Designing and Prototyping a Low-Cost and Automated Canopying Imaging System with Chlorophyll Fluorescence Imaging

by

Tony Pham

(Under the direction of Mark Haidekker)

Abstract

Canopy imaging is a noninvasive form of crop monitoring used to measure the growth of plants in horticultural studies. This imaging technique utilizes cameras and image processing to capture plant canopies and separate them from their background. Most canopy imaging systems use RGB cameras and separate the plants based on green pigments. This technique works well most of the time, but separation of plants becomes difficult for species that have non-green leaves. This work describes the design and development of a canopy imaging device that utilizes chlorophyll fluorescence: light emitted by plant chlorophyll as a response to receiving light energy. Chlorophyll fluorescence is easy to image and occurs so long as plants contain chlorophyll, independent of coloration. In order to develop a device that improves upon current technologies, emphasis was placed on adding useful features to it, including on-device image processing and pipeline automation, while also reducing total cost.

Index words: Chlorophyll fluorescence, Canopy imaging, Raspberry Pi, ImageJ, CIFS, Automation, Cost reduction, Device prototyping, Horticulture

Designing and Prototyping a Low-Cost and Automated Canopying Imaging System with Chlorophyll Fluorescence Imaging

by

Tony Pham

B.S., The University of Georgia, 2018

A Thesis Submitted to the Graduate Faculty
of The University of Georgia in Partial Fulfillment
of the
Requirements for the Degree

Master of Science

Athens, Georgia

2021

Designing and Prototyping a Low-Cost and Automated Canopying Imaging System with Chlorophyll Fluorescence Imaging

by

Tony Pham

Major Professor: Mark Haidekker

Committee: Marc van Iersel
Kyle Johnsen
Rodney Averett

Electronic Version Approved:

Ron Walcott
Dean of the Graduate School
The University of Georgia
August 2021

Acknowledgments

First of all, I will gratefully acknowledge the financial support provided for my research from the US Department of Agriculture through the USDA-NIFA-SCRI-006529 grant.

I would next like to acknowledge my major professor, Dr. Mark Haidekker, for supporting me with both my research as well as my academic goals. Had he not reached out to me about performing research under him, chances are that I would not have chosen to advance my education. Because of Dr. Haidekker, many educational opportunities opened for me such as learning various programming languages and experiencing the basics of bioinformatics. For this, I am grateful and give Dr. Haidekker my utmost thanks.

Finally, I would like to acknowledge both my family and close friends who have provided great support throughout the duration of my program. They have helped me to relieve plenty of stress as well as provided guidance to me in non-academic areas. It is tough to say just how much these people have contributed to my success, but thank you very much.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis presents an exposition on the research and development of a canopy imaging device which uses chlorophyll fluorescence, made for the monitoring of plant growth and health. Because typical RGB canopy imaging experiences issues with non-uniform color distribution of some plants, the approach of imaging chlorophyll fluorescence, light emitted by all plants during photosynthesis, was applied to the core of this device's development. In order to make the canopy imaging device more accessible and practical for scaling up within plant growth systems, a goal to minimize costs for production of this device was also added since general equipment costs for chlorophyll fluorescence imaging remains expensive, ranging in the hundreds of dollars per camera and thousands of dollars per system. Finally, an objective of maximizing flexibility for system settings was included to increase the practicality of using this device further, especially because many horticultural growth systems differ in environmental conditions and so require different parametric changes to the imaging device. In fulfilling all the mentioned goals and objectives, a canopy imaging device was created that, while requiring more time investment to use and being less accurate than its expensive counterparts, is able to monitor the growth and health of all fluorescing plants regardless of coloration as well as allow for easy customization and scaling up of growth systems at a low cost.

Chapter 2

Background

2.1 Chlorophyll Fluorescence

Chlorophyll fluorescence imaging is a more recently implemented technique used to monitor the growth and health of plants in agriculture [6, 14]. It has become increasingly common in plant physiology studies as it provides a noninvasive and valuable method of measuring the efficiency of plant chlorophyll and, in turn, plant health and growth.

As light energy is absorbed by plant chlorophyll, it has three possible pathways. The light energy may either drive photosynthesis in the plant, be dissipated as heat, or be re-emitted as light [9]. This re-emission of light is what we call chlorophyll fluorescence. Because these pathways work in competition, chlorophyll fluorescence can be used to determine the efficiency of photosynthesis which relates to the plant's health and growth. This fluorescence is emitted at a wavelength starting from approximately 660 nm (Fig. 2.1) which is in the far-red region of the light spectrum and can be easily measured if the surrounding light sources emit a negligible amount of far red or near-infrared (NIR) light, allowing for clear separation. For this purpose, specific LEDs tend to be chosen as primary excitation sources for chlorophyll fluorescence studies.

A phenomenon called fluorescence quenching is one major drawback for the quantitative measurement of chlorophyll fluorescence (yet not so much for more qualitative measurements). When chlorophyll is initially exposed to light, its fluorescence response will spike and peak for about one second. Afterward, the response decreases significantly over the course of several minutes (Fig. 2.2) [13]. This phenomenon poses a problem because it is caused by two main physiological processes, 'photochemical' and 'non-photochemical' quenching, whose

contributions must be differentiated from one another to get usable quantitative information from chlorophyll fluorescence measurements [16].

One practical method of circumventing this problem was termed the 'light doubling' technique [2, 18], which approximately removes the photochemical quenching contribution. In this technique, the light source used to excite the chlorophyll and initiate the fluorescence response would be pulsed at a high intensity for a brief duration. If the excitation pulse is short enough, then no non-photochemical quenching nor changes to photosynthesis would occur either. This allows for relative measurements of chlorophyll fluorescence to be obtained, which can be used to calculate quantitative properties of photosystem II (PSII) [16], a protein complex in the chloroplast responsible for catalyzing the initial steps of photosynthesis. Fortunately, since the premise of canopy imaging is only concerned with the ability to distinguish leaf areas from non-leaf areas, quantitative measurements of PSII are not needed, and fluorescence quenching would not be an issue for the purposes of this thesis. Nonetheless, this is useful information to keep in mind should implementation of quantum yield measurements become desired in the future.

The typical instruments for measuring chlorophyll fluorescence are fluorometers which can detect the intensity of emitted fluorescence from plants or other substances. Some fluorometers are modulated, a method of rapidly pulsing the measurement light, which allows them to be used even with background lighting [18]. The major downside to using fluorometers for measuring chlorophyll fluorescence is that neither plant size/area nor fluorescence mapping can be imaged which both provide useful information on the plants. For example, fluorescence images of leaves have shown that photosynthetic performance can be highly heterogeneous across the leaves. As such, cameras have become more commonly used in taking chlorophyll fluorescence images [1].

Because the aim of my chlorophyll fluorescence imaging device is to analyze canopy size, using a camera instead of a fluorometer is a necessity. As mentioned before, canopy imaging only requires qualitative measurements of chlorophyll fluorescence, so the 'light doubling'

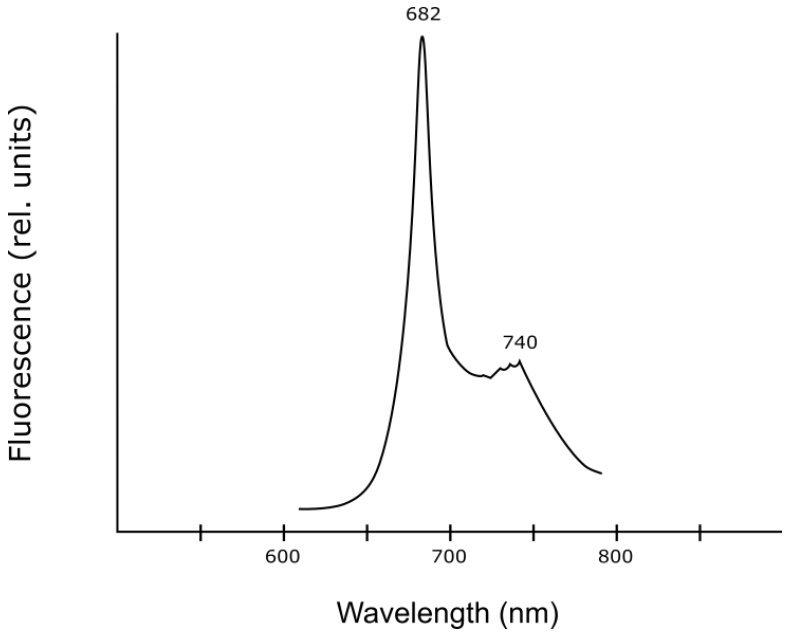

Figure 2.1: The approximate chlorophyll fluorescence spectrum of plants. It peaks strongly at 682 nm and once again at 740 nm. Adapted from Krause [15].

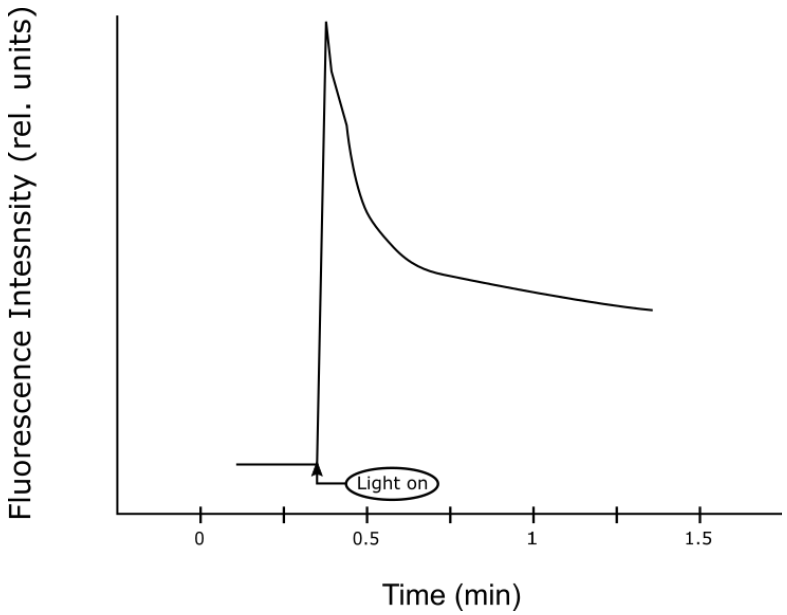

Figure 2.2: An example of fluorescence quenching. The fluorescence response of chlorophyll to light exposure from a dark environment is seen here. Adapted from Krause [15].

technique will not be implemented in this version of the imaging device. However, future improvements could certainly focus on making use of the technique to provide more options for measuring chlorophyll fluorescence. The method for capturing chlorophyll fluorescence with a camera involves using an excitation light to trigger the plant's fluorescence response. A blue LED is usually used as the excitation light because they have a near optimum excitation wavelength as well as a smaller far-red tail than white LEDs. Then, an image is taken of the plant through an IR cut-on filter so that most unwanted visible light, specifically the strong blue light, would not be picked up by the camera and only the chlorophyll fluorescence is imaged.

2.2 Canopy Imaging

Canopy imaging is a form of noninvasive crop monitoring that uses a camera to take images of the crops from directly above. Currently, the most common form of canopy imaging uses RGB setups which capture all visible light. For instance, Bumgarner [3] performed a study where RGB canopy imaging was used to supplement other more direct forms of crop measurement on lettuce. In this study, the images taken of the lettuce were analyzed using a software called WinCAM 2009 Regular. The process of analyzing images through WinCam 2009 consisted of loading batches of the raw images, having an operator manually select a set of colors from within the images to designate as background or leaf coverage, and running the image analysis program through the batch of images, resulting in pixel counts for the leaf and background of each image. A percentage would be calculated from the pixel measurements to find the amount of canopy cover compared to the amount of background in each image. This approach worked well for fully green lettuce plants, but analysis was much less successful for species such as 'Flagship' leaf lettuce which has primarily red leaves, where plants were unable to be separated from the background.

Other image analysis software used for canopy imaging include ImageJ from NIH, which can perform threshold-based pixel count measurements but normally requires more user

input for complex images, and specialty-made software such as Easy Leaf Area, developed by Easlon and Bloom [7]. Easy Leaf Area performs similarly to WinCAM 2009 Regular but is built off Python with the OpenCV library and also allows for rapid processing of image batches and customization by users. The program utilizes a red calibration patch of a known fixed area to calculate leaf areas for images during analysis. This calibration patch requires that the operator place the patch in a similar area and height relative to the plants, in order to avoid problems from image distance and lens distortion. Similar to WinCam 2009, a batch of images is loaded into Easy Leaf Area, 8-bit values [0-255] are selected as minimal green and red thresholds, and the program is run over the image batch, resulting in leaf area measurements. Using the minimal 8-bit values, Easy Leaf Area is able to distinguish the background, the leaf coverage, and the red calibration patch. Unfortunately, Easlon does note that while the program is able to accept more selection criteria for non-green plants, adjusting the software for such purposes would require greater Python knowledge. Also, issues could emerge in calibration due to adjusting settings for red-pigmented leaves. Otherwise, analysis of mixed colored plants was not mentioned.

Due to the shortcomings of RGB image analysis for canopy imaging, other forms of plant monitoring are being incorporated into canopy imaging and developed, hence the purpose of this paper. Previously mentioned, light energy that is absorbed by plant chlorophyll can also be dissipated as heat, in competition to photosynthesis and chlorophyll fluorescence. Using the heat given off by light exposed plants, Osroosh [17] developed a low-cost mixed thermal-RGB imager, similar in principle to the canopy imaging system developed in this thesis. The system that Osroosh made included a micro-controller (the Raspberry Pi), a RGB camera board, and a thermal imager among other components. It allowed for measurement of the average temperature of leaves as well as canopy size, and the total costs ranged between approximately three and four hundred dollars. Osroosh combined RGB and thermal imaging by using the captured thermal image as a mask for the RGB images, separating the plants from the background based on the resulting combination. While this approach was successful

for larger canopy sizes, there were significant measurements errors with small canopies due to imperfect alignment of the images [17]. Fortunately, the size of canopies would not be a specific issue for chlorophyll fluorescence imaging as there is no need for the alignment of multiple image sources.

In explaining the concept of canopy imaging, it is important to discuss the relationship between the pixels of captured images and the projected area, or the actual size of the plants that are imaged. Specifically, the geometric relationship can be described as ray geometry (Fig. 2.3). Due to factors including the distance between the camera and the plants, the camera's field of view, any barrel distortion, and the angle of the camera's axis, the conversion from pixels to units of area is specific to each imaging setup. Therefore, calibration should be performed in order to obtain true area measurements. As seen with Easy Leaf Area, calibration can be done by imaging a specific object, such as a ruler or colored patch, with known measurements to calculate the conversion. This method becomes more difficult at the edges of images if there is significant barrel distortion of the camera or if the camera has a wide-angled lens. This places emphasis on using cameras with negligible barrel distortion and a narrower field of view.

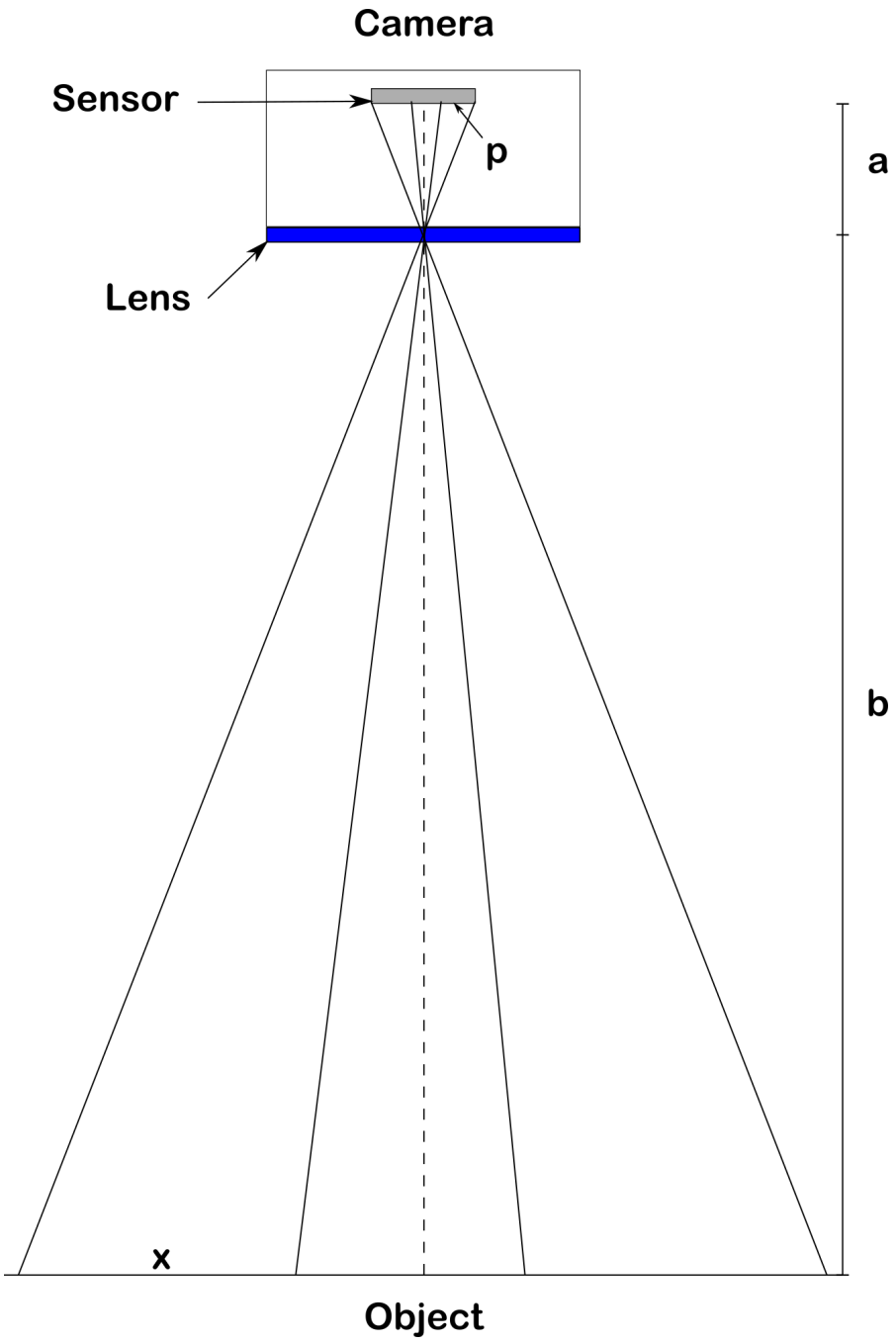

Figure 2.3: Ray geometry is the geometric relationship that best describes how pixels of images taken by a camera can be related to actual area measurements. Here, a is the distance between the camera sensor and the lens while b is the distance between the lens and the imaged object. Assuming that $a \ll b$, $x = \frac{b}{a} * p$, where x is a true area measurement and p is the pixel size. Additionally, factors such as lens distortion and field of view will alter the relationship between x and p as pixels move further from the center of the sensor.

Chapter 3

Problem Analysis and Objectives

3.1 Shortcomings with RGB Canopy Imaging

When performing canopy imaging, an image is taken by a camera and processed through a program, such as the aforementioned WinCam 2009 and Easy Leaf Area, to separate the plants from any background. RGB cameras are currently the most commonly used device for this application and work well for most plants since the majority of them are completely green, which is easily distinguished from dark backgrounds. However, as noted before, when imaging plants with non-uniform color distribution, the image processing encounters problems with separation of the plants. This is because colors such as dark red, which naturally occurs on a variety of plants and crops such as lettuce, is difficult to distinguish from dirt and other background substrate. Some recently developed programs have been able to process these images more reliably, however as RGB images can vary greatly between setups and plants, using these programs correctly would require more experience with them as well as greater user input. For example, the program Easy Leaf Area stated that analysis of non-green plants could be performed but would require some alteration of the program's Python code rather than simple adjustments to the minimal green and red thresholds.

3.2 Fluorescence Imaging Costs

As mentioned previously, the two main devices for imaging chlorophyll fluorescence are fluorometers and cameras. While fluorometers are readily available for commercial purchase and use, the cheaper point fluorometer variants that run in the range of hundreds of dollars

only allow for single wavelength analysis while the more useful bench top fluorometers that provide spectra analysis cost upwards of tens of thousands of dollars, keeping in mind that fluorometers do not capture whole plants or leaves.

On the other hand, chlorophyll fluorescence imaging systems that use cameras are scarcely available commercially, and the purchasable systems are specially made and cost much more than most fluorometers. This leads scientific researchers to build their own systems from components bought separately, whose total costs typically range in the upper hundreds to thousands of dollars. The majority of these costs come from the camera, filter, and lens. For example, the FLIR brand of cameras is commonly deployed in custom setups involving fluorescence imaging as well as thermal imaging [8, 11, 12], which cost approximately three hundred dollars each. Combined with a suitable lens and filter, the costs get bumped up another several hundred dollars.

3.3 Inconveniences of Scaling Up

Most chlorophyll fluorescence imaging systems, both custom-made and commercially bought, are large, stationary setups that require the users to place their plants inside the systems. This method works fine for most scientific studies as plants are normally grown in trays of pots that can be moved readily. Even so, the need to continuously move plants in order to take canopy images becomes problematic as studies and processes are scaled up, for example, from five trays of plants to dozens. Furthermore, in situations where crops are planted in the field and cannot be moved, these setups are entirely impossible to use.

3.4 Overall Design Goals

Based on the problems that have been presented, there were three main goals for the development of this imaging system: 1) make plant separation easier using chlorophyll fluorescence, 2) reduce the total costs to a commercially affordable price, and 3) design the system to

be mobile and easy to deploy. Additionally, automation of the canopy imaging process was considered a bonus objective for better user experience.

In order to avoid the problem with plant separation that RGB imaging encounters, the system was designed based off of using chlorophyll fluorescence which can be imaged and processed the same way regardless of plant color. This meant that the components needed for this imaging system included a preferably monochrome camera, a filter, and an excitation light. I will discuss more about why the camera needs to be monochrome later.

A design approach similar to that of Osroosh [17] was implemented to reduce costs as well as allow for the system to have versatility with its placement. To clarify, the device would be designed with the capability to be mounted and used in pre-existing growth systems, without the need to build any independent enclosures. The use of a micro-controller such as the Raspberry Pi allowed for several key benefits. Those are that the system could operate while being mobile so long as it is powered, with the potential of using batteries instead of wired power supplies, as well as that the micro-controller was able to support a variety of suitable cameras at a low cost. As far as ease of use goes, that came down to how easily the system could be made to set up, which was quite simple it turned out through the use of backing up OS images/files and readily downloadable scripts.

Chapter 4

Materials

4.1 The Raspberry Pi

The Raspberry Pi is a relatively cheap brand of micro-computer, commonly used in custom-made electronic devices. The micro-computer provides plenty of uses that made development of this canopy imaging system much easier. On its own, this machine is essentially a small computer, similar in performance to a 5-year-old vintage laptop, able to install and run programs such as ImageJ among many others. Additionally, the Raspberry Pi provides electrical signal pins and native camera support, allowing for easy image capture and triggering of excitation light for the plants.

In choosing which model of Raspberry Pi would be used in the imaging device, some key features were prioritized. Of course, native camera support was necessary, but in order to make the device as mobile and easy to deploy as possible, the ability to connect to wireless networks as well as compatibility with more recent cameras were also used as determining factors. With regard to on-board memory, the imaging system did not require much, so I decided on 2 GB as the difference in cost for upgrade from 1 GB to 2 GB was negligible. 2 GB provides more than enough memory to run the operating system, ImageJ, and any image processing. With ImageJ running, the Raspberry Pi showed a memory usage of under 300 MB, with approximately an additional two bytes of RAM per byte of image size opened. Tacking on additional programs such as SAMBA or Xvfb, which will both be covered later, 2 GB was plenty of memory, and any further upgrade was unnecessary. Ultimately, the newer Raspberry Pi 4 Model B was chosen as it should be compatible with all cameras adapted

for the Raspberry Pi as well as able to connect to wireless networks without any additional hardware.

4.2 Cameras

When choosing an appropriate camera for the canopy imaging system, several key features were tested to decide which one was the most fitting. These key features were: 1) sensitivity to IR spectra, 2) monochrome image capture, 3) a global shutter, 4) high image resolution, and finally, 5) low cost.

The first condition, IR sensitivity, came from the fact that chlorophyll fluorescence is emitted in the far-red to IR part of the light spectrum. If a camera lacked IR sensitivity, the images it captured would show little chlorophyll fluorescence, and the plants would be much more difficult to distinguish from its background. Unfortunately, most commercial cameras, including those designed to be compatible with the Raspberry Pi, contained a built-in IR filter. This is because most cameras utilize silicon sensors which are sensitive to wavelengths far past the visible light spectrum (Fig. 4.1). These wavelengths are undesirable for most forms of imaging, and IR filters are the best solution to removing them. Removal of these filters is possible, but the process would more than likely damage the camera, especially for smaller ones, and make them unusable.

As mentioned before, a monochrome camera is highly preferable when imaging chlorophyll fluorescence, although not entirely required. This is because of a drop in resolution and performance with RGB cameras due to the use of Bayer filters. A Bayer filter is placed in front of a camera's sensor and is a grid pattern of red, green, and blue pixels (Fig. 4.2). Bayer filters provide RGB cameras with the information needed to generate colored images, but in order to get their final images, the cameras must interpolate the grid-based information which causes these cameras to perform more slowly and often results in poorer image quality. Since chlorophyll fluorescence imaging is only concerned with the intensity of captured light,

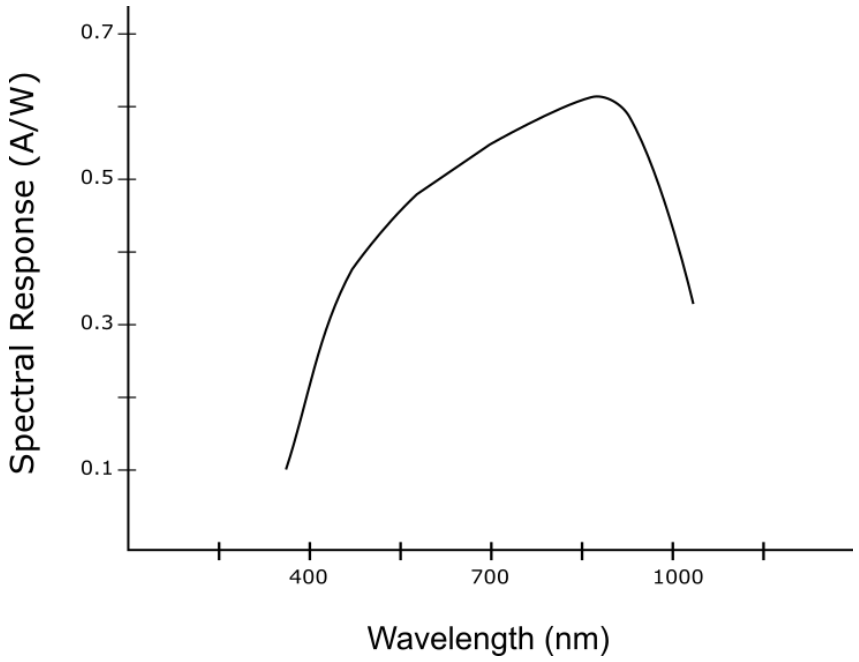

Figure 4.1: The spectral response of silicon approximately ranges from 350 nm to 1100 nm which extends well into the IR spectrum. Wavelengths past the visible light spectrum tend to be undesirable as images may capture reflective light that cannot be seen by the human eye. IR filters can be used to account for these wavelengths. Adapted from Chander [4].

being chlorophyll fluorescence, monochrome cameras can be used for better performance and resolution.

In quantifying how high of a resolution would be suitable for canopy imaging, reference must be made back to the discussion on the relationship between pixels and projected area of plant canopies (Section 2.2), as a higher resolution would provide more pixels per image which allows for more precise measurements of area to be calculated. An exact pixel count requirement may be difficult to quantify, but the idea is that a greater resolution would result in higher precision of the calculated projected area. Otherwise, cameras normally come with either rolling shutters or global shutters. Rolling shutters capture images by scanning across the pixels of their sensors in a wave-like pattern which means parts of each image are exposed at different times. Meanwhile, global shutters expose their pixels

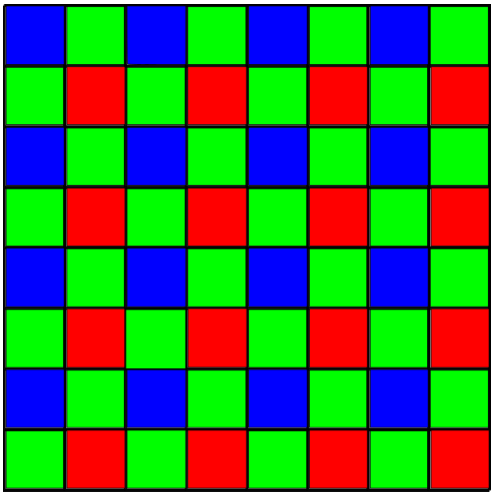

Figure 4.2: An example of the color grid pattern of a Bayer filter. Due to the distribution of red, green, and blue pixels, only half of a RGB camera's resolution can be used for interpolation of green information while only 1/4 of the resolution can be used for blue and red.

simultaneously, so images will be completely synced. Because of this, global shutters are preferred for chlorophyll fluorescence imaging due to the use of pulsed-LED exposure which would cause rolling shutters to potentially capture unusable images. While canopy imaging does not pulse LED lights to capture chlorophyll fluorescence, it does not hurt to include a camera with a global shutter, especially this device may be used to obtain quantitative data in the future.

Since the decision was made to base this imaging system off of the Raspberry Pi, possible camera choices were already narrowed down to compatible ones which typically cost under $50 USD; a detailed overview of costs can be found in Section 4.5. Additionally, Raspberry Pi compatible cameras readily come with relatively high resolutions, typically 1280x720, as well as a global shutter. This allowed me to focus camera testing on the remaining two criteria, IR sensitivity and monochrome capture.

In order to test the selection of possible cameras, a testing setup was built from ThorLabs optical rails, a large aluminum heat-sink plate, strong LED modules (100A series, Satistronix,

Shenzen, China) for excitation, custom relays and computer case fans. A custom power supply to provide 36V, 12V, and 5V outputs was also made by modifying an existing 36V power supply and connecting a custom-made buck-mode switching converter. To test the cameras, each one was mounted onto the aluminum plate with a pothos plant placed underneath. The custom power supply was turned on to power both the fans and LED relays. Finally, a simple script was run on the Raspberry Pi to trigger image capture of the testing camera and to turn on the excitation lights during said image capture.

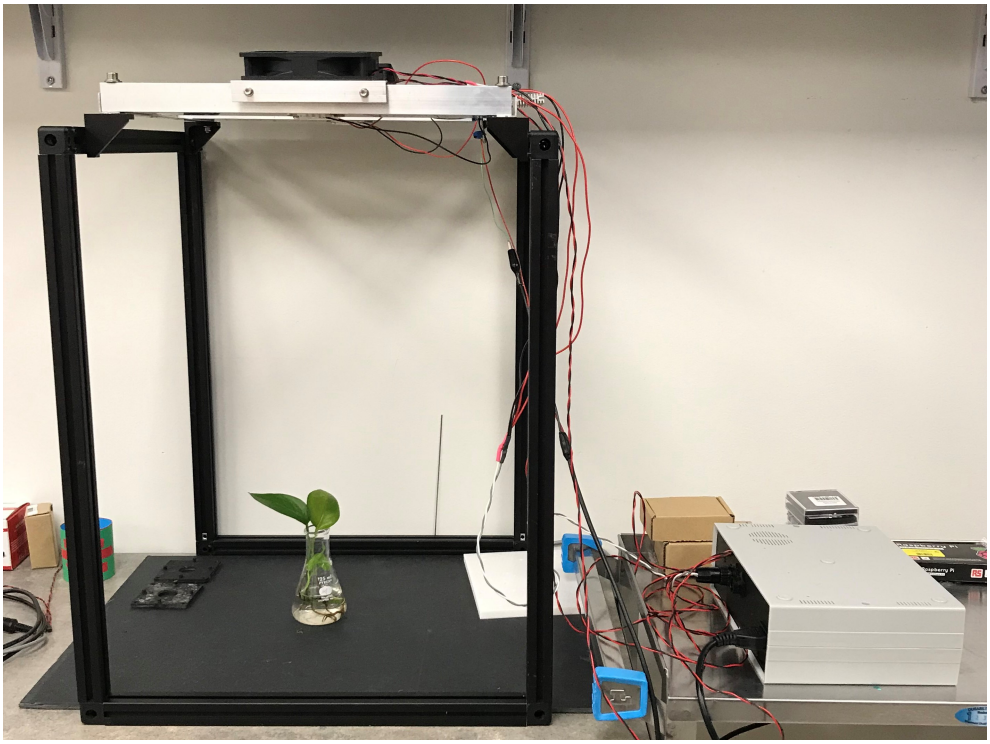

Figure 4.3: The testing setup used for all the cameras and filters. A black tarp would be used to cover the setup during testing to block out any ambient light.

Based on the criteria set earlier, two main candidates were found, Raspberry Pi's NoIR Camera v2 and ArduCam's MIPI Camera series. The first three cameras to be bought and tested were Raspberry Pi's NoIR Camera v2, ArduCam's UC-580 Rev.B, and ArduCam's UC-545 Rev.C. The ArduCam UC-580 Rev.B was pruchased accidentally despite having a built-in IR filter, but its test images were a good example of what happens to chlorophyll

fluorescence images when the IR light region is cut. The test images from each of the cameras can be seen in Figure 4.4. In the sample test images, the Pi NoIR Camera v2's image appeared red because it was not monochrome and chlorophyll fluorescence only registers in the red channel. Notably, the Pi NoIR Camera v2 was the only native Raspberry Pi camera found to have no built-in IR filter. The fact that it was a native camera allowed for easier use since no additional files or programs needed to be installed for it, so this camera mainly served as a backup plan in the case that a suitable third-party monochrome camera could not be found. Most notably, the ArduCam UC-580 Rev.B produced a grainy, lower contrast image because of its IR filter which blocked a large portion of any chlorophyll fluorescence. The ArduCam UC-545 Rev.C did fit all the criteria and captured decent images of chlorophyll fluorescence. Its main downside, however, was its low resolution, maxing out at 640x480.

After testing the first three cameras, two more ArduCam series cameras were bought, the UC-599 and UC-599 Rev.B. The only difference between these two cameras were the lenses installed on them. The UC-599 had a 130-degree lens while the UC-599 Rev.B had a low-distortion 70-degree lens. Both cameras were monochrome and IR sensitive with a maximum resolution of 1280x800. Based on their test images (Fig. 4.5), both cameras provided images with similar resolutions, but the UC-599 Rev.B's image was more zoomed in due to its narrower lens. Additionally, the UC-599 Rev.B showed more of the plant's details because of this narrow focus. In the end, the choice of camera was between these two new ArduCam models. Because it had a low distortion lens, the UC-599 Rev.B was chosen as the most suitable candidate for the imaging device.

4.3 Filters

Chlorophyll fluorescence is emitted between approximately 660 nm and 780 nm, with the majority of it occurring in the lower half. During camera testing, a large custom-made band-pass filter from Omega Optical, Inc was used with a transmission range of 660 nm to 740 nm.

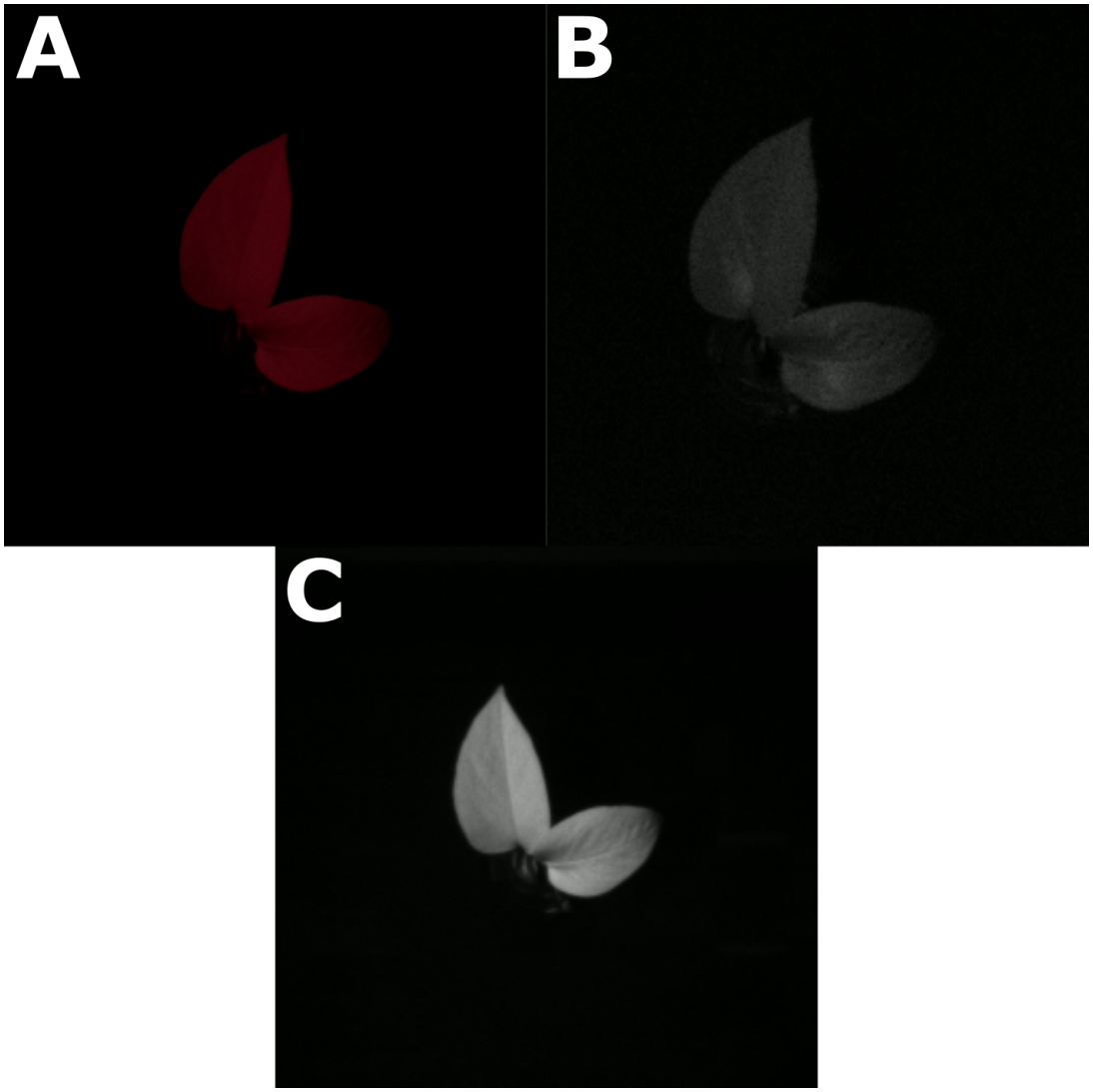

Figure 4.4: Sample images from the initial set of camera tests for the: A) Pi NoIR Camera v2, B) ArduCam UC-580 Rev.B, and C) ArduCam UC-545 Rev.C. All of these images were taken under the same lighting conditions.

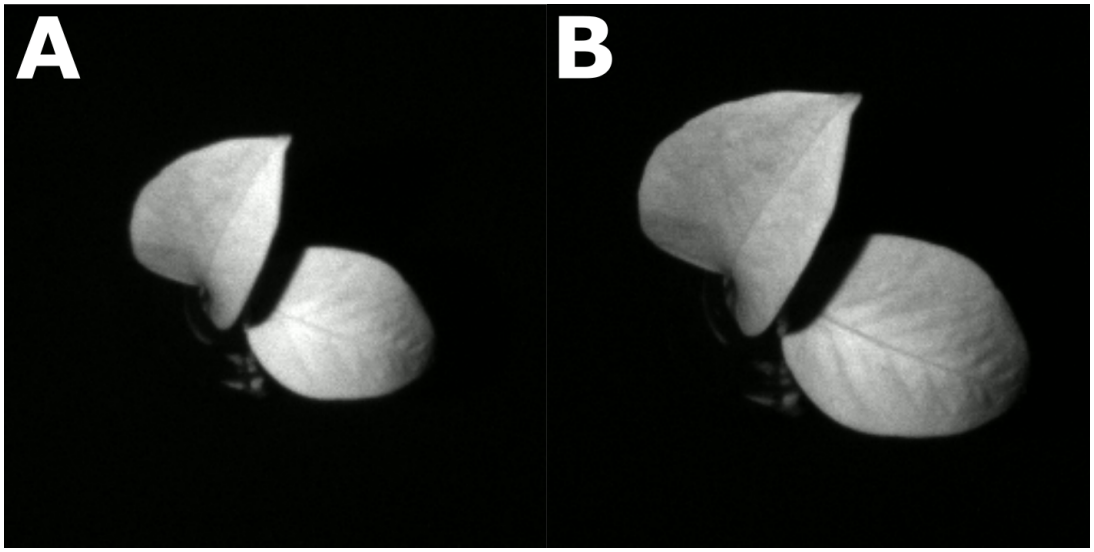

Figure 4.5: Sample images from the second set of camera tests for the: A) ArduCam UC-599 and B) ArduCam UC-599 Rev.B. Images were taken under the same lighting conditions between these two cameras and the initial batch of tested cameras.

This filter worked perfectly for testing, but its large size and high cost made it impossible to use for the imaging device.

With regard to size, a filter with a diameter of at least 25 mm was needed to fully cover the lens of the UC-599 Rev.B. Larger sizes would work just as well for covering the camera, but making the device as compact as possible would allow it to be more mobile and easier to use. Additionally, filters become much more costly the larger they are. At first, smaller band-pass filters with similar transmission ranges as the large, custom filter were investigated. However, band-pass filters are costly even with a small diameter such as 25 mm, costing as much as two hundred dollars each, so I opted to use long-pass filters instead. Long-pass filters are suitable for chlorophyll fluorescence imaging because there are no significant emissions at the longer wavelengths, past the far-red and NIR regions that chlorophyll fluorescence occupies.

Four long-pass filters were obtained from Edmund Optics as well as third-party sellers on eBay. The two long-pass filters obtained from Edmund Optics were an optical cast plastic

filter and a colored glass filter. The plastic filter was extremely cheap and had a cut-on wavelength of about 670 nm. Its only downside was having roughly 5 percent transmission in the 500 nm range, which is green light in the visible light spectrum. This could pose problems for images taken in the presence of background light, but since canopy imaging of chlorophyll fluorescence is performed in the dark with a sole excitation light that is normally blue, this plastic filter would be fine for the device's needs.

On the other hand, the glass long-pass filter from Edmund Optics had a cut-on wavelength of 685 nm and costed roughly four times as much as the plastic filter, which was still relatively cheap compared to any previously mentioned band-pass filters. This glass filter had no other downsides beyond the higher cost, so it would be suitable for use with some ambient light. Both of these Edmund Optics filters were tested in the same camera testing setup over a range of gain and exposure settings using the UC-599 Rev.B camera.

As for the long-pass filters from eBay, there were two cheap gelatin filters, the Wratten 70 and Wratten 89B. These gel filters could not be found from typical online vendors, so third-party sellers from eBay were the only option. The main benefits to these Wratten gelatin filters were that they can be cut to fit and that they were about as cheap as the Edmund Optics plastic filter. The Wratten 70 was a long-pass filter meant for infrared photography, having a cut-on wavelength of 650 nm, while the Wratten 89B had a cut-on wavelength of 690 nm for very deep red photography. Both gelatin filters were tested in the same way as the Edmund Optics filters, over the same range of settings and with the UC-599 Rev.B camera.

In selecting these long-pass filters, two main aspects were focused on: the cut-on wavelength and the cost. For the purpose of chlorophyll fluorescence imaging, a long-pass filter would be suitable so long as its cut-on wavelength is no less than about 650 nm. Lower cut-on wavelengths would allow greater bleed-through of unwanted light, making it more difficult to separate plants from their background. Meanwhile, the cost of the filter contributed to the total costs of the imaging device, so a lower cost was preferred.

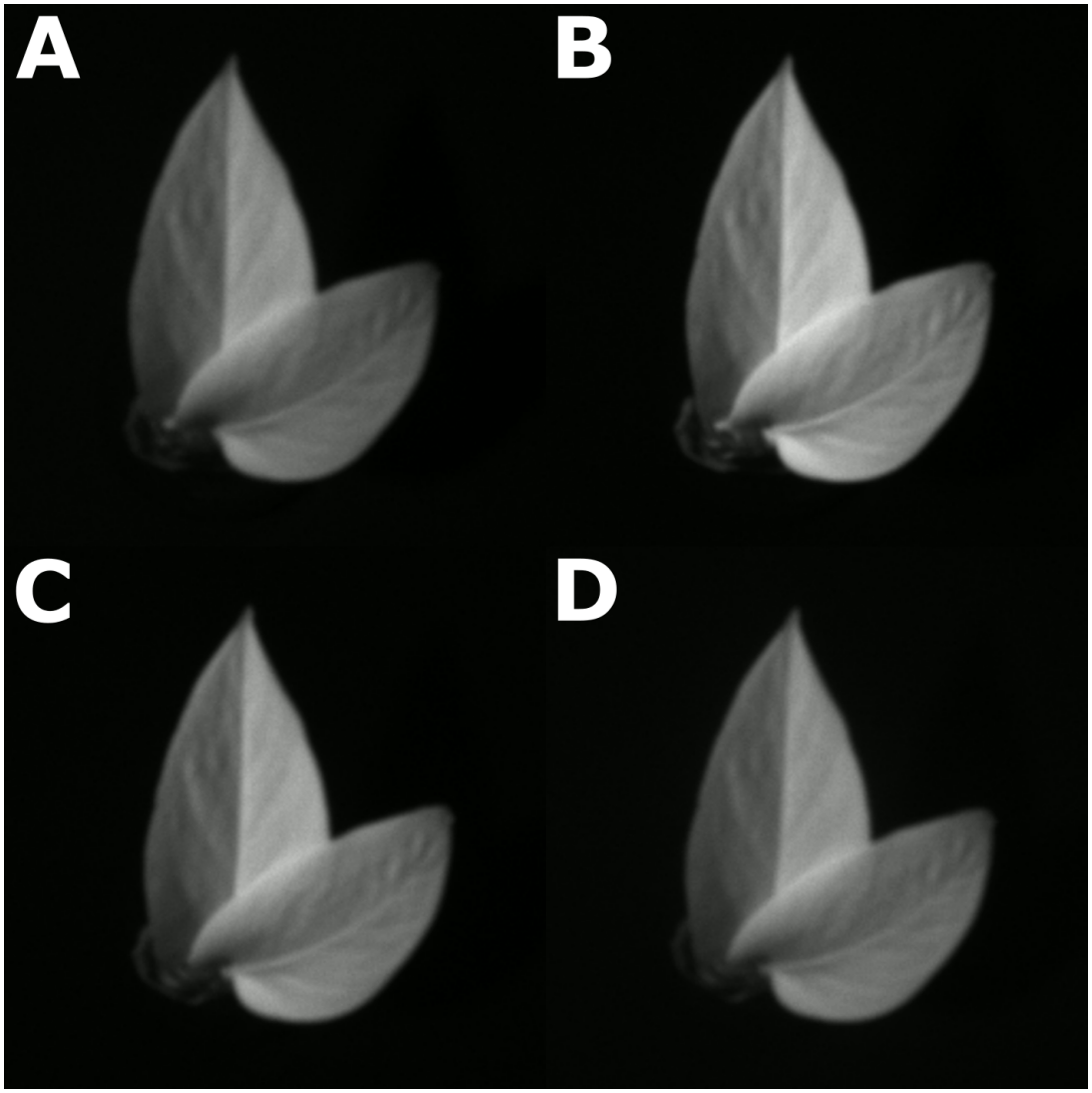

Figure 4.6: Sample images from the filter tests for the: A) Edmund Optics Cast Plastic Filter, B) Edmund Optics Colored Glass Filter, C) Wratten 70 Gelatin Filter, and D) Wratten 89B Gelatin Filter. All images were taken using the ArduCam UC-599 Rev.B camera. Slight differences in gain and exposure settings were used in these sample images as each filter had a different transmission of chlorophyll fluorescence. Filters with high transmittance showed better detail at lower gain and exposure settings while those with lower transmittance performed better with more gain and exposure.

Looking at the test images for each of these four filters (Fig. 4.6), they all performed well and relatively equally. The Edmund Optics glass filter produced images with noticeably less blur, but that came at a higher price. Taking into account both cost and availability, I ended up choosing the Edmund Optics optical cast plastic filter for this canopy imaging device. The cost of the glass filter was too much for the little improvement it provided while the gelatin filters could not be reliably sourced. However, as mentioned before, the plastic filter could have some problems with background light, making it unsuitable for more quantitative uses of chlorophyll fluorescence. If this device were to be upgraded in the future to include more of those features, switching to the Edmund Optics glass filter would be a reasonable upgrade.

4.4 3D-printed Parts

The final piece of hardware necessary for the canopy imaging device was the device's enclosure. Due to having multiple electronic parts and for convenience, the enclosure was 3D-printed from custom-made models. In designing the device's enclosure, emphasis was placed on being able to mount each of the electronic parts and completely blocking unwanted light from entering the camera, especially through seams or cracks, with the exception of through the long-pass filter. To prevent any internally reflected light from entering the camera lens, a matte chalkboard paint was sprayed onto the inner walls of the device that surround the camera. Additionally, a mounting plate was necessary to allow for mounting the device wherever needed. In total, the imaging device contains a Raspberry Pi, a separate camera board, and a filter. Connected to the Raspberry Pi were a power cable and two wires attached to digital output pins as well as the camera's ribbon cable. The wires were used to trigger a relay for turning on excitation lights.

After several iterations of 3D-printing and making small adjustments, I ended up with four separate parts: a mounting plate, a Raspberry Pi enclosure and camera mount, a camera cover, and a filter holder. The part that served as both a Raspberry Pi enclosure and a camera

mount was the most complex in shape and was the most difficult to print, messing up during printing most often due to being the tallest part and having issues with the printer itself, either the unwinding of filament or improper attachment to the print bed. Once the issues with the printer were addressed, printing the parts became easy and reliable. Overall, the majority of changes done to the design of the parts were to fix hole sizes for screws and to make the parts as compact as possible. In order to mount the entire device, chains or wire may be threaded through the large holes on the mounting plate and tied to an overhang. A diagram of how all the parts were assembled for the imaging device can be seen in Figure 4.7, and images of the fully assembled device are provided by Figure 4.8. Otherwise, all the files for the parts can be found in the GitHub link located in Section 4.5.

4.5 Costs and Sourcing

Tables 4.1, 4.2, and 4.3 contain lists of all the components that were tested for the canopy imaging device along with a complete cost breakdown of the device. The listed costs do not take into account any taxes or other fees associated with purchase nor the cost of lighting components because the imaging device made use of excitation light provided at the imaging site. Nonetheless, lighting components including LEDs and relays could be roughly estimated to cost under $50 USD. For the 3D-printed parts that cannot be commercially purchased, STL files can be found on the public GitHub page for this project (https://github.com/tmp52468/Haidekker-Lab/). The STL files were made specifically for the Artillery Sidewinder X1 3D printer, so adjustments may be needed for other printers. Additionally, a complete parts list with URL links for purchase as well as further instructions can also be found on the GitHub link.

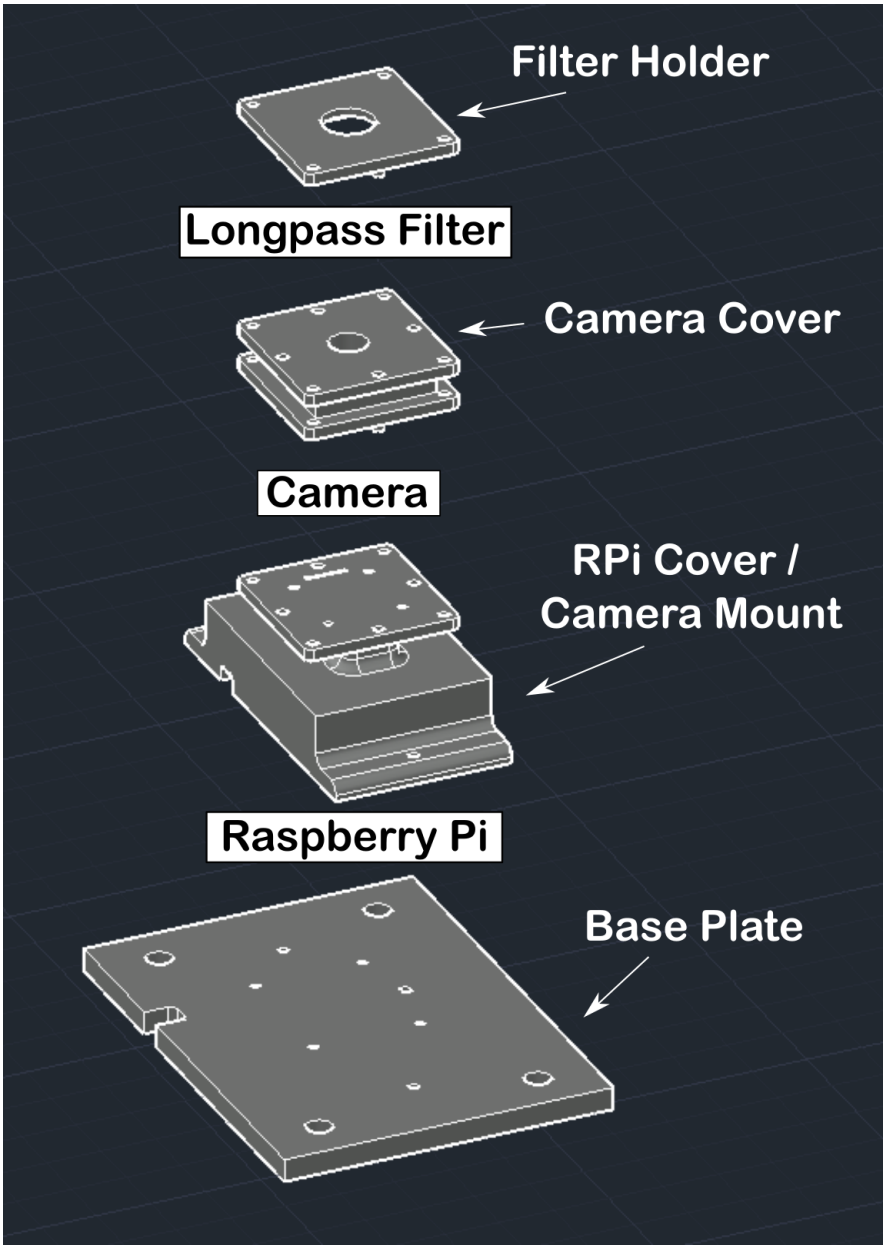

Figure 4.7: An assembly diagram of the imaging device including positions for its 3D-printed parts and the electronics, being the Raspberry Pi, camera, and filter.

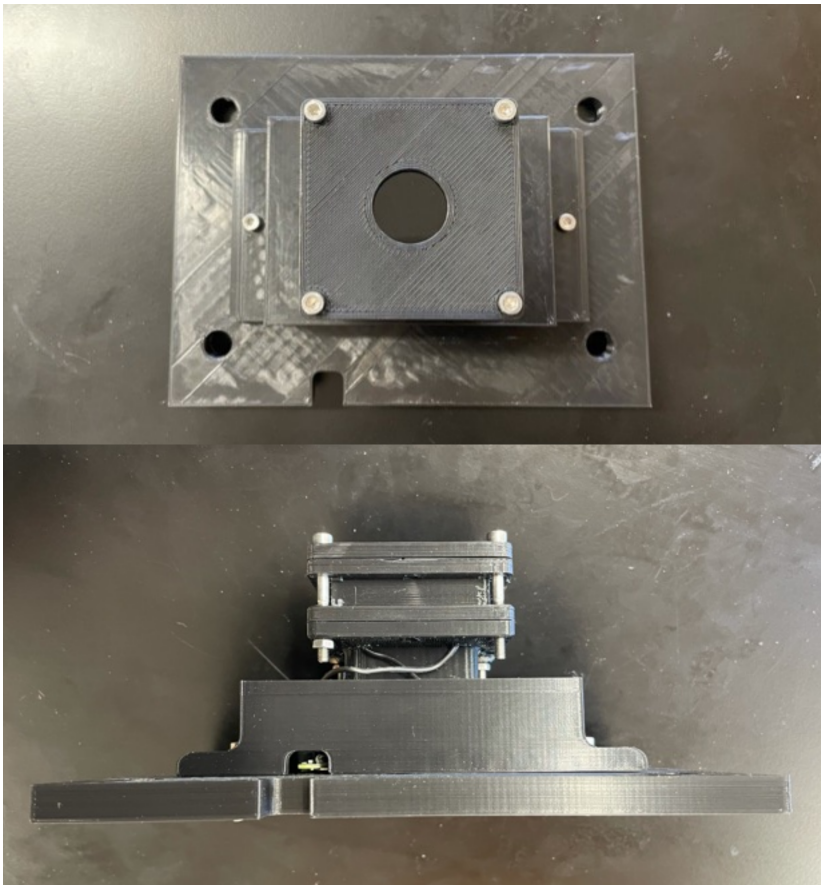

Figure 4.8: Images of the imaging device after being fully printed and assembled with its electronics.

Table 4.1: The costs and other details for each potential camera that was tested.

Camera	Vendor	Cost (USD)	Notes
ArduCam UC-580 Rev.B	Uctronics	29.99	Not IR Sensitive
ArduCam UC-545 Rev.C	Uctronics	25.99	Low Resolution
Pi NoIR Camera V2	Adafruit	29.95	Not Monochrome
ArduCam UC-599	Uctronics	41.99	Wide-Angle Lens
ArduCam UC-599 Rev.B	Uctronics	41.99	Low Distortion Lens

Table 4.2: The costs and notes for each of the tested filters.

Filter	Cost (USD)	Notes
Omega Optical Band-pass Filter	220.00	Very Expensive
Edmund Optics Plastic Long-pass Filter	12.50	Transmits Green Light
Edmund Optics Glass Long-pass Filter	49.00	Slightly Expensive
Wratten 70 Gelatin Filter Sheet	92.00	Makes 4 filters; Poor Sourcing
Wratten 89B Gelatin Filter Sheet	65.00	Makes 4 filters; Poor Sourcing

Table 4.3: The total costs of the imaging device without consideration of taxes or shipping and handling, as of March 2021. Some prices are rough estimates as many alternatives and/or bundles can be found for those items or materials.

Part	Cost (USD)
Raspberry Pi 4 B 2GB	35.00
32GB microSD card	5.00
Raspberry Pi power supply	7.95
ArduCam UC-599 Rev.B	41.99
Edmund Optics Plastic Long-pass Filter	12.50
PLA 3d-Printing Plastic	10.00
Misc. (Wires, Screws, etc.)	10.00
Total Cost	122.44

Chapter 5

Software

5.1 Raspberry Pi Image Capture

Before testing the potential cameras, two scripts were written for image capture. One script (Appendix A) was written for the RPi NoIR Camera v2 while the other (Appendix B) was made for all the ArduCam cameras. In essence, both scripts did the same four processes: 1) initialize the camera, 2) tune the camera settings, 3) turn on the pins corresponding to the excitation lights, and 4) take an image of the chlorophyll fluorescence. Additionally, both scripts were written in Python.

In the first script (Appendix A), the PiCamera package[1] was imported (line 5) in order to recognize and control the RPi NoIR Camera v2 for testing. Otherwise, the Raspberry Pi's GPIO and time packages were also imported. The GPIO package gives access to the Raspberry Pi's signal pins which allows for switching on of the excitation lights, connected to the RPi's pins through electric wires. GPIO stands for "general purpose input/output" and refers to the settings of the digital pins on the Raspberry Pi. Meanwhile, the time package was used to pause the script during execution if necessary.

After importing the packages, references and objects were initialized in the script. Initially, the cameras were also tested for quantitative measurements of chlorophyll fluorescence, so a pin was designated for the background light (line 8-9). However, as the project focused more on canopy imaging, background light removal became no longer necessary nor wanted, so lines pertaining to imaging with background light were commented out. Aside from the

[1]PiCamera can easily be installed onto the Raspberry Pi using apt, the system's package manager.

background light, the excitation light was also assigned to a pin (line 10-11). Finally, a camera object was initialized with the PiCamera package (line 13). In initializing the camera object, the resolution and frame rate were also set. These settings can be changed depending on the camera and needs of the system, but since chlorophyll fluorescence imaging only requires a single frame, the frame rate setting did not matter.

Once all the necessary references and objects had been set, the script then adjusted the camera's other settings. In this case, the ISO, or 'sensitivity', and shutter speed were adjusted (line 17-20). In the PiCamera package, ISO functions similarly to gain, and using the shutter speed parameter is another way of adjusting exposure time, since there is not a parameter for exposure time itself. Additionally, the auto exposure and auto white balance of the camera were also turned off (line 21-24). This was necessary for chlorophyll fluorescence imaging since leaving these functions on could greatly alter images of the same setup from one another, especially in low light conditions. Alteration of the images could lead to poor threshold processing and near impossible plant separation.

Next, the excitation light pin was set to digital high (line 32), switching on a relay for the strong blue excitation light. As mentioned before, an excitation light is used to get a fluorescence response from the plants to be imaged. Following this, a frame was captured as the raw chlorophyll fluorescence image to be processed (line 34). Finally, the pin was set back to digital low, and all the pins were reset in the GPIO (line 36-38). The amount of time that the blue excitation light remained on was approximately the same as the shutter speed that was set for the camera as the digital pin was set to high just before image capture and to low immediately after.

5.2 ArduCam Image Capture

Similar to the Pi NoIR Camera's testing script, the ArduCam image capture script (Appendix B) performed the same key processes. The cameras tested from ArduCam all used an interface known as MIPI, which is a widely used camera interface known for its ease of use and support

of high-performance applications. This script was adapted from one of ArduCam's sample preview scripts and was also the final one used in the canopy imaging device. The main difference between this script and the Pi NoIR's script was the use of ArduCam's unique package[2], as well as the V4L2 package[3]. Fortunately, all of ArduCam's MIPI cameras could be used with this single script without any changes.

As with the Pi NoIR's script, the necessary packages were imported first (line 1-4), and a pin was set for the excitation light (line 9-10). Following this, the camera object was established with ArduCam's MIPI Camera package (line 28-30). In this program, the resolution of the camera was not set because each of the potential cameras were already defaulted to their maximum resolution. Additionally, since this script was adapted from a sample preview script, the function for previewing image capture (line 35) was used during testing and did get commented out for the final imaging script.

A function for adjusting the camera settings using the V4L2 package was defined before the main portion of the script (line 12-22). Here, the settings controlled were exposure and gain. In general, gain and exposure were set as low as possible while maintaining distinction between the plants and background. Same as before, the camera's auto exposure function was also turned off (line 19). After initialization of the camera object, this adjustment function was called (line 36). Next, the excitation pin was turned on (line 47), and a frame was captured by the camera in the 'jpeg' format. This frame was then saved to a file (line 52-53) to be processed later. Afterward, the excitation light pin was turned off and reset while the camera object was closed (line 61-66).

5.3 ImageJ Macro Command

ImageJ was chosen to be the imaging device's processing program because it is the most commonly used image processing program for canopy and chlorophyll fluorescence imaging

[2]Instructions for installing ArduCam's package can be found on their official website.
[3]The V4L2 package can be installed using apt, similar to PiCamera.

[5, 10, 19]. It has command line execution, which is necessary for the automation of this system, and is also easy to install onto the Raspberry Pi because there is an official package for it. In fact, ImageJ can be run headless without a login through terminal commands which is what allows its automation. Otherwise, ImageJ contains all the basic functions that this system needed for chlorophyll fluorescence imaging, and its steps are outlined in Appendix D. Briefly, the functions were: 1) load an image file, 2) convert to 16-bit format, 3) remove noise through "despeckle", 4) convert to binary through "Intermodes" thresholding, 5) remove noise once again by "despeckle", and 6) measure number of white pixels. In the event that some processing methods need to be added to the imaging system and aren't available in ImageJ, the image processing program can be replaced by a custom processing script using open source libraries such as OpenCV.

From the command line, the Raspberry Pi version of ImageJ was able to run an ImageJ macro to process any available image using its batch or headless macro mode. Using these modes and a macro script helped to keep all processing steps consistent between images as well as improved efficiency when processing many images. One important note about the Raspberry Pi's official version of ImageJ is that it cannot run its headless mode without detecting a connection to a display, such as a computer monitor. This meant that the program could not run without having a connection to one of its HDMI ports and was a problem mainly for automation of the Raspberry Pi. An easy solution that was found for this issue was to implement Xvfb[4], otherwise known as the X virtual frame buffer. This program created a virtual graphical interface which the Raspberry Pi recognized as a display and used in the place of a physical HDMI connection. Installation of Xvfb is similar to that of ImageJ as it has its own official package. To use the virtual frame buffer, a simple shell script was made to run upon each booting of the Raspberry Pi by adding execution of the script inside the rc.local file. The contents of the Xvfb shell script are seen below:

```
1  sudo Xvfb :1 -screen 0 1024x768x24 </dev/null &
```

[4]Both ImageJ and Xvfb can be installed to the Raspberry Pi using apt as well.

```
2  export DISPLAY=":1"
```

In order to run the batch or macro mode of ImageJ, a macro script must be made. Fortunately, macro scripts can be recorded by running ImageJ and selecting to record a macro which then creates the script off of functions run from the user interface. It is important to record the macro from the version of ImageJ that will be running it as the ImageJ macro language differs slightly between the original ImageJ, the one that was installed to the Raspberry Pi, and ImageJ2, which is the newer version. The macro script used for this imaging device (Appendix C) was recorded on the Raspberry Pi itself.

When executing the ImageJ macro mode from the command line, a string of arguments was added. The macro first parsed this argument string (line 1), which contained two arguments, and set them in an array. The macro command can take any number of arguments so long as they are contained within a single string for the command line execution and can be parsed inside the macro script. After reading the argument string, the script then converted the image into a 16-bit format (line 3) which was necessary for the threshold function. It then ran the "despeckle" tool which removed some noise from the image, performed an automatic threshold function, and ran the "despeckle" tool once again (lines 4-11). The automatic threshold function in this macro script used the "Intermodes" setting which was found to perform the best when testing all the options over a set of canopy images. Most notably, the "Intermodes" setting distinguished smaller plants much better than other popular options such as Otsu's method (Fig. 5.1). As described on ImageJ's official website, the "Intermodes" threshold assumes a bimodal histogram which is repeatedly smoothed until there are only two local maxima. The threshold is then calculated as the average of those two maxima.

Once the image had been run through the "despeckle" tool a second time, it had been completely processed and was saved to a file, labeled by the first argument given (line 14). Next, the script used the second argument to discern if multiple plants were imaged and needed to be separately analyzed or if all the imaged plants should be analyzed together, to

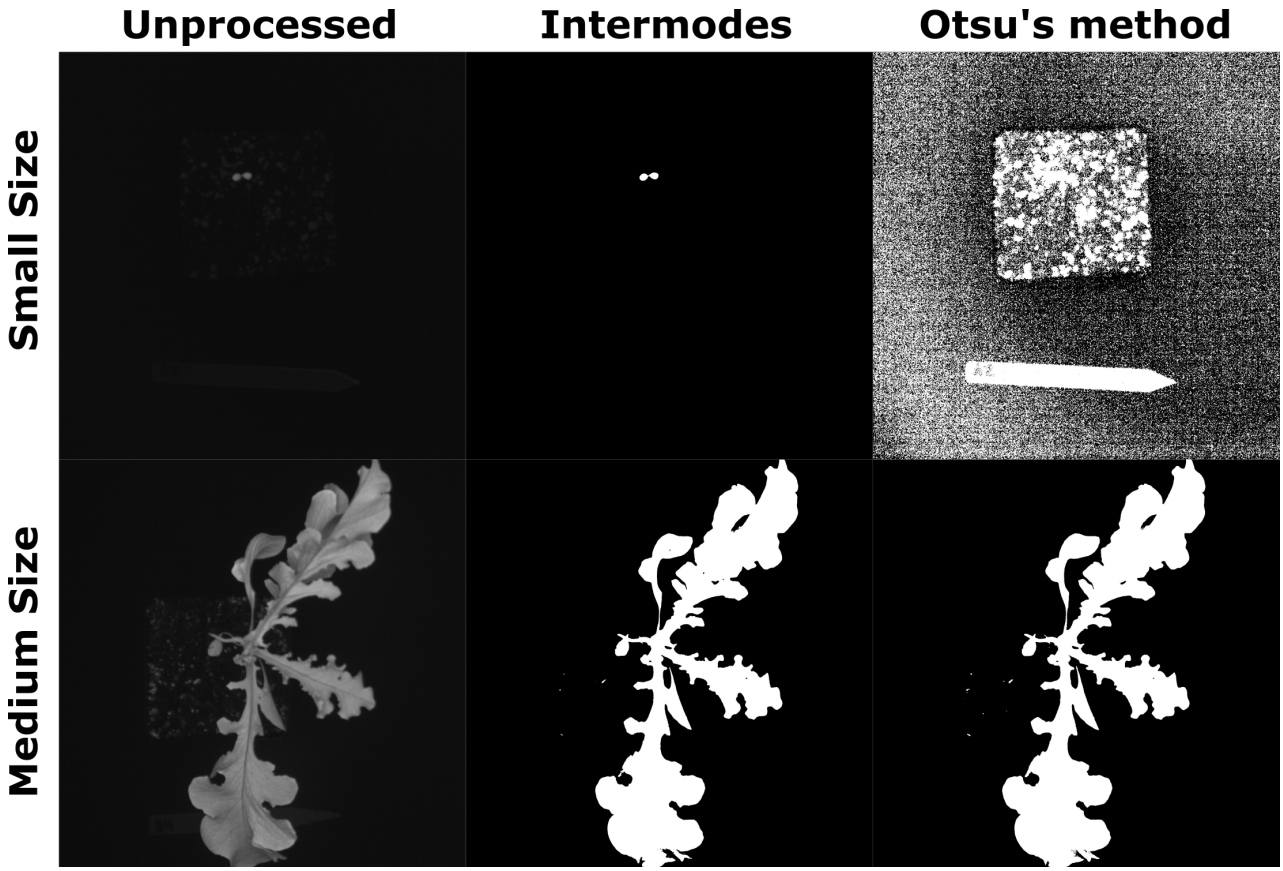

Figure 5.1: Sample images from the threshold methods test. The plants were imaged with a labeling stick for size comparison in the same imaging setup. Between all of the auto-thresholding methods provided by ImageJ, Intermodes performed the best between all sample images.

get the total canopy size. For the needs of this imaging device, the "single" option was mainly used. However, some testing on the possibility of separately analyzing multiple plants was done in case future implementation of this is desired. If the second argument of the command line execution was for "multiple" plants, then the script would use the "Analyze Particles" function to measure each blot of separated plant and then export a spreadsheet of the plant measurements as well as a secondary processed image with the outlines of each plant (lines 16-23). It is important to note that the multiple-plant analysis option is not fully developed and is picky in that the imaged plants must be adequately separated and consistent in shape. Otherwise, if the "single" option was inputted for the second argument, then the script selected the separated plants, which turned white from the automated threshold function, and measured the total canopy size, saving the results to a spreadsheet file (lines 24-29).

After processing and measuring the image, the program finally closed out all of the windows and shut down (lines 31-42). This portion of the script may seem exaggerated, but it is to make sure that all of ImageJ has finished and is ready for its next macro when executed. The largest benefits to using ImageJ were not only that it practically had every image processing function that would be needed but also that its macro scripts could be easily customized to perform any processing functions that would be performed on it normally.

5.4 Rclone Cloud Transfer

From the Raspberry Pi, there are four possible methods of extracting the chlorophyll fluorescence images and analysis results. The first and most obvious is to manually log into the Raspberry Pi and copy the files onto a connected USB. This method is the least convenient since the canopy imaging device would have to be unmounted from its setup and disassembled in order to access directly. The other obvious method is to connect remotely via a secure shell (ssh) into the Raspberry Pi from another system on the same wireless network and download the files. This would be possible to automate, but there are two easier methods

that could be used. When automating the imaging system's operations, two methods were implemented for file transfer: 1) cloud storage transfer and 2) SMB file sharing.

The program used in implementing cloud storage transfer for the Raspberry Pi was Rclone, a command line program written and maintained by Nick Craig-Wood to allow interaction of the Raspberry Pi with many various cloud storage services such as OneDrive, Google Drive, and Dropbox. The particular service used for this imaging device was OneDrive, but any of the supported cloud services may be substituted into the automated pipeline. To install Rclone, an installation shell script can be downloaded and run through the command line[5]. When setting up Rclone on the Raspberry Pi, a new connection must be configured for each cloud service account to be used. The new connection will ask for which cloud service it should link with and then ask for login information of the specific account. After logging into the cloud service, the connection should be completed. Using Rclone is very simple once the connection or connections have been made, and only one command line function is needed for syncing with the cloud service. The "sync" command takes two arguments, an input directory and an output directory. Essentially, the command will copy and delete files so that the output directory contains exactly the same files as the input directory. This is an important detail of the Rclone command as files may be lost if the input directory does not contain everything that is desired. To avoid this potential problem, the automated pipeline (Appendix D) imported the OneDrive directory at the start of the pipeline and exported the local directory at the end, once all the image capture and processing have been completed.

5.5 SMB File Sharing

The second file transfer method used in automating the canopy imaging pipeline was SMB, which stands for "server message block"[6]. SMB is a Microsoft client-server protocol that allows for users, including the imaging device's Raspberry Pi, to create, open, and edit files

[5]Instructions are available on the Rclone Project's official website.

[6]SMB comes already installed on the latest version of Raspbian, the Raspberry Pi's OS

on a remote host located within the same local network. By setting up a file server with SMB, a directory on the server can be made available for other systems on the same network, such as a Raspberry Pi, to connect directly to. Once a file server was set up, the imaging device could mount, or connect, one of its local directories to the shared directory of the server. In order to ensure that this local directory was automatically mounted when the imaging system was powered on, the Raspberry Pi was set to wait for network connection on boot, and the specific SMB connection command was added to the Raspberry Pi's fstab file which handles the mounting of system files. Once the Raspberry Pi's local directory was mounted, all files from the server should appear in it, and any changes made to that directory will directly affect the server's files. This meant that the imaging system's automated canopy imaging pipeline can simply move the desired files into its mounted local directory for file transfer. The largest downside to implementing SMB file sharing was having to set up a shared directory. This did not have to be done on a dedicated file server, but the process required some expertise. Aside from setting up the SMB file sharing itself, the Raspberry Pi was readily compatible with the SMB protocol, so connecting to the shared directory was simple and only required one command, which asked for info such as the shared directory's name and IP address along with user login information.

5.6 Automation of Imaging Pipeline

While the canopy imaging system can be run solely with the image capture script and ImageJ macro script, using command line functions, this can become tedious with the amount of functions needed for each image capture and processing performed. Therefore, a pipeline was made to simplify the process, using a shell script (Appendix D). The shell script used Bash, a Unix shell and command language, to run through all the commands written in it. In essence, this shell script was just a list of all the command line functions that should be run for chlorophyll fluorescence imaging of the plant canopy. Two major benefits to using a shell script were that variables such as path names could be declared and that the steps or

commands can be modified easily. Running the shell script was as simple as entering it into the command line with Bash or by making the Bash file executable.

First, the shell script declared all the variables it will use, mostly path strings for the directories (lines 3-11). These variables can be changed depending on each user's needs for the system. The RCLONE and SAMBA variables should correspond to the local directories made on the Raspberry Pi for syncing with the online cloud storage and mounting to the SMB shared folder. The ONEDRIVE variable corresponds to the connection made on the Rclone program and is able to direct to any folder located on the OneDrive account. Finally, the TYPE variable is used to determine what type of image processing is done in the ImageJ part of the pipeline. This should either be "single" or "multiple", based on the ImageJ macro script that was made. Otherwise, the NAME variable can be changed to fit and differentiate the imaging system, and the DATE variable can be left as is. Both of these variables were used in the ImageJ macro script to name the processed files, avoiding naming conflicts which would cause deletion of previous files.

After variable declaration, the shell script then synced the local Rclone directory with the online storage space (line 14). Next, it ran the ArduCam image capture script with Python to get the raw image of the plant canopy (line 19) and moved it into the image processing folder (line 24). Following this, the script set the DISPLAY variable of the environment to that of the virtual graphical interface, created by Xvfb (line 27). This was necessary since the default DISPLAY variable was set to that of the physical HMDI port, and leaving it as is would cause the ImageJ macro function to fail without being connected to a monitor. The ImageJ macro function was called next with several options for amount of allocated memory, the source image file, the macro script file, and a string of macro arguments (line 28). After the ImageJ function was completed, the original raw image file was renamed to match those of the processed images, and all the files were either copied or moved into the file transfer folders, using the SAMBA and RCLONE variables previously declared (lines 33-35). Finally, the OneDrive cloud directory was synced with the local Rclone directory

once again to obtain the new files. A diagram of how the imaging devices, file server, and cloud storage would be connected can be seen in Figure 5.2.

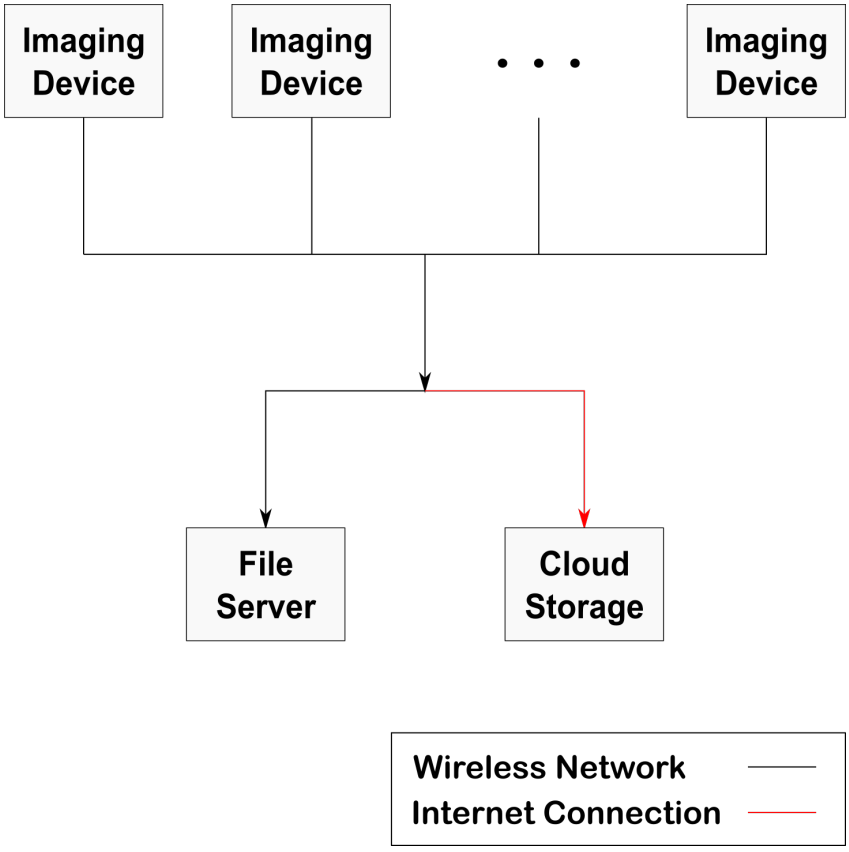

Figure 5.2: A visualization of the potential connection network of the imaging devices with a file server and/or cloud storage service. All image acquisition and analysis was performed on the devices while the resulting files were transferred through the wireless network to the file server and internet connection to cloud storage.

As mentioned before, this shell script can be easily customized. For example, if a system were to not use Rclone, all lines that pertain to it could be commented out or removed without affecting the rest of the script. Also, any of the variables can be adjusted to fit how the system was set up. When manually testing this shell script, one main issue was encountered with ImageJ. Seemingly randomly, the pipeline would stall during the ImageJ macro, precisely when the image was run through the threshold function. Even after greatly increasing the allocated memory and remaking the macro script, this problem did not get

resolved. The conclusion as to why this issue occurred was due to the plugin that ImageJ uses for its threshold function. Unfortunately, further inspection of the plugin was not possible, so a work-around was used to accommodate the problem. As this problem only happened occasionally and didn't affect the imaging system beyond stalling the pipeline, the best method of dealing with it was found to be canceling the operation and running the pipeline again. For manual use, this process was simple enough, and for automation, the script was just scheduled to run multiple times. The resulting files would be the same regardless of how many times the script was run, as long as the scheduled times were close enough.

After creating the shell script pipeline, automating the canopy imaging process was simple. Unix systems such as the Raspberry Pi contain a program called crontab. `crontab` is a command scheduler that will automatically run commands given to it at a specified time and date. Options for scheduling commands in crontab are almost limitless. Commands can be scheduled to run at set intervals of time or at a specific time, or even on specific days of the week. By adding the command for running the shell script to crontab and specifying the times and days to run it, the Raspberry Pi will attempt to run the command whenever the specified time arrives as long as it is powered on. One important detail that may become relevant is that crontab will base its operations on the Raspberry Pi's time. Additionally, Raspberry Pi's lose their time setting upon powering down. Therefore, if the Raspberry Pi is not connected to the internet, its time may deviate from real time, and the crontab commands may not run as desired. This was best resolved by occasionally setting the Raspberry Pi's time through SSH using the "date -s" command.

5.7 Data Flow

From the automated imaging devices, images and measurements were received. After each time the canopy imaging pipeline was run, a raw image, an automatically processed image, and a canopy size measurement were transferred in the form of two jpeg files and a csv file. Using the pixel count measurement in the csv file, either percent growth or canopy area can

be calculated. To calculate the percent growth, the pixel count was simply devided by the final pixel count from the same image set and converted to a percentage. Meanwhile, canopy area was calculated by multiplying the pixel count measurement by a conversion ratio found through calibration.

Chapter 6

Experiment 1: High Light vs Low Light using Default Settings

6.1 Overview

In testing my imaging device, experiments were designed to evaluate how accurately these device could measure canopy size compared to a control canopy imaging device that has been used for studies previously. The experiments were performed over several weeks for the automated imaging to take place during an entire growth period for a set of plants. At the end of the growth period, images taken by the automated imaging device and the control device were compared based on their canopy size measurements.

Referring back to Section 2.2, a calibration had to be done in order to calculate true area measurements from images taken by the cameras during canopy imaging. For the control imaging system, a single image was taken of a ruler which was used to approximate the length of each side of a pixel and calculate the conversion ratio from an amount of pixels to square millimeters. The calibration image was taken under the same conditions as during the experiments, so accounting for the effect of distance on conversion was not necessary as this one calibration image would give a close approximation. Ultimately, the conversion ratio was found to be 0.539 mm^2 per pixel for the control system. As for my automated imaging devices, several images of a grid, consisting of 5 x 5 cm squares, were taken at various distances, from 30 cm to 60 cm. These images were taken by the camera, an ArduCam UC-599 Rev.B. Using ImageJ, the calibration images were measured for various distances and used to find the relationship between camera distance and pixel-to-area conversion ratio (Fig. 6.1). By using the relationship equation found, the specific conversion ratio for any

setup with my imaging device can be calculated based on the distance from the cameras to the plant canopies.

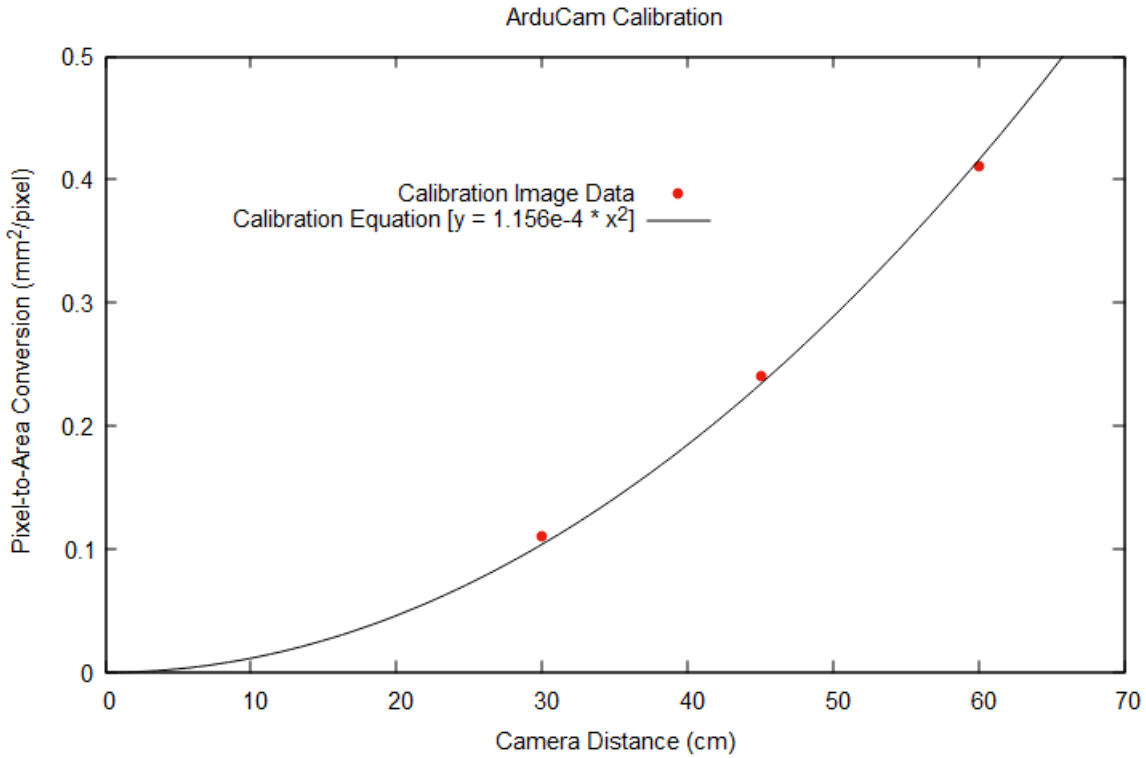

Figure 6.1: The power equation, $y = \frac{115.6}{1000000}x^2$, best describes the relationship between pixel-to-area conversion and camera distance for the new imaging system's camera, the ArduCam UC-599 Rev.B. This equation can be used to find an approximation of the conversion ratio for any setup that uses this camera.

To give perspective on the contribution of camera angle to the accuracy of canopy imaging, a second calibration image of the grid was taken by the ArduCam UC-599 Rev.B at 45 cm distance with a 6 degree angle shift from perfect alignment (Fig. 6.2). Performing the same analysis through ImageJ as with the previous images yielded an approximate ratio of 20.3 pixels to cm on the far left side of the grid and an approximate ratio of 18.2 pixels to cm on the far right. A difference of 2.1 pixels to cm demonstrates how a slight angle of the camera's orientation can greatly alter the size of images at a distance of only 45 cm.

Because there are many factors involved with the measurement of plant canopy area, there is also a large region of error. These factors include not only the ones mentioned in Section

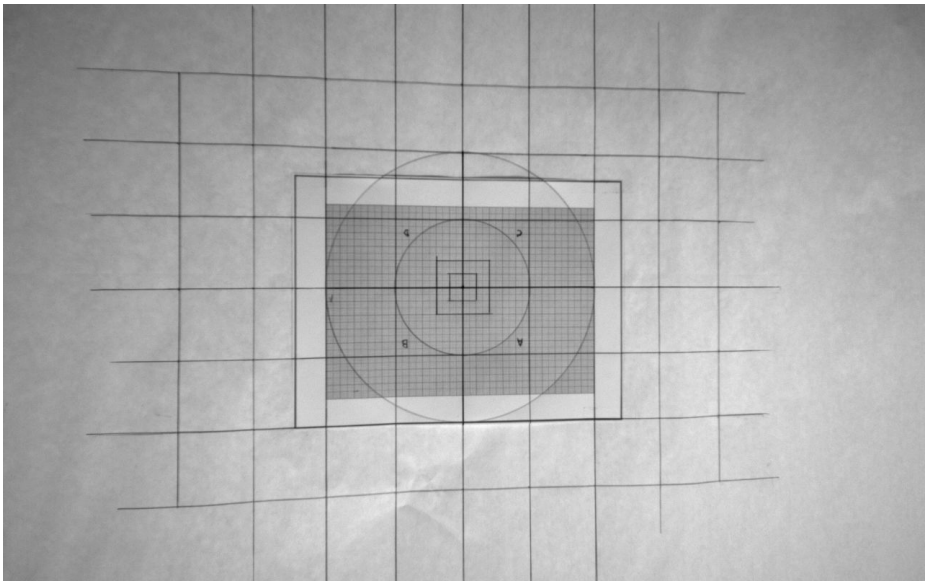

Figure 6.2: An oblique image of the calibration grid taken by an ArduCam UC-599 Rev.B. An angle of 6 degrees was used for taking the image. Analysis found approximate ratios of 20.3 pixels to cm on the far left vertical line of the grid and 18.2 pixels to cm on the far right line.

2.2, concerning the relationship between image pixels and projected area, but also the fact that the distance between the camera and canopies continuously changes during the growth period as well as how different leaves of a canopy will differ in height, causing taller ones to take up more pixels than others. Taking this large region of error into account, a relatively high goal precision of 15 percent was used when comparing the true area measurements calculated from my imaging devices to those from the control system.

6.2 Setup

To test the functionality and accuracy of this canopy imaging device, two devices were set up inside a horticultural growth chamber to monitor the growth of plants over an entire growth cycle. For this experiment, "little gem" lettuce was the plant of choice due to the height restrictions within the growth chamber. Since the height of each growth bed within

the chamber is rather low at approximately 19 in or 48 cm, a smaller species of plant was needed so that the plants would not exceed the viewing range of the mounted cameras too quickly. All plants were grown in individual pots, placed on trays of six. Each camera device was responsible for routinely imaging and measuring the growth of its own tray of plants. Between the two setups, one set of plants was grown in low lighting conditions, at a photosynthetic photon flux density of approximately 70 μmol/m^2/s provided by white LEDs, while the other was grown in high light, at about 170 μmol/m^2/s. To give perspective on the intensity of light, a well-lit office would contain approximately 10 μmol/m^2/s while full sunlight at noon during summer achieves approximately 2000 μmol/m^2/s. Both sets of plants had 20-hour photo-periods per day. Imaging would be timed for the few hours that the grow lights turned off. Otherwise, every other environment condition within the growth chamber was kept equal and as constant as possible. The temperature was kept at an average of 23.3 °C, and the average CO_2 level was kept to roughly 700 ppm.

As the control, a custom-made fluorescence imaging system, utilizing a digital monochrome camera (CM3-U3-31S4M-CS, Teledyne FLIR LLC), was used to capture canopy images of both sets of plants to be compared to the images taken from my imaging devices. This FLIR imaging system has been used for previous studies and can only be used manually, so there were fewer images taken with it over the duration of the experiment. Additionally, since different cameras have different resolutions along with the fact that each setup has differing heights, the exact amount of pixels measured through processing will differ between the camera systems, so either a calibration must be done for each system to estimate the true area measurements found or a relative comparison between the percentage of growth captured can be performed. For this study, both methods were used, and the true area calculations were utilized in finding the deviations between the systems' data.

When setting up the imaging devices, the camera settings were left to their defaults, which were the best settings found for the plastic filter when performing the filter tests. These settings were: gain set to 1, and exposure set to 2500. Neither of these settings have

units because of ArduCam's custom library. Due to the limitations of the internet service at the university, the imaging devices could not be connected to the internet directly, but they were connected to a wireless network set up with the file server that was built for this purpose. This meant that the Rclone functionality of the imaging devices was removed, and only the SMB file sharing was used. Therefore, the canopy imaging devices would just transfer its files to the file server each time it performed the automated imaging pipeline. Otherwise, the file server was able to be connected to the internet through an ethernet cable which allowed users to SSH into the server to then download all the image files received from the devices themselves. To adjust the systems for any deviations in time, the following command was entered into each system's command line periodically through SSH in order to re-calibrate the correct time and date.

```
sudo date -s "DD MTH YYYY HH:MM:SS"
```

6.3 Results

Over the 22-day growth period, canopy images were taken of the plants daily by the new imaging systems that were mounted above the growth beds. Any images taken before day 6 of the experiment showed no useful information, either because the plants had not sprouted yet or due to having too many plants. In images where the seeds had not sprouted yet, poor automated image processing occurred due to having very little contrast (Fig. 6.3). This meant that the new imaging devices provided inaccurate measurements before any of the plants sprouted. Several seeds were planted in each pot at the start of the experiment, so most pots had several seedlings after the first few days. On day 5, all extra seedlings were removed from the pots, resulting in only one in each. From then forth, the plants were not altered in any way beyond being moved into and out of the control imaging system for imaging.

During the course of this experiment, there was one incident where connection between an imaging system and the file server was lost. From day 8 to day 13, no new images were

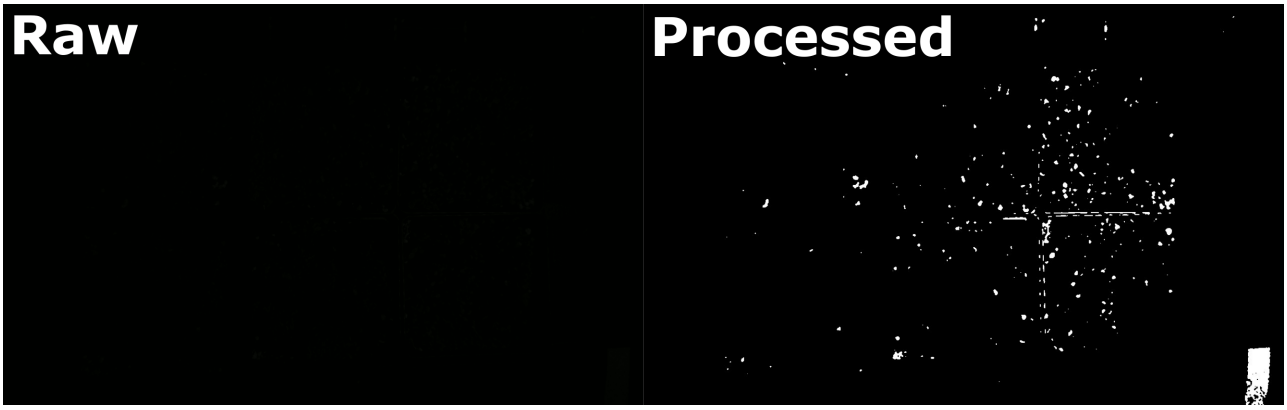

Figure 6.3: An example of how automatically processed images end up when contrast is low in the original image. The object in the bottom right of the image is a normal plant label stick.

being sent from the imaging system of the high-light growth bed. After finding this issue, an attempt to SSH into the device from the file server was performed, and it was discovered that the imaging system was not connected to the wireless network. The system was manually restarted by unplugging and re-plugging its power supply which allowed it to reconnect to the network. Once the system reconnected to the network, the missing images were automatically transferred to the file server. This meant that the system was still running its imaging pipeline during the period of disconnection, but the files just could not be transferred. From this point on, the system worked as usual for the remainder of the experiment.

As mentioned before, the growth beds had fairly low ceilings. This meant that the new imaging devices were mounted rather close to the plant trays, so the plants grew outside of the cameras' viewing ranges toward the end of the growth period. This can be seen in both high and low light systems (Fig. 6.4 and 6.5) where the outer leaves of the plants get cut off by the images. Obviously, this leads to some inaccuracy with the measurement of the plant canopies, but it should not affect any measurements taken during the earlier stages of growth.

In addition to the close placement of the imaging devices, the excitation lights needed for chlorophyll fluorescence imaging were forced to be mounted to the side instead of directly above the plants. Ideally, the lights would be mounted directly above the plants to evenly light the canopies, so this placement created an intensity gradient in the images, which can easily be seen going right to left in the day-22 images of Fig. 6.4 and 6.5. This gradient causes problems for the automated image processing because poor threshold values become more likely to be used. Manual processing, on the other hand, would not suffer from this issue. Also, as the plants grew taller, the ones closer to the excitation lights would block the light from reaching the further plants, causing more inaccuracy in the imaging. Ultimately, these experimental constraints caused a small amount of deviation in the canopy size measurements of the new imaging devices.

During the course of this experiment, images were also taken by the control imaging system (Fig. 6.6 and 6.7), a large, stationary setup with high-quality parts that costed over a thousand dollars in total. In these images, the plants were relatively small due to the distance between them and the camera. In order to use this imaging system, the plants would be moved into an enclosure and sealed off from outside light before being imaged using a computer program. Once imaged, the plants would be returned to the growth chamber. To process these images and obtain canopy size measurements, a reference image processing script, a custom Python program utilizing the OpenCV library, would be run. This custom program was created by a previous member of the lab in which the experiment was run, and it took several inputs including a range of possible threshold values, directory paths, and an image format. To get useful data, the minimum and maximum threshold values for the program would be manually adjusted until the program was able to successfully process all the images in the input directory and output all processed images and histograms as well as a .csv file containing the desired measurements. Overall, this process was simple and only took a few minutes to do after downloading and installing the necessary software.

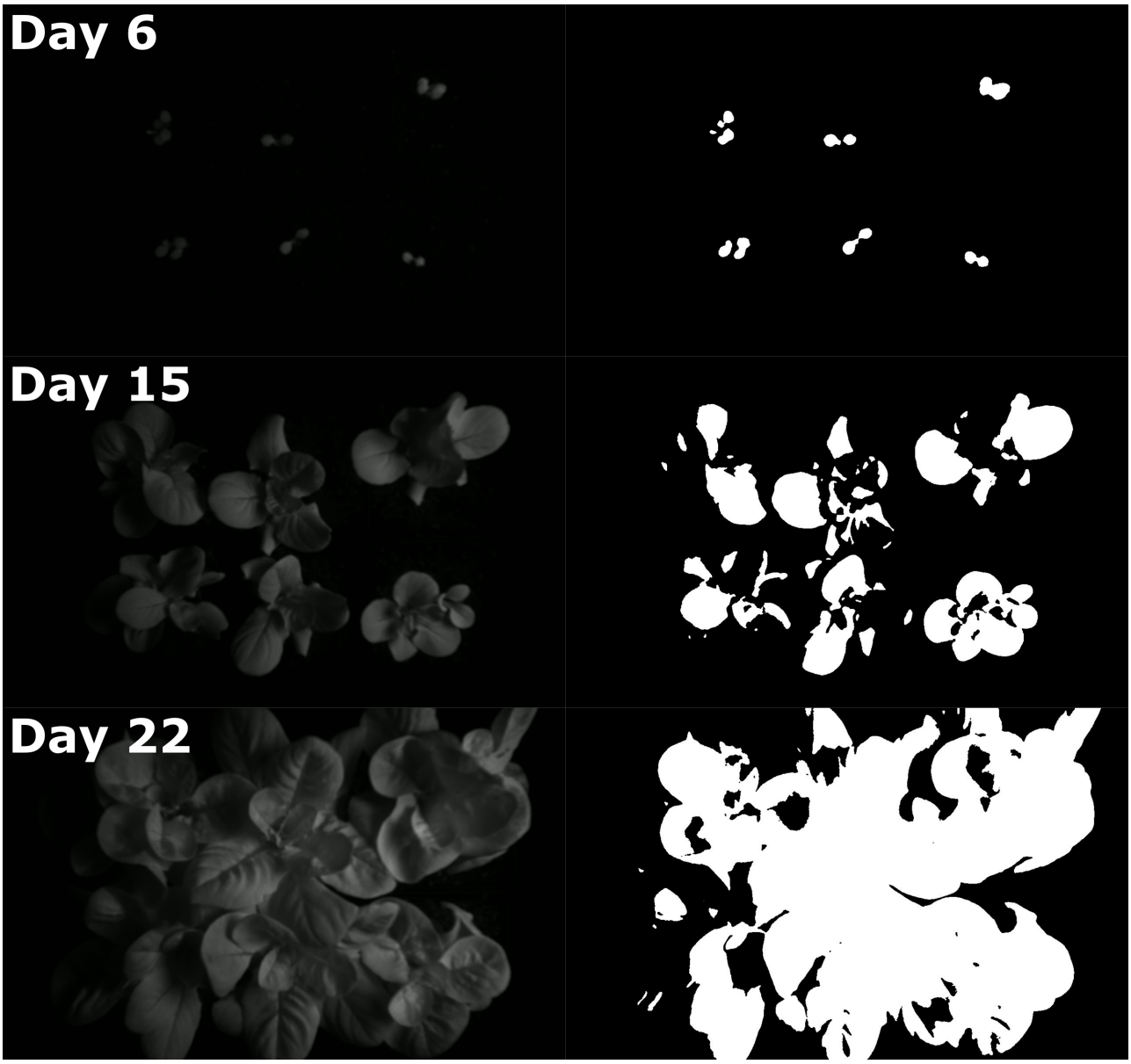

Figure 6.4: Sample images taken by the new imaging device for the high-light growth bed on days 6, 15, and 22 of experiment 1. The images were processed using the new device's automated image processing pipeline.

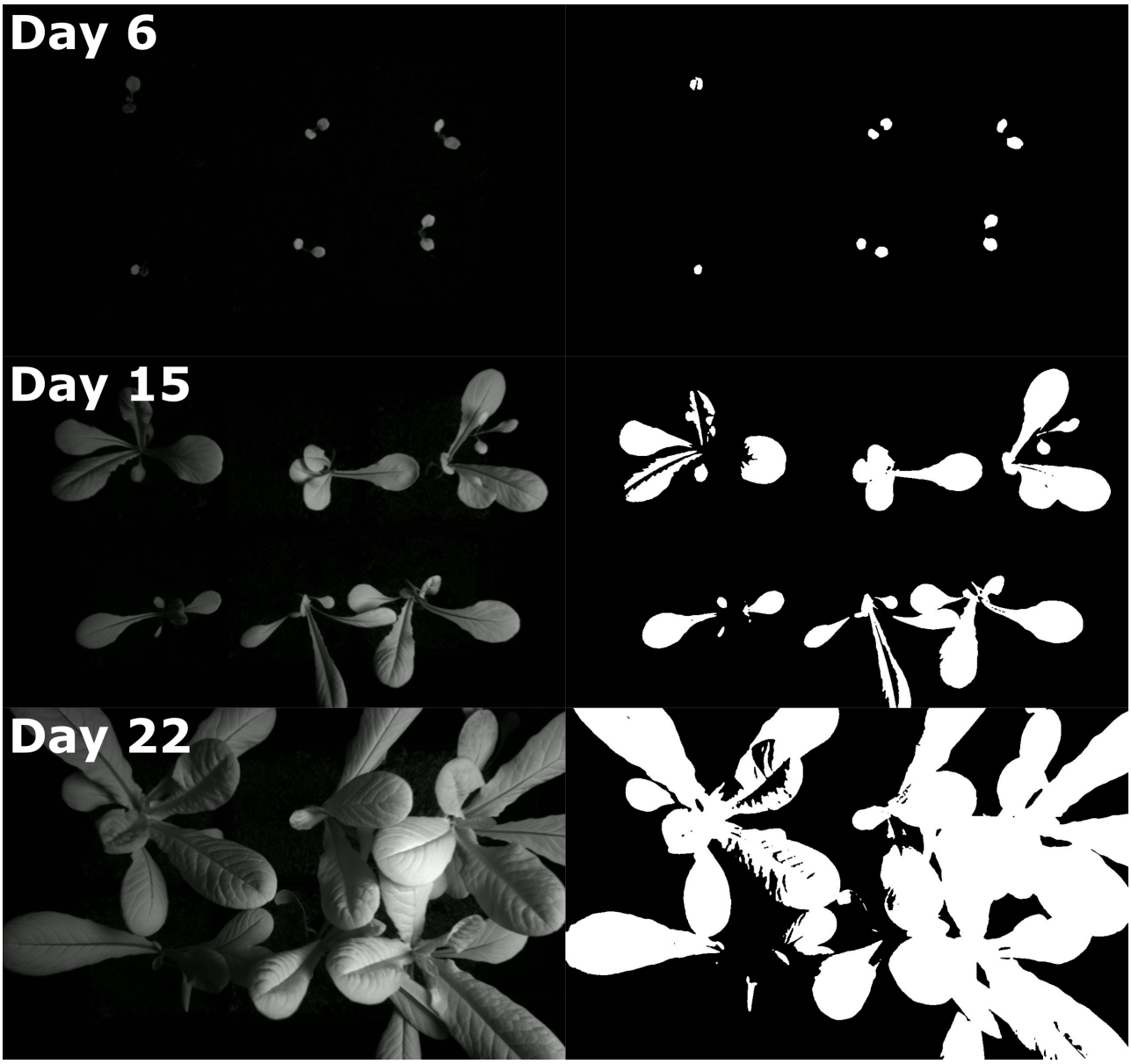

Figure 6.5: Sample images taken by the new imaging device for the low-light growth bed on days 6, 15, and 22 of experiment 1. The images were processed using the new device's automated image processing pipeline.

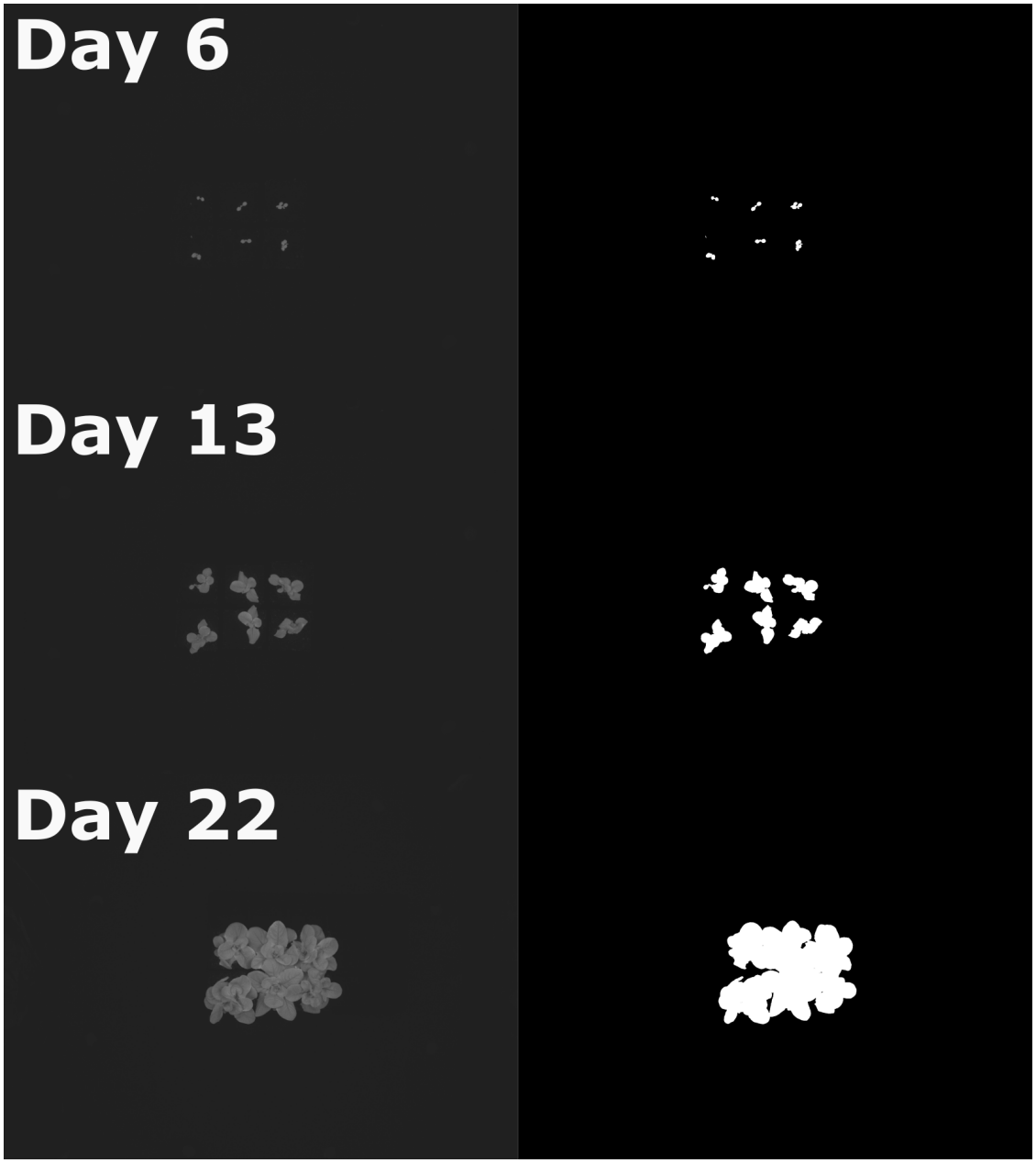

Figure 6.6: Sample images taken by the control imaging device of the high-light plants on days 6, 13, and 22 of experiment 1. The images were processed manually, using a custom Python script.

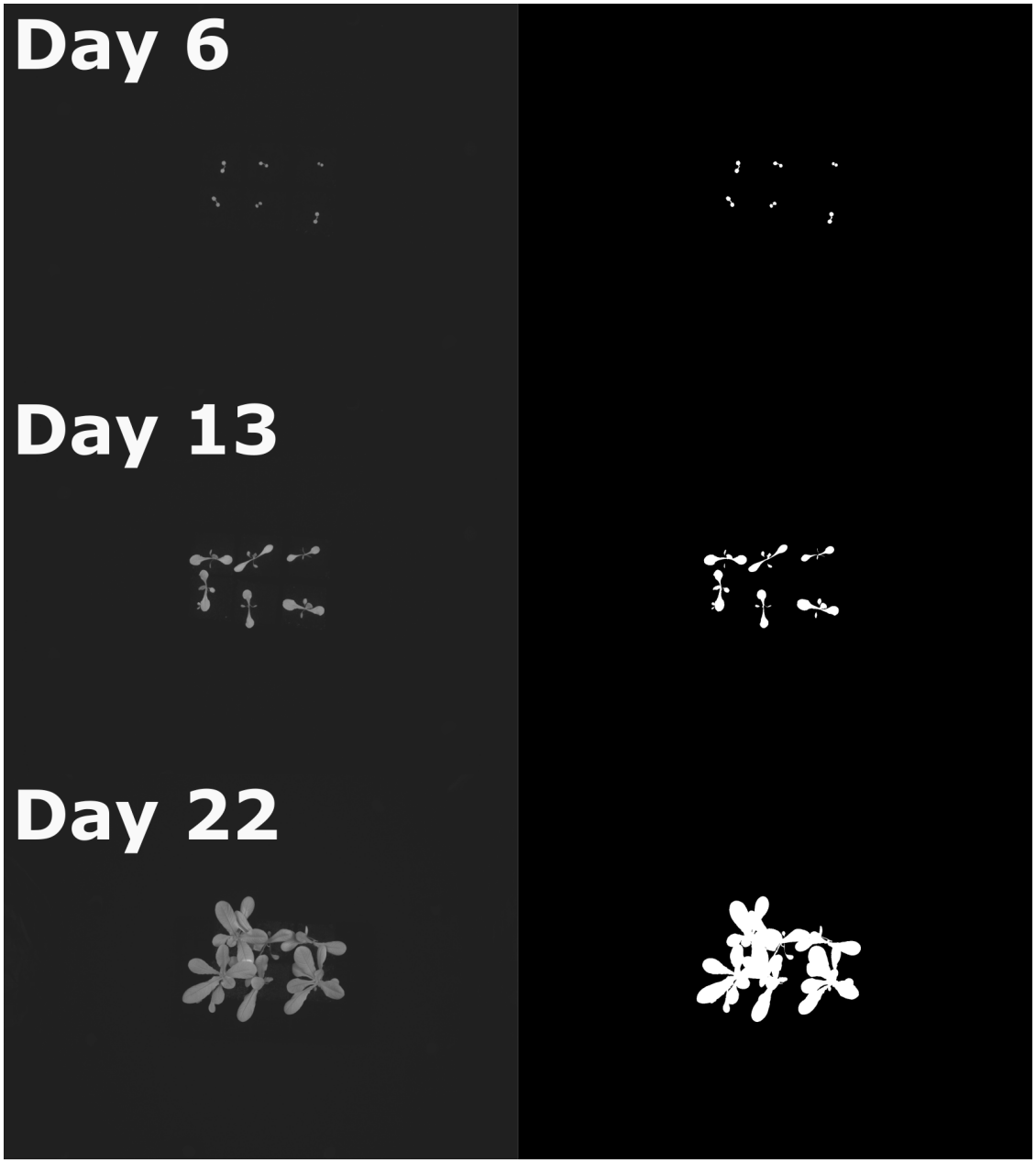

Figure 6.7: Sample images taken by the control imaging device of the low-light plants on days 6, 13, and 22 of experiment 1. The images were processed manually, using a custom Python script.

In order to compare the measurements from the new imaging systems to those from the control imaging system, percentages of canopy size were calculated by dividing the pixel measurement of each image by that of the final pixel measurement, the largest one, at the end of the growth period from the corresponding image set. These percentages were calculated for all measurements found by both the automatically processed images from my new devices and the manually processed images from the control device. Additionally, the same reference processing script was used to manually process all raw images taken by the new imaging devices (which can be seen in Fig. 6.8 and 6.9), and the percentage calculation was done for those measurements as well. Two graphs were generated for comparing the three different canopy area percentages based on the two different lighting levels (Fig. 6.11 and 6.10).

On each graph (Fig. 6.11 and 6.10), there are three lines: 1) the canopy area percentages calculated from the automated pipeline of my imaging systems, 2) the percentages found from the reference image processing on the raw images taken by my imaging systems, and finally, 3) the area percentages found from performing the same reference image processing on the control system images. Noticeably, there is a dip in the canopy area measurement for images taken by the new systems on day 19 of both graphs. This point corresponds with when the plants began to grow outside of the field of view of the cameras, causing the small dip in canopy area measurement.

Looking at Fig. 6.10 for the low-light growth bed, it is seen that the measurements found by manually processing the images from my imaging system are very close to the control measurements while the values from the automated pipeline began to drastically deviate occasionally after 15 days. This points to a problem with the automated image processing done by the ImageJ macro and was most likely due to a large gradient in the fluorescence intensity of the images which can cause varying results with how automated thresholding calculates threshold values. The large gradients were likely a result of the the grown plants casting shadows on adjacent ones and having the excitation light mounted to the side of the plants, as mentioned previously. Possible solutions, without physically adjusting the position

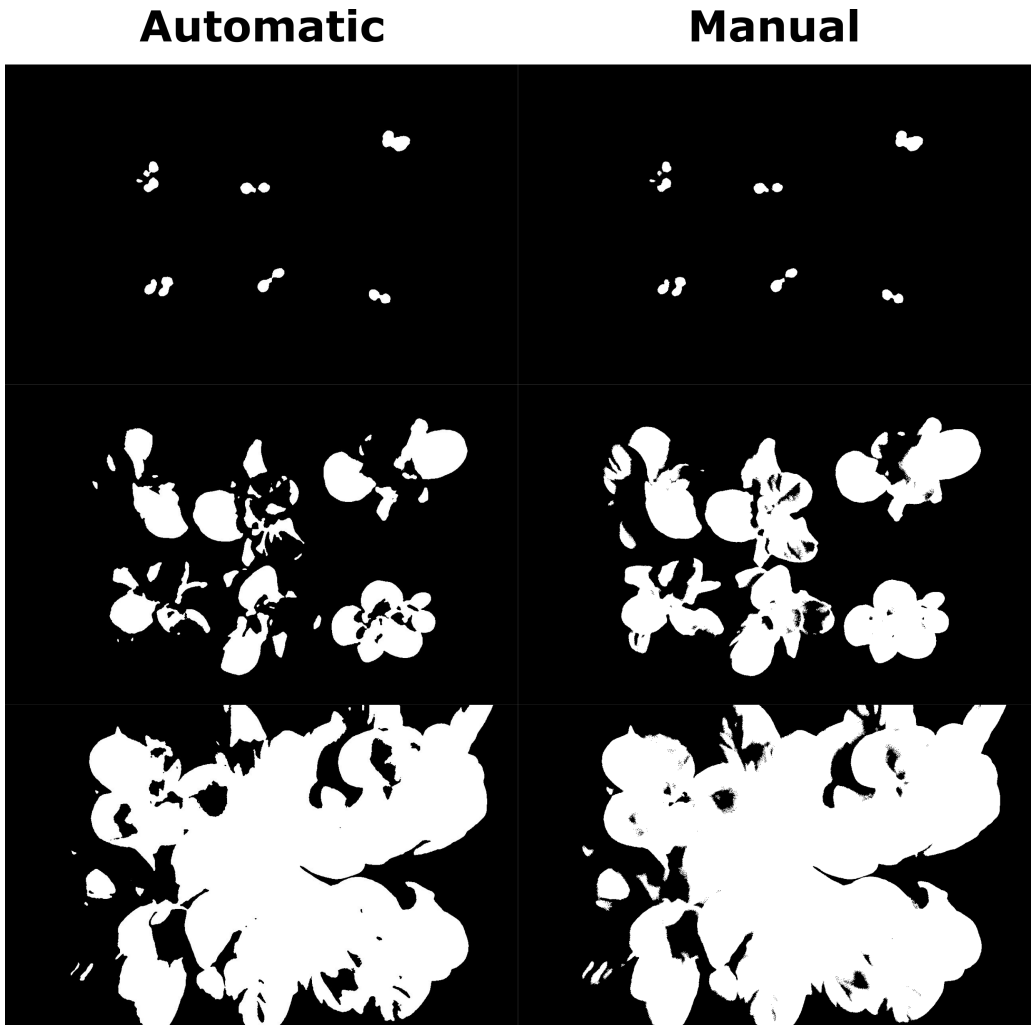

Figure 6.8: A comparison of the automatically and manually processed images based on those taken of the high-light growth bed on days 6, 15, and 22 of experiment 1.

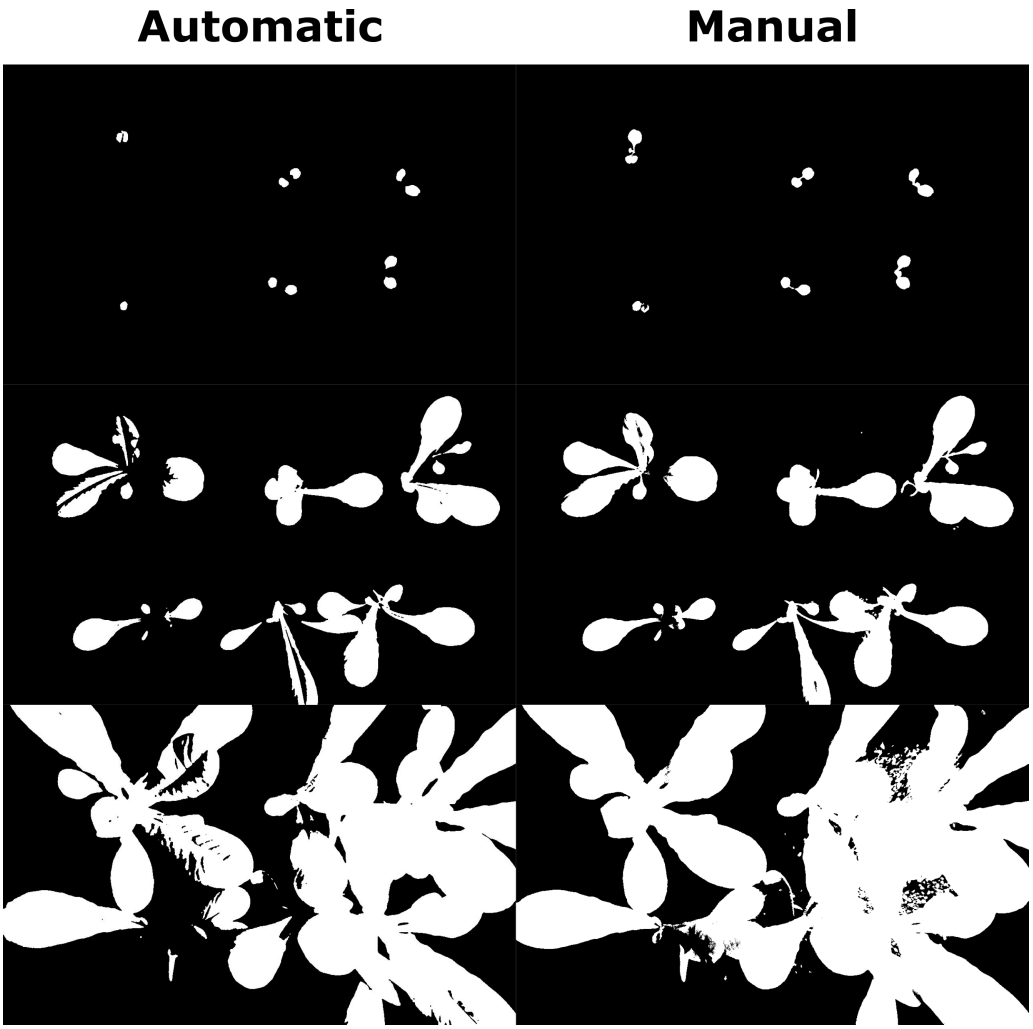

Figure 6.9: A comparison of the automatically and manually processed images based on those taken of the low-light growth bed on days 6, 15, and 22 of experiment 1.

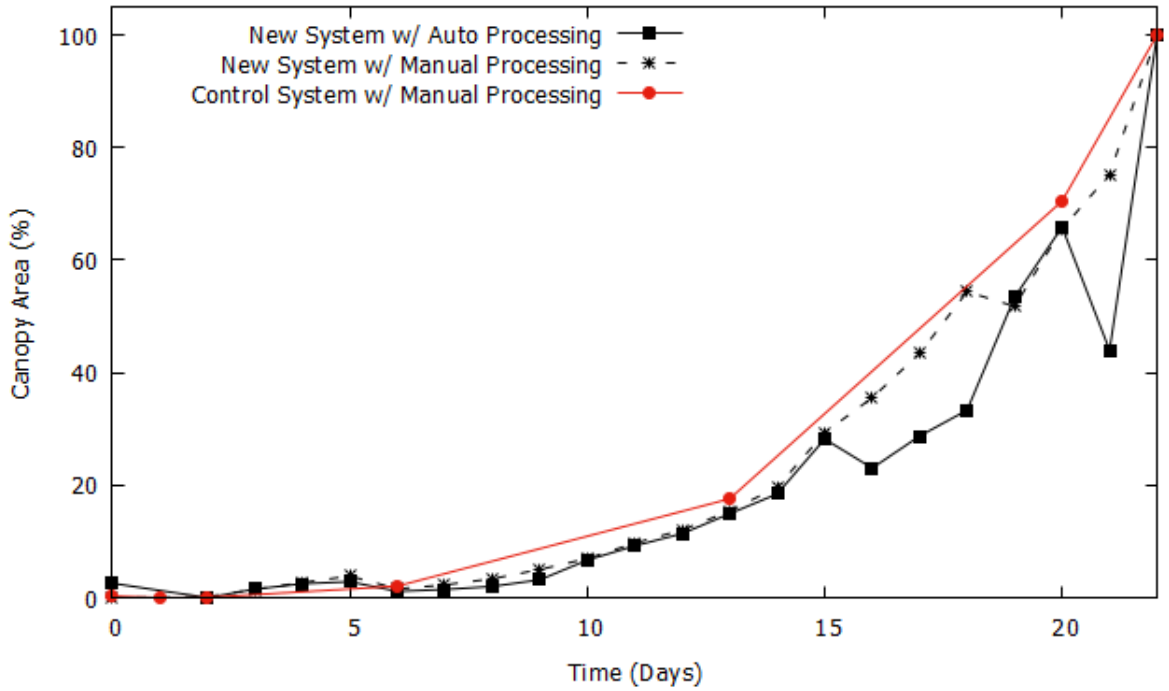

Figure 6.10: A visual comparison of the percent growth measurements calculated from the image sets captured for the low-light grown plants.

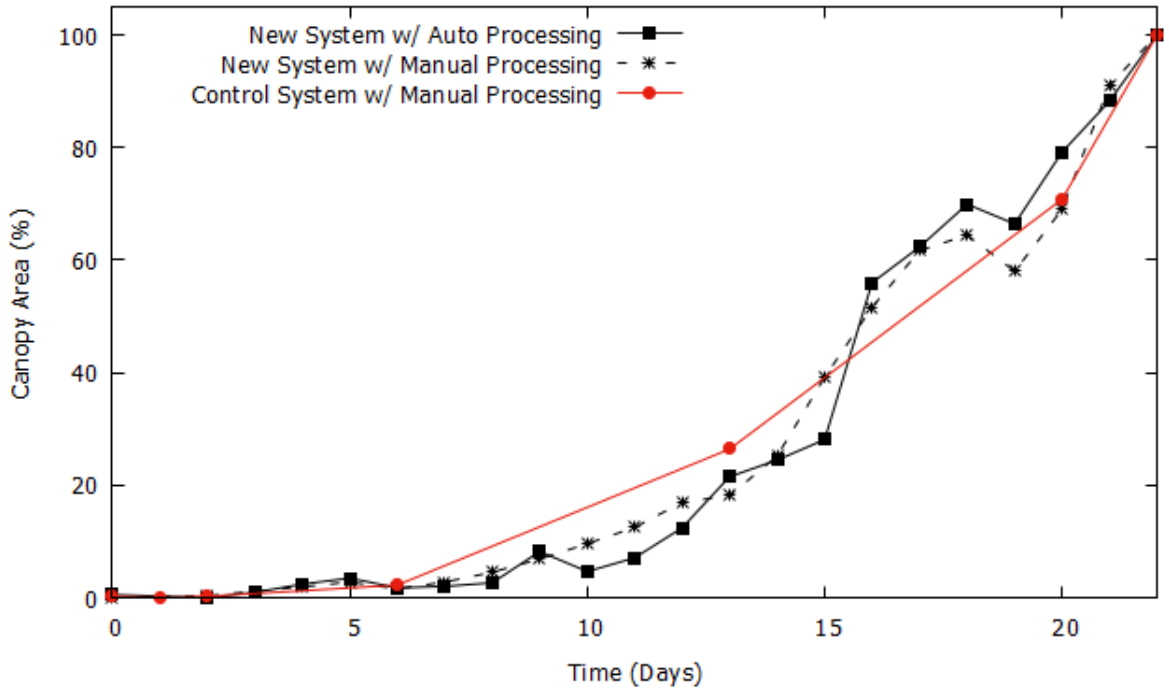

Figure 6.11: A visual comparison of the percent growth measurements calculated from the image sets captured for the high-light grown plants.

of the light, would be to increase gain or exposure with image capture or find a more suitable automated thresholding algorithm.

Meanwhile, the data from the high-light growth bed in Fig. 6.11 seems rather consistent between all the measurements. The deviations between the growth percentages found from my systems and those of the control system are probably due to the constraints of the growth chamber. Because the growth beds have low tops, the plants were very close to the imaging systems mounted above them. Coupled with the non-ideal placement of the excitation light, these constraints would lead to slight inaccuracies with the amount of chlorophyll fluorescence actually captured.

In order to calculate the true area of the plant canopies imaged, conversion ratios had to be used. For the control system's images, the conversion ratio previously found, 0.539 mm^2 per pixel, was used to convert the pixel counts into mm^2. Then, an equation of best fit was found for the control area measurements. For my new imaging devices, a conversion ratio was calculated from the equation found in Figure 6.1 of Section 2.2. While the total height of the growth beds was about 19 inches, the height of both the plant pots and the imaging devices shortened the distance between the cameras and plants to roughly 9.5 inches, or 24 cm. Using the calibration equation, a conversion ratio of 0.06659 mm^2 per pixel was calculated. All of the pixel count measurements were then converted to areas using this ratio, and all the area measurements were plotted based on experimental lighting conditions, same as before with the growth percentage measurements.

Looking at the graphs for canopy area (Fig. 6.12 and 6.13), a major trend can be seen in that the area measured through my new imaging systems deviated more from the control measurements as the plants grew throughout their growth periods. This was particularly true for the high-light growth bed which appeared to deviate further from the control measurements. One main reason for this trend is the poor capture of chlorophyll fluorescence due to inadequate excitation light, causing the measurements to not fully cover the entire plant canopy. As the canopies grow larger, a greater portion of it cannot be measured by the

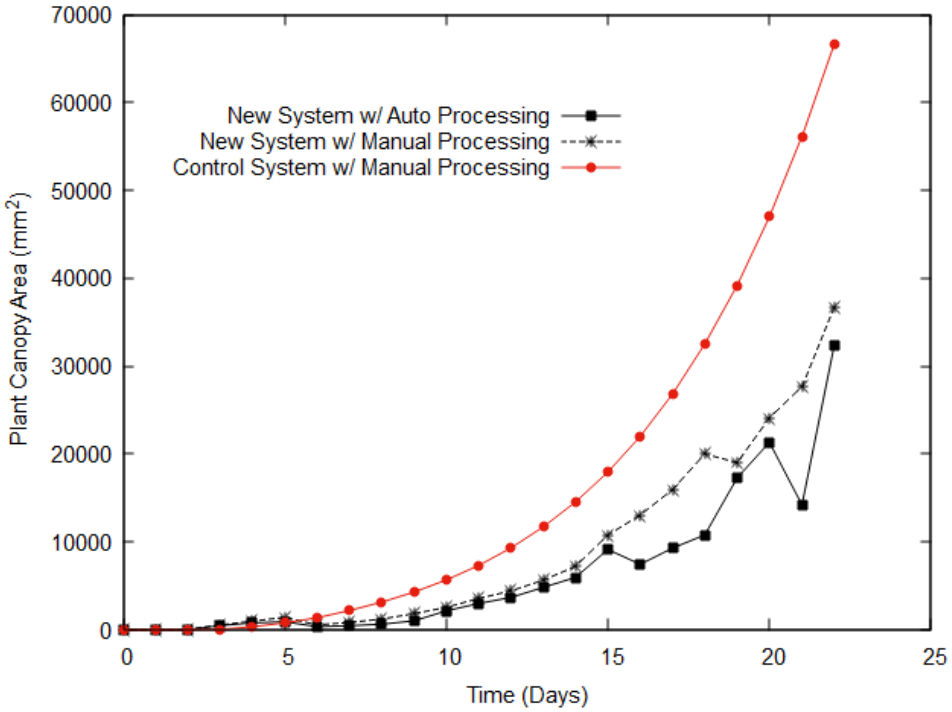

Figure 6.12: A visual comparison of the canopy area measurements calculated from the image sets captured for the low-light grown plants.

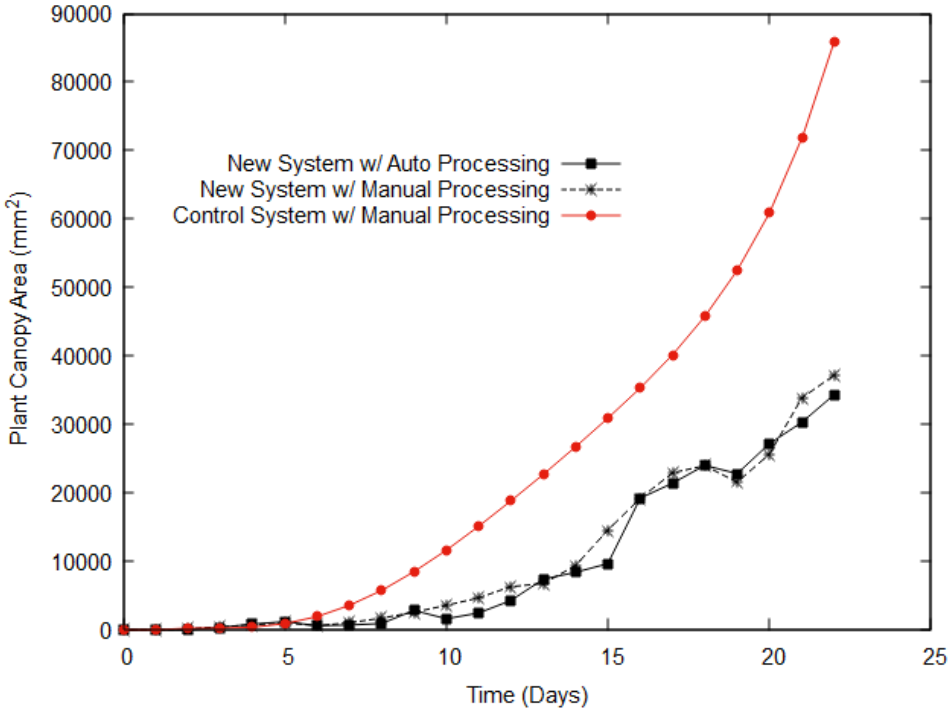

Figure 6.13: A visual comparison of the canopy area measurements calculated from the image sets captured for the high-light grown plants.

imaging. An example of poor chlorophyll fluorescence capture can be seen in Figure 6.4 of the high-light growth bed. Otherwise, the measurement lines resembled those of the growth percentage graphs (Fig. 6.10 and 6.11) strongly, which makes sense since both percent growth and area are proportionally related to pixel count.

The percent error for each measurement was found using this equation: $E = |100\% \cdot \frac{A_{Pi} - A_{FLIR}}{A_{FLIR}}|$, and an average percent error was calculated for each set of measurements. An average percent error of 61 percent was found for the automated area measurements of the low-light growth bed along with an average percent error of 52 percent for the reference processing script measurements. As for the high-light growth bed, the automated area measurements had an average percent error of 64 percent, and its reference script measurements had an average percent error of 59 percent. All of these percent errors were calculated with measurements starting from day 5 when the extra seedlings were removed, as the earlier measurements would have greatly skewed the calculations. Regardless, it is evident from the area measurements that my imaging devices did not meet the initial goal precision of 15 percent. Since the main issue for the poor canopy imaging seemed to be inadequate chlorophyll fluorescence capture, increasing the images' contrast should be the main priority in improving the imaging device's accuracy. This can be done by either increasing the amount of excitation light or by increasing exposure and gain in the camera's settings.

Chapter 7

Experiment 2: High Light vs Low Light using Adjusted Settings

7.1 Setup

Based on the findings of the first experiment, a second experiment was run with some changes to the settings of the image capture software. The aim of this second experiment was to obtain better contrast within the images and allow for better chlorophyll fluorescence capture. To that end, adjusting the physical arrangement of the imaging setups would have been ideal. However, that was not possible due to the restrictions of the growth beds, so instead, manual iteration through setting changes was done to find the highest possible values before overexposure of the raw images. During the testing, gain was found to be extremely sensitive as increasing it at all without decreasing the exposure led to overexposure of the images and poor automated processing. This is because gain is based on a logarithmic scale which means any change to its value would exponentially affect images. The exposure setting, on the other hand, was much more cooperative, going up to 4000 before any signs of overexposure. Ultimately, the camera settings for both new imaging systems in the growth chamber were changed to: gain set to 1, and exposure set to 4000. Compared to the first experiment, only the exposure was increased, by 1500.

Otherwise, the only other change to this experiment in comparison to the first one was the lighting cycle for the plants. The daily photo-period for the plants was cut back to 16 hours from the original 20 hours, and the lighting level of the high-light growth bed was adjusted to 270 μmol/m^2/s. The lighting level of the low-light growth bed was left as is, at 70 μmol/m^2/s. The other environmental conditions in the growth chamber were kept to an

average temperature of 23.3 °C and average CO_2 level of roughly 700 ppm, as was with the first experiment. "Little gem" lettuce was used for this experiment as well.

7.2 Results

The plants were grown for a 23-day growth period. Similar to the previous experiment, multiple seeds were planted into each pot, and the extra seedlings were removed after sprouting, on day 10. This meant that images before day 11 of the experiment provided inaccurate measurements.

After initiating the experiment, automated images were received from the new imaging systems each day. On day 6 of the experiment, the system attached to the high-light growth bed was found to be unresponsive. This was the same system that disconnected from the wireless network in the previous experiment. Several attempts to restart the device did not fix the issue, and it was later found that the electrical outlets for the entire racking system, on which the high-light growth bed was, had been tripped. This discovery came late into the experiment, toward the end of the growth period, and required the imaging system to be unmounted for testing. Because of that, no measurements for the high-light growth bed could be used.

On the other hand, the low-light growth bed was stationed on a different racking system, so it was not affected by the power trip. While the growth period of this experiment was comparable to that of the previous one, a noticeable difference in the size of the plants can be seen in its images (Fig. 7.1) due to the decrease in photo-period. Note that the light level of the low-light grow bed was not changed, so these plants received less light compared to the first experiment. This difference in size is important because it means that the plants did not grow large enough during the experiment to be cut off by the camera's viewing range, which should hopefully lead to better accuracy in the measurements.

In a growth percentage comparison between the different imaging systems and processing methods (Fig. 7.3) similar to comparisons done for the initial experiment, the same trend

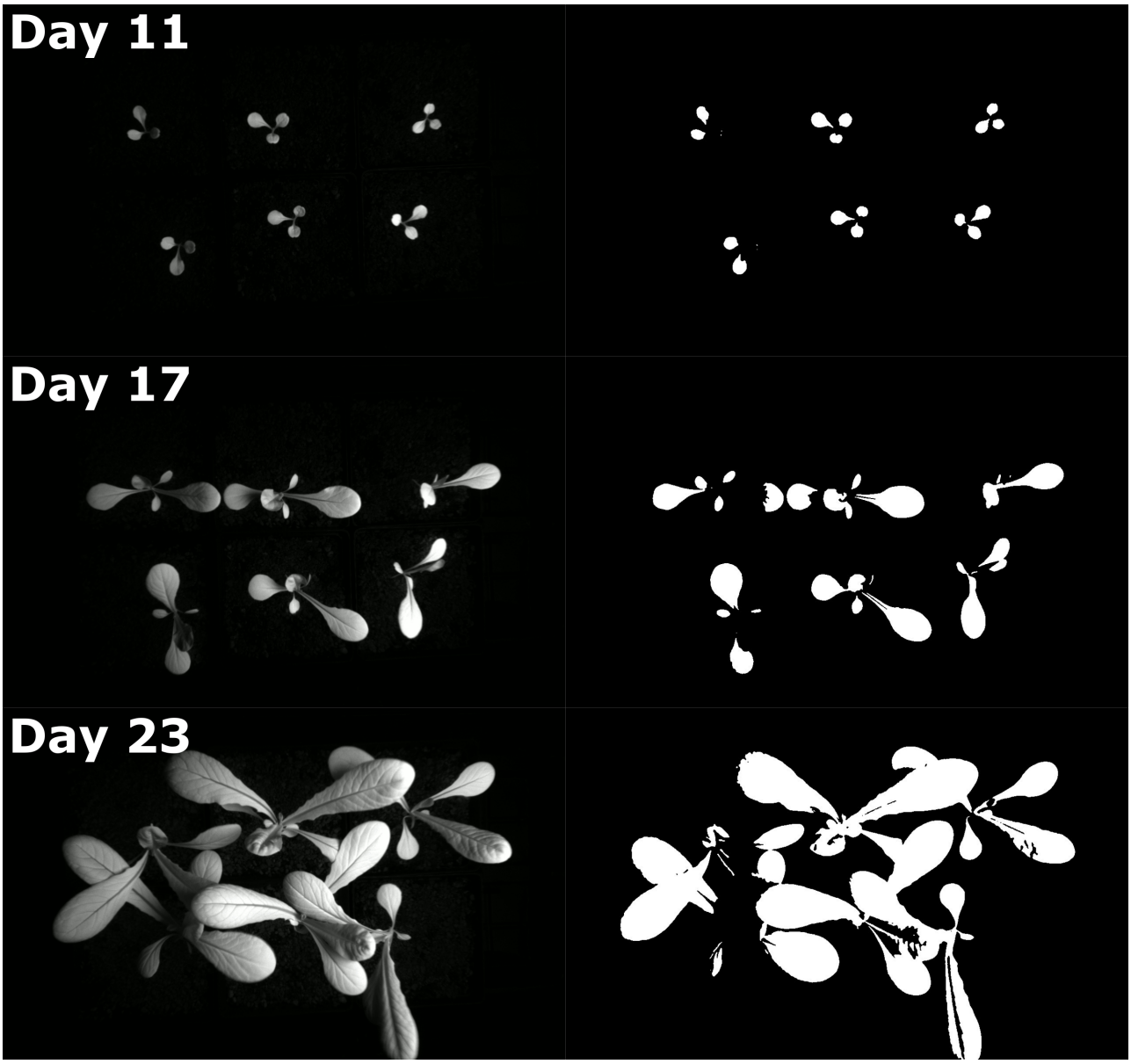

Figure 7.1: Sample images taken by the new imaging device for the low-light growth bed on days 11, 17, and 23 of experiment 2. The images were processed using the new device's automated image processing pipeline.

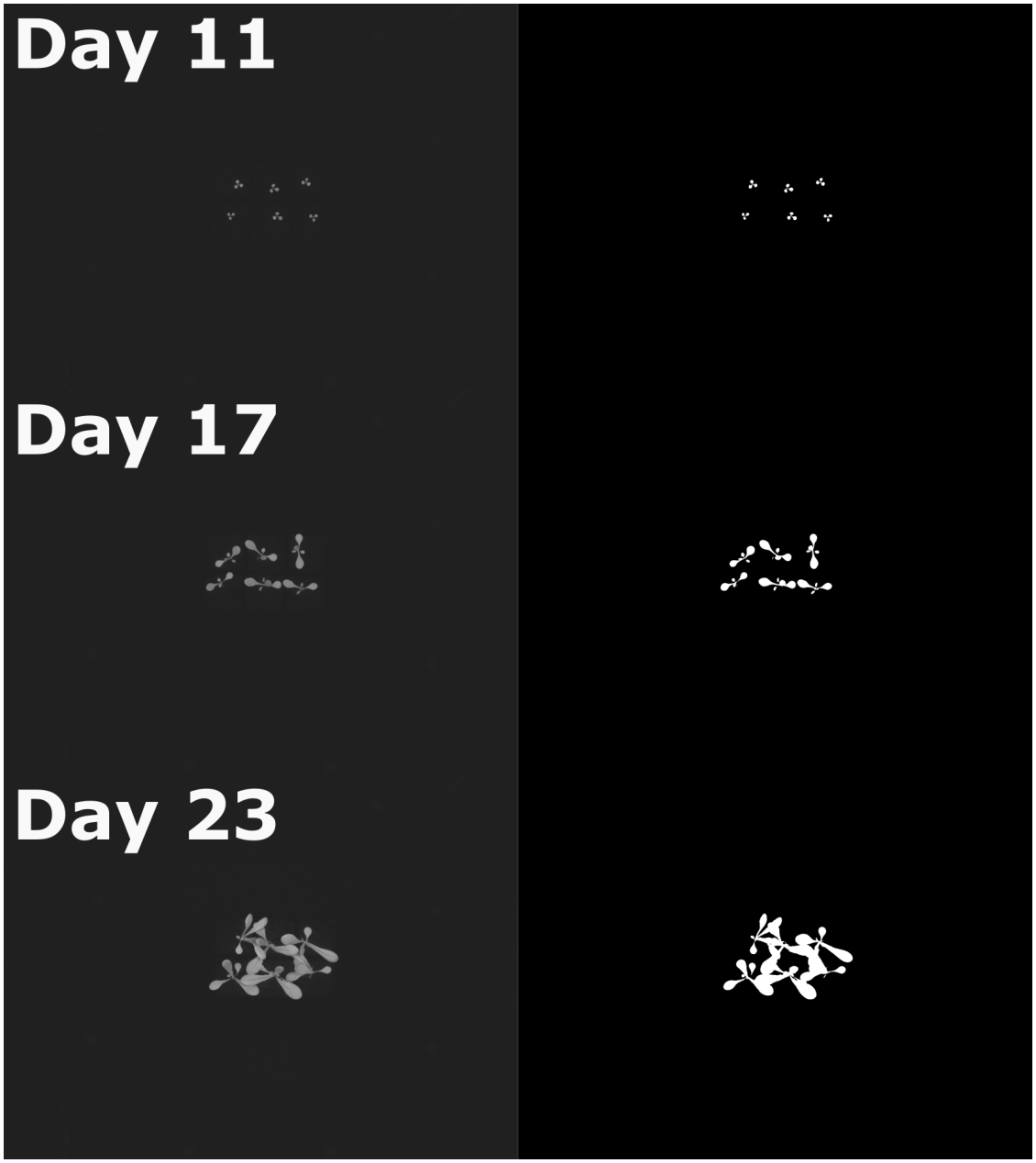

Figure 7.2: Sample images taken by the control imaging device of the low-light plants on days 11, 17, and 23 of experiment 2. The images were processed manually, using the reference Python script.

was observed where manually processed measurements from the new imaging devices were closer to the control measurements. However, for this sample of data, the automatically processed measurements for canopy size did not deviate as much from the manually found measurements of the same images as in the first experiment.

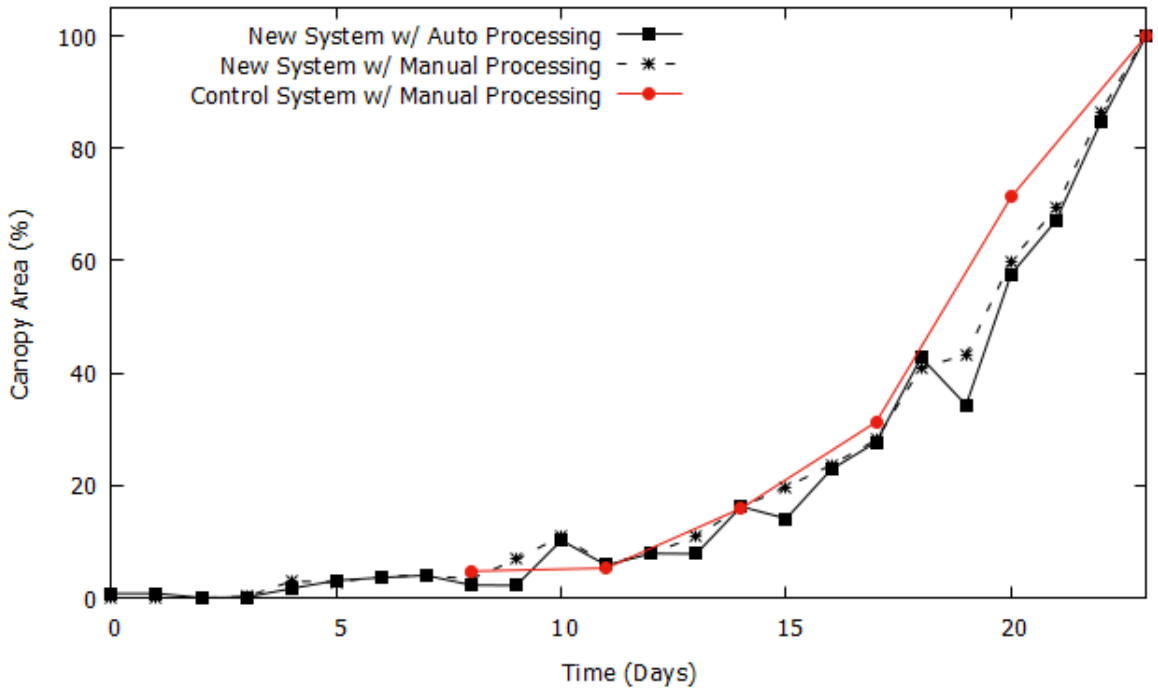

Figure 7.3: A visual comparison of the canopy size measurements calculated from the image sets captured for the low-light grown plants.

With the exception of day-19 of the growth period, the growth percentage measurements found through the new imaging device were fairly accurate relative to the measurements from the control system. The measurements obtained from day-10 were skewed due to having many sprouted seedlings that needed removal, as mentioned before. There was still some deviation in the measurements that were the result of setup constraints, and comparing reference script processing versus automatic processing of the new system's images showed that the automated ImageJ processing proved inferior to the custom Python script that was used manually.

As with the first experiment, the area was calculated for each of the canopy images, using the previous conversion ratios of 0.539 mm^2 per pixel for the control images and 0.082 mm^2 per pixel for my imaging devices. The same method of finding an equation of best fit was performed for the control system, and the resulting area comparison was graphed in Figure 7.4. Based on the graph, the same main trend was observed of increasing deviation with time, as the canopies grew larger. It is assumed that the main cause for this trend also remained the same, being that there was poor image capture of chlorophyll fluorescence. This can be seen in the example images for my imaging device (Fig. 7.1).

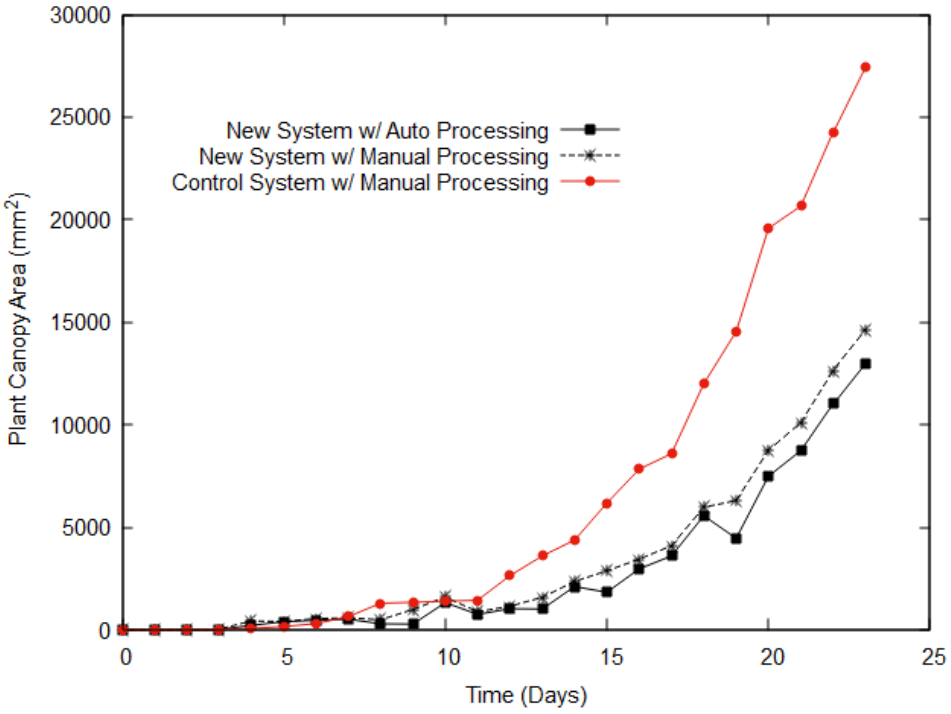

Figure 7.4: A visual comparison of the canopy size measurements calculated from the image sets captured for the low-light grown plants.

For the automated area measurements, an average percent error of 55 percent was calculated. Meanwhile, an average percent error of about 49 percent was found for the reference script area measurements. Like before, only measurements taken after the day the extra seedlings were removed were used in calculating the average percent errors. Compared to the

first experiment, my imaging devices performed slightly better with smaller average percent errors. The better accuracy may be due in part to the plant canopies growing less through the experiment's duration as well as the increase in the camera's exposure setting.

Chapter 8

Discussion

8.1 Positives Outcomes

Through the design of this canopy imaging system, all of the original goals set for it were accomplished along with the addition of some other helpful capabilities. By implementing chlorophyll fluorescence imaging, plant separation through image processing can be done for all plants regardless of coloration. This means that the canopy of red-leaf plants can be imaged and measured without running into conflict with any image background. Careful selection and testing of components allowed for this imaging device to be built for a low cost of approximately 125 USD (Table 4.3) in comparison to the typical thousand USD or more that is needed for custom chlorophyll fluorescence imaging setups. Finally, the system was built to be compact in a way that is easy to mount in most locations which helps with its versatility. This coupled with the low cost makes the system easy to scale up with.

In addition to fulfilling the intended goals, this imaging device also automates the canopy imaging process and provides a high level of customization. Due to utilizing a Raspberry Pi, the device can automate all processes performed by the Raspberry Pi through a program called Crontab which both reduces the amount of manual labor needed to run studies and obtains more data for users. Also, since the imaging pipeline uses fully custom scripts, customization of the settings and pipeline is easy and readily available. Customization includes things such as settings for image capture and processing as well as what pipeline processes are utilized and their ordering. Additionally, hardware parts such as the camera, filter, and even 3d-printed materials can be changed to suit different setups and needs.

In review of the experimental results, my new imaging systems performed adequately in regard to measuring growth percentage but poorly for true area measurements. This means that while the device is able to capture chlorophyll fluorescence accurately enough to depict the growth cycle of plants, the setup used in the experiments did not provide enough chlorophyll fluorescence to fully measure the canopy area. Therefore, the results are promising for the automated imaging device, but further testing and improvements to the design will be needed to prove its viability.

8.2 Canopy Area Measurement Error

During the course of the experiments, this canopy imaging device ran into a couple of problems. The main issue was large percent errors found in the canopy area measurements for both automated processing and manual processing of its images. While the device was still able to measure growth percentage, accurate area measurements proved difficult to obtain since large portions of the plant canopy could not be seen on its images. The main cause of the large measurement deviations was poor chlorophyll fluorescence capture within the setups, partially due to the height restrictions of the experiment site. To remedy this problem, more even excitation light distribution is needed. This can be achieved by either altering the canopy imaging setup so that more light is present and the canopies are fully lit and/or improving the imaging device's design to include its own excitation light such as a LED ring placed around the camera lens. Additionally, since the distance between the devices' cameras and the plants was so small, varying between approximately 7 and 11 inches (18 and 28 cm, respectively), the innate error caused by the growth of plants had a large impact on the canopy area measurements. This is to say that canopy imaging setups where the distance between the plants and imager is short will naturally have greater amounts of error, and increasing the height of the imaging device within its setup will help decrease the canopy area deviation too.

8.3 Image Processing Stalling

A problem encountered during the development of this imaging device was with the ImageJ macro script. Mentioned before, the canopy imaging pipeline would occasionally stall when the ImageJ macro attempted to run its autothreshold function. This stalling would occur randomly without notice, and a week of troubleshooting pointed to a plugin used in ImageJ's autothreshold function. Since there was no way to fix the issue at that point, a workaround was implemented into the device's automated pipeline which was to run the pipeline multiple times to increase the chances of obtaining usable images and data. This solution worked but can become tedious to keep track of and may not be practical in the long run, so resolving the stalling issue is still a priority. The best solution to remove the pipeline stalling would be to remove ImageJ itself and replace it with a different image processing program. For example, the reference image processing script used in the experiments could be automated for this imaging device. Otherwise, installation of the newer ImageJ2 may help to resolve this problem. Do note, however, that the Raspberry Pi only officially supports the original ImageJ, and varying success has been found with compiling the newer ImageJ2 onto the Raspberry Pi.

8.4 Disconnection and Power Outages

During the first experiment, one of the new imaging systems lost connection to the file server for about five days. Fortunately, it was able to reconnect after manually restarting, and all the images taken over those five days were retrieved. During the second experiment, the same system lost connection about halfway through and would not reconnect even after manually restarting several times. It was later found that the system lost power due to a power trip of the electric outlets which the device was connected to. In either case, the connection issues did not seem to be innately a problem of the imaging device but more so that of the particular environment. Moving the wireless access point closer or into the same room as

the devices may resolve any problems with network disconnection while the addition of a backup battery to the device would prevent any issues that could arise from power outages.

8.5 Device Setup

Finally, because the device heavily focused on using custom programs to perform its software processes, especially the automation, there was a steep learning curve for setting up the system as well as utilizing it to suit more specific needs. The process of setting up this imaging device is straightforward but long and tedious due to the numerous programs it uses. Fortunately, much of the installation of software can be bypassed through using a backup image of the Raspberry Pi which can be pre-loaded into a microSD card for the device. Even so, users still must enter the Raspberry Pi, either directly or through SSH, to adjust settings such as its network settings, pipeline variables, and automated task schedule. Otherwise, a clear installation guide of the programs used by the imaging device would aid in the difficulty of setting up this device.

8.6 Future Improvements

While the imaging system can fulfill its purpose of canopy imaging through chlorophyll fluorescence, emphasis must be placed on resolving its issue with area measurements. Mentioned earlier, incorporating the excitation lights into the imaging device would solve issues with insufficient excitation or poor placement of the lighting component. This would require some re-designing of the 3d-printed parts, but that can easily be done. Otherwise, several improvements can be made to it through future testing and upgrading. These improvements come in both its hardware and software. In terms of hardware, the most obvious upgrades would be to the filter and camera quality. As mentioned before, the total costs for the current design of this imaging device is well below the goal price, so there is plenty of leeway in the budget for purchasing higher quality hardware if suitable parts are found.

As for software, the most problematic part of the system is currently ImageJ. As mentioned before, the image processing's stalling is most likely due to a problem with ImageJ's plugins, so the best solution for the imaging device would be to replace ImageJ with a different processing component. For this purpose, the reference image processing script could be automated for the canopy imaging pipeline. While this would require more user input for customization initially, the imaging pipeline would yield better results without any stalling. Otherwise, a different custom program that utilizes an algorithm capable of auto-thresholding could be used for less user input.

To address the possibility of network disconnections, an automated script that is run server-side could help by detecting when there is a missing connection to the imaging device from the network and warning the operator of the missing connection. This solution would not immediately fix the disconnection issue, but it would work toward minimizing any loss of data should an imaging device lose its connection to the wireless network. As for the other main issue experienced during the experiments which was loss of power to the imaging devices, a reasonable solution would be developing a backup battery for the Raspberry Pi so that the devices could switch to the batteries if their power supplies lose power. If possible, it would be ideal for the devices to switch back to AC power should it become re-established and for the backup batteries to recharge at that point too.

If my imaging device were to be developed into a product, several improvements would be needed including the problem solutions mentioned earlier. Additionally, refining of the pipeline such as replacing the image processing would improve the device's reliability. Furthermore, development of a user interface to allow operators to adjust settings and deploy the device more easily. Finally, streamlining the production of the device enclosure would reduce costs and allow for better ergonomics. With these improvements, especially the user interface, expert installation shouldn't be necessary if a user manual can be provided.

Chapter 9

Conclusion

In closing, the work of this exposition should be considered a partial success by achieving the goal of designing and assembling a functional canopy imaging device that utilizes chlorophyll fluorescence to overcome a major issue with normal RGB canopy imaging yet requires further testing and improvement to become a viable option in scientific studies and as a commercial product. Fortunately, due to how modular this imaging device is, improvements and solutions to issues can be implemented easily. By selectively choosing parts that are low in cost and testing their functionality, the imaging device was able to be built for much lower than the original budget estimate of 200 USD while adequately measuring canopy size. Additionally, the use of a Raspberry Pi 4 as the main controller of the system allowed it not only to automate the canopy imaging process but also provide a high level of customization.

Besides the issue with area measurements, the imaging device did encounter some problems with its software. These were mostly contained to the image processing and the wireless network connection. The imaging pipeline would stall during processing with ImageJ, which is an issue that cannot be fixed, so replacing ImageJ with custom processing scripts would be the best option. As for the wireless connection issues, changes to the positioning of the wireless access point of the file server should solve the remaining problems. Once the problems are addressed along with the inaccurate area measurements, the imaging device can move toward more active deployment.

Bibliography

[1] Baker, Neil R., Rosenqvist, Eva. (2004) Applications of chlorophyll fluorescence can improve crop production strategies: an examination of future possibilities. *Journal of Experimental Botany*, Volume 55, Issue 403, Pages 1607–1621. https://doi.org/10.1093/jxb/erh196

[2] Bradbury, M., Baker, NR. (1981) Analysis of the slow phases of the in vivo chlorophyll fluorescence induction curve, Changes in the redox state of photosystem II electron acceptors and fluorescence emission from photosystems I and II. *Biochimica et Biophysica Acta*, 635, Pages 542-551. Cited by Maxwell (2000).

[3] Bumgarner, Natalie R., Miller, Whitney S., Kleinhenz, Matthew D. (2012) Digital Image Analysis to Supplement Direct Measures of Lettuce Biomass. *HotTechnology*, Volume 22, Issue 4, Pages 547-555.

[4] Chander, Subhash et al. (2015) A study on Spectral Response and External Quantum Efficiency of Mono-Crystalline Silicon Solar Cell. *International Journal of Renewable Energy Research*, Volume 5, Issue 1, Pages 41-44.

[5] Chaerle, Laury et al. (2004) Thermal and Chlorophyll-Fluorescence Imaging Distinguish Plant-Pathogen Interactions at an Early Stage. *Plant and Cell Biology*, Volume 45, Issue 7, Pages 887-896. https://doi.org/10.1093/pcp/pch097

[6] Daley, Paul F. (1995) Chlorophyll fluorescence analysis and imaging in plant stress and disease. *Canadian Journal of Plant Pathology*, Volume 17, Issue 2. https://doi.org/10.1080/07060669509500708

[7] Easlon, Hsien Ming, Bloom, Arnold J. (2014) Easy Leaf Area: Automated Digital Image Analysis for Rapid and Accurate Measurement of Leaf Area. *Applications in Plant Sciences*, 2(7). https://doi.org/10.3732/apps.1400033

[8] Elkins, Claudia and van Iersel, Marc W. (2020) Longer Photoperiods with the Same Daily Light Integral Improve Growth of Rudbeckia Seedlings in a Greenhouse. *HortScience*, Volume 55, Issue 10, Pages 1676-1682. https://doi.org/10.21273/HORTSCI15200-20

[9] Govindjee. Chlorophyll a fluorescence: A bit of Basics and History. In: Papageorgiou GC, Govindjee (Editors). Chlorophyll a fluorescence. A signature of photosynthesis. Dordrecht, The Netherlands: Springer (2004)

[10] Herritt, Matthew T. et al. (2021) FLIP: FLuorescence Imaging Pipeline for field-based chlorophyll fluorescence images. *SoftwareX*, Volume 14, Article 100685. https://doi.org/10.1016/j.softx.2021.100685

[11] Jayalath, Theekchana C. and van Iersel, Marc W. (2021) Canopy Size and Light Use Efficiency Explain Growth Differences between Lettuce and Mizuna in Vertical Farms. *Plants*, 10(4), 704. https://doi.org/10.3390/plants10040704

[12] Jones, Hamlyn G. et al. (2009) Thermal infrared imaging of crop canopies for the remote diagnosis and quantification of plant response to water stress in the field. *Functional Plant Biology*, 36(11), 978-989. https://doi.org/10.1071/FP09123

[13] Kautsky, H., Appel, W., Amann, H. (1960) Chlorophyllfluorescenz und kohlensaureassimilation. *Biochemische Zeitschrift*, 322, Pages 277-292. Cited by Maxwell (2000).

[14] Krause, G. H. and Weis, E. (1991) Chlorophyll Fluorescence and Photosynthesis: The Basics. *Annual Review of Plant Physiology and Plant Molecular Biology*, 42:313-49.

[15] Krause, G.H. and Weis, E. (1984) Chlorophyll fluorescence as a tool in plant physiology. *Photosynth Res 5*, Pages 139–157. https://doi.org/10.1007/BF00028527

[16] Maxwell, Kate, Johnson, Giles N. (2000) Chlorophyll fluorescence—a practical guide. *Journal of Experimental Botany*, Volume 51, Issue 345, Pages 659–668. https://doi.org/10.1093/jexbot/51.345.659

[17] Osroosh, Yasin, Khot, Lav R., Peters, Troy R. (2018) Economical thermal-RGB imaging system for monitoring agricultural crops. *Computers and Electronics in Agriculture*, Volume 147, Pages 34-43. https://doi.org/10.1016/j.compag.2018.02.018

[18] Quick, W.P., Horton, P. (1984) Studies on the induction of chlorophyll fluorescence in barley protoplasts. I. Factors affecting the observation of oscillations in the yield of chlorophyll fluorescence and the rate of oxygen evolution. *Proceedings of the Royal Society of London B*, 220, 361-370. Cited by Maxwell (2000).

[19] Xiong, Yedan et al. (2019) Digital Image Analysis of Old World Bluestem Cover to Estimate Canopy Development. *Agronomy Journal*, Volume 111, Issue 3, Pages 1247-1253. https://doi.org/10.2134/agronj2018.08.0502

Appendix A

RPi NoIR Camera Test Script

```
01 #Fluorescence Imaging for Raspberry Pi Camera
02
03 import RPi.GPIO as GPIO    #Import the GPIO library
04 import time    #Import the time library
05 from picamera import PiCamera    #Import the camera library
06
07 GPIO.setmode(GPIO.BOARD)    #Set pin name mode
08 #GPIO.setup(32, GPIO.OUT)
09 #pwmAct = GPIO.PWM(32, 1000)    #Set pin 32 to pwm for actinic led
10 GPIO.setup(33, GPIO.OUT)
11 ledEx = 33    #Set pin 33 to pwm for excitation led
12
13 camera = PiCamera(resolution=(1280,720), framerate=30)    #Set up the
   raspbery pi camera
14
15
16 #Main Program
17 camera.iso = 100    #Set the ISO/sensitivity to desired value. 100 - 800
18 time.sleep(2)    #Wait for the automatic gain control to settle
19 #print(camera.exposure_speed)
20 camera.shutter_speed = 10000    #Set the camera shutter speed in microseconds
21 camera.exposure_mode = 'off'    #Set auto exposure to off
22 g = camera.awb_gains
23 camera.awb_mode = 'off'    #Set auto white balance to off
24 camera.awb_gains = g    #Restore auto white balance gains
25
26 #dc = 50    #Set the desired duty cycle of pwm pins
27 #pwmAct.start(dc)    #Start actinic led at duty cycle
28 #time.sleep(600)    #Wait 10 minutes for plant to react
29
30 #camera.capture('Actinic.jpg')    #Capture actinic image
31
32 GPIO.output(ledEx, GPIO.HIGH)    #Start excitation led at duty cycle
33
```

```
34 camera.capture('Excitation.jpg')    #Capture excitation image
35
36 GPIO.output(ledEx, GPIO.LOW)    #Stop excitation led
37 #pwmAct.stop()    #Stop actinic led
38 GPIO.cleanup()    #Reset pins
39
40 print("DONE")    #Signal completion of program
```

Appendix B

Final ArduCam Python Image Capture Script

```
01 import arducam_mipicamera as arducam
02 import v4l2 # sudo pip install v4l2
03 import RPi.GPIO as GPIO
04 import time
05
06 GPIO.setmode(GPIO.BOARD)
07 #GPIO.setup(32, GPIO.OUT)
08 #pwmAct = GPIO.PWM(32, 1000)
09 GPIO.setup(33, GPIO.OUT)
10 pwmEx = 33
11
12 def set_controls(camera):
13     try:
14         print("Setting the Exposure...")
15         camera.set_control(v4l2.V4L2_CID_EXPOSURE, 2500)
16         print("Setting the Gain...")
17         camera.set_control(v4l2.V4L2_CID_GAIN, 3) # Range 0-15
18         print("Disable Auto Exposure...")
19         camera.software_auto_exposure(enable = False)
20
21     except Exception as e:
22         print(e)
23
24 #dc = 50
25
26 if __name__ == "__main__":
27     try:
28         camera = arducam.mipi_camera()
29         print("Open camera...")
30         camera.init_camera()
31         #print("Setting the resolution...")
32         #fmt = camera.set_resolution(1920, 1080)
33         #print("Current resolution is {}".format(fmt))
34         #print("Start preview...")
```

```
35          #camera.start_preview(fullscreen = False, window = (0, 0, 1280, 720))
36          set_controls(camera)
37
38          time.sleep(1)
39
40          #pwmAct.start(dc)
41          #time.sleep(10)
42
43          #frame = camera.capture(encoding = 'jpeg')
44          #frame.as_array.tofile("actinic.jpg")
45          #set_controls(camera)
46
47          GPIO.output(pwmEx, GPIO.HIGH)
48
49          time.sleep(1)
50
51          #print("Current resolution is {}".format(fmt))
52          frame = camera.capture(encoding = 'jpeg')
53          frame.as_array.tofile("cf_image.jpg")
54
55          time.sleep(1)
56
57          GPIO.output(pwmEx, GPIO.LOW)
58          #pwmAct.stop()
59
60          #Release memory
61          del frame
62          #print("Stop preview...")
63          #camera.stop_preview()
64          print("Close camera...")
65          camera.close_camera()
66          GPIO.cleanup()
67          print("DONE")
68
69      except Exception as e:
70          print(e)
```

Appendix C

ImageJ Macro Script

```
01 args = split(getArgument()," ");
02 print("Starting processing...");
03 run("16-bit");
04 run("Despeckle");
05 //run("Threshold...");
06 setAutoThreshold("Intermodes dark");
07 call("ij.plugin.frame.ThresholdAdjuster.setMode", "B&W");
08 setOption("BlackBackground", true);
09 run("Convert to Mask");
10 print("Image threshold done...");
11 run("Despeckle");
12
13 print("Saving processed image...");
14 saveAs("Jpeg", args[0] + ".processed.jpg");
15
16 if (args[1] == "multiple") {
17   print("Multiple plants...");
18   //run("Invert");
19   run("Analyze Particles...", "size=200-Infinity circularity=0.00-1.00
     show=Outlines display clear");
20   saveAs("Jpeg", args[0] + ".multiple.jpg");
21   close();
22   saveAs("Results", args[0] + ".csv");
23 }
24 if (args[1] == "single") {
25   print("Single plant...");
26   run("Create Selection");
27   run("Measure");
28   saveAs("Results", args[0] + ".csv");
29 }
30
31 if (isOpen("Results")) {
32   selectWindow("Results");
33   run("Close");
```

```
34 }
35 if (isOpen("Log")) {
36   selectWindow("Log");
37   run("Close" );
38 }
39
40 close("*");
41 print("DONE");
42 exit();
```

Appendix D

Automated Pipeline Shell Script

```
01 #!/bin/bash
02
03 # variables
04 RCLONE="/home/pi/rclone" # Local Rclone directory for cloud sync
05 ONEDRIVE="<onedrive_name>:<path/to/onedrive/directory>" # OneDrive directory
   on cloud storage space
06 SAMBA="/home/pi/smbshare/<subdirectory>" # Local SambaShare/CIFS directory
07 TYPE="single" # Type of analysis ('single' or 'multiple')
08
09 # Set specific system name and current date
10 NAME=<nameofsystem>
11 DATE=$(date +%F)
12
13 # Load in current files from the cloud server
14 rclone sync -v $ONEDRIVE $RCLONE/
15
16 sleep 1s
17
18 # Run the imaging script
19 python /home/pi/MIPI_Camera/RPI/python/cf_capture.py
20
21 sleep 1s
22
23 # Move the new image into the processing folder
24 mv /home/pi/cf_image.jpg /home/pi/CF_Images/cf_image.jpg
25
26 # Process the raw image with the ImageJ macro
27 export DISPLAY=:1
28 imagej -x 1000 -i /home/pi/CF_Images/cf_image.jpg -b /home/pi/CF_Imaging.ijm
   "/home/pi/CF_Images/$NAME_$DATE $TYPE"
29
30 sleep 5s
31
32 # Move the processed (and unprocessed) image(s) to the synced folders
```

```
33 mv /home/pi/CF_Images/cf_image.jpg /home/pi/CF_Images/$NAME_$DATE.jpg
34 cp /home/pi/CF_Images/* $SAMBA/
35 mv /home/pi/CF_Images/* $RCLONE/
36
37 sleep 1s
38
39 # Upload the new files back to the cloud server
40 rclone sync -v $RCLONE/ $ONEDRIVE
```