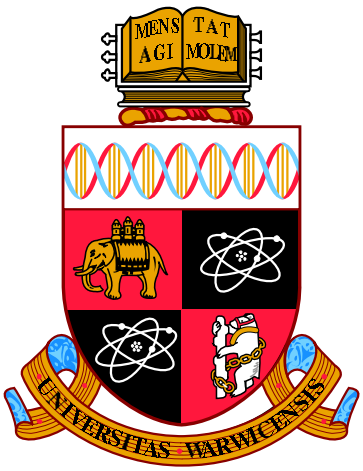


Computer Vision-Based Phenotyping for Classification of Plant Physiological States

by

Alex Foote

Technical Report

Submitted to the University of Warwick

in partial fulfilment of the requirements

for the degree of

Master of Engineering

School of Engineering

March 2020

Contents

Abstract

The continuing rise in average global temperature, and the incidence of extreme weather events caused by climate change, present significant challenges for crop production. To reduce the prevalence of undernutrition in a growing global population, crop production will need to increase despite these challenges. This will require the development and selection of plant breeds suitably able to produce high yields under stressful climactic conditions. An important step in this process is plant phenotyping, which quantifies the physical and physiological responses of a plant to imposed environmental conditions through the measurement of phenotypes such as plant height and convex hull volume.

This report presents the development of software written in MATLAB (referred as the developed software) that constructs three-dimensional (3D) plant models from multi-view depth images, and extracts relevant phenotypes from these models. Depth images input to the developed software are converted to point clouds and an initial rough segmentation is performed using manually defined parameters. The remaining point clouds consist of the plant, the pot it is in, and the surface where the pot is placed. Planes are fitted to the surface and the top of the pot, and all points in and below these planes are removed, leaving only the plant. The plant point clouds are then registered to the same frame and the resultant point cloud is processed to reduce noise and correctly orient it in space. Measurements of the holistic phenotypes of plant height, convex hull volume, Plant Aspect Ratio, and Plant Aerial Density are extracted from the point clouds. To allow non-expert users to efficiently utilise the system, a Graphical User Interface and detailed instructions for its use are provided.

Comparisons to measurements of the aforementioned phenotypes extracted from 3D models generated by the state-of-the-art depth image based phenotyping system show the developed software generates point clouds and extracts phenotypes that are accurate, and therefore suitable for use in plant phenotyping applications. The developed software has certain advantages

over the state-of-the-art, as it is able to more effectively capture small leaves, and more accurately orient the plant models in 3D space. Additionally, significant improvements to the level of automation are made, making the developed software more suitable for use in large-scale phenotyping applications. Further improvements to the software could include the integration of colour image data to allow for automated segmentation and a wider range of measurable phenotypes, or applying methods to segment the generated plant models into their respective organs, i.e., leaves, stem, etc.

List of Tables

List of Figures

Abbreviations

- 3D - Three-dimensional
- RGB - Red-Green-Blue
- PMVS - Patch-based Multi View Stereo
- 4D - Four-dimensional
- MRI - Magnetic Resonance Imaging
- FRET - Fluorescence Resonance Energy Transfer
- QTL - Quantitative Trait Locus
- VIS - Visible Light
- NIR - Near Infrared
- SIR - Shortwave Infrared
- CFI - Chlorophyll Fluorescence Imaging
- MSAC - M-estimator Sample Consensus
- ICP - Iterative Closest Point
- SFM - Structure-from-Motion
- SFS - Shape-from-Silhouette
- RMSE - Root Mean Square Error
- PAR - Plant Aspect Ratio
- MEC - Minimum Enclosing Circle

- PAD - Plant Aerial Density
- NRMSE - Normalised Root Mean Square Error
- ρ - Pearson's correlation
- Red Green Blue-Depth - RGB-D
- μCT - X-ray Microcomputed Tomography

Chapter 1

Introduction

1.1 Motivations

In 2017, 10.8% of the global population, corresponding to 810 million people, were undernourished [1], and undernutrition is estimated be responsible for close to half of all child deaths annually [2]. Decreasing the number of undernourished people and improving global food security is a major challenge, made even more difficult by a growing global population that is expected to reach approximately 10 billion by 2050 [3]. Additionally, the majority of this growth is expected to come from regions with the greatest food insecurity, with the population of Africa - where 19.9% of their population are undernourished, the highest proportion of any region [1] - forecast to double in the same timeframe [4]. This population growth, combined with the increased demand for biofuel, means that the demand for cereal crops, which constitute roughly half of the daily calories of individuals in developing countries [5], is predicted to increase by approximately 50% by 2050 [6].

Climate change also presents major challenges regarding food production, with changes in temperature, rainfall, and other factors, as well as an increase in the frequency of extreme weather events negatively affecting crop yield [7]. Climate change will have a greater impact on areas of low latitude, that correspond to developing regions with poor food security, such as sub-Saharan Africa [8].

To meet the demands of increased cereal production and reduce the prevalence of undernutrition, plant breeds that produce greater yields are required. These plants will also need to be able to survive in the increasingly stressful conditions that climate change will create. To identify and select breeds with these desirable characteristics, the ability to measure the response of the physical and physiological traits of a plant to a particular environment is crucial. Historically, these traits have been measured by hand, a slow and labour intensive process that often becomes a bottleneck in plant breeding [9]. Additionally, some measurements could only be taken at a single timepoint, as they required the destruction of the plant. These problems with manual plant phenotyping have motivated research into the development of automated plant phenotyping systems, which can extract useful measurements at high-throughputs, non-destructively, so

the phenotype can be measured across several timepoints.

1.2 Aim of Project

This project aims to develop highly automated plant phenotyping software. The software will take as input multiple depth images from several different viewpoints of plant, and construct a three-dimensional (3D) model of the plant. It will then extract desired holistic phenotypes from the plant model, and produce graphs of the change in the traits of a plant over the course of several timepoints. To fulfil this aim, the following objectives have been proposed:

- Develop software that constructs 3D models of plants from a set of depth images
- Choose a set of holistic phenotypes to extract from the 3D models, based on a review of the plant phenotyping literature
- Develop software that measures the desired phenotypes from the 3D models
- Evaluate the accuracy of the software and determine its utility for plant phenotyping applications
- Produce recommendations for future research

1.3 Structure of Report

The report is structured as follows. Chapter 2 presents a literature review summarising the relevant research in the field of plant phenotyping. Chapter 3 presents the proposed phenotyping method. Chapter 4 presents the graphical user interface and the software documentation of the phenotyping method. The results of experiments evaluating the accuracy of the phenotyping software are presented and analysed in Chapter 5. Finally, Chapter 6 presents the conclusions of the project, and recommendations for future research.

Chapter 2

Literature review and technical background

2.1 Plant phenotyping

2.1.1 Background

An organism's phenotype is the physical expression of its genes, and it is significantly affected by the organism's environment. Important environmental factors that affect a plant's phenotype include temperature, carbon dioxide concentration of the air, and availability of water [7]. To breed plants with desirable characteristics, e.g., high yield at high temperatures, a number of plants with different genotypes are grown in a controlled greenhouse environment, and various aspects of the plants' phenotypes, e.g., the height, convex hull volume, and number of leaves, are measured over time. This process is known as plant phenotyping, and can be combined with plant genotyping to identify quantitative trait loci (QTLs). The information gained from phenotyping a small number of genotypes can in turn be used to select the genotypes with desirable combinations of alleles at important QTLs from a much larger number of genotypes, *in silico* [10]. This reduces the number of plants that need to be grown and phenotyped to identify desirable genotypes, significantly increasing efficiency.

The importance of plant phenotyping has led to a significant amount of research conducted into the development of high-throughput automated plant phenotyping systems, that can classify phenotypes quickly and accurately with little human intervention. A majority of this research has focused on extracting measurements using 2D imaging methods such as infrared imaging, visible light imaging, fluorescence imaging and others [11]. Recently, 3D phenotyping has become more popular due to a number of advantages of the approach, including the ability to take a wider variety of measurements with greater accuracy [12]. 3D models can also be saved and re-analysed in the future if new measurements are required, whereas new and novel measurements may not be able to be extracted from images not intended for capturing the required information.

2.1.2 Phenotypes

A wide variety of phenotypes have been presented in the literature, aiming to quantify different aspects of a plant's overall phenotype. Plant phenotypes can be grouped into two main categories: holistic, or non-complex phenotypes, which can be extracted from the plant as a whole; and component, or complex phenotypes, that are extracted from a representation of the plant segmented into its organs (the stem, the leaves, etc.) [12]. Holistic and component phenotypes can be divided into the sub-categories of primary and derived phenotypes. Primary phenotypes quantify basic traits such as plant height and width, whereas derived phenotypes use primary traits to generate more complex phenotypic measures, such as aspect ratio [13].

2.2 3D plant models

2.2.1 Data representation

3D plant data is generally represented in one of three ways - using a point cloud, a polygonal mesh, or a voxel representation.

Point clouds

Point clouds consist of a set of points defined by their 3D coordinates. They are often the initially obtained 3D data representation in phenotyping software, and measurements may be extracted directly from them or they may first be processed further to create a mesh.

Frequently, several point clouds from different views will be obtained by imaging the plant, and these point clouds are then aligned through the process of registration. Two point clouds can be aligned together using various registration algorithms, that compute either a rigid or non-rigid transformation that is applied to the 'moving', or source point cloud to align it to the 'fixed', or reference point cloud. An important registration algorithm is the Iterative Closest Point (ICP) algorithm [14, 15], which estimates rigid transformations. The ICP algorithm matches each point in the moving cloud to its estimated corresponding point in the fixed cloud, and then estimates the rigid transformation which minimises the sum of the squared errors for the point pairs, where the error is the Euclidean distance between the pair of points. This transformation is applied, and the procedure is repeated until either the difference between consecutive transformations falls below a specified threshold, or a specified number of iterations is reached.

Polygonal meshes

In plant phenotyping, polygonal meshes are generally created by performing surface reconstruction on a point cloud [16]. They represent the surface of an object using a set of polygons. Point clouds may be meshed to facilitate the extraction of certain measurements, such as surface area.

Voxel representation

3D plant models may also be represented using voxels, which are the 3D analog of pixels. This representation can be produced by the Space Carving algorithm [17], and may also be converted to a mesh for phenotype extraction, for example by the marching cubes algorithm [18].

2.2.2 Construction of 3D plant models

Plants have a number of characteristics that make constructing their 3D models challenging. Plants are reflective, and many species are relatively uniform in colour and texture, making the extraction or identifying features difficult. Self-occlusions, e.g., overlapping leaves, are also very common in plants, which can lead to gaps in the 3D model. Furthermore, thin leaves may fail to be captured by low resolution imaging methods, such as structured light imaging [19]. There are two categories of image-based construction methods, active methods and passive methods. Active methods illuminate the plant directly, whereas passive methods do not.

Active methods

Some active methods generate depth images, where each pixel of the image contains the distance between the sensor and an object point. Different methods measure this distance in different ways. Structured light techniques shine a known pattern of light onto an object and measure the deformation in the pattern to extract depth information [20]. Time of flight methods, such as Lidar, project light, often infrared or near-infrared, onto an object and measure the time taken for the light to be reflected back to the sensor, which can be used to estimate depth.

Passive methods

Passive methods generate 3D models from RGB images captured with a calibrated camera. Camera calibration consists of estimating the internal and external parameters of a camera to enable the mapping of 2D image points to 3D world points. This allows distances to be extracted from calibrated images. Camera calibration uses images of a checkerboard pattern with known dimensions to extract the parameter matrix [21].

Common passive methods include Structure-from-Motion (SFM) [22, 23], Shape-from-Silhouette (SFS) [24], and Space Carving [17]. These three methods reconstruct a 3D model from RGB images captured from multiple viewpoints around an object. To accurately construct a 3D model using these methods requires many images from different, useful viewpoints, which is challenging to achieve with the conventional method of rotating a plant on a turntable and capturing images from a static camera position. Gibbs *et al.* presented a method to automatically identify and acquire images from viewpoints where more data were needed using a robot arm [25], constructing accurate models.

2.2.3 Segmentation

3D plant segmentation attempts to assign each element of a plant model, e.g., a point, polygon, or voxel, a label denoting which class of organs it belongs to, e.g., the leaves, shoot, inflorescence etc. This is a challenging task that is less researched than 3D plant model construction or 2D plant image segmentation, with some studies performing manual 3D segmentation to obtain organ level measurements [12]. A promising approach was presented by Elnashef *et al.*, who developed a segmentation algorithm that computes the first- and second-order tensors of a plant point cloud to segment it into its requisite organs [26]. They demonstrated the software could accurately segment point clouds of plant species with varying morphologies, an important advantage over systems that rely on assumptions that limit the application to plants of a particular morphology.

2.3 Paper review reports

As part of the literature review conducted for this project for acquiring relevant background knowledge, ten detailed review reports of ten journal papers [10, 13, 21, 24, 25, 27–31] where each summarises a particular paper were written. The three most relevant review reports are presented in Section 2.4 to Section 2.6 in full.

2.4 Plant Phenotyping: An Active Vision Cell for Three-Dimensional Plant Shoot Reconstruction

A report on a paper discussing a novel method for automatically capturing images of a plant from useful viewpoints, by Gibbs *et al.* [25].

2.4.1 Introduction

Plant phenotyping methods that use 3D models have advantages over those that use 2D models, allowing for more traits to be extracted with potentially greater accuracy. However, generating 3D models of plants has a number of challenges, in particular occlusions caused by overlapping leaves. This paper investigates an active computer vision approach to overcome this difficulty, which involves identifying which camera views provide the most useful information about the 3D structure of the plant.

2.4.2 Method

Overview

The proposed method captures images from arbitrary camera positions to construct an initial model of the plant, then uses this model to improve the image set by capturing more images from appropriate camera positions and removing redundant images.

Equipment and calibration

A camera is attached to a robot arm that has six degrees of freedom, three in translation and three in rotation. It captures images of a plant placed on a turntable.

To allow for 3D information to be extracted from the captured images, the system must be calibrated. Firstly, the camera is calibrated so the 2D images can be mapped to the 3D world. Next, the robot arm is calibrated so the position of its end effector which attaches to the camera is known relative to the base of the arm, allowing for the end effector to be positioned in a specified location. Then, both the position of the robot base relative to the turntable, and the position of the end effector relative to the camera are calculated. Finally, the centre of rotation of the turntable is determined, to account for the rotating coordinate frame.

Construction of inital model

To construct the basic model, a number of images are captured with a Red-Green-Blue (RGB) filter to remove the white background of the scene, leaving only the plant. Firstly, an image is taken with the camera levels with the turntable and in line with its centre of rotation. This image is used only for centring; adjusting the camera position such that the entire plant is in view. This is achieved by evaluating if the plant is fully contained in the image and if it occupies a large enough portion of the image. Next, the plant is rotated and an image is captured for every rotation of 36°. A model is generated from these images using space carving[1]. This model is used to estimate the plant height, and the camera is moved to this height and recentred. The plant is again rotated and images captured for every 36°, but this time the intial position is offset from that of the first image set by 12°. These images are used to improve the model, and the height is extracted from the new model. Images are then taken at twice the height of the plant, every 36°and offset by another 12°. The model is again improved, and is now ready to be used in the next stage of the refinement.

Model refinement

The refinement starts by removing redundant images. An image is classified as redundant if the voxels (i.e., 3D pixels) in the model are still seen by at least three camera positions when the image is removed. Ray tracing is used to evaluate whether a voxel is seen by a particular camera position.

Next, the model is used to determine where additional images are needed. The voxels are clustered into small groups to reduce the number of camera positions to be evaluated. Each cluster is then scored based on the number of camera positions in which it is viewed, and the angle between those camera positions. A score below the chosen threshold denotes the cluster as undersampled, i.e., in need of more images.

[1]Space carving extracts the plant's silhouettes from the images, projects them into 3D space and determines the intersection of the resulting volumes to generate a 3D model.

Additional camera positions are then computed for each undersampled cluster. For each cluster, a view sphere is constructed around the cluster, about the surface of which camera positions are evaluated. Firstly, the point on the sphere normal to the cluster is evaluated, and then viewpoints further away from this point are tested if closer points fail. For a point to be a valid camera position, the robot arm must be able to reach it, and from it the cluster must be visible.

Once positions have been found for each undersampled cluster, an image is captured from each position. The final refined model is generated using Patch-based Multi View Stereo (PMVS), which detects and matches features of the images to construct the 3D model.

2.4.3 Comparison to other approaches

Other approaches for constructing 3D plant models have significant limitations. For example, rule based approaches based on prior assumptions about plant structure are not generalisable to many species due to the large variations in structure between species.

Non-active vision approaches often use static or arbitrary camera positions. Static methods use fixed camera positions for all plants, and arbitrary methods use random camera positions. Six experiments were conducted, each with a different plant species, comparing the models produced with PMVS using images from each method. The experimental results show that the active vision approach outperformed the static and arbitrary methods across all species in both accuracy and number of images used. Higher accuracy, as measured by comparison to a ground truth model, was consistently higher, and fewer images were used, which decreases processing time.

The ground truth models were generated using microcomputed x-ray topography, which produces highly accurate models. This method, however, is not appropriate for large scale plant phenotyping due to the very expensive equipment and long processing times, taking hours to image a plant compared to the minutes taken by the proposed method.

2.4.4 Future work

Plant phenotyping is currently a bottleneck in the identification of more productive plant species, which has important implications for food production. Accurate 3D plant models, as generated by the proposed method, could help reduce this bottleneck by facilitating the measurement of new traits that affect crop yield.

The phenotyping of plants in field environments is underdeveloped compared to phenotyping in controlled laboratory environments, and so more research investigating techniques that work in the field is needed.

2.5 Plant Phenomics, From Sensors to Knowledge

A report on a survey paper discussing the field of plant phenomics, focussing on current and future phenotyping methodologies, by Tardieu *et al.* [10].

2.5.1 Introduction

Plant phenomics involves the capture and analysis of information about various aspects of plant development, and the use of this information to model plant physiological behaviour. This paper considers the challenge of taking captured plant data from sensors and extracting useful information, and argues that current techniques will need to be integrated together at scale to meet these challenges.

2.5.2 Phenotyping platforms

Plants can exhibit significant variations in their structure and physiology in response to changing environmental conditions, making the ability to phenotype transient states important. Additionally, it is necessary to study plants over a range of scales, both spatially and temporally, in order to understand how their environment affects their phenomes from the field level to the cellular level in the long and short term. To address these needs, there are several primary phenotyping platforms that each tackle a particular aspect.

High-precision

High-precision platforms investigate the mechanisms by which plants respond to changes in their environment. They operate at the scale of single plants, monitoring the growth, structure, and composition of plant organs such as the leaves and roots. Techniques used in high-precision platforms include 4-dimensional (4D) visible light imaging[2] to monitor leaf growth, magnetic resonance imaging (MRI) to monitor the transport of elements through plants, and fluorescence resonance energy transfer (FRET) sensors to monitor hormonal changes within cells. High-precision platforms are useful for better understanding of plant physiology and stress response, but are currently unable to analyse large numbers of plants, as is needed to evaluate the effects of a range of environment on a range of genotypes.

Whole-plant, multi-environment

Whole-plant, multi-environment platforms allow for the analysis of the variation in response to particular environmental conditions for a range of plant genotypes. They generally operate in controlled environments, such as greenhouses, allowing for the automated monitoring of both the plants and the environment, and perform 4D imaging. Traits such as growth rate can be measured directly from this imaging, and complex traits such as photosynthesic ability can be

[2] 4D imaging refers to imaging plants in 3D across time.

obtained using models. Whole-plant, multi-environment platforms cannot be used to investigate yield, as the greenhouse environment used in these platforms is not suitable for useful yield measurements.

Field multi-environment networks

Field multi-environment networks are large-scale field studies that investigate yield stability (the variation in the yield of a plant in differing environmental conditions). The networks consist of studies conducted across large geographical areas with diverse environmental conditions. Field studies of a genotype's yield are conducted in the different conditions, that allow for the identification of parts of the genome that are related to yield in particular conditions. These parts of the genome are called quantitative trait loci (QTLs).

The effect that a particular allele for a given QTL has on yield is often conditional, i.e., an allele may have a positive effect on yield in one condition but a negative effect in a different condition. This necessitates the use of a large variety of environmental conditions to accurately phenotype a particular genotype of a plant. Phenotyping many genotypes in this manner allows for the identification of the best combination of alleles for a given condition, and this combination can then be used to select tens of thousands of plants that will produce a high yield without having to phenotype them.

Cross-scale meta-analyses

Each of the above platforms has particular weaknesses, and none provide a comprehensive picture of plant response over the full range of scales. A combination of approaches allows for a deeper understanding of the phenotypic response of a genotype to different environments. This combination of approaches is referred as 'cross-scale meta-analyses', and a broader approach is becoming increasingly necessary in plant phenomics.

A number of challenges arise when trying to implement cross-scale meta-analyses, due to the new requirement that data gathered with each platform must be applicable across the temporal and other spatial scales. To allow the comparison of plant response between experiments, consistent environmental data must be captured, and phenotype measurements must be taken under reproducible conditions. Data from disparate sources, such as environmental sensor data and images, must be integrated into mixed models. Models must be developed that use the mechanisms proposed from high-precision platforms to predict plant response at other scales, e.g., yield in a field study. These challenges have begun to be addressed in the first few cross-scale meta-analyses, but the overarching problem of collecting large amount of data in a variety of experiments in such a way that knowledge can be extracted from all the data in combination remains.

2.5.3 Future work

The primary area that will facilitate development in plant phenomics is the management and storing of the captured data from experiments. To allow experimental data to be used across studies and re-analysed at any point in the future, systems that facilitate good practice in data capture must be implemented. Standardised methods for capturing and storing phenotypic data and its associated metadata are being developed, and will need to be further improved. As the scale and complexity of experiments increases, standardisation through defined ontologies for data capture will become ever more important.

Additionally, the authors suggest that increasing the use of 'big data' approaches to analysing data is desirable. For example, moving from brittle hand-crafted image processing tools that primarily work in narrow scenarios to more flexible deep-learning methods could allow for the additional data generated from cross-scale meta-analyses to be utilised more effectively.

Farmers are increasingly using sensors to monitor the environmental conditions under which their crops are growing, and this data, combined with their data on yield, provides potentially useful information about plant phenomes at a scale greater even than field multi-environment networks. This presents a promising possibility of combining this real-world data with data from cross-scale meta-analyses to further understand the interactions between plant genetics and their environment, and the associated effect on yield.

2.6 Sensing Technologies for Precision Phenotyping in Vegetable Crops: Current Status and Future Challenges

A report on a survey paper discussing the various sensing technologies used in plant phenotyping, by Tripodi *et al.* [27].

2.6.1 Introduction

Phenotyping remains a bottleneck in the process of analysing plant response to stress. The selection of genotypes with beneficial characteristics in particular environments is of significant importance in improving worldwide food security, and phenotyping is a major part of this process. Plant phenotyping is becoming increasingly automated through the use of sensors to gather plant data and the development of software to analyse this data. The continuing improvement in these areas will help to remove this bottleneck.

2.6.2 Phenotyping approaches

It is beneficial to phenotype plants over a range of scales, from the organ scale to the field scale. Different scales allow the analysis of different aspects of plant stress response, and require different approaches to phenotyping. High resolution approaches analyse plants at the level of the whole plant, its organs and even its cells. Phenotyping at the level of cells typically requires

invasive approaches, and so is not suited to high throughput, as is required for large scale field analysis. Analysis of plant structure and growth can be accomplished with high throughput. However, to link physiological changes in the plants to a specific cause generally requires controlled conditions through, for example, the use of a greenhouse. Controlled environments make the measurement of yield irrelevant, due to the disparity between greenhouse conditions and those in the real world. In contrast, studies conducted in the field are useful for examining the effect of environmental conditions on yield, but the uncontrolled environment makes associating a response to an environmental cause challenging.

2.6.3 Sensing technologies

A wide variety of sensors are used in plant phenotyping. Many measure some aspect of the electromagnetic radiation that plants absorb, reflect, transmit, and emit.

Visible light (VIS) plant imaging is used for the measurement of a variety of traits, such as plant biomass, stem angle, and growth rate. This is primarily derived from the analysis of plant structure and its changes over time. Some of these traits can be captured from 2D images of plants, whereas others may require the construction of 3D plant models from a number of 2D images due to the complexity of plant structure. VIS imaging can also be used to identify plant stress, both biotic and abiotic.

Near infrared (NIR) and shortwave infrared (SIR) imaging are useful for measuring the reflectance of radiation with a wavelength in the range of 900 nm to 1700 nm. NIR imaging allows for the measurement of various indices related to the use of water in plants. SIR imaging is useful for the detection of biotic and abiotic stresses, and when performed in controlled environments can be used to link imposed environmental conditions to particular stress responses. NIR and SIR imaging can be performed using hyperspectral cameras or spectroradiometers, with the main difference being that spectroradiometers capture optical information but no spatial information, i.e., they do not capture images. Hyperspectral cameras do capture images, which allows for a wider variety of indices to be measured, but are significantly more expensive and produce large volumes of data, making data handling more difficult. The choice between the two pieces of equipment may therefore vary depending on the resources available to a researcher and the indices they want to analyse.

Fluorescence spectroscopy is used to measure leaf chlorophyll content and indices related to photosynthesis, such as photosynthetic efficiency. One method for performing fluorescence spectroscopy is chlorophyll fluorescence imaging (CFI), which induces fluorescence by projecting light of differing wavelengths onto a plant, and uses cameras to capture the fluorescent response.

Thermal imaging captures longwave infrared radiation, with wavelengths between 3 μm and 14 μm. Radiation given off by plants in the form of heat is in this wavelength spectrum, which allows for the analysis of transpiration regulation by the stomata, as transpiration and leaf temperature are correlated. Since there are several environmental factors that also influence leaf temperature, attributing observed temperature changes to stomatal variation can be chal-

lenging. Measuring the environmental conditions in the plant's surroundings can help combat this challenge.

MRI allows for the direct capture of the 3D structure of plants. Additionally, it can be used to capture 3D data for plant roots whilst they are in soil, a task not possible with VIS imaging. It is therefore useful for analysing variations in root structure caused by different environmental conditions.

2.6.4 Future work

As automation in phenotyping continues to improve, systems must be implemented to ensure that captured data is of high quality throughout a phenotyping pipeline. Human supervision, for example in the form of spot checks, could be one method by which data quality is ensured.

Appropriate analysis of captured data is another major challenge. Attributing observed measurements to specific causes requires careful control and/or measurement of environmental conditions. The increasing volume of data generated will also require new techniques to manage and analyse, and this must be investigated in parallel to improving phenotyping technologies to ensure that data management does not become a bottleneck.

Robotics is likely to play an increasingly important role in plant phenotyping as systems become increasingly automated. The use of robots to automate the collection of data using sensors is likely to become more common, particularly in high throughput phenotyping studies in controlled greenhouse environments.

Chapter 3

Method

3.1 Initial investigation

During the initial development phase, the construction of 3D plant models from multi-view RGB images, captured with a calibrated camera, was investigated. A dataset containing RGB images of plants taken from 10 viewpoints [32] was tested with VisualSFM [22, 23] software, which demonstrated that dataset was unsuitable for 3D reconstructions using the intended methods, due to the small number of viewpoints. For this reason, a dataset of depth images [33] was investigated, and initial reconstruction attempts suggested that these would be suitable for constructing accurate 3D plant models.

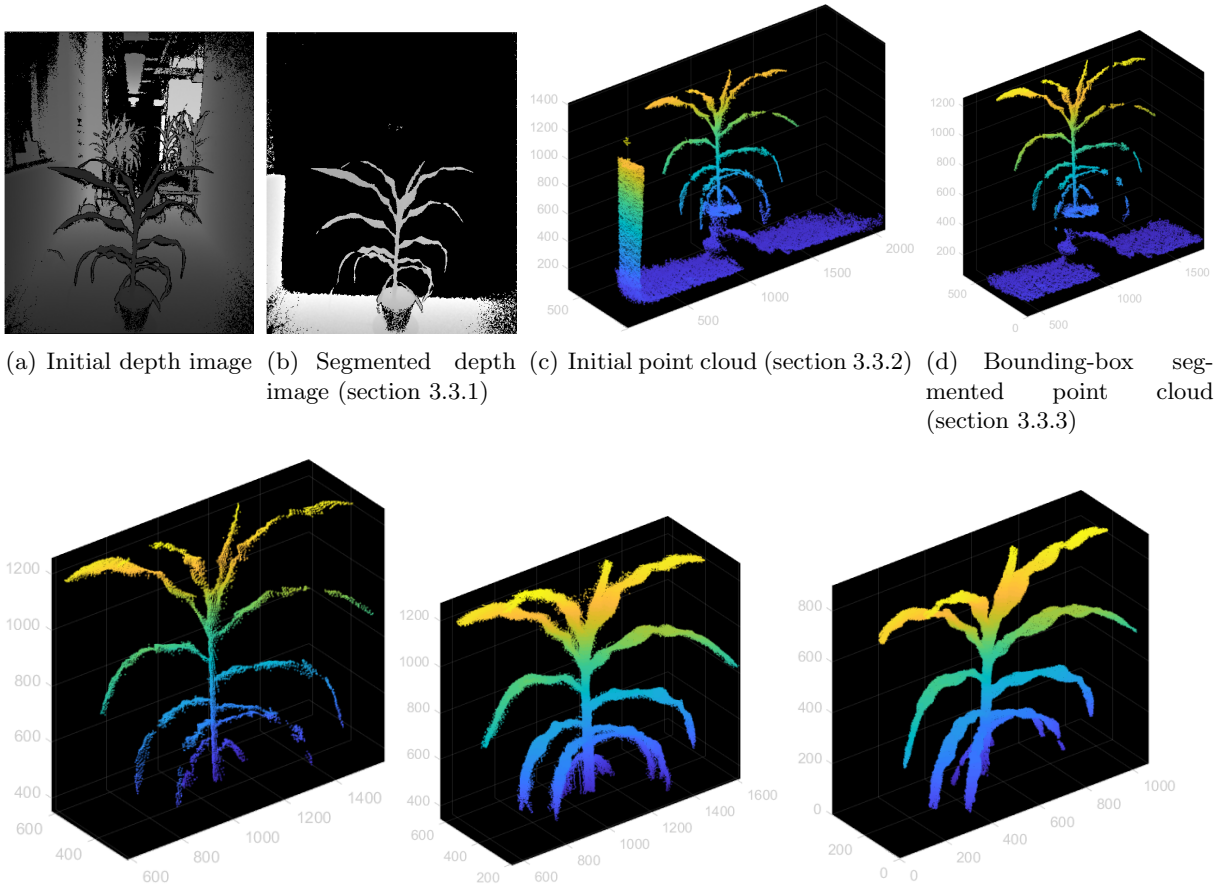
3.2 Overview of method

The software presented in this report was created in MATLAB [34], and using its Computer Vision Toolbox [35]. The input to the software consists of a set of depth images taken from several camera views positioned 360°around a plant, and a parameter file, whose contents and creation are presented in the next chapter. The software consists of two main stages - point cloud construction, as illustrated in Figure 3.1, and measurement extraction.

3.3 Point cloud construction

3.3.1 Depth image foreground and background removal

It is essential to remove the foreground and background scene (i.e., the parts of the image that do not contain the plant) to facilitate the extraction of the plant only. To remove the foreground and background of a depth image, all pixels that have a depth value smaller or larger than two user defined thresholds are set to 0. This step speeds up the creation of a point cloud from a depth image as it reduces the number of points to be created. Additionally, during the development of the software it was noted to be easier to specify a depth threshold for a depth



(a) Initial depth image (b) Segmented depth image (section 3.3.1) (c) Initial point cloud (section 3.3.2) (d) Bounding-box segmented point cloud (section 3.3.3)

(e) Fully segmented point cloud (section 3.3.3) (f) Registered point cloud (section 3.3.4) (g) Final processed point cloud (section 3.3.5)

Figure 3.1: The processing stages for constructing a point cloud from multiple depth images

image than for a point cloud. Thus, this step helps to make the software user-friendly and decreases the time needed to identify appropriate parameters needed for analysis.

3.3.2 Depth image to point cloud conversion

To convert a depth image to a point cloud, the image coordinates and depth values $[x_{image}, y_{image}, z_{depth}]$ need to be mapped to the correct 3D world coordinates $[x_{world}, y_{world}, z_{world}]$, using the following equations:

$$x_{world} = (x_{image} - c_x)/f_x \tag{3.1}$$

$$y_{world} = z_{depth} \tag{3.2}$$

$$z_{world} = (y_{image} - c_y)/f_y \tag{3.3}$$

where $[c_x, c_y]$ are the camera centre coordinates, and $[f_x, f_y]$ are the camera focal lengths in the x and y directions, respectively. The camera centre and focal lengths can be determined through camera calibration, or the approximate default values provided can be used. The

$[x_{world}, y_{world}, z_{world}]$ coordinates are combined with their corresponding depth values, and a matrix of the world coordinates is used to create a MATLAB pointCloud object.

3.3.3 Point cloud segmentation

The newly created point cloud is then segmented into plant points and non-plant points, where the non-plant points are then removed. This is achieved as follows.

Firstly, any points whose (x, y) coordinates are within the thresholds defined in the bounding box parameter are removed. This results in a rough segmentation which removes any large objects far away from the plant, leaving only the plant and its pot. Note that it is helpful if the surface where the pot is sat, e.g., the floor or a table, is retained.

If the surface is present then a plane is fitted to the surface using the MATLAB implementation of the M-estimator Sample Consensus (MSAC) algorithm [36]. All points within and below the plane are then removed. To determine whether points are below the plane, the distance d from the plane is calculated using

$$d = (\mathbf{q} - \mathbf{p}) * \mathbf{n} \tag{3.4}$$

where $\mathbf{q}$ is a point in the point cloud, $\mathbf{p}$ is a point on the plane and $\mathbf{n}$ is the normal to the plane. All points on the same side of the plane will have the same sign. The points below the plane are defined as the set of points with the opposite sign to a point known to be on the same side of the plane as the plant. The larger z-coordinate of the bounding box is used as the known plant-side point, as this represents a point above which there are no plant points - therefore it must be above the plant and therefore above the surface.

The same procedure is followed to remove the plane of the top of the pot and all the points below it. If a surface was present, the normal to the surface plane is used as an initial guess for the normal to the top of the pot. This is based on the assumption that the top of the pot and the surface are parallel. If a surface was not present, the normal to the z plane is used. This sometimes results in parts of the point cloud other than the top of the pot being identified as a plane and thus removed, resulting in a malformed point cloud. For an improved segmentation the surface upon which the pot is sitting should not be removed in the initial bounding box segmentation.

3.3.4 Point cloud registration

A common method for registering a set of point clouds captured from multiple views of an object is pairwise registration [37]. This method registers the second point cloud in the set to the first, and records the transform used to do so. Next, the third point cloud is registered to the second, and the first transform is then also applied to the transformed third point cloud, to transform it to the reference frame of the first point cloud. The second transform is combined with the first, and the process continues, registering pairs of point clouds and accumulating the transforms.

The above algorithm is effective and accurate when at each stage the pair registration is accurate. However, when a low quality pair registration occurs, the bad transform can affect the accumulated transform, causing the registration of the final point cloud to be erroneous. Errors occurring upstream in the software, e.g., when segmenting the point cloud, would sometimes cause low quality point clouds to be passed to the registration algorithm, resulting in poor registrations. To fix this, several modifications to the above standard algorithm are made as follows.

Firstly, a threshold for the Root Mean Square Error (RMSE) of the registration between two point clouds is added, below which a newly transformed point cloud would not be added to the overall point cloud. This threshold is set at 10mm, a value that was empirically determined. Incorporating an RMSE threshold as a constraint means that the accumulated transform no longer maps a transformed point cloud to the reference frame of the first point cloud when a poor registration has occurred. To remedy this, a secondary registration between the transformed point cloud and the overall point cloud is applied. This corrects for missing transforms, and has the additional benefit of improving the registration of many already good pair registrations, as it allows for extra data from other previously registered point clouds to be used to register a new point cloud.

In addition to this measure, the previous good transform is recorded as point clouds are registered, and used as a substitute for a bad transform in the accumulated transform. As point clouds are generally captured at approximately even intervals around an object, the previous good transform serves as a relatively close approximation of the true transform. This helps the ICP algorithm [14, 15] being used to register two point clouds to converge, as the new point cloud and the overall point cloud will be closer to one another than if no transform was accumulated.

3.3.5 Point cloud processing

Once an overall registered point cloud has been obtained from a set of point clouds, it is processed to reduce noise. Noise is reduced using a k-nearest neighbour denoising algorithm [38] that removes outlier points based on their distance to their k-nearest points. The fifty nearest points were used, and the threshold for concluding a point was an outlier was set to 0.5 standard deviations from the mean average distance between all points and their fifty nearest neighbours. These parameter values were determined empirically.

To allow for accurate measurements to be extracted from the point cloud, the orientation of the point cloud has to be matched to the orientation of the real world plant when the point clouds were captured. When capturing datasets, the camera will likely not be level with the top of the plant pot, resulting in the point cloud appearing rotated relative to the base coordinate system. To account for this rotation, the top plane of the pot extracted in the point cloud segmentation stage is registered, using the ICP algorithm, to a generated plane of points which is parallel to the x and y axes, and this transform is applied to the plant point cloud. As the pot plane is always approximately horizontally symmetric, the transform could be incorrect by

180° about the horizontal plane. To check for this error, the relative position of the highest and lowest points along the vertical z-axis is monitored. If the highest point is lower than the lowest point after the transformation, the plant point cloud has been flipped by 180°, and so a transformation matrix corresponding to a negative 180° rotation about the horizontal plane is applied to correct for this.

Finally, the principal axis of the plant point cloud is identified, and the point cloud is rotated such that its principal axis is parallel with the x-axis of the base coordinate system. The principal axis is defined as the horizontal axis along which the plant has the greatest width. To determine this, the point cloud is iteratively rotated about the z-axis by 5° up to a combined rotation of 180°, and the width of the plant along the x-axis, defined as the difference between the x coordinates of the point with the largest x coordinate and the point with the smallest x coordinate, is measured. The angle θ at which the maximum width occurs is recorded and the point cloud is rotated about the z-axis by $-\theta$ to align the principal axis with the x-axis.

3.4 Measurement extraction

As the plant point clouds are not segmented into their component organs, all the measurements extracted from the point clouds are holistic, i.e., taken over the whole plant. Thus, a combination of primary, e.g., plant height, and derived, e.g., plant aspect ratio, phenotypes are extracted for plant phenotyping.

3.4.1 Height

Height is defined as the vertical distance from the base of the plant to the highest plant point. To extract the height, the difference between the z coordinates of the point with the largest z coordinate and the point with the smallest z coordinate is determined. Plant height is a trait useful for tracking plant growth over time.

3.4.2 Convex hull volume

The convex hull volume of a plant point cloud is used to quantify plant size. This metric is measured using the MATLAB function convhull [39], which, given a set of 3D points as input, computes the 3D convex hull and returns the volume.

3.4.3 Plant aspect ratio

Plant aspect ratio (PAR), first introduced in [40], is defined as the ratio between the height of the plant and the diameter of the minimum enclosing circle (MEC) of the projection of the point cloud along the vertical z-axis, i.e.,

$$PAR = Height/Diameter_{MEC}. \tag{3.5}$$

To project a point cloud along one of its axis, the corresponding axis coordinate of each point is removed. For example, to project the point $[1, 2, 3]$ along the x-axis would result in the point $[2, 3]$, and projecting the same point along the y-axis would yield the point $[1, 3]$. The MEC is computed using an open source tool written in MATLAB [41]. PAR quantifies the leaf extent of a plant relative to its height.

3.4.4 Plant aerial density

Plant aerial density (PAD) was introduced in [13] to quantify foliage density. In the original paper, this trait was extracted from 2D images, and so had to be adapted in this project to be extracted from 3D point clouds. PAD is extracted along each of the x, y, and z-axes, using

$$PAD = Projection_{PA}/Projection_{CHA} \tag{3.6}$$

where $Projection_{PA}$ is the total plant area of the projection of the point cloud along the appropriate axis, and $Projection_{CHA}$ is the convex hull area of that projection, determined using the MATLAB function, convhull, over the projected set of 2D points.

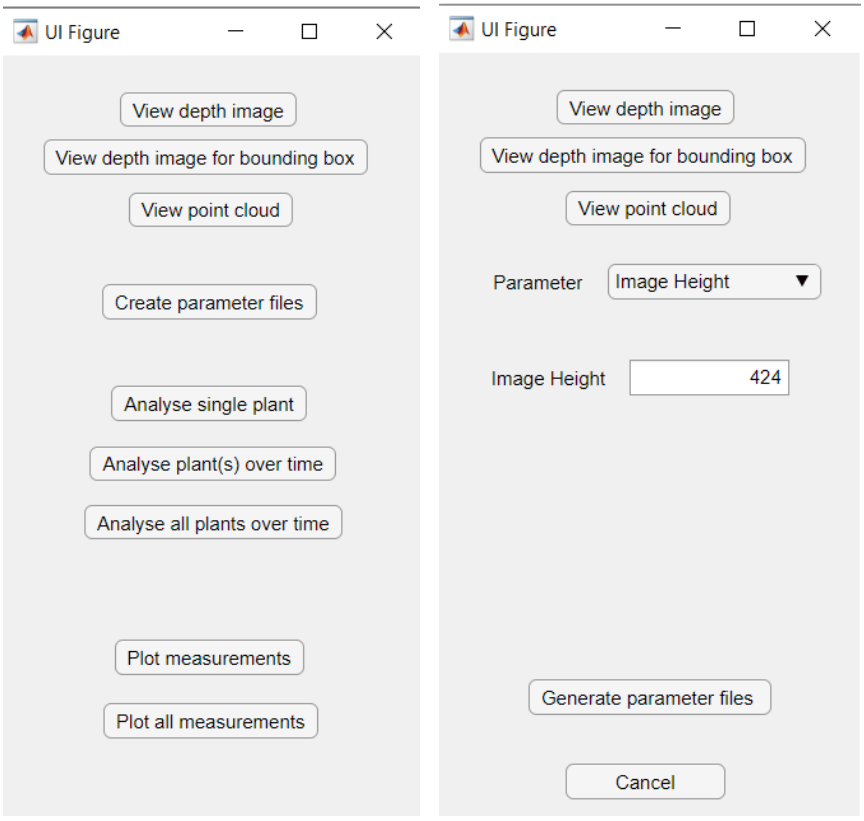
To determine $Projection_{PA}$ from the projected set of points, the coordinates of all the points are first rounded to the nearest integer, and any duplicate coordinates created are removed. A zero matrix with dimensions equal to the ranges of the two sets of coordinates is created. For each point in the set, the entry in the matrix indexed by the point's coordinates, as well as all adjacent points (using 8-bit connectivity), is set to 1. $Projection_{PA}$ is equal to the number of matrix entries with value 1.

Chapter 4

GUI and Software Documentation

4.1 Graphical User Interface

To improve the usability of the software by non-experts for phenotyping applications, a Graphical User Interface (GUI), as shown in Figure 4.1(a), was created. Instructions for using the GUI to construct and measure point clouds are presented in Section 4.2.



(a) Initial GUI (b) GUI for creating parameter files

Figure 4.1: Examples of GUI menus.

4.2 Software Documentation

4.2.1 Data preparation

The first step in using this software is to store the data in the appropriate format and in the appropriate folder structure. Datasets must be organised in the folder structure as shown in Figure 4.2. Each dataset is organised first by its imaging date and then by plant identification (ID). All depth images must contain 'Depth Image' within their filename so that the software can identify them.

```
\---Dataset
    +---Time Point 1
    |   +---Plant 1
    |   |       DepthImage 1.png
    |   |       DepthImage 2.png
    |   |       DepthImage 3.png
    |   |       DepthImage 4.png
    |   |       parameters.mat
    |   |
    |   +---Plant 2
    |   \---Plant 3
    \---Time Point 2
```

Figure 4.2: The required folder structure for a dataset.

4.2.2 Creating parameter files

The next step is to use the GUI to generate parameter files for each set of depth images. The parameters consist of:

- Image Height - the number of rows in a depth image
- Image width - the number of columns in a depth image
- Number of Views - the number of depth images taken
- Vertical axis - the axis along which the plant is upright: can be x or y
- Remove plane - controls whether the software tries to remove the plane that the plant pot is placed on: can be true or false
- Remove pot - controls whether the software tries to remove the pot that the plant is grown in: can be true or false
- Depth thresholds - foreground and background cutoff distances
- Bounding box - Left, right, top, and bottom coordinates that define a box which only contains the plant and potentially the pot, and the surface where it is placed

- Save point cloud - controls whether the constructed point cloud is saved in the folder containing the depth images: can be true or false
- Save measurements - controls whether the extracted measurements are saved in the folder containing the depth images: can be true or false

Generally, the same parameter values will suit many sets of depth images, and these files can be generated at the same time. Suitable values for the depth thresholds and the bounding box may change if the camera position relative to the plant is changed significantly, which may occur between image capture sessions.

To start creating parameter files, click the 'Create Parameter Files' button on the initial GUI as shown in Figure 4.1(a). This brings up a new interface window for creating parameters as shown in Figure 4.1(b). Go through the drop down list of parameters by clicking the down-arrow list of 'Parameter', specifying the suitable parameter values for the the desired set of depth images. Suitable parameter values should be straightforward to specify for all parameters except the depth thresholds and bounding box.

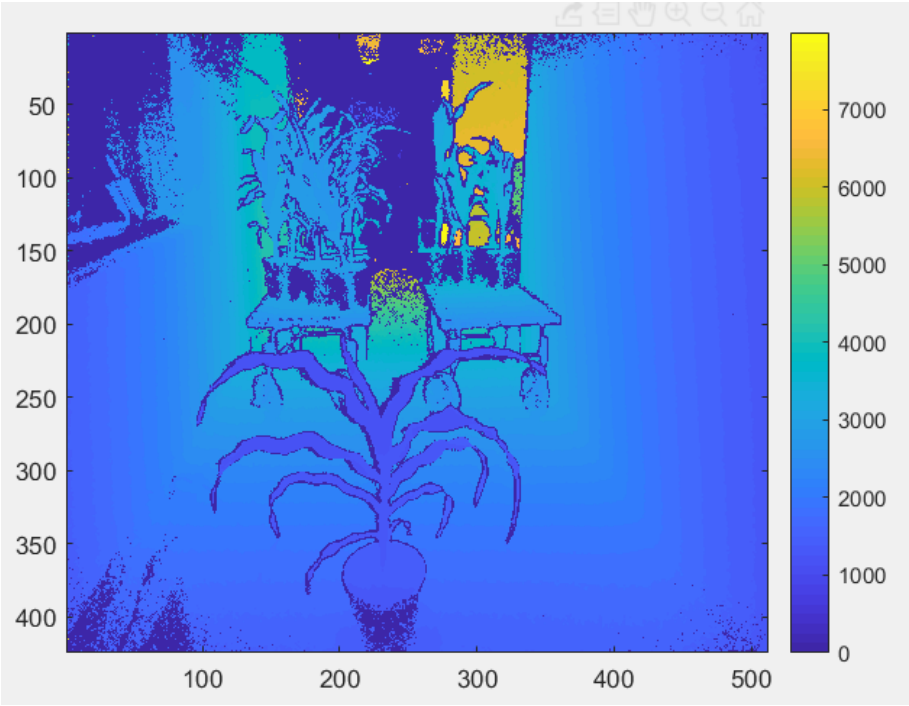


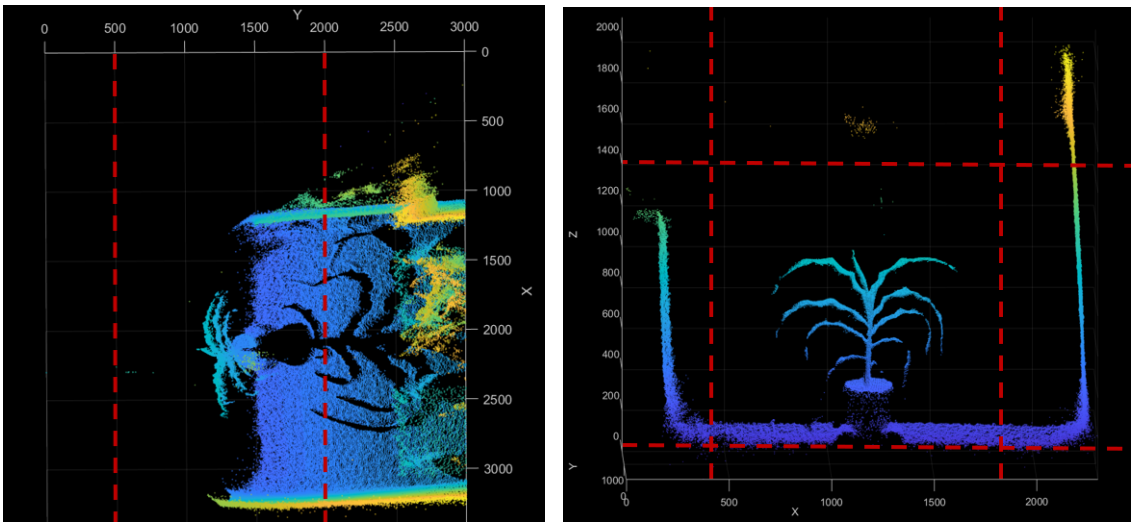
Figure 4.3: An example of a depth image.

To determine the depth thresholds, first click 'Create Parameter Files' and select 'Depth thresholds' from the drop-down list. Use the 'View depth image' button to select and view a depth image with a colour scale to indicate depth, as shown in Figure 4.3. This should allow you to get a rough estimate of the foreground and background thresholds by using the colour scale on the right to estimate the location of the plant. Enter these values into their respective boxes, then click 'View depth image for bounding box' and select a depth image. The depth

image will be segmented using the foreground and background thresholds and displayed as a point cloud. This allows for more precise thresholds to be determined. Appropriate thresholds should be chosen to cut out any objects in the foreground or background.

Once depth thresholds have been determined, enter them into their boxes and select 'Bounding box' from the drop-down list of 'Parameter'. Once again, click 'View depth image for bounding box', and rotate the point cloud to determine suitable cutoff thresholds for the bounding box parameters. Enter these into the appropriate boxes.

When specifying values for the depth thresholds or the bounding box, try to specify values that remove all objects from around the plant (excluding the pot and the surface where it is placed) without cropping too closely to the plant. This allows the parameters to be used for multiple plants that may moderately differ in size. See Figure 4.4 for examples.



(a) Example depth thresholds

(b) An example bounding box

Figure 4.4: Examples of suitable values for depth thresholds and a bounding box, depicted by the red dotted lines

Once all suitable parameters have been defined, click 'Generate parameter files' and select the folder that contains all the sets of depth images for which the parameters are suitable. Selecting a folder with sub-folders will generate the parameter files for all its subfolders, and will save them in the appropriate folder. This can be used to create parameter files for all plants at a particular time point, or even the entire dataset.

4.2.3 Analysing plants

Once parameter files have been generated, there are three options for analysing plants in the initial GUI. Analysing a plant consists of constructing a point cloud and extracting measurements from the point cloud. 'Analyse a single plant' is self-explanatory and can be used to test whether the parameters for a plant result in an accurate point cloud. 'Analyse plant(s) over time' prompts the user to select a time point folder, and then to specify the IDs of all plants to be analysed.

'Analyse all plants over time' is also self-explanatory. When a user chooses to analyse multiple plants, progress status consisting of the plant ID and distance through the dataset will be displayed in the MATLAB command window.

Assuming the parameters 'Save point cloud' and 'Save measurements' are set to true, each generated point cloud and its extracted measurements will be saved in the folder with the corresponding depth images. The saved point clouds can be viewed using 'View point cloud' in the GUI, and graphs of measurements (Figure 4.5) can be plotted using 'Plot measurements' in the initial GUI, which follows the same selection process as 'Analyse plant(s) over time'. 'Plot all measurements' is self-explanatory, and is useful for checking that measurements appear to follow the correct trends, e.g., height increases over time.

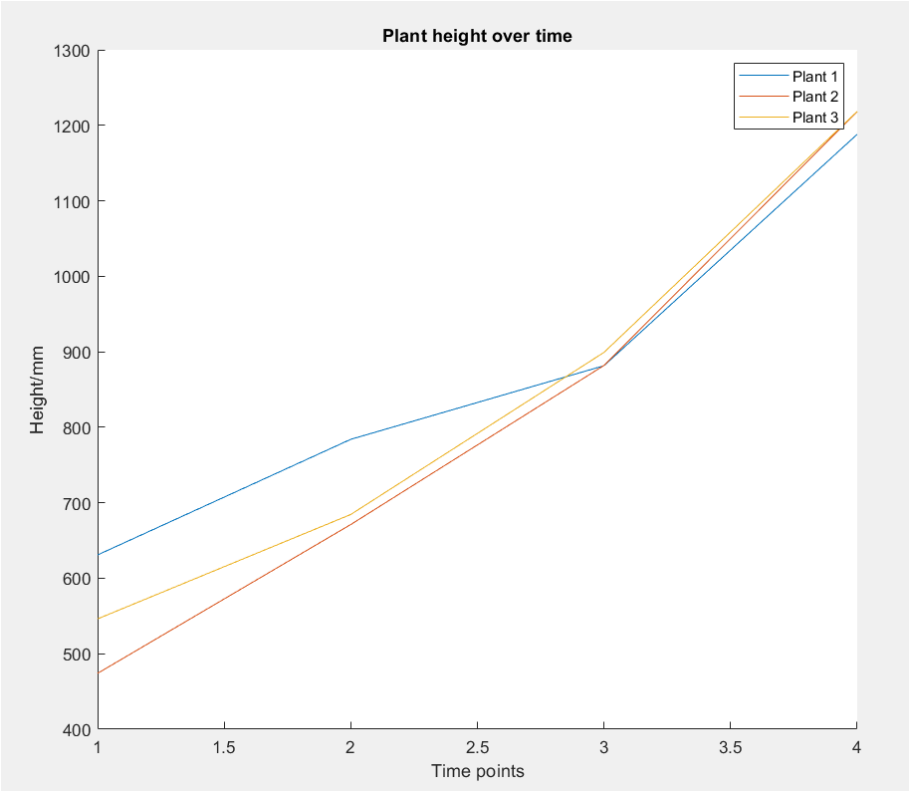


Figure 4.5: An example of a graph of the height of three plants plotted over four time points.

4.2.4 MATLAB Code

A large number of functions were created to perform the various processing tasks for the software. These are presented below, split into sections based on their broader purpose and presented in the order that they are called (where applicable). Functions that are called within other functions are indented. To be concise, functions defined within the GUI app file are not presented, and if a function is called in multiple locations it is only presented in the first place it is called. The entire codebase is presented in full in Appendix A.

GUI

- plant_analysis_app.mlapp
  - generate_parameter_files.m
    - get_end_paths.m

Control files

- analyse_over_time.m
- analyse_plant.m
- analyse_saved_point_cloud.m

Constructing 3D model

- get_depth_ims.m
- construct_point_cloud.m
  - segment_depth_im.m
  - depthImage2PC.m
    - imagePoint2worldPoint.m
  - segment_point_cloud.m
  - remove_plane.m
  - remove_pot.m
  - registerPCs.m
  - shift_reference.m
    - rotate_to_vertical.m
      - get_transformation_matrix.m
    - rotate_to_principal_axis.m
    - normalise_position.m

Extract phenotypes

- get_measurements.m
  - get_projection.m

Chapter 5

Results

To assess the accuracy of the software, a publically available dataset published by McCormick *et al.* [33] will be utilised. This dataset contains depth images of 297 plants, each imaged at four time points over the course of 17 days. Each plant at each time point has 12 depth images captured from 360° view around it (Figure 5.1), achieved by rotating the plant on a turntable and keeping the camera in a fixed position. The dataset also contains 3D meshes of each plant at each time point, created by the semi-automated reconstruction software they presented in the paper.

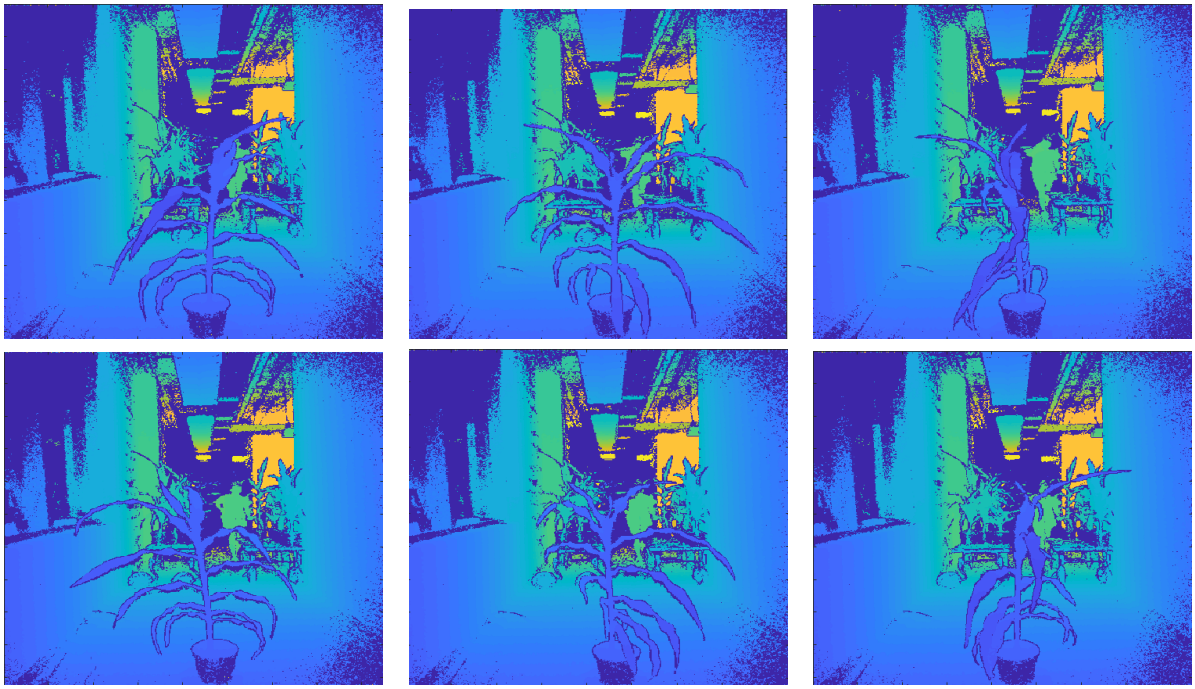


Figure 5.1: A sample of six depth images of a plant, taken at approximately 60° intervals.

McCormick *et al.* [33] demonstrated that the measurements they extracted from these meshes were strongly correlated with manual measurements taken from the plants, and showed

that these measurements were suitable for successful QTL identification. For these reasons, the meshes provided in the dataset are suitable to use as a form of ground truth with which to compare the point clouds generated by the software developed in this project, referred to as the developed software.

Additionally, the software developed by McCormick *et al.* [33] represents the state-of-the-art in 3D plant phenotyping using depth images. A qualitative comparison of the plant models generated by the state-of-the-art and the developed software is made, and the level of automation of the two pieces of software are also analysed.

5.1 Quantitative assessment

To obtain a quantitative assessment of accuracy, the developed software was used to process the entire dataset, and phenotype measurements presented in Section 3.4 were extracted from both the newly generated point clouds and the ground truth meshes **(as illustrated in Figure...) (Tardi: add a figure to show some example point clouds and the grouth truth meshes.)** The Normalised Root Mean Square Errors (NRMSEs) between the sets of phenotype measurements was calculated using

$$NRMSE = \frac{\sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}}{y_{max} - y_{min}} \ , \tag{5.1}$$

where n is the number of point clouds measured; y_i is the ground truth phenotype measurement from the ith mesh; $\hat{y}_i$ is the phenotype measurement from the ith point cloud; and y_{max} and y_{min} are respectively the maximum and minimum values of the ground truth phenotype measurements. The NRMSE allows comparisons between the accuracy of two different phenotype measures of different scales, which is important as the PADs are in the order of 10^0, whereas the convex hull volumes are in the order of 10^8.

In addition to the NRMSEs, the Pearson's correlations (ρ) between the mesh and point cloud phenotype measurements were calculated using

$$\rho = \frac{E[(Y - \mu_Y)(\hat{Y} - \mu_{\hat{Y}})]}{\sigma_Y \sigma_{\hat{Y}}} \ , \tag{5.2}$$

where E is the expected value operator, Y is the set of ground truth values for a phenotype measurement, $\hat{Y}$ is the set of measured values for a phenotype measurement, and μ_Y, $\mu_{\hat{Y}}$, σ_Y, and $\sigma_{\hat{Y}}$ are respectively the means and standard deviations of Y and $\hat{Y}$.

As shown in Table 5.1, the height, convex hull volume, and PAR have NRMSEs below 0.10 and ρ at or above 0.90, showing that the values of the phenotype measurements extracted from the point clouds and the ground truth meshes are very close, and have very strong linear correlations. The strength of the correlations is illustrated in Figure 5.2. The low errors and high correlations strongly suggest that the developed software generates point clouds suitable

for extracting these phenotype measurements at high enough accuracies to allow for applications in plant phenotyping, such as QTL mapping.

Table 5.1: The NRMSE and ρ for each phenotype measurement

Measurement	**NRMSE**	$\boldsymbol{\rho}$
Height	0.05	0.95
Convex Hull Volume	0.05	0.95
PAR	0.07	0.90
PAD (x)	0.15	0.75
PAD (y)	0.10	0.78
PAD (z)	0.10	0.74

Table 5.1 also shows that the PAD extracted along each axis show consistently higher NRMSEs and weaker correlations than the other phenotypes, with NRMSEs ranging from 0.10 to 0.15 and ρ ranging from 0.74 to 0.78 (table 5.1). A larger spread of points in Figure 5.2 also indicates this lower correlation. The measurements of PAD are still likely to be sufficiently accurate for plant phenotyping applications. However to be confident of the accuracy, further validation, possibly through comparing the PADs extracted using the developed software and PADs extracted using the conventional 2D method in [13], is needed.

5.2 Qualitative assessment

In addition to the quantitative assessment in Section 5.1, a qualititative assessment is necessary. This is due to the limitations of using the meshes provided in the dataset as ground truth. Comparison between the meshes and the generated point clouds allows conclusions to be drawn about the utility of the developed software in plant phenotyping. This is because as the meshes are sufficiently accurate to be used for these applications, and if the point clouds are sufficiently accurate with regard to the meshes, they must also be suitable for these applications. Furthermore, since the meshes may not be perfectly accurate themselves, the point clouds could be more accurate representations of the plants. However, this would not be captured by the assessment of their accuracy relative to the imperfect meshes.

To examine whether the point clouds have any advantages over the meshes, a random sample of 10 plants over each of their four time points was taken, and the point clouds and meshes were visually compared. In general the meshes and the point clouds are very similar, as supported by the quantitative assessment. However, the point clouds sometimes captured small, thin leaves to a greater degree than the meshes, which often missed them entirely. This is particularly prominent during the first time point, when the plants are smallest. This is demonstrated in Figures 5.3(a), 5.3(b), 5.3(c), and 5.3(d), which show two example pairs of a point cloud and its corresponding mesh, where the point cloud has captured the smallest leaf lowest on the plant, whereas the mesh has not.

Additionally, the generated point clouds and meshes often have slightly different stem

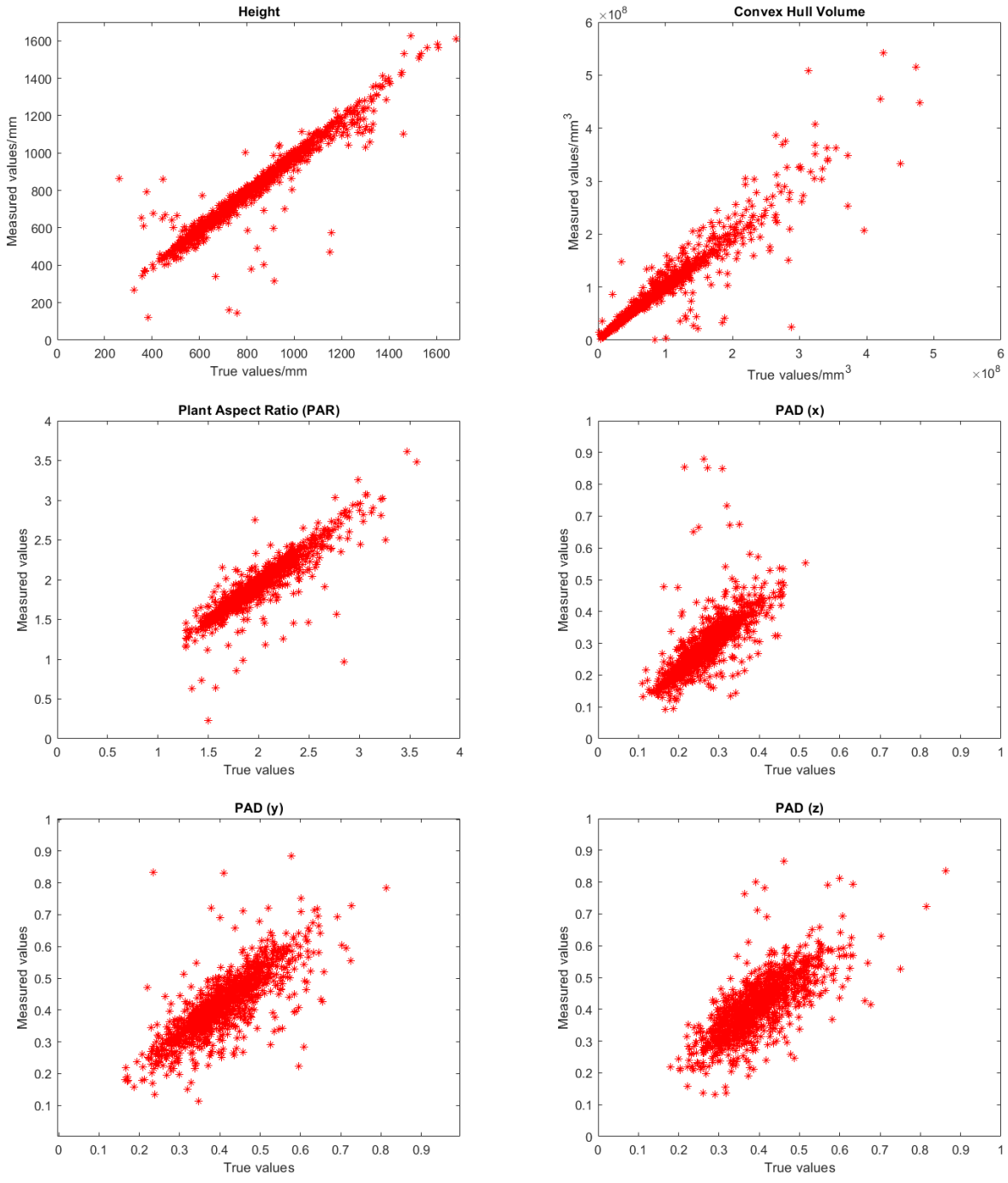


Figure 5.2: Graphs of the ground truth values against the measured values for each phenotype measurement.

angles. This is because the developed software rotates the point clouds to an upright position by aligning the plane of the top of the plant pot with the horizontal plane, as described in Section 3.3.5, whereas the meshes were aligned to an upright position manually, during a manual processing stage. This may present an additional advantage to the generated point clouds, as

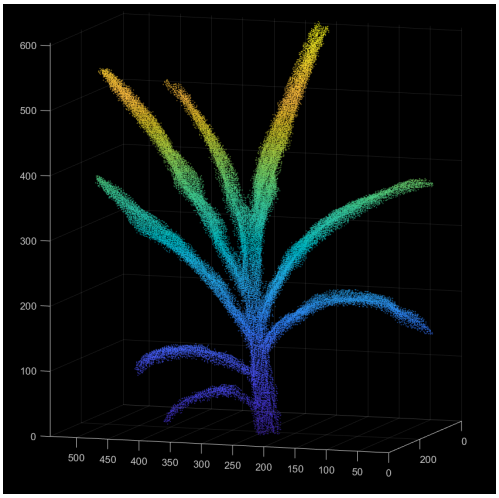
it suggests their orientation is more accurate relative to the plants, which would allow for the more accurate extraction of phenotypes such as stem angle, an important phenotype that affects yield [42]. A comparison demonstrating the difference in stem angle is shown in Figures 5.3(e) and 5.3(f).

5.2.1 Level of automation

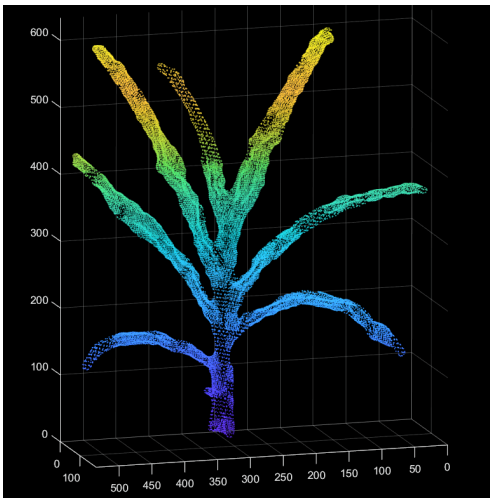
The software presented by McCormick *et al.* in [33] is described as semi-automated, due to a manual processing stage after the construction of each point cloud and before the point clouds are meshed. At this stage, errors that occurred in the registration process and noise in the point cloud are removed manually. Even assuming the processing stage is quick, in the order of a few minutes, this represents a significant amount of human time when the number of point clouds is large. The amount of intervention scales linearly with the number of plants and the number of time points, potentially reducing the feasibility of using the software for large-scale phenotyping studies. A more fully automated plant phenotyping software is thus desirable.

In contrast, the developed software requires the specification of the parameter values described in Section 4.2.2. To determine and validate suitable parameter values may take in the order of 30 minutes. With the possible exceptions of the background threshold and the bounding box, the parameters will be consistent across a plant phenotyping study, i.e., the amount of intervention is independent of the number of plants and time points. The background threshold and bounding box only need to be specified when there is a significant change in camera position relative to the plant, or when objects are introduced into the area surrounding the plant. Automated plant phenotyping systems employed in large-scale studies generally keep both of these variables constant, resulting in a single background threshold and bounding box needing to be specified. Even when data collection is not fully automated, these variables are generally kept constant for each imaging session.

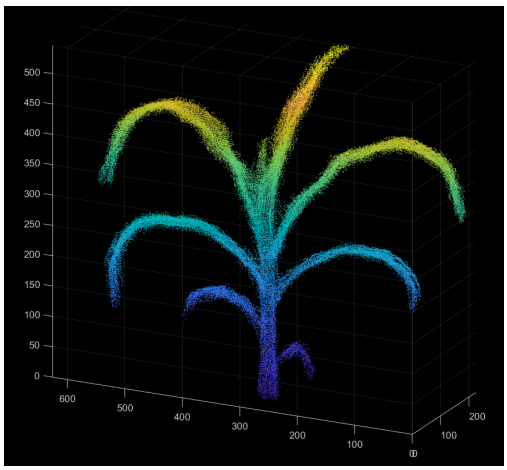
In the case of the dataset used for evaluation in this project, these variables needed to be specified at each different time point, i.e., just four times for a dataset of 1,188 sets of depth images. This represents an estimated 120 minutes of human time to utilise the developed software, compared to at least 20 hours of time for the state-of-the-art, even assuming the manual processing stage takes just a minute to complete. This represents an order of magnitude less human time required, for a moderately large dataset. This makes the developed software much more suitable for large-scale highly automated phenotyping studies than the state-of-the-art.



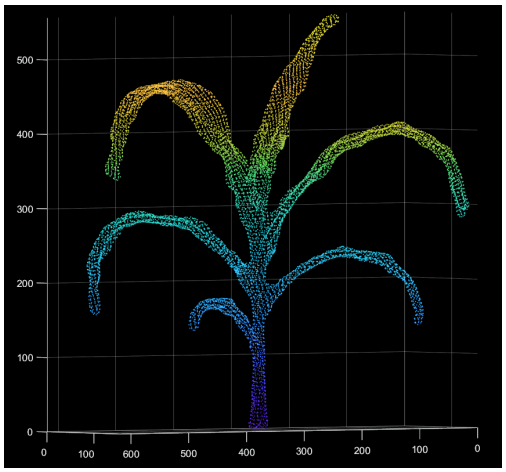
(a) Point cloud 1



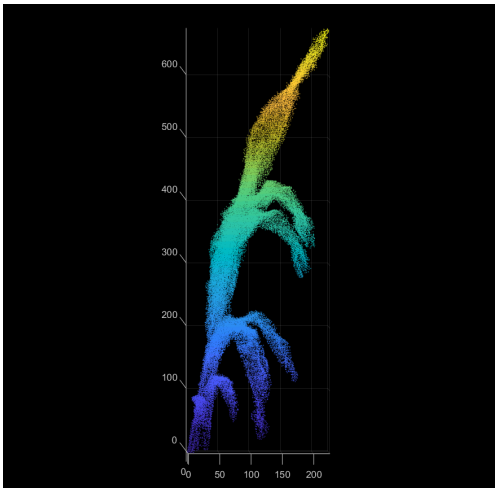
(b) Mesh 1



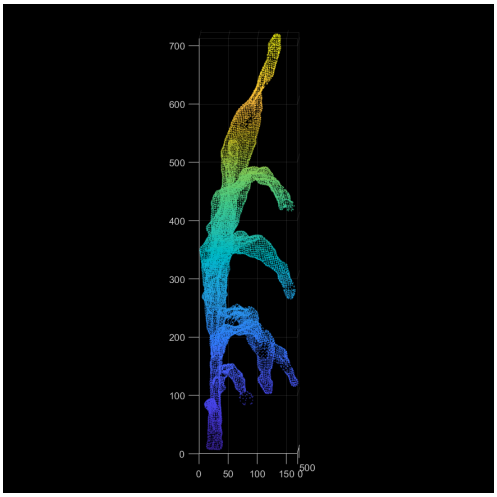
(c) Point cloud 2



(d) Mesh 2



(e) Point cloud 3



(f) Mesh 3

Figure 5.3: Three point cloud and mesh pairs. In the first and second pairs the point cloud has captured a small leaf (the smallest leaf lowest on the plant) that the mesh has not. In the third pair there is a difference in stem angle.

Chapter 6

Conclusions and future work

The quantitative results presented in Section 5.1 provide strong evidence for the utility of the developed software in important plant phenotyping applications, such as QTL mapping. The software is able to construct point clouds and extract measurements of plant height, convex hull volume, PAR and PAD, and these measurements show strong correlations with measurements obtained using the state-of-the-art depth imaging software. Furthermore, in certain circumstances the developed software offers qualitative improvements, better capturing small leaves and potentially providing point clouds with a more accurate orientation.

Further validation of the accuracy of generated point clouds using an external source of ground truth would provide additional confidence for the utility of using depth imaging for plant phenotyping. Ground truth could be obtained using manual measurements of plants, or ideally through the construction of 3D models using other methods such as X-ray microcomputed Tomography (μCT). μCT produces highly accurate point clouds, but is not suitable for large scale phenotyping due to the length of time it takes to construct models, and the high equipment cost [25]. As such, it can be useful for validation of more scalable methods.

The developed software is more fully automated, requiring less human intervention than the state-of-the-art method in [33]. In particular, the amount of human time required is generally constant for high-throughput phenotyping systems even as the number of plants being imaged increases, in comparison to the state-of-the-art software where it increases linearly. For the dataset used for evaluating the software in this report, the estimated human time required to analyse the dataset using the developed software is an order of magnitude lower than for the state-of-the-art software.

An important area of future work will be to continue to develop more fully automated plant phenotyping pipelines. Improving the method of segmenting the depth images or point cloud into plant and non-plant points will help to achieve this. A promising avenue for developing automated segmentation is the use of Red Green Blue-Depth (RGB-D) imaging, which combines colour images and depth images. This would allow plant segmentation methods developed for use on RGB images to be applied. Plant segmentation from RGB images has been well researched [43], and existing techniques could be applied directly to RGB-D images to solve the

segmentation problem. RGB-D imaging can be conducted using readily accessible equipment, such as the low-cost Microsoft Kinect camera, making it suitable for use in large scale automated phenotyping studies.

The usage of RGB-D images also presents the possibility of enhancing point clouds with additional data, in this case colour data. Colour is an important data source for monitoring plant stress, and incorporating 3D and colour data would allow for the extraction of additional phenotype information. Future research could also investigate integrating data obtained from hyperspectral imaging, allowing many new phenotypes to be extracted, including, for example, important indicators of plant water use [27].

Segmenting plant point clouds into their organs, i.e., the leaves, stem, etc., allows component phenotypes such as leaf number, leaf length, and stem angle to be obtained. Extending the developed software to have this capability would increase its utility in plant phenotyping. This could be achieved by applying existing 3D plant model segmentation algorithms, or through the development of novel methods. A promising method was presented by Shi *et al.* [44], which used deep learning and multi-view RGB images to produce point clouds segmented with greater accuracy than was achievable when solely segmenting 2D images. Their software was only tested on plant seedlings, so future work could involve combining this method with RGB-D imaging to construct segmented models of large plants, where relying solely on multi-view RGB images can result in gaps in data due to self-occlusions.

Bibliography

[1] FAO, "The state of food security and nutrition in the world," 2019.

[2] R. Black, C. Victora, S. Walker, Z. Bhutta, P. Christian, M. de Onis, M. Ezzati, S. Grantham-McGregor, J. Katz, R. Martorell, and R. Uauy, "Maternal and child undernutrition and overweight in low-income and middle-income countries," *Lancet*, vol. 382, pp. 427–451, 2013.

[3] United Nations, Department of Economic and Social Affairs, Population Division, "World population prospects 2019," 2019.

[4] J. Cleland, "World population growth; past, present and future," *Environmental and Resource Economics*, vol. 55, 2013.

[5] FAO, "The future of food and agriculture – trends and challenges," 2017.

[6] ——, "Global agriculture towards 2050," 2009.

[7] S. S. Myers, M. R. Smith, S. Guth, C. D. Golden, B. Vaitla, N. D. Mueller, A. D. Dangour, and P. Huybers, "Climate change and global food systems: Potential impacts on food security and undernutrition," *Annual Review of Public Health*, vol. 38, no. 1, pp. 259–277, 2017.

[8] R.T.Watson, M.C.Zinyowera, and R.H.Moss, *The Regional Impacts of Climate Change: An Assessment of Vulnerability. Summary for Policymakers.* Cambridge University Press, 1997, accessed February 25, 2019. [Online]. Available: https://archive.ipcc.ch/ipccreports/sres/regional/index.php?idp=0

[9] F. Fiorani and U. Schurr, "Future scenarios for plant phenotyping," *Annual Review of Plant Biology*, vol. 64, no. 1, pp. 267–291, 2013.

[10] F. Tardieu, L. Cabrera-Bosquet, T. Pridmore, and M. Bennet, "Plant phenomics, from sensors to knowledge," *Current Biology*, vol. 27, no. 15, pp. 770–783, 08 2017.

[11] L. Li, Q. Zhang, and D. Huang, "A review of imaging techniques for plant phenotyping," *Sensors (Basel, Switzerland)*, vol. 14, pp. 20 078–20 111, 11 2014.

[12] S. Paulus, "Measuring crops in 3d: using geometry for plant phenotyping," *Plant Methods*, vol. 15, no. 103, 2019.

[13] S. D. Choudhury, S. Bashyam, Y. Qiu, A. Samal, and T. Awada, "Holistic and component plant phenotyping using temporal image sequence," *Plant Methods*, vol. 14, no. 35, 2018.

[14] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and Vision Computing*, vol. 10, no. 3, pp. 145–155, 1992.

[15] P. Besl and N. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[16] M. P. Pound, A. P. French, E. H. Murchie, and T. P. Pridmore, "Automated recovery of three-dimensional models of plant shoots from multiple color images," *Plant Physiology*, vol. 166, no. 4, pp. 1688–1698, 2014.

[17] K. Kutulakos and S. Seitz, "A theory of shape by space carving," *International Journal of Computer Vision*, vol. 38, no. 3, pp. 199–218, 2000.

[18] W. Lorensen and H. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 163–169, 1987.

[19] J. Gibbs, M. Pound, A. French, D. Wells, E. Murchie, and T. Pridmore, "Approaches to three-dimensional reconstruction of plant shoot topology and geometry," *Functional Plant Biology*, vol. 44, 2016.

[20] T. Bell, B. Li, and S. Zhang, *Structured Light Techniques and Applications*, 02 2016.

[21] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[22] C. Wu, "Towards linear-time incremental structure from motion," *Proceedings - 2013 International Conference on 3D Vision, 3DV 2013*, pp. 127–134, 2013.

[23] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz, "Multicore bundle adjustment," in *CVPR 2011*, 2011, pp. 3057–3064.

[24] K. Cheung, S. Baker, and T. Kanade, "Shape-from-silhouette across time part i: Theory and algorithms," *International Journal of Computer Vision*, vol. 62, pp. 221–247, 2005.

[25] J. A. Gibbs, M. Pound, A. P. French, D. M. Wells, E. Murchie, and T. Pridmore, "Plant phenotyping: An active vision cell for three-dimensional plant shoot reconstruction," *Plant Physiology*, vol. 178, no. 2, pp. 524–534, 2018. [Online]. Available: http://www.plantphysiol.org/content/178/2/524

[26] B. Elnashef, S. Filin, and R. N. Lati, "Tensor-based classification and segmentation of three-dimensional point clouds for organ-level plant phenotyping and growth analysis," *Computers and Electronics in Agriculture*, vol. 156, pp. 51–61, 2019.

[27] P. Tripodi, D. Massa, A. Venezia, and T. Cardi, "Sensing technologies for precision phenotyping in vegetable crops: Current status and future challenges," *Trends in Plant Science*, vol. 23, no. 10, pp. 883–898, 2018.

[28] J. G. andd Nathan Miller, E. Spalding, S. Kaeppler, and N. de Leon, "Tips: a system for automated image-based phenotyping of maize tassels," *Plant Methods*, vol. 13, 2017.

[29] A. K. Singh, B. Ganapathysubramanian, S. Sarkar, and A. Singh, "Deep learning for plant stress phenotyping: Trends and future perspectives," *Trends in Plant Science*, vol. 23, pp. 883–898, 2018.

[30] J. Humplík, D. Lazár, A. Husičková, and L. Spíchal, "Automated phenotyping of plant shoots using imaging methods for analysis of plant stress responses – a review," *Plant Methods*, vol. 11, 2015.

[31] R. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *IEEE Journal of Robotics and Automation*, vol. 3, pp. 323–344, 1987.

[32] S. D. Choudhury, S. Maturu, V. Stoerger, A. Samal, and T. Awada, "3d image-based plant phenotyping research: Dataset, algorithm and analysis," 2019.

[33] R. McCormick, S. Truong, and J. Mullet, "3d sorghum reconstructions from depth images identify qtl regulating shoot architecture," *Plant Physiology*, vol. 172, pp. 823–834, 2016.

[34] MATLAB, *version 9.6.0 (R2019a)*. Natick, Massachusetts, United State: The MathWorks Inc., 2020.

[35] The MathWorks, Inc., *Computer Vision Toolbox*, Natick, Massachusetts, United State, 2020. [Online]. Available: https://uk.mathworks.com/help/vision/

[36] P. Torr and A. Zisserman, "Mlesac: A new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding*, vol. 78, pp. 138–156, 2000.

[37] The MathWorks, Inc., "3-d point cloud registration and stitching," 2020. [Online]. Available: https://uk.mathworks.com/help/vision/examples/3-d-point-cloud-registration-and-stitching.html

[38] R. Rusu, Z. Marton, M. Dolha, and M. Beetz, "Towards 3d point cloud based object maps for household environments," *Robotics and Autonomous Systems Journal*, 2008.

[39] The MathWorks, Inc., "convull," 2020. [Online]. Available: https://uk.mathworks.com/help/matlab/ref/convhull.html

[40] S. Das Choudhury, V. Stoerger, A. Samal, J. Schnable, Z. Liang, and J.-G. Yu, "Automated vegetative stage phenotyping analysis of maize plants using visible light images," in *KDD: Data Science for Food , Energy and Water*, 08 2016.

[41] J. D'Errico, "A suite of minimal bounding objects," 2020. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/34767-a-suite-of-minimal-bounding-objects

[42] S. D. Choudhury, S. Goswami, S. Bashyam, A. Samal, and T. Awada, "Automated stem angle determination for temporal plant phenotyping analysis," *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pp. 2022–2029, 2017.

[43] E. Hamuda, M. Glavin, and E. Jones, "A survey of image processing techniques for plant extraction and segmentation in the field," *Computers and Electronics in Agriculture*, vol. 172, pp. 184–199, 2016.

[44] W. Shi, R. Zedde, H. Jiang, and G. Kootstra, "Plant-part segmentation using deep learning and multi-view vision," *Biosystems Engineering*, vol. 187, pp. 81–95, 2019.

Appendices

A.1 Program implementation

This appendix contains all of the processing functions written for this project. The only file not included is the GUI app, due to its length and as it is not relevant to the main purpose of the project, i.e., it is not part of the software that actually constructs and measures point clouds. The functions are presented in the following pages in alphabetical order.

```
function [pcs, measurements_4D] = analyse_over_time(base_path,
 plant_id)
    % Calls analyse_plant and passes it the path for each time point
 for a
    % particular plant

    pcs = {};
    measurements_4D = {};

    count = 0;

    folder_contents = dir(base_path);

    for i = 1:numel(folder_contents)
        item = folder_contents(i);

        if item.isdir && ~strcmp(item.name, '.') &&
 ~strcmp(item.name, '..')
            count = count + 1;
            path = fullfile(base_path, item.name, plant_id);

            [pc, measurements] = analyse_plant(path);

            pcs{count} = pc;
            measurements_4D{count} = measurements;
        end
    end
end
```

Published with MATLAB® R2019a

```
function [pc, measurements] = analyse_plant(path)
    % Analyses a plant by constructing its point cloud and extracting
    % measurements

    load(fullfile(path, 'parameters.mat'));

    depth_ims = get_depth_ims(path, im_height, im_width, im_no,
 plant_axis);

    pc = construct_point_cloud(depth_ims, do_remove_plane,
 do_remove_pot, ...
        depth_thresholds, bounding_box);

    [height, convex_hull_volume, PD_x, PD_y, PD_z,
 plant_aspect_ratio, ...
        bi_angular_convex_hull_area_ratio] = get_measurements(pc);

    measurements.height = height;
    measurements.plant_aspect_ratio = plant_aspect_ratio;
    measurements.bi_angular_convex_hull_area_ratio =
 bi_angular_convex_hull_area_ratio;
    measurements.convex_hull_volume = convex_hull_volume;
    measurements.PD_x = PD_x;
    measurements.PD_y = PD_y;
    measurements.PD_z = PD_z;

    if save_pc
        pc_filename = fullfile(path, 'pointCloud.ply');
        pcwrite(pc, pc_filename);
    end

    if save_measurements
        save(fullfile(path, 'pointCloudMeasurements.mat'), '-
struct', 'measurements');
    end
end
```

Published with MATLAB® R2019a

```
function measurements = analyse_saved_point_cloud(path,
 save_measurements)
    % Extracts measurements from a saved point cloud

    pc = pcread(path);

    split_path = split(path, '\');
    folder_path = join(split_path(1:end-1), '\');
    full_file_name = split(split_path{end}, '.');
    file_name = full_file_name{1};

    [height, convex_hull_volume, PD_x, PD_y, PD_z,
 plant_aspect_ratio, ...
        bi_angular_convex_hull_area_ratio] = get_measurements(pc);

    measurements.height = height;
    measurements.plant_aspect_ratio = plant_aspect_ratio;
    measurements.bi_angular_convex_hull_area_ratio =
 bi_angular_convex_hull_area_ratio;
    measurements.convex_hull_volume = convex_hull_volume;
    measurements.PD_x = PD_x;
    measurements.PD_y = PD_y;
    measurements.PD_z = PD_z;

    if save_measurements
        save(fullfile(folder_path{1},
 strcat(file_name, 'Measurements.mat')), '-struct', 'measurements');
    end
end
```

Published with MATLAB® R2019a

```
function pc = construct_point_cloud(depth_ims, do_remove_plane,
 do_remove_pot, depth_thresholds, bounding_box)
    % Constructs a point cloud from a set of depth images

    is_first_plant = true;
    is_first_scene = true;

    pc_count = 0;

    foreground_cutoff = depth_thresholds(1);
    background_cutoff = depth_thresholds(2);

    rmse_cutoff = 15;

    reference_vector = [0, 0, 1];

    for i = 1:size(depth_ims, 3)
        depth_im = uint16(depth_ims(:, :, i));

        segmented_im = segment_depth_im(depth_im, foreground_cutoff,
 background_cutoff);

        pc_count = pc_count + 1;

        pc = depthImage2PC(segmented_im);

        pc = segment_point_cloud(pc, bounding_box);

        if do_remove_plane && do_remove_pot
            [pc, plane_model] = remove_plane(pc, reference_vector,
 bounding_box(4));
            [pc, pc_pot_plane] = remove_pot(pc, plane_model.Normal,
 bounding_box(4));
        elseif do_remove_plane
            pc = remove_plane(pc, reference_vector, bounding_box(4));
        elseif do_remove_pot
            [pc, pc_pot_plane] = remove_pot(pc, reference_vector,
 bounding_box(4));
        end

        pc = pcdownsample(pc, 'gridAverage', 2);
        pc_new = pcdenoise(pc, 'NumNeighbors', 50, 'Threshold', 0.01);

        if ~is_first_plant
            if is_first_scene
                [pc_scene, tform_prev, tform_total, is_first_scene,
 rmse] = registerPCs(0, pc_base, pc_new, 0, 0, is_first_scene,
 rmse_cutoff);
            else
                [pc_scene, tform_prev, tform_total, is_first_scene,
 rmse] = registerPCs(pc_scene, pc_base, pc_new, tform_prev,
 tform_total, is_first_scene, rmse_cutoff);
```

```
            end
        end

        is_first_plant = false;

        pc_base = pc_new;
    end

    pc = pcdownsample(pc_scene, 'gridAverage', 1);

    pc = pcdenoise(pc, 'NumNeighbors', 50, 'Threshold', 0.5);

    if do_remove_pot
        pc = shift_reference(pc, pc_pot_plane, true);
    else
        pc = shift_reference(pc, 0, false);
    end
end
```

Published with MATLAB® R2019a

```
function pc = depthImage2PC(depth_im)
    % Converts a depth image to a point cloud

    im_dimensions = size(depth_im);

    height = im_dimensions(1);
    width = im_dimensions(2);

    points = zeros(height*width, 3);

    for i = 1:height
        for j = 1:width
            z = double(depth_im(i, j));

            [x, y] = imagePoint2worldPoint(j, i, z, height, width);

            points(((i - 1) * width) + j, :) = [x, z, -y];
        end
    end

    pc = pointCloud(points);
    pc = normalise_position(pc);
    pc = pcdownsample(pc, 'gridAverage', 1);
end
```

Published with MATLAB® R2019a

```
function generate_parameter_file(path, im_height, im_width, im_no,
 plant_axis, do_remove_plane, do_remove_pot, depth_thresholds,
 bounding_box, save_pc, save_measurements)
    % Saves a parameter file containing a structure with the specified
    % parameter values

    parameters.im_height = im_height;
    parameters.im_width = im_width;
    parameters.im_no = im_no;
    parameters.plant_axis = plant_axis;
    parameters.do_remove_plane = do_remove_plane;
    parameters.do_remove_pot = do_remove_pot;
    parameters.depth_thresholds = depth_thresholds;
    parameters.bounding_box = bounding_box;
    parameters.save_pc = save_pc;
    parameters.save_measurements = save_measurements;

    save(fullfile(path, 'parameters.mat'), '-struct', 'parameters');
end
```

Published with MATLAB® R2019a

```
function depth_ims = get_depth_ims(path, im_height, im_width, im_no,
 plant_axis)
    % Retrieves a set of depth images from the folder specified by
 path

    folder_contents = dir(path);

    rotate = false;

    if strcmp(plant_axis, 'x')
        height_store = im_height;
        im_height = im_width;
        im_width = height_store;

        rotate = true;
    end

    depth_ims = zeros(im_height, im_width, im_no);

    im_count = 0;

    for i = 1:numel(folder_contents)
        item = folder_contents(i);

        if ~item.isdir && contains(item.name, 'DepthImage')
            im_count = im_count + 1;

            file_path = fullfile(path, item.name);

            depth_im = imread(file_path);

            if rotate
               depth_im = imrotate(depth_im, 90);
            end

            depth_ims(:, :, im_count) = depth_im;
        end
    end
end
```

Published with MATLAB® R2019a

```
function paths = get_end_paths(directory, paths)
    % Returns a cell array containing the base paths of a folder,
 i.e., the
    % folders that only have files in them

    folder_contents = dir(directory);

    end_path = true;

    for i = 1:numel(folder_contents)
        item = folder_contents(i);
        if item.isdir && ~strcmp(item.name, '.') &&
 ~strcmp(item.name, '..')
            end_path = false;
            paths = get_end_paths(fullfile(directory, item.name),
 paths);
        end
    end

    if end_path
        paths{numel(paths) + 1} = directory;
    end
end
```

Published with MATLAB® R2019a

```
function [height, convex_hull_volume, PD_x, PD_y, PD_z,
 plant_aspect_ratio, bi_angular_convex_hull_area_ratio] =
 get_measurements(pc)
    % Extracts measurements from a point cloud

    pc = shift_reference(pc, 0, false);

    pc_matrix = double(pc.Location);

    X = pc_matrix(:, 1);
    Y = pc_matrix(:, 2);
    Z = pc_matrix(:, 3);

    height = max(Z) - min(Z);

    [~, XZ_conv_hull_area] = convhull(X, Z);
    [~, YZ_conv_hull_area] = convhull(Y, Z);

    bi_angular_convex_hull_area_ratio = XZ_conv_hull_area/
YZ_conv_hull_area;

    [~, XY_conv_hull_area] = convhull(X, Y);

    [~, min_enclosing_circle_radius] = minboundcircle(X, Y);

    plant_aspect_ratio = height/min_enclosing_circle_radius;

    [~, convex_hull_volume] = convhull(X, Y, Z);

    XZ_proj = get_projection(X, Z);
    YZ_proj = get_projection(Y, Z);
    XY_proj = get_projection(X, Y);

    PD_x = nnz(XZ_proj)/XZ_conv_hull_area;
    PD_y = nnz(YZ_proj)/YZ_conv_hull_area;
    PD_z = nnz(XY_proj)/XY_conv_hull_area;
end
```

Published with MATLAB® R2019a

```
function projection = get_projection(X, Y)
    % Returns a binary image created from a set of 2D points

    round_X = int32(X);
    round_Y = int32(Y);

    norm_X = round_X - min(round_X) + 1;
    norm_Y = round_Y - min(round_Y) + 1;

    unique_coords = unique([norm_X, norm_Y], 'rows');
    unique_X = unique_coords(:, 1);
    unique_Y = unique_coords(:, 2);

    projection = zeros(max(norm_Y), max(norm_X));

    directions = [0, 0; -1, 0; 1, 0; 0, -1; 0, 1; 1, 1; 1, -1; -1, 1;
 -1, -1];

    for i = 1:numel(unique_X)
        for j = 1:size(directions, 1)
            try
                projection(unique_Y(i) + directions(j, 1), unique_X(i)
 + directions(j, 2)) = 1;
            catch
            end
        end
    end

    projection = bwmorph(projection, 'close');
    projection = imfill(projection, 'holes');
end
```

Published with MATLAB® R2019a

```
function tform_matrix = get_transformation_matrix(angle, axis)
    % Computes the affine transformation matrix that rotates a set of
    % points by a given angle around a given axis

    if strcmp(axis, 'x')
        tform_matrix = [1 0 0 0;
            0 cos(angle) sin(angle) 0;
            0 -sin(angle) cos(angle) 0;
            0 0 0 1];
    elseif strcmp(axis, 'y')
        tform_matrix = [cos(angle) 0 -sin(angle) 0;
            0 1 0 0;
            sin(angle) 0 cos(angle) 0;
            0 0 0 1];
    else
        tform_matrix = [cos(angle) sin(angle) 0 0;
            -sin(angle) cos(angle) 0 0;
            0 0 1 0;
            0 0 0 1];
    end
end
```

Published with MATLAB® R2019a

```
function [world_x, world_y] = imagePoint2worldPoint(x, y, z, height,
 width)
    % Converts a point in a depth image to a point in 3D space based
 on the
    % focal length of the camera. Default focal length is calculated
 from
    % a scale factor that was "empirically determined to represent the
    % distance between two pixels at a depth of 500mm", determined by
 the
    % publishers of the original sorghum dataset. See
    % https://github.com/MulletLab/SorghumReconstructionAndPhenotyping
 for
    % more information

    pixel_to_mm_scale_factor = 1.4089;
    focal_length = 1/pixel_to_mm_scale_factor;
    base_distance = 500;

    normalised_z = z/base_distance;

    world_x = (x - width/2) * normalised_z / focal_length;
    world_y = (y - height/2) * normalised_z / focal_length;
end
```

Published with MATLAB® R2019a

```
function pc_normalised = normalise_position(pc)
    % Shifts a point cloud such that the minimum values of x, y, z are
 0

    xyz_shifted = pc.Location;

    for i = 1:3
        xyz_shifted(:, i) = xyz_shifted(:, i) - min(xyz_shifted(:,
 i));
    end

    pc_normalised = pointCloud(xyz_shifted);
end
```

Published with MATLAB® R2019a

```
function [pc_registered, tform_prev, tform_total, is_first_scene,
 rmse] = registerPCs(pc_scene, pc_base, pc_new, tform_prev,
 tform_total, is_first_scene, rmse_cutoff)
    % Registers a new point cloud into the overall point cloud

    grid_size = 2;

    moving = pcdownsample(pc_new, 'gridAverage', grid_size);
    fixed = pcdownsample(pc_base, 'gridAverage', grid_size);

    if is_first_scene
        [tform, ~, rmse] = pcregistericp(moving, fixed, 'Extrapolate',
 true);

        if rmse < rmse_cutoff
            tform_total = tform;
            tform_prev = tform;

            is_first_scene = false;

            pc_aligned = pctransform(moving, tform_total);

            pc_registered = pcmerge(pc_base, pc_aligned, grid_size);

        else
            tform_total = 0;
            tform_prev = 0;

            is_first_scene = true;

            pc_registered = 0;
        end

    else
        [tform, ~, rmse] = pcregistericp(moving, fixed, 'Extrapolate',
 true, ...
        'MaxIterations', 20);

        temp_tform_total = affine3d(tform_total.T * tform.T);
        pc_aligned = pctransform(moving, temp_tform_total);

        moving = pcdownsample(pc_aligned, 'gridAverage', grid_size);
        fixed = pcdownsample(pc_scene, 'gridAverage', grid_size);

        [tform_tidy, ~, rmse_tidy] = pcregistericp(moving, fixed, ...
            'Extrapolate', true);

        if rmse_tidy < rmse_cutoff
            tform_total = affine3d(temp_tform_total.T * tform_tidy.T);
            tform_prev = tform;

            pc_registered = pctransform(pc_aligned, tform_tidy);
```

```
            pc_registered = pcmerge(pc_scene, pc_registered,
 grid_size);

        else
            tform_total = affine3d(tform.T * tform_total.T);
            pc_registered = pc_scene;
        end

        rmse = min(rmse, rmse_tidy);
    end
end
```

Published with MATLAB® R2019a

```
function [pc_remain, plane_model] = remove_plane(pc, reference_vector,
 plant_side_point)
    % Fits a plane to a point cloud and removes all points within the
    % plane, as well as all points on the non-plant side of the plane

    max_distance = 50;
    max_angle = 25;

    [plane_model, inlier_indices, outlier_indices] =
 pcfitplane(pc, ...
            max_distance, reference_vector, max_angle);

    pc_remain = select(pc, outlier_indices);
    pc_plane = select(pc, inlier_indices);

    z_coord = -plane_model.Parameters(4)/plane_model.Parameters(3);

    height = size(pc_remain.Location, 1);

    distances = zeros(height, 1);

    for i = 1:height
        distances(i) = (pc_remain.Location(i, :) - [0, 0, z_coord]) *
 plane_model.Normal';
    end

    reference_distance = [0, 0, plant_side_point] *
 plane_model.Normal';

    D = distances < 0;

    if reference_distance < 0
        plant_indices = find(D);
    else
        plant_indices = find(~D);
    end

    pc_remain = select(pc_remain, plant_indices);
end
```

Published with MATLAB® R2019a

```
function [plant_pc, pot_plane] = remove_pot(pc, reference_vector,
 plant_side_point)
    % Fits a plane to the top of a pot and removes the plane points
 and all
    % points on the non-plant side of the plane

    max_distance = 30;
    max_angle = 20;
    confidence = 60;

    [plane_model, inlier_indices, outlier_indices] =
 pcfitplane(pc, ...
            max_distance, reference_vector, max_angle, 'Confidence',
 confidence);

    pot_plane = select(pc, inlier_indices);
    remain_pc = select(pc, outlier_indices);

    z_coord = -plane_model.Parameters(4)/plane_model.Parameters(3);

    height = size(remain_pc.Location, 1);

    distances = zeros(height, 1);

    for i = 1:height
        distances(i) = (remain_pc.Location(i, :) - [0, 0, z_coord]) *
 plane_model.Normal';
    end

    reference_distance = [0, 0, plant_side_point] *
 plane_model.Normal';

    D = distances < 0;

    if reference_distance < 0
        plant_indices = find(D);
        pot_indices = find(~D);
    else
        plant_indices = find(~D);
        pot_indices = find(D);
    end

    plant_pc = select(remain_pc, plant_indices);
    pot_pc = select(remain_pc, pot_indices);
    pot_pc = pcmerge(pot_pc, pot_plane, 1);
end
```

Published with MATLAB® R2019a

```
function principal_pc = rotate_to_principal_axis(pc)
    % Rotates a point cloud so that the widest axis is parallel to the
 x
    % axis

    max_width = 0;
    principal_angle = 0;

    interval = 5;
    angle = interval*pi/180;

    tform_matrix = get_transformation_matrix(angle, 'z');
    tform = affine3d(tform_matrix);

    rotated_pc = pc;

    for i = 0:interval:180
        pc_matrix = rotated_pc.Location;
        X = pc_matrix(:, 1);

        width = max(X) - min(X);

        if width > max_width
            max_width = width;
            principal_angle = i*pi/180;
        end

        rotated_pc = pctransform(rotated_pc, tform);
    end

    tform_matrix = get_transformation_matrix(principal_angle, 'z');
    tform = affine3d(tform_matrix);
    principal_pc = pctransform(pc, tform);
end
```

Published with MATLAB® R2019a

```
function pc = rotate_to_vertical(pc_plant, pc_pot)
    % Registers the extracted pot plane to a generated horizontal
 plane,
    % and applies the resultant transfrom to rotate the plant point
 cloud
    % to its vertical position

    num_points = 5000;
    width = 200;

    plane = zeros(num_points, 3);
    plane(:, 1) = randi(width, num_points, 1);
    plane(:, 2) = randi(width, num_points, 1) - width;

    pc_plane = pointCloud(plane);

    pc_pot = pcdenoise(pc_pot, 'NumNeighbors', 10, 'Threshold', 0.05);

    [~, ~, tform_total, ~, ~] = registerPCs(0, pc_plane, pc_pot, 0, 0,
 true, inf);

    [~, max_indices] = max(pc_plant.Location);
    [~, min_indices] = min(pc_plant.Location);

    max_index = max_indices(3);
    min_index = min_indices(3);

    pc = pctransform(pc_plant, tform_total);

    if pc.Location(max_index, 3) < pc.Location(min_index, 3)
        switched = true;
    else
        switched = false;
    end

    if switched
        tform_flip = affine3d(get_transformation_matrix(pi, 'x'));
        pc = pctransform(pc, tform_flip);
    end
end
```

Published with MATLAB® R2019a

```
function segmented_im = segment_depth_im(depth_im, min_threshold,
 max_threshold)
    % Returns a segmented depth image with the foreground and
 background
    % removed based on the specified thresholds

    depth_im(depth_im > max_threshold | depth_im < min_threshold) = 0;

    segmented_im = depth_im;
end
```

Published with MATLAB® R2019a

```
function pc_segmented = segment_point_cloud(pc, bounding_box)
    % Segments a point cloud based on the specified bounding box

    points = pc.Location;
    x = points(:, 1);
    y = points(:, 2);
    z = points(:, 3);

    segmented_indices = x > bounding_box(1) & x < bounding_box(2)
 & ...
        z > bounding_box(3) & z < bounding_box(4);

    seg_x = x(segmented_indices);
    seg_y = y(segmented_indices);
    seg_z = z(segmented_indices);

    pc_segmented = pointCloud([seg_x, seg_y, seg_z]);
end
```

Published with MATLAB® R2019a

```
function pc = shift_reference(pc_plant, pc_pot, do_vertical_rotate)
    % Shifts a point cloud to the standardised position

    if do_vertical_rotate
        pc = rotate_to_vertical(pc_plant, pc_pot);
    else
        pc = pc_plant;
    end

    pc = rotate_to_principal_axis(pc);

    pc = normalise_position(pc);
end
```

Published with MATLAB® R2019a