ACQUISITION OF LEAF POINT CLOUD DATA VIA STEREO IMAGES AND COMPUTER VISION METHODS

By

Daniel Gros

Senior Thesis in Computer Engineering

University of Illinois at Urbana-Champaign

Advisors: Professor Narendra Ahuja & John M. Hart

December 2023

Abstract

UIUC researchers are experimenting with genetic modifications on plants and tracking their growth to measure the effectiveness of these modifications on the plant's ability to survive in drier and hotter climates. Metrics that communicate the quality of growth can be gathered from a leaf's phenotype data, including its width. This research project aims at making extraction of phenotype data easier, as in the case of leaf width, by providing researchers with an accurate and precise model of individual leaves.

A vision-based method using stereo images was developed to acquire a 3D model of a leaf. First a stereo camera was utilized to capture recordings of a scene and generate point clouds. Then, the point clouds were projected onto a 2D representation to run a trained instance segmentation model on them to detect leaves in the 2D representation and predict leaf masks. These masks were used as filters on these 2D representations to acquire leaf point clouds after reprojection to 3D.

This research project and the resulting software produced by it demonstrate an effective workflow for acquiring leaf point clouds with birds-eye-view stereo images. Although the accuracy of some measurements doesn't make the product of this project ready to be shared, the results give empirical evidence that the method utilized is able to go through the process of acquiring leaf models for plants in environments with dense leaf clusters. These products are a step in allowing leaf phenotype measurements to be taken with ease, helping researchers track plant growth and improve their experiments.

Code is publicly available.[1]

Keywords: Instance Segmentation, Sorghum, Point Cloud, Plant Phenotype, 3D to 2D projection

[1] https://github.com/danielgros/Plants2LeafPC

Acknowledgments

I want to thank the following people for the help with my Senior Thesis:

Professor Ahuja for agreeing to be my thesis advisor and sticking with me despite the issues along the way including topic change and completion delay.

John M. Hart for being my mentor throughout the thesis project, helping guide the project and my work, and being a source of encouragement.

Jeremy Ruhter for being my contact in the greenhouse, for working with me to setup the ZED camera, and for being patient with all the issues during setup.

Tracy Johnson for providing space and plants for us to photograph and for being very informative relating to Sorghum characteristics.

Yunxuan Yang for helping with the annotation of images and Aaron Gros for helping with the visualization of point clouds.

Contents

1. Introduction

Crop production is essential in providing food and other resources to society. The growth of plants depends on their genotype, environment, and how they are cared for [4]. Plant scientists continuously assess phenotypic traits as an expression of the plant's genotype to help them generate new genetic variations of crops that show desired traits. Assessing these traits often involves a manual, in-field, time-consuming assessment [20]. In contrast to traditional phenotyping performed manually, vision-based systems have the potential for an objective and automated assessment with high spatial and temporal resolution, providing a method of performing the same assessment at a larger scale, in less time, and more objectively. One of such systems' objectives is to detect and segment individual leaves of each plant since this information correlates to the growth stage and provides phenotypic traits[1]. With growing concerns for the effects of climate change on crop production these studies are becoming increasingly popular and increasingly important. This senior thesis project aims to offer an alternative to in-field assessments by producing a vision-based method of collecting phenotype traits via leaf point clouds.

1.1 Background

The increasing severity of climate change is significantly impacting plant growth. Changing temperature, precipitation quantity and timing, atmospheric carbon dioxide levels, soil moisture, and nutrient availability can affect photosynthesis, respiration, and water use affecting plant growth [2]. As a result plant growth worldwide is at an exponential decline affecting the ability of plants to produce the expected yields. Consequently, research is being conducted in many institutions including the efforts of Professor Leakey at the University of Illinois at Urbana-Champaign to genetically modify plants to be able to withstand these changing environments. These experiments require large amounts of phenotype data to be consistently observed and measured by a team of field data collectors. The manual methods of collecting the phenotype data can slow the research down, limit the scale of the research, and even affect the research negatively via imprecise data recordings and larger measurement errors. As a result, current phenotyping methods are a bottleneck in plant gene research [3].

For the visual components of the recorded phenotype data, computer vision methods are being utilized to facilitate the data collection process. For example, one of the important metrics that communicates

effective growth in plants is leaf size. Measuring leaf width is difficult under field conditions and common practice is to remove, scan, and measure leaves individually [4]. This is a slow and tedious process prone to error. Approaches have been developed that utilize computer vision methods to automate and increase the reliability of the measurements. However, these computer-vision-based approaches remain in their infancy and can be inaccurate when it comes to leaves outside of the laboratory which are in dense clusters, overlap, and can curve and curl. This research project aims at making extraction of phenotype data easier, as in the case of leaf width, by providing researchers with an accurate and precise model of individual leaves.

This research project aims to accomplish this in an accessible and noninvasive manner: without the need for specialized equipment and without the need for a significant number of people to take the measurements by hand. This can be accomplished with the use of stereoscopic images. After performing initial image processing, 2D stereo images can be used to perceive depth the same way our eyes perceive depth. With this perception of depth one can generate depth maps, point clouds, and meshes. Thus, it is possible to digitize the 3D structure of a cluster of leaves with a pair of offset 2D images. The challenging part is isolating individual leaves that can be measured.

1.2 Research Goals

The aim for this research project is to enable researchers who are conducting experiments on plants that require the collection of phenotype data to perform the acquisition of data more easily and quickly.

The research goals are to define a method and develop a program that researchers can use to acquire leaf models for plants in environments with dense leaf clusters.

Questions this research will address are:

What is an effective process at isolating and modeling a leaf from stereo images? This includes,

What is an effective way of setting up a stereo camera for data collection? What is an effective way of capturing recordings with a camera over the course of months? What is an effective way of generating point clouds from stereo image captures? What is an effective way of projecting a point cloud to a 2D representation of itself? What is an effective way of training an instance segmentation model to detect

and segment sorghum leaves? What is an effective way of scaling leaf masks to their appropriate dimensions? And what is an effective way of cropping a point cloud utilizing the appropriate leaf masks?

1.3 Significance

With the growing effects of climate change crop yields are declining and researchers are hard at work developing new breeds of plants to thrive in the new environments. This project intends to provide these researchers with the tools to conduct this research more effectively and efficiently.

Although this project serves as a first step in the automation of leaf phenotype data collection, the defined process and developed program can be altered for other applications that require modeling of objects. This could include other types of research: automating food harvesting, facilitating detection of large tumors, etc. This project can serve as a first step in development in the space of object modeling with stereo images.

1.4 Structure

Chapter 1 has outlined the purpose of the research project and what it entails. It focuses on describing at a high level the problem the project intends to address, the method to address the problem, and the potential applications of the project.

Chapter 2 outlines the survey conducted to determine the most effective design choices. It describes the different methods for performing instance segmentation on leaves and the different methods for measuring object width on 3D models of objects.

Chapter 3 describes the methodology used to accomplish the project goals. This chapter steps through the setup involved for data acquisition, point cloud generation, the process in creating, training, and using the instance segmentation model, and the code written to extract the 3D structure of an individual leaf for future measurement of leaf aspects.

Chapter 4 communicates the level of effectiveness of the research project's results. It defines these results at every stage of the project.

Chapter 5 analyzes the results outlined in chapter 4. It describes the findings of the project and the work that can be done to continue the project and answer further research questions.

Chapter 6 outlines the answers to the research questions and how the research project contributed to the technical space. This final chapter also describes the opportunities for further research utilizing the research from this project and extending its functionality and application.

2. Literature Review

During the establishment of the structure of the project, two areas needed surveying to inform the direction of the project. First was determining the appropriate method for performing instance segmentation on a leaf; surveying 3D and 2D instance segmentation methods and deciding which algorithm within the appropriate category was best suited for the task. Second was an exploration into the methods in which a 3D model of a leaf can be utilized to acquire width measurements. This survey was performed to verify that the acquisition of leaf phenotype data, like width, from a 3D model of a leaf is simple and reliable.

2.1 Instance Segmentation on Leaves

A survey into the current instance segmentation methods for leaves resulted in two common approaches: 3D training and inference on point clouds [3], [5], [6] and the traditional training and inference on 2D images [4], [7], [8].

2.1.1 3D Point-Based Model for Instance Segmentation on Leaves

Current point cloud instance segmentation approaches on leaves involve extending a deep neural network that consumes point cloud data. One of the foundational approaches is called PointNet developed by C Qi, H Su et al. This network provides a unified approach able to perform object classification, part segmentation, and semantic segmentation with point clouds. The architecture is visualized in Figure 1. The classification network takes n points as input, applies input and feature transformations, aggregates point features by max pooling, which results in output scores that are classification scores for k classes. The segmentation network extends the classification network, concatenates global and local features, and outputs per point scores [9].

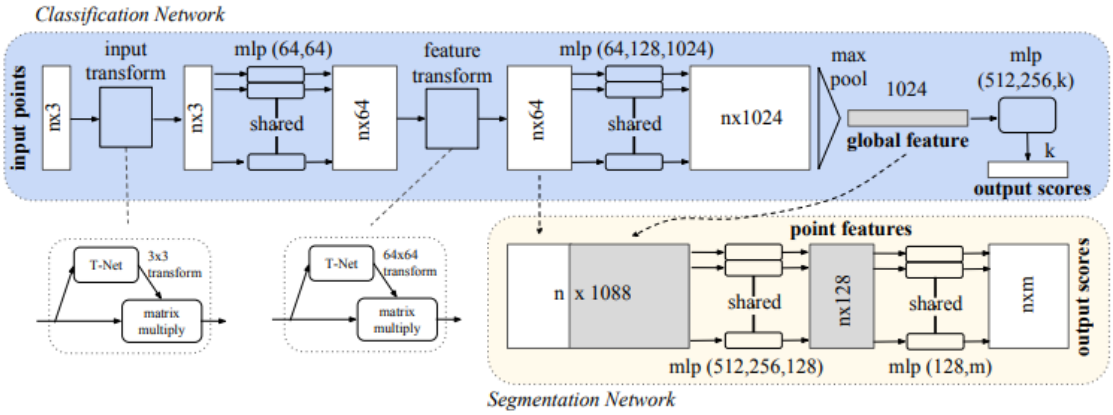

Figure 1. PointNet Architecture [9]

This architecture is applied to plant and leaf instance segmentation in the three following papers. Y. Li et al developed a process of segmenting point clouds with a focus on high-throughput data acquisition before deep learning. The data acquisition was performed using the MVS-Pheno platform to acquire high-throughput data and an internally built labeling tool, Label3DMaize, was used for the labeling. For the deep learning segmentation, Y. Li et al used PointNet to implement stem-leaf and organ instance segmentation achieving mean accuracy scores over 0.91 and f1-scores over 0.85 [3]. D. Li et al used PlantNet and improved upon it with a 3D Edge Preserving Sampling strategy for preprocessing input points, a Local Feature Extraction Operation module based on dynamic graph convolutions, and a semantic-instance Feature Fusion Module. The results outperformed other instance segmentation deep learning networks by over 13.3 percent [5]. K. Turgut et al compared the latest available point cloud segmentation approaches on instance segmentation with rosebush including PointNet and its improved version PointNet++. They utilized a labeled rosebush dataset and a synthetic model. The addition of the synthetic model improved the results of the instance segmentation model [6].

The papers demonstrate the effectiveness of point cloud instance segmentation. However, this effectiveness comes with its conditions. Y. Li et al outline that the great performance of the model is mainly due to a dataset of high quality and diversity [3]. This condition is particularly relevant as Y. Li et al and D. Li et al express how there is a shortage of well-labeled point cloud datasets in the plant domain [3], [5]. Consequently, pursuit of a point-based convolutional neural network model involves the acquisition of a high quality and diverse dataset. Acquisition of high quality point cloud data is not an accessible process and requires specific environmental conditions and tools: no wind disturbances, indoors, even and consistent lighting [3]. Not only is the acquisition of this data much more difficult, the

annotation of this data and procedures for constructing 3D models for training are both much more time demanding and error-prone than the image-based model counterpart [6].

2.1.2 2D Image-Based Model for Instance Segmentation on Leaves

Most current 2D image instance segmentation approaches on leaves involve extending and training the Mask R-CNN model for the purpose of segmenting leaves. K. He et al's Mask R-CNN is self-described as "simple to train and […] easy to generalize to other tasks" and is the reason it's widely used for all types of applications including instance segmentation with leaves. Another improvement of Mask R-CNN compared to its older counterparts is its ability to accurately perform instance segmentation even among overlapping objects.

Mask R-CNN is an extension of Faster R-CNN which is an extension of Fast R-CNN. Fast R-CNN takes as input an image and object proposals. The network processes these with convolutional and max pooling layers that produce a feature map. The feature map is used to calculate softmax probability estimates and four numbers that encode refined bounding-box positions for each of the K object classes [10]. Figure 2 shows the architecture of Fast R-CNN. Faster R-CNN attaches a "Region Proposal Network"(RPN) before the Fast R-CNN architecture which reduces the number of object proposals and quickens the model [11]. Mask R-CNN leaves the RPN intact but adds a parallel process along with the Fast R-CNN architecture. The parallel process outputs a binary mask prediction for each region of interest along Fast R-CNN [12]. Figure 3 shows the architecture of Mask R-CNN.

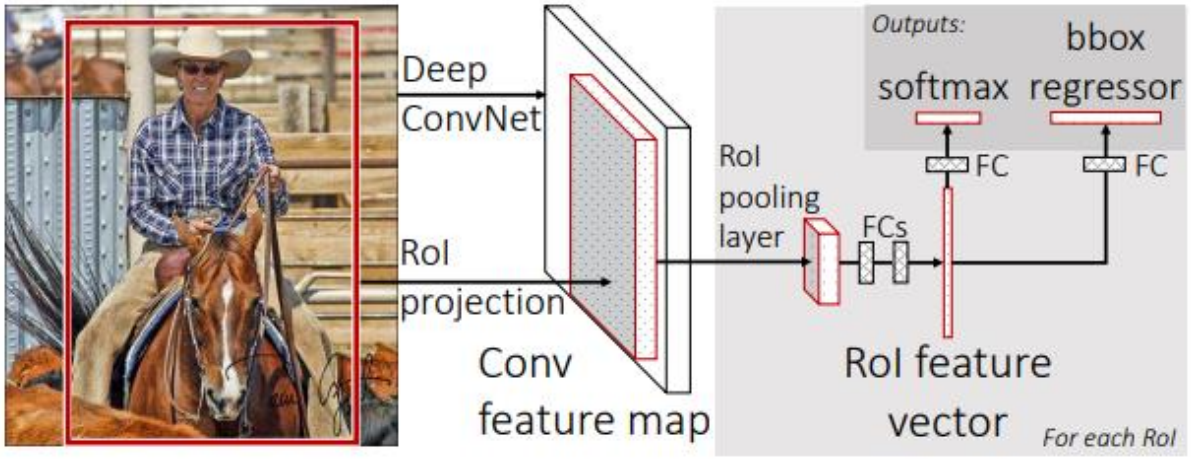

Figure 2. Fast R-CNN Architecture [10]

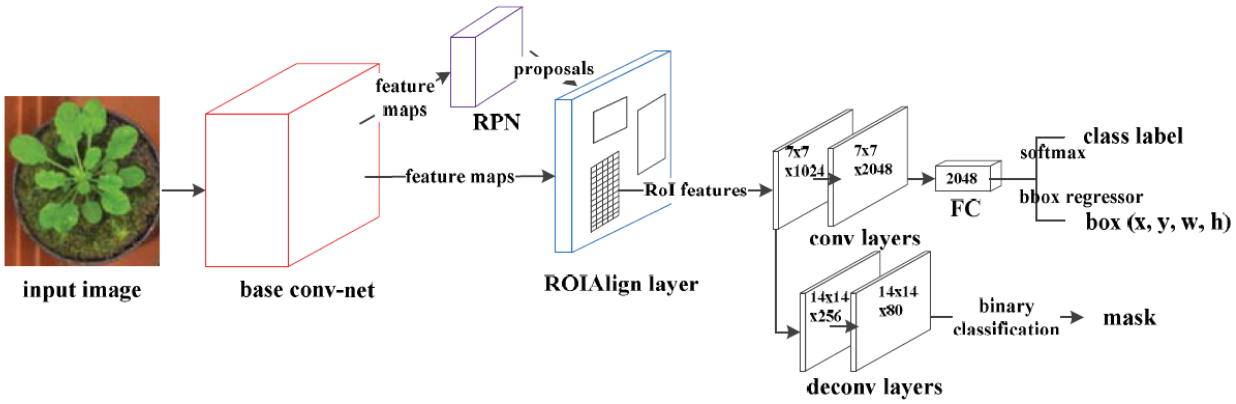

Figure 3. Mask R-CNN Architecture [7]

Altogether, Mask R-CNN includes three functional branches of classification, bounding-box (bbox) regression and mask branches, and the instance segmentation process is as follows. Firstly, input one image into a base convolutional net to obtain the feature maps. Secondly, a Region Proposal Network (RPN) recommends a set of Regions of Interest (ROIs) that may include instances based on the feature maps. Thirdly, realign each ROI feature through a ROIAlign layer. Finally, predict the object category, box location and binary mask (indicates each pixel as target or background) for each ROI in parallel.[7]

This architecture is applied to plant and leaf instance segmentation in the three following papers. L. Xu et al tested different extensions of Mask R-CNN and trained the models to segment and count the number of leaves. The resulting model was able to accurately count leaves, even when overlapping, when feeding the model top view photos of individual round-leafed plants [7]. J. Schrader et al developed an android application that obtains the leaf size via instance segmentation with Mask R-CNN after placing picked leaves on a solid background [4]. J Champ et al extended Mask R-CNN for the fine detection and segmentation of weeds and crops [8].

The papers and application of Mask R-CNN to a plethora of segmentation applications demonstrate its effectiveness for a variety of objects in difficult environments. L Xu et al outlines that the reason for Mask R-CNN's effectiveness in leaf segmentation can be attributed to the RPN network that takes multi-scale anchors for the region of interest propositions, making leaves with different sizes, ocluded leaves, and leaves of different shapes recognizable [7]. Although Mask R-CNN is more accurate under these conditions compared to other models, it can still produce erroneous results with small leaves and heavy occlusion. That said, the model's great effectiveness at segmenting different kinds of leaves with small amounts of data makes it an attractive model to use with leaf segmentation.

2.1.3 Selected Instance Segmentation Approach

Despite the greater accuracy of 3D point-based models that utilize the extra dimension for segmentation, the approach is not feasible for this project as it requires the acquisition of high quality and diverse data with intrusive technologies like LiDAR scanners in heavily controlled environments. The constraints of this project include data acquisition solely obtained through stereoscopic images in environments with dense foliage that cannot be altered. This makes it difficult to develop accurate and reliable 3D point-based models for this project. In addition, one of the project's aims is for the continual application to other research teams with varied environments and plant types, for example using images from row farming as training data. The difficulty in acquisition and annotation of 3D point-based datasets makes that path unappealing. This project will perform the instance segmentation tasks via the 2D image-based model Mask R-CNN. This approach will allow the research team the flexibility to collect data in changing environments, dense foliage, without tampering with the plants, and without the need of excessive time in annotation.

2.2 Leaf Area Measurements on Point Clouds and Meshes

To better understand the workflow that will be used in conjunction with this research project to obtain phenotype data, a survey into the methods of obtaining leaf width using a leaf point cloud was conducted.

Calculations of a distance across a 3D surface irrespective of rigid deformations (commonly occurring in leaves), also known as a geodesic distance, are often calculated using shortest path algorithm due to the graph-like nature of point clouds and meshes. Most existing shortest path algorithms fall under single-source shortest path algorithms and all-pairs shortest path (APSP) algorithms. When dealing with large datasets it's common practice to use single source shortest path algorithms and methods which utilize heuristic and landmarks strategies to minimize the otherwise heavy performance load. Traditionally algorithms like Dijkstra's, Breadth-First-Search, and Bellman Ford have been used [13]. Graph walks as well as other papers outline more effective ways of calculating geodesic distance involving differentiable and more time efficient methods as well as handling noisier data [14]. Due to the smaller nature of the leaf datasets in this project, it would not be inadvisable to use APSP algorithms to assist with the calculation of leaf width.

Conducting this survey has demonstrated methods of calculating phenotype data from a leaf's point cloud or mesh, proving the usefulness of such data.

3. Methodology

The project is split into six parts: data collection, point cloud and image generation, point cloud projection, acquisition of leaf mask, scaling of leaf mask, and point cloud cropping.

The data collection phase involves determining what data to collect, where to collect the data, and how to collect the data. As the project was performed in partnership with a lab working on sorghum genetic modifications, the collected data are images of sorghum plants at different stages of growth. The data was collected in two different greenhouse rooms where genotype experiments were being performed. And the data was collected via a stereo camera called the “ZED” camera by Stereo Labs. The data is collected in a proprietary SVO format that needs to be processed by the ZED SDK to obtain more malleable data that can be used for this project’s purpose.

The generation of the point cloud of the scene is obtained by first extracting the coordinates and color values of all the points from the SVO recordings using the ZED SDK and provided API, and then formatting the data into a data structure that is easy to read and manipulate. The 2D image generation is performed using the provided ZED SDK API. These images are used for validation purposes and for the training dataset.

The point cloud projection is used for training the leaf instance segmentation model, for the point cloud cropping, and re-projection to a point cloud. The projection is performed by correlating the angle of a point, using the normal of the camera lens, to a pixel position in the corresponding 2D left image.

The acquisition of a leaf mask is obtained by training an instance segmentation model to segment leaves with annotated images of sorghum leaves at different stages of its lifecycle, in three different rooms, with three different distances between the camera and the plants.

The scaling of the leaf mask is performed via nearest neighbor interpolation.

The point cloud cropping is performed by using the leaf mask as a filter in the length-width coordinate system of the point cloud projection. The projection is then transformed to a point cloud where the outcome is a point cloud of a leaf.

Figure 4 describes the flow of the research project’s product.

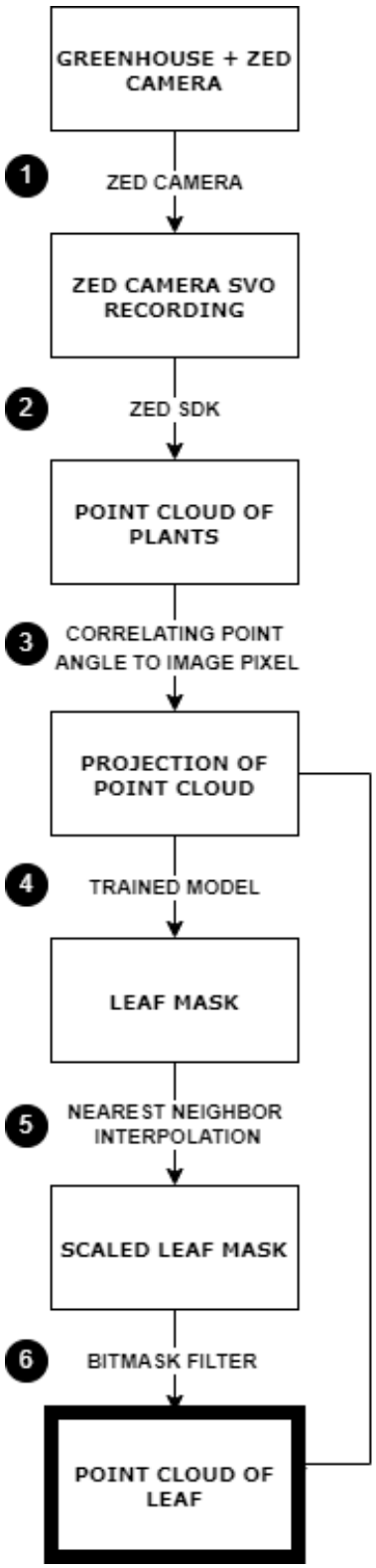

Figure 4. Project Workflow

3.1 Data Collection

3.1.1 Greenhouse Setup

The research project was performed with a focus on its application to the Sorghum plant. The first stage of data collection was performed in a predefined environment selected by the genetic researchers who were partnered with this project. This environment was a room in the University of Illinois Plant Biology Greenhouse filled with different genetic varieties of Sorghum. Figure 5 showcases the room at the start of the data collection process. After 9 weeks the Sorghum plants in the first greenhouse room matured to a point where there were no more significant changes in leaf size. Figure 6 showcases the room at this point.

Figure 5. First Greenhouse Room at Week 0 of Collection

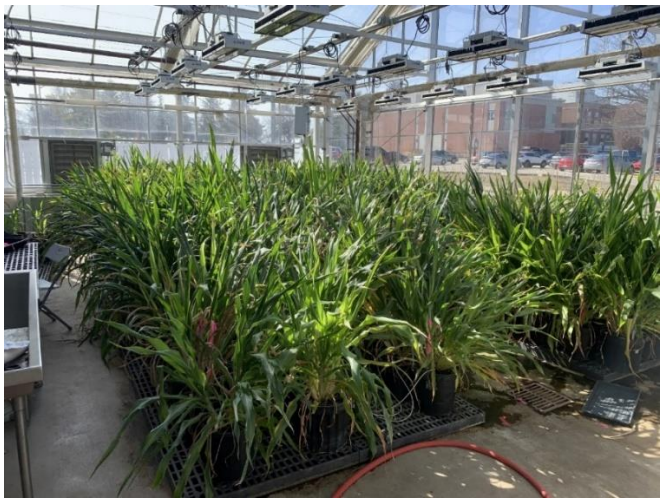

Figure 6. First Greenhouse Room at Week 9 of Collection

As a result of the unchanging leaf size, the setup was moved to a second greenhouse room to obtain more useful training data. In the second room Sorghum plants with no genetic modifications at different points in the growth cycle were grown and photographed. The second room focused on young sorghum plants as technical issues caused loss of data in the early days of data collection in the first room. Figure 7 showcases the setup in the second room.

Figure 7. Second Greenhouse Room at Week 0 of Collection

3.1.2 ZED Camera and Recording Setup

The ZED camera and ZED box—a processing unit used for the camera—were fastened onto a wooden plank and placed near the ceiling. Figure 8 is an image taken of the camera and box fastened to a wooden plank attached near the ceiling.

To access the ZED box and the connected ZED camera, a switch was set up to be used as an interface between a laptop and the ZED box. The ZED camera was connected to the ZED box through a USB type A 3.0 cable. The ZED box was connected to a switch through an ethernet cable. And the switch was connected to a laptop through an ethernet cable. Figure 9 showcases the switch and the power strip used to power the ZED Box, ZED Camera, and switch.

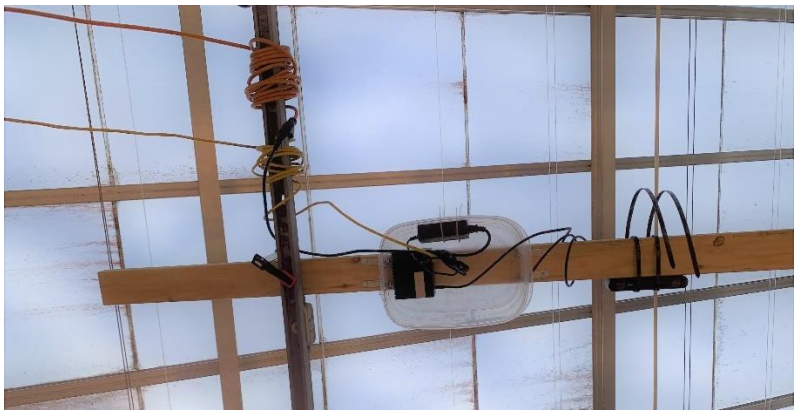

Figure 8. Bottom-Up View of ZED Camera and ZED Box on Plank

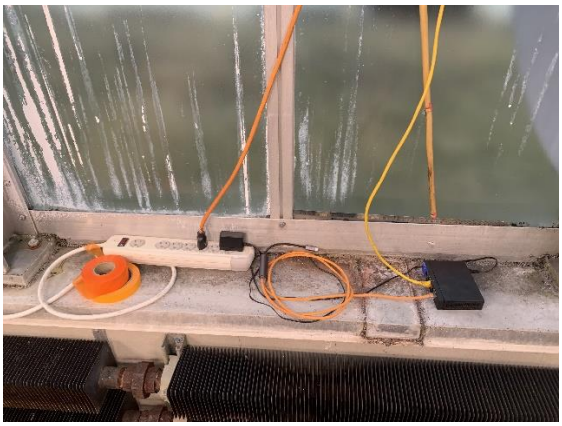

Figure 9. Switch and power strip connected to the ZED Box and ZED Camera

Issues arose during this portion of the data collection setup. The biggest issues were networking related. Initial attempts at connecting the ZED box to the school's network via the switch (to be able to access the camera remotely) were unsuccessful as the school's infrastructure didn't allow for it. Other local networking issues arose involving the static vs dynamic IP configuration of the ZED box.

Once access to the ZED box and ZED camera were established, the camera was set up to take recordings. The ZED API, integrated with the ZED SDK on the ZED Box, was utilized to capture recordings. The image settings weren't adequate for the scene and had to be tweaked, including the exposure, gamma, and white balance. Unfortunately, the ZED hardware made this process difficult. The 'auto' versions of these settings were ineffective. Figure 10 and 11 show common occurrences when capturing images with the ZED camera where the auto exposure and auto white balance didn't behave adequately.

Figure 10. ZED Image Capture at 'auto' exposure

Figure 11. ZED Image Capture at 'auto' white balance

Despite manual modification of the exposure and white balance settings, the behavior often didn't change. The conclusion arrived after experimenting with resolving the issue was that manually entering settings required more 'camera on' time to calibrate. When directing the ZED camera to take longer captures, these manual setting modifications were enacted and resulted in appropriate images as showcased in Figure 12.

Figure 12. ZED Image Capture properly calibrated.

Once the ZED camera was calibrated to capture the appropriate recordings, scripts were written to automatically collect them. The recordings captured by the ZED Camera are in an 'SVO' format. 'SVO' recordings are various frame captures from the ZED stereo camera and can be processed by the ZED SDK to create plain 2D images, point clouds, meshes, and other renderings not relevant in this project. The code that directs the camera to capture SVO recordings involves initializing the camera with the appropriate parameters and camera settings and then recording the frames.[2]

To capture SVO recordings recurringly over the course of a few months, a bash script was written that ran the SVO recording capture script in the background. At first the 'nohup' Linux command was used to run recordings in the background. However, this command led to some issues and thus the 'screen' Linux command was used for the background operation. An issue relating to camera overheating arose as well. The only permanent solution found was making the resolution of the recordings 1080 p instead of the 2K used to capture recordings in the first few months of the data collection portion of the project.

3.1.3 Control Photos

Once data collection was setup, control photos were taken. These were taken to determine the accuracy of the point clouds generated from the camera's recordings. A ruler at different distances from the camera and at different positions in the camera's field of view was photographed. Figure 13 shows an image of a ruler directly below the camera to measure inaccuracy as the measurement target gets

[2] For code excerpt see Appendix A.1

further from the camera. Figure 14 shows an image of a ruler on the far-right side of the image to measure distortion when nearing the edges of the camera's field of view.

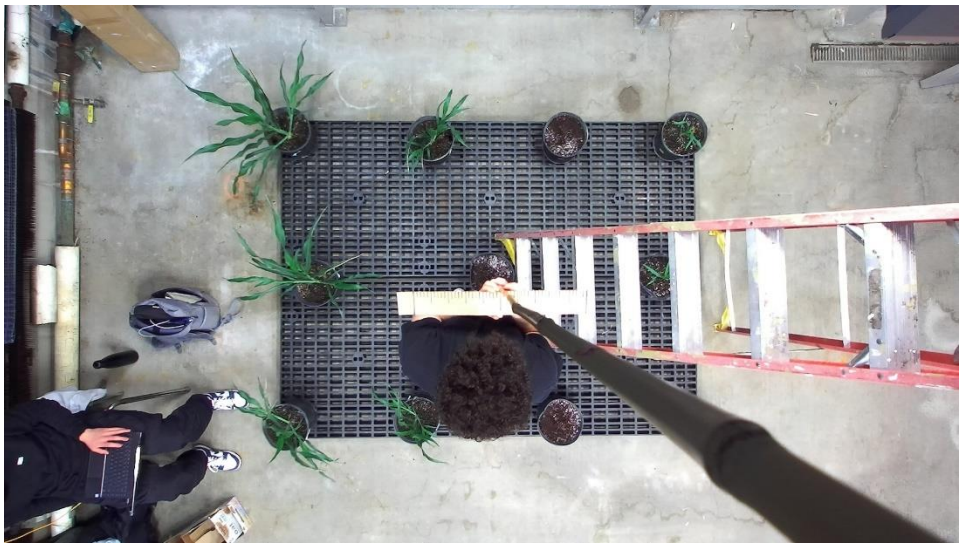

Figure 13. Calibration Photo of Ruler at 150cm from Camera

Figure 14. Calibration Photo of Ruler at 150cm from Camera on Right

3.2 Point Cloud Generation

3.2.1 ZED SDK Setup

The ZED SDK contains software that's able to interpret the proprietary SVO recordings and generate point clouds from them. To process the SVO recordings more quickly and away from ZED Box in the Greenhouse Room, the ZED SDK was installed onto a different machine. Standard setup for the ZED SDK is performed on the host machine's OS. This was the path taken in the project: downloading the ZED

SDK on a utilized anaconda virtual environment. However, complications arose related to the CUDA dependency: the version was incompatible with the ZED SDK. CUDA's intertwined relationship with the operating system and graphics card of the host machine made it difficult to continue the standard installation path for the ZED SDK. To avoid needing to make a great number of changes to the host machine, an alternative approach was taken. A publicly available Docker image with the ZED SDK was installed onto the system and a container was run from it. Within this container it was possible to make use of the ZED SDK and generate the point clouds from the collected recordings.

Additional work was done in setting up the docker container to make it persistent and to make SVO recording data saved within the docker container persist on container destruction using Docker Volumes. This made the data processing more reliable and resistant to data loss.

3.2.2 Point Cloud Generation and Visualization using ZED SDK

Once the ZED SDK was set up and the recordings were transferred into the container with the ZED SDK installed, sample code provided with the ZED SDK was run. The output was of an OBJ file type with specified vertices and their colors. This was visualized using MeshLab as shown in Figure 15. The associated 2D image is shown in Figure 16.

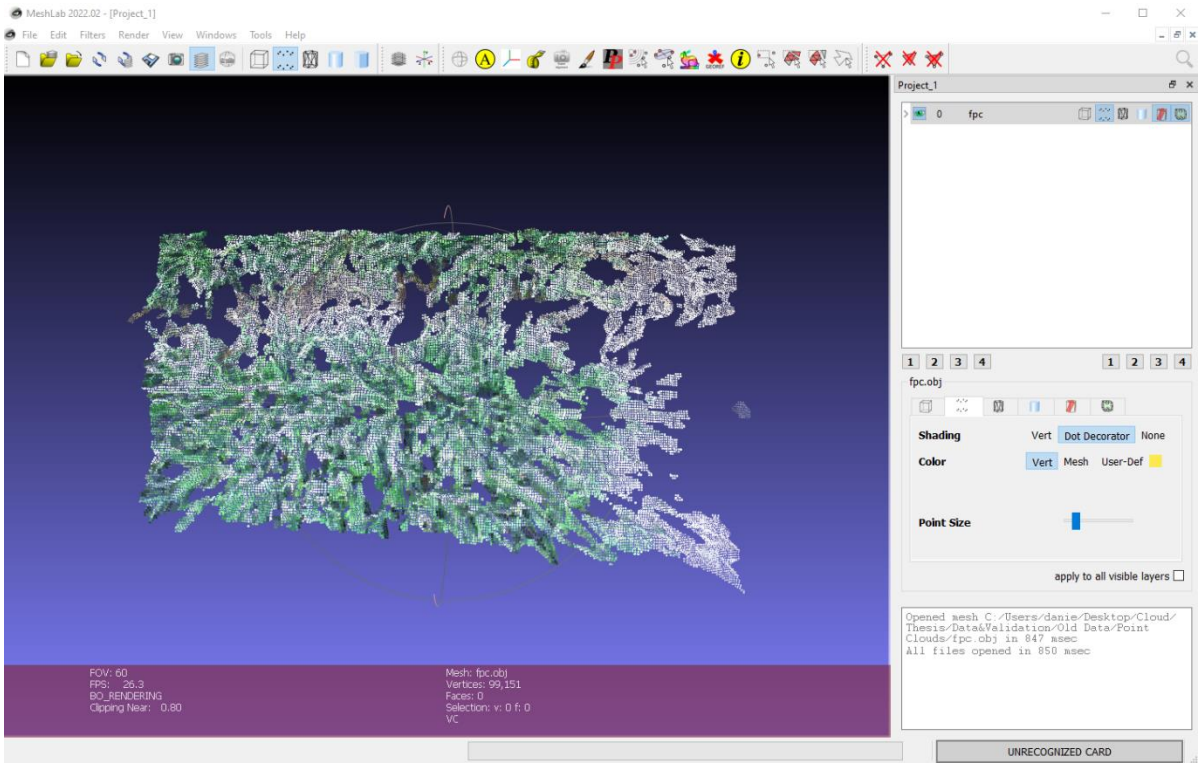

Figure 15. Visualization of Point Cloud with Sample ZED SDK Code

Figure 16. 2D Rendering with Sample ZED SDK Code

Further point cloud renderings, particularly of the control and calibration images intended to validate the camera's efficacy, showed underwhelming results. Another rendering of a point cloud is visualized in figure 17 with the associated 2D image shown in figure 18.

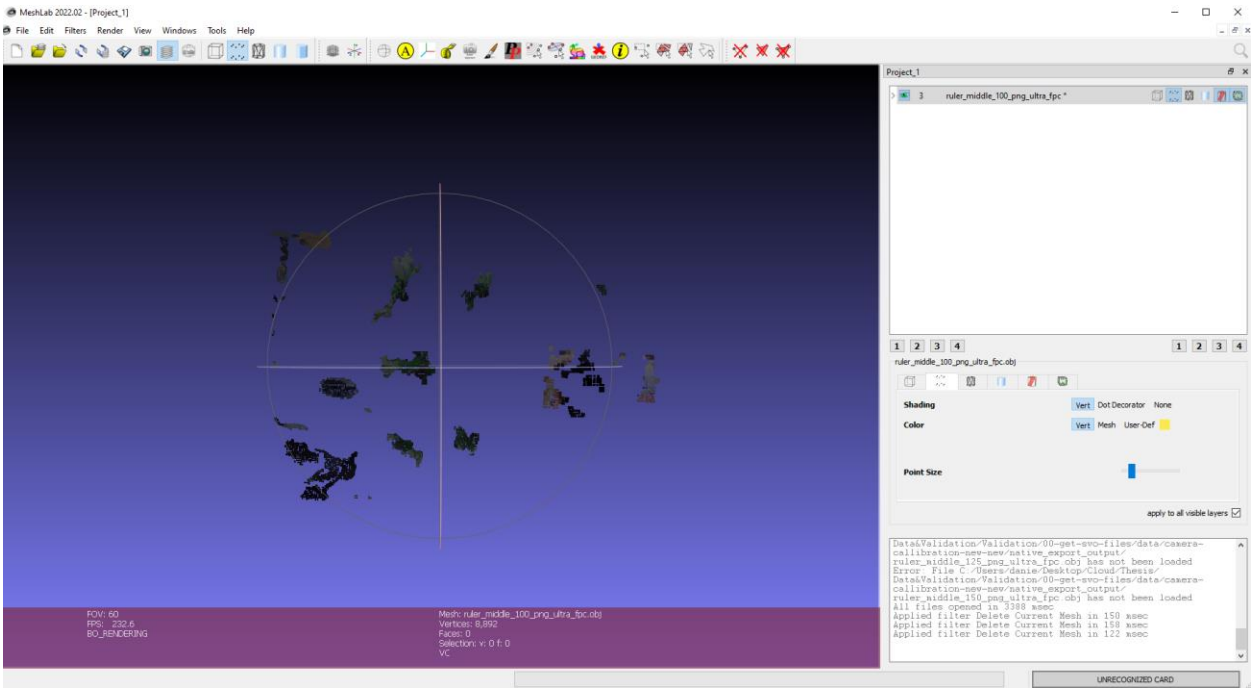

Figure 17. Visualization of Point Cloud with Sample ZED SDK Code

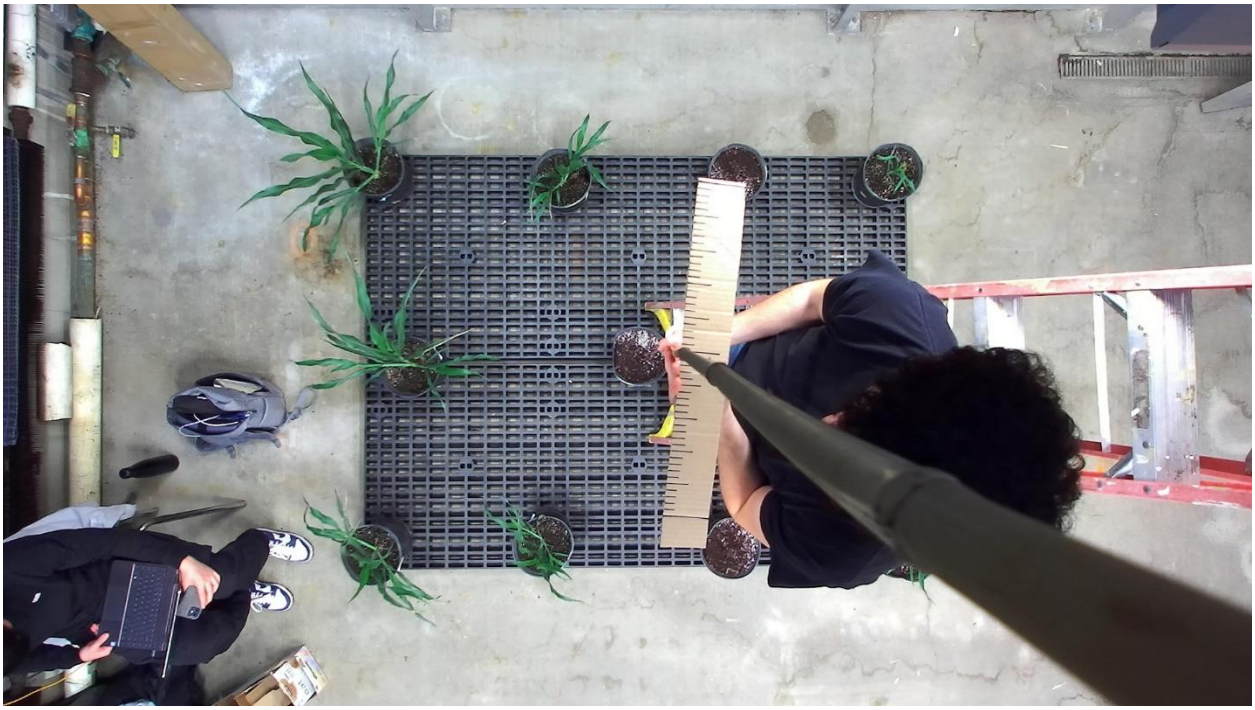

Figure 18. 2D Rendering with Sample ZED SDK Code

While these renderings showcase point clouds with an unexpectedly low number of points, other renderings could not be produced at all due to an error from the lack of data. This is demonstrated in figure 19.

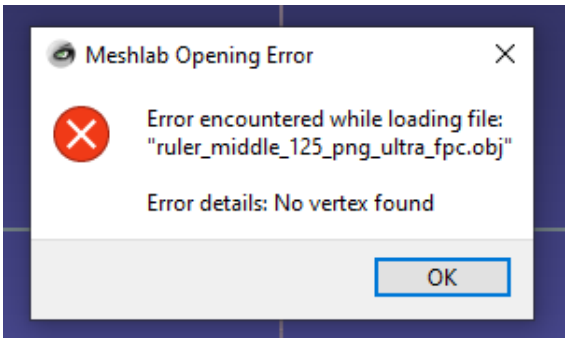

Figure 19. Error Response from Visualization of Point Cloud with Sample ZED SDK Code

The bad nature of these results encouraged the custom development of point cloud generation code. The ZED SDK contains functionality for generating the point cloud data in different manners. While the

sample point cloud generation code uses its spatial mapping feature to generate what it calls a “fused point cloud,” the custom code written utilizes the ZED Camera’s Depth Sensing and image data to gather XYZ and RGBA values[3]. The resulting data file shows an increase in data points between 2,000% and 20,000% using the custom code. For example, figure 20 shows side by side two renderings where the sample code generated a file with 118,791 (on the left side of the figure) points while the custom code generated a file with 2,662,689 points (on the right side of the figure). In every instance, the custom code results in more data points and more detailed renderings.

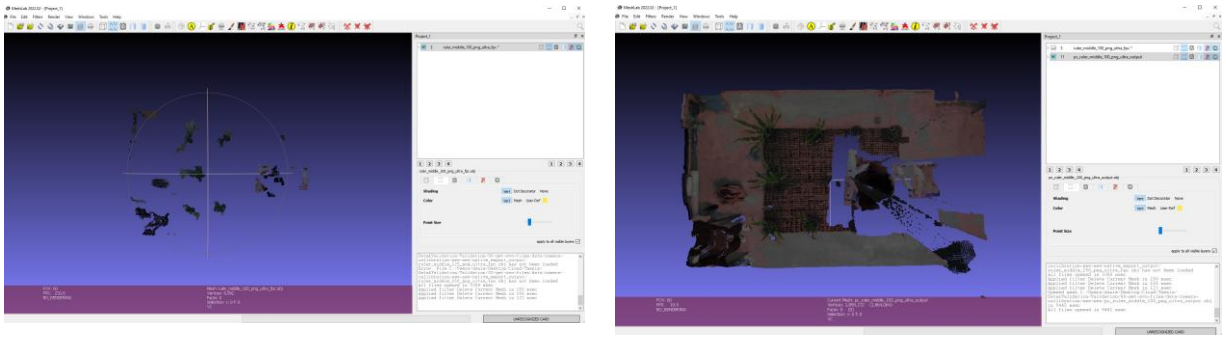

Figure 20. Visualization of Point Cloud with Sample Code (left) and Custom Point Cloud Generation Code (right)

3.2.3 Visualization of Point Cloud

To verify the efficacy of the ZED Camera and the Point Cloud Generation Software, physical measurements taken in the Greenhouse were compared with digital measurements from the generated point clouds. Visualizing the point clouds with MeshLab did not provide a simple way of measuring distance across points. To better accomplish this, Open 3D was used to visualize the point clouds. Figure 21 demonstrates the visualization with Open 3D.

[3] For more information read Appendix A.2

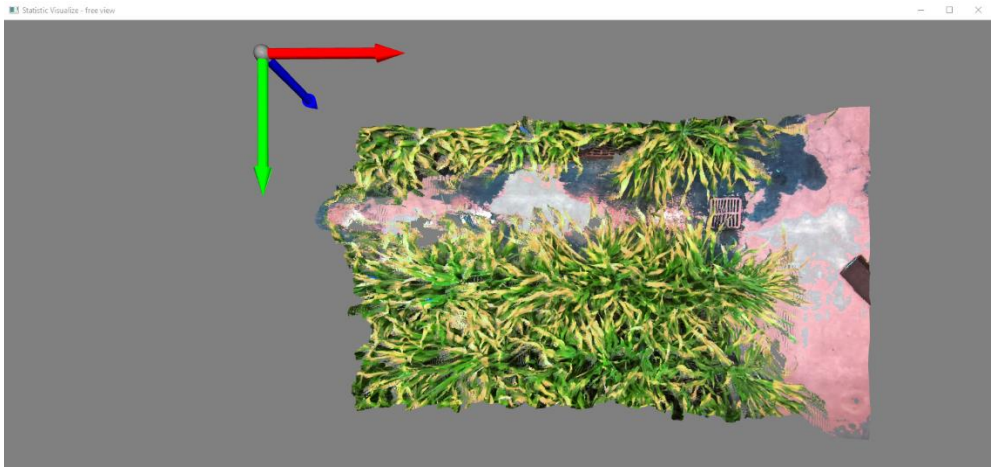

Figure 21. Visualization of Point Cloud in Open 3D

Using Open 3D's "VisualizerWithEditing," selected points in the graphical user interface were printed in the command line interface, retrieved, and used to calculate distances between points. This process is demonstrated in figure 22 where one of the control photos is used to measure the digital measurement equivalent to a 5cm gap on the ruler. The two points used to calculate distance are represented by the blue and yellow spheres in the figure.

Figure 22. Point Cloud

3.3 2D Representation of Point Cloud

3.3.1 Point Cloud Projection

The project intends to perform semantic segmentation on 2D images and use these 2D masks to filter the point cloud. Applying a 2D mask to a 3D point cloud can have its complications even when constraining the point cloud cropping to only the x and y dimensions. To remedy such complications, the point cloud is transformed to its two-dimensional representation.

After an initial failure of using the camera's intrinsic and extrinsic properties to project the point cloud onto a 2D representation of itself, a more generic approach was taken.

To create a 2D representation of a point cloud, points in a point cloud must be placed into a two-dimensional array. To perform this task, points need a relationship that assigns them to a specific place in the 2D array. To establish such a relationship the point cloud must be categorized. Looking down on the scene, from the perspective of the camera lens, in the same direction as the z-axis, assume the point cloud is split into a grid where each cell in the grid is an infinitely long pyramid where the tip starts at the lens and the base extends infinitely. The number of cells in the grid in the x direction should correspond with the number of pixels in the image's width axis. The number of cells in the grid in the y direction should correspond with the number of pixels in the image's height axis. These are the categories, where each cell defines what place in the 2D array a specific point would be in. Now all the points in the point cloud are within a cell in this grid. So, assuming the dimensions of the 2D array are *W* by *V*, and the camera lens represents the (x, y, z) coordinate of (0, 0, 0), the *W* position a point is placed in is defined by

$$W\ Position = \frac{Arc\ Tangent\left(\frac{X\ Coordinate}{Z\ Coordinate}\right) + |X\ Min\ Angle|}{|X\ Min\ Angle| + |X\ Max\ Angle|} * Image\ Width$$

Where $Arc\ Tangent\left(\frac{X\ Coordinate}{Z\ Coordinate}\right)$ represents the angle between the line that extends from the camera lens to the point AND the normal of the camera lens (theta in figure 23). This quantity is added to the Min Angle and divided by the angle of the entire scene to get the ratio of the point's angle to the angle of the entire scene. This ratio is multiplied by the image width to get the 2D array position.[4]

[4] See Appendix A.3 for code excerpts that utilize this method to perform the projection

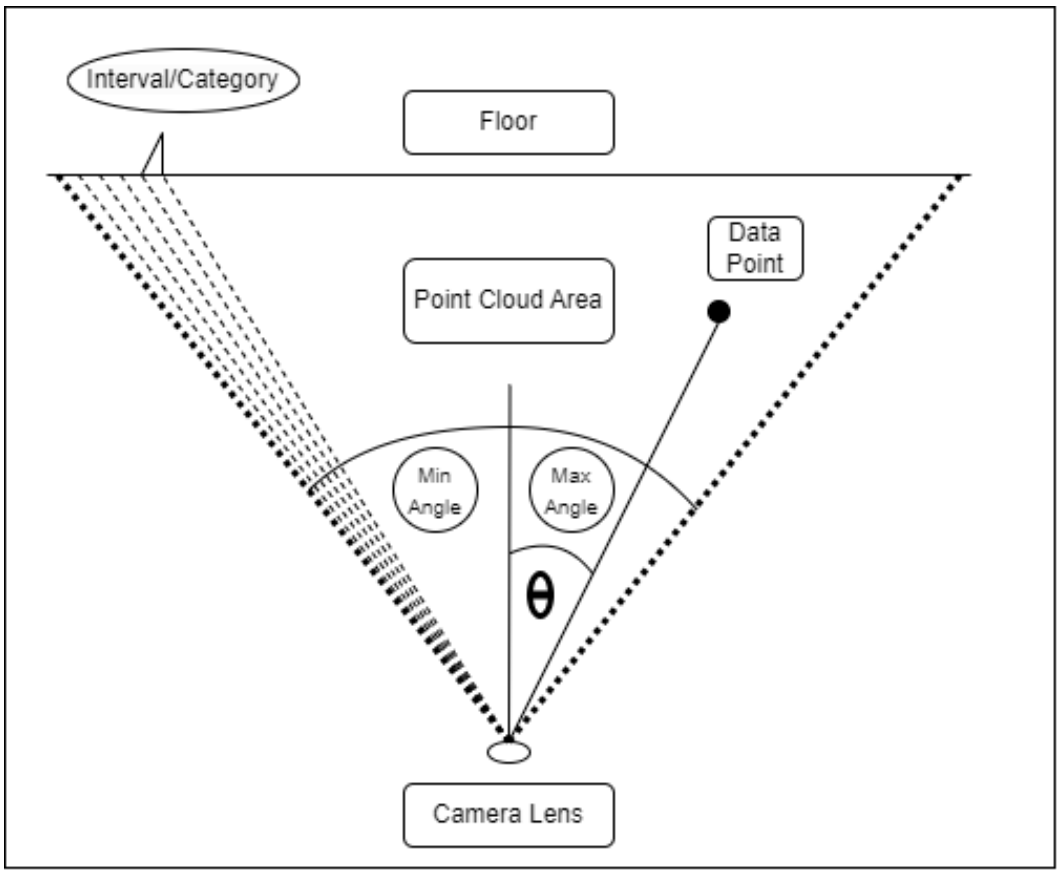

Figure 23. Diagram of Point Cloud Projection in one axis

3.3.2 2D Image from Projection

The point cloud projection acquired is a three-dimensional array of shape “height of the stereo image” by “width of the stereo image” by “number of points in the point cloud assigned to the present pixel,” where the last dimension’s size is variable. To transform this 2D representation into a 2D image, all the color values at each pixel-position are averaged. This results in a two-dimensional array representing all the pixels in the image where each item corresponds to the color value of that pixel.

The following figures (24, 25, and 26) show the point cloud, associated left image, and projected 2D image representation of the point cloud.

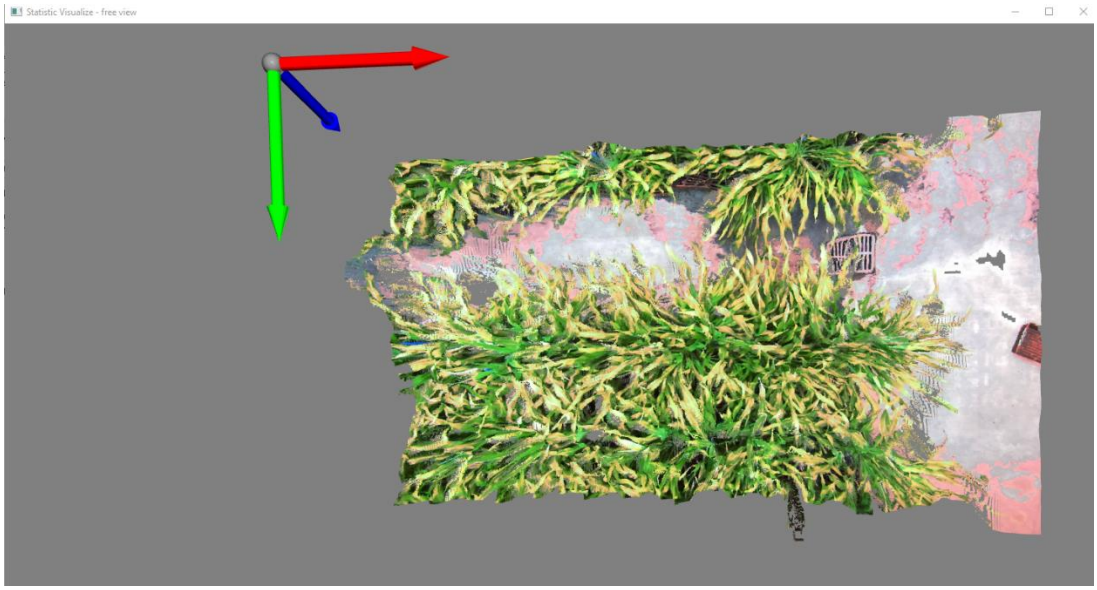

Figure 24. Point Cloud

Figure 25. Left Image

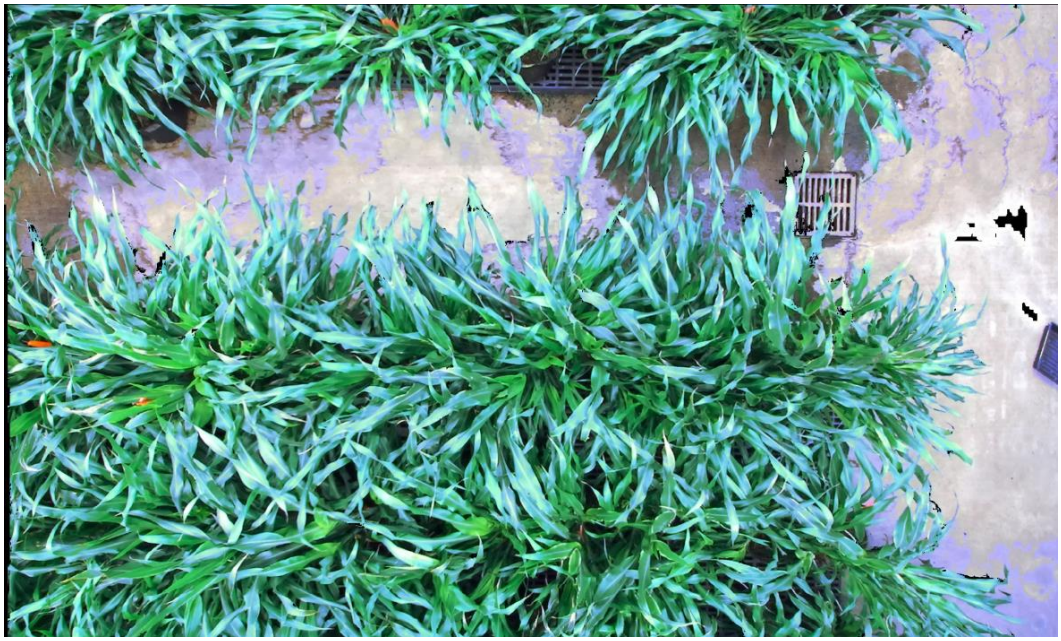

Figure 26. Projected 2D Image

3.4 Leaf Mask Acquisition with Instance Segmentation Model

To provide the model with a larger and more diverse dataset, sorghum leaf images taken for another project on a Go Pro were used along with the 2D representations of point clouds taken from the ZED Camera recordings. The images were annotated using the open source “VGG Image Annotator.” After annotation, the images went through an “augmentation” process to help the model with training and testing diversity. This process involves transforming raw images through color modifications, rotations, scaling, etcetera. After image augmentation the images were fed through an open-source implementation of Mask R-CNN written by Matterport to train a model to detect and segment leaves. The sequence described is visualized in figure 27.

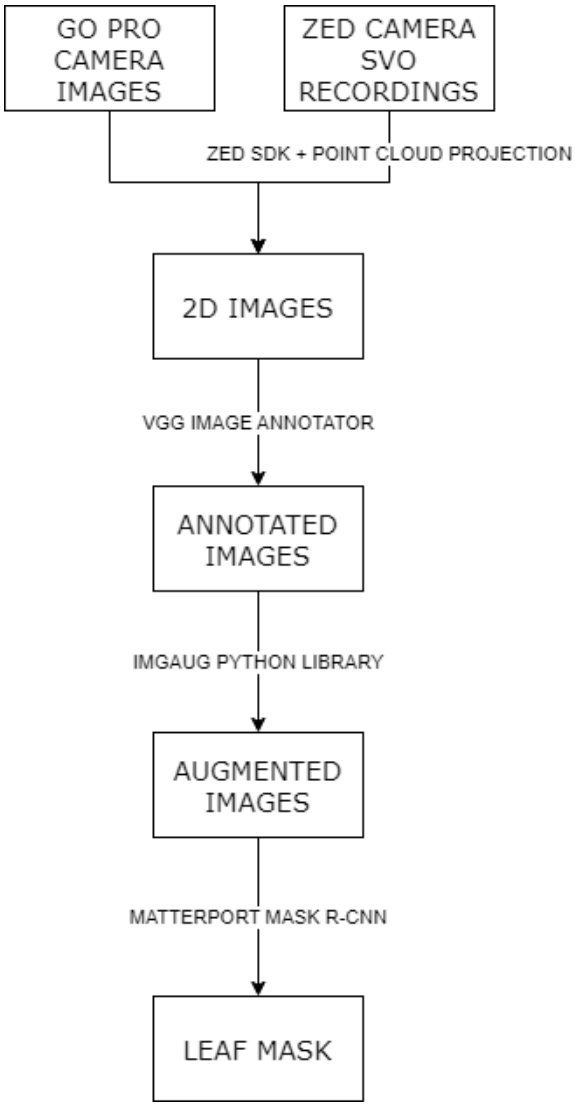

Figure 27. Training Mask R-CNN Model Workflow

3.4.1 Image Annotation

For annotation of the images, this project used the open source VGG Image Annotator. As no edits to the source code were required for this project and the tool is available for use online, this project utilized the online version of VGG Image Annotator. Annotation was accomplished by importing an image, annotating relatively flat leaves with little obstructions using the polygon tool, and exporting the annotations in a .JSON file. Figure 28 showcases the tool being used. Since the project served as an initial proof of concept with the goal of a minimum viable implementation, the instance segmentation model was only trained on flat and unobstructed leaves. Figure 29 shows a fully annotated image where only leaves that were unobstructed and had little curvature were annotated.

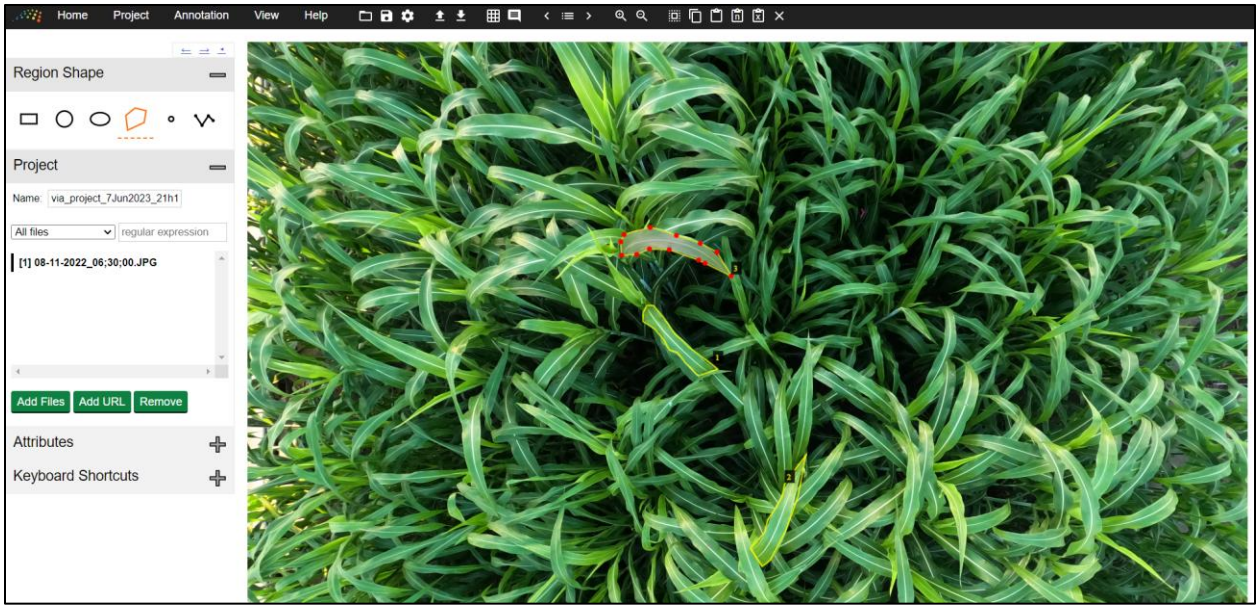

Figure 28. VGG Image Annotator

Figure 29. Annotated Image

3.4.2 Image Augmentation

Once the annotations are complete the image augmentation is applied to the images. The augmentation of the images was accomplished via the python imgaug library. The transformations used on the images

include rotation, vertical/horizontal flipping, color changes, blurring, and scaling.[5] Figure 30 showcases one of the images transformed via a few of the image augmentation methods. Annotations are transformed as necessary along with the images during the image augmentation process.

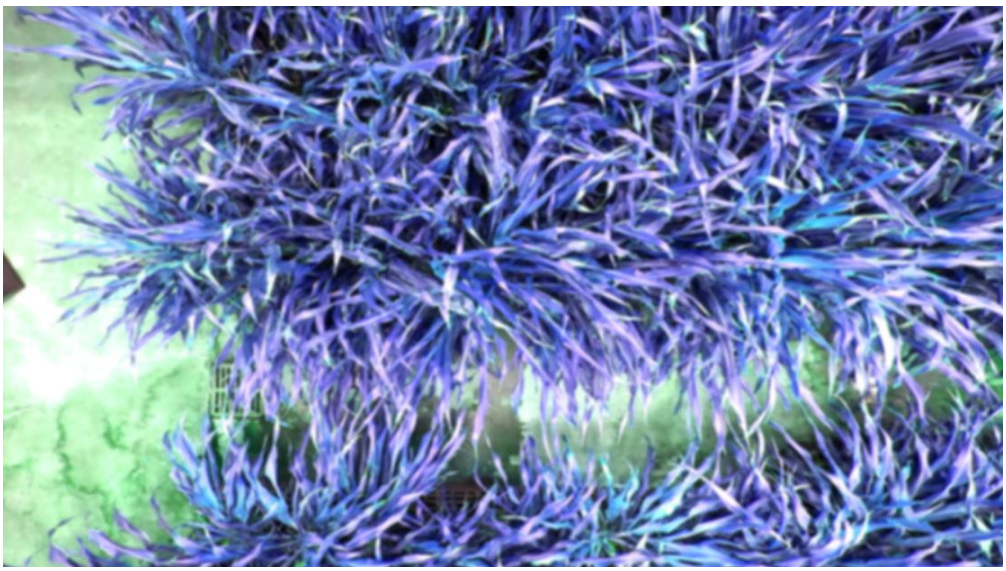

Figure 30. Augmented Image

3.4.3 Model Training

An open-source implementation of Mask R-CNN[6] with tensorflow and keras from Matterport was used to implement the instance segmentation model. The model was trained on a remote machine with access to two GPUs for training and image processing purposes. The remote machine operated on two Nvidia GPUs and CUDA version 9.0.

Table 6[7] outlines the configurations for the Matterport implementation of Mask R-CNN. Almost all parameters are the default ones. The only changed parameter is that of "GPU_COUNT" to specify the number of GPUs this project had access to.

The model was trained on the 2017 COCO dataset as a baseline and the sorghum plant dataset built for this project. The model was trained for 30 epochs each with 100 steps.

[5] See Appendix A.4 for a code excerpt
[6] For more information on how the model is trained read "Mask R-CNN" [12]
[7] See Appendix B

3.4.4 Model Prediction

The same configuration parameters as in table 6[8] were used to predict leaves with the trained model.

Using Matterport's Mask-RCNN model to predict the leaf masks in an image involves 3 main stages.

Stage 1 runs a binary classifier on boxes that cover the entire image and returns "object/no-object" scores. Boxes that are likely to obtain a leaf (high "object" score) get passed on to stage 2. Often these boxes that are classified as likely to contain the object do not cover objects fully. To fix this issue regression is performed to refine the location and size of the box to correctly encapsulate the boundaries of the object. The output of this stage is shown in figure 31.

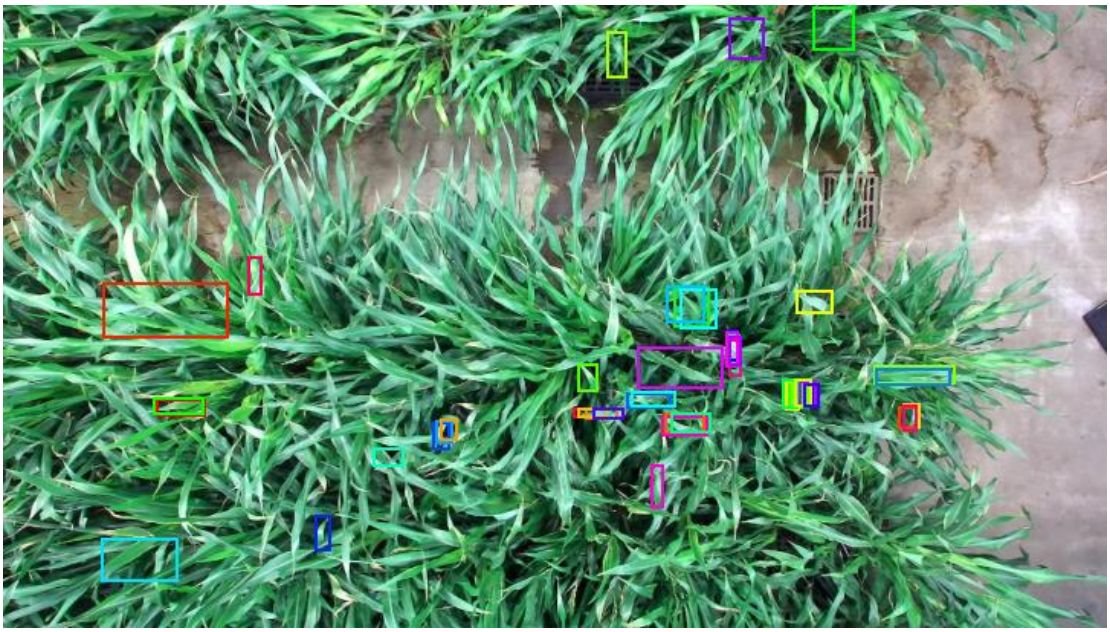

Figure 31. Final Proposal for Boxes with high "object" Scores

Stage 2 runs the model's classifier head on the boxes that got high object scores and generates the probability of these boxes containing the object. Boxes with low scores do not pass through to stage 3. In other applications this stage is responsible for classifying all the different types of objects. Since this project is only concerned with classifying one type of object, a sorghum leaf, this stage is simpler. The output of this stage is shown in figure 32.

[8] See Appendix B

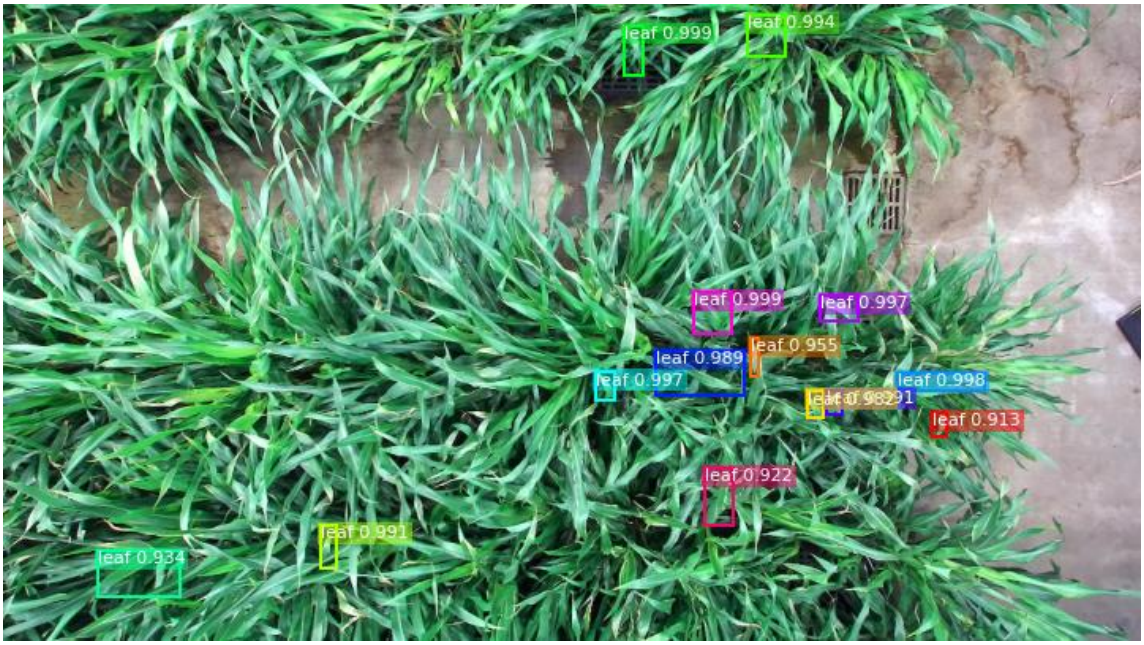

Figure 32. Boxes with high probability scores

Stage 3 runs the model's mask head on the boxes that got high probability scores to generate segmentation masks. One mask is predicted for each box. Figure 33 shows the final mask predictions on the original image.[9]

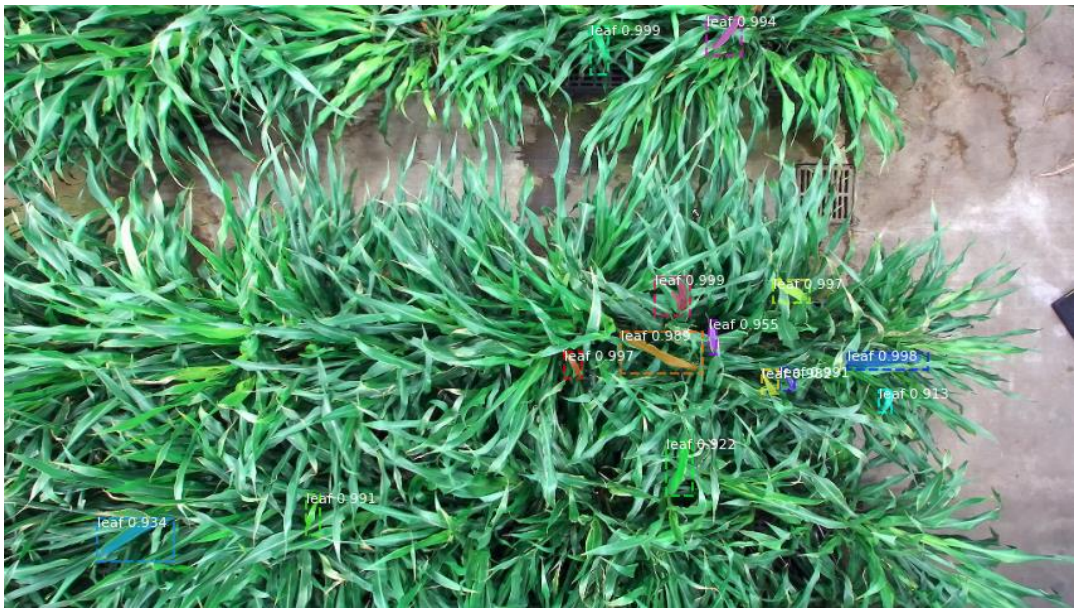

Figure 33. Predicted Masks Superimposed on Image

[9] See Appendix A.4 for a code excerpt that performs prediction utilizing Matterport's Mask R-CNN

3.4.5 Mask Isolation

The final part of this section of the project was isolating one leaf mask in an image to be used in the succeeding steps. This involved two transformations.

The mask objects from the model predictions are 3D arrays, where the first two dimensions correspond to the two dimensions of the image, and the third dimension corresponds to an array of true and false values that represent whether that corresponding pixel is a part of each mask or not. Thus, returned masks objects are of size 1024 by 1024 by number of masks in the image. To isolate one mask, the returned masks object was transformed to various 1024 by 1024 by 1 mask objects, one for each mask. This was the first transformation.

The second transformation pertained to an aspect of the training and prediction process that has not been covered yet for clarity. The image preprocessing before using the model for training or prediction resizes all images to 1024 by 1024 pixels. Establishing this restriction lends itself to helping the model. The complete image that results from using the model to predict masks is shown in figure 34. To be able to apply the isolated masks on the projected point clouds that have not been resized to 1024 by 1024 pixels, these “black horizontal bars” need to be removed from the mask objects obtained from the model predictions. Performing this is simple as the mask objects from the model predictions are one-to-one with the pixels of the image, thus cropping the mask objects works the same as cropping the image. After this transformation, the individual mask is obtained. Figure 35 visualizes one of the masks.

Figure 34. Predicted Masks Superimposed on Resized Image

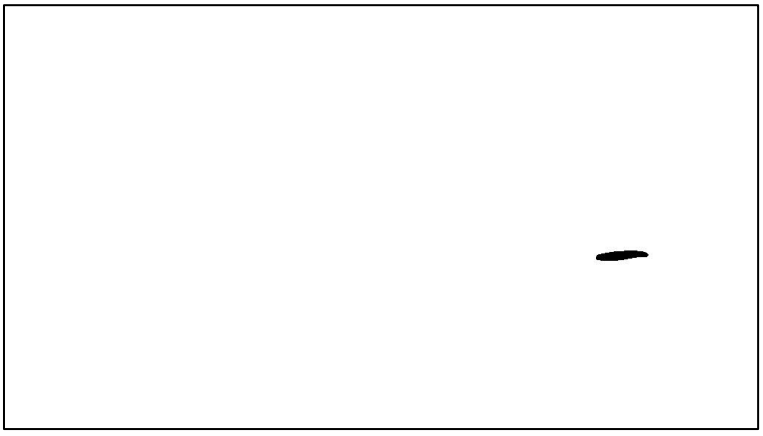

Figure 35. Isolated Mask Visualized as Black Pixels on a White Background

3.5 Leaf Mask Scaling

The leaf masks that result from the instance segmentation model's inference are not scaled appropriately to be able to apply the masks onto the projected point clouds. During the inference process the model resizes and compresses the images for ease and efficiency of data processing. The resulting leaf masks cannot be applied to the stereo images or projected point cloud before scaling the masks appropriately.

Due to the work in removing the parts of the mask object that do not correspond with pixels in the images in section 3.4.5 mask isolation, all that's left is scaling the mask appropriately in the x and y coordinates. Undoing the image compression results in needing to upscale the mask object and is accomplished via nearest-neighbor interpolation.[10] As the x-y ratio before and after scaling remains almost the same, there is no visual difference between the mask before and after scaling unless superimposed on the image of the leaves. This is shown in figure 61 in section 4.5.

3.6 Point Cloud Cropping

With a 2D representation of the point cloud and a leaf mask that was scaled appropriately to this 2D representation, all the components needed to crop the point cloud and acquire a point cloud of an individual leaf are available.

Applying the scaled leaf bitmask on the 2D representation of the point cloud yields the 2D representation of the point cloud of an individual leaf. Extracting all the points listed in this representation and visualizing it yields the point cloud of an individual leaf[11] as shown in figure 36. Further proof of this is shown by converting all the points in the leaf to red and displaying the entire scene. This is shown in figure 37.

[10] See appendix A.5 for code excerpt
[11] See appendix A.6 for code excerpt

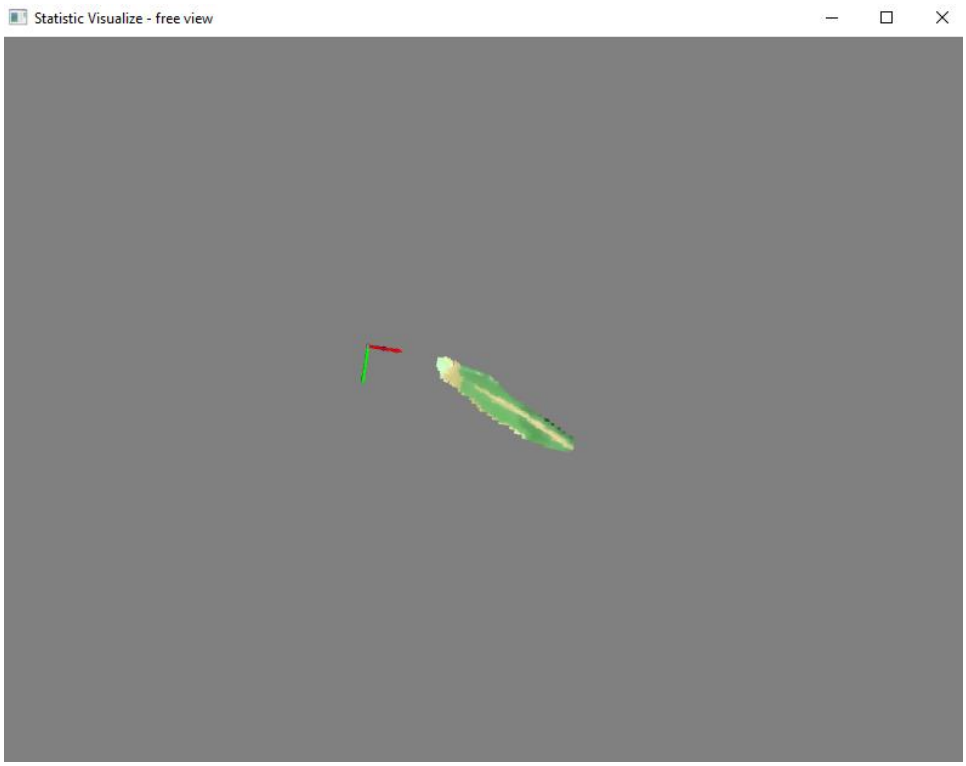

Figure 36. Point Cloud of an Individual Leaf

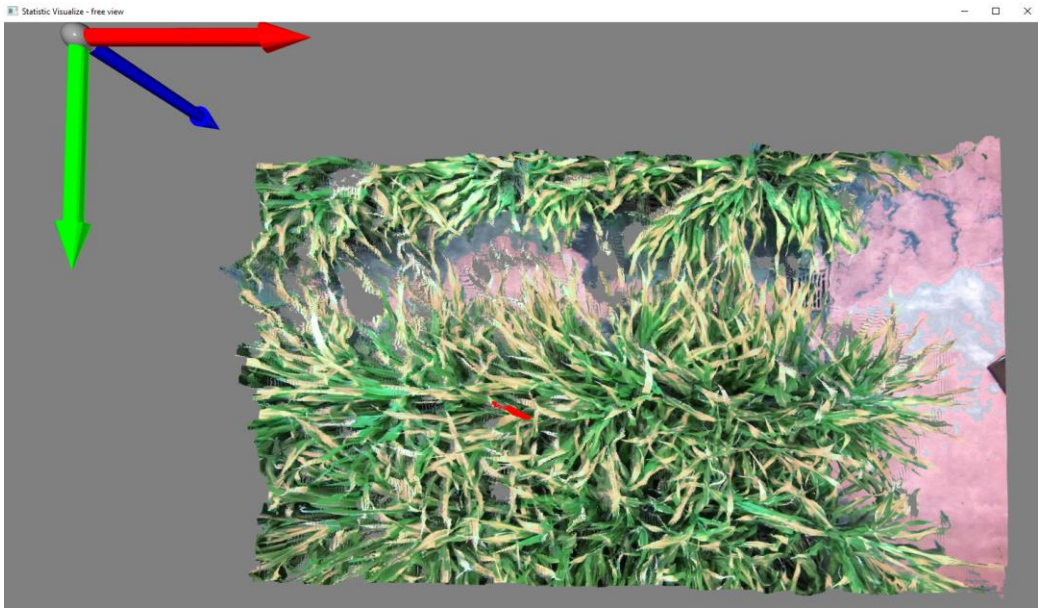

Figure 37. Point Cloud of The Scene with the Red Pixels representing One Leaf

4. Research Results

All the six steps outlined in the methodology chapter have their own technique for validation. This chapter will outline the results of each section of the project and demonstrate how each section was validated.

4.1 Data Collection and Point Cloud Generation & Rendering

To validate that the camera produces accurate data, and the renderings accurately capture the information in the SVO recordings, a set of control photos were taken with the ZED camera to compare real and vision-based measurements. Examples of these control recordings are displayed as 2D images and as point cloud renderings in section 3.1.3 Control Photos.

After generating the point clouds of the control recordings and rendering them using a point cloud visualization tool powered by the open 3d python library, vision-based measurements were taken along the ruler to populate metrics that help demonstrate the accuracy and precision of depth, length, and width vision-based measurement. Two areas of loss of accuracy were identified: distance from camera and distance to the edge of the field of view (FOV) of the camera. The hypotheses of the impact of these metrics were that a higher distance from the camera would correlate with lower accuracy of measurements and data collected closer to the edge of the camera FOV would be impacted by lens distortion and contribute to lower accuracy of measurements.

4.1.1 Distance From Camera's Effects on Accuracy

Example renderings of control recordings taken to determine the impact distance from camera has on measurement accuracy are shown in figures 38, 39, 40, and 41.

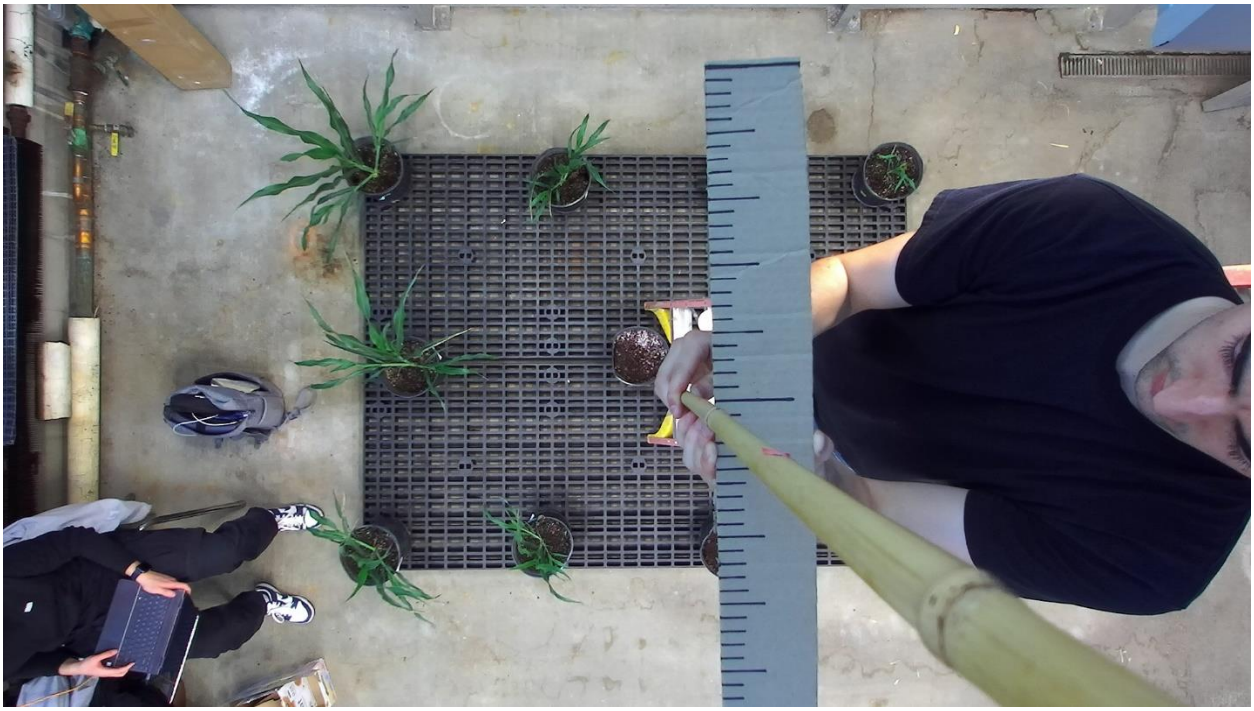

Figure 38. Control Photo of a Ruler 50 cm From the Camera's Lens

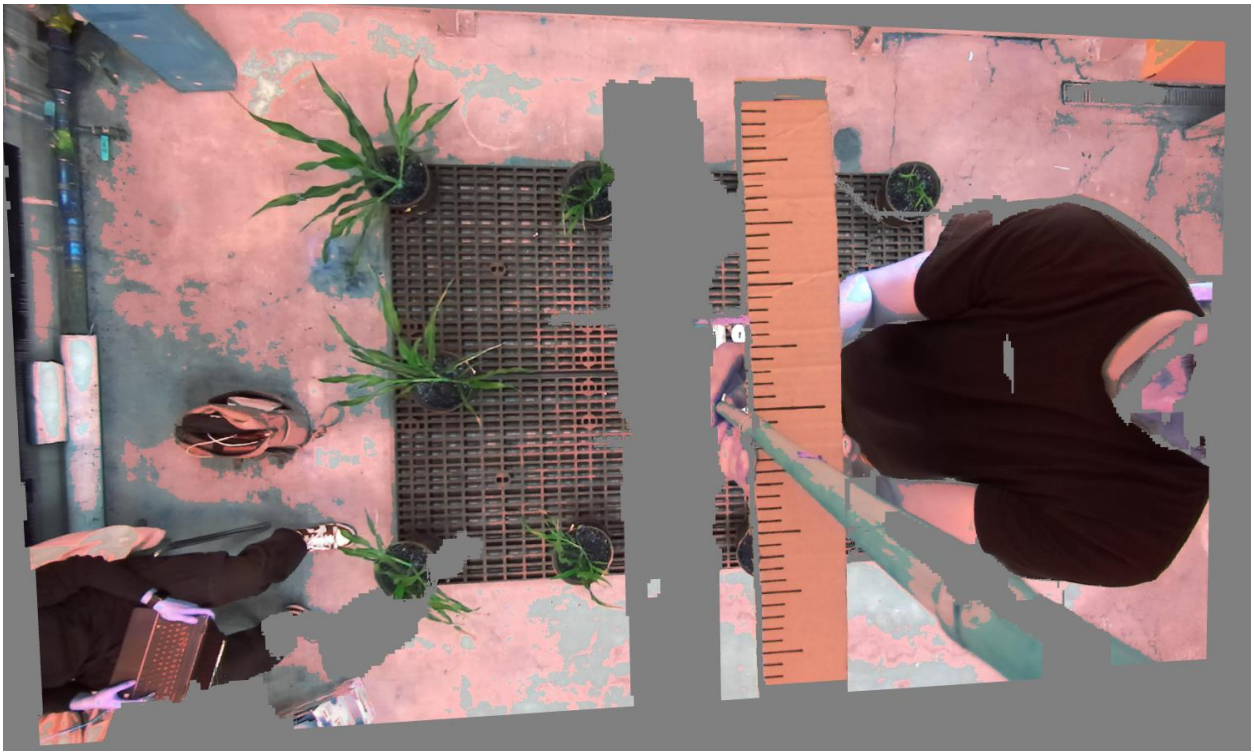

Figure 39. Point Cloud Rendering of a Ruler 50 cm From the Camera's Lens

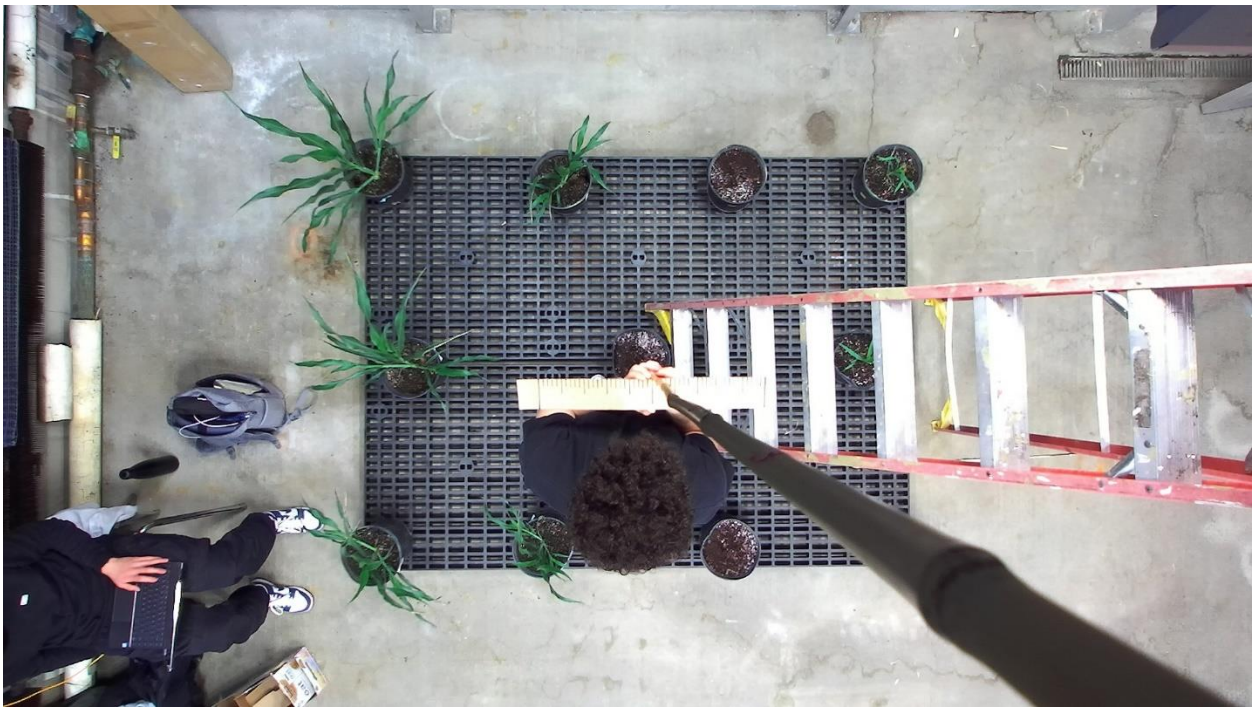

Figure 40. Control Photo of a Ruler 150 cm From the Camera's Lens

Figure 41. Point Cloud Rendering of a Ruler 150 cm From the Camera's Lens

Table 1 and 2 showcase data intended to demonstrate trends with regards to accuracy dependent on the measurement's distance to the camera. Table 1 showcases depth measurements taken across the different recordings. Table 2 showcases length/width measurements taken across the different recordings. Vision-based measurements taken at 25 cm from the camera are unpopulated as the point clouds generated from the recordings were not able to generate points of objects at 25 cm or closer to the camera. The generated point clouds included objects at 50 cm and further from the camera. These measurements are populated and are used to determine trends in the accuracy of recordings and rendered point clouds across the different dimensions. These measurements were taken directly underneath the camera lens so that distortion from the closeness to the camera's viewing edge would have no impact on the inaccuracy.

Table 1. Depth Measurements Taken at Different Distances from Camera

Real Distance	Point Cloud Distance	|Expected – Actual|	Percent Error
25 cm	-	-	-
50 cm	50.87 cm	0.87 cm	1.74 %
75 cm	76.02 cm	1.02 cm	1.36 %
100 cm	102.94 cm	2.94 cm	2.94 %
125 cm	127.84 cm	2.84 cm	2.27 %
150 cm	162.42 cm	12.42 cm	8.28 %

Table 2. Length/Width Measurements Taken at Different Distances from Camera

Distance From Camera	Real Distance	Point Cloud Distance	|Expected – Actual|	Percent Error
25 cm	1 cm	-	-	-
50 cm	1 cm	1.0127 cm	0.0127 cm	1.27 %
75 cm	1 cm	1.0246 cm	0.0246 cm	2.46 %
100 cm	1 cm	1.0335 cm	0.0308 cm	3.08 %
125 cm	1 cm	1.0845 cm	0.0335 cm	3.35 %
150 cm	1 cm	1.1850 cm	0.0845 cm	8.45 %

The associated graph displaying the trend of the data in Table 1 and best fitting curve equation is shown in figure 42. The associated graph displaying the trend of the data in Table 2 and best fitting curve equation is shown in figure 43.

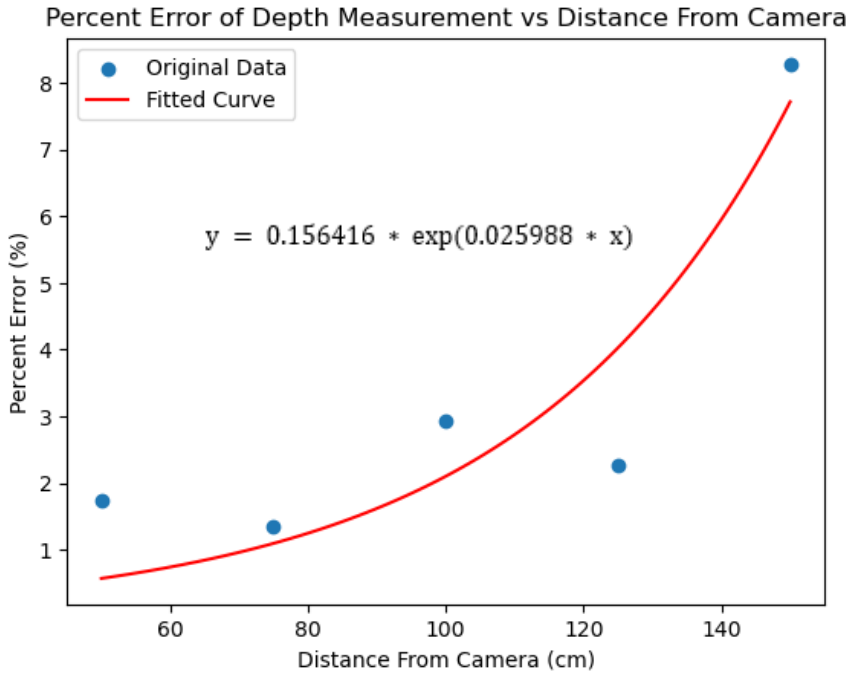

Figure 42. Best Fitting Curve Graph of Percent Error of Depth Measurement vs Distance from Camera

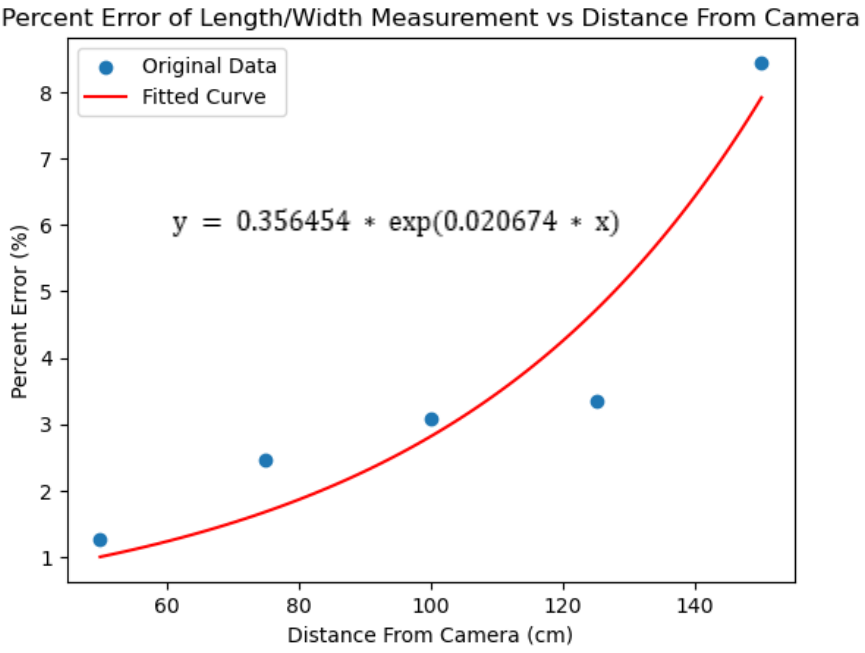

Figure 43. Best Fitting Curve Graph of Percent Error of Length/Width Measurement vs Distance from Camera

4.1.2 Distance from FOV of the Camera's Effect on Accuracy

Example renderings of control recordings taken to determine the impact distance from FOV of the camera has on measurement accuracy are shown in figures 44 and 45.

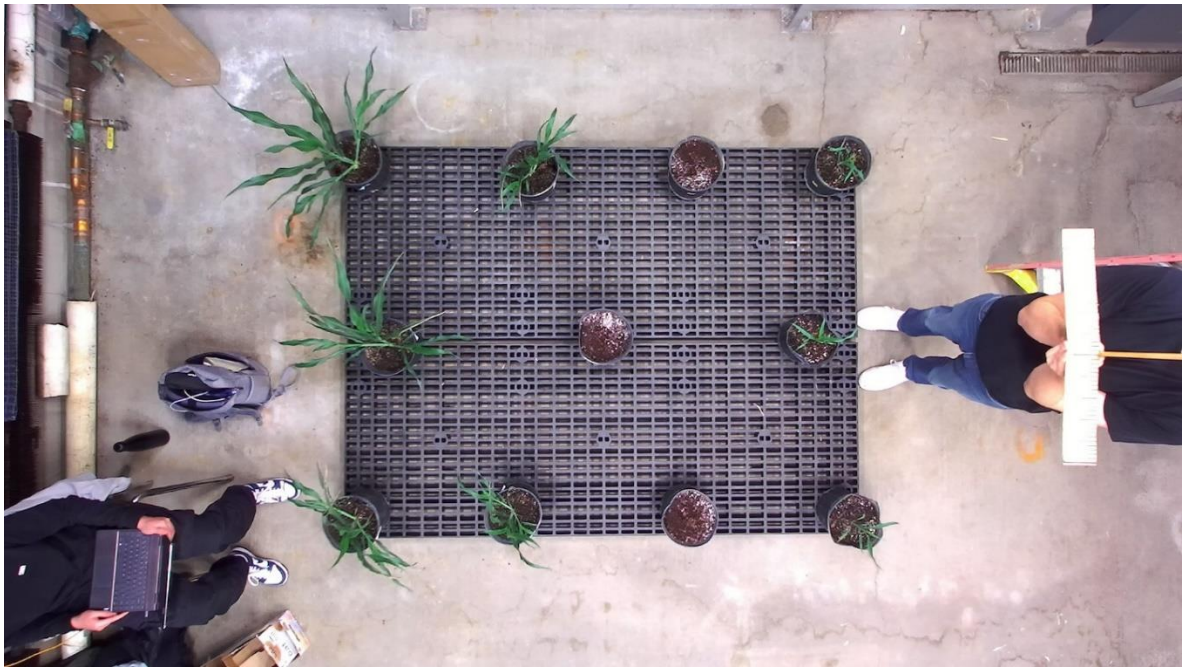

Figure 44. Control Photo of a Ruler 150 cm From the Plane of The Camera's Lens

Figure 45. Control Photo of a Ruler 150 cm From the Plane of The Camera's Lens

To measure the effect that taking measurements closer to the camera's viewing edge has on accuracy, distance to viewing edge was defined as a fraction: the angle between the camera lens's normal and the line from the camera lens to the point as a portion of the maximum angle (the angle between the camera lens's normal and the line from the camera lens to the furthest point on the edge of the point cloud as shown in figure 46). Thus, dividing the former angle over the latter, the outcome is a ratio of how far to the viewing edge the point is, where 0 denotes being as far as possible (in the center of the image), and 1 denotes being as close as possible to the viewing edge (at the edge of the image). Figure 46 shows a diagram that visualizes the calculation of the angles on one axis.

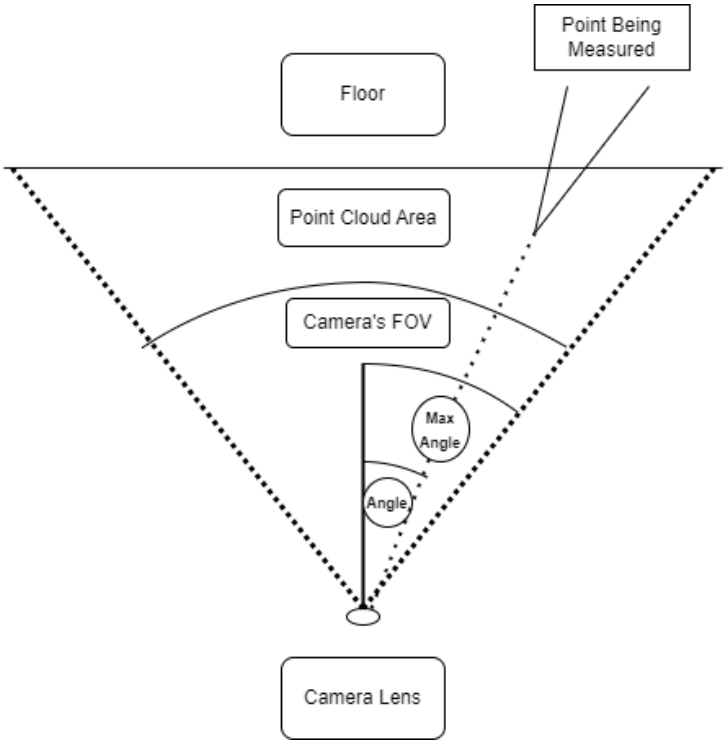

Figure 46. Calculations of 'Point Angles' in the Point Cloud

Table 3 showcases data intended to demonstrate trends with regards to accuracy dependent on the measurement's distance to the edge of the FOV of the camera. The table contains length/width measurements taken across the different recordings. Measurements were taken to test accuracy in the closeness to the viewing edge in the x axis and the y axis. Measurements were conducted so that the axis not being measured had a 0-degree angle with the camera lens. Measurements were all conducted at 150cm from the camera lens. The vision-based measurements are populated and are used to determine trends in the accuracy of recordings and rendered point clouds. Since these measurements were conducted at 150cm from the camera lens, inaccuracy due to "Distance from Camera Lens"

influenced the measurement results. To diminish this impact, the percentage error caused by the distance from the camera is subtracted from the percentage error calculated with these measurements. The equation in figure 43 is used where x is substituted with the distance from the camera lens in the measurements, 150 cm. This resulted in a 7.937% contribution from the distance to the camera lens. However, this includes a "base inaccuracy." As 50cm resulted in the lowest percent error, the percent error at 50cm is deemed as "base inaccuracy" not resultant of the distance from camera but due to the camera's accuracy in measurements in general. This "base inaccuracy" (0.9987 %) is added back for a more meaningful picture of the percent error.

Table 3. Length/Width Measurements Taken at Different Locations in View and Distances from Camera

Distance to Viewing Edge	Real Distance	Point Cloud Distance	Percent Error	Final Percent Error
0.1035	10 cm	10.831 cm	8.31 %	1.37 %
0.6447	10 cm	10.838 cm	8.38 %	1.44 %
0.6625	10 cm	11.303 cm	13.03 %	6.09 %
0.8578	10 cm	11.850 cm	18.50 %	11.56 %

The chart demonstrating the accuracy of the vision-based measurements relative to the actual measurements is shown in figure 47 along with its best fitting curve equation.

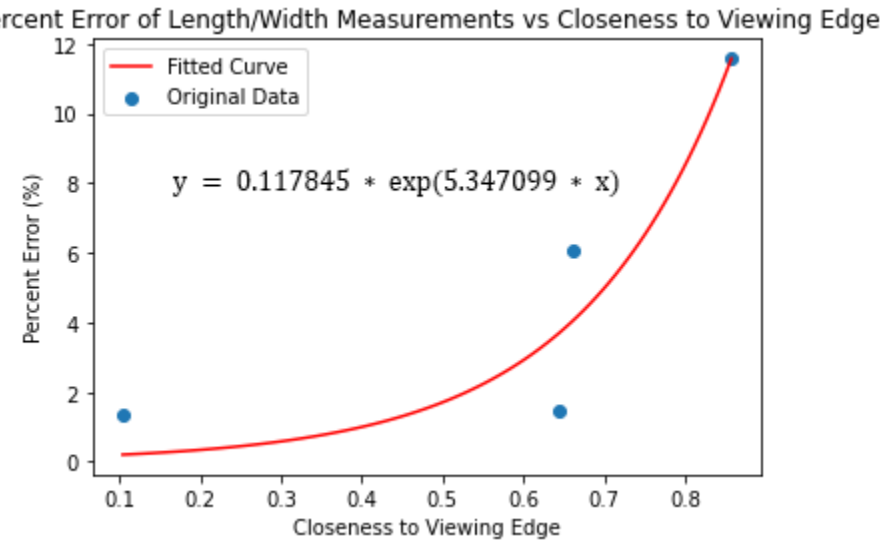

Figure 47. Best Fitting Curve Graph of Percent Error of Length/Width Measurement vs Closeness to Viewing Edge of Camera

4.1.3 Combined Effect on Accuracy

The combined effects on measurement accuracy due to inaccuracy from the measurement distance to the camera's lens and the closeness to the camera's viewing edge is represented as total percent error in figure 48 and figure 49

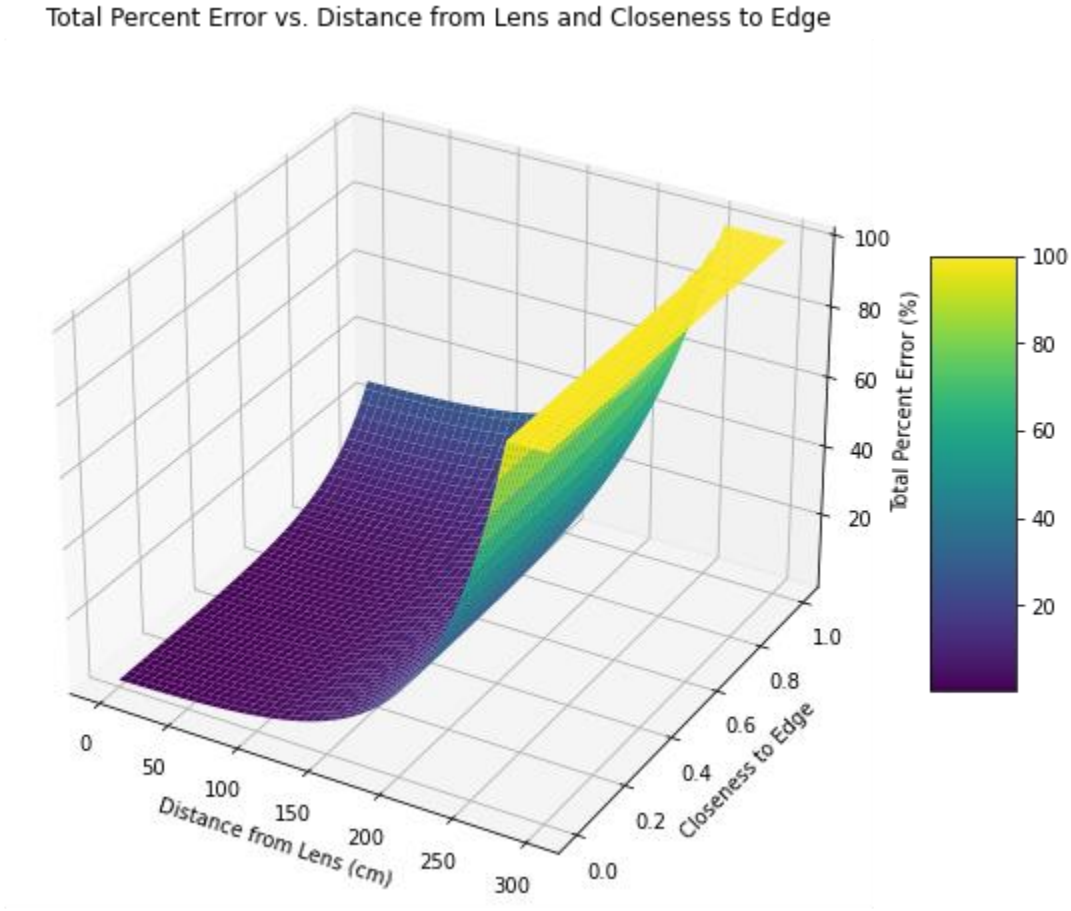

Figure 48. 3D Plot of Total Percent Error vs Distance From Lens and Closeness to Viewing Edge

$$Total\ Percent\ Error = 1 - (1 - (0.356454 * \exp(0.020674 * distanceFromLens)) * (1 - 0.039843 * \exp(6.510183 * closenessToEdge)))$$

Figure 49. Equation for Total Percent Error vs Distance From Lens and Closeness to Viewing Edge

Reducing the boundaries to only those that fall within reliable outcomes and a percent error lower than 20% yields figure 50.

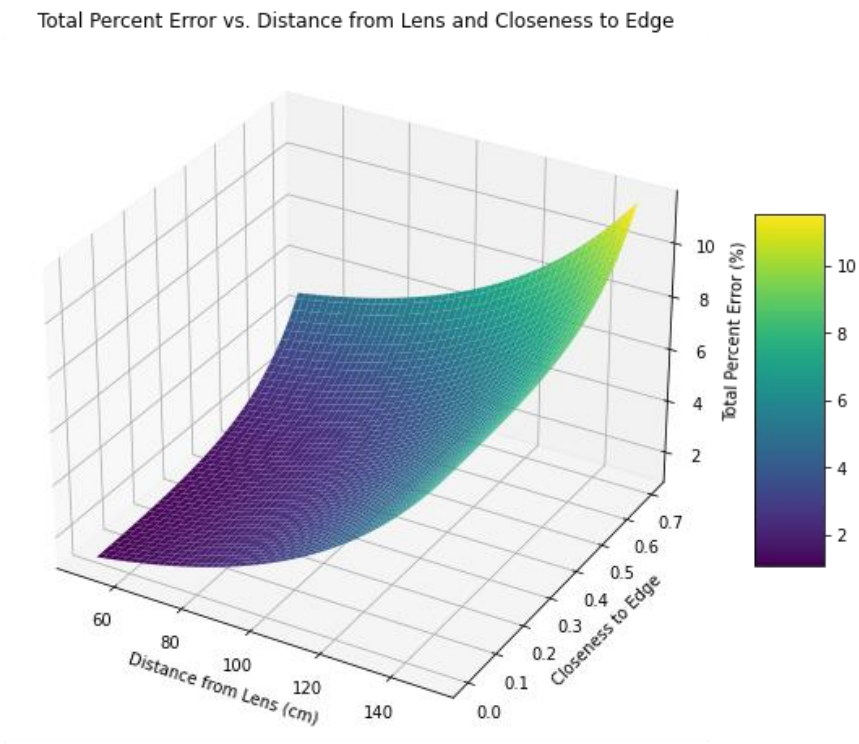

Figure 50. 3D Plot of Total Percent Error vs Distance From Lens and Closeness to Viewing Edge with Boundaries

Figure 51 shows the curve Closeness to Edge vs Distance From Lens that amounts to 5% total error. Anything within this curve will theoretically be within 5% of inaccuracy.

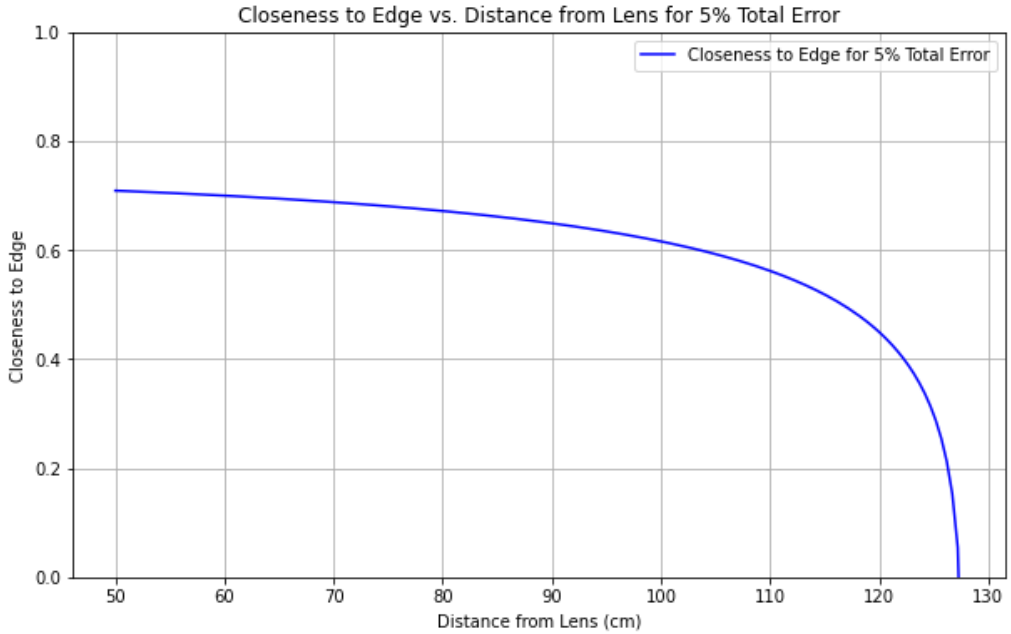

Figure 51. Curve for Closeness to Edge vs Distance From Lens when Total Percent Error is 5%

4.1.4 Precision Measurements Across Images

For growth studies in which phenotype data is being compared across different images, it is most important for the data to be precise. That is, it is important for measurement variability to be low across different images and locations in the images so that an indication of change in measurements correlates to physical changes. To measure the precision of the ZED Camera and its recordings, length/width measurements of a ruler were taken across different control images and locations in the control images. This data is displayed in Table 4. The relative standard deviation is calculated to measure the level of precision. This is calculated by dividing the standard deviation of the range over the average of the range and multiplying that quantity by 100. The relative standard deviation measurements at each distinct distance from the camera are shown in Table 4.

Table 4. Length/Width Measurements Taken at Different Locations in View and Distances from Camera

Distance From Camera (cm)	Length/Width Measurements (cm)										Relative Standard Deviation (%)
50	51.5	49.7	48.3	52.1	51.4	51.2	49.1	51.1	-	-	2.668759744
75	56.7	48.4	49.9	50.1	53.5	50.8	51.3	51.1	55.9	-	5.435697946
100	52.8	50.0	49.7	54.3	50.8	54.3	47.7	-	-	-	4.864811708
125	55.1	49.8	53.1	53.6	58.3	54.7	50.9	50.4	50.1	54.5	5.196696921
150	57.9	57.6	54.6	55.8	57.9	-	-	-	-	-	2.580153341

As Table 4 demonstrates, the relative standard deviation remains below 5.5% across the different distances to the camera. Figure 52 shows a chart with the data points and best fitting curve of the relative standard deviation across the different distances from the camera.

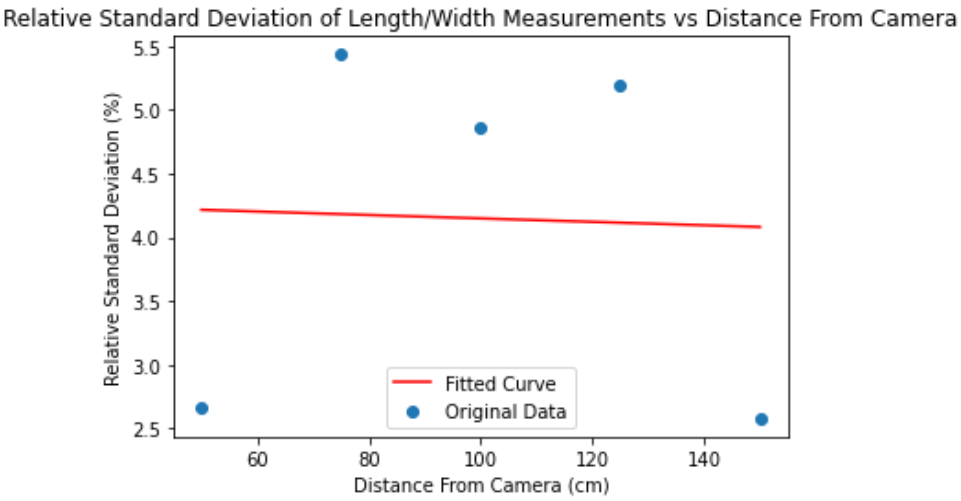

Figure 52. Best Fitting Curve Graph of Percent Error of Length/Width Measurement vs Distance to Viewing Edge of Camera

4.2 2D Representation of Point Cloud

4.2.1 Point Cloud Projection

To validate the efficacy of the transformation from a point cloud to a 2D representation of the point cloud, the transformation back to a point cloud was measured for any differences to the original point cloud. This was performed in two ways: a quantitative data comparison and a visual inspection of the renderings. Both comparisons yielded minimal differences. The projected point cloud resulted in a 1.15% data loss in terms of the number of data points. Figure 53 shows renderings of the original point cloud on the left and the projected point cloud on the right.

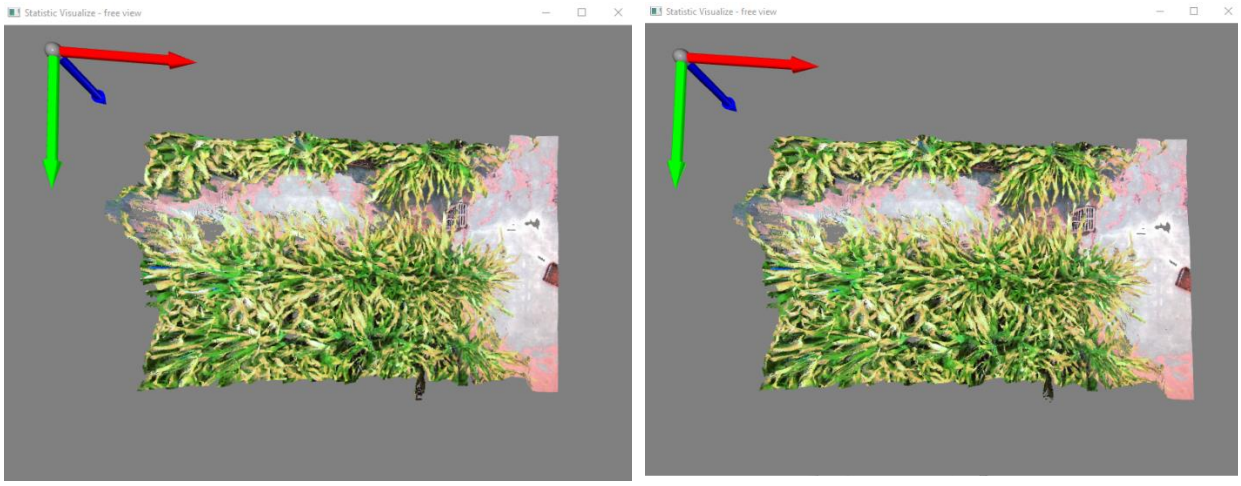

Figure 53. Point Cloud Comparison (pre-transformation on the left, post-transformation on the right)

4.2.2 2D Image from Projection

The images produced from the 2D representations of the point clouds sometimes suffer from having horizontal or vertical lines or large regions with no data. Figure 54 shows the image produced from the 2D representation of the point cloud along with the point cloud for reference.

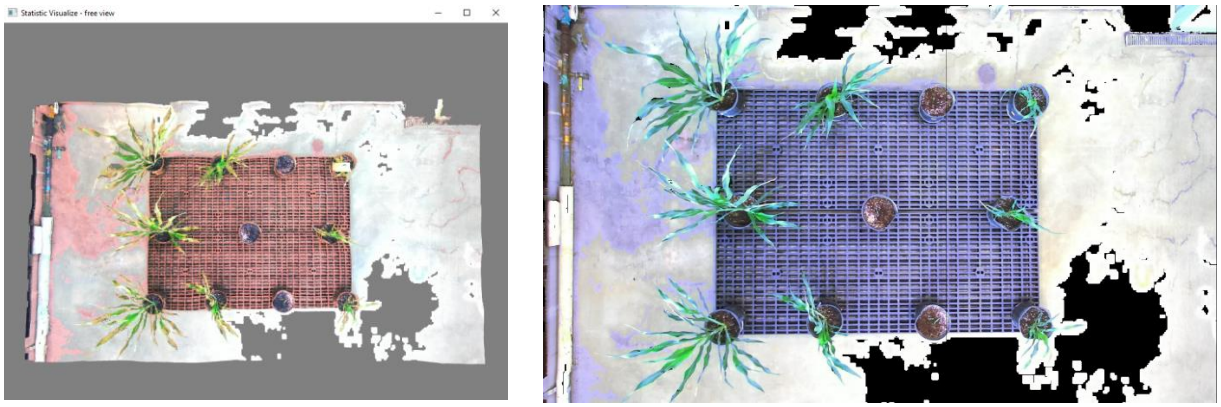

Figure 54. Point Cloud Rendering (left) and Image of 2D Representation of Point Cloud (right)

4.3 Leaf Mask Acquisition with Instance Segmentation Model

4.3.1 Model Prediction

The project uses two methods to validate that the trained model produces predictions that are accurate and effective.

A qualitative visual inspection was performed to measure the predictions' accuracy. The project uses the Matterport Mask-RCNN's "splash" mechanism which uses the mask to superimpose color in a grayscale image. An example is shown in figure 55. All color splash renderings demonstrate results like this one: superimposed masks that correspond with leaves.

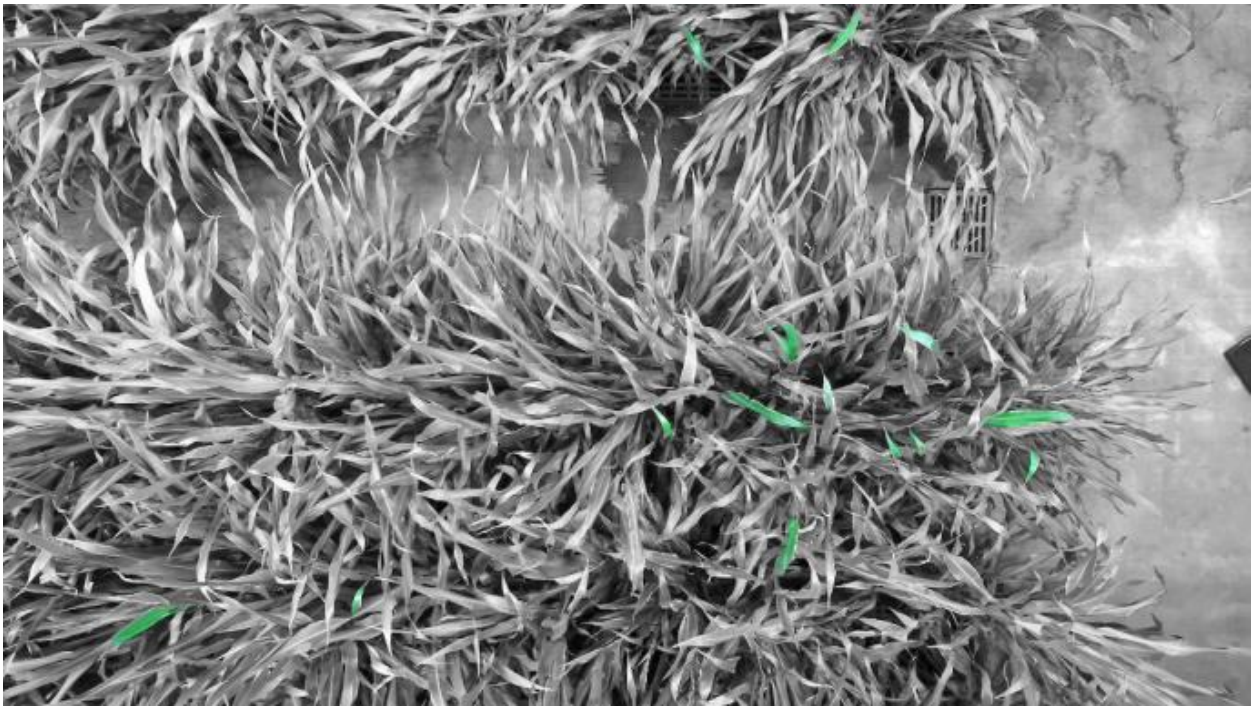

Figure 55. “Color splash” mechanism Applied to Leaf Predictions

A quantitative measure was performed of the model’s IoU (Intersection of Union), mean precision, mean recall, f1 score, and AUC-PR (Area under Curve for Precision-Recall). These metrics measure different aspects of the model’s efficacy. Precision measures the proportion of true positive predictions in the total predicted positives, where high precision indicates the model rarely labels a negative instance as positive (there are few false positives). Recall measures the proportion of positive instances correctly identified, where high recall indicates the model rarely misses labeling a positive instance as positive (there are few false negatives). The F1 score is the harmonic mean of precision and recall, it’s a metric of the model’s balance between high recall and high precision. AUC-PR evaluates the model’s performance across changing thresholds, measuring the impact on precision as recall increases or vice versa.

The IoU across the test image predictions was measured at 6.8%. Figure 57 shows a table with the IoU values of the test image in figure 56.

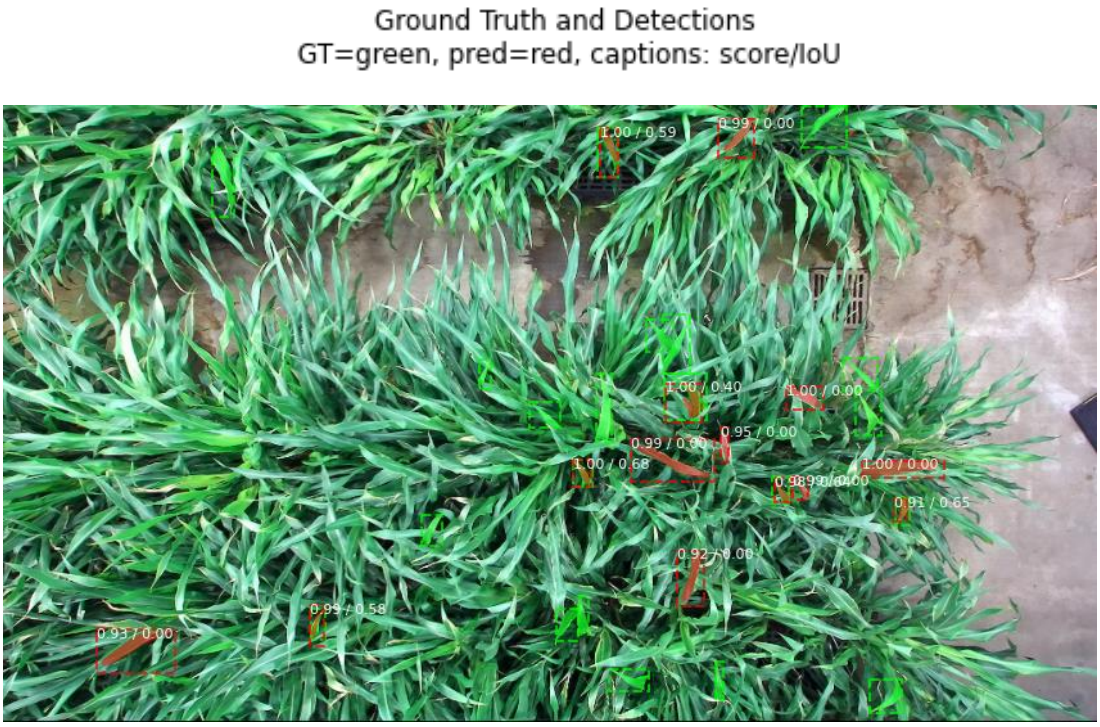

Figure 56. Ground Truths and Predictions Visualized on Image

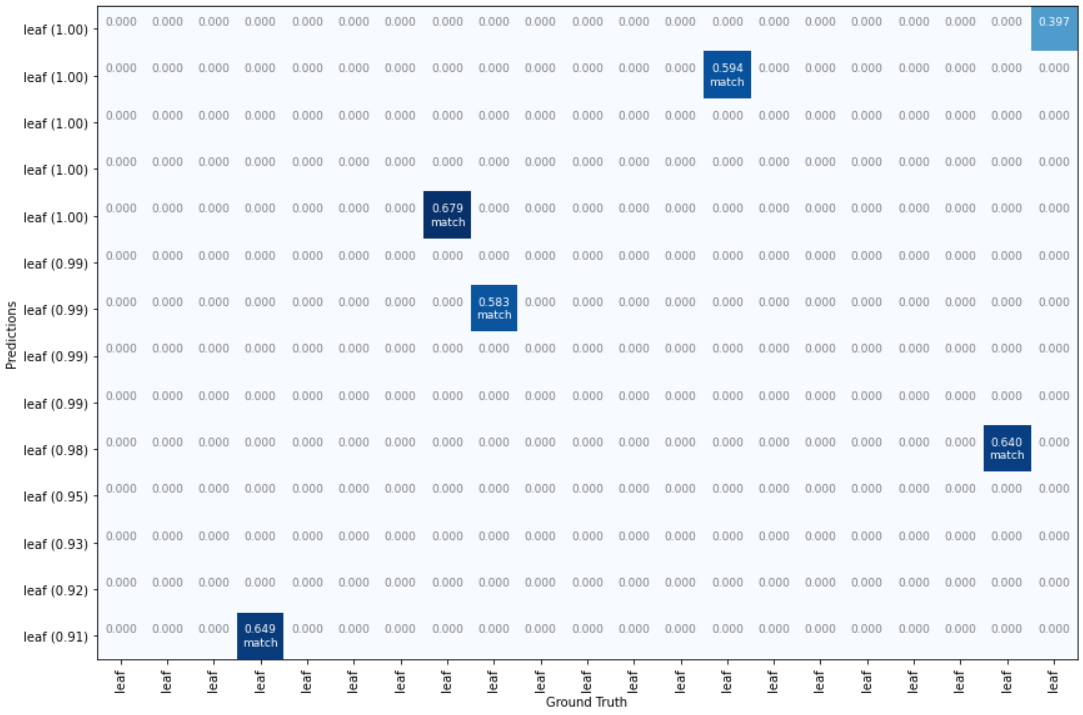

Figure 57. Table Plot of IoU for Predicted and Ground Truth Leaves

Precision, Recall, F1 Score, and AUC-PR were measured across different IoU Thresholds. The data is displayed in Table 5. Figure 58 shows this data in a chart.

Table 5. Precision, Recall, F1 Score, and AUC-PR at Different IoU Thresholds

IoU Threshold	Precision	Recall	F1 Score	AUC-PR
0.0	0.898	0.359	0.513	0.566
0.5	0.584	0.250	0.350	0.228
0.8	0.318	0.193	0.240	0.127

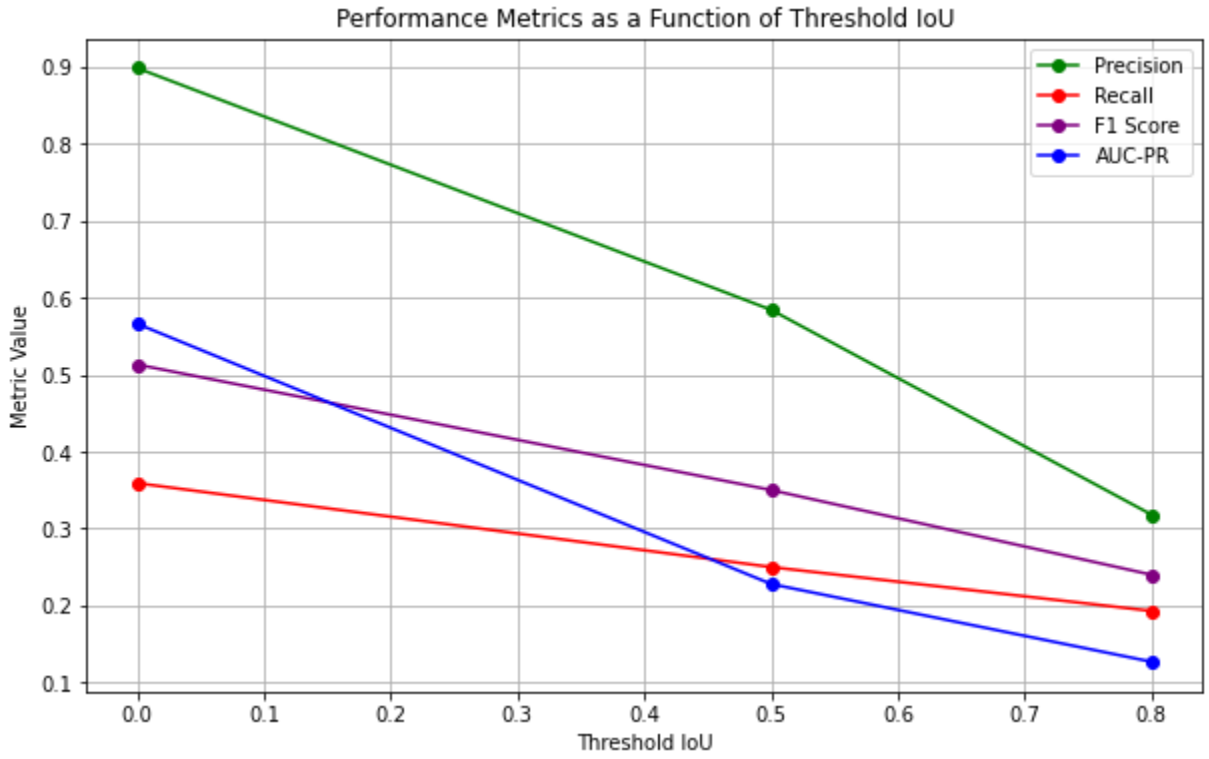

Figure 58. Precision, Recall, F1 Score, and AUC-PR at Different IoU Thresholds

4.3.2 Mask Isolation

In order to validate that an individual leaf mask is being isolated and can be used as a filter, the mask was converted into a black and white image of which the background is white and the mask is black. As figures 59 and 60 demonstrate, the individual masks match exactly with one of the leaf masks represented in the Matterport Mask-RCNN model predictions image.

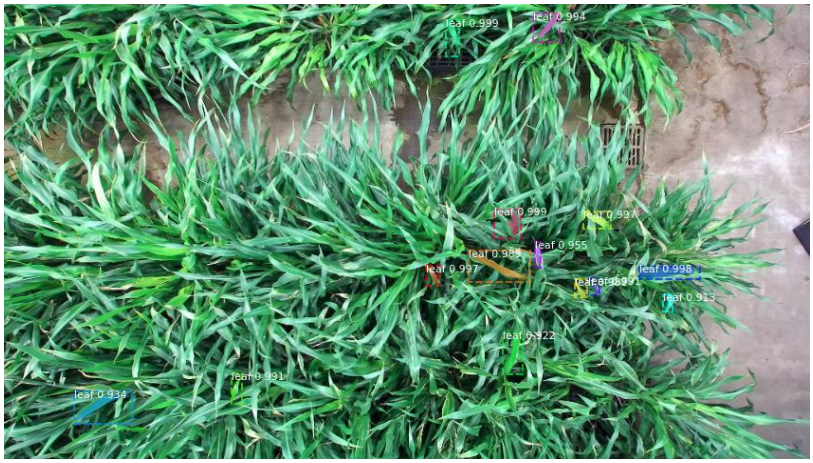

Figure 59. Leaf Masks in Original image

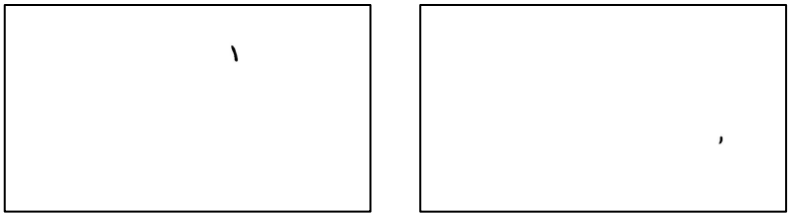

Figure 60. Isolated Leaf Masks

4.5 Leaf Mask Scaling

Validation of the leaf mask scaling was done visually. The red pixels in the image represent the leaf mask. Figure 61 shows the leaf mask before and after scaling where the latter has the red mask properly superimposed over a leaf.

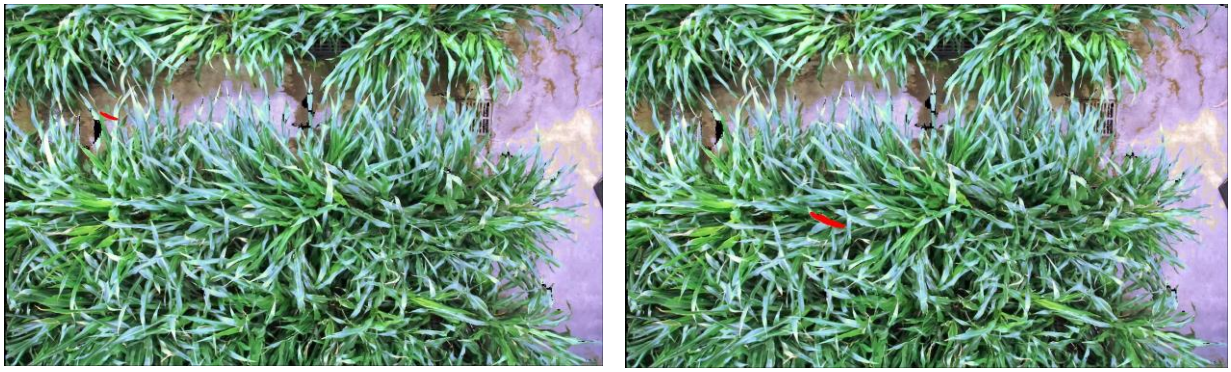

Figure 61. Before Scaling (left) and After Scaling (right)

4.6 Point Cloud Cropping

The point cloud cropping was validated using fabricated inputs with an expected result: a point cloud of a known shape filtered by mask of a known shape produces a point cloud of a known shape. The inputs are shown in figure 62: a hollow cube being filtered with a circle mask. The result of filtering the hollow cube with a circle is as expected and shown in figure 63.

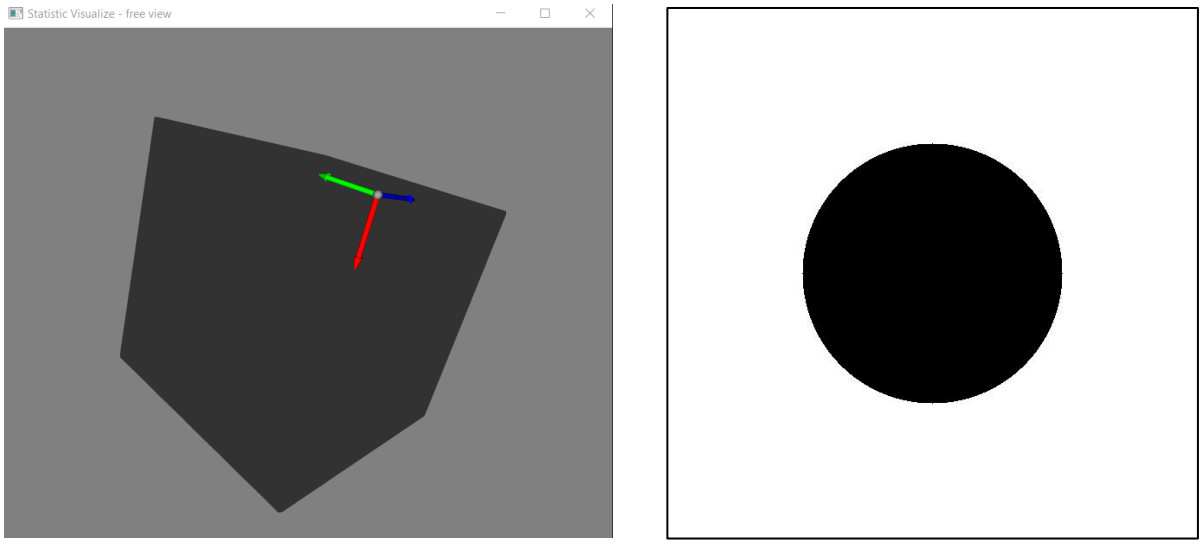

Figure 62. Point Cloud Input (left) and Mask Input (right)

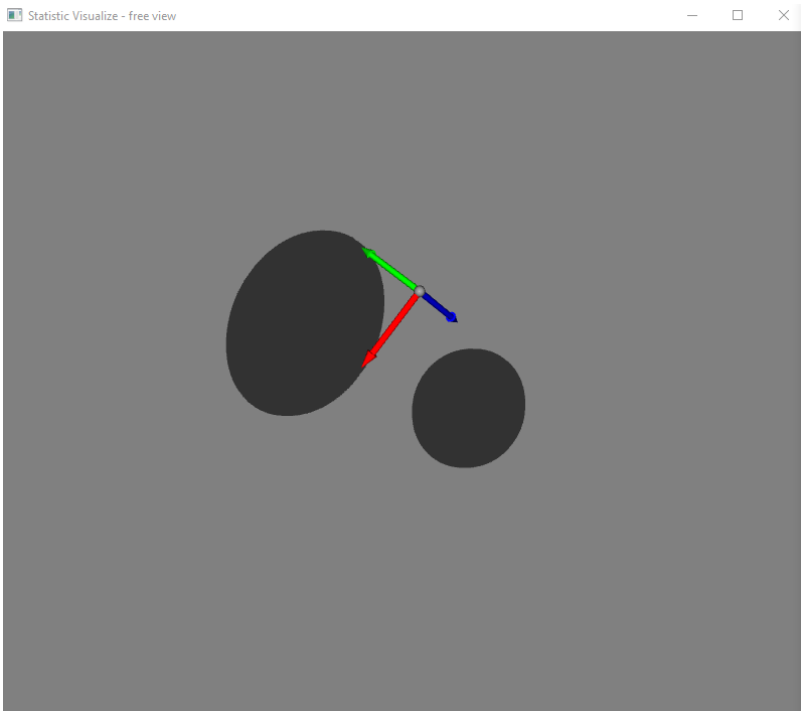

Figure 63. Expected Result

Validation of the accuracy of the cropped leaf was also conducted via visual inspection. Pixels corresponding to the leaf crop were made red. Figure 64 shows a comparison between

- the scaled mask superimposed on the image of the 2D representation of the point cloud (left)
- and the leaf point cloud emphasized with red pixels in a point cloud of the entire scene (right)

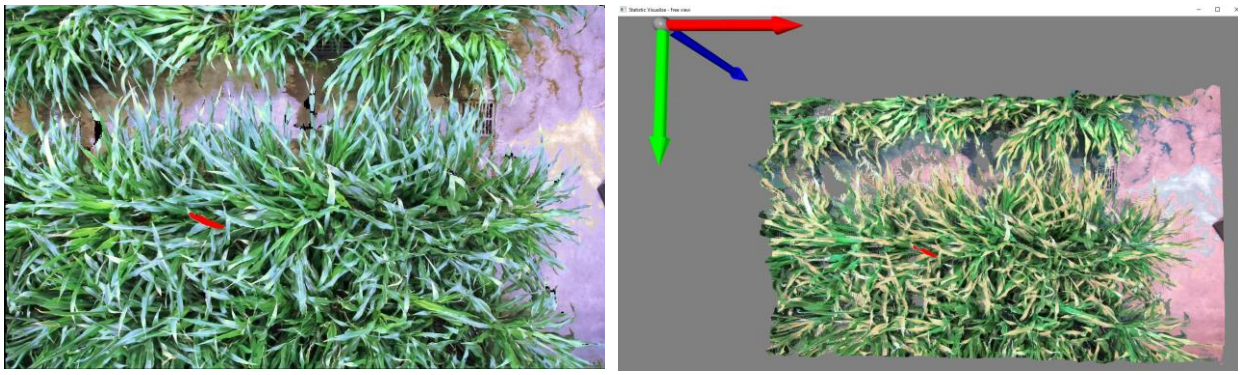

Figure 64. Scaled Mask Emphasized (left) and Leaf Emphasize in Point Cloud (right)

To demonstrate the three-dimensional nature of the cropped point cloud, figure 65 shows the same leaf point cloud from a side view.

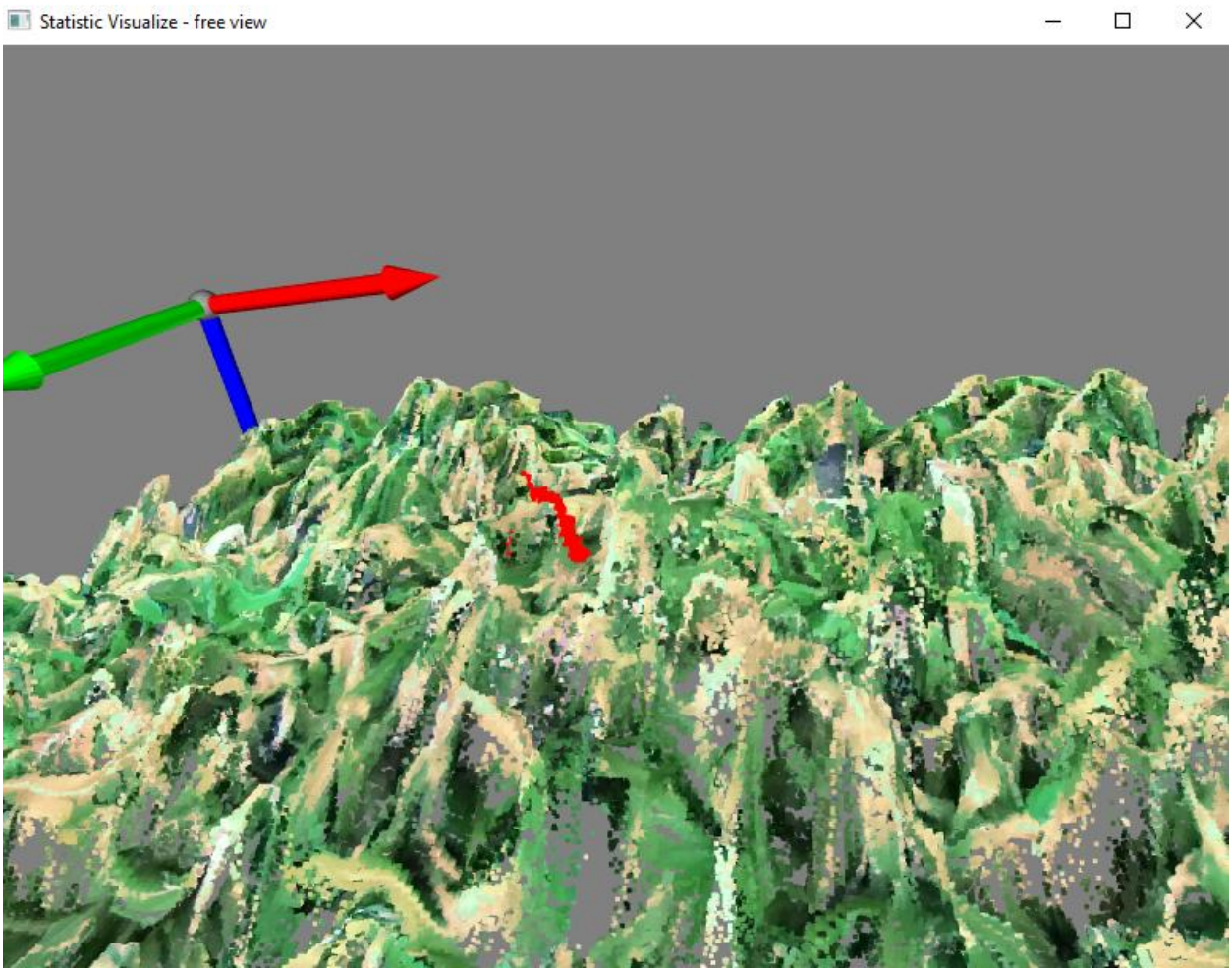

Figure 65. Leaf Point Cloud Side View

5. Discussion

The goal of this research is to define a method and develop a program that researchers can use to acquire leaf models for plants in environments with dense leaf clusters. The results of the project demonstrate successful aspects and areas needing improvement. Although the accuracy of some measurements doesn't make the product of this project ready to be shared, the results give empirical evidence that the method utilized is able to go through the process of acquiring leaf models for plants in environments with dense leaf clusters.

5.1 Data Collection and Rendering

5.1.1 Key Findings

Section 4.1 showcases the findings from the data collection and rendering validation. The results showcase greater inaccuracy as the object moves further from the camera and closer to the edges of the camera's view.

The percentage error of distance (width, length, and depth) measurements at 150 cm from the camera or closer can reach 8.5%. As you get closer to the camera the percentage of error decreases. Using the equation in figure 43 to calculate estimated distance from camera to maintain a percentage error equal to or lower than 5%, measurements must be at 122.13 cm from the camera or closer—assuming the measurements aren't being affected by inaccuracy due to closeness to viewing edge.

Distance measurements closer to the edges of the camera's field of view established a more significant loss of accuracy. At 150cm from the camera lens, measurements taken as close as 85.78 % of the way to the viewing edge of the camera resulted in an 11.56% error. All results measuring for the effect of closeness to the edge of the camera's field of view indicate measurements are inaccurate at these locations. Using equation in figure 47 to calculate estimated closeness to camera to maintain a percentage error equal to or lower than 5%, measurements must be 70.09% percent of the way to the viewing edge of the camera or less—assuming the measurements aren't being affected by inaccuracy due to distance from camera lens.

Taking both variables into account section 4.1.3 demonstrates the compound effect on inaccuracy. If measurements are within certain bounds accurate measurements are possible. Figure 66 below shows that for accuracy measurements within 5% error a closeness of 60% to the edge and a distance from

lens of 100 cm are the bounds. For accuracy measurements within 3% error a closeness of 40% to the edge and a distance from lens of 90 cm are the required bounds.

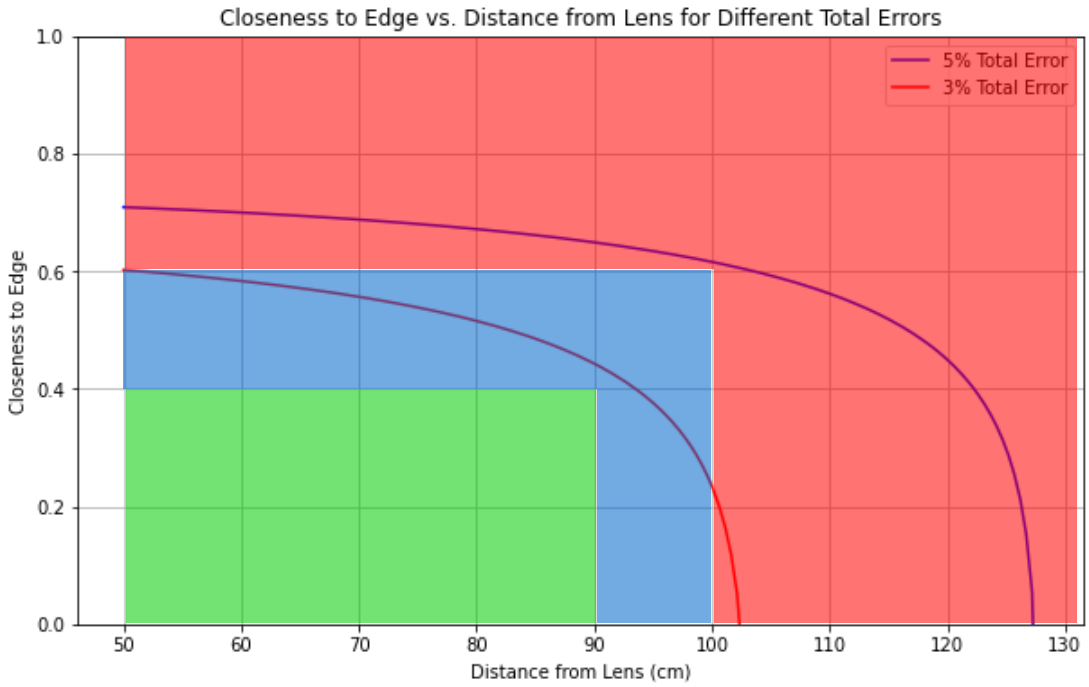

Figure 66. Curve for Closeness to Edge vs Distance From Lens when Total Percent Error is 5%

With this information, it is concluded that if measurements are conducted within the outlined constraints, the accuracy of vision-based measurements with the ZED camera is high and performing vision-based measurements with the ZED camera recordings is reliable.

It is also important to note that in measuring change over time, more important than accuracy is the precision of measurements. Figure 52 in section 4.1.4 demonstrates that across distances, measurements from different images and locations maintain precision. The equation derived from the data estimates that measurements fall within 4.5% relative standard deviation across distances. Since the precision of data is relatively high, even if measurements are taken outside of the accuracy bounds and result in inaccuracy, measurements can be expected to be inaccurate in the same way every time making it reliable to effectively track change even across inaccurate measurements.

5.1.2 Limitations

Although the accuracy of most relevant points is high, there are many limitations observed from the visualization of the produced point clouds.

There is a huge loss of accuracy and sometimes even a loss of data when depth changes occur quickly in the scene. This is demonstrated in figures displaying point clouds in Section 3 of this paper. This is something to consider when working with the ZED Camera in the future: foliage from plants being photographed should be at about the same depth level to maximize recording accuracy.

Another limitation applies to objects closer than 50cm to the camera's lens. The camera's functionality is impeded at these close distances and often objects are ignored, or the coordinates are erroneous. This is visualized in Figures 67, 68, 69, and 70 which depict the same scene from different angles. Figure 67 and 68 show the image from above where the ruler that was placed close to the camera is visible, but as you change the angle you can see in Figure 69 that all the data points that make up the ruler were erroneously marked to be much further from the camera than in reality. Figure 70 shows the profile view of the scene where the blue arrow represents the depth axis. The yellow sphere represents the data point closest to the camera that was collected, which is 51.5 cm away from the camera despite there being objects as close as 25 cm away.

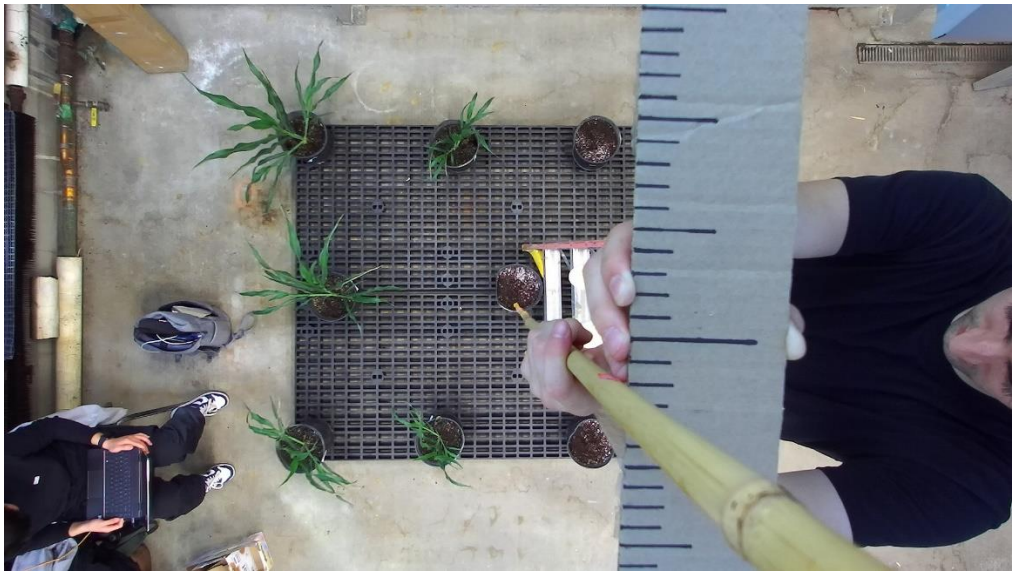

Figure 67. Left Image of the Scene

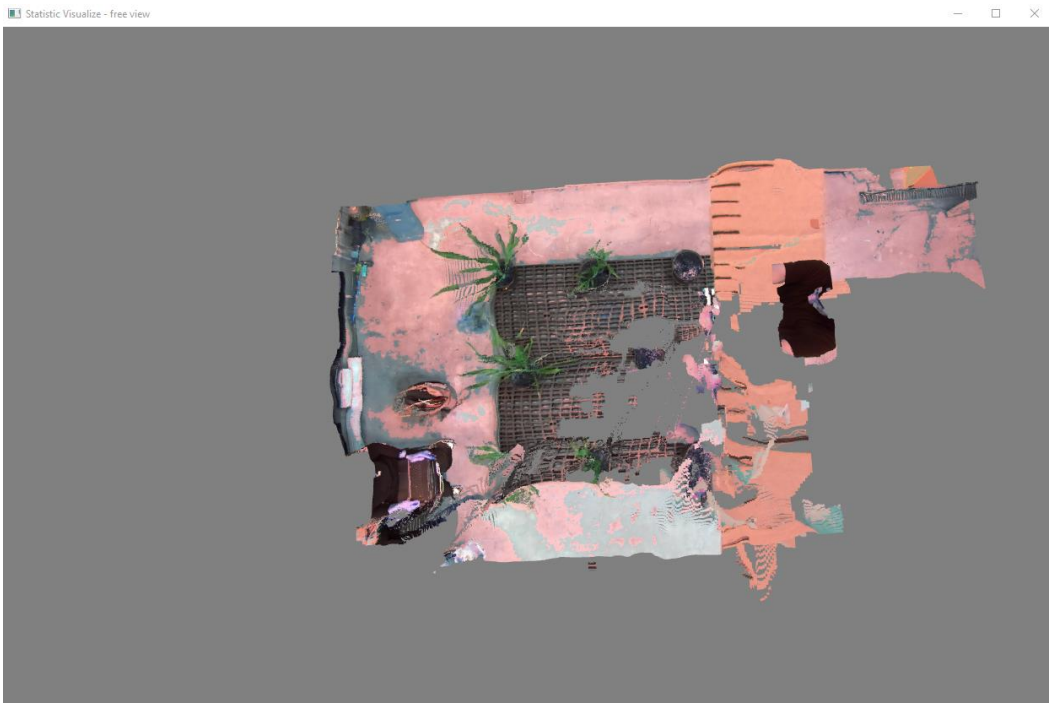

Figure 68. Highest Data Point

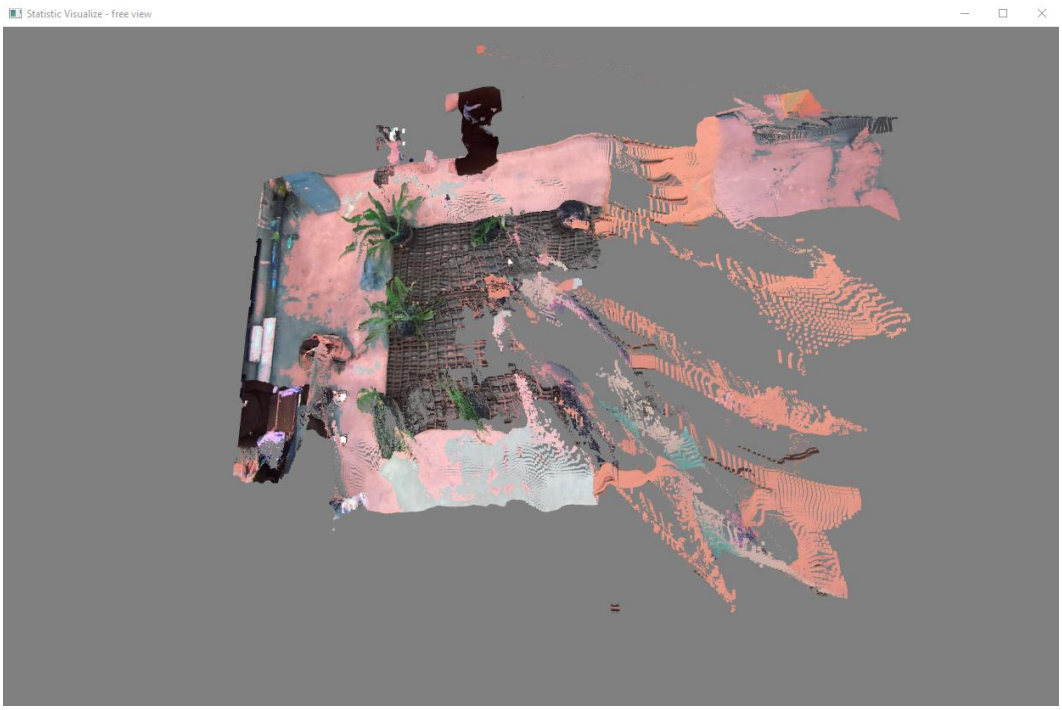

Figure 69. Erroneous Points

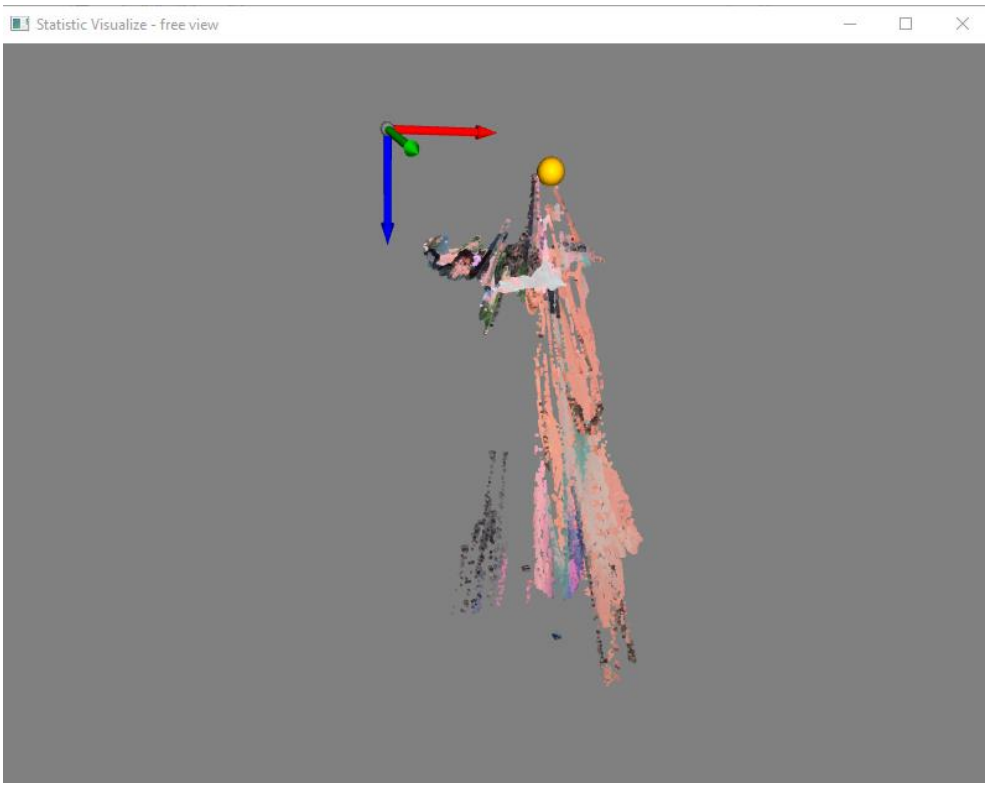

Figure 70. Erroneous Points

5.1.3 Future Work

The ZED SDK provides a lot of tweakable components that can make it behave in a more niche manner. Although this project involved a lot of tweaking to get the ZED Camera to operate correctly and generate precise point clouds, greater experimentation with the parameters in the ZED SDK may yield better results when it comes to the limitations outlined.

Open-source libraries are available that can create point clouds from stereo images. Although the ZED SDK with the ZED Hardware is more likely the highest-performing option when using the ZED Camera, it is worth testing other methods of point cloud generation from stereo images.[12]

[12] Utilizing the Point Cloud Library with the ZED Camera: https://github.com/stereolabs/zed-pcl

5.2 2D Representation of Point Cloud

5.2.1 Key Findings

As shown in Figure 53, the visual inspection shows no noticeable changes in the point cloud. The data loss from projection is due to rounding performed during the interval categorization in the code. The rounding only affects data points at the edge of the camera's field of view and their absence has a very low impact on the quality of the projection.

The black regions in the images are due to regions in the point cloud without data. This can be caused by many factors, most commonly depth changes cause loss of data acquisition with the ZED camera. The black vertical or horizontal lines in the images are due to the way the camera captures images: there are angles in which no data is captured. This is clearer by removing the code that deleted these pixel-wide without-data lines from the image. An example of one of these "unfixed" images is shown in figure 71.

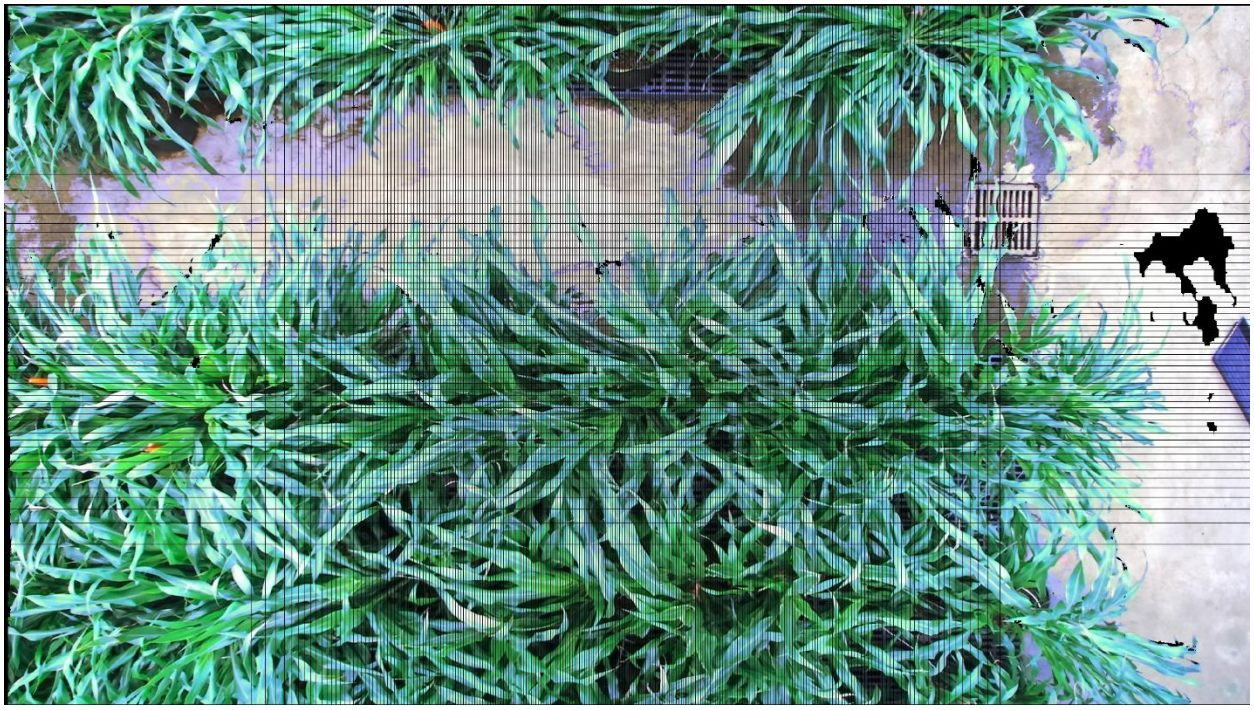

Figure 71. Rendering of Projection with Missing Data Points

Based on the visual comparison and the data loss being less than 1.2% of boundary points, the 3D to 2D projection of point clouds is successful.

5.2.2 Limitations

Due to the nature of using a projection for training the leaf mask model and then for cropping, there is an associated loss of detail. As the scene is in 3D and the desired output is a 3D point cloud, it would be more precise to avoid transformations that may result in losses of data. However, as this project is intended to be one of greater accessibility, this is an agreed upon limitation and one that is difficult to circumvent as 2D instance segmentation is computationally less intense and acquisition of 2D training data is much more accessible and less time consuming than 3D training data.

The images produced from the 2D representations of the point clouds are used for the training of the instance segmentation model. Producing bad images means bad training data, bad model, bad masks, and bad leaf point clouds. Some projections suffer from having horizontal or vertical lines or large regions with no data as demonstrated in figure 54. These issues cascade and affect other parts of the project. Unfortunately, the ZED Camera and Point Cloud Generation technology from stereo images are a limitation in this project's performance.

5.2.3 Future Work

The loss of boundary data points due to rounding errors should be addressed in future work. In addition, the black regions in these projections further showcase generated point clouds that are low quality due to the inability of the data collection phase to capture consistently precise recordings. This is something that was addressed in Section 5.1 and is subject to future work.

5.3 Leaf Mask Acquisition with Instance Segmentation Model

5.3.1 Key Findings

The results of the model's predictions showcase underwhelming findings.

The model's mean IoU is extremely small at 6.8%.

At an IoU threshold of 0.0, the mean precision is higher at 89.8% and the mean recall is lower at 35.9%. This conveys that when it comes to leaf detection (not segmentation), there are very few false positives and many false negatives, meaning the model doesn't detect enough leaves compared to ground truth.

As the IoU threshold increases, precision, recall, f1 score, and AUC-PR scores decrease. When the IoU threshold is 0.5, mean precision is at 58.4%, mean recall at 25%, mean f1 score at 35%, and mean AUC-

PR at 22.8%. This conveys that as the threshold of what constitutes a true prediction increases to need a greater proportion of intersection to union, every metric of efficacy decreases. In other words, the model performs poorly for segmentation tasks, the segmentation masks do not accurately fit the leaves as described by the ground truth annotations.

Despite the model's lack of efficacy, color splash renderings of the model's predictions such as the one shown in figure 55 show that it's able to predict leaf masks despite these setbacks. And although the masks may have some segmentation errors, the model and its predictions serve as a first step in an effective leaf instance segmentation model based on Mask R-CNN.

The results of the mask isolation as shown in figure 61 are as expected and show successful isolation of masks using the model's predictions.

5.3.2 Limitations

Due to the nature of the project in serving as a first step in comprehensive leaf point cloud generation, during annotation leaves that were curved or seemed obstructed were excluded from annotation. This line of exclusion was not well defined and most likely influenced unexpected behavior in the model. This resulted in lower number of predictions than desired.

This model was trained using 54 annotated images with low amounts of diversity which substantially limits the model in being able to perform well in diverse environments. Applying this model to a new environment and expecting accurate results would require additional annotated images and retraining the model.

5.3.3 Future Work

Improvements can be made in three different areas: general accuracy, number of predicted masks, and environmental diversity the model can be applied to. All of these can be improved with a larger dataset that includes more annotated leaves and more varied environments.[13]

In addition to creating a larger dataset, tweaking the model to work best with leaves should be explored. Since this project was more process focused, deeper exploration into Mask R-CNN and

[13] A publicly available sorghum dataset: https://www.kaggle.com/competitions/sorghum-id-fgvc-9/data?select=sample_submission.csv

tweaking the algorithm was not explored. Future work in this area could help the instance segmentation perform with higher accuracy.

5.4 Leaf Mask Scaling

5.4.1 Key Findings

Across all the different test images, the leaf masks scaled accordingly and were visually shown to be superimposed on the appropriate leaf.

5.4.2 Limitations

Due to the nature of upscaling the leaf masks with nearest-neighbor interpolation, scaled masks appear blockier than ideal and are not able to maintain the appropriate leaf boundaries. This results in less accurate masks and in less accurate point clouds of leaves.

5.4.3 Future Work

To avoid the side effects that come from upscaling, there are two options: downscale the image and point cloud or remove the need for scaling. Downscaling the point cloud leads to a reduction in points, possible transformation side effects, and as a result a leaf point cloud that would provide less accurate phenotype measurements. To remove the need for scaling, the instance segmentation model needs to be altered to no longer compress or resize the image during pre-processing.

5.5 Point Cloud Cropping

5.5.1 Key Findings

The validation using the hollow cube and circle mask proves the point cloud cropping portion of the project reliable. The resulting cropped point cloud demonstrates that the mask was accurately used as a filter on the hollow cube.

Figures 64 and 65 show the point cloud cropping of a leaf. The scene is effectively reversed from being a 2D representation of a point cloud to a point cloud. The top-down and profile view demonstrates the points are continuous and it appears to be a leaf. These images demonstrate that a leaf can be accurately cropped from the point cloud of the scene.

5.5.2 Limitations

Limitations in the point cloud cropping results arise from performing segmentation and manipulations on two dimensional representations of the scene. Points that may not be a continuous part of the leaf may be included in the leaf mask and subsequently in the cropped point cloud. This is demonstrated with figure 72 which is a zoomed in screenshot of figure 65 (profile view of a leaf mask). This figure showcases seemingly disjointed parts on the left vs the right, these are not all part of one leaf. However, this is not clear from the top-down view shown in figure 37 in section 3.6. As a result, the instance segmentation model is not able to differentiate these points as pertaining to separate leaves. This is a limitation from performing the instance segmentation on a 2D representation of the three-dimensional.

Figure 72. Close View of Leaf (red pixels denote leaf)

5.5.3 Future Work

To fix the discontinuous nature of some cropped point clouds, all points that are not a part of the largest continuously connected region in a leaf mask should be removed from the leaf mask. This way, smaller regions outside of the boundaries of the leaf that are misclassified do not interfere with phenotype measurements.

Another potential way of fixing incorrect semantic segmentation performed by the model using 2D representations of the scene could be to use the 3D representations of the leaf masks to inform the model whether the mask is correctly segmented or not. A method of accomplishing this would be to use

multi-view CNNs. Multiple projections of the 3D objects into 2D images from different viewpoints would allow more comprehensive feature extraction from the scene, and greater accuracy when producing leaf masks. This method still makes use of traditional 2D CNNs which are more highly efficient than 3D methods of instance segmentation. [5]

6. Conclusion

6.1 Research Answers

The research aim for this research project is to enable researchers who are conducting experiments on plants that require the collection of phenotyping data to perform the acquisition of data more quickly and precisely. Specifically, by defining a method and developing a program that researchers can use to acquire leaf point clouds for plants in environments with dense leaf clusters. And these point clouds can be used to easily extract phenotype data.

The research addressed the question posed in the introduction relating to the design of the project.[14] An effective process at isolation and modeling leaves from stereo images was developed, a diagram showcasing this process is shown in figure 73 below. And this process was implemented successfully, as discussed in sections 3, 4, and 5 of this paper.

[14] For more information read section '1.2' of this paper

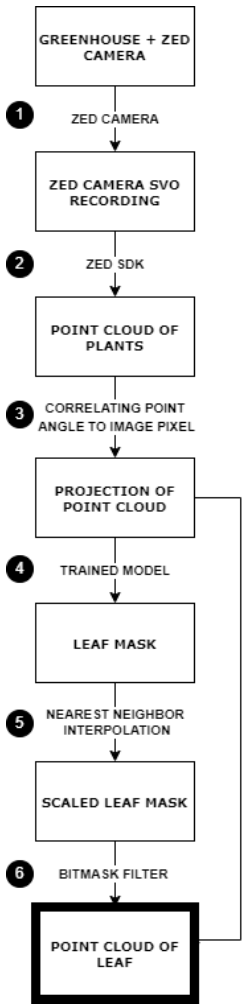

Figure 73. Project Workflow

Each step along this process was validated and demonstrated to be working in this paper. As discussed in section 5, the point clouds generated from the SVO recordings captured by the ZED Camera were shown to be accurate within certain boundaries and precise throughout the imaged environment. The transformation in acquiring a 2D representation of the point cloud had minute amounts of data loss and was shown to be an accurate portrayal of the scene by rendering an image of the representation. The model trained on leaf instance segmentation was demonstrated to be sufficiently effective for the purposes of this project. Mask scaling was visually validated to be effective and point cloud cropping passed the predefined tests. After point cloud cropping, the project yielded accurate leaf point clouds that can be used for measuring phenotype data such as leaf width.

The product of this project is a reproducible repository that, provided with a ZED Camera and ZED Box, can capture recordings of leaves and generate point clouds of these individual leaves.

6.2 Contribution

This project provides plant researchers with a tool to facilitate measuring phenotype data of sorghum leaves in their studies. This tool facilitates plant researchers in conducting their studies quicker, more accurately, and on a larger scale. One example of the studies this project can help with is one being conducted at the University of Illinois where plant genetic researchers are working on genetically modifying Sorghum to be more resistant to the changes caused by climate change. The greater success of these studies will help prepare humanity for the worsening climate including assisting the potential food scarcity issue.

Although this project was specifically designed for obtaining sorghum leaf point clouds for plant genetic researchers, the process and program can be altered and applied to other applications requiring measurements of plants, like maize, and even other objects. This project serves as a baseline for future development and process improvement in the space of object measurement with stereo images. And though the project has some flaws it serves as a first step in the construction of widely applicable object measurement capabilities.

6.3 Further Research Opportunities

6.3.1 Improvements

After completion of the data collection phase and obtaining the necessary recordings, the point clouds generated from these recordings demonstrated a lack of data in certain recordings. The primary culprit of recordings with insufficient data was depth changes in the scene being recorded. These gaps in data points result in obtaining fewer leaf point clouds from a scene and impedes researchers' ability in tracking phenotype changes in the leaves. Further research improving the acquired recordings from the ZED Camera may be beneficial. It may also be beneficial to look into other stereo cameras or other methods of generating point clouds without the use of the ZED SDK. There is available open-source software that can generate point clouds from stereo images and is worth exploring.

The transformation of the point cloud to a two-dimensional representation of itself has some rounding issues that cause some data loss at the boundaries of the point cloud. Further research is recommended in fixing these boundary rounding errors.

As this project served as a first step in the direction of Leaf Point Cloud acquisition from stereo images, leaves that were heavily obstructed or curved were disregarded during annotation and thus disregarded

in the model's predictions of leaves. Further research is recommended to expand the model's definition of a leaf and include these more obstructed and curved leaves. Apart from adding these types of leaves into the dataset, a generally larger dataset is recommended for a more effective model. This project involved a small dataset with low amounts of diversity, further research would expand the dataset. In addition to creating a larger dataset, tweaking the model to work best with leaves should be explored.

Due to the nature of upscaling the leaf masks with nearest-neighbor interpolation, scaled masks appear blockier than ideal and are not able to maintain the appropriate leaf boundaries. The instance segmentation model needs to be altered to no longer compress or resize the image during pre-processing so that upscaling of leaf masks is not needed.

Points that may not be a continuous part of the leaf are sometimes included in the leaf mask and subsequently in the cropped point cloud. Further research is needed to remove segments included in these leaf point clouds that are not a part of the targeted leaf. A possible avenue is to target the incorrect semantic segmentation performed by the model using 2D representations of the scene. Further research could change the model so that 3D representations of the leaf masks can inform the model on correct semantic segmentation decisions. A method of accomplishing this would be to use multi-view CNNs.

6.3.2 Additions

To utilize this project for phenotype data extraction of leaves in a scene, a few additional steps are required. This sub section will be an overview of those steps, these are shown in figure 74.

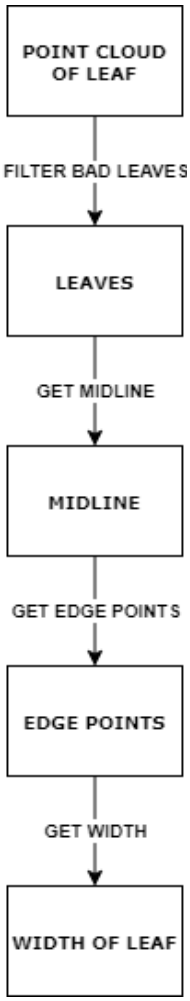

Figure 74. Next Steps Workflow

Once the point cloud of a leaf is acquired, these point clouds should undergo a filtering process to prevent erroneous data from influencing the perceived growth rate. Once bad point clouds are filtered out, a smoothing algorithm should be applied to the leaf to reduce noise and irregularities. Li et al make use of the Moving Least Squares method to perform local fitting and surface estimation on the points on the leaf to smooth the leaf. [15] Once the leaf is smoothed, the process of acquiring leaf width can begin.

Wang et al proposes to first use Principal Component Analysis to get axes of the leaf. Once axes are established that define the length, width, and depth of the leaf, establish a bounding box as tightly as possible around the leaf. Using the edge points that intersect with the bounding box, measure length, width, and depth through cross-section method or geodesic distance method. [16] Calculation of leaf width via the geodesic distance of the corresponding two edge points is more accurate but much less

efficient than the cross-section method. Further research can involve discussing the trade off and making a choice between the two.

Li. et al proposes a different method in which a “midrib” is determined by calculating the smallest path between the two furthest points in the leaf point cloud. Then width can be calculated by getting the smallest path between the two furthest points to the plane corresponding to the first midrib. [15]

Both methods rely on the entirety of the leaf being present in the point cloud, as opposed to a portion of it. To guarantee this, additional research in improving detection and segmentation of obstructed leaves is required.

Appendix A: Project Code

The code excerpts in this section are missing certain lines of code for ease of reading and simplification.

A.1 Data Collection

```
cam = sl.Camera()

init = sl.InitParameters()
init.camera_resolution = sl.RESOLUTION.HD2K
init.depth_mode = sl.DEPTH_MODE.ULTRA
init.SDK_verbose = 1
init.SDK_verbose_log_file = "/home/user/Documents/SVO_recording/SDK_log_file/log_file"
# other initial parameters have been left blank because their default value is adequate

recording_param = sl.RecordingParameters(path_output, sl.SVO_COMPRESSION_MODE.H265)

runtime = sl.RuntimeParameters()
cam.set_camera_settings(sl.VIDEO_SETTINGS.BRIGHTNESS, -1)
cam.set_camera_settings(sl.VIDEO_SETTINGS.CONTRAST, -1)
cam.set_camera_settings(sl.VIDEO_SETTINGS.HUE, -1)
cam.set_camera_settings(sl.VIDEO_SETTINGS.SATURATION, -1)
cam.set_camera_settings(sl.VIDEO_SETTINGS.SHARPNESS, -1)
cam.set_camera_settings(sl.VIDEO_SETTINGS.GAMMA, -1)
cam.set_camera_settings(sl.VIDEO_SETTINGS.GAIN, 0)
cam.set_camera_settings(sl.VIDEO_SETTINGS.EXPOSURE, 5)
cam.set_camera_settings(sl.VIDEO_SETTINGS.WHITEBALANCE_TEMPERATURE, -1)
# other video settings have been left blank because their default value is adequate

frames_recorded = 0

while frames_recorded < 100:
    if cam.grab(runtime) == sl.ERROR_CODE.SUCCESS :
        frames_recorded += 1
```

Figure 75. Capturing an SVO recording with the ZED Camera

A.2 Point Cloud Generation

```
cam = sl.Camera()
runtime = sl.RuntimeParameters()
mat = sl.Mat()

cam.retrieve_image(mat, sl.VIEW.LEFT)
cv2.imwrite(output_path_image, mat.get_data())
cam.retrieve_measure(mat, sl.MEASURE.XYZRGBA)

pc_list = []

for i in range(0, mat.get_width()):
    for j in range(0, mat.get_height()):
        err, val = mat.get_value(i,j)
        if not np.isnan(val[0]) and not np.isinf(val[0]):
            colorVal = ''.join('{:0>8b}'.format(c) for c in struct.pack('!f', val[3]))
            A, R, G, B = (colorVal[0:8], colorVal[8:16], colorVal[16:24], colorVal[24:32])
            pc_list.append([val[0], val[1], val[2], int(R, 2), int(G, 2), int(B, 2), int(A, 2)])
```

Figure 76. Code for utilizing the ZED SDK to process the SVO recordings into point clouds

A.3 2D Representation of Point Cloud

```
left_image = cv2.imread(left_image_path)
height, width, _ = left_image.shape
img = np.zeros([height, width, 3])
point_cloud = np.empty((height, width), dtype=object)
for i in range(height):
    for j in range(width):
        point_cloud[i, j] = []
overlap = np.zeros([height, width])
with open(point_cloud_path, mode="r") as file:
    reader = csv.reader(file)
    max_x_degree, min_x_degree, max_y_degree, min_y_degree = getMaxDegrees(reader)
with open(point_cloud_path, mode="r") as file:
    reader = csv.reader(file)
    anglesToIndexLoop(
        reader,
        img,
        point_cloud,
        overlap,
        width,
        height,
        max_x_degree,
        min_x_degree,
        max_y_degree,
        min_y_degree)

for r in range(0, height):
    for c in range(0, width):
        if overlap[r, c] > 1:
            img[r, c, 0] /= overlap[r, c]
            img[r, c, 1] /= overlap[r, c]
            img[r, c, 2] /= overlap[r, c]
```

Figure 77. Setup for 3D to 2D Projection

```
 1. def anglesToIndex(x_degree, y_degree, width, height, max_x_degree, min_x_degree, max_y_degree,
min_y_degree):
 2.     x_interval = (max_x_degree + abs(min_x_degree)) / width
 3.     y_interval = (max_y_degree + abs(min_y_degree)) / height
 4.     x_shifted = x_degree + ((max_x_degree + abs(min_x_degree)) / 2)
 5.     if x_shifted < 0:
 6.         x_shifted = -1
 7.     y_shifted = y_degree + ((max_y_degree + abs(min_y_degree)) / 2)
 8.     if y_shifted < 0:
 9.         y_shifted = -1
10.     x_pos = (x_shifted) / x_interval
11.     y_pos = (y_shifted) / y_interval
12.     if x_pos > width - 1:
13.         x_pos = -1
14.     if y_pos > height - 1:
15.         y_pos = -1
16.     return int(x_pos), int(y_pos)
17.
18. def anglesToIndexLoop(reader, img, point_cloud, overlap, width, height, max_x_degree,
min_x_degree, max_y_degree, min_y_degree):
19.     first_line = True
20.     for lines in reader:
21.         if first_line:
22.             first_line = False
23.             continue
24.         x_coord = float(lines[0])
25.         y_coord = float(lines[1])
26.         z_coord = float(lines[2])
27.         r = int(lines[3])
28.         g = int(lines[4])
29.         b = int(lines[5])
30.         x_degrees = math.degrees(math.atan(x_coord / z_coord))
31.         y_degrees = math.degrees(math.atan(y_coord / z_coord))
32.         x_pos, y_pos = anglesToIndex(
33.             x_degrees,
34.             y_degrees,
35.             width,
36.             height,
37.             max_x_degree,
38.             min_x_degree,
39.             max_y_degree,
40.             min_y_degree,
41.         )
42.         if x_pos == -1 or y_pos == -1:
43.             continue
44.         img[y_pos, x_pos, 0] += r
45.         img[y_pos, x_pos, 1] += g
46.         img[y_pos, x_pos, 2] += b
47.         point_cloud[y_pos, x_pos].append(lines)
48.         overlap[y_pos, x_pos] += 1
```

Figure 78. Functions that are called for purposes of 3D to 2D projection

A.4 Leaf Mask Acquisition with Instance Segmentation

```
config = leaves_model.LeavesConfig()
config = InferenceConfig()

# Load validation dataset
dataset = leaves_model.LeavesDataset()
dataset.load_leaves(LEAVES_MODEL_DIR, "val")
dataset.prepare()

# Create model in inference mode
with tf.device(DEVICE):
    model = modellib.MaskRCNN(mode="inference", model_dir=MODEL_DIR, config=config)

model.load_weights(weights_path, by_name=True)

for image_id in image_ids:
    image, image_meta, gt_class_id, gt_bbox, gt_mask = modellib.load_image_gt(
        dataset, config, image_id, use_mini_mask=False
    )
    info = dataset.image_info[image_id]
    imageName = info["id"]

    # Run object detection
    results = model.detect([image], verbose=1)
    r = results[0]
    rawMasks = r["masks"]

    # define
    masks = {}
    numMasks = len(rawMasks[0][0])
    for i in range(0, numMasks):
        key = "mask_" + imageName + "_num_" + str(i)
        masks[key] = []

    listKeys = list(masks.keys())

    # populate
    for rowIndex, rowValue in enumerate(rawMasks):
        for key in listKeys:
            masks[key].append([])
        for columnIndex, columnValue in enumerate(rowValue):
            for key in listKeys:
                masks[key][rowIndex].append([])
            for numMask, maskValue in enumerate(columnValue):
                key = "mask_" + imageName + "_num_" + str(numMask)
                masks[key][rowIndex][columnIndex].append(maskValue)

    # resize mask
    height_mask, width_mask, _ = r["masks"].shape

    display_instances_rois = np.zeros((1, 4))
    display_instances_masks = np.zeros((height_mask, width_mask, 1))
    display_instances_class_ids = np.zeros((1))
    display_instances_class_names = dataset.class_names
```

Figure 79. Mask Prediction with Trained Model

```
augmentation = iaa.SomeOf((4), [
    iaa.Fliplr(0.5),
    iaa.Flipud(0.5),
    iaa.OneOf([iaa.Affine(rotate=90),
               iaa.Affine(rotate=180),
               iaa.Affine(rotate=270)]),
    iaa.Multiply((0.8, 1.5)),
    iaa.GaussianBlur(sigma=(0.0, 5.0)),
    iaa.WithColorspace(
        to_colorspace="HSV",
        from_colorspace="RGB",
        children=iaa.WithChannels(0, iaa.Add((0, 50))))
])
```

Figure 80. Image Augmentation Configuration

A.5 Leaf Mask Scaling

```
height, width, _depth = np.array(mask).shape

# Calculate scaling factors
height_scale = new_height / height
width_scale = new_width / width

# Scale the mask using nearest-neighbor interpolation
scaled_mask = zoom(mask, (height_scale, width_scale, 1), order=0, mode='nearest')

# Convert the scaled mask to boolean values (True/False)
scaled_mask = scaled_mask >= 1

```

Figure 81. Leaf Mask Scaling Code

A.6 Point Cloud Cropping

```
# list of each leaf mask on the image
list_keys = list(masks.keys())

projected_image_height, projected_image_width = np.array(projected_point_cloud_data).shape

for mask_key in list_keys:
    cropped_pc = []
    emphasized_pc = []

    for height_coordinate in range(0, projected_image_height):
        for width_coordinate in range(0, projected_image_width):
            if masks[mask_key][height_coordinate][width_coordinate][0]:
                for point in projected_point_cloud_data [height_coordinate][width_coordinate]:
                    cropped_pc.append(point)
                    emphasized_pc.append(color_point(point))
            else:
                for point in projected_point_cloud_data [height_coordinate][width_coordinate]:
                    emphasized_pc.append(point)

```

Figure 82. Point Cloud Cropping Code

Appendix B: Instance Segmentation Model Parameters

Table 6. Configurations parameters for Matterport's Mask R-CNN Model [17]

BACKBONE	resnet101
BACKBONE_STRIDES	[4, 8, 16, 32, 64]
BATCH_SIZE	1
BBOX_STD_DEV	[0.1 0.1 0.2 0.2]
COMPUTE_BACKBONE_SHAPE	None
DETECTION_MAX_INSTANCES	100
DETECTION_MIN_CONFIDENCE	0.9
DETECTION_NMS_THRESHOLD	0.3
FPN_CLASSIF_FC_LAYERS_SIZE	1024
GPU_COUNT	2
GRADIENT_CLIP_NORM	5
IMAGES_PER_GPU	1
IMAGE_CHANNEL_COUNT	3
IMAGE_MAX_DIM	1024
IMAGE_META_SIZE	14
IMAGE_MIN_DIM	800
IMAGE_MIN_SCALE	0
IMAGE_RESIZE_MODE	square
IMAGE_SHAPE	[1024 1024 3]
LEARNING_MOMENTUM	0.9
LEARNING_RATE	0.001
LOSS_WEIGHTS	{'rpn_class_loss': 1.0, 'rpn_bbox_loss': 1.0, 'mrcnn_class_loss': 1.0, 'mrcnn_bbox_loss': 1.0, 'mrcnn_mask_loss': 1.0}
MASK_POOL_SIZE	14
MASK_SHAPE	[28, 28]
MAX_GT_INSTANCES	100
MEAN_PIXEL	[123.7 116.8 103.9]
MINI_MASK_SHAPE	(56, 56)
NAME	leaves_zed
NUM_CLASSES	2
POOL_SIZE	7
POST_NMS_ROIS_INFERENCE	1000
POST_NMS_ROIS_TRAINING	2000
PRE_NMS_LIMIT	6000
ROI_POSITIVE_RATIO	0.33
RPN_ANCHOR_RATIOS	[0.5, 1, 2]
RPN_ANCHOR_SCALES	(32, 64, 128, 256, 512)
RPN_ANCHOR_STRIDE	1

RPN_BBOX_STD_DEV	[0.1 0.1 0.2 0.2]
RPN_NMS_THRESHOLD	0.7
RPN_TRAIN_ANCHORS_PER_IMAGE	256
STEPS_PER_EPOCH	100
TOP_DOWN_PYRAMID_SIZE	256
TRAIN_BN	FALSE
TRAIN_ROIS_PER_IMAGE	200
USE_MINI_MASK	TRUE
USE_RPN_ROIS	TRUE
VALIDATION_STEPS	50
WEIGHT_DECAY	0.0001

References

[1] J. Weyler, F. Magistri, P. Seitz, J. Behley, and C. Stachniss, "In-Field Phenotyping Based on Crop Leaf and Plant Instance Segmentation," *Proc. - 2022 IEEE/CVF Winter Conf. Appl. Comput. Vision, WACV 2022*, pp. 2968–2977, 2022, doi: 10.1109/WACV51458.2022.00302.

[2] G. C. Nelson *et al.*, "Climate change effects on agriculture: Economic responses to biophysical shocks," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 111, no. 9, pp. 3274–3279, Mar. 2014, doi: 10.1073/PNAS.1222465110/SUPPL_FILE/SD03.TXT.

[3] Y. Li *et al.*, "Automatic organ-level point cloud segmentation of maize shoots by integrating high-throughput data acquisition and deep learning," *Comput. Electron. Agric.*, vol. 193, p. 106702, Feb. 2022, doi: 10.1016/J.COMPAG.2022.106702.

[4] J. Schrader, G. Pillar, and H. Kreft, "Leaf-IT: An Android application for measuring leaf area," *Ecol. Evol.*, vol. 7, no. 22, pp. 9731–9738, Nov. 2017, doi: 10.1002/ECE3.3485.

[5] D. Li *et al.*, "PlantNet: A dual-function point cloud segmentation network for multiple plant species," *ISPRS J. Photogramm. Remote Sens.*, vol. 184, pp. 243–263, Feb. 2022, doi: 10.1016/J.ISPRSJPRS.2022.01.007.

[6] K. Turgut, H. Dutagaci, G. Galopin, and D. Rousseau, "Segmentation of structural parts of rosebush plants with 3D point-based deep learning methods," *Plant Methods*, vol. 18, no. 1, pp. 1–23, Dec. 2022, doi: 10.1186/S13007-022-00857-3/TABLES/11.

[7] L. Xu, Y. Li, Y. Sun, L. Song, and S. Jin, "Leaf instance segmentation and counting based on deep object detection and segmentation networks," *Proc. - 2018 Jt. 10th Int. Conf. Soft Comput. Intell. Syst. 19th Int. Symp. Adv. Intell. Syst. SCIS-ISIS 2018*, pp. 180–185, Jul. 2018, doi: 10.1109/SCIS-ISIS.2018.00038.

[8] J. Champ, A. Mora-Fallas, H. Goëau, E. Mata-Montero, P. Bonnet, and A. Joly, "Instance segmentation for the fine detection of crop and weed plants by precision agricultural robots," *Appl. Plant Sci.*, vol. 8, no. 7, p. e11373, Jul. 2020, doi: 10.1002/APS3.11373.

[9] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D

Classification and Segmentation.” pp. 652–660, 2017.

[10] R. Girshick, “Fast R-CNN.” pp. 1440–1448, 2015. Accessed: Jun. 05, 2023. [Online]. Available: https://github.com/rbgirshick/

[11] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, Accessed: Jun. 05, 2023. [Online]. Available: https://github.com/

[12] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 386–397, Mar. 2017, doi: 10.1109/TPAMI.2018.2844175.

[13] R. A. Potamias, A. Neofytou, K. M. Bintsi, and S. Zafeiriou, “GraphWalks: Efficient Shape Agnostic Geodesic Shortest Path Estimation.” pp. 2968–2977, 2022.

[14] A. Agathos and P. Azariadis, “Optimal Point-to-Point geodesic path generation on point clouds,” *Comput. Des.*, vol. 162, p. 103552, Sep. 2023, doi: 10.1016/J.CAD.2023.103552.

[15] H. Li *et al.*, “Automatic Branch–Leaf Segmentation and Leaf Phenotypic Parameter Estimation of Pear Trees Based on Three-Dimensional Point Clouds,” *Sensors 2023, Vol. 23, Page 4572*, vol. 23, no. 9, p. 4572, May 2023, doi: 10.3390/S23094572.

[16] Y. Wang, Y. Chen, X. Zhang, and W. Gong, “Research on Measurement Method of Leaf Length and Width Based on Point Cloud,” *Agric. 2021, Vol. 11, Page 63*, vol. 11, no. 1, p. 63, Jan. 2021, doi: 10.3390/AGRICULTURE11010063.

[17] “GitHub - matterport/Mask_RCNN: Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow.” https://github.com/matterport/Mask_RCNN (accessed May 31, 2023).