total number of leaves in the plot divided by the number of plants. Plants in a plot compete and interact with each other. There are no clear boundaries between plants that can be used to separate them. Therefore, data points such as pixel intensities can be assigned to the wrong plant for analysis. Future work for plant-level studies will focus on data assignments using clustering algorithms.
 - My current methods focus on traits estimation for each date independently. Future work includes analyzing traits using temporal information.

- Future work for our distributed system with web interface includes making the system simpler to use. The user will be able to select plots in the map view. Once the plot is selected, the user will be able to select data types (RGB, hyperspectral, thermal, or LIDAR) and resolution to process. I will also add more image analysis tools to the system.

- Plant material segmentation

I proposed a thresholding method for plant material segmentation. In the future, I will look into learning-based methods such as decision trees [179] and deep learning.

- Leaf segmentation

I described three leaf segmentation methods based on the shape of the leaves. The third method, "Leaf Segmentation by Functional Modeling" combines and advances ideas from the two other methods, and it segments leaves by representing leaf shapes with piece-wise polynomials. However, piece-wise polynomials are limited when representing smooth curves. The joints of polynomials can be non-differentiable, since the continuity in differentiation is not considered at joints when estimating each polynomial piece of the function. To solve this problem, I can use spline curves [161]. Spline curve is a polynomial representation defined by control points and degree of continuity. A high degree spline curve considers both local and global curvatures. Therefore, it is an ideal representation of leaves. To find the splines for leaf shapes, I can use a learning-based method with the ground truth polygon. However, the spline representation of a set of points is not unique. For example, a point set forming a single line can be represented by splines with any amount of control points on the line. This will make learning methods difficult to converge. To solve this issue, I need to abstract the ground truth polygons into splines with a fixed amount of control points and the degree of continuity. Ideally, I want to design a training procedure that can abstract both ground truth polygons (contains no noise) and input image features (contains noise). Once the abstract repre-

sensation of a leaf is generated from the input, I need to recover or reconstruct textures that are lost from the abstraction. A GAN [151] approach can be used in this process. In addition to the spline representation, future work also 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 [173]. Furthermore, surface features that are not used in my current methods will be combined with the shape feature in my leaf model.

- Plant location

I explored two learning base methods (MIL and CNN) that classify pixel locations into plant center or not plant center. The estimated probability maps for the plant centers are post-processed by merging centroids or morphological operations. To improve the post-processing step, I can use borrow ideas from deep learning literature, such as [84, 91], that studies this Non-Maximum Suppression problem. Another issue with my current methods is the imbalance of training data. There is only one positive sample for plant center ground truth, while all other image locations are the negative samples. My current methods randomly sampled the non-center locations as negative samples during training. Therefore, the full potential of the training dataset is not utilized because of the class imbalance. I will address this issue in the future by adding distance-dependent weights to the training samples.

Plant centers can be defined as the intersection of plant leaves. In mature plant datasets where my leaf segmentation can be applied, I can use the estimated leaf shape functions and orientation from my leaf segmentation methods to estimate the intersection of the leaves. The additional information on intersections can improve my existing plant location method.

7.3 Publications Resulting From This Work

1. **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. **Y. Chen**, J. Ribera, C. Boomsma, and E. J. Delp, “Locating Crop Plant Centers From UAV-Based RGB Imagery“, *Proceedings of the IEEE International Conference on Computer Vision, Workshop on Computer Vision Problems in Plant Phenotyping*, October 2017, Venice, Italy.
3. **Y. Chen**, J. Ribera, C. Boomsma, and E. J. Delp, “Estimating Plant Centers Using A Deep Binary Classifier“, *Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation*, April 2018, Las Vegas, Nevada
4. **Y. Chen**, S. Baireddy, E. Cai, C. Yang, and E. J. Delp, “Leaf Segmentation by Functional Modeling“, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, June 2019, Long Beach, CA
5. J. Ribera, D. Guera, **Y. Chen**, and E. J. Delp, “Locating Objects Without Bounding Boxes“, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2019, Long Beach, CA
6. J. Ribera, **Y. Chen**, C. Boomsma, and E. J. Delp, “Counting Plants Using Deep Learning“, *Proceedings of the IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, November 2017, Montreal, Canada
7. J. Ribera, F. He, **Y. Chen**, A. F. Habib, and E. J. Delp, “Estimating Phenotypic Traits From UAV Based RGB Imagery“, *ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Workshop on Data Science for Food, Energy, and Water*, August 2016, San Francisco, CA.

VITA

VITA

Yuhao Chen was born in Shanghai, China. In 2015, he received his Bachelor of Science degree in Purdue University, West Lafayette, Indiana, USA. He started his PhD program at Purdue and joined the Video and Image Processing (VIPER) laboratory in 2015. While in the graduate program at Purdue, he worked on projects sponsored by the Advanced Research Projects Agency-Energy (ARPA-E). Mr. Chen has two year of industry experience with Qualcomm. His current research interests include machine and deep learning, image processing, and computer vision. He is a student member of the IEEE and the IEEE Signal Processing Society. He has been a reviewer of the Plant Methods and the Winter Conference on Applications of Computer Vision (WACV).

REFERENCES

REFERENCES

- [1] S. K. Panguluri and A. A. Kumar, “Phenotyping in sorghum,” *Phenotyping for Plant Breeding: Applications of Phenotyping Methods for Crop Improvement*. New York, NY: Springer New York, 2013, vol. 1, pp. 73–110. [Online]. Available: <http://dx.doi.org/10.1007/978-1-4614-8320-5>
- [2] F. Fiorani and U. Schurr, “Future scenarios for plant phenotyping,” *Annual Review of Plant Biology*, vol. 64, pp. 267–291, February 2013. [Online]. Available: <https://doi.org/10.1146/annurev-arplant-050312-120137>
- [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.
- [4] 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. [Online]. Available: <http://dx.doi.org/10.1016/j.tplants.2015.10.015>
- [5] D. Tilman, C. Balzer, and B. L. Befort, “Global food demand and the sustainable intensification of agriculture,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 108, no. 50, October 2011.
- [6] R. C. OMalley and J. R. Ecker, “Linking genotype to phenotype using the Arabidopsis unimutant collection,” *The Plant Journal*, vol. 61, no. 6, pp. 928–940, March 2010. [Online]. Available: doi.org/10.1111/j.1365-313X.2010.04119.x.
- [7] D. Weigel and R. Mott, “The 1001 genomes project for Arabidopsis thaliana,” *Genome Biology*, vol. 10, no. 5, p. 107, May 2009. [Online]. Available: doi.org/10.1186/gb-2009-10-5-107
- [8] M. Yano and R. Tuberosa, “Genome studies and molecular genetics from sequence to crops: genomics comes of age,” *Current Opinion in Plant Biology*, vol. 12, no. 2, pp. 103–106, April 2009. [Online]. Available: doi.org/10.1016/j.pbi.2009.01.001
- [9] D. C. Koboldt, K. Steinberg, D. E. Larson, R. K. Wilson, and E. R. Mardis, “The next-generation sequencing revolution and its impact on genomics,” *Cell*, vol. 155, no. 1, pp. 27–38, 2013. [Online]. Available: <http://dx.doi.org/10.1016/j.cell.2013.09.006>
- [10] S. B. Cannon, G. D. May, and S. A. Jackson, “Three sequenced Legume genomes and many crop species: Rich opportunities for translational genomics,” *Plant Physiology*, vol. 151, no. 3, pp. 970–977, November 2009. [Online]. Available: doi.org/10.1104/pp.109.144659

- [11] A. Miyao, Y. Iwasaki, H. Kitano, J. Itoh, M. Maekawa, K. Murata, O. Yatou, Y. Nagato, and H. Hirochika, "A large-scale collection of phenotypic data describing an insertional mutant population to facilitate functional analysis of rice genes," *Plant Molecular Biology*, vol. 63, no. 5, pp. 625–635, March 2007. [Online]. Available: <http://dx.doi.org/10.1007/s11103-006-9118-7>
- [12] J. W. White, P. Andrade-Sanchez, M. A. Gore, K. F. Bronson, T. A. Coffelt, M. M. Conley, K. A. Feldmann, A. N. French, J. T. Heun, D. J. Hunsaker, M. A. Jenks, B. A. Kimball, R. L. Roth, R. J. Strand, K. R. Thorp, G. W. Wall, and G. Wang, "Field-based phenomics for plant genetics research," *Field Crops Research*, vol. 133, pp. 101–112, July 2012. [Online]. Available: <http://dx.doi.org/10.1016/j.fcr.2012.04.003>
- [13] J. L. Araus and J. E. Cairns, "Field high-throughput phenotyping: the new crop breeding frontier," *Trends in Plant Science*, vol. 19, no. 1, pp. 52–61, 2014. [Online]. Available: <http://dx.doi.org/10.1016/j.tplants.2013.09.008>
- [14] N. R. Council, "Sorghum," *Lost Crops of Africa. Volume I: Grains*. Washington, DC: The National Academies Press, 1996, vol. 1, pp. 127–144. [Online]. Available: <https://doi.org/10.17226/2305>
- [15] R. L. Monk, F. R. Miller, and G. G. McBee, "2nd southern biomass energy research conference sorghum improvement for energy production," *Biomass*, vol. 6, pp. 145 – 153, 1984. [Online]. Available: [http://dx.doi.org/10.1016/0144-4565\(84\)90017-9](http://dx.doi.org/10.1016/0144-4565(84)90017-9)
- [16] A. Almodares and M. R. Hadi, "Production of bioethanol from sweet sorghum: A review," *African Journal of Agricultural Research*, vol. 4, no. 9, pp. 772–780, 2009.
- [17] W. L. Rooney, J. Blumenthal, B. Bean, and J. E. Mullet, "Designing sorghum as a dedicated bioenergy feedstock," *Biofuels, Bioproducts and Biorefining*, vol. 1, pp. 147–157, October 2007. [Online]. Available: <http://dx.doi.org/10.1002/bbb.15>
- [18] J. D. Plessis, *Sorghum production*. Department of Agriculture of Republic of South Africa, 2003. [Online]. Available: http://www.nda.agric.za/docs/Infopaks/FieldCrops_Sorghum.pdf
- [19] R. L. Vanderlip, *How a Sorghum Plant Develops*. Manhattan, KS: Cooperative Extension Service, Kansas State University, 1972.
- [20] G. A. Johnson, G. P. Cofer, S. L. Gewalt, and L. W. Hedlund, "An engineering approach to image-based phenotyping," *Proceedings of the IEEE International Symposium on Biomedical Imaging*, pp. 381–383, July 2002, Washington, DC. [Online]. Available: <http://dx.doi.org/10.1109/ISBI.2002.1029273>
- [21] A. Mutka and R. Bart, "Image-based phenotyping of plant disease symptoms," *Frontiers in Plant Science*, vol. 5, no. 734, January 2015. [Online]. Available: <http://dx.doi.org/10.3389/fpls.2014.00734>
- [22] A. Bucksch, J. Burridge, L. M. York, A. Das, E. Nord, J. S. Weitz, and J. P. Lynch, "Image-based high-throughput field phenotyping of crop roots," *Plant Physiology*, vol. 166, no. 2, pp. 470–486, October 2014. [Online]. Available: <http://dx.doi.org/10.1104/pp.114.243519>

- [23] N. Fahlgren, M. Feldman, M. A. Gehan, C. S. M. S. Wilson, D. W. Bryant, S. T. Hill, C. J. McEntee, S. N. Warnasooriya, I. Kumar, T. Ficor, S. Turnipseed, K. B. Gilbert, T. P. Brutnell, J. C. Carrington, T. C. Mockler, and I. Baxter, “A versatile phenotyping system and analytics platform reveals diverse temporal responses to water availability in *Setaria*,” *Molecular Plant*, vol. 8, no. 10, June 2015. [Online]. Available: <http://dx.doi.org/10.1016/j.molp.2015.06.005>
- [24] P. Chaudhary, S. Godara, A. N. Cheeran, and A. K. Chaudhari, “Fast and accurate method for leaf area measurement,” *International Journal of Computer Applications*, vol. 49, pp. 22–25, July 2012.
- [25] C. Lü, H. Ren, Y. Zhang, and Y. Shen, “Leaf area measurement based on image processing,” *Proceedings of the International Conference on Measuring Technology and Mechatronics Automation*, pp. 580–582, March 2010. [Online]. Available: <https://doi.org/10.1109/ICMTMA.2010.141>
- [26] M. Minervini, M. M. Abdelsamea, and S. A. Tsafaris, “Image-based plant phenotyping with incremental learning and active contours,” *Ecological Informatics*, vol. 23, pp. 35–48, September 2014. [Online]. Available: <http://dx.doi.org/10.1016/j.ecoinf.2013.07.004>
- [27] R. F. McCormick, S. K. Truong, and J. E. Mullet, “3D sorghum reconstructions from depth images identify QTL regulating shoot architecture,” *Plant Physiology*, vol. 172, no. 2, pp. 823–834, October 2016. [Online]. Available: <http://dx.doi.org/10.1104/pp.16.00948>
- [28] J. A. Gibbs, M. Pound, A. French, D. Wells, E. Murchie, and T. Pridmore, “Active vision and surface reconstruction for 3d plant shoot modelling,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, April 2019.
- [29] F. Apelt, D. Breuer, Z. Nikoloski, M. Stitt, and F. Kragler, “Phytotyping 4d: a light-field imaging system for non-invasive and accurate monitoring of spatio-temporal plant growth,” *The Plant Journal*, vol. 82, no. 4, pp. 693–706, April 2015. [Online]. Available: <http://dx.doi.org/10.1111/tpj.12833>
- [30] M. G. S. Fernandez, Y. Bao, L. Tang, and P. S. Schnable, “A high-throughput, field-based phenotyping technology for tall biomass crops,” *Plant Physiology*, vol. 174, no. 4, pp. 2008–2022, 2017.
- [31] N. Higgs, B. Leyeza, J. Ubbens, J. Kocur, W. v. d. Kamp, T. Cory, C. Eynck, S. Vail, M. Eramian, and I. Stavness, “Protractor: A lightweight ground imaging and analysis system for early-season field phenotyping,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, June 2019, Long Beach, CA. [Online]. Available: http://openaccess.thecvf.com/content_CVPRW_2019/html/CVPPP/Higgs_ProTractor_A_Lightweight_Ground_Imaging_and_Analysis_System_for_Early-Season_CVPRW_2019_paper.html
- [32] F. A. Vega, F. C. Ramírez, M. P. Saiz, and F. O. Rosúa, “Multi-temporal imaging using an unmanned aerial vehicle for monitoring a sunflower crop,” *Biosystems Engineering*, vol. 132, pp. 19–27, January 2015. [Online]. Available: <http://dx.doi.org/10.1016/j.biosystemseng.2015.01.008>

- [33] J. Liu and E. Pattey, "Retrieval of leaf area index from top-of-canopy digital photography over agricultural crops," *Agricultural and Forest Meteorology*, vol. 150, no. 11, pp. 1485–1490, October 2010.
- [34] 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.
- [35] P. Hu, S. C. Chapman, X. Wang, A. Potgieter, T. Duan, D. Jordan, Y. Guo, and B. Zheng, "Estimation of plant height using a high throughput phenotyping platform based on unmanned aerial vehicle and self-calibration: Example for sorghum breeding," *European Journal of Agronomy*, vol. 95, pp. 24–32, April 2018.
- [36] 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. [Online]. Available: <http://dx.doi.org/10.3390/agronomy4020279>
- [37] W. Guo, B. Zheng, T. Duan, T. Fukatsu, S. Chapman, and S. Ninomiya, "Easypcc: Benchmark datasets and tools for high-throughput measurement of the plant canopy coverage ratio under field conditions," *Sensors*, vol. 17, no. 4, p. 798, April 2017. [Online]. Available: <http://dx.doi.org/10.3390/s17040798>
- [38] S. Sankaran, L. R. Khot, C. Z. Espinoza, S. Jarolmasjed, V. R. Sathuvalli, G. J. Vandemark, P. N. Miklas, A. H. Carter, M. O. Pumphrey, N. R. Knowles, and M. J. Pavsek, "Low-altitude, high-resolution aerial imaging systems for row and field crop phenotyping: A review," *European Journal of Agronomy*, vol. 70, pp. 112 – 123, October 2015. [Online]. Available: <http://dx.doi.org/10.1016/j.eja.2015.07.004>
- [39] R. Sugiura, N. Noguchi, and K. Ishii, "Remote-sensing technology for vegetation monitoring using an unmanned helicopter," *Biosystems Engineering*, vol. 90, no. 4, pp. 369–379, April 2005.
- [40] P. Hu, W. Guo, S. C. Chapman, Y. Guo, and B. Zheng, "Pixel size of aerial imagery constrains the applications of unmanned aerial vehicle in crop breeding," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 154, pp. 1–9, August 2019.
- [41] J. Li, F. Zhang, X. Qian, Y. Zhu, and G. Shen, "Quantification of rice canopy nitrogen balance index with digital imagery from unmanned aerial vehicle," *Remote Sensing Letters*, vol. 6, no. 3, pp. 183–189, March 2015. [Online]. Available: <http://dx.doi.org/10.1080/2150704X.2015.1021934>
- [42] T. Duan, S. C. Chapman, Y. Guo, and B. Zheng, "Dynamic monitoring of ndvi in wheat agronomy and breeding trials using an unmanned aerial vehicle," *Field Crops Research*, vol. 210, pp. 71–80, August 2017.
- [43] A. B. Potgieter, B. George-Jaeggli, S. C. Chapman, K. Laws, L. A. S. Cadavid, J. Wixted, J. Watson, M. Eldridge, D. R. Jordan, and G. L. Hammer, "Multi-spectral imaging from an unmanned aerial vehicle enables the assessment of seasonal leaf area dynamics of sorghum breeding lines," *Frontiers in Plant Science*, vol. 8, p. 1532, September 2017.

- [44] 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. [Online]. Available: <http://dx.doi.org/10.1007/s00138-015-0728-4>
- [45] 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.
- [46] C. Zhou, H. Ye, Z. Xu, J. Hu, X. Shi, S. Hua, J. Yue, and G. Yang, "Estimating maize-leaf coverage in field conditions by applying a machine learning algorithm to UAV remote sensing images," *applied sciences*, vol. 9, no. 11, p. 2389, June 2019.
- [47] T. K. Ho, "Random decision forest," *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 278–282, August 1995, Montreal, Canada.
- [48] A. K. Singh, B. Ganapathysubramanian, S. Sarkar, and A. Singh, "Deep learning for plant stress phenotyping: Trends and future perspectives," *Trends in Plant Science*, vol. 23, no. 10, pp. 883–898, October 2018.
- [49] A. Singh, B. Ganapathysubramanian, A. 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. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1360138515002630>
- [50] J. R. Ubbens and I. Stavness, "Deep plant phenomics: A deep learning platform for complex plant phenotyping tasks," *Frontiers in Plant Science*, vol. 8, p. 1190, 2017.
- [51] T. Alkhudaydi, D. Reynolds, S. Griffiths, J. Zhou, and B. d. l. Iglesia, "An exploration of deep-learning based phenotypic analysis to detect spike regions in field conditions for uk bread wheat," *Plant Phenomics*, vol. 2019, pp. 883–898, May 2019.
- [52] M. P. Pound, J. A. Atkinson, D. M. Wells, T. P. Pridmore, and A. P. French, "Deep learning for multi-task plant phenotyping," *Proceedings of the International Conference on Computer Vision Workshops (ICCVW)*, October 2017, Venice, Italy. [Online]. Available: [10.1109/ICCVW.2017.241](https://arxiv.org/abs/10.1109/ICCVW.2017.241)
- [53] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *arXiv preprint arXiv:1505.04597*, May 2015. [Online]. Available: <http://arxiv.org/abs/1505.04597>
- [54] Y. Toda and F. Okura, "How convolutional neural networks diagnose plant disease," *Plant Phenomics*, vol. 2019, March 2019.
- [55] M. P. Pound, J. A. Atkinson, A. J. Townsend, M. H. Wilson, M. Griffiths, A. S. Jackson, A. Bulat, G. Tzimiropoulos, D. M. Wells, E. H. Murchie, T. P. Pridmore, and A. P. French, "Deep machine learning provides state-of-the-art performance in image-based plant phenotyping," *Gigascience*, vol. 6, no. 10, pp. 1–10, October 2017.

- [56] A. Masjedi, J. Zhao, A. M. Thompson, K. Yang, J. E. Flatt, M. M. Crawford, D. S. Ebert, M. R. Tuinstra, G. Hammer, and S. Chapman, "Sorghum biomass prediction using uav-based remote sensing data and crop model simulation," *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, pp. 7719–7722, July 2018.
- [57] M. A. Gehan, N. Fahlgren, A. Abbasi, J. C. Berry, S. T. Callen, L. Chavez, A. N. Doust, M. J. Feldman, K. B. Gilbert, J. G. Hodge, J. S. Hoyer, A. Lin, S. Liu, C. Lizárraga, A. Lorence, M. Miller, E. Platon, M. Tessman, and T. Sax, "PlantCV v2: Image analysis software for high-throughput plant phenotyping," *PeerJ*, December 2017.
- [58] G. Lobet, X. Draye, and C. Périlleux, "An online database for plant image analysis software tools," *Plant Methods*, vol. 9, no. 38, pp. 1–8, October 2013.
- [59] J. Ribera, Y. Chen, C. Boomsma, and E. J. Delp, "Counting plants using deep learning," *Proceedings of the IEEE Global Conference on Signal and Information Processing*, November 2017, Montreal, Canada.
- [60] "Django: The Web framework for perfectionists with deadlines," <https://www.djangoproject.com/>.
- [61] "Amazon Web Services (AWS) - Cloud Computing Services," <https://aws.amazon.com/>.
- [62] M. K. Agoston, *Computer Graphics and Geometric Modeling*. Springer London, 2005. [Online]. Available: <http://dx.doi.org/10.1007/b138805>
- [63] G. Huang, Z. Liu, K. Q. Weinberger, , and L. van der Maaten, "Densely connected convolutional networks," *arXiv preprint arXiv:1608.06993*, August 2016. [Online]. Available: <http://arxiv.org/abs/1608.06993>
- [64] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, December 2015.
- [65] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, November 1998.
- [66] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Proceedings of Advances in Neural Information Processing Systems*, pp. 1097–1105, December 2012, Lake Tahoe, NV.
- [67] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *Proceedings of European Conference on Computer Vision*, pp. 818–833, September 2014, Zurich, Switzerland.
- [68] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, September 2014. [Online]. Available: <http://arxiv.org/abs/1409.1556>

- [69] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, June 2015, Boston, MA.
- [70] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, June 2016, Las Vegas, NV.
- [71] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, June 2016, Las Vegas, NV.
- [72] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, March 1994.
- [73] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, vol. 9, pp. 249–256, May 2010, Sardinia, Italy.
- [74] 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. [Online]. Available: doi.org/10.1109/TPAMI.2016.2577031
- [75] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *arXiv preprint arXiv:1506.02640*, May 2016. [Online]. Available: <http://arxiv.org/abs/1506.02640>
- [76] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. C. Berg, "SSD: Single shot multibox detector," *Proceedings of the European Conference on Computer Vision*, October 2016, Amsterdam, The Netherlands. [Online]. Available: https://doi.org/10.1007/978-3-319-46448-0_2
- [77] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, Columbus, OH.
- [78] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, September 2013.
- [79] R. Girshick, "Fast R-CNN," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448, December 2015, Santiago, Chile.
- [80] I. Sa, Z. Ge, F. Dayoub, B. Upcroft, T. Perez, and C. McCool, "Deepfruits: A fruit detection system using deep neural networks," *Sensors*, vol. 16, no. 8, pp. 1222–1244, August 2016. [Online]. Available: <https://dx.doi.org/10.3390/s16081222>

- [81] 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.
- [82] J. Redmon and A. Farhadi, “YOLO9000: Better, faster, stronger,” *arXiv preprint arXiv:1612.08242*, December 2016. [Online]. Available: <http://arxiv.org/abs/1612.08242>
- [83] J. Redmon and A. Farhadi, “YOLOv3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, April 2018. [Online]. Available: <http://arxiv.org/abs/1804.02767>
- [84] S. Liu, D. Huang, and Y. Wang, “Adaptive nms: Refining pedestrian detection in a crowd,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6459–6468, June 2019, Long Beach, CA. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Liu_Adaptive_NMS_Refining_Pedestrian_Detection_in_a_Crowd_CVPR_2019_paper.html
- [85] J. Fan, W. Xu, Y. Wu, and Y. Gong, “Human tracking using convolutional neural networks,” *IEEE transactions on Neural Networks*, vol. 21, no. 10, pp. 1610–1623, August 2010. [Online]. Available: doi.org/10.1109/TNN.2010.2066286
- [86] X. Zhou, D. Wang, and P. Krähenbühl, “Objects as points,” *arXiv preprint arXiv:1904.07850*, April 2019. [Online]. Available: <http://arxiv.org/abs/1904.07850>
- [87] H. Law and J. Deng, “CornerNet: Detecting objects as paired keypoints,” *Proceedings of the European Conference on Computer Vision*, September 2018, Munich, Germany. [Online]. Available: https://eccv2018.org/openaccess/content_ECCV_2018/papers/Hei-Law-CornerNet-Detecting-Objects-ECCV_2018-paper.pdf
- [88] X. Zhou, J. Zhuo, and P. Krähenbühl, “Bottom-up object detection by grouping extreme and center points,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2019. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/papers/Zhou_Bottom-Up_Object-Detection_by_Grouping_Extreme_and_Center_Points_CVPR_2019_paper.pdf
- [89] Y. Liu, M. Shi, Q. Zhao, and X. Wang, “Point in, box out: Beyond counting persons in crowds,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6469–6478, June 2019, Long Beach, CA. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Liu_Point_in_Box_Out_Beyond_Counting_Persons_in_Crowds_CVPR_2019_paper.html
- [90] J. T. Barron, “A general and adaptive robust loss function,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4331–4339, June 2019, Long Beach, CA. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Barron-A-General-and-Adaptive-Robust-Loss-Function_CVPR_2019_paper.html

- [91] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” *Proceedings of the International Conference on Computer Vision*, pp. 2980–2988, October 2017, Venice, Italy. [Online]. Available: http://openaccess.thecvf.com/content_iccv_2017/html/Lin_Focal_Loss_for_ICCV_2017_paper.html
- [92] J. D. Vylder, F. Vandenbussche, Y. Hu, W. Philips, and V. D. S. Dominique, “Rosette tracker: An open source image analysis tool for automatic quantification of genotype effects,” *Plant physiology*, vol. 160, no. 3, pp. 1149–1159, November 2012.
- [93] M. V. Giuffrida, M. Minervini, and S. Tsafaris, “Learning to count leaves in rosette plants,” *Proceedings of the Computer Vision Problems in Plant Phenotyping*, pp. 1.1–1.13, September 2015, Swansea, UK. [Online]. Available: <https://dx.doi.org/10.5244/C.29.CVPPP.1>
- [94] O. L. Tessmer, Y. Jiao, J. A. Cruz, D. M. Kramer, and J. Chen, “Functional approach to high-throughput plant growth analysis,” *BioMed Central Systems Biology*, vol. 7, no. 6, p. S17, December 2013.
- [95] J. Ribera, F. He, Y. Chen, A. F. Habib, and E. J. Delp, “Estimating phenotypic traits from UAV based RGB imagery,” *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Workshop on Data Science for Food, Energy, and Water*, August 2016, San Francisco, CA.
- [96] B. T. Kitano, C. C. T. Mendes, A. R. Geus, H. C. Oliveira, and J. R. Souza, “Corn plant counting using deep learning and uav images,” *IEEE Geoscience and Remote Sensing Letters*, pp. 1–5, August 2019. [Online]. Available: <https://doi.org/10.1109/LGRS.2019.2930549>
- [97] J. Hopcroft and R. Tarjan, “Algorithm 447: Efficient algorithms for graph manipulation,” *Communications of the ACM*, pp. 372–378, June 1973.
- [98] S. Ghosal, B. Zheng, S. C. Chapman, A. B. Potgieter, D. R. Jordan, X. Wang, A. K. Singh, A. Singh, M. Hirafuji, S. Ninomiya, B. Ganapathysubramanian, S. Sarkar, and W. Guo, “A weakly supervised deep learning framework for sorghum head detection and counting,” *Plant Phenomics*, vol. 2019. [Online]. Available: <https://doi.org/10.34133/2019/1525874>
- [99] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, July 2017, Honolulu, HI.
- [100] A. Vit, G. Shani, and A. Bar-Hillel, “Length phenotyping with interest point detection,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019, Long Beach, CA. [Online]. Available: http://openaccess.thecvf.com/content_CVPRW_2019/html/CVPPP/Vit_Length_Phenotyping_With_Interest_Point_Detection_CVPRW_2019_paper.html
- [101] C. Liu, X. Weng, and Y. Mu, “Recurrent attentive zooming for joint crowd counting and precise localization,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1217–1226, June 2019, Long Beach, CA. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Liu_Recurrent_Attentive_Zooming_for_Joint_Crowd_Counting_and_Precise_Localization_CVPR_2019_paper.html

- [102] J. Wan, W. Luo, B. Wu, A. B. Chan, and W. Liu, "Residual regression with semantic prior for crowd counting," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4036–4045, June 2019, Long Beach, CA. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Wan_Residual_Regression_With_Semantic_Prior_for_Crowd_Counting_CVPR_2019_paper.html
- [103] M. Zhao, J. Zhang, C. Zhang, and W. Zhang, "Leveraging heterogeneous auxiliary tasks to assist crowd counting," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 12736–12745, June 2019, Long Beach, CA. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Zhao_Leveraging_Heterogeneous_Auxiliary_Tasks_to_Assist_Crowd_Counting_CVPR_2019_paper.html
- [104] Q. Wang, J. Gao, W. Lin, and Y. Yuan, "Learning from synthetic data for crowd counting in the wild," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8198–8207, June 2019, Long Beach, CA. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Wang_Learning_From_Synthetic_Data_for_Crowd_Counting_in_the_Wild_CVPR_2019_paper.html
- [105] L. Shen and L. Bai, "A review on Gabor wavelets for face recognition," *Pattern Analysis and Applications*, vol. 9, no. 2-3, pp. 273–292, August 2006.
- [106] S. Shan, P. Yang, X. Chen, and W. Gao, "Adaboost Gabor Fisher classifier for face recognition," *Proceedings of International Workshop on Analysis and Modeling of Faces and Gestures*, pp. 279–292, October 2005, Beijing, China.
- [107] C. Liu and H. Wechsler, "Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition," *IEEE Transactions on Image Processing*, vol. 11, no. 4, pp. 467–476, April 2002.
- [108] J. G. Daugman, "Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters," *Journal of the Optical Society of America A*, vol. 2, no. 7, pp. 1160–1169, 1985.
- [109] P. Viola, J. C. Platt, C. Zhang, *et al.*, "Multiple instance boosting for object detection," *Proceedings of the Neural Information Processing Systems Conference*, January 2005, Vancouver, Canada.
- [110] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez, "Solving the multiple instance problem with axis-parallel rectangles," *Artificial Intelligence*, vol. 89, no. 1-2, pp. 31–71, January 1997.
- [111] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. I–I, April 2001, Kauai, HI.
- [112] B. Babenko, M. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1619–1632, August 2011.

- [113] Y. Xu, J. Zhu, E. Chang, and Z. Tu, "Multiple clustered instance learning for histopathology cancer image classification, segmentation and clustering," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 964–971, June 2012, Providence, RI.
- [114] P. Dollár, B. Babenko, S. Belongie, P. Perona, and Z. Tu, "Multiple component learning for object detection," *Proceedings of The European Conference on Computer Vision*, pp. 211–224, October 2008, Marseille, France.
- [115] B. Zeisl, C. Leistner, A. Saffari, and H. Bischof, "On-line semi-supervised multiple-instance boosting," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1879–1879, June 2010, San Francisco, CA.
- [116] T. Lee, "Image representation using 2D gabor wavelets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 10, pp. 959–971, October 1996.
- [117] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, June 2006.
- [118] D. M. W. Powers, "Evaluation: from precision, recall and F-factor to ROC, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, December 2011.
- [119] Y. Chen, J. Ribera, C. Boomsma, and E. J. Delp, "Locating crop plant centers from UAV-based RGB imagery," *Proceedings of the IEEE International Conference on Computer Vision, Workshop on Computer Vision Problems in Plant Phenotyping*, October 2017, Venice, Italy.
- [120] M. Sonka, V. Hlavac, and R. Boyle, "Binary dilation and erosion," *Image Processing, Analysis, and Machine Vision*. Stamford, CT: Cengage Learning, 2014, pp. 687–693.
- [121] O. Janssens, J. D. Vylder, J. Aelterman, S. Verstockt, W. Philips, D. V. D. Straeten, S. V. Hoecke, and R. V. Walle, "Leaf segmentation and parallel phenotyping for the analysis of gene networks in plants," *Proceedings of the European Signal Processing Conference*, September 2013, Marrakech, Morocco.
- [122] J. Pape and C. Klukas, "Utilizing machine learning approaches to improve the prediction of leaf counts and individual leaf segmentation of rosette plant images," *Proceedings of the Computer Vision Problems in Plant Phenotyping*, pp. 3.1–3.12, September 2015, Swansea, UK. [Online]. Available: <http://dx.doi.org/10.5244/C.29.CVPPP.3>
- [123] 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.
- [124] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, April 2018. [Online]. Available: <https://doi.org/10.1109/TPAMI.2017.2699184>

- [125] T. Takikawa, D. Acuna, V. Jampani, and S. Fidler, “Gated-scnn: Gated shape cnns for semantic segmentation,” *arXiv preprint arXiv:1907.05740*, July 2019. [Online]. Available: <http://arxiv.org/abs/1907.05740>
- [126] Z. Wei, J. Zhang, L. Liu, F. Zhu, F. Shen, Y. Zhou, S. Liu, Y. Sun, and L. Shao, “Building detail-sensitive semantic segmentation networks with polynomial pooling,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7115–7123, June 2019, Long Beach, CA. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Wei_Building_Detail-Sensitive_Semantic_Segmentation_Networks_With_Polynomial_Pooling_CVPR_2019_paper.html
- [127] V. Kulikov, V. Yurchenko, and V. Lempitsky, “Instance segmentation by deep coloring,” *arXiv preprint arXiv:1807.10007*, July 2018. [Online]. Available: <http://arxiv.org/abs/1807.10007>
- [128] V. Kulikov and V. Lempitsky, “Instance segmentation of biological images using harmonic embeddings,” *arXiv preprint arXiv:1904.05257*, April 2019. [Online]. Available: <http://arxiv.org/abs/1904.05257>
- [129] J. Ahn, S. Cho, and S. Kwak, “Weakly supervised learning of instance segmentation with inter-pixel relations,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2209–2218, June 2019, Long Beach, CA. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Ahn_Weakly_Supervised_Learning_of_Instance_Segmentation_With_Inter-Pixel_Relations_CVPR_2019_paper.html
- [130] Y. Zhu, Y. Zhou, H. Xu, Q. Ye, D. Doermann, and J. Jiao, “Learning instance activation maps for weakly supervised instance segmentation,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3116–3125, June 2019, Long Beach, CA. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Zhu_Learning_Instance_Activation_Maps_for_Weakly_Supervised_Instance_Segmentation_CVPR_2019_paper.html
- [131] D. Ward, P. Moghadam, and N. Hudson, “Deep leaf segmentation using synthetic data,” *arXiv preprint arXiv:1807.10931*, August 2018. [Online]. Available: <http://arxiv.org/abs/1807.10931>
- [132] 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.
- [133] 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.
- [134] A. Kirillov, R. Girshick, K. He, and P. Dollar, “Panoptic feature pyramid networks,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6399–6408, June 2019, Long Beach, CA. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Kirillov_Panoptic_Feature_Pyramid_Networks_CVPR_2019_paper.html

- [135] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollar, “Panoptic segmentation,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9404–9413, June 2019, Long Beach, CA. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Kirillov-Panoptic_Segmentation_CVPR_2019_paper.html
- [136] Y. Xiong, R. Liao, H. Zhao, R. Hu, M. Bai, E. Yumer, and R. Urtasun, “UPSNet: A unified panoptic segmentation network,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8818–8826, June 2019, Long Beach, CA. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Xiong-UPSNet_A_Unified_Panoptic_Segmentation_Network_CVPR_2019_paper.html
- [137] H. Liu, C. Peng, C. Yu, J. Wang, X. Liu, G. Yu, and W. Jiang, “An end-to-end network for panoptic segmentation,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6172–6181, June 2019, Long Beach, CA. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Liu_An-End-To-End_Network_for_Panoptic_Segmentation_CVPR_2019_paper.html
- [138] Y. Li, X. Chen, Z. Zhu, L. Xie, G. Huang, D. Du, and X. Wang, “Attention-guided unified network for panoptic segmentation,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7026–7035, June 2019, Long Beach, CA. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Li-Attention-Guided-Unified-Network_for_Panoptic_Segmentation_CVPR_2019_paper.html
- [139] 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. [Online]. Available: 10.1109/CVPRW.2018.00298
- [140] 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. [Online]. Available: <http://arxiv.org/abs/1606.04797>
- [141] M. Kass, A. Witkin, and D. Terzopoulos, “Snakes: Active contour models,” *International Journal of Computer Vision*, pp. 321–331, 1988.
- [142] E. S. L. Gastal and M. M. Oliveira, “Domain transform for edge-aware image and video processing,” *ACM Transactions on Graphics*, vol. 30, no. 4, 2011.
- [143] V. Caselles, R. Kimmel, and G. Sapiro, “Geodesic active contours,” *International Journal of Computer Vision*, vol. 22, no. 1, pp. 61–79, 1991.
- [144] S. Osher and J. A. Sethian, “Fronts propagating with curvature dependent speed: Algorithms based on hamilton-jacobi formulations,” *Journal of Computational Physics*, vol. 79, no. 1, pp. 12–49, 1988.
- [145] D. Acuna, A. Kar, and S. Fidler, “Devil is in the edges: Learning semantic boundaries from noisy annotations,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2019, Long Beach, CA.

- [146] L. Chen, J. T. Barron, G. Papandreou, K. Murphy, and A. L. Yuille, "Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2016, Las Vegas, NV. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Chen_Semantic_Image_Segmentation_CVPR_2016_paper.pdf
- [147] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *Proceedings of the International Conference on Learning Representations*, April 2017, Toulon, France.
- [148] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," *Proceedings of the Conference on Neural Information Processing Systems*, December 2018, Montreal, Canada .
- [149] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y. Jiang, "Pixel2mesh: Generating 3d mesh models from single rgb images," *Proceedings of the European Conference on Computer Vision*, September 2018, Munich, Germany.
- [150] H. Ling, J. Gao, A. Kar, W. Chen, and S. Fidler, "Fast interactive object annotation with curve-gcn," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2019, Long Beach, CA.
- [151] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Proceedings of the Conference on Neural Information Processing Systems*, pp. 2672–2680, December 2014, Montreal, Canada. [Online]. Available: <https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf>
- [152] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, December 2013. [Online]. Available: <http://arxiv.org/abs/1312.6114>
- [153] A. V. D. Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," *Proceedings of the Conference on Neural Information Processing Systems*, December 2017, Long Beach, CA. [Online]. Available: <https://papers.nips.cc/paper/7210-neural-discrete-representation-learning.pdf>
- [154] A. Razavi, A. V. D. Oord, and O. Vinyals, "Generating diverse high-fidelity images with vq-vae-2," *arXiv preprint arXiv:1906.00446*, June 2019. [Online]. Available: <http://arxiv.org/abs/1906.00446>
- [155] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, November 2014. [Online]. Available: <http://arxiv.org/abs/1411.1784>
- [156] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *arXiv preprint arXiv:1703.10593*, March 2017. [Online]. Available: <http://arxiv.org/abs/1703.10593>

- [157] D. Kuznichov, A. Zvirin, Y. Honen, and R. Kimmel, “Data augmentation for leaf segmentation and counting tasks in rosette plants,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019. [Online]. Available: http://openaccess.thecvf.com/content_CVPRW_2019/papers/CVPPP/Kuznichov_Data_Augmentation_for_Leaf_Segmentation_and_Counting_Tasks_in_Rosette_CVPRW_2019_paper.pdf
- [158] K. Wada, “labelme: Image Polygonal Annotation with Python,” <https://github.com/wkentaro/labelme>, 2016.
- [159] B. Epshtein, E. Ofek, and Y. Wexler, “Detecting text in natural scenes with stroke width transform,” *Computer Vision and Pattern Recognition (CVPR)*, pp. 2963–2970, June 2010, San Francisco, CA. [Online]. Available: <http://dx.doi.org/10.1109/CVPR.2010.5540041>
- [160] N. Otsu, “A threshold selection method from gray-level histograms,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, pp. 62–66, January 1979. [Online]. Available: <https://dx.doi.org/10.1109%2FTSMC.1979.4310076>
- [161] A. Watt, *3D Computer Graphics*, 3rd ed. Boston, MA: Addison Wesley, 1999.
- [162] I. Jolliffe, *Principal Component Analysis*, 2nd ed. New York, NY: Springer, 2002. [Online]. Available: doi.org/10.1007/b98835
- [163] 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. [Online]. Available: doi.org/10.1109/TPAMI.1986.4767851
- [164] E. R. Davies, *Computer and Machine Vision*, 4th ed. Amsterdam, Netherlands: Elsevier, 2012. [Online]. Available: <https://doi.org/10.1016/C2010-0-66926-4>
- [165] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, 3rd Edition*. Cambridge, Massachusetts: The MIT Press, 2009.
- [166] 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.
- [167] S. M. Stigler, “Gauss and the invention of least squares,” *The Annals of Statistics*, pp. 465–474, 1981.
- [168] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960. [Online]. Available: doi.org/10.1115/1.3662552
- [169] P. Thévenaz, T. Blu, and M. Unser, “Interpolation revisited,” *IEEE Transactions on medical imaging*, pp. 739–758, July 2000.
- [170] G. McLachlan, S. Lee, and S. Rathnayake, “Finite mixture models,” *Annual Review of Statistics and Its Application*, vol. 6, no. 1, pp. 355–378, 2019. [Online]. Available: <https://doi.org/10.1146/annurev-statistics-031017-100325>
- [171] J. Serra, *Image Analysis and Mathematical Morphology*. Orlando, FL: Academic Press, Inc., 1983.

- [172] H. Scharr, M. Minervini, A. French, C. Klukas, D. Kramer, X. Liu, I. Luengo, J. Pape, G. Polder, D. Vukadinovic, X. Yin, and S. Tsafaris, “Leaf segmentation in plant phenotyping: a collation study,” *Machine Vision and Applications*, vol. 27, no. 4, pp. 585–606, May 2016. [Online]. Available: doi.org/10.1007/s00138-015-0737-3
- [173] M. Minervini, A. Fischbach, H. Scharr, and S. Tsafaris, “Finely-grained annotated datasets for image-based plant phenotyping,” *Pattern Recognition Letters*, vol. 81, no. 1, pp. 80–89, October 2016. [Online]. Available: doi.org/10.1016/j.patrec.2015.10.013
- [174] T. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, “Microsoft COCO: Common objects in context,” *arXiv preprint arXiv:1405.0312*, May 2014. [Online]. Available: http://arxiv.org/abs/1405.0312
- [175] S. Suzuki and K. Ge, “Topological structural analysis of digitized binary images by border following,” *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, 1985.
- [176] G. Koch, “Siamese neural networks for one-shot image recognition,” Master’s thesis, University of Toronto, Toronto, Canada, 2015.
- [177] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278 – 2324, November 1998.
- [178] A. J. Smola and B. Schölkopf, “A tutorial on support vector regression,” *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004.
- [179] J. R. Quinlan, “Induction of decision trees,” *Machine Learning*, vol. 1, no. 1, pp. 81–106, March 1986.