ROBOTIC SYSTEMS FOR FIELD-BASED HIGH-THROUGHPUT PLANT PHENOTYPING

by

Rui Xu

(Under the Direction of Changying Li)

Abstract

The global population is predicted to reach 9 billion by 2050, which requires the current food production to double to meet the global demand for food, feed, fiber, and bioenergy. This is a tall order and brings challenges to plant breeders to find genotypes with high yield and high-stress tolerance to adapt to the changing climate in the next 30 years. High-throughput phenotyping that uses modern imaging and sensing technologies to accelerate the breeding of specific crop genotypes is a promising way to solve the challenges. This dissertation focused on developing ground and aerial robotic systems for field-based high-throughput phenotyping and developing novel data processing methods to measure phenotypical traits.

In this study, an unmanned aerial system that integrated color, multispectral, thermal cameras, and LiDAR sensor was developed to measure phenotypical traits at the plot level. A data processing pipeline was developed to extract phenotypical traits from the raw data,

including canopy height, canopy cover, canopy volume, canopy vegetation index, and canopy temperature. The aerial system was also used to detect and count cotton blooms using the proposed novel bloom counting algorithm that uses Structure from Motion and Convolutional Neural Network. The unmanned aerial system and data processing methods can be effective and efficient tools for field-based high-throughput phenotyping.

A modular agricultural robotic system (MARS) was developed, which consists of several modules. Different combinations of modules can form robot configurations for different purposes. The software was developed based on the Robot Operating System (ROS). The robot can auto navigate in the field, and several field tests showed the robot's usefulness in high throughput phenotyping. MARS robots can be easily adapted to different agricultural tasks and affordable and effective platforms for researchers and growers.

The next generation Berry Impact Recording Device (BIRD Next) is an upgrade of the previous design. The sensor was designed to measure the mechanical impacts of small fruits and vegetables. The sensing range and frequency of the BIRD Next were significantly improved than the previous design. It integrates wireless communication (Bluetooth) and wireless charging, making the sensor waterproof and useable for produces whose processing involves water.

INDEX WORDS: High-throughput phenotyping, Unmanned aerial vehicle, Remote sensing, Convolutional neural network, Agricultural robot, Berry impact recording device

ROBOTIC SYSTEMS FOR FIELD-BASED HIGH-THROUGHPUT PLANT PHENOTYPING

by

RUI XU

B.S., China Agricultural University, China, 2011

M.S., University of Georgia, 2015

A Dissertation Submitted to the Graduate Faculty of the

University of Georgia in Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

ATHENS, GEORGIA

2021

ROBOTIC SYSTEMS FOR FIELD-BASED HIGH-THROUGHPUT PLANT PHENOTYPING

by

Rui Xu

Major Professor: Changying Li

Committee: Javad Mohammadpour
Glen C Rains
Sergio Bernardes

Electronic Version Approved:

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

DEDICATION

To my parents, friends and people who have helped and supported me.

Acknowledgements

I would like to express my sincere appreciation to my advisor Dr. Changying Li for his mentoring and support throughout my Ph.D. study. Without his guidance and persistent help, this dissertation would not have been possible.

I would like to thank my committee members, Dr. Javad Mohammadpour, Dr. Glen C. Rains, and Dr. Sergio Bernardes, for their guidance and support. I also would like to extend my gratitude to my collaborators: Dr. Fumiomi Takeda and Dr. Andrew H. Paterson, for their assistance in collaborative studies. Besides, I would like to thank my friends and lab colleagues, including Ms. Xueping Ni, Dr. Weilin Wang, Dr. Yu Jiang, Ms. Lin Qi, Dr. Shangpeng Sun, Dr. Yi Fang, Dr. Mengyun Zhang, Mr. Zikai Wei, Dr. Ruoyu Zhang, Mr. Jon Roberson, Mr. Rikki Brown, Mr. Tsunghan Han, Mr. Jawad Iqbal, Mr. Javier Rodriguez Sanchez, Mr. Robby Ratajczak, Mr. Jessie Kuzy and Mr. Joshua Griffin. Their accompany and assistance help me achieve one of the most important milestones in my life. I would also like to acknowledge the service and support from staff and colleagues in the College of Engineering and Graduate School during my study at the University of Georgia.

Lastly, I would like to thanks my parents, who are always supporting me and helping me. I am deeply indebted to their love and care.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The global population is estimated to reach nearly 10 billion by 2050, which requires sustainably doubling the agricultural output from 2005 to 2050 to fulfill the global need for food, feed, fiber, and bioenergy in 2050 [1]. Consequently, global agricultural productivity must grow by an average rate of 1.73% annually from 2010 to 2050 [2]. However, global agricultural productivity has only been rising by an average annual rate of 1.63%, below its target value. More seriously, modern agriculture also faces challenges from climate change, resource depletion and labor shortage, making it even challenging to increase agricultural productivity in a sustainable way.

Several potential solutions have been proposed and attempted to address these issues. The first solution is precision agriculture, which aims to improve crop management by minimizing the agricultural input (e.g., water, fertilizer and pesticide) and maximizing the output (e.g.,

yield and quality). This requires continuously monitoring crop growth to support management decisions. The second solution is plant genome engineering and phenotyping, which aims to breed new crop cultivars that can naturally produce high yield, better quality and tolerate various environments (e.g., saline soil and drought). Breeding such cultivars require measuring plant phenotypical traits at a large scale to understand the interaction between the genotype and phenotypes in dynamic environments, which is the primary goal of high-throughput phenotyping. Both solutions need an automated way to evaluate crops in the field in a high-throughput manner, which makes it is essential to develop innovative technologies for such purposes.

Over the past decade, field-based high-throughput phenotyping (FHTP) has made considerable progress in evaluating plant phenotypes in the field. Many field-based high-throughput phenotyping platforms (FHTPPs) have been developed and used in breeding programs [3, 4]. However, one major limitation of those FHTPPs is that they require human intervention fully or partially, and thus their throughput is limited and not practical for large fields. Therefore, it is necessary to develop fully automated robotic systems for FHTP to further increase the throughput.

1.2 Objectives

This dissertation aims to develop robotic systems and data analytical methods for field-based high-throughput phenotyping, and to develop the next generation Berry Impact Recording Device (BIRD Next). Specific objectives include:

1. Develop a multi-sensor unmanned aerial system for field-based high-throughput plant phenotyping;

2. Develop a data processing pipeline to extract phenotypical traits at plot-level from aerial data;

3. Develop an image processing method to count cotton blooms using aerial color images;

4. Develop a modular agricultural robotic system for ground data collection;

5. Develop the next generation Berry Impact Recording Device (BIRD Next) to integrate wireless communication and wireless charging.

1.3 Overview of the Dissertation Chapters

This dissertation contains eight chapters, where chapters 3 to 6 represent four manuscripts written individually and remain in a general manuscript format. Chapter 1 introduces the significance of this study and defines the objectives of this dissertation. Chapter 2 reviews the current development of ground phenotyping robots.

Chapter 3 describes the development of a multi-sensor unmanned aerial system for field-based high-throughput plant phenotyping and a data processing pipeline to extract plot-level phenotypical traits, including morphological traits (canopy height and volume), canopy vegetation indices, and canopy temperature. The system was validated in the field.

Chapter 4 explores the application of aerial multispectral imaging for cotton phenotyping. A data processing algorithm was developed to extract cotton plant height, and a preliminary study on cotton bloom detection using the multispectral image was conducted.

Chapter 5 develops a novel image processing algorithm to localize and count cotton blooms from aerial color images. The algorithm uses a convolutional neural network for cotton bloom detection.

Chapter 6 reports the development of a modular agricultural robotic system. A design concept was proposed, and two types of robots were implemented based on the design concept. The performance of the robots was tested in the field.

Chapter 7 introduces the hardware and software design of the BIRD Next.

Chapter 8 provides general conclusions and future research directions.

Chapter 2

Review of Ground Field-Based High-Throughput Phenotyping Systems: Focus on Phenotyping Robots

2.1 Introduction

Plant phenotyping is an emerging science that links genomics with plant ecophysiology and agronomy. The assessment of plant phenotypes in the field can be labor-intensive and inefficient. The emergence of Field-based High-throughput Phenotyping (FHTP) is to increase the throughput through sensing technologies and data processing algorithms. The FHTP

systems integrate sensors with mobile platforms to collect data in the field with minimal or no human intervention.

Many FHTP systems have been developed so far, including aerial and ground systems. While aerial systems can provide much higher efficiency and coverage than ground systems, ground systems are used primarily to collect high-resolution data for measuring phenotypical traits at fine levels, such as the plant and organ level. Early ground FHTP systems were based on tractors, and later pushcarts were developed to replace tractors. The recent trend is to use phenotyping robots to automate the data collection.

Most agricultural environments are unstructured that can change rapidly in time and space; therefore, a phenotyping robot needs to be intelligent to operate by itself. A robot's intelligence follows a sensing-thinking-acting cycle where the robot needs to understand its surrounding environment, make decisions based on the environment and its states in the environment, and perform certain operations to achieve its goals. For phenotyping robots, the intelligence mainly focuses on navigation in unstructured environments because the phenotyping robot's primary mission is to collect data autonomously in the field. A typical phenotyping robot primarily consists of the mobile platform, sensors including phenotyping sensors for measuring phenotypical traits and perception sensors for navigation, and computing units for data collection and robot navigation (Figure 2.1). Manipulators, such as robotic arms, are sometimes used in phenotyping robots to measure certain phenotypical traits.

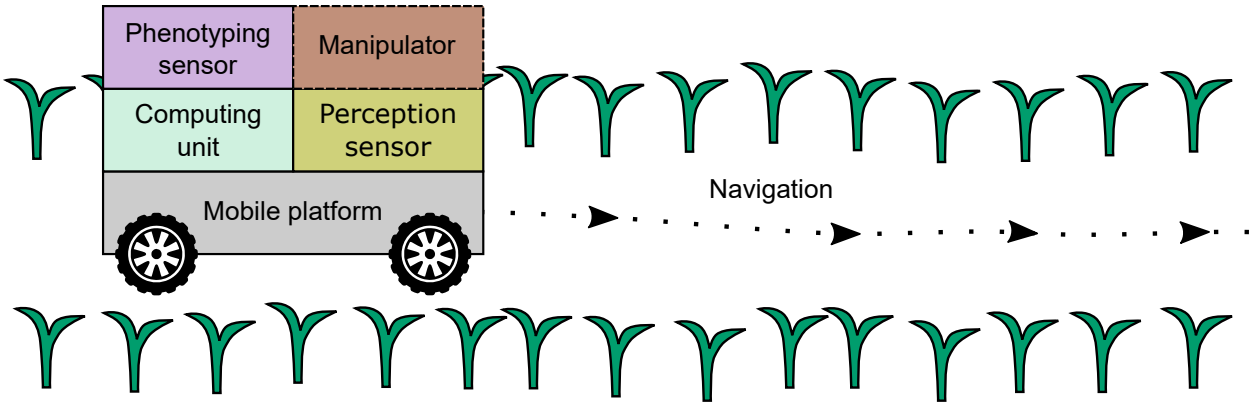

Figure 2.1: Diagram of a phenotyping robot.

This chapter's main objective is to review the existing ground phenotyping robots for FHTP to provide readers with a general understanding of the current development progress of the phenotyping robot. First, this chapter first provides a brief review of the ground phenotyping systems using nonrobotic systems and then a detailed review of ground phenotyping robots from the view of the robot's main components, including mobile platforms, sensors, manipulators, computing units, and software. Second, it reviewed the navigation algorithms and simulation tools developed for phenotyping robots and the applications of phenotyping robot. Last, this chapter discusses current limitations and challenges and concludes with directions for future research.

2.2 Ground Field-based Phenotyping Systems

The earliest developed ground FHTP systems developed were based primarily on tractors because of their wide availability and ease in modification for mounting sensors. However, the soil compaction created by tractors makes them unsuitable for frequent data collection.

Therefore, a lightweight pushcart, or motorized cart, was developed to replace tractors. Since tractors and pushcarts still need manual control, the gantry and cable-driven platforms were developed to automate data collection fully. However, gantry and cable-driven platforms are not mobile and can only cover certain fields, limiting the number of experimental plots. Their high construction and maintenance costs also limit their usage. Table 2.1 lists the ground FHTP systems in literature.

Table 2.1: Summary of the ground field-based high-throughput phenotyping systems

System	**Platform**	**Sensors**	**Crop**	**Phenotypical traits**
BreedVision [5]	Tractor	3D camera	Triticale	Plant height
		Light curtain		Canopy reflectance
		Laser distance sensor		
		Hyperspectral camera		
		Color camera		
[6]		Infrared temperature sensor	Cotton	Canopy height
		Ultrasonic sensor		Canopy temperature
		Spectral reflectance sensor		NDVI
[7]		Ultrasonic Sensor	Cotton	Plant height
		Spectral reflectance sensor		Ground cover fraction
		Color camera		NDVI
		Infrared thermometer		Canopy temperature
[8]		Infrared thermometer	Soybean	Canopy temperature
		Ultrasonic sensor	Wheat	Canopy height
		Laser distance sensor		Vegetation index
		Spectral reflectance sensor		
Phenoliner [9]		Color camera	Grape	Plant count
		NIR camera		
		Thermal camera		
		Hyperspectral camera		
ProTractor [10]		Color camera	Brassica	Seedling count

System	Platform	Sensors	Crop	Phenotypical traits
GPhenoVision [11]		RGBD camera Thermal camera Hyperspectral camera	Cotton	Plant height Projected leaf area Canopy volume Canopy temperature Width in-row Width across-row
[8]	Pushcart	Ultrasonic sensor NDVI sensor Infrared thermometer Spectrometer Color camera	Soybean Wheat	Canopy height NDVI NDRE Canopy temperature Green pixel fraction
Phenocart [12]		Spectral reflectance sensor Color camera Infrared thermometer	Wheat	Canopy temperature NDVI
Proximal Sensing Cart [13]		Ultrasonic sensor Infrared thermometer Spectral reflectance sensor Color camera	Cotton	Canopy height Canopy temperature NDVI Canopy cover Crop water stress index Leaf area index
Phenocart [14]		RGB camerea NIR camera	Wheat	Biomass NDVI
[15]		Hypersecptral camera	Tobacco	
Professor [16]	Motorized pushcart	Not specified	Wheat Maize	

System	Platform	Sensors	Crop	Phenotypical traits
Field Scanalyzer [17]	Gantry	Thermal camera	Wheat	Plant morphology
		Chlorophyll fluorescence imager		Canopy temperature
		3D Laser Scanner		Spectral indices
		Color camera		
		Hyperspectral camera		
PhénoField [18]		LiDAR sensor	Wheat	Green cover fractions
		Spectroradiometer		Green Area Index
		Color camera		Average Leaf Angle
				Meris Terrestrial Chlorophyll Index
				Plant height
Field Phenotyping	Cable system	Color camera	Wheat	Enhanced NDVI
Platform [19]		NIR camera		Canopy cover
		Laser scanner		Canopy height
		Thermal camera		Canopy temperature
		Ultrasonic sensor		Canopy spectral reflectance
		Spectrometer		
		Multispectral camera		
NU-Spidercam [20]		Multispectral camera	Maize	Canopy cover
		Thermal camera	Soybean	NDVI
		Spectrometer		Canopy temperature
		3D LiDAR		Canopy height
				Canopy reflectance

2.2.1 Tractor based systems

Early tractor based FHTP systems were modified from high-clearance tractors mounted with multiple sensors to measure plant phenotypical traits. One well-known system was developed by USDA Maricopa Agricultural Center at Maricopa, Arizona. Their system had eight sets of infrared-thermometer and ultrasonic sensors that measure the temperature and height of the canopy of eight rows at a single pass [6]. An Real-time Kinematic Global Navigation Satellite System (RTK-GNSS) was mounted onto the tractor to georeference measurements. Another representative system is BreedVision, which used a tractor with an enclosure to carry multiple sensors [5]. The BreedVision integrated a 3D time-of-flight camera, a light curtain sensor, a laser distance sensor, a hyperspectral camera, and a color camera. As imaging technologies advanced, the later developed FHTP platforms mainly used imaging sensors to collect data. For example, the GPhenoVision is a tractor-based multi-sensor system that integrated a hyperspectral, thermal, and RGBD camera [11]. The GPhenoVision system has been used to measure morphological traits for cotton [21].

The high payload capacity of the tractor-based FHTP system eases the carrying of heavy equipment, such as the imaging chamber, so that the environmental conditions can be controlled partly for data collection. However, the tractor's heavy weight can create soil compaction with frequent data collection, which could interrupt the crop's growth. Additionally, the data acquisition system can suffer from the tractor's vibrations, which potentially could damage the sensors and affect the data quality if the vibrations are not properly isolated. The field soil conditions (such as muddy soil after rain) could limit the operation of the tractor.

Because tractors need to be driven manually, the lack of precise controls for the tractor's speed and trajectory can affect specific sensors, such as the push-broom hyperspectral camera. Furthermore, there is the potential risk for damaging the crops with manual driving, but this is a common issue for manually-controlled, ground FHTP systems.

2.2.2 Pushcart and motorized cart

As an alternative to the tractor, pushcart can be assembled easily with low-cost materials. It was developed to solve the soil compaction issue of the tractor. Most cart-type systems were made of metal frames with bicycle wheels, which makes them low cost and lightweight [8, 12, 14, 13, 16, 15]. The frame structure eases the mounting of sensors. Since the pushcart is activated manually, its position is easier to control than that of tractors such that it can stop at any position and scan the entire field [15]. The motorized cart uses electronic motors to move the cart, meaning it can be controlled remotely. Professor is a platform that uses two DC motors for driving and two DC motors for steering [16]. Its frame is made of aluminum extrusions and its width and height can be adjusted with an inner frame. It is manually-controlled with a remote controller. The major drawback of the pushcart and motorized cart is that they requires manual power and manual control, which is inefficient and not practical for large field.

2.2.3 Gantry and cable-driven system

A gantry is an overhead, bridge-like structure that supports equipment such as a crane. The gantry-based FHTP system uses a gantry to carry sensors and can move linearly on parallel

rails. The camera can move along the gantry bridge as well as vertically, making the camera move in XYZ axes. One well-known gantry-based system is the Field Scanalyzer that was developed by LemnaTec [17]. The high payload (500 kg) enables the system to carry heavy sensors, such as the chlorophyll fluorescence imager (120 kg). PhénoField is a gantry system managed by the applied research institute ARVALIS in France that was used primary for wheat breeding [18]. It has a mobile rainout shelter equipped with irrigation booms to control the water stress for desired plots.

Like the gantry-based system, the cable-driven system is a fixed-site system where the sensors were suspended from cables supported by towers at the outside corners of the field. The movements of the sensors were driven by cable winches. Some representative systems include the Field Phenotyping Platform (FIP) at the Swiss Federal Institute of Technology in Zurich [19] and the NU-Spidercam from the University of Nebraska-Lincoln [20].

The advantages of gantry-based and cable-driven platforms are that they do not make physical contact with the soil or plants in addition to being fully autonomous. The primary disadvantage is high construction and maintenance cost, which can limit their usage in breeding programs. Other disadvantages include limited field coverage and a fixed experimental site.

2.3 Phenotyping Robot

2.3.1 Mobile platform

The development of agricultural robots has advanced dramatically in the past decade to address labor shortages in agricultural. The advantages of automation make the agricultural robot a promising means for managing large farms with minimal human labor and make it an ideal solution for FHTP. Thus, many phenotyping robots have been developed to replace the tractor and pushcart (Figure 2.2).

A phenotyping robot can be classified into four categories based on the driving mechanism: wheeled robot, tracked robot, legged robot, and wheel-legged robot. The wheeled robot uses wheels to drive the robot while the tracked robot uses tracks. These are the two most used forms. The legged robot uses articulated legs to provide locomotion, such as a hexapod robot [22]. The wheel-legged robot combines the wheels with articulated legs, so it has more control of locomotion. Table 2.2 summarizes some developed phenotyping robots mentioned in the literature.

Figure 2.2: Phenotyping robots. A) TerraSentia [23]. B) Vinobot [24]. C) RobHortic [25]. D) Shrimp [26]. E) Robotanist [27]. F) Ladybird [28]. G) A robot based on LT2 [29]. H) VinBot [30]. I) MARIA [31]. J) Thorvald II [32]. K) AgBotII [33]. L) BoniRob [34]. M) Armadillo Scout [35]. N) PHENObot [36]. O) Phenobot 3.0 [37]. P) Phenobot 1.0 [38]. Q) TERRA-MEPP [39]. R) Flex-Ro [40]

.

Table 2.2: Summary of phenotyping robots. 4WD4WS: four-wheel driving and four-wheel steering. 2WD2WS: two-wheel driving and two-wheel steering.

Robot	**Drive Mechanism**	**Phenotyping Sensor**	**Perception Sensor**	**Computing Unit**	**Crop**	**Study**
VinBot [30]	Skid steering	RGBD camera 2D LiDAR	RTK-GNSS IMU 2D LiDAR	Not specified	Vineyard	[41, 30]
Shrimp [26]		3D LiDAR Color camera Hyperspectral camera	RTK-GNSS IMU	Not specified	Orchard	[26, 42, 43]
Robotanist [27]		Stereo camera Color camera	RTK-GNSS IMU 2D LiDAR Stereo camera	Intel NUC	Sorghum	[44]
Vinobot [24]		Trinocular camera	GPS LiDAR	PC	Maize Sorghum	[45]
TerraSentia [23]		Multispectral camera 2D LiDAR Hyperspectral camera Color camera RGBD camera	GNSS Gyroscope	Raspberry Pi Jetson TX2	Maize	[46, 47] [48]
MARIA [31]		2D LiDAR RGBD camera	RTK-GNSS IMU	Jetson Nano	Not specified	[45]

Robot	Drive Mechanism	Phenotyping Sensor	Perception Sensor	Computing Unit	Crop	Study
RobHortic [25]	Differential drive	Thermal camera Multispectral camera Color camera NIR camera Hyperspectral camera	RTK-GNSS	PC	Horticultural crop	
AgBotII [33]	2WD2WS	Color camera	RTK-GNSS	PC	Row crop	[49]
Phenobot 1.0 [38]		Stereo camera	RTK-GNSS	Laptop	Sorghum	[38, 50]
Phenobot 3.0 [37]	Articulated steering	Stereo camera	RTK-GNSS	Not specified	Sorghum	
Thorvald II [32]	Configuration depended	Application depended	GPS IMU 2D LiDAR	Computer	Row crop Orchard Greenhouse	[32, 51] [52, 53]
Ladybird [28]	4WD4WS	Color camera 2D LiDAR Hyperspectral camera Stereo camera Thermal camera	2D LiDAR RTK-GNSS IMU	Industrial computer	Row crop	[54]
AgRover [55]		Not specified	RTK-GNSS	Not specified	Row crop	
Flex-Ro [40]		Color camera Ultrasonic sensor Infrared thermometer Spectrometer	GNSS Obstacle detector	Laptop	Row crop	
Armadillo Scout [35]	Tracked	Application depended	GNSS 2D LiDAR	Single board computer	Not specified	
PHENObot [36]		Color camera	RTK-GNSS	Industrial computer	Vineyard	[56, 57]

Robot	**Drive Mechanism**	**Phenotyping Sensor**	**Perception Sensor**	**Computing Unit**	**Crop**	**Study**
TERRA-MEPP [39]		Stereo camera Depth camera Color camera	RTK-GNSS Wheel encoder Gyroscope	Intel NUC MyRIO	Sorghum	
[58]		Color camera	RTK-GNSS	Raspberry Pi 3	Soybean	
[29]		Color camera		Raspberry Pi 3		
BoniRob [34]	Wheel-legged	Application depended	RTK-GNSS Inertical sensor 3D LiDAR	Embbed PC		[59, 60] [61, 62]

Wheeled robot

The wheeled robot is the most common type of phenotyping robot. Wheeled robots can be classified broadly into two categories: robots with locally-restricted mobility (such as differential drive robots) and robots with full mobility (such as omnidirectional robots). Differential drive robots, such as skid-steering robots, are the most common robots used for phenotyping because of their simplicity in mechanical structure and motion control. For example, the VinBot [30], Robotanist [27], Vinobot [24], RobHortic [25], Shrimp [26], and TerraSentia [23] are differential drive robots. The locomotion of the differential drive robot is controlled by the forward/backward and turning speeds of the wheels. It can achieve in-place rotation. Commercial robotic platforms, such as Jackal and Husky from Clearpath, are differential drive robots commonly used for FHTP [24].

Some wheeled robots use front wheels for steering and back wheels for driving. The locomotion is controlled by the forward/backward speed and turning angle, similar to an automobile. One example is the Phenobot 1.0, which was modified from a small tractor [50]. Its redesigned version, Phenobot 3.0, uses articulated steering [37]. The AgBotII robot is also a front-wheel steering and back-wheel-driveing robot, but was designed mainly for weed management [33].

Unlike the above-mentioned robots with restricted locomotion , omnidirectional robots can move in any direction without restrictions, enabling extra maneuverability and terrain adaptation. The Thorvald II [32] and Ladybird [28] are two representative four-wheel drive and four-wheel steering (4WD4WS) robots. The Ladybird has a liftable cover with solar

panels that extend the operation time. The Thorvald II is a modular robotic system that consists of several robot modules that can be reconfigured into different robots other than the 4WD4WS robot, such as a 2WD0WS robot. The Thorvald II robot also included a suspension module that can make the robot run on rough terrain more stably. The Thorvald II robots are commercially available through Saga Robotics and have been used for wheat phenotyping, strawberry harvesting, polytunnel, and greenhouse applications [32, 51, 52, 53]. AgRover [55] and Flex-Ro [40] are another two 4WD4WS robots. AgRover AgRover has a three-point chassis design to ensure all the wheels can have traction on uneven terrain. Flex-Ro is powered by hydraulic instead of electronic motors.

Tracked robot

Tracked robot uses tracks to increase the contact area with the ground, and thus its terrain adaptability is better than the wheeled robot. It can operate on rough terrains and soil conditions (e.g., muddy fields) that wheeled robots cannot. Armadillo and its improved version, Armadillo Scout, are tracked robots featuring a modular design for the track module and a robot computer platform FroboBox running the modular robot architecture, FroboMind, based on ROS [35]. TERRA-MEPP is a tracked robot designed for phenotyping energy sorghum [39]. It uses a tracked platform to carry a vertical, extendable mast (up to 4.88 m), so the sensor can capture the top view of plants. Commercial tracked robot platforms, such as LT2 from SuperDroid Robots, were used in some studies [58, 29].

Legged robot

Legged robot has excellent terrain adaptability and can be used in a complex environment, such as an agricultural field. However, not many legged robots have been developed for agricultural purposes, which could be because of the complexity in controlling the robot's locomotion and its low efficiency working on large farms [63]. With the recent advances in robotic technologies and the commercial success of legged robots, such as the Spot from Boston Dynamics, the legged robot could be a useful platform for FHTP.

Wheel-legged robot

Wheel-legged robot combines the advantage of the wheeled and legged robot. It offers speeds as high as the wheeled robot and the high terrain adaptability of the legged robot. Wheel-legged robot can achieve high maneuverability and adjust the robot's dimensions (width and height) to adapt to different field layouts [64]. One well-known wheel-legged robot is the BoniRob, which has four legs with steerable wheels [34]. This robot can adjust its width and height by adjusting the legs' posture and can achieve the same maneuverability as a 4WD4WS robot. BoniRob has a detachable module that reconfigures the robot to perform different tasks by changing the module. The downside of the wheel-legged robot is that its complexity increases the cost of the robot and makes it less robust than a wheeled robot.

2.3.2 Sensors and manuipulators

The primary function of a phenotyping robot is to measure phenotypical traits, so the robot usually carries multiple sensors to capture related information for phenotypical traits. Furthermore, sensors enabling the robot to self-drive and avoid obstacles are necessary. Manipulators are needed when making contact measurements and destructive measurements for certain phenotypical traits, such as the stalk strength of sorghum [27].

Sensors

The sensors used in phenotyping robots include the phenotyping sensors for measuring phenotypical traits and perception sensors for robots sensing the environment to make decisions. The phenotyping sensors and the perception sensors can interchangeable or be independent. The phenotyping sensors include non-contact sensors, such as imaging sensors, and contact sensors, such as a penetrometer. The most widely-used non-contact sensors are the color camera, multispectral camera, hyperspectral camera, thermal camera, stereo camera, and LiDAR sensor [65, 3]. For example, the Ladybird robot is equipped with a color camera, a 2D LiDAR, a hyperspectral camera, a stereo camera, and a thermal camera [28]. Most phenotyping robots provide mounting points to carry different sensors according to the targeted phenotypical traits. Some phenotyping robots carry environmental sensors such as soil sensors to measure environmental conditions, which are useful meta data for data processing [24, 28, 31].

In the context of the phenotyping robot, the perception sensors are used primarily for path planning. The robot needs to sense the surrounding environment and estimate its position within the environment to navigate itself to reach the targets. The Global Navigation Satellite System (GNSS) and the Inertial Measurement Unit (IMU) are often used to obtain global position and posing. Vision sensors and ranging sensors such as stereo cameras and LiDAR sensors can be used for localization and obstacle detection using the simultaneous localization and mapping (SLAM) algorithms [66]. Fusing sensory data from different sources can improve localization accuracy.

Manipulators

Manipulators, primarily robotic arms, are used commonly in agricultural robots, such as weeding and harvesting robots. However, manipulators are not very common in phenotyping because most phenotypical traits can be measured remotely. Manipulators are useful for phenotyping robots when the phenotypical traits need to be measured contactively or at a specific location (e.g., certain leaf). For example, the Robotanist robot uses a three degree of freedom robotic arm to measure the stalk strength of sorghum [27]. Sensors mounted on the robotic arm can be used to change sensing position/pose actively, such as sensing individual plants from multiple viewing angles [24, 67]. Other applications include collecting biological samples [68], leaf probing [69], soil sampling [59, 31], digging plants for root phenotyping, and fruit mapping [70].

2.3.3 Computing unit and software

Computing unit

Both single-board computers and embedded systems are used commonly in robotic systems because of their small size, low power usage, and light weight. As the brain of the robot, the primary function of the computing unit is to process sensor's data, make decisions, and control the robot. Selection of a computing unit should consider power consumption, computing performance, size, weight, interfaces, supported operating system, and available resources. Multiple computing units can be used to distribute the computing load, for example, one to perform path planning for the robot and another to process the phenotyping sensors' data. [27].

Software

The Robot Operating System (ROS) is a widely-used, middleware framework for developing robotic software because it provides an integrated environment that can greatly accelerate software development [71]. ROS has become an industrial standard for robotics and supports a wide range of hardware and algorithms commonly used in robotics. FroboMind is a software architecture built upon ROS and designed for agricultural robots [72]. Other robot software architectures can be found in [72].

2.3.4 Navigation

Navigation is an essential component of automation in robotics and includes three fundamental problems: localization, path planning, and map-building. A typical agricultural environment includes many crop rows in straight lines, and the robot needs to travel along the crop rows. Therefore, a phenotyping robot's primary navigation objective is to follow the crop row and switch between rows. GNSS, vision sensors, and LiDAR sensors are commonly used for localization and path planning, and this chapter focuses on the navigation algorithms based on these sensors in the agricultural environment.

GNSS-based navigation

As a global positioning technology, GNSS has been used widely to localize robots in field applications. GNSS-based guidance systems have been developed for agricultural machinery and robots [73]. The RTK-GNSS can provide positioning accuracy up to within a centimeter but is not always adequate for localization when used as a single positioning sensor. The positioning accuracy of GNSS can be affected by the obstruction of line-of-sight to satellites, multipath issues, and interference from other RF sources. Therefore, it is typically used with other sensors, such as the IMU and wheel encoder, to improve the localization accuracy.

The typical application of GNSS-based navigation is to make the robot follow preset paths using path-following algorithms, such as pure pursuit controller and its variants [74]. The path-following algorithm can be designed using conventional control theories, which require the robot's kinematic model [55]. Deep reinforcement learning can also be used for following

paths, does not require the robot's kinematic model, and can learn the kinematics implicitly through training [75].

In agricultural environments such as orchards, the GNSS can be unreliable because the robot frequently could move under a tree canopy blocking the satellite's signals to the GNSS receiver. The GNSS-based navigation is not suitable for dynamic environments with unexpected changes or events in the environment because a lack of understanding of the environments. In those cases, vision-based and LiDAR-based navigation algorithms can be used.

Vision-based navigation

Vision-based guidance keeps the robot following crop rows using machine vision. Color cameras typically are used to detect crop rows and calculate the robot's orientation relative to the crop row [76]. Stereo vision can provide depth information, which can help detect cotton crop rows with different illumination conditions and weed pressure than a single camera [77, 78]. Besides the traditional machine vision techniques, the deep learning methods can obtain directly the crop row's orientation from raw images [79].

Vision-based navigation relies on the image feature of the crop rows and can suffer from illumination changes and low texture. Typically, it is used with GNSS guidance, for example, fusing the vision guidance and GNSS guidance results or using the vision guidance for row following and switch to GNSS guidance when the robot switches between rows.

LiDAR-based navigation

LiDAR can measure the distance between objects. Like vision-based navigation, LiDAR-based navigation relies on landmarks that can differentiate crop rows, such as the plant, trunk, and poles in a polytunnel [80, 81, 82, 52]. The crop row measured by LiDAR sensors is represented as points aligned with some noise. Because the LiDAR sensor is subject to noise, it is difficult to detect crop rows from noisy points. A standard method is to detect the crop row using line detection algorithms, such as Hough transform and random sample consensus (RANSAC) [82, 81, 52]. Another method is to model the LiDAR measurements and noise using a particle filter and estimate the robot's heading and lateral deviation relative to the crop row [83, 84].

Using the LiDAR sensor alone can make it challenging to understand the surrounding environment because of the coarse data. The vision sensor can be used to provide complementary information to exclude the LiDAR points of no interest from data processing. For example, image features were used to separate the LiDAR points of the trunk from other objects in vineyards so that crop rows could be detected correctly [85]. The LiDAR sensor also can be used for obstacle avoidance, but can falsely detect grass, weed, and plant leaves as obstacles, so using vision sensors can help identify real obstacles.

2.3.5 Simulation

Simulation of the robotic system and its operating environment can accelerate the development of robotic systems through quick and efficient tests and validation of the robot's design

without physically building the robots [34, 86, 87, 81]. Simulation is also useful for developing and testing control algorithms, navigation algorithms, and data processing algorithms [87, 75, 52, 88]. It is easy to create repeatable testing conditions in simulations for the robot, a process that can be difficult in a real environment.

There are many simulation platforms, and popular platforms include Player [89], Webots [90], Gazebo [91], and V-REP [92]. A complete review of the simulation platforms can be found in [93]. Some simulators and frameworks customized for agricultural robotics and farm machinery have been designed based on professional simulation platforms, such as the Agricultural Architecture (Agriture) [94] and AgROS [95].

2.4 Applications of Phenotyping Robot

2.4.1 Measuring plan traits

The primary mission of a phenotyping robot is to measure phenotypical traits for plants. The phenotypical traits include but are not limited to plant architecture, biomass, and emergence rate. The traits the robot can measure depend on the phenotyping sensors carried by the robot, but the phenotyping sensors can be changed in most phenotyping robots. Robotanist uses stereo imaging and deep learning to measure the stalk width of the sorghum [44]. Stalk strength can be measured using the manipulator on the Robotanist. More plant architecture traits of sorghum, such as plant height, width, volume, surface area, and stem diameter, were measured using stereo vision on the Phenobot [38, 50]. The Vinobot can obtain detailed 3D models of the plants from the stereo images to extract phenotypical traits, such as plant

height and leaf area index [24]. As a generic phenotyping robot, the Ladybird is equipped with a stereo camera, color camera, hyperspectral camera, thermal camera, and LiDAR sensor, which can be used for different crops and can extract phenotypical traits at the plot level [28]. The Shrimp robot, mainly used in orchards, has a similar configuration of phenotyping sensors as Ladybird to provide rich information for phenotyping, such as mapping canopy volume, flowers, fruit, and yield for almond orchards [42]. The plant emergence rate can be measured using image or LiDAR sensors. A mobile robot running over the crop row was used to count maize at the early growth stage using the LiDAR sensor [96]. The maize can also be counted through visual tracking using TerraSentia running between crop rows [97, 23].

2.4.2 Counting fruits

In vineyards and orchards, automatic fruit counting is valuable for yield estimation and production management. This method poses two technical challenges: correctly detecting the fruits, and removing duplicated detections to get the correct count. The detection of the fruit can be achieved through machine vision techniques. Images are collected using a color camera on a mobile robot (Shrimp), and a Faster R-CNN model was used to detect the mango fruits [26, 43]. The detections of the same fruit from different images were registered using epipolar geometry to remove repeated count. The location of the detected fruits can be calculated through triangulation, and the robotic system can provide both the count and location of the fruits. The PHENObot uses artificial light to image grape trees in the night to remove background trees [57]. The 3D model of the grape tree was constructed using Structure from Motion (SfM) to count berries and measure berry size.

2.4.3 Collecting phenotyping dataset

Advances in deep learning show great potential for solving agricultural problems that could not be solved easily before [98, 99]. Deep learning requires a large amount of data to train reliable deep learning models that can be used in various agricultural environments. Therefore, a phenotyping robot can be a useful tool for collecting agricultural datasets for deep learning because it was designed to collect data in the field. An image dataset was collected using BoniRob for weed detection [60]. BoniRob also was used to collect datasets containing georeferenced multispectral images, and RGBD images and LiDAR data were collected for plant classification, localization and mapping on sugar beet field [61]. The Ladybird was used to collect a high-resolution multimodal dataset for Brassica with manual measurements of phenotypic traits [100]. The dataset can be used to explore new data processing algorithms for phenotyping.

2.5 Discussion

Despite the recent advances in phenotyping robots, there remain limitations. First, some phenotyping robots have been designed for specific crops and field layouts, which limits their use in other crops and field layouts. For example, robots designed for vineyards, such as PHENObot, may not be suitable for row crops because the robots' dimensions cannot fit within the row spacing [36]. Reconfigurable robots with a modular design, such as Tharvold II robotic system, might offer a promising solution for making the robot reusable for different crops. Second, the cost of phenotyping robots can be prohibitively high. The mobile platform

itself can cost tens of thousands of dollars, and the total cost of a phenotyping robot is even higher with phenotyping sensors. Although some low-cost robots have been developed, such as the TerraSentia, their use is limited because of their low payload and small size. Third, the data collection efficiency of phenotyping robots remains too low for large fields with tens of thousands of plots in practice. For example, a typical travel speed of a robot is 0.5 m/s, and the length of a plot is 3 m. A single robot would take at least 1.7 hours to scan 1000 plots of 3 m length at a travel speed of 0.5 m/s. The lengthy scanning time can make time-sensitive traits (e.g., canopy temperature) inconsistent across plots. The efficiency can be improved by using robot swarms to scan the field simultaneously, which requires coordination of the robots and can result in a higher operation costs. Fourth, although most phenotyping robots can achieve autonomous navigation, they require human supervision to prevent damaging plants if the navigation malfunctions.

Designing a phenotyping robot that can work in the unstructured and dynamically changing agricultural environment can be challenging. The robot's design (e.g., the dimension of the robot) is constrained by agronomic practices such as row spacing, which usually vary crop by crop, making it challenging to design a robot to work under the constraints properly without sacrificing the versatility. Navigation in cluttered environments is challenging, especially in GNSS-denied areas like sub-canopy. A complex navigation algorithm using vision and LiDAR is needed for those environments.

2.6 Conclusion

This chapter reviewed the existing phenotyping robots and their applications in row crops and orchards. Several research directions were identified. The first direction is to develop low-cost phenotyping robots that can be used in different crop and field layouts. The second direction is to develop innovative control algorithms for robot swarms (including aerial robots) to coordinate in large fields to improve data collection efficiency. The third direction is to develop advanced navigation algorithms to enable robust navigation without damaging plants. The fourth direction is to develop real-time data processing algorithms to extract phenotypical traits that can be deployed on robots with edge computers.

Chapter 3

Development of A Multi-Sensor Unmanned Aerial System for Field-Based High-Throughput Phenotyping[1]

[1]Rui Xu and Changying Li. To be submitted to *Remote Sensing*.

Abstract

Unmanned aerial vehicles have been used widely in plant phenotyping. This paper develops an unmanned aerial system that integrate a color, multispectral, thermal camera and a LiDAR sensor. The acquisition software was designed based on the Robot Operating System, which can visualize and record data. The design of the unmanned aerial system was opensourced. A data processing pipeline was proposed to preprocess the raw data and extract phenotypical traits at plot level, including morphological traits (canopy height, canopy cover, and canopy volume), canopy vegetation index, and canopy temperature. The multispectral, thermal camera was calibrated in the lab and in the field. The system was validated through field data collection in a cotton field. The temperature from the thermal image had a mean absolute error of 1.02°C and the canopy NDVI had a mean relative error of 6.6% compared to the ground measurements. The maximum canopy height had an error of 0.1 m compared to manual measurements. The system demonstrated can be a useful platform for plant breeding and precision management.

3.1 Introduction

High-throughput phenotyping is a method used to accelerate the breeding of certain genotypes [101]. In the past, collecting field data required significant human labor, so designing an automatic phenotyping platform to work in the field has attracted researchers' attention in

both plant science and engineering. The early solutions for such a platform is utilizing a ground vehicle (either tractor or robotic platform) with sensors to acquire the desired data [5, 6, 11]. The ground vehicle is driven either manually or automatically through each plot and data are collected and stored into a computer. The ground vehicle has the advantage of a large payload that can easily carry multiple sensors simultaneously. In addition, the ground platform can control the data collection environment (such as light conditions) with a well-designed enclosure, which guarantees data quality. However, the ground platform also has several disadvantages. For example, the data scan speed is low, planting layout for certain crops needs to be adjusted to accommodate the vehicle, frequent data collection can cause soil compaction, and the platform is difficult for use in a wide range of crops once the design is fixed.

The Unmanned Aerial Vehicle (UAV), an alternative platform, can address the disadvantages of the ground platform to some degree. Comparing to the ground platform, UAV can provide quick data scan speed and cover a large field. Because there is no intervention between the plants and UAV, it can be adapted easily for use in different types of crops and at different growth stages. Furthermore, UAV can be controlled automatically by its onboard autopilot systems, so it requires less human intervention during data collection. However, UAV has limitations as well. The payload of UAV is much lower than the ground vehicle, so it cannot carry heavy equipment. The data quality is more likely to be affected by environments because of a lack of environmental controls. The spatial resolution of aerial data is usually lower than that of ground systems.

There have been many research projects on UAV-based high throughput phenotyping published in recent years [102, 103]. Various sensors can be integrated with the UAV platform, including a color camera, a thermal camera, a hyperspectral camera, a multispectral camera and a Light Detection and Ranging sensor (LiDAR) [104]. Those sensors can provide spatial and spectral information about the plant phonemics, which can be used to measure directly or indirectly various phenotypic traits. Color images can be used to estimate leaf area index (LAI), crop emergence, and count flowers [105, 106, 107]. With the Structure from Motion (SfM) algorithm, color images can generate a Digital Surface Model (DSM), which can be used to estimate plant height, predict biomass/yield, detect crop lodging, and measure canopy structure [108, 109, 110, 111, 112]. Multispectral imaging can provide vegetation indices (such as NDVI) which have proven to correlate with leaf area index, plant disease, yield, and plant nutrient deficiency [113, 114, 115, 116, 117]. Thermal images have been used frequently to measure canopy temperature, which is an indicator of stomatal conductance and plants' response to water stress [118, 119]. Therefore, thermal images can be used to detect water stress [120, 121, 122]. Hyperspectral imaging can provide more spectral information of vegetation compared to multispectral imaging, so it can be used to monitor plants' physiological conditions [123]. Some applications have shown that the photochemical reflectance index from hyperspectral imaging can be used to access the water stress in maze [124].

Besides the great potential to use UAV for high throughput phenotyping, several challenges remain for future research. First, most of the UAV platforms were designed based on the particular aerial vehicle and sensors; therefore, the design is difficult to reuse for other

research. One solution is to design a multi-sensor data acquisition system that can be deployed on different UAV platforms [125, 126]. Second, since certain sensors can be affected by environmental conditions that potentially can cause inconsistent data between different data collections, calibration methods need to be developed for different types of sensor. For example, thermal imaging is affected by the atmosphere and needs to be calibrated properly in the field to obtain accurate thermal readings [127] Last, converting the sensory data into meaningful phenotypical traits for variety selection and plant growth requires a complex set of data processing algorithms and tools. Therefore, designing a framework that can incorporate existing and future algorithms remains a big task for researchers.

Although UAV has been used widely for plant phenotyping, it has remained a challenge to develop a UAV system for researchers without an engineering background. Therefore, the main objective of this paper is to design an unmanned aerial system that can be used by other researchers. Specific objectives include: 1) designing a data acquisition system that can be attached easily to different UAV platforms, 2) calibrating the sensors, and 3) designing a data processing pipeline for extracting phenotypical traits from the raw data. The design of the system will be open-sourced and available to all researchers.

3.2 System Design

The unmanned aerial system (UAS) adopted a modularized design that separated the system into an aerial platform and a data acquisition system (DAS). In this way, the DAS can be carried by different aerial platforms or even by ground platforms. We designed two versions of

DAS over the past five years. The first version of DAS consists of a DSLR camera (Lumix G6, Panasonic, Japan), a multispectral camera (RedEdge, MicaSense, WA, USA), and a thermal camera (Tau 2, FLIR Systems, OR, USA) (Table 3.1). The multispectral camera has five bands: Blue (475 nm), Green (560 nm), Red (668 nm), RedEdge (717 nm), and Near-Infrared (NIR, 840 nm). The second version replaced the DSLR camera with an industrial camera (GrassHopper3, FLIR Systems, OR, USA) and added a 3D LiDAR sensor (VLP-16, Velodyne Lidar, CA, USA). The first version DAS was used from the year of 2015 to 2018 for data collection and is introduced briefly in this paper. The second version is an improved version and the focus of this paper. The second version has been used for data collection since 2019.

Table 3.1: Specification of the sensors.

	Thermal camera	**Multispectral camera**	**DSLR color camera**	**Industrial color camera**	**LiDAR**
Manufacture	FLIR	MicaSense	Panasonic	FLR systems	Velodyne
Model	Tau 2	RedEdge	Lumix G6	GrassHopper3	VLP-16
Size (mm^3)	$44.5 \times 44.5 \times 30$	$113 \times 65 \times 46$	$122 \times 85 \times 71$	$44 \times 29 \times 58$	$103 \times 103 \times 72$
Weight (g)	112	150	390	90	830
Resolution	640×512	1280×800	4608×3456	2448×2048	Vertical: 1.33° Horiz.: 0.1°–0.4°
Focal length (mm)	25	5.4	14-42	5	N/A
Max FPS (Hz)	30	1	1	75	20
Spectral range (nm)	7.5–13.5	475, 560, 668, 717, 840	N/A	N/A	N/A
Accuracy (cm)	N/A	N/A	N/A	N/A	Up to ±3

3.2.1 First version DAS

The first version of DAS consisted of the DSLR camera, multispectral camera, and thermal camera. The cameras were controlled by a single-board computer (Raspberry Pi 3) (3.1A). The cameras and Raspberry Pi were mounted on a camera case and the camera case was mounted on the drone (Figure 3.1B). The GNSS/IMU (VN200, VectorNav, TX, USA) was

used to record the position and pose of the images. Two DC-DC converters were used to power the thermal camera, multispectral camera, and Raspberry Pi from the power of the drone. The DAS was carried by a high payload drone (S1000+, DJI, China). A customized program running on the Raspberry Pi was developed to trigger the cameras and record thermal images and geolocations onto the on-board memory.

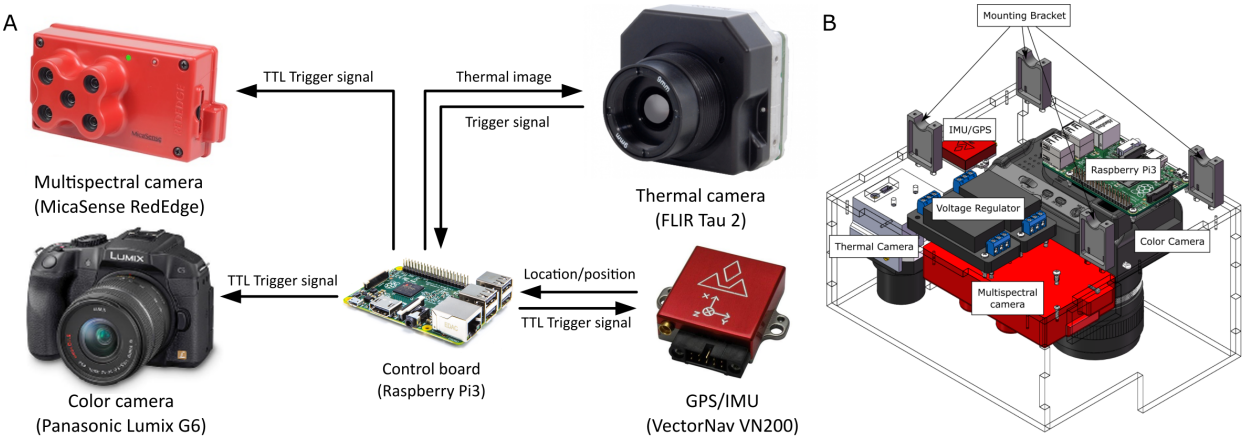

Figure 3.1: System diagram (A) and mechanical structure (B) of the first version of the data acquisition system.

3.2.2 Second version DAS

The second version of DAS consisted of three imaging sensors (thermal, color, and multispectral cameras), a LiDAR sensor, and a single board computer. The single board computer is a Manifold from DJI. Table 3.1 lists the information for the sensors. All the sensors were connected mechanically by sensor brackets, which were designed in CAD software and fabricated using 3D printer. All the 3D models were open sourced and can be downloaded from our GitHub (https://github.com/asilan/UAS_sensor_mount).

The brackets holding the sensors (blue parts in Figure 3.2) were printed in ASB plastic using a 3D printer (uPrint SE Plus, Stratasys, USA). The connection brackets (green parts in Figure 3.2) to connect the DAS with the aerial platform was printed in carbon fiber using another 3D printer (Markforged Onyx Pro, Markforged, USA) to enhance the strength of connection. The connection bracket has a slide slot which allows easy attachment and detachment of the DAS to and from the aerial platform.

The imaging sensors were mounted to one bracket so they can be detached from the DAS and used as an independent system. This is helpful for different use cases, such as when only the images data are needed or only low-payload drones are available. The imaging sensors can be configured as horizontal or vertical mounts (Figure 3.3). The imaging sensor faces frontward on the horizontal mount, which is useful for applications such as imaging the vineyard between rows. The vertical mount can be used to acquire data from above the plant canopy.

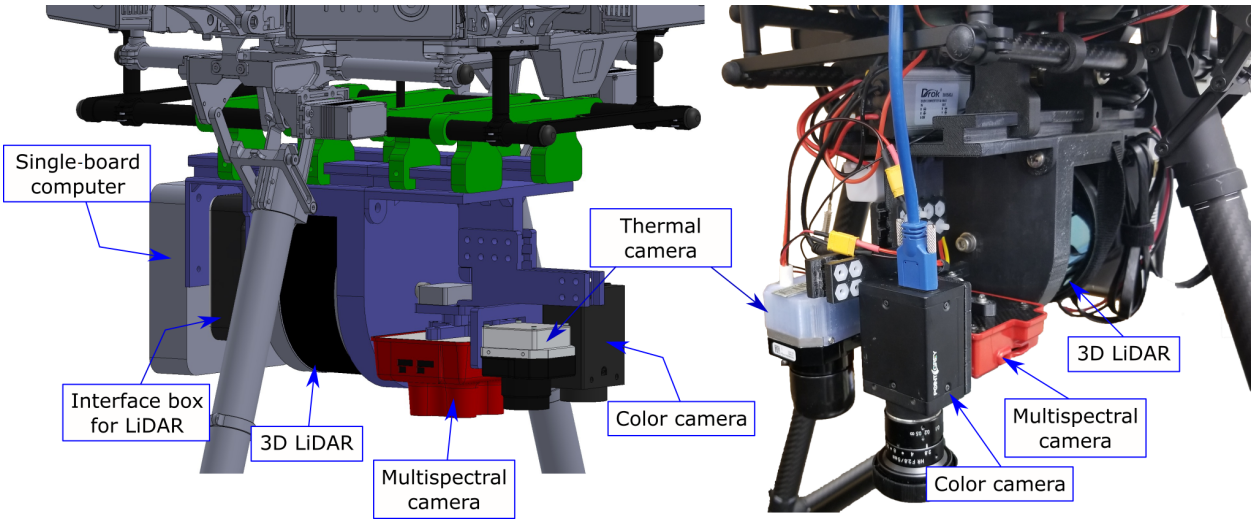

Figure 3.2: Mechanical structure of the DAS. Left figure is the rendered 3D model and right figure is the real system.

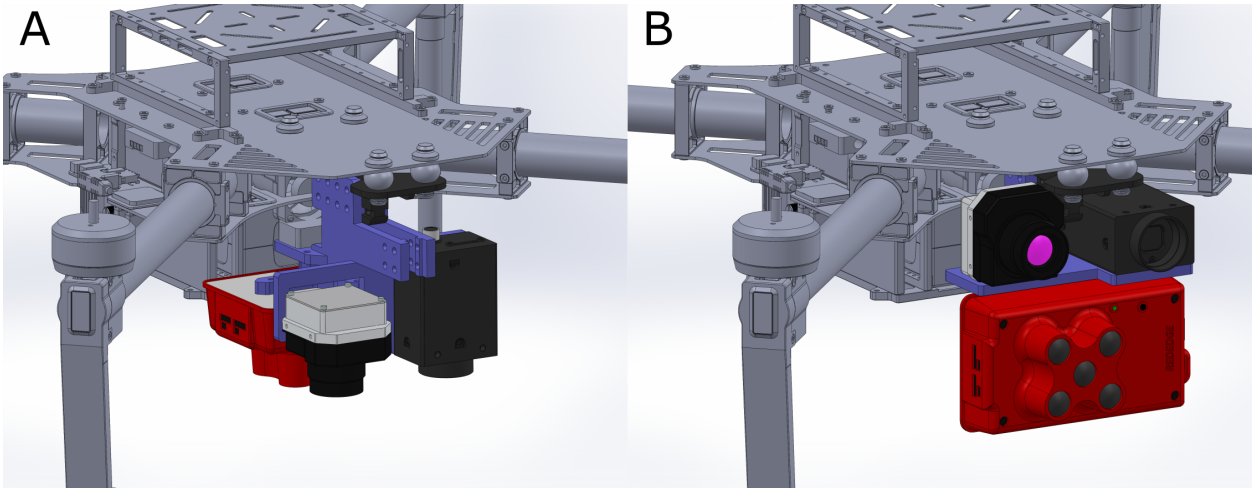

Figure 3.3: Imaging sensors mounted on a low-payload drone (DJI Matrice 100). A) Vertical mounting. B) Horizontal mounting.

Electronic design

The main power (18 V) of the DAS comes from the aerial system and directly powers the single-board computer. A DC-DC converter is used to convert the main power to 12 V and another DC-DC to convert 12 V to 5 V (Figure 3.4). The LiDAR is powered at 12 V, the thermal and multispectral camera is powered at 5 V, and the color camera is powered at 5 V through the USB port from the single-board computer. The DAS can be powered with voltage from 14 V and 26 V.

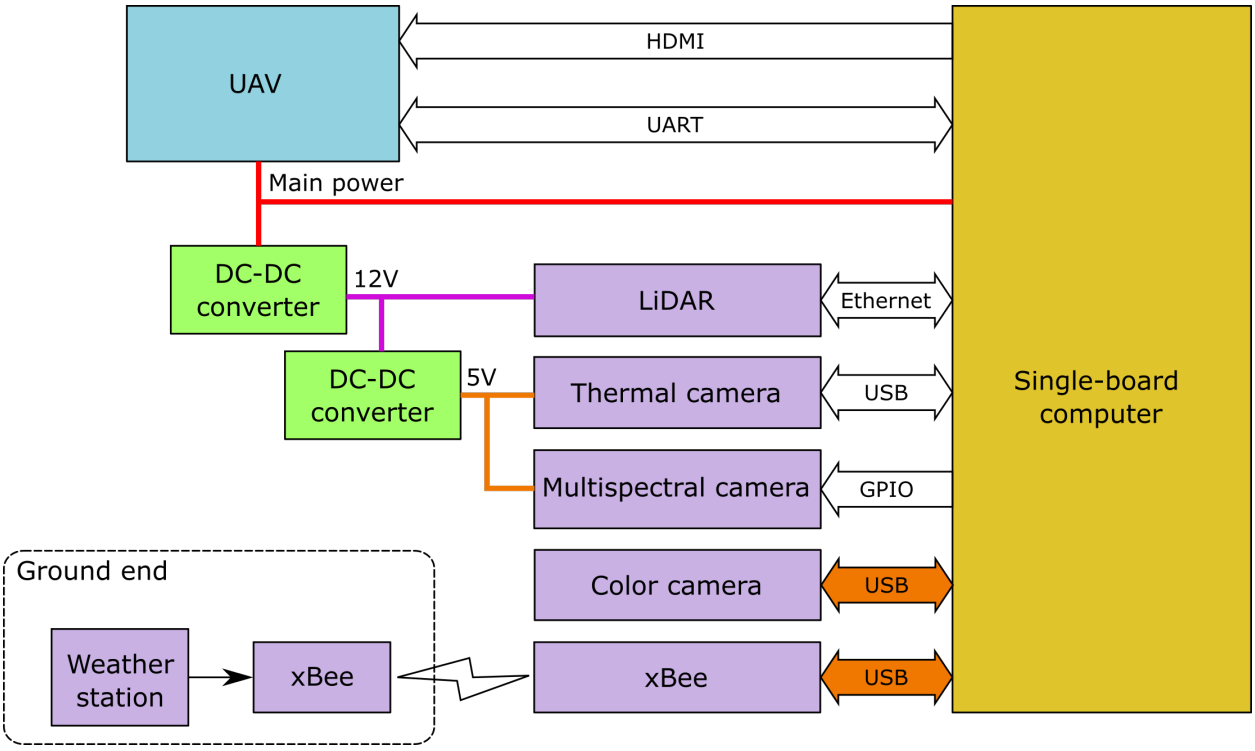

Figure 3.4: Diagram of the electronic connection of the UAS.

The single board computer (Manifold, DJI, China), running Ubuntu 14.04 armhf, is used to control the sensors and acquire data from the sensors (Figure 3.4). Controlling the thermal camera and color camera is achieved through USB and the images were transmitted through USB. The LiDAR data is transmitted to the single board computer through Ethernet. Since the multispectral camera is an independent system that has its own image processing and storing system, we only implemented the trigger function through a GPIO pin to synchronize it with other cameras, although completely controlling the multispectral camera is possible by using the serial and Ethernet interface on the camera. The single board computer can interface with the UAV through a serial port to obtain real-time flight data (e.g. position and posing of the drone). However, if the drone cannot output its flight data, additional positioning hardware (GPS and IMU) are needed for data post processing. The High-Definition

Multimedia Interface (HDMI) port of the single board computer is connected to the HDMI transmitter on the drone so the user can see the Ubuntu desktop in real-time.

The FLIR tau 2 camera originally only provided a 50-pin interface that includes a parallel digital port to output digital data. We designed a circuit based on the cypress EZ-USB chip (CY7C68013A, Cypress Semiconductor, USA) to convert the parallel digital output to a USB protocol so it could interface with the single board computer through a USB port. This circuit makes it possible to acquire raw digital radiometric data from the camera without using a frame grabber. To assist the calibration of the thermal images, a ground weather station was used to measure the temperature of the calibration target, air temperature, humidity, air pressure and downwelling thermal radiation. Those data are transmitted to the single board computer in real-time through xBee (xBee Pro S1, Digi International, USA).

Software design

The data acquisition software was implemented using the Robot Operating System (ROS) release indigo on the single-board computer (Figure 3.5). The *flir_tau2* node was implemented for the thermal camera to get raw digital images from the thermal camera. The *redege* node was implemented to trigger the multispectral camera. The *flea3* node was used to capture color images. All the cameras are synchronized through the *trigger* topic. We used the ROS package *dji_sdk* provided by DJI to get the drone's location and pose, and the *velodyne* node was used to get LiDAR data. The purpose of the *weather_sensor* node was to receive the data from the weather station and publish the data as a ROS topic. The data recording was

achieved by the *rosbag* node. The *rosbag* node saves the sensor data, drone flight data, and weather data to a bag file.

Three data visualization nodes, *thermal_display*, *color_display* and *pcl_visualization*, were implemented to visualize the color image, thermal image, and the LiDAR data. The data visualization node creates a window on the Ubuntu desktop to display the sensor data, which allows the user to monitor the data in real-time through radio transmission. The user also can check the ROS topics on the computer on the ground if the computer and the single-board computer connect to the same wireless network.

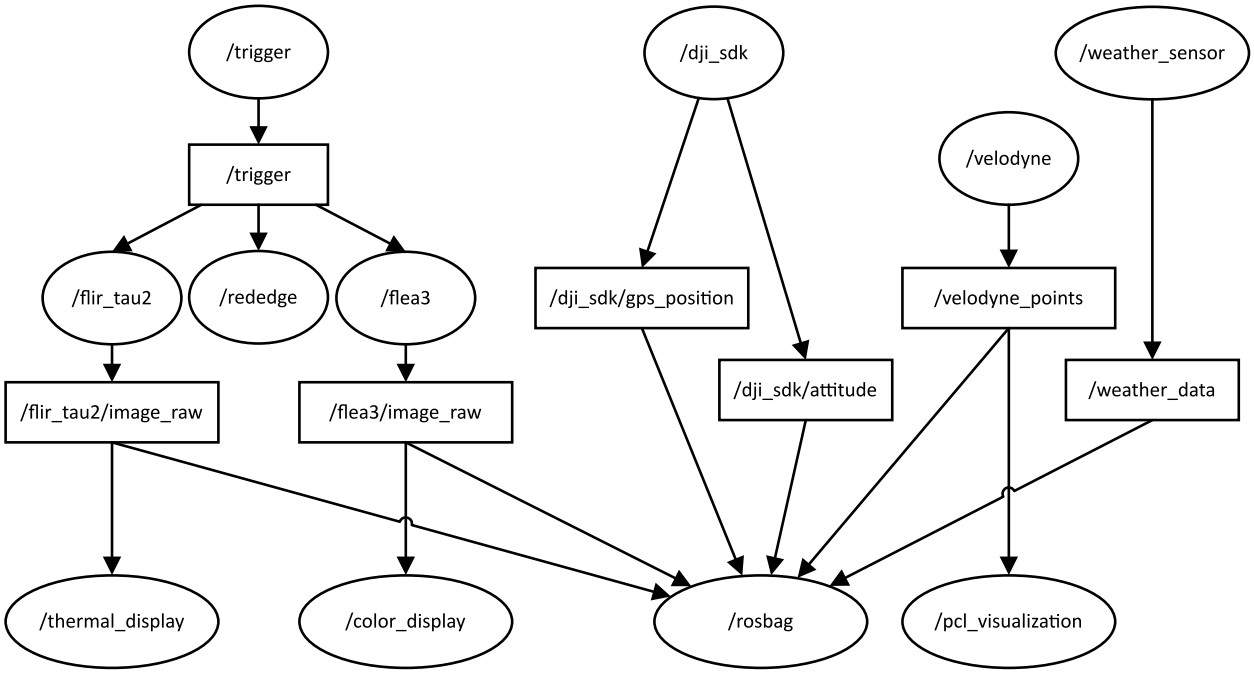

Figure 3.5: ROS computation graph. The oval indicates the node and the rectangle indicates the topic.

3.3 Camera Calibration

3.3.1 Geometric calibration

Geometric calibration is the process of estimating the camera's parameters, which can be used to correct for lens distortion. We used the Camera Calibrator from MATLAB (MATLAB 2018, MathWorks, USA) to find the parameters for each camera, including the parameters for the camera's intrinsic matrix, radial distortion model (Equation 3.1), and tangential distortion model (Equation 3.2). The radial and tangential distortions are modeled as the following equations.

$$\begin{bmatrix} x_{radial} \\ y_{radial} \end{bmatrix} = (1 + K_1 r^2 + K_2 r^4 + K_3 r^6) \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.1}$$

$$\begin{bmatrix} x_{tangential} \\ y_{tangential} \end{bmatrix} = \begin{bmatrix} 2P_1 xy + P_2(r^2 + 2x^2) \\ P_1(r^2 + 2y^2) + 2P_2 xy \end{bmatrix} \tag{3.2}$$

where $r = \sqrt{(x - x_p)^2 + (y - y_p)^2}$ is the distance between the pixel (x, y) to the principle point (x_p, y_p).

For the color and multispectral camera, we printed the chessboard on white paper. For the thermal camera, the chessboard was printed using a post printer, and the difference in the emissivity between the ink and paper creates a pattern in the thermal image. Four heat lamps were used to heat the chessboard to make the pattern clearer. The grid size of the chessboard is 21 mm × 21 mm for the color and multispectral camera and 110 mm × 110 mm

for the thermal camera. A large chessboard was used because the thermal camera needs to take images at a large distance to get focused images. Ten pictures from different angles were taken for each camera. Two tungsten halogen lamps were used for the multispectral camera to provide additional light sources for the NIR band.

3.3.2 Vignette calibration

Vignetting is an effect of the radial falloff of intensity from the center of the image, including natural vignetting, pixel vignetting, optical vignetting, and mechanical vignetting [128]. There are two ways to model the vignette effect. The first way is to model the vignette effect ($V(x, y)$) as a high order polynomial (Equation 3.3) and assumed zero vignette effect ($V(x_v, y_v) = 1$) at vignette center (x_v, y_v).

$$V(x, y) = V(r) = 1 + \alpha_1 r + \alpha_2 r^2 + \alpha_3 r^3 + \alpha_4 r^4 + \alpha_5 r^5 + \alpha_6 r^6 \tag{3.3}$$

$$r = \sqrt{(x - x_v)^2 + (y - y_v)^2} \tag{3.4}$$

where r is the distance of the pixel from the vignette center, and α_1 to α_6 are the model parameters.

The second way is to model the vignette effect as a look up table that stores the correction coefficients for each pixel. The second method can be more accurate that the first method but requires multiple tables for different camera settings. Since the first method is simple to implement and is effective, it was chosen to model the vignetting of the multispectral camera.

One way to measure the vignette is using the camera to image a scene with uniform luminance. For the multispectral camera, we used an integrating sphere to create such scene. The setup of the integrating sphere is similar to [129], but we added a Teflon sheet to cover the exit port so the light could be more evenly distributed after passing the Teflon sheet. A black body was used for the same purpose for the thermal camera. We took 10 images for each camera and used the mean image to find the model parameters. The model parameters (x_v, y_v and α_1 to α_6) were estimated using least squares fitting. The value of $V(x, y)$ for fitting the model is the raw pixel intensity subtracting the black level and normalized against the pixel intensity at the vignette center.

3.4 Data Processing

3.4.1 Overall pipeline

The data processing pipeline includes data preprocessing and phenotypical traits extraction (Figure 3.6). The data preprocessing calibrates the images, generates georeferenced orthomosaics, and constructs 3D models. With the preprocessed data, various phenotypical traits can be extracted, such as canopy morphological traits, canopy vegetation index, and canopy temperature. Some data preprocessing can be done in real-time by the onboard computer, which helps with real-time actions such as coordination with the ground robots. We implemented the online image geometric correction for the thermal and color images. We mainly used Metashape (Metashape Professional 1.6.4, Agisoft, Russia) to process the image data offline.

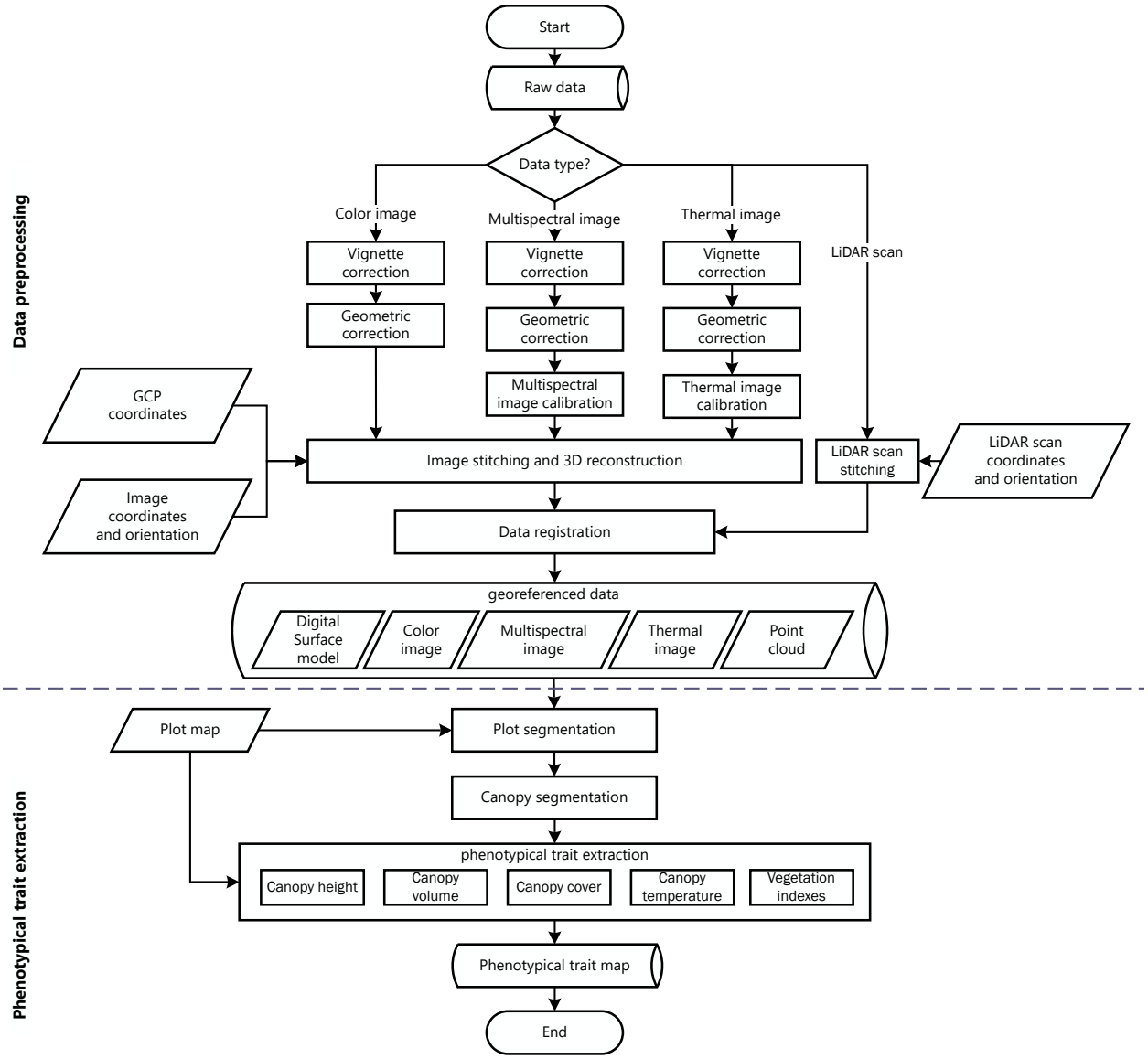

Figure 3.6: Overall data processing pipeline.

3.4.2 Data preprocessing

Vignette correction

Vignette correction was done using the vignette model obtained from the vignette calibration. The corrected image intensity ($I_{corrected}$) is calculated using equation 3.5 from the original

image intensity (I) and the vignette correction factor (V).

$$I_{corrected}(x,y) = \frac{I(x,y) - BL(x,y)}{V(x,y)} \tag{3.5}$$

Geometric correction

The geometric correction was done using the coefficients of the lens distortion model obtained from the geometric calibration. The geometric correction was done using the *image_geometry* package and coded in the ROS nodes for the color camera and thermal camera. The calibrated images were published as *flir_tau2/rect_image* and *flea3/rect_image* topics.

Multispectral camera calibration

The raw multispectral images needed to be calibrated to get reflectance off the ground objects. The calibration included three steps: radiometric calibration, sunlight intensity correction, and reflectance calculation. The radiometric calibration is used to convert the raw digital value of the multispectral image to absolute spectral radiance values. The radiometric calibration model provided by MicaSense (https://support.micasense.com/hc/en-us/articles/115000351194-RedEdge-Camera-Radiometric-Calibration-Model) was used to compensate for sensor black-level, the sensitivity of the sensor, sensor gain and exposure settings, and optical vignette effects (equation 3.6).

$$L(x,y) = \frac{1}{V(x,y)} \cdot \frac{a_1}{g} \cdot \frac{p(x,y) - p_{BL}}{t_e + a_2 y - a_3 t_e y} \tag{3.6}$$

where L is the absolute spectral radiance values in $\mathrm{W\,m^{-2}\,sr^{-1}\,nm^{-1}}$, V is the correction factor from the vignette model, p is the normalized raw pixel value, p_{BL} is the normalized black level value, a_1 to a_3 are the radiometric calibration coefficients that can be get from the image meta data, t_e is the image exposure time, g is the sensor gain setting, y is the row number, and (x, y) is the pixel position.

After radiometric calibration, the radiance of each multipsectral image was corrected against the sun light intensity to make the radiance reflects the same sunlight intensity. The intensity of the sunlight was measured using the downwelling light sensor provided by MicaSense.

The reflectance panel was imaged before and after the flight and was used to convert the radiance to reflectance. A scale factor was calculated between the mean radiance and the reflectance of the reflectance panel. Since the calibration procedure was implemented in the Metashape software, we directly used the reflectance calibration function to calculate the reflectance.

Thermal camera calibration

Thermal image is affected by the environment. The accuracy of the temperature measurement from the thermal image is largely depended upon how much the environmental effects were compensated, especially when the object was far away from the camera. It is essential for aerial thermal imaging to correct the atmospheric effects to acquire accurate temperature reading.

According to the Stefan-Boltzmann law, the total radiation emitted by the object is expressed as Equation 3.7.

$$B(T) = \varepsilon \cdot \sigma \cdot T^4 \tag{3.7}$$

where ε is the emissivity and σ is the Stefan-Boltzmann constant ($5.67 \times 10^{-8}\,\mathrm{W\,m^{-2}\,K^{-4}}$), T is the absolute temperature (K).

The total radiation received by the thermal camera (W_{sensor}) consists of emission of the object (E_{obj}), the emission of the surroundings and reflected by the object (E_{refl}), and the emission of the atmosphere (E_{atm}) (Equation 3.8) [130].

$$W_{sensor} = E_{obj} + E_{refl} + E_{atm} \tag{3.8}$$

W_{sensor} can be expressed as

$$W_{sensor} = \sigma \cdot BT_{sensor}^4 \tag{3.9}$$

where BT_{sensor} is the brightness temperature provided by the thermal camera by setting the emissivity of the object to 1 and distance to 0.

Since the surface radiance is received partially by the camera and some is absorbed by the atmosphere, E_{obj} can be expressed as Equation 3.10 as a function of the transmittance of the atmosphere (τ) and the object's temperature (T_{obj}).

$$E_{obj} = \tau \cdot \varepsilon \cdot \sigma \cdot T_{obj}^4 \tag{3.10}$$

E_{refl} is of the reflected radiation of the surroundings (which is the downwelling atmosphere radiation) and it is partly absorbed by the atmosphere. Thus, E_{refl} can be expressed as the following equation,

$$E_{refl} = \tau \cdot (1 - \varepsilon) \cdot \sigma \cdot T_{refl}^4 \tag{3.11}$$

E_{atm} is the emission of the atmosphere that reaches to the thermal camera, which is the upwelling atmosphere radiation. It can be expressed as as the following equation,

$$E_{atm} = \varepsilon_{atm} \cdot \sigma \cdot T_{atm}^4 = (1 - \tau) \cdot \sigma \cdot T_{atm}^4 \tag{3.12}$$

where T_{atm} is the temperature of the atmosphere (air temperature), and $(1 - \tau)$ is the emissivity of the atmosphere.

Equation 3.13 can be obtained to retrieve the temperature of the object by combing Equation 3.8, 3.9,3.10,3.11 and 3.12.

$$T_{obj} = \sqrt[4]{\frac{BT_{sensor}^4 - \tau \cdot (1 - \varepsilon) \cdot T_{refl}^4 - (1 - \tau) \cdot T_{atm}^4}{\tau \cdot \varepsilon}} \tag{3.13}$$

To solve the equation 3.13, the transmittance of the atmosphere, the emissivity of the object, the air temperature, and the reflected temperature needs to be supplied. The air temperature can be measured by a temperature sensor. The reflected temperature can be indirectly measured by measuring the apparent temperature of an aluminum plate or directly measured by measuring the sky's apparent temperature using an infrared thermometer.

The transmittance of the atmosphere (τ) is estimated using the water vapor content (WVC) and flight height above the ground (distance) with the following equation [131].

$$\tau = K_{atm} \cdot \exp\left[-\sqrt{d}\left(\alpha_1 + \beta_1\sqrt{\omega}\right)\right] + (1 - Katm) \cdot \exp\left[-\sqrt{d}\left(\alpha_2 + \beta_2\sqrt{\omega}\right)\right] \tag{3.14}$$

where d is the distance (m), and $K_{atm} = 1.9$ is the scalling factor of atmospheric damping, $\alpha_1 = 0.0066$ and $\alpha_2 = 0.00126$ are the attenuation of the atmosphere without water vapour, and $\beta_1 = -0.0023$ and $\beta_2 = -0.0067$ are the attenuation of water vapour. The WVC can be estimated using the air temperature and humidity using the following equation [131].

$$\omega = \omega_{\%} \cdot \exp\left(h_1 \cdot T_a^3 + h_2 \cdot T_a^2 + h_3 \cdot T_a + h_4\right) \tag{3.15}$$

where ω is the water vapour content (mm), $\omega_{\%}$ is the relative humidity (ranging from 0-1; dimensionless), T_a is the air temperature (°C), $h_1 = 6.8455 \times 10^{-7}$, $h_2 = -2.7816 \times 10^{-4}$, $h_3 = 6.939 \times 10^{-2}$ and $h_4 = 1.558$ [131].

It was reported that the emissivity of vegetation ranges from 0.96 to 0.99, while the emissivity of soil is around 0.98 [132, 133]. The inaccuracy in emissivity can lead to temperature errors of up to several degree of Celsius [134]. In this study, we estimated the emissivity of the canopy and soil from the Normalized Difference Vegetation Index (NDVI) using the method proposed by [135].

To assist the thermal calibration and validate the thermal measurement, we made two thermal calibration targets and a customized device to record atmospheric conditions, including air temperature, air pressure, and humidity (Figure 3.7). Similar to other studies,

the calibration target was made from a 0.6 m × 0.6 m aluminum plate whose top surface was painted by flat paint and the back was covered by the polystyrene insulation foam [127]. The edge of the plate was wrapped with aluminum tape to assist auto-detection of the target. One target (cold target) was painted in white to create a low-temperature reference, and the other (hot target) was painted in black to create a high-temperature reference. The top surface temperature of the target was measured by a resistance temperature detector (RTD) (700-102AAB-B00, Honeywell, NC, USA) attached to the back of the aluminum plate. The emissivity of the white paint and black paint was measured to be 0.968 and 0.984 using the reference temperature method [134]. A thermal camera pointing upward is used to measure the reflected temperature. All the ground measurements can be transmitted to the DAS through xBee in real-time.

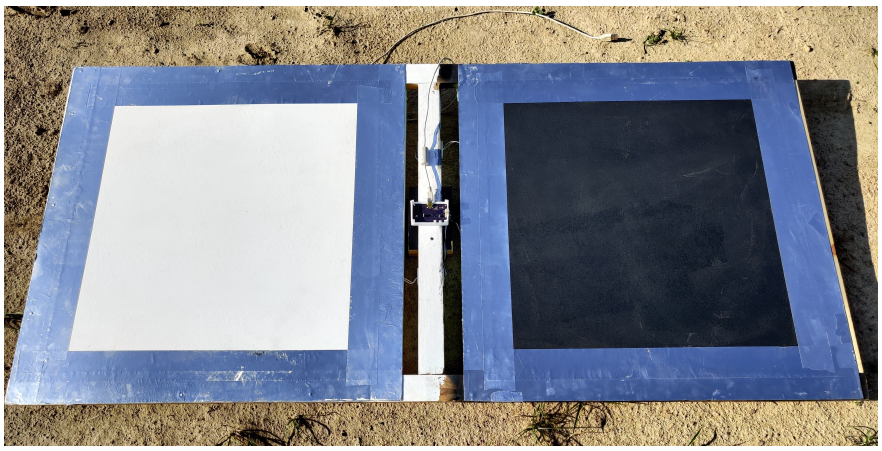

Figure 3.7: Thermal calibration targets.

LiDAR data processing

LiDAR scans can be assembled into a point cloud using the scans' position and pose in the Easting Northing UP (ENU) frame. Therefore, RTK-GNSS and IMU should be used to

measure the accurate position and posing so the LiDAR scans can be assembled accurately. Since the current UAS does not have an RTK-GNSS and IMU, the LiDAR data was not used in this study.

Orthomosaics and 3D model

The images from the same flight were stitched to generate the ortho-mosaic. In addition, a 3D point cloud and DSM can be generated using SfM. In this study, the Metashape software was used to generate the DSM of the field from the color images.

Data registration

Data registration includes the registration of the orthomosaics from different cameras and the registration of the orthomosaics and the LiDAR data. In this project, the registration of the orthomosaics is based on the geoinformation of the orthomosaics. The accuracy of the registration will depend on the accuracy of the geoinformation. Increasing the accuracy of each image's geolocation, such as using an RTK-GNSS and ground control points (GCPs), will improve the overall accuracy of the geoinformation of the orthomosaic.

3.4.3 Extraction of phenotypical traits

With the information captured by the imaging sensors and LiDAR, multiple phenotypical traits can be extracted. Some basic phenotypical traits can be extracted directly after data preprocessing, which includes morphological traits, such as canopy height (CH), canopy cover (CC) and canopy volume (CV), vegetation indices (VI) derived from the multispectral image,

and canopy temperature derived from the thermal image. A program was created to extract the phenotypical traits automatically using QGIS 3.10 and Python 3. The extracted traits were saved to a csv file and visualized in QGIS.

Plot segmentation

The data of the entire field can be separated into each plot using the plot layout map. The data of each plot contained the 2D images and 3D model of the plot. Because each plot can be seen by several images, a plot can have multiple plot images, including the image extracted from the orthomosaics. The plot images from different view angles could provide more information about the plot, such as detecting cotton flowers [107].

The 3D model of the plot needed to be normalized against the ground level so the height of the data points represented the height above the ground, namely the plot height model. A 2D plane that represents the ground level can be found by fitting the plot model using MLESAC [136]. If there are not enough ground points inside the plot model because of canopy enclosure, the ground plane can be represented by the bare ground model, or Digital Elevation Model (DEM), collected before the plant germinated [108]. However, this requires additional data collection and accurate georegistration between the bare ground model and the plot model. After finding the ground level, the plot model was subtracted by the ground level so it represent the height above ground. In this study, both the plot model and the bare ground model were used to get the plot height model and the results to calculate the morphological traits were compared.

Canopy segmentation

The canopy data needs to be segmented from the plot data to calculate phenotypical traits for the plant canopy. The segmentation result is a canopy mask generated from the 2D plot image or a canopy model generated from the plot height model. Color, spectral, and height differences between the canopy and ground can be used for the segmentation [116]. In this study, the thresholding method was used to segment the canopy using the color plot image and 3D plot model. Using deep neural networks such as Mask R-CNN may generate better results but is out of this paper's scope.

Morphological traits

Morphological traits are important phenotypical traits that are often related to plant growth status. For example, canopy height and volume can quantify the cotton biomass, which is correlated with the yield. The morphological traits are calculated from the canopy model.

Canopy height The canopy height was calculated using the canopy model. Some studies used an average or median canopy height to represent the canopy height, which is suitable for crops whose canopy are evenly distributed, such as barley and wheat [108, 137, 112]. Some studies have used a percentile of the canopy height to represent the height of the canopy, which can reduce the effect of uneven distribution of the canopy [111, 138]. This study calculated the average, maximum, median, and 50th to 99th percentiles with a 10th interval for the canopy height.

Canopy volume The canopy volume was calculated using the canopy model. The canopy volume can be represented by the volume of the mesh that encloses the canopy model. The mesh can be found using a convex hull algorithm. However, since the canopy model generated by the aerial images or LiDAR usually lacks information under the canopy surface, the volume directly from the mesh can underestimate the canopy. Instead, we proposed to calculate the 2.5D volume as the canopy volume. The canopy model was first converted to a depth image whose intensity represented the height, and then the 2.5D volume of the depth image was calculated using equation 3.16.

$$CV = \Delta x \Delta y \sum_i \sum_j d_{i,j} \tag{3.16}$$

where Δx and Δy is the ground sample distance and $d_{i,j}$ is the depth of the pixel.

Canopy cover Canopy cover is defined as the ratio of the vertical projected area of the canopy to the area of the plot. After segmenting the canopy, the area of the canopy can be calculated as the ratio of the pixel area of the canopy mask to the pixel area of the plot.

Vegetation index

Various vegetation index can be extracted from the multispectral images [139]. In this paper, we calculated the Normalized Difference Vegetation Index (NDVI) and the Normalized Difference Red Edge Index (NDRE), which are two commonly used indexes. The mean value was calculated within the canopy as the vegetation index of the canopy.

Canopy temperature

The canopy temperature is calculated using the plot image from the thermal orthomosaic. Similar to the vegetation index, the canopy temperature is the average temperature within the canopy.

3.5 Data Collection

3.5.1 Test field

The test field was a cotton field located on a research farm at The University of Georgia located in Watkinsville, Georgia. The layout of the cotton field had 96 plots arranged in 8 columns and 12 rows (Figure 3.8). The length of each plot was about 3 m, and there was a 1.5 m long alley between the plots. The distance between the crop rows was 1.8 m.

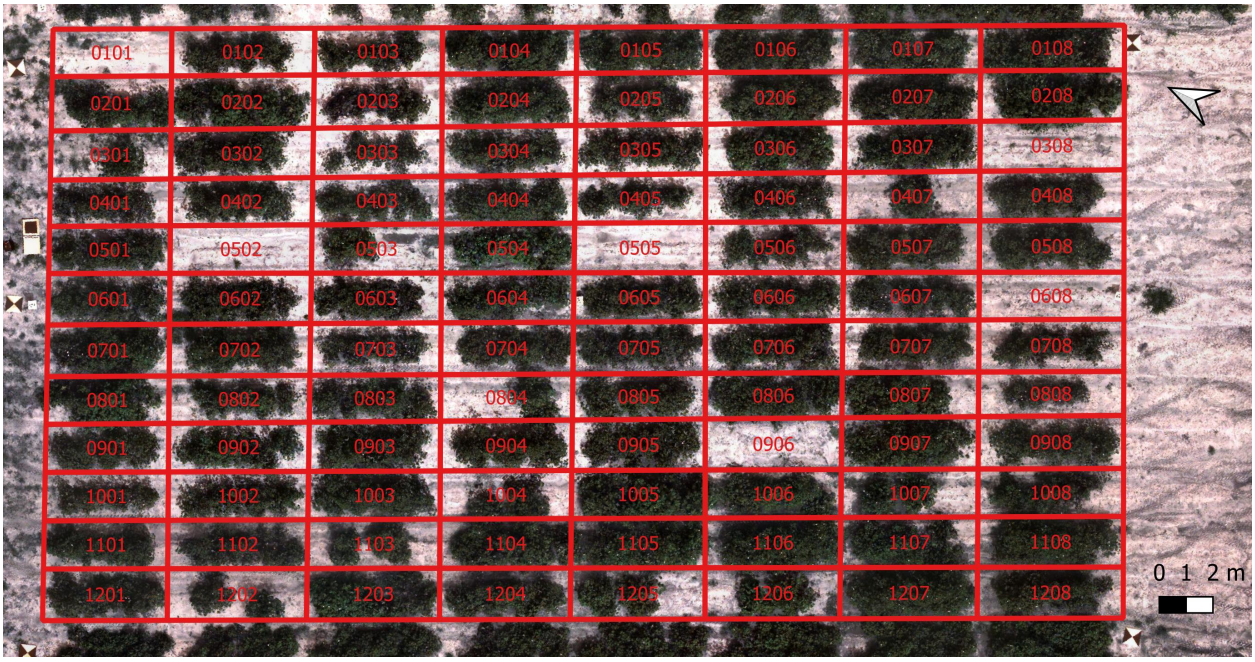

Figure 3.8: Field layout.

3.5.2 Field preparation

To facilitate the data collection throughout the season, some preparation had to be done before the live data collection. First, to improve the accuracy of the georeferencing of the orthomosaic, ground control points (GCP) had to be deployed in the field. The GCP is a ground target with a known geolocation. The GCP usually has a distinct pattern that can be identified easily in the images. Photogrammetry software, uch as Agisoft Metashape provides GCP patterns that can be identified automatically by the software. In this study, we made two sets of GCPs: color GCPs for color and multispectral cameras, and thermal GCPs for thermal camera. We used the 12 bit circular barcode generated by the Metashape as the pattern of the color GCP (Figure 3.9). The pattern was made from a black vinyl sheet and pasted onto a white acrylic panel. The size of the acrylic panel was 0.6 m $\times$ 0.6 m

and the center point radius of the circular barcode was 30 mm. The thermal GCP was made of a wood panel with a size of 0.8 m × 0.8 m, and the wood panel was evenly divided into four quadrants with two opposite quadrants painted in black and two covered by aluminum tape (Figure 3.9). With the distinct emissivity of the black paint and aluminum, the thermal GCPs could be seen easily in the thermal images. The Metashape software recommends using at least 8 GCPs. We used nine color GCPs and six thermal GCPs. The color GCPs were fixed in the field throughout the season. The thermal GCPs were deployed in the field when collecting data because the thermal GCPs were not weather proof. The geolocation of the color GCPs were measured periodically using a RTK-GNSS throughout the season to correct the movement of the GCPs during the season. The geolocation of thermal GCPs was obtained from the georeferenced color orthomosaic.

Another important preparation step is to map the field at the beginning of the plant season. The field mapping serves two purposes. The first purpose is to generate a plot layout map to be used for plot segmentation. The second purpose is to generate the bare ground model. In this study, the field mapping was done one week after planting.

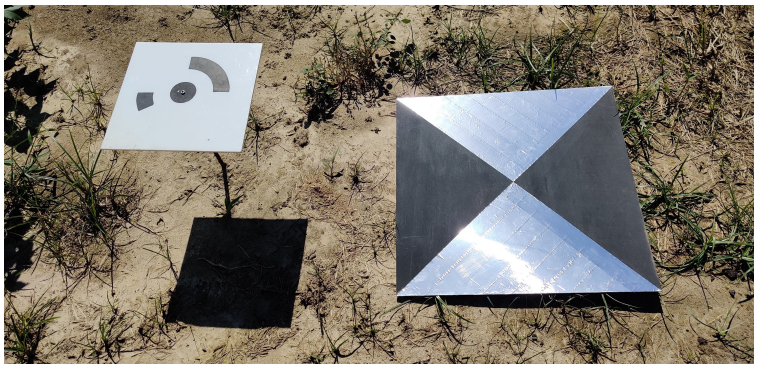

Figure 3.9: Color GCP (left) and thermal GCP (right) in the field.

3.5.3 Flight path planning

Most drones can fly automatically on a preset flight path using autopilot. Two primary factors, flight height above ground and image overlap, needed to be considered when planning the flight path. The flight height affects the ground resolution of the aerial data. The image overlap affects the quality of image stitching and 3D reconstruction. The image overlap is affected by the flight speed, flight height, image recording frequency, and flight path. Within the same flight path, lower flight height, faster flight speed, and lower image recording frequency resulted in a lower overlap. Increasing the flight height, reducing the flight speed, and increasing image recording frequency can increase forward overlap. Increasing the flight height and reducing the distance between flight lines can increase the side overlap. Another way to increase the overlap is to change the flight path pattern with more flight lines, such as the cross-stitch flight path [140].

On the one hand, lower flight height and higher image overlap can effectively increase the image resolution and quality of the image stitching and 3D reconstruction. On the other hand, it increases the overall flight time and collected more images to process, which increases the data processing time. Therefore, the balance between flight height, flight speed, and image overlap had to be considered. We herein proposed a general decision-making process. The first step is to select the appropriate flight height based on the desired ground image resolution. The second step is to decide the image side overlap. [141] suggested that a minimum of a 67% side overlap should be used to get complete 3D reconstruction. Based on our experience, a good range of side overlap is 70% to 80% considering the flight time,

area coverage, and processing time. Once the flight height and side overlap is decided, the flight path can be generated. The final step is to decide the forward overlap and flight speed. Higher forward overlap can create better and completer 3D reconstruction results [141]. It is noteworthy that the camera should not move larger than half of the Ground Sampling Distance (GSD) during exposure to get a sharp image. In this case, the flight speed should be smaller than $\frac{dh}{2T_e f}$, where d is the pixel pitch, h is the flight height, T_e is the exposure time and f is the focal length.

3.5.4 Flight campaign

The DJI Matrice 600 and the second version of the DAS were used for data collection. The Pixel4D mapper was used to generate the flight path and control the drone. The flight path was generated using a 25 m flight height above the ground with 70% side overlap and 90% forward overlap (Figure 3.10). The speed of the drone was set at 1 m/s. The flight path was calculated based on the parameters of the thermal camera because it has the smallest field of view among all the cameras. For the color and multispectral cameras, the actual image overlap was higher in practice. Because of the DJI manifold's limited data saving speed, the camera triggering frequency was set to a 0.5 Hz frequency to avoid data loss, although the color and thermal cameras support much higher frequency. The LiDAR sensor was configured to output data at 20 Hz. The scanning range of the LiDAR sensor was configured to $\pm 70°$ to remove unwanted data.

The flight campaign was performed on October 1st, 2020 at 2:30 p.m. The weather was sunny, with an average wind speed of 1.5 m/s. The reflectance target for the multispectral

camera was imaged before and after the flight. The thermal calibration targets were mounted on a mobile robot that moved along the border of the field so that the thermal calibration targets could be imaged multiple times by the thermal camera during the flight (Figure 3.10). The thermal calibration targets were imaged nine times in total.

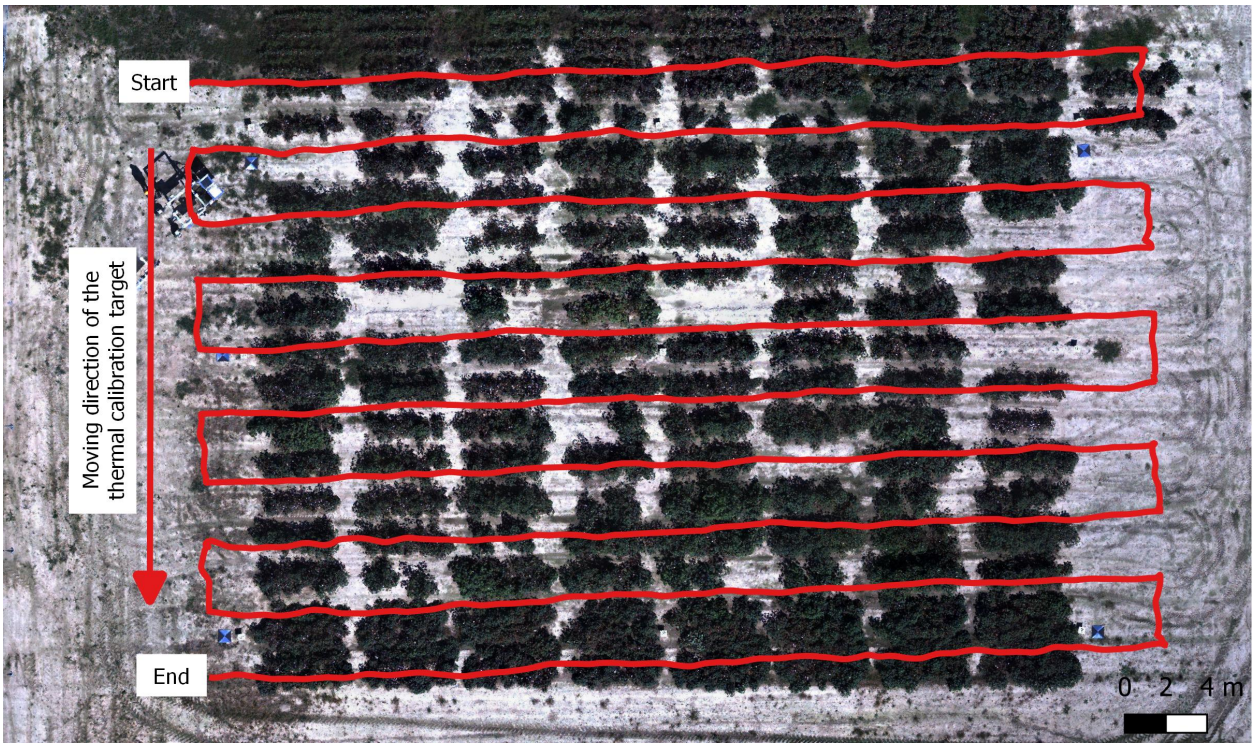

Figure 3.10: Flight path recorded by the drone. Distance between flight line is 3.3 m.

3.5.5 Ground data collection

Ground data were manually collected for one crop row in the field on the same time as the flight campaign, and were used as references to validate the extracted traits from images. For each plot, the height of each plant was manually measured using a ruler. The average height and maximum height was calculated as the average canopy height and maximum canopy height. The NDVI of each plot was measured using a handheld NDVI sensor (GreenSeeker,

Trimble, CA, USA). Each plot was measured four times at different locations, and the average value was used as the reference of the canopy NDVI. The thermal calibration targets were used as references for validating the thermal images. The temperature measured by the RTD sensor on the thermal calibration targets was used as ground truth of the targets' surface temperature and was compared with the temperature measured by the thermal camera. The extracted canopy height and canopy NDVI were compared with the manual measurements. The mean absolute error (MAE) (Equation 3.17) and mean relative error (MRE) (Equation 3.18) were used to evaluate the error.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\text{measured}_i - \text{truth}_i| \tag{3.17}$$

$$MRE = \frac{1}{N}\sum_{i=1}^{N}\frac{|\text{measured}_i - \text{truth}_i|}{\text{truth}_i} \tag{3.18}$$

3.6 Results

3.6.1 Camera calibration

Camera geometric distortion

Table 3.2 shows the cameras' intrinsic parameters and the parameters of the distortion models. The multispectral camera's nominal focal length is 5.4 mm, but the focal lengths over the five bands ranged from 5.469 mm in the Red band to 5.513 mm in the Green band. The color camera's and thermal camera's focal lengths are very close to their nominal focal length, which is 5 mm and 25 mm, respectively. The NIR band has the largest offset of the principal point

from the origin (central pixel) in the multispectral camera. The thermal camera's principal point also has a large offset from the origin in the y-axis. The total distortion (combining the radial and tangential distortion) of the three cameras is visualized by Figure 3.11 and Figure 3.12. The three cameras have low distortion, and the distortion is only noticeable in the border of the image.

In practice, camera distortion optimization is usually part of the bulk bundle adjustment of the photogrammetry process for image stitching and 3D reconstruction, so it is unnecessary to perform the correction separately. However, the model parameters in Table 3.2 can be used as the initial values for the camera distortion optimization in the photogrammetry process.

Table 3.2: Camera calibration parameters for the color, multispectral, and thermal camera.

	Color	Multispectral camera					Thermal
	camera	Blue	Green	Red	RedEdge	NIR	camera
Focal length (mm)	5.004	5.470	5.513	5.469	5.477	5.499	25.099
y_p (mm)	0.072	0.036	0.041	0.023	0.028	0.049	-0.049
y_p (mm)	0.020	0.061	0.000	-0.045	-0.015	0.098	-0.240
Skew angle (rad)	8.26E-04	-1.00E-03	-1.87E-03	-7.75E-04	-1.14E-03	-5.82E-04	-3.12E-03
K_1	-5.42E-02	-9.78E-03	-9.85E-03	-2.61E-02	-8.10E-03	-1.09E-02	-3.27E-01
K_2	1.08E-01	-1.57E-01	-1.27E-01	-5.42E-02	-2.61E-01	-7.20E-02	1.77E+00
K_3	-4.44E-02	-4.38E-01	-3.90E-01	-4.66E-01	1.71E-01	-8.09E-01	-3.61E+01
P_1	-1.34E-03	2.08E-03	5.72E-04	-1.55E-03	-2.10E-03	6.55E-05	8.50E-03
P_2	1.25E-03	5.86E-05	2.76E-03	1.95E-03	7.78E-04	1.64E-03	6.73E-04

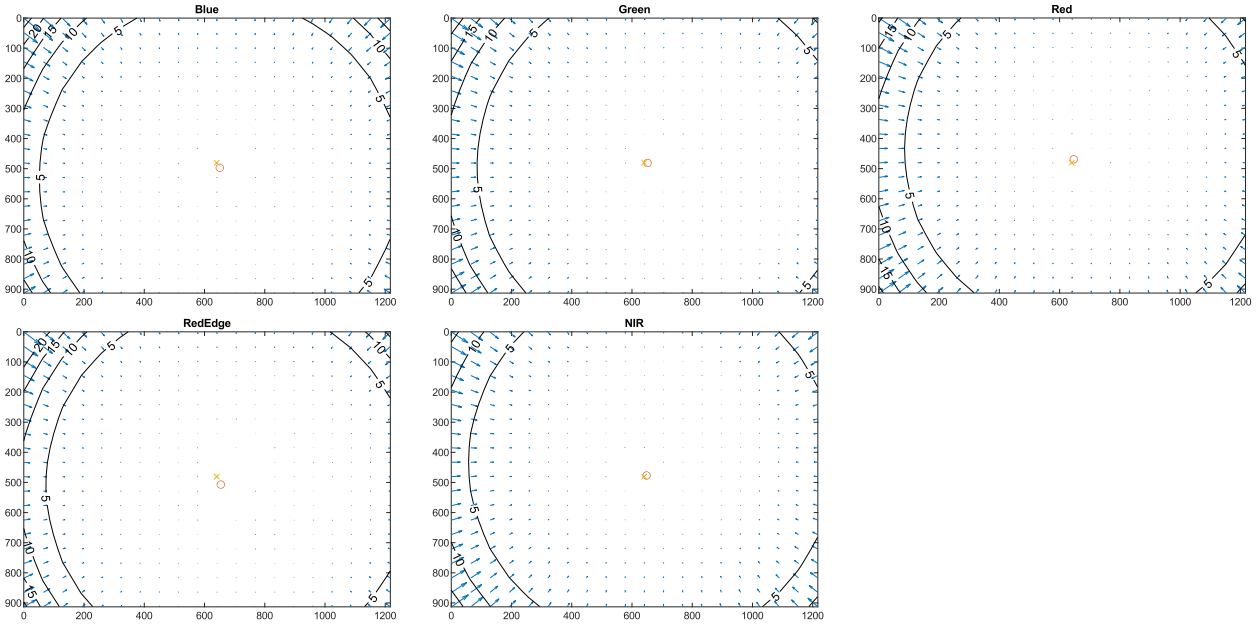

Figure 3.11: Geometric distortion of the five bands of the multispectral camera. The cross indicate the image center. The orange circle indicates the principal point. The contour lines indicate distortion in pixel.

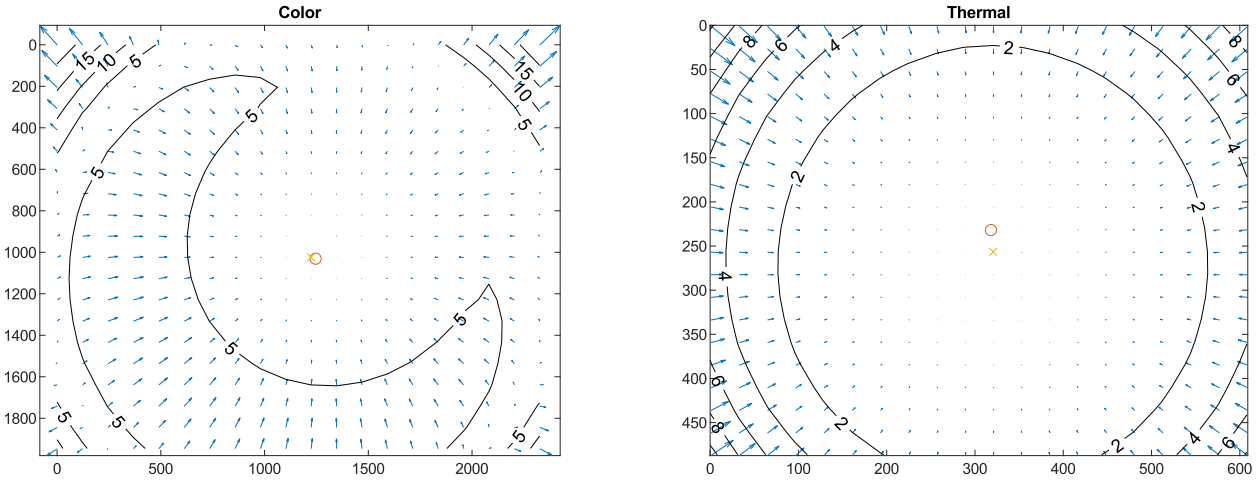

Figure 3.12: Geometric distortion of the color and thermal camera. The cross indicate the image center. The orange circle indicates the principal point. The contour lines indicate distortion in pixel.

Camera vignetting

Figure 3.13 showed the multispectral camera has vignetting that the image is brighter than the border, which is mainly caused by the lens. Each band has different vignetting and should be corrected differently. The vignette effect can be removed after applying the vignette correction model.

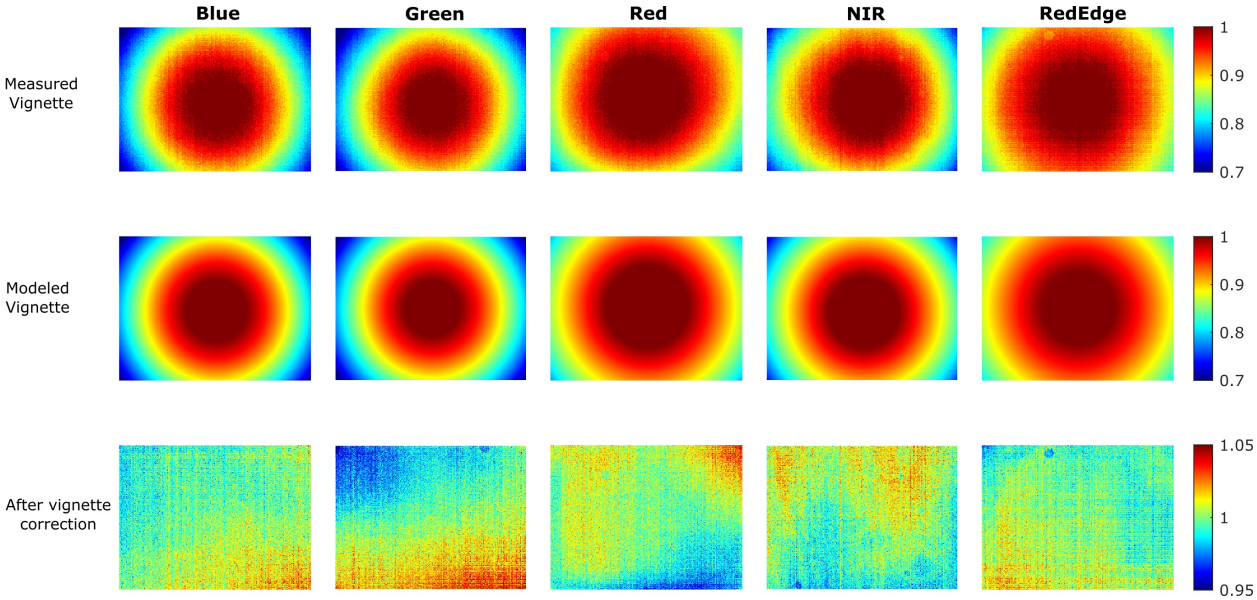

Figure 3.13: Vignette effect of the multispectral camera.

The vignette effect of the thermal camera showed a similar pattern to that of the multispectral camera (Figure 3.14). The pixel's temperature can have up to 3 °C difference without vignette correction. Unlike the multispectral camera, whose vignetting mainly results from the lens, the vignetting of an uncooled thermal camera can be caused by the differences in each pixel's response to the irradiance, which depends on the detector's temperature and is affected by the ambient temperature. In this case, multiple vignetting correction models need to be created for different ambient temperatures. In this study, we did not perform vignette

correction for the thermal camera. Instead, we only used the central portion of the image to measure the temperature since the vignette effect in the central portion is not significant. A more thorough study in the thermal camera's vignette correction was planned for future work.

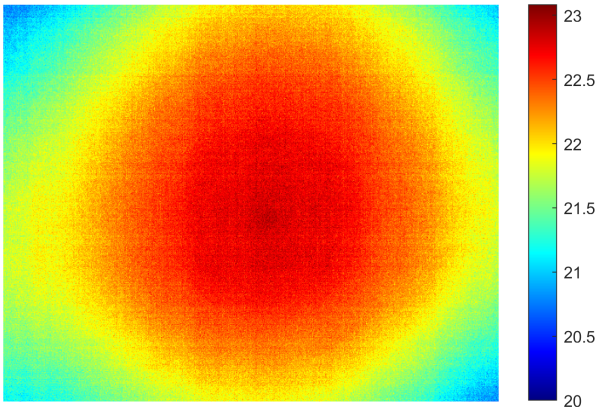

Figure 3.14: Vignette effect of the thermal camera. The color bar indicates the temperature in degree of Celsius.

3.6.2 Data preprocessing

Accuracy of the thermal camera calibration

During the flight, the air temperature and humidity were relatively stable but varied from 28 3°C to 31 7°C and from 35.7% to 40.5%, respectively (Figure 3.15). The transmittance of the atmosphere has a mean value of 0.9249, with a standard deviation of 0.0011. Because the transmittance is close to 1, the upwelling atmosphere radiation has little impact on the thermal image. After calibration, the temperature of the two calibration targets from the thermal image is highly correlated with the temperature measured by the RTD sensor with

a MAE of 1 02°C (Figure 3.16). The MAE was significantly reduced comparing to the MAE of 6.62 °C before calibration.

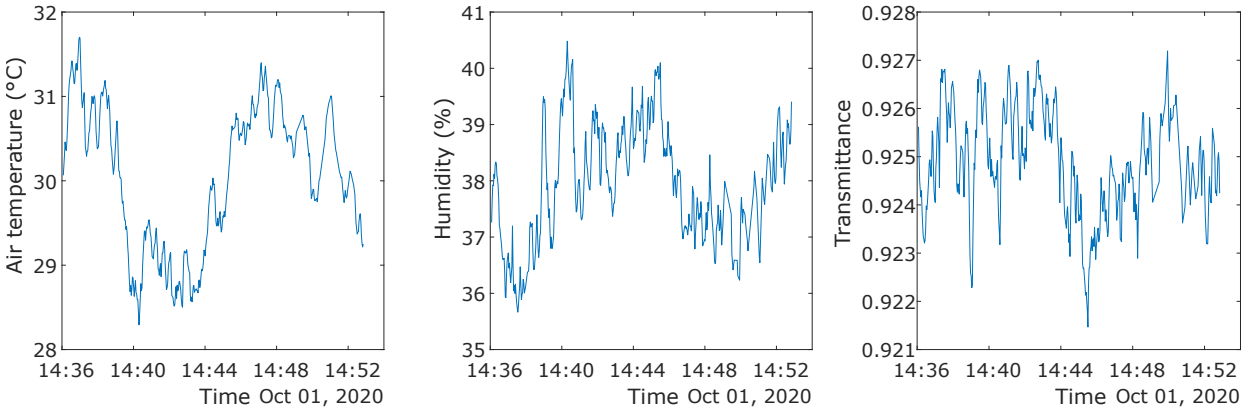

Figure 3.15: Atmospheric condition during the flight. The transmittance of the atmosphere is estimated using Equation 3.14.

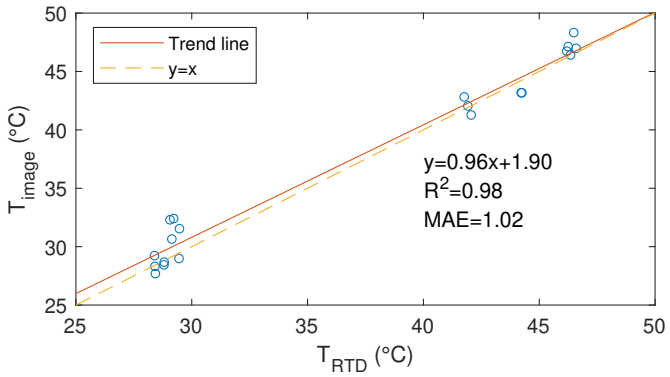

Figure 3.16: Correlation of the image measured temperature (T_{image}) and RTD measured temperature (T_{RTD}) for the thermal calibration targets.

Results of orthomosaic generation

With GCPs and image locations provided by the drone, the Metashape successfully generated georeferenced orthomosaics for color, multispectral, and thermal images 3.17. All the orthomosaics and DEM were registered based on the geoinformation. Although each camera has a different field of view and image resolution, the resulted GSD is about 0.016 m for all

of the cameras. Because the color camera has the largest FOV, followed by the multispectral camera and thermal camera, the color orthomosaic had the largest coverage, which covered area outside of the cotton field. Therefore, the FOV of the color camera can be reduced (by increasing the focal distance) to increase the GSD without reducing the coverage of the field.

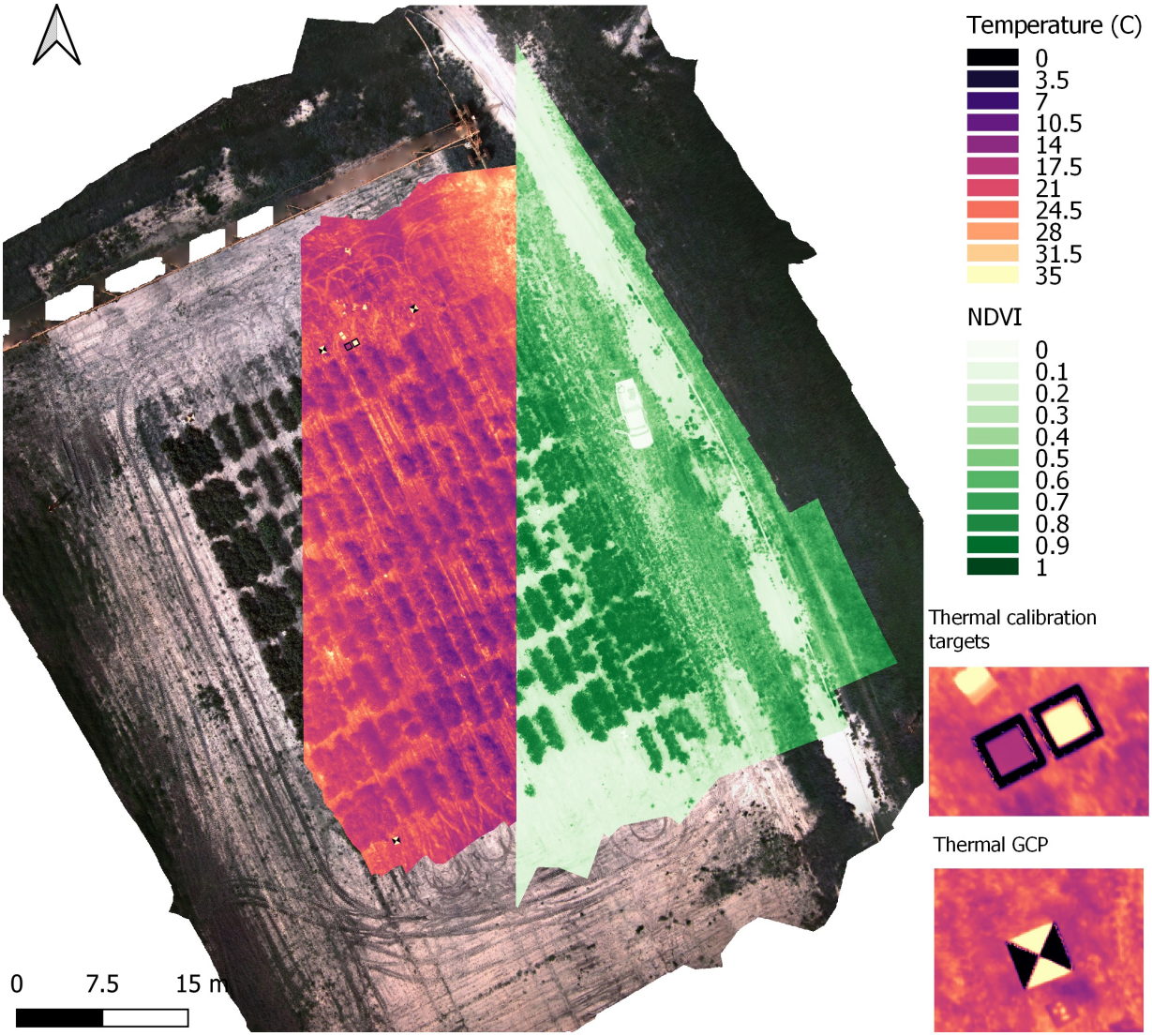

Figure 3.17: The orthomosaic overlays of color (left), temperature (middle), and NDVI (right). A zoom-in image of the thermal GCP and calibration targets were shown on the right.

3.6.3 Phenotypical traits extraction

Results of the canopy segmentation

Since the canopy's color is different from that of the soil, a simple thresholding method can segment the canopy very efficiently (Figure 3.18). For crops with a large canopy such as cotton, the plot height model can be used to remove weeds that are lower than a threshold (0.2 m for this study). Incorrect canopy segmentation can affect the accuracy of the phenotypical traits because the calculation of canopy phenotypical traits rely on the canopy segmentation.

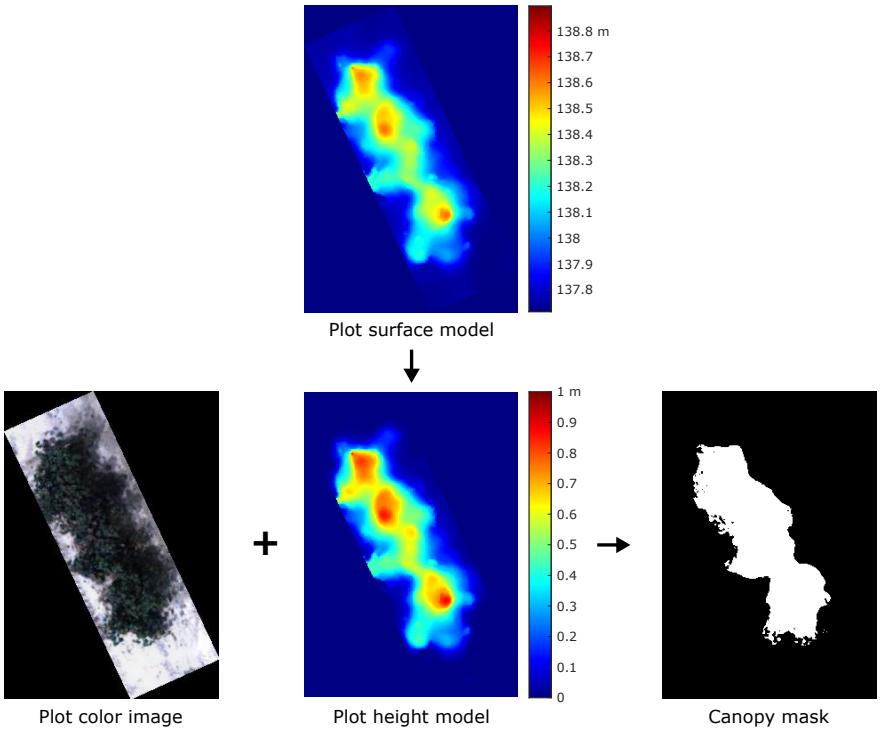

Figure 3.18: Illustration of the canopy segmentation result. The plot DSM was first subtracted by the ground plant to get the plot height model, and then the canopy mask was generated using the plot color image and the plot height model.

Accuracy of the canopy height

The canopy height extracted from the color images has a high correlation with the manual measurement. The image extracted maximum canopy height has an MAE of 0.1 m and lower value compared with the manual measurement, which is consistent with other studies in which the error of the canopy height is within the range of 0.07 m to 0.1 m using the photogrammetric method (Figure 3.19) [116, 111, 112]. The error of the canopy height largely depends on the accuracy of the DSM, which is affected by the GSD. Incorrect detection of the ground plane can also lead to large errors in generating the plot height model, which usually occurs at the enclosed canopy with little ground. In this case, the bare ground model should be used to detect the ground plane. Other environmental factors such as plant movement because of wind also contribute to the error.

The 99th percentile canopy height had the smallest MAE compared with the manually measured average canopy height (Table 3.3), but the median canopy height had the largest coefficient determination. It is hard to conclude which statistical metrics can better represent the true average canopy height with such a small validation dataset. The choice of statistic metric can be dependent on the type of crop and should be determined experimentally by collecting the validation dataset in the field.

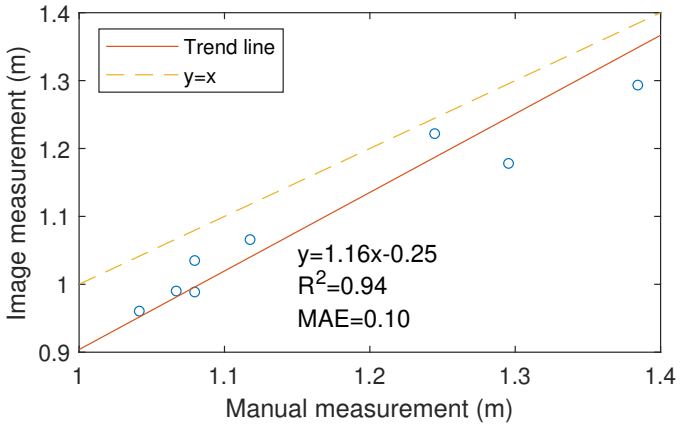

Figure 3.19: Correlation of the image measured maximum canopy height and the manually measured maximum canopy height.

Table 3.3: Comparing the manually measured average canopy height and extracted canopy height from images. H_{mean}: mean canopy height. H_{median}: median canopy height. H_{max}: maxminum canopy height. H_{np}: n-th percentile canopy height MAE: mean absolute error. R^2: coefficient of determination using linear model.

	H_{mean}	H_{median}	H_{max}	H_{50p}	H_{60p}	H_{70p}	H_{80p}	H_{90p}	H_{99p}
MAE (m)	0.501	0.531	0.100	0.531	0.462	0.395	0.308	0.196	0.060
R^2	0.742	0.799	0.691	0.799	0.651	0.543	0.481	0.619	0.766

Accuracy of the canopy vegetation index

The image extracted canopy NDVI had an MAE of 0.0518 with an MRA of 6.6% compared with the ground measured NDVI, which means the multispectral image can measure the vegetation index of the canopy accurately. Other studies have shown that the MicaSense Redege camera could provide accurate NDVI measurement compared to the GreenSeeker sensor [142]. The multispectral image can reveal the variations within the canopy, which is an advantage over a point-based sensor.

Data visualization

The extracted phenotypical were visualized in QGIS (Figure 3.20). The color represents the value, and the polygon outlines the canopy. The visualization is helpful for researchers to examine the data and present the results to general audiences. It is also helpful to visualize the variation between plots.

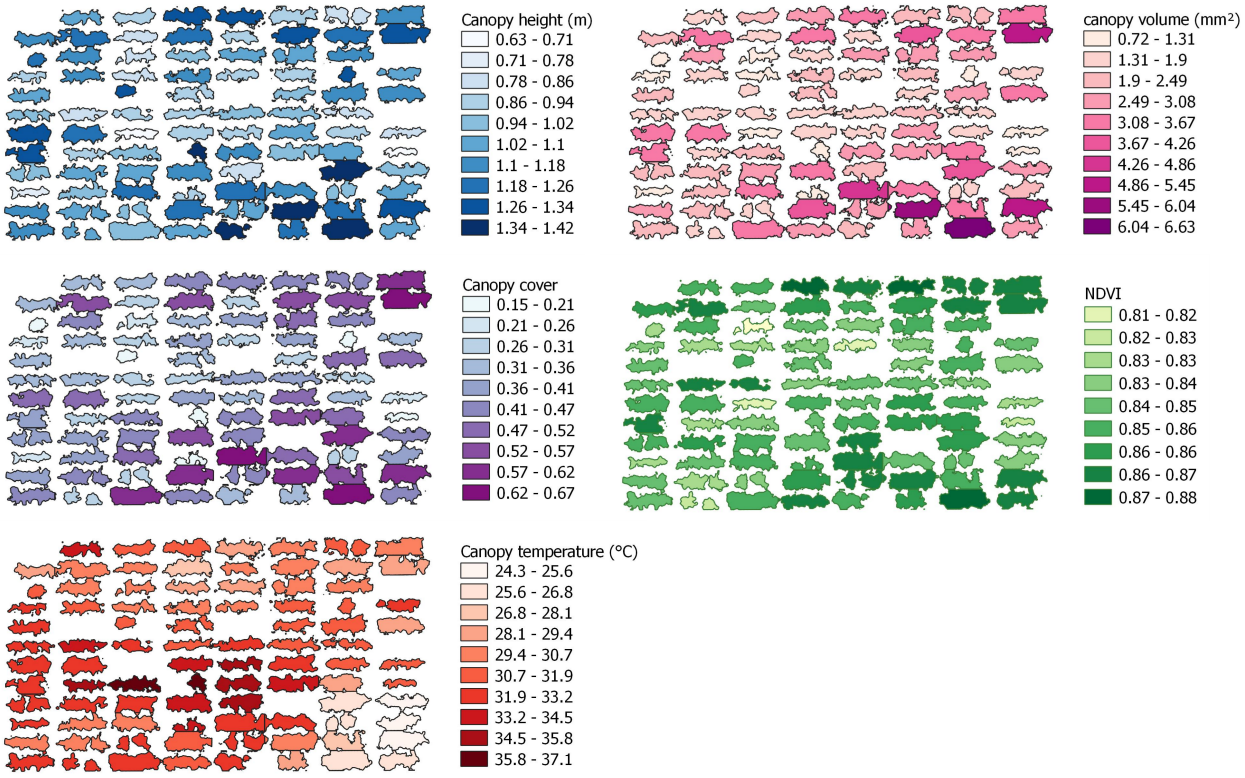

Figure 3.20: Visualization of the extracted phenotypical traits. The 99th percentile canopy height was used to represent the canopy height.

3.7 Discussion

UAV has been used widely for plant phenotyping, and color, thermal, and multispectral are the most commonly used sensors in the literature. The advantage of using a multi-sensor

system is that the sensors' rich information can measure more complex phenotypical traits than a single sensor. For example, the cotton yield can be estimated more accurately using multiple image features extracted from color, multispectral, and thermal images than a single image feature extracted from a single camera [126]. Different types of sensors can provide complementary information to improve the accuracy of the phenotypical traits. For example, a high-resolution color image can segment the canopy more accurately, and can be used to calculate the canopy temperature from the low-resolution thermal images more accurately. The downside of the multi-sensor system is the relatively high payload requirement of the UAV to carry multiple sensors simultaneously. Therefore, our DAS was designed so that each sensor can be attached/detached to the DAS, so a low-payload drone can be used when only certain sensors are used. In this way, the UAS can be customized to save cost on the sensor and drone.

The data processing pipeline proposed in this study can extract basic phenotypical traits from color, multispectral, and thermal camera, and can be used to inference more complicated phenotypical traits. For example, the morphological traits can be used to monitor the growth of the crop and estimate the yield. The vegetation index and canopy temperature can be used to evaluate the water status of the plant. The data processing pipeline used to extract phenotypical traits was implemented in Python and QGIS, in which it is easy to integrate other methods and visualize the results.

3.8 Conclusions

This paper designed a multi-sensor unmanned aerial system that can be used for plant phenotyping. Field data were collected in a cotton field using the system, and phenotypical traits at the plot level, including canopy height, canopy cover, canopy volume, canopy vegetation index, and canopy temperature, were extracted using the proposed data processing pipeline. The thermal camera and multispectral camera were calibrated in the lab and field. The temperature from the thermal image had a MAE of 1 02°C after field calibration. The canopy NDVI measured by the multispectral image had a MRE of 6.6% comparing to the ground measurement. The maximum canopy height had an error of 0.1 m compared to manual measurement, similar to other studies.

The design of the UAS was open-sourced and can be used by other researchers. We will continue to improve the system by adding an RTK-GNSS and IMU to the system so the LiDAR scans can be correctly stitched. A data management system that integrates data storing, data processing, and data visualization was planned for future work.

Chapter 4

Multispectral Imaging and Unmanned Aerial Systems for Cotton Plant Phenotyping[1]

[1]Xu, Rui, Changying Li, and Andrew H. Paterson. "Multispectral imaging and unmanned aerial systems for cotton plant phenotyping." *PloS one* 14, no. 2 (2019): e0205083. Reprinted here with permission of the publisher.

Abstract

This paper demonstrates the application of aerial multispectral images in cotton plant phenotyping. Four phenotypic traits (plant height, canopy cover, vegetation index, and flower) were measured from multispectral images captured by a multispectral camera on an unmanned aerial system. Data were collected on eight different days from two fields. Ortho-mosaic and digital elevation models (DEM) were constructed from the raw images using the structure from motion (SfM) algorithm. A data processing pipeline was developed to calculate plant height using the ortho-mosaic and DEM. Six ground calibration targets (GCTs) were used to correct the error of the calculated plant height caused by the geo-referencing error of the DEM. Plant heights were measured manually to validate the heights predicted from the imaging method. The error in estimation of the maximum height of each plot ranged from -40.4 to 13.5 cm among six datasets, all of which showed strong linear relationships with the manual measurement ($R^2 > 0.89$). Plot canopy was separated from the soil based on the DEM and normalized differential vegetation index (NDVI). Canopy cover and mean canopy NDVI were calculated to show canopy growth over time and the correlation between the two indices was investigated. The spectral responses of the ground, leaves, cotton flower, and ground shade were analyzed and detection of cotton flowers was satisfactory using a support vector machine (SVM). This study demonstrated the potential of using aerial multispectral images for high throughput phenotyping of important cotton phenotypic traits in the field.

4.1 Introduction

To meet the demands of the predicted global population of 9 billion by the year 2050, the crop production must double from 2010 to 2050 [1]. This is a tall order, challenging plant breeders to find genotypes with high yield, as well as high-stress tolerance to adapt to the changing climate in the next 30 years. Recent technological advances in molecular biology have offered tools that can significantly accelerate the breeding process [143]. However, phenotyping has become the bottleneck to using these new technologies to their full potential. Screening genotypes — in order to select those with the most desirable traits — heavily relies on the ability to characterize and measure traits [144]. Therefore, many plant breeders and engineers recognize the need for a high-throughput phenotyping (HTP) system capable of efficiently and accurately measuring phenotypic traits [144, 101].

The development of an HTP system is challenging in both the platform design and the associated data processing methods. Development of a field-based high-throughput phenotyping system (FHTPS) is even more challenging due to heterogeneous field conditions and uncontrolled environments, which can affect the data quality and make results difficult to interpret [101]. Some FHTPSs utilize ground vehicles (either tractors or robotic platforms) equipped with sensors to acquire data [5, 145, 11]. Ground vehicles have the advantage of carrying large payloads and can easily include multiple sensors at a time. In addition, ground platforms can control the data collection environment (such as light conditions) to some degree with well-designed enclosures, thus guaranteeing data quality. However, ground platforms also have several disadvantages: the data collection speed is low, frequent data

collection can cause soil compaction, the wheels can damage the crop, and the platform is difficult to adjust for a wide range of crops once the design is fixed. As an alternative to ground platforms, unmanned aerial systems (UAS) can address the disadvantages of ground platforms to some degree. Compared to ground platforms, UAS can provide superior data acquisition speed and larger spatial coverage [103]. Since no interaction exists between the plots and the UAS, UAS can be easily adapted to different types of crops and different growth stages [140]. Furthermore, UAS can be automatically controlled by its onboard autopilot system, and therefore requires less human intervention during data collection.

Traditional precision agriculture and remote sensing studies have shown the broad applications of satellite or airborne images for crop management, stress detection, and yield estimation [146]. The general research methodology and data analysis techniques from remote sensing can be readily used for aerial images with high spatial and temporal resolutions taken by UAS, which can significantly benefit high-throughput phenotyping research. Simple traits such as plant height and canopy cover can be measured using aerial imaging to monitor crop growth and estimate the final yield. Plant height measured from crop surface model generated by aerial color images was used to develop regression models to predict the biomass for barley and the best model achieved a relative error of 54 04% [108]. The regression model was improved by combining vegetation indices from ground-based hyperspectral reflectance data, achieving 44 43% relative error on biomass prediction [109]. Spectral response of the canopy — measured using aerial hyperspectral or multispectral imaging — can be used to detect biotic and abiotic stress of the plant [147]. It was shown that Huanglongbing-infected citrus trees can be detected using various vegetation indices with 85% classification accuracy [148].

Tomato spot wilt disease in peanuts was best detected by normalized difference red edge (NDRE) using multispectral imaging [115]. Previous studies showed that aerial hyperspectral or multispectral imaging was useful to detect water stress [149, 150]. Normalized difference vegetation index (NDVI) showed highest correlation ($R^2 = 0.68$) with water potential in grape vineyard among seventeen vegetation indices, which showed NDVI could be a good indicator for long-term water deficits [151]. NDVI can also be used to separate olive tree crown from the soil in aerial images [110]. Canopy temperature is an indicator of water stress, but it is extremely sensitive to small changes in the environment, making the ability of the UAS to quickly and repeatedly measure many plots preferable to ground platforms for measuring this trait [119, 152]. Traits that require continuous measurements, such as flowering time, are also well suited to aerial imaging.

Although applications of UAS in agriculture have been studied for a wide range of crops, only a few applications of aerial imaging for cotton have been reported despite the importance of cotton as an industrial crop for producing natural fibers. One study used low-altitude aerial color images to estimate plant height and yield and they found that the cotton unit coverage can be better correlated with yield than plant height [153]. Another study used aerial color images to monitor cotton growth by measuring plant height and canopy cover and estimate the yield from them using various regression models [154]. The result showed good correlation between plant height and canopy cover, but a low correlation between the estimated yield and observed yield ($R^2 = 0.5$ for the best model). Due to the lack of manual measurement, the accuracy of the plant height and canopy cover was unknown in this study. Despite the two relevant studies, the use of aerial images to measure other important phenotypic traits such

as flowering and boll opening has received little attention. There is still a gap in knowledge and technical development that hinders capitalizing the latest UAS and imaging technologies for cotton breeding.

To explore the potential of cotton phenotyping using UAS, this paper focuses on measuring multiple traits including plant height, canopy cover, vegetation index, and flower detection using a multispectral camera. Specifically, this paper develops a method to measure cotton plant heights with a UAS-based FHTP system, validating the accuracy of the method with manual measurements. Canopy cover, the proportion of the ground covered by the crop canopy, will be derived based on the crop surface model and the normalized differential vegetation index (NDVI). The feasibility of flower detection based on the spectral response will be explored, providing a foundation for future quantification of the distribution of flowering over the growing season.

4.2 Materials and Methods

4.2.1 Test fields

The study was carried out on Plant Science Farm at University Georgia, Athens campus and the owner of the land gave permission to conduct the study on this site. Furthermore, the field studies did not involve endangered or protected species. Two test fields were used in this study, both of which located at the University of Georgia Plant Sciences Farm (33°52′2.8″N, 83°32′39.5″W) in Watkinsville, GA (Figure 4.1). A total of 240 plots of cotton were planted in field 1 on June 15, 2015, arranged in 20 columns and 12 rows, with a 1.8 m alley between

each column. Each plot was 3 m long and 0.9 m wide, and 15 seeds were planted in each plot. Six genotypes were used in field 1. Genotype 1 to 5 were provided by a cotton breeder from the University of Georgia, Tifton Campus. They were GA2011158, GA230, GA2010074, GA2009037 and GA2009100, respectively. Genotype 6 was Americot UA48. Each column had twelve plots and each genotype with two replicates were randomly assigned to each column. The germination rate of the plots varied from 13% to 100%, resulting in variance in the plot density.

Due to lack of high accurate surveying tool, no ground control point was used to correct the aerial triangulation error of the crop surface model. Instead, six round stainless steel plates with a diameter of 30 cm, which were referred as ground calibration targets (GCTs), were raised around 1.5 m above the ground using plastic pipes and placed around the border of the test field. The GCTs were painted in black patterns on white background (Figure B.1). Those GCTs were used to evaluate and calibrate the crop surface model. To avoid the power lines over the field during data collection, only 48 plots (from column 7 to column 10) were selected as test plots. However, the plants from row 1 to row 6 were seriously damaged by a tractor after October 7. Therefore, only 24 plots from row 7 to row 12 were used for data analysis since October 7.

Because of late planting, plants in field 1 did not produce many flowers over the season. Therefore, we used field 2 to collect cotton flower images. Field 2 had 288 plots arranged in 16 columns and 18 rows and it was based on complete random block design. We selected 22 plots to develop and validate the method for cotton flower detection.

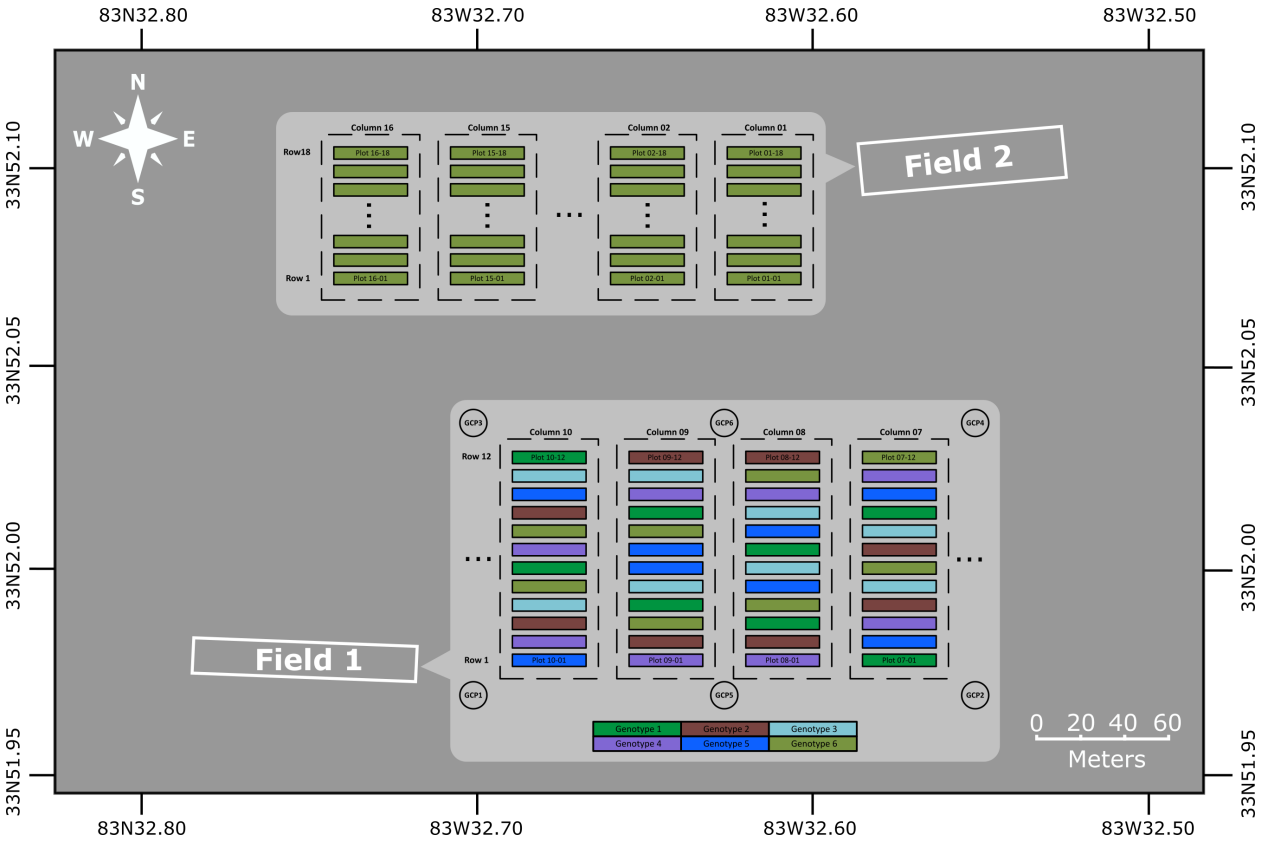

Figure 4.1: Location and plot layout of the two test fields. The genotypes of the plots in field 1 were indicated in different colors.

4.2.2 Image collection

Aerial multispectral images of the two fields were acquired using an octocopter (s1000+, DJI, China) carrying a lightweight multispectral camera (RedEdge, MicaSense, WA). The multispectral camera was mounted on a gimbal and faced the ground. The multispectral camera has five global shutter camera modules that provide five band images — blue, green, red, near-infrared (NIR), and red edge (RE). The center wavelengths of each band are 475 nm, 560 nm, 668 nm, 840 nm and 717 nm, respectively. The multispectral camera has a low accuracy GPS (2.5 m) which can record the geographic coordinates of the image. The

size of the band image is 1280×800. In order to validate flower detection results from multispectral images, aerial color images of field 2 were acquired (at the same time as the multispectral images) using a color camera (Lumix DMC-G6KK, Panasonic, Japan) that provides a maximum image size of 4624×3472.

To collect the aerial multispectral imaging data, the drone flew autonomously along a preset flight path at a speed of 2.5 m/s and a height of 20 m above ground level (AGL) for field 1, achieving ∼1.5 cm ground sample distance for multispectral images. The endlap and sidelap of the collected multispectral images were over 85%. To image cotton flowers as closely as possible, while maintaining the same flight speed, we set the flight height at 15 m AGL for field 2, which was the minimum height that downdraft of the UAV did not disturb the plants and images had enough overlap. In reality, the drone flew lower than 15 m due to the wind condition and error of altitude sensor, which gave the actual ground sample distance of ∼0.8 cm for the multispectral images. The endlap was over 80% and sidelap was over 50% for the multispectral images. The color and multispectral cameras were triggered simultaneously at the frequency of 1 Hz with a customized triggering circuit. The exposure time and aperture were manually set for the color camera according to the light conditions of the field to get the best image quality and other parameters used auto-settings. At each data collection, a reflectance panel provided by MicaSense was imaged before and after the flight. In total, we collected one set of images for the field 2 and seven sets of images for the field 1 (Table 4.1). The field 2 images were used to explore the feasibility of flower detection using multispectral images, while the field 1 images were used to calculate plant heights, canopy cover, and vegetation index.

Table 4.1: Data collection summary.

Field	Date	Time	Flight height	Flight speed	Flight time
Field 2	8/28/2015	12:00 PM	15 m	2.5 m/s	6 min
	9/18/2015	1:00 PM	20 m	2.5 m/s	5 min
	9/30/2015	12:00 PM	20 m	2.5 m/s	6 min
	10/7/2015	4:00 PM	20 m	2.5 m/s	5 min
Field 1	10/16/2015	11:00 AM	20 m	2.5 m/s	5 min
	10/19/2015	2:00 PM	20 m	2.5 m/s	5 min
	10/23/2015	10:00 AM	20 m	2.5 m/s	5 min
	10/30/2015	10:00 AM	20 m	2.5 m/s	5 min

4.2.3 Reference measurement

On each data collection day (except September 18, 2015), the height of each plot in the field 1 was manually measured as the ground truth. For each plot, individual plants were measured using a ruler and the height of the tallest plant within the plot was used as the maximum plot height. All 48 plots were measured on September 30 and October 7, and 24 plots (from rows 7 through 12) were measured on the other four days. The vertical distance between the edge of the GCT to the ground was measured four times using a ruler with millimeter accuracy at even spacing along the edge of the GCT and the average value of the four measurements was used as the GCT height reference. The manual measurements of plant height were used to calculate the accuracy of results from aerial images.

4.2.4 Plot height, canopy cover and vegetation index extraction

Multispectral images from the field 1 were used to calculate plot height, canopy cover, and vegetation index. The data processing flowchart consisted of five steps (Figure 4.2). The

first step was to perform aerial triangulation and generate the digital elevation model (DEM) and ortho-mosaic. Two software/services were used in this step: PhotoScan and MicaSense ATLAS. Both of them can generate DEM and ortho-mosaic. The PhotoScan (PhotoScan Professional 1.2.6, Agisoft LLC, Russia) can generate better DEM with higher resolution than MicaSense ATLAS (atlas.micasense.com). However, the ortho-mosaic from PhotoScan is not radiometrically calibrated, resulting in incorrect vegetation indices. In contrast, the ortho-mosaic from ATLAS is radiometrically calibrated using the images of the reflectance panel. Therefore, we used the DEM generated from PhotoScan, and the ortho-mosaic from ATLAS for the subsequent data processing. The processing time for each step in PhotoScan was summarized in Table 4.2.

Table 4.2: Summary of the image processing time for generating ortho-mosaic and DEM in PhotoScan. Dense cloud and DEM was not built for 8/28 dataset since the ortho-mosaic was used for flower detection.

Dataset	**Images**	**Align photos**	**Build dense cloud**	**Build ortho-mosaic**	**Build DEM**	**Total time**
8/28	230	55 min, 14 sec	-	1 min, 46 sec	-	57 min
9/18	178	39 min, 55 sec	24 min, 57 sec	1 min, 46 sec	6 sec	66 min, 44 sec
9/30	171	41 min, 26 sec	31 min, 10 sec	1 min, 17 sec	6 sec	73 min, 59 sec
10/07	134	30 min, 8 sec	13 min, 31 sec	40 sec	6 sec	44 min, 25 s
10/16	137	31 min, 36 sec	19 min, 24 sec	49 sec	5 sec	51 min, 54 sec
10/19	135	29 min, 28 sec	15 min, 27 sec	52 sec	6 sec	45 min, 53 sec
10/23	114	22 min, 34 sec	9 min, 40 sec	2 min, 15 sec	6 sec	32 min, 35 sec
10/30	132	28 min, 13 sec	12 min, 35 sec	2 min, 20 sec	6 sec	43 min, 14 sec

The second step was to align the PhotoScan DEM and ATLAS mosaic using OpenCV (OpenCV 2.5). The scale-invariant feature transform (SIFT) image features were first calculated for each band image of the PhotoScan and ATLAS ortho-mosaics and their common image features were matched. Then the projective transformation from ATLAS ortho-mosaic

to PhotoScan ortho-mosaic was calculated based on the locations of the common feature points. Finally, the ATLAS ortho-mosaic was transformed to the image frame of the PhotoScan ortho-mosaic, which is same as the PhotoScan DEM, using the projective transformation. Then the ATLAS ortho-mosaic and the DEM were aligned pixel by pixel.

The third step was plot segmentation, which was to divide the entire DEM and ortho-mosaic into individual plots (plot DEM and plot image) based on plot length and width. As a result, each plot-level DEM or ortho-mosaic has only plants and the surrounding soil. The area of the plot DEM and plot images were kept the same. The pixel value of the DEM indicates the altitude above sea level, therefore, the fourth step was to adjust plot DEMs so that the pixel value represented the relative height from the ground. For this purpose, the maximum likelihood estimation sample consensus (MLESC) algorithm was used on the DEM to find the best fitting plane to represent the ground surface since the ground within one plot can be assumed to be flat [136]. The MLESC was applied to the ground pixels whose normalized intensity on the red band image was larger than 0.4. The threshold was chosen arbitrarily but the misclassified ground pixels due to the threshold will not affected the ground surface significantly because the MLESC is robust to outliers. The threshold of MLESC was set to 0.1 m. The ground surface was subtracted from the DEM to get the relative height. Meanwhile, the vegetation index for each plot was calculated. Several vegetation indices can be derived from the five bands of the multispectral images, such as normalized difference vegetation index (NDVI) and normalized difference red edge (NDRE). A complete list of vegetation indices can be found in Torino et al.'s study [139]. In our study, only the NDVI was calculated using the red (R) and near-infrared band (NIR) using equation

4.1.

$$NDVI = \frac{NIR_{840} - R_{668}}{NIR_{840} + R_{668}} \tag{4.1}$$

The subscript indicates the wavelength (nm).

The fifth step was to separate the canopy from the ground using plot height and plot NDVI by a threshold method. A threshold of 0.5 for NDVI was used to separate the vegetation from the soil [155] and a threshold of 20 cm for DEM was used to remove weeds. The maximum plot height was calculated as the largest height value within the canopy. When calculating the maximum plot height, only the central 75% of the canopy was used to avoid pixels from nearby plots. The canopy cover was calculated as the ratio of the number of canopy pixels to the total pixels of the plot. The NDVI of the plot was calculated using the mean NDVI value of the canopy.

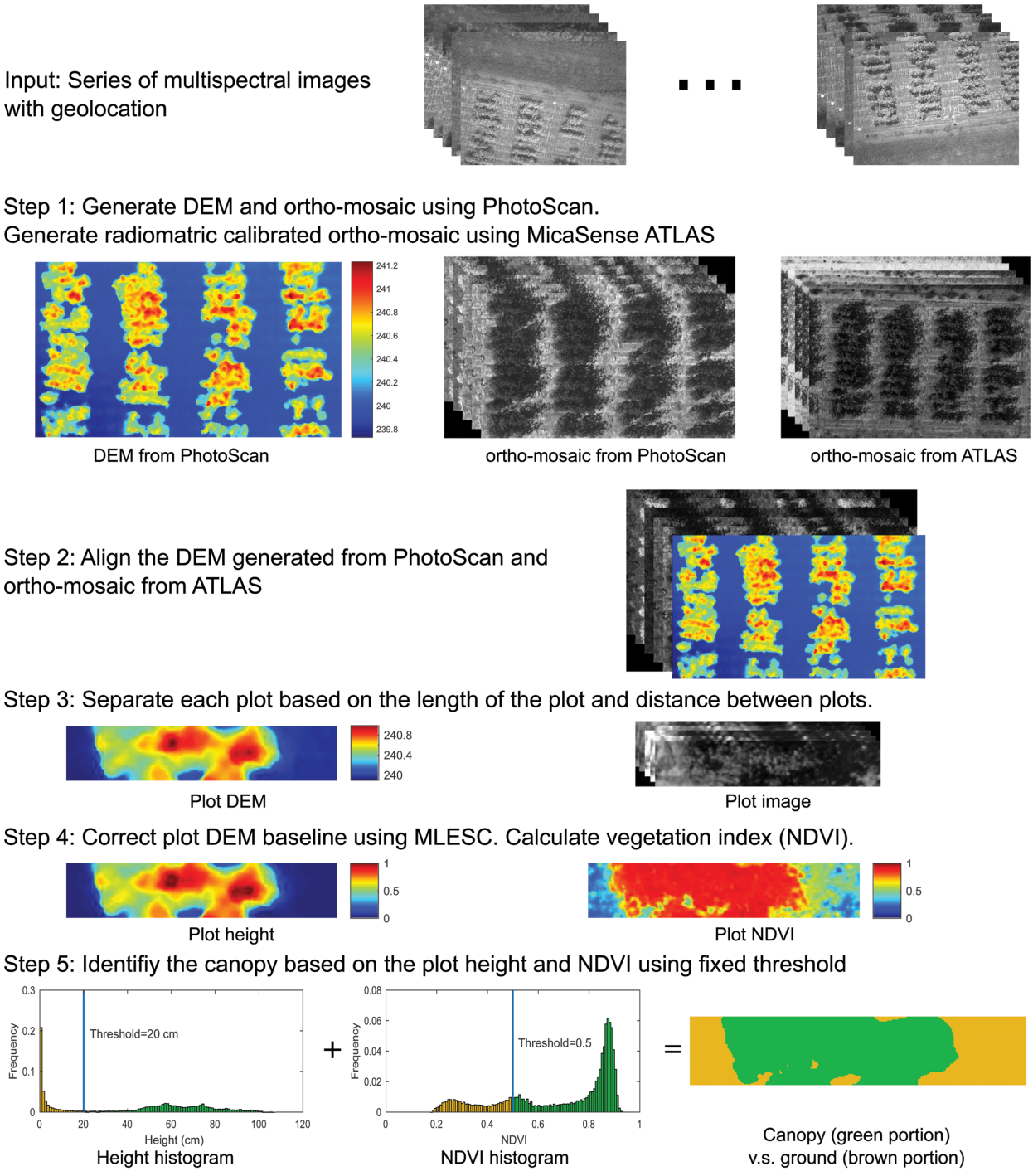

Figure 4.2: Data processing flowchart. DEM: digital elevation model. MLESC: maximum likelihood estimation sample consensus. NDVI: normalized difference vegetation index.

4.2.5 GCT height

For each dataset, the height of each GCT was calculated from the DEM following the same procedure as the plot height in order to evaluate the accuracy of the model and height calculation algorithm. However, due to the shape of the GCT, the last step was replaced by calculating the average GCT pixel values as the height of the GCT.

4.2.6 Flower detection

Flower detection was performed using the field 2 image set. The ortho-mosaic was first generated using PhotoScan and was divided into plot images using the same procedure as in the height calculation. For each plot image, each pixel was classified into four different categories (flower, canopy, ground, ground shade) based on the raw pixel value using support vector machine type 1 (C-SVM) [156]. The SVM can return a categorical label and probability of being in each category. Using the criteria of the probability being in flower category larger than 0.75 rather than the categorical label to classify a pixel as a flower pixel was found helpful to prevent misclassifying some leave spots with high reflectance as flower pixels. The flower pixels generated a flower mask for each plot. The number of flowers was obtained by counting the connected components in the flower mask. The SVM was trained using the raw digital count of the pixels manually selected from the images for each category. The penalty of the C-SVM was set to 1, and the kernel function was Gaussian radial basis function with a gamma value of 0.2. The classification accuracy of the SVM was examined using four-fold

cross validation. The number of flowers detected from the multispectral image was compared with the results of manual identification using the corresponding color image.

4.2.7 Statistical analysis

The error of the maximum plot height was calculated by the difference between the calculated heights (CH) and the measured height (MH). Root mean square error (RMSE) and relative root mean squared error (R-RMSE) was calculated for each dataset using equations 4.2 and 4.3. Linear regression between the measured height and calculated height was performed and the coefficient of determination (R^2) was calculated for each dataset. All the statistical analysis were done in MATLAB (2016a, MathWorks).

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(CH_i - MH_i)^2} \tag{4.2}$$

$$R\text{-}RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{CH_i}{MH_i} - 1\right)^2} \tag{4.3}$$

Where N is the total number of plots.

4.3 Results

4.3.1 GCT height

The accuracy of calculated plot heights was affected by the accuracy of the DEM. The heights of GCTs were used to examine the accuracy of the DEM because the rigid body of the GCTs

was less prone to error than plants. The error of the GCT height varied for different date and among the GCTs (Figure 4.3A). The mean errors for 9/30, 10/7, 10/16, 10/19, 10/23 and 10/30 were −9.8 cm, −9.4 cm, −7.6 cm, −8.7 cm, −8.3 cm and −6.7 cm, respectively. This suggested that there were systematic errors in the DEM that vary by day. The systematic errors resulted from the georeferencing error of the images and triangulation error during dense point cloud generation. The common practice for error correction is to use accurately surveyed ground control points to correct the error introduced by inaccurate georeferencing of the aerial images, or to scale the DEM using a reference object with its true size. The second method was adopted to correct the systematic error by calculating the scale factor between the calculated GCT height and the measured height. After error correction, the errors of the GCT heights were reduced to −2.6–3.2 cm (Figure 4.3B).

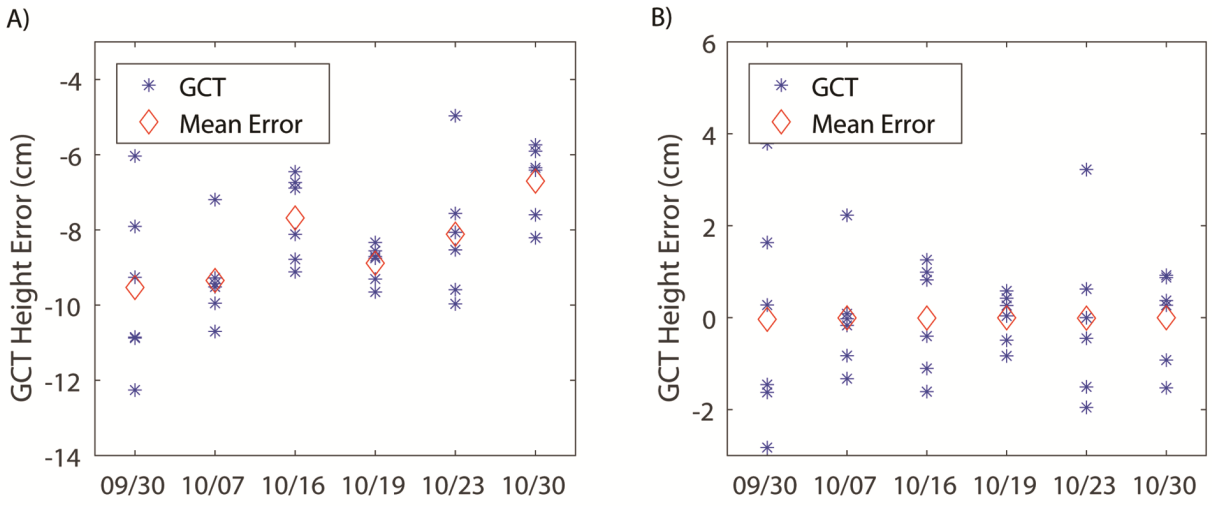

Figure 4.3: Accuracy of height measurements of the ground calibration targets by the imaging method. A) The error in calculating ground calibration target heights before error correction. B) The error in calculating ground calibration target heights after error correction.

4.3.2 Maximum plot height

The calculated plant heights were adjusted using the scale factor from the GCTs. After correction, the mean errors of the maximum height for 9/30, 10/07, 10/16, 10/19, 10/23, and 10/30 were −4.2 cm, −4.3 cm, −7.1 cm, −8.0 cm, −5.6 cm, and −12.1 cm, respectively (Figure 4.4A). The mean of the relative error for each day ranged from -3.9% to -10.2% (Figure 4.4B). This suggested that the calculated maximum plot heights were consistently smaller than the reference height measured manually, which is consistent with the findings in Huang et al.'s study [153]. The errors for all data sets, ranging from -39.0 to 13.5 cm, approximately followed a normal distribution with a mean of -6.2 cm and a standard deviation of 7.5 cm (Figure 4.5). The coefficient of determination (R^2) between the calculated maximum plot heights and manually measured maximum plot heights was between 0.9 and 0.96, showing a strong linear relationship between calculated heights and the reference heights (Figure 4.6).

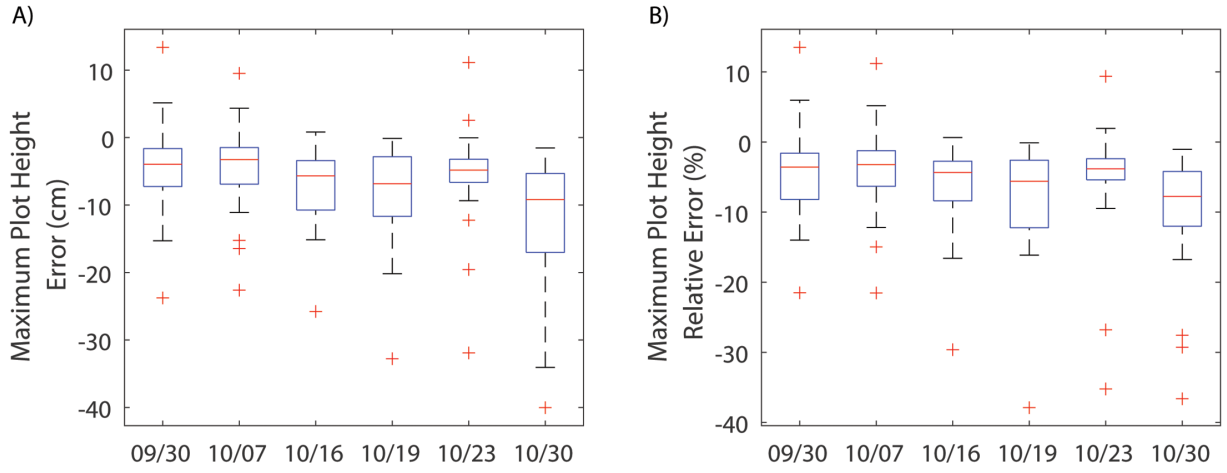

Figure 4.4: Error of the calculated maximum plot height. A) The absolute error in calculating maximum plot heights. B) The relative error in calculating maximum plot height.

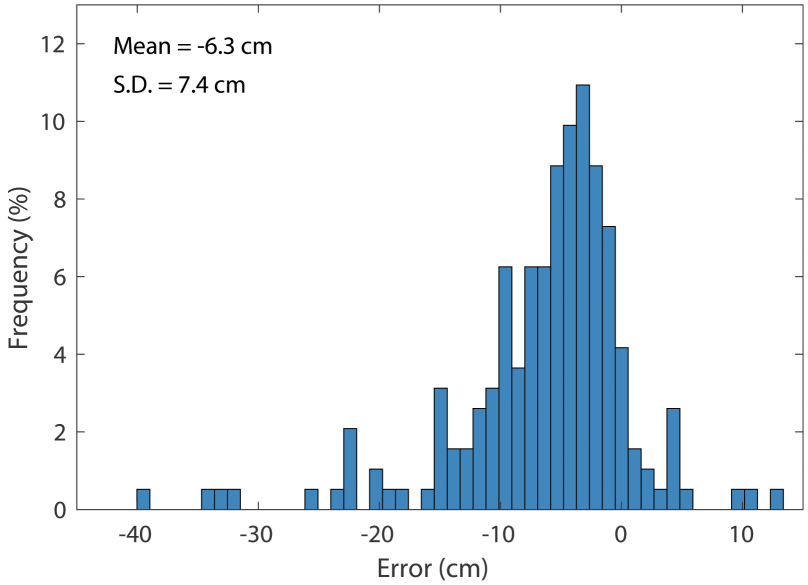

Figure 4.5: The error distribution for the maximum plot height.

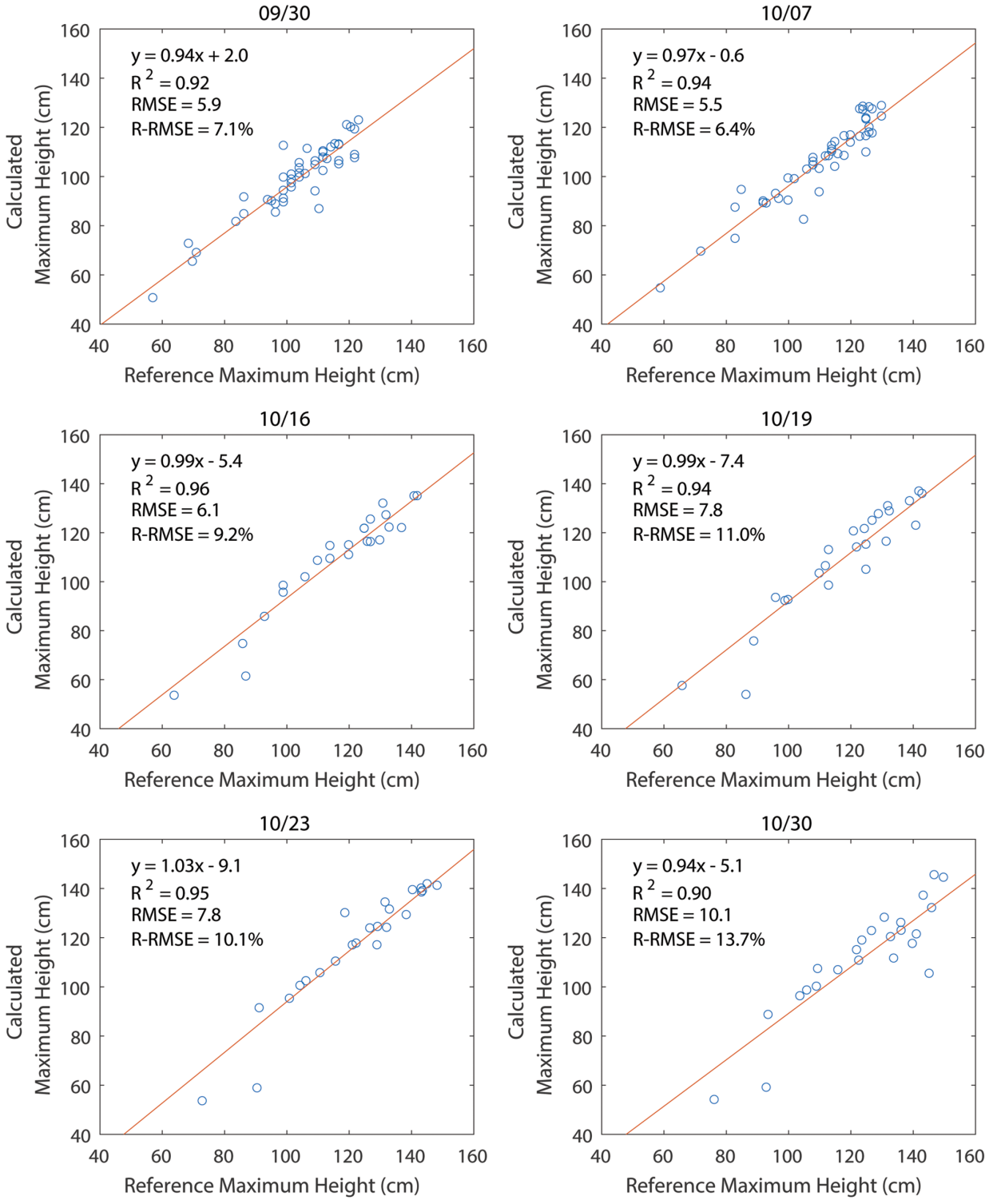

Figure 4.6: Correlation between calculated maximum plot heights and manually measured maximum plot heights using a linear model.

4.3.3 Canopy cover and vegetation index

Canopy cover indicates the proportion of the ground area covered by the canopy, which can be used to quantify canopy development over time (Figure B.2A). Ideally over time the canopy cover increased until reaching full coverage of the ground. Plots with larger canopy cover can capture more sunlight, which potentially can produce more cotton fiber. The NDVI usually increases as the canopy develops, reaching a maximum the canopy fully develops, then declines as the plant remobilizes resources into the bolls and starts to defoliate (Figure B.2B).

Studies have shown that NDVI has a high linear correlation ($R^2 > 0.6$) with leaf area index (LAI), which is an important index to quantify plant canopy [157]. Since NDVI and LAI are both correlated with canopy cover, a linear relationship was found between NDVI and canopy cover (Figure 4.7). The coefficient of determination(R^2) was low because the data collection was taken in the late vegetative stage, where correlation between LAI and NDVI was declined. Another reason of the low R^2 could be due to the late planting that caused the cotton plants to grow differently from normal growth. The linear relationship became weak after the canopy fully closed due to the defoliation of the leaves.

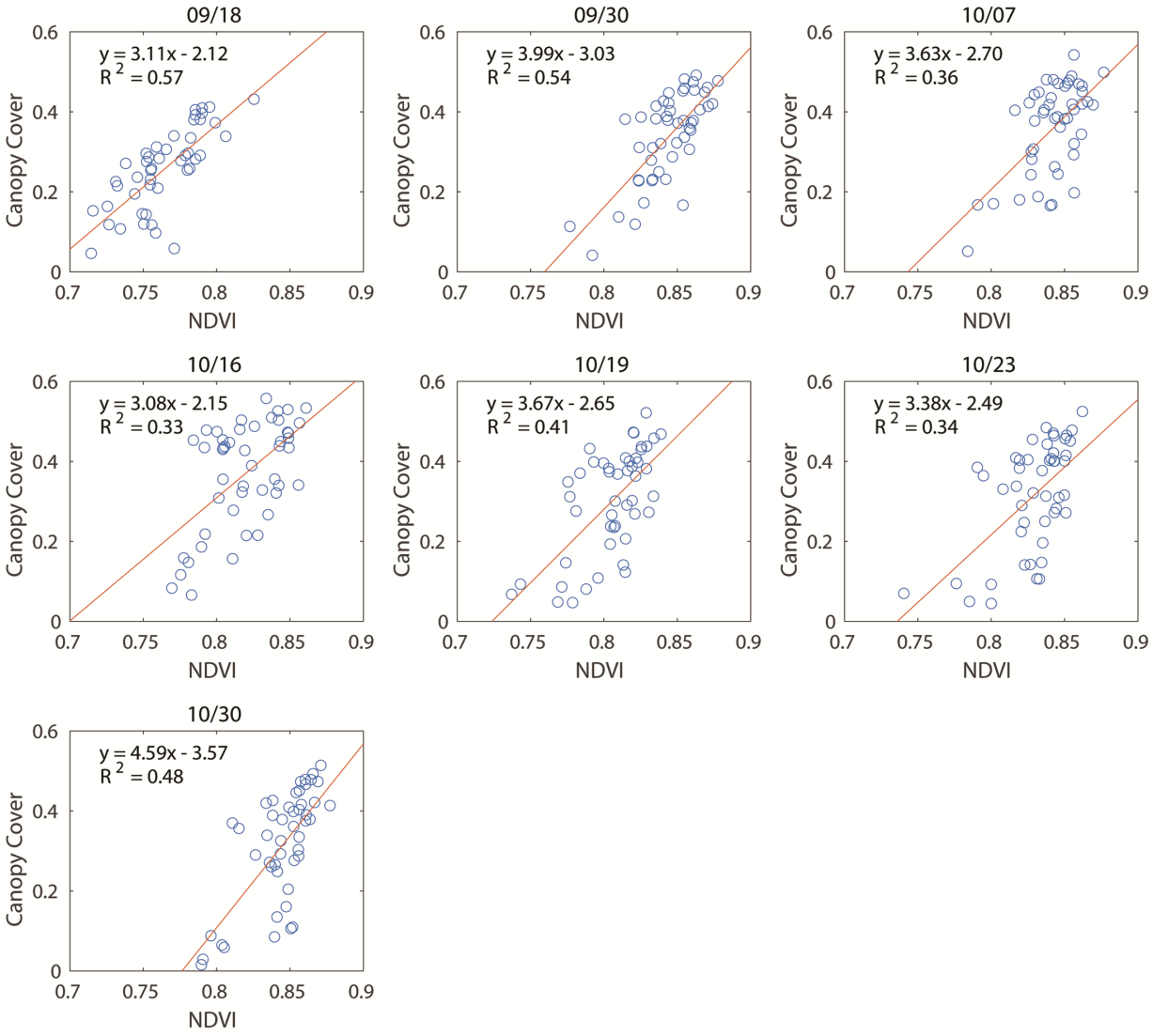

Figure 4.7: Correlation between NDVI and canopy cover on different dates.

4.3.4 Flower detection

Multispectral images showed the clear separation between flowers and other objects (canopy, ground and ground shade, Figure 4.8A). Since the multispectral images were taken under auto exposure for each channel, each band image always gave the best contrast between the plant and ground. Flowers showed the highest pixel value on the blue, green, and red channels;

therefore, flowers can be clearly separated from other objects using these three channels. The ground was clearly separated from other objects on the NIR channel. Because of the clear separation of the spectrum between different categories, we were able to train the SVM with 100% classification accuracy for flower and non-flower classes for the training set at the pixel level. The detection results from multispectral images showed few false negatives and false positives (Table 4.3). In a few cases, all the flowers were detected and match with the color images (Figure 4.9A), while in most case, the flowers cannot be detected with 100% accuracy for several reasons. First, the shadow of the leaves decreased the pixel value of flowers inside the canopy, and the SVM could not detect those flowers because their spectrum was closer to the canopy spectrum, which caused false negatives (Figure 4.9B). Second, leaves with high specular reflectance could be misclassified as flowers because they gave high pixel values, and thus caused false positives (Figure 4.9C). Third, because of the resolution limitation of the multispectral images, a cotton flower was only several pixels large in the images. This increased the risk of missing small flowers or flowers partly hidden by leaves. In Figure 4.9D, two partly hidden flowers were not detected in the multispectral image.

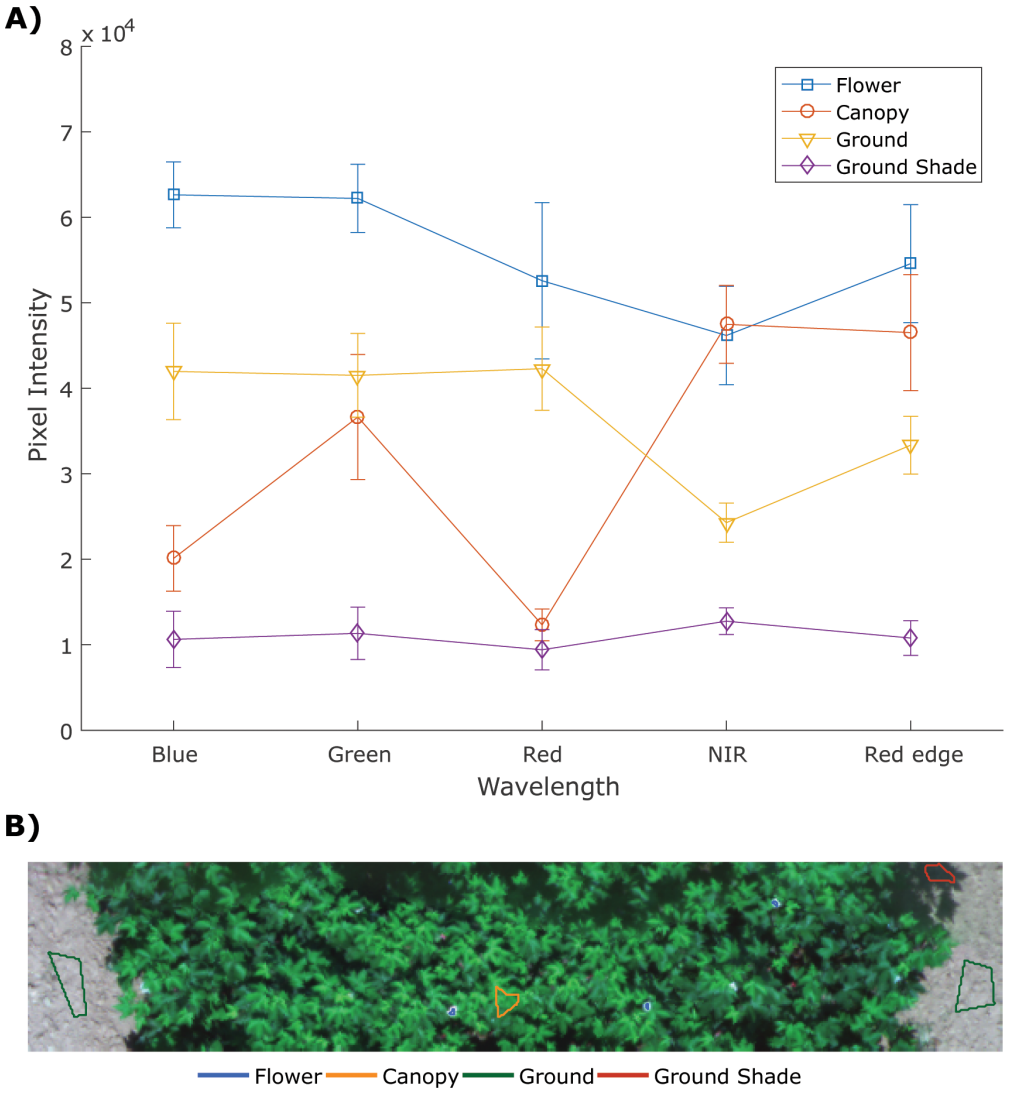

Figure 4.8: Spectrum of the objects in the field. A) Pixel values of four objects (flower, canopy, ground, and ground shade). B) The four regions of interest in the image.

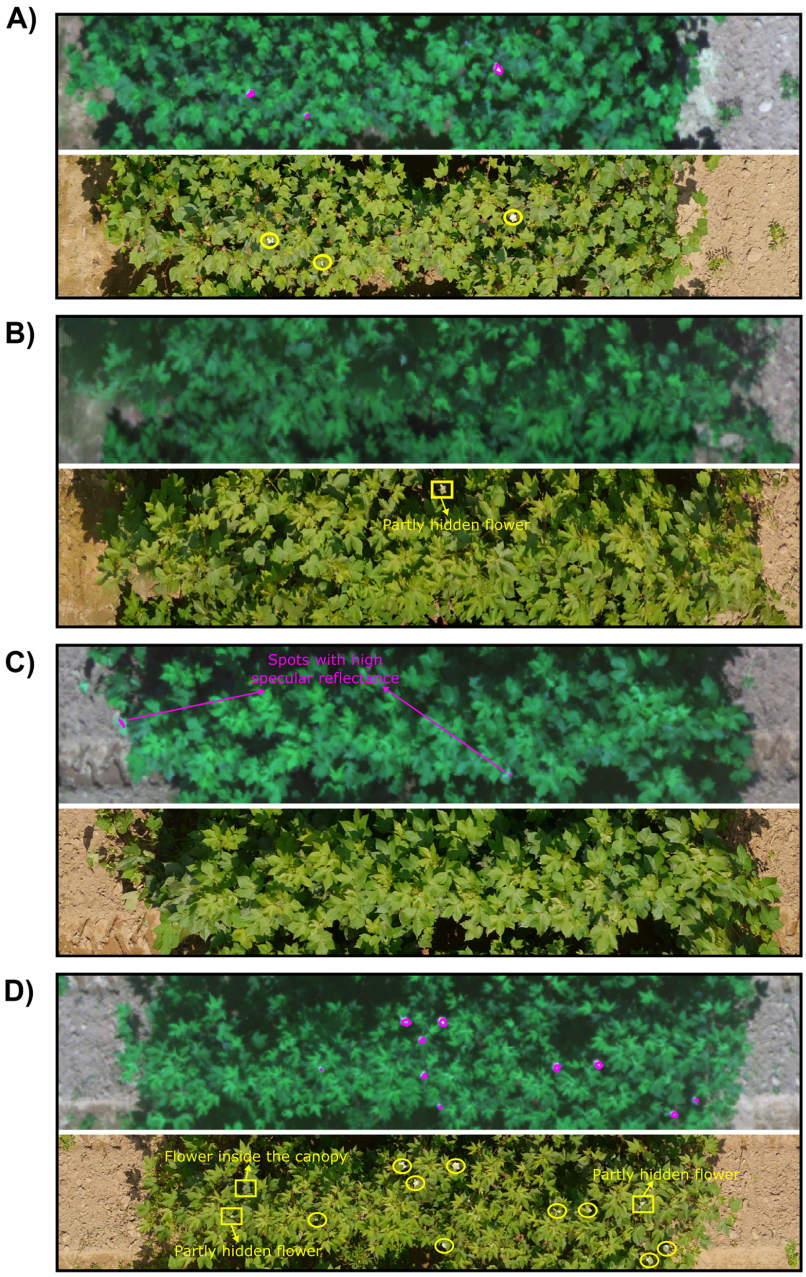

Figure 4.9: Example plots of the automatically detected flowers in the multispectral images (composited from the blue, red, and green band) and manually identified flowers in the color images. In the multispectral images (top image in each panel), the magenta lines indicate detected flowers. In the color images (bottom image in each panel), the yellow circles indicate detected flowers and the yellow squares indicate undetected flowers in multispectral images. A) All the flowers in the color image were detected in the multispectral image. B) The flower inside the canopy was not detected in the multispectral image. C) Spots with high specular reflectance from the leaves were misclassified as flowers in multispectral images while the color images showed no flowers. D) Most of the flowers in the color image were detected in the multispectral image. Two partly hidden flowers and one flower inside the canopy were not detected in the multispectral image

Table 4.3: Comparison of the flower detection using multispectral images with the manual detection using color images.

Plot	Flowers in color image	Flowers detected by multispectral image		
		False negative (percent rate)	False positive (percent rate)	True positive (percent rate)
1	4	1(25)	0(0)	3(75)
2	1	0(0)	0(0)	1(100)
3	4	3(75)	0(0)	1(25)
4	3	1(33)	0(0)	2(67)
5	4	1(25)	0(0)	3(75)
6	4	1(25)	1(25)	2(50)
7	4	1(20)	1(20)	3(60)
8	7	2(22)	2(22)	5(56)
9	5	1(17)	1(17)	4(67)
10	4	1(20)	1(20)	3(60)
11	5	3(60)	0(0)	2(40)
12	4	2(50)	0(0)	2(50)
13	4	0(0)	3(43)	4(57)
14	9	0(0)	3(27)	8(73)
15	4	2(33)	0(0)	4(67)
16	3	2(67)	0(0)	1(33)
17	1	0(0)	1(50)	1(50)
18	1	1(100)	0(0)	0(0)
19	5	1(20)	0(0)	4(80)
20	5	1(17)	0(0)	5(83)
21	5	0(0)	1(17)	5(83)

4.4 Discussion

This study demonstrated the success of using aerial multispectral images to measure plant height, canopy cover, and vegetation index, as well as the feasibility to detect cotton flowers. The UAS greatly improved the data collection throughput in the field and provided ability to continuously monitor cotton growth over the season.

In terms of accuracies for plant height measurement, the root mean squared error (RMSE) ranged from 5.7 cm to 10.4 cm (the relative root mean squared error (R-RMSE) ranged from 6.6% to 13.6%). Because of the lower ground sample distance, the accuracy is better than most of the studies on tree height measurement. One study achieved the RMSE between 20 and 45 cm (R-RMSE ranged from 6.55% to 19.24%) for olive tree height [110]. Another study measured single olive tree using multispectral imaging and the average error was 22 cm (6.32%) and 53 cm (15.55%) for 50 m and 100 m flight height, respectively [158]. Compared with similar studies on crops with lower height such as barley and wheat, the proposed method achieved a similar accuracy [108, 111]. The error of the height measurement is mostly contributed by the error of the DEM, which can be largely corrected using the GCTs. However, certain errors involved in the DEM generation process cannot be corrected in the case of maximum height. For example, the highest point may be smoothed out due to the size of the ground pixel or the depth filter during the construction of DEM because of the movement of plant leaves, resulting in generally lower value than the true maximum height. Another significant error source is the algorithm to extract height from DEM, where the plot segmentation and baseline correction introduced errors into the height calculation. During the plot segmentation, only the central 75% of the plot canopy was used to avoid the overlap with nearby plots, potentially excluding the highest points and resulting in a lower calculated maximum height than the manual measurement. The resulted errors can be as large as -40.4 cm, but most of the errors were within -20 cm to 20 cm. The baseline correction can correct the unevenness of the ground (soil contours left by the tractor), which could be an issue for manual measurement since the base of the plant changed with the soil contour. This

can be revealed by the standard deviation of the difference between the ground plane and the DEM for the ground pixels which was in the range of 1 - 12 cm for plot DEMs from all data sets. Similar results were also found by Chu et al. that the bare soil areas have $\sim$5 cm uncertainty in DEM [154].

Because the maximum plot height can be affected by the above error sources, it may not accurately reflect the overall canopy for the plot. The DEM not only provides height information but also the spatial distribution of the height. In this case, the height histogram presents the distribution of the height, which can better describe the development of the canopy (Figure 4.10). For example, changes over time of the height from low to high indicate the growth of the canopy. The histogram can be divided into two parts using a threshold of 20 cm, where the left part and right part indicate the height distribution of the ground and canopy, respectively. The metrics for characterizing the histogram, such as mean, standard deviation, skewness, kurtosis, and percentile, can be used to characterize canopy height. These metrics can be directly obtained with existing statistical tools. For example, the 99th percentile of canopy height distribution within a plot were used to represent the height of wheat [111].

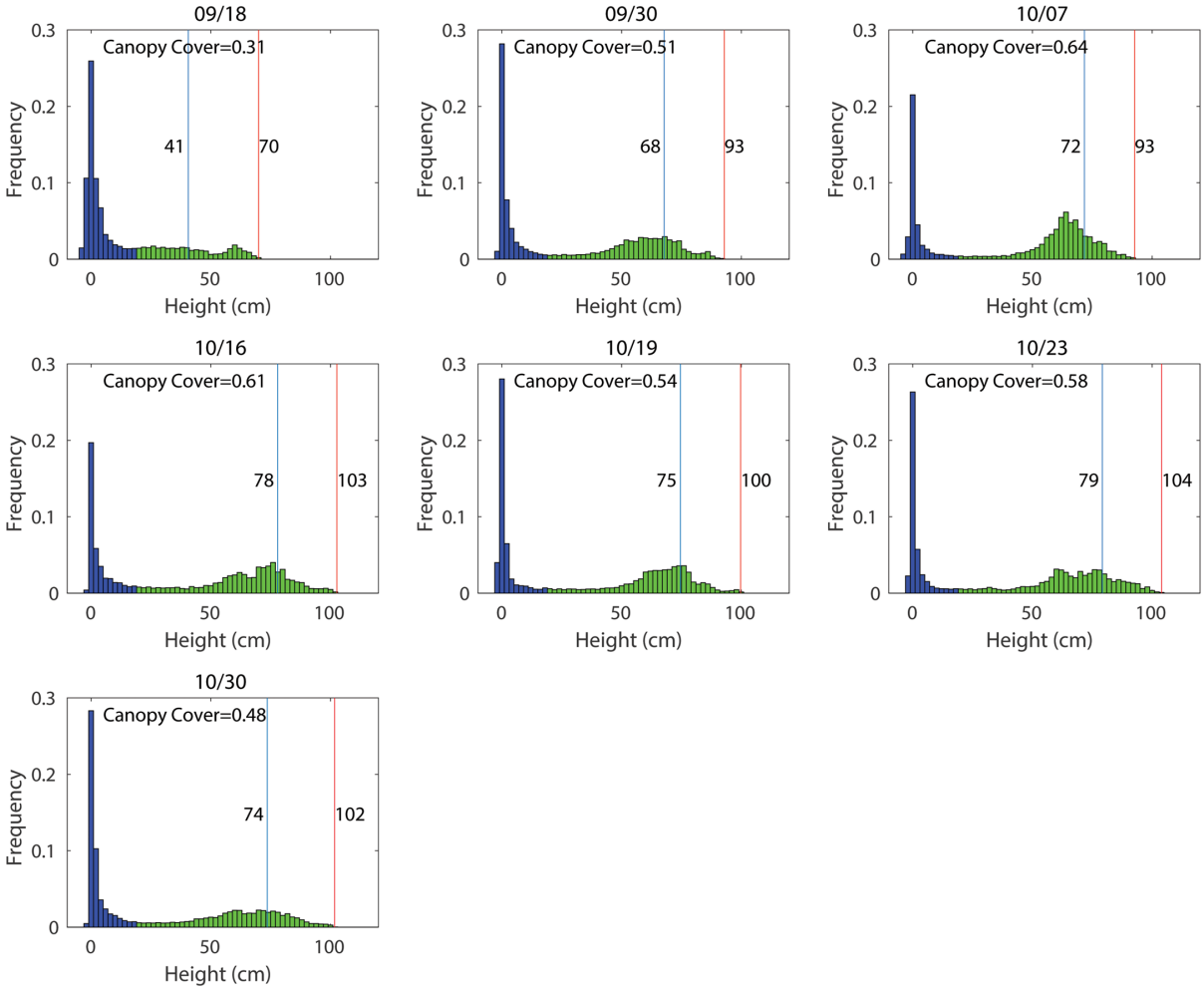

Figure 4.10: Histogram of the height for one plot over 42 days. Blue and green areas indicate the ground and canopy, respectively. Blue and red lines respectively indicate the 85th and 100th percentile (maximum) of canopy heights.

Flowering time is an important factor that effects the cotton yield [159]. Currently there is no high-throughput method to measure flowering time other than human visual evaluation in the field. The preliminary results have shown that the multispectral images have the potential to detect cotton flowers, which can be very helpful to continuously monitor the flower development over the season, although more research needs to be done to overcome its limitations. For example, the flowering count can be underestimated comparing to human

evaluation because flowers hidden inside the canopy may not be captured by the image. Using oblique imagery to provide different views of the canopy to increase the chance to capture the flowers inside canopy could be a solution to solve the underestimation.

The main contribution of this study is that we developed a methodology to measure multiple phenotypic traits using multispectral images on a UAS. These image-derived traits were validated against manual measurements, providing confidence of our method (Table A.1). Compared to other methods used to calculate crop/tree heights, our method has two key improvements. First, it does not require the bare soil model that was used in other papers [108, 154]. It is particular useful for perennial crops and trees whose bare soil model cannot be measured. Second, using the fitted ground surface as baseline can compensate the unevenness of the ground terrain in comparison to the average or minimum height of the surrounding soil that were used in the tree height estimation studies [110, 160, 158]. Another unique contribution of this study is that we demonstrated the feasibility of detecting cotton flowers for the first time, making it possible to quantitatively monitor cotton flower development over time. We acknowledge that our methodology has certain limitations and some aspects can be improved. For example, the georeferencing error of the ortho-mosaic and DEM limited the usage of the automatic plot segmentation based on the geolocation of the plots. The error would be greatly reduced if accurately surveyed ground control points were used. The resolution of the multispectral camera used in this study was relatively low and it limited the accuracy of the result. In addition, two independent software/services were used to generate ortho-mosaics and DEMs, which is not a cost effective and elegant solution.

In the future, we will incorporate the whole data processing pipeline into PhotoScan so the process can be streamlined and results can be directly generated by one software.

4.5 Conclusion

In this paper, we developed a methodology of using multispectral images acquired by an UAS to measure cotton plant height, canopy cover, and vegetation index, as well as to detect flowers, which is an important step towards bringing UAS and advanced sensor technologies to cotton breeding. The ground calibration targets were effective to substantially reduce systematic errors in the digital elevation model due to georeferencing errors caused by the low-accuracy GPS on the UAS and lack of accurately surveyed ground control points. Histograms of plot DEM presented more information about the distribution of plant canopy height than the maximum height, and can be used to derive statistics to characterize the plot height. Correlation between the canopy cover and NDVI was found, and gradually declined after the canopy fully closed. The difference in the spectral response among ground, leaves, and cotton flowers showed the potential of detecting cotton flowers using multispectral images, which can be used in the future to assess the timing of cotton flowering and its distribution over time. Although we only demonstrated the application of aerial multispectral images for cotton phenotyping in this study, we will integrate a color camera and thermal camera into the UAS in the future. The combination of color, multispectral, and thermal images could provide a wide range of information about crops, which can be used to measure more complicated traits for cotton and other crops alike.

Chapter 5

Aerial Images and Convolutional Neural Network for Cotton Bloom Detection[1]

[1]Xu, Rui, Changying Li, Andrew H. Paterson, Yu Jiang, Shangpeng Sun, and Jon S. Robertson. "Aerial images and convolutional neural network for cotton bloom detection." *Frontiers in plant science* 8 (2018): 2235. Reprinted here with permission of publisher

Abstract

Monitoring flower development can provide useful information for production management, estimating yield and selecting specific genotypes of crops. The main goal of this study was to develop a methodology to detect and count cotton flowers, or blooms, using color images acquired by an unmanned aerial system. The aerial images were collected from two test fields in four days. A convolutional neural network (CNN) was designed and trained to detect cotton blooms in raw images, and their 3D locations were calculated using the dense point cloud constructed from the aerial images with the structure from motion method. The quality of the dense point cloud was analyzed and plots with poor quality were excluded from data analysis. A constrained clustering algorithm was developed to register the same bloom detected from different images based on the 3D location of the bloom. The accuracy and incompleteness of the dense point cloud were analyzed because they affected the accuracy of the 3D location of the blooms and thus the accuracy of the bloom registration result. The constrained clustering algorithm was validated using simulated data, showing good efficiency and accuracy. The bloom count from the proposed method was comparable with the manual count with an error of -4 to 3 blooms for the field with a single plant per plot. However, more plots were underestimated in the field with multiple plants per plot due to hidden blooms that were not captured by the aerial images. The proposed methodology provides a high-throughput method to continuously monitor the flowering progress of cotton.

5.1 Introduction

Improving cotton yield is one of the main objectives in cotton breeding projects and cotton growth and production management. Yield can be defined as the product of the number of bolls produced per unit area and the mass of lint per boll (a common measure of boll size). Cotton yield is associated with many physiological variables and environmental factors. However, an increase in cotton yield is generally associated with an increase in the number of bolls regardless of genotype or environment [161, 162]. Flower and boll retention will affect the final number of bolls produced. Therefore, processes that affect flower and boll retention will have a significant impact on yield.

Flowering is important to cotton yield because pollinated flowers form cotton bolls. Within a given genotype, seasonal flower production per unit area is more closely related to yield than boll retention percentage [163, 164]. Since a cotton flower is white (cream-color for some upland germplasms and yellow for Pima) on the first day it opens and turns pink within 24 hours, it is unlikely to mistake an old bloom for a new bloom on separate days. Therefore, if bloom counts are obtained daily then the seasonal total counts can be calculated. The flowering time (the time of the first flower opening) and the peak bloom time can be determined accordingly, both of which are critical to production management. The timing and duration of the flowering stage also reflect the difference in growth habits of different genotypes, which can help breeders select specific genotypes, for example, short-season or long-season genotypes. Although manual counting is perhaps the simplest and most reliable

way to count flowers, it is tedious and inefficient and requires a massive amount of labor, which is not practical for large fields.

Imaging methods can improve the efficiency of manual counting. Many studies have been done on flower detection and classification using color images [165, 166, 167, 168, 169, 170]. Flowers usually have distinct color and texture from the background; therefore, traditional image processing methods such as color and texture analysis can be used to segment flower pixels [170]. The number of flowers can be calculated using morphological operations on the segmented flower pixels or correlated with flower pixel percentage [168, 170]. Flower features can be extracted from flower images and are used to recognize flower species [167]. Machine learning techniques can be used to classify different flower types, which could be useful to determine the age of cotton flowers based on differences in color and shape [169]. Deep learning methods such as the convolutional neural network (CNN) have been demonstrated to be effective in recognizing flower species [171]. CNN showed advantages over traditional machine learning methods because it does not require extraction of image features.

Although imaging methods have proved to be efficient in detecting flowers, the image collection throughput limits its usage in agriculture because agriculture usually deals with large fields. Therefore, improving the data collection throughput is important. In this case, the use of an unmanned aerial vehicle (UAV) is preferred over a ground vehicle or robot because a UAV can provide superior data collection speed and larger spatial coverage. UAVs also do not interact with the plants, so constant data collection will not cause soil compaction and plant damage, which can happen when using a ground vehicle. Although many researchers have used UAVs for agriculture studies in recent years, only a few used

UAVs to count flowers for crops. For example, aerial multispectral imaging has been used to calculate flower fraction in oilseed rape and monitor the peach flower (Fang et al., 2016; Horton et al., 2017). However, those studies only segmented flowers from the canopy rather than counting the flowers. To our knowledge, no study has been done to count cotton flowers using aerial images. The main reason for this is the low image resolution compared to images taken from ground platforms, making image processing challenging.

In this paper, we aimed to develop a methodology for counting cotton blooms using aerial color images. The overall objective of this paper was to develop a data processing pipeline to detect and count the number of newly opened cotton flowers using aerial images. Specific objectives were to: 1) build and train a CNN to classify flowers, 2) construct dense point clouds from raw images and evaluate the quality, 3) develop an algorithm to register flowers, and analyze its accuracy and efficiency using simulated data, and 4) evaluate the performance of the data processing pipeline compared with manual counting. To avoid ambiguity, we used the term bloom to refer to a newly opened flower to distinguish it from its other growth stages.

5.2 Materials and Methods

5.2.1 Test fields

Two test fields were used, both of which were located in the Plant Science Farm in Watkinsville, GA (33°43′37.80″N, 83°17′57.52″E) (Figure 5.1). Field 1 had 132 plots with a single cotton plant in each plot, arranged in 12 rows and 11 columns with a 1.5 m (5 ft) distance in both

row and column directions. The cotton in field 1 was planted on May 25, 2016. Field 2 had 128 plots of cotton, arranged in 16 rows and 8 columns (Figure 5.1). Each plot was 3 m (10 ft) long and 1.5 m (5 ft) wide, with a 1.8 m (6 ft) alley between each plot. Fifteen cotton plants were planted in each plot with equal spacing of 0.15 m (6 inch) on June 13, 2016. The plant density varied because of the different number of plants germinated, resulting in some empty plots in field 1 and field 2. The number of germinated plants in each plot for field 2 was recorded two weeks after planting (June 8, 2016) to be used to calculate the average number of blooms for each plant.

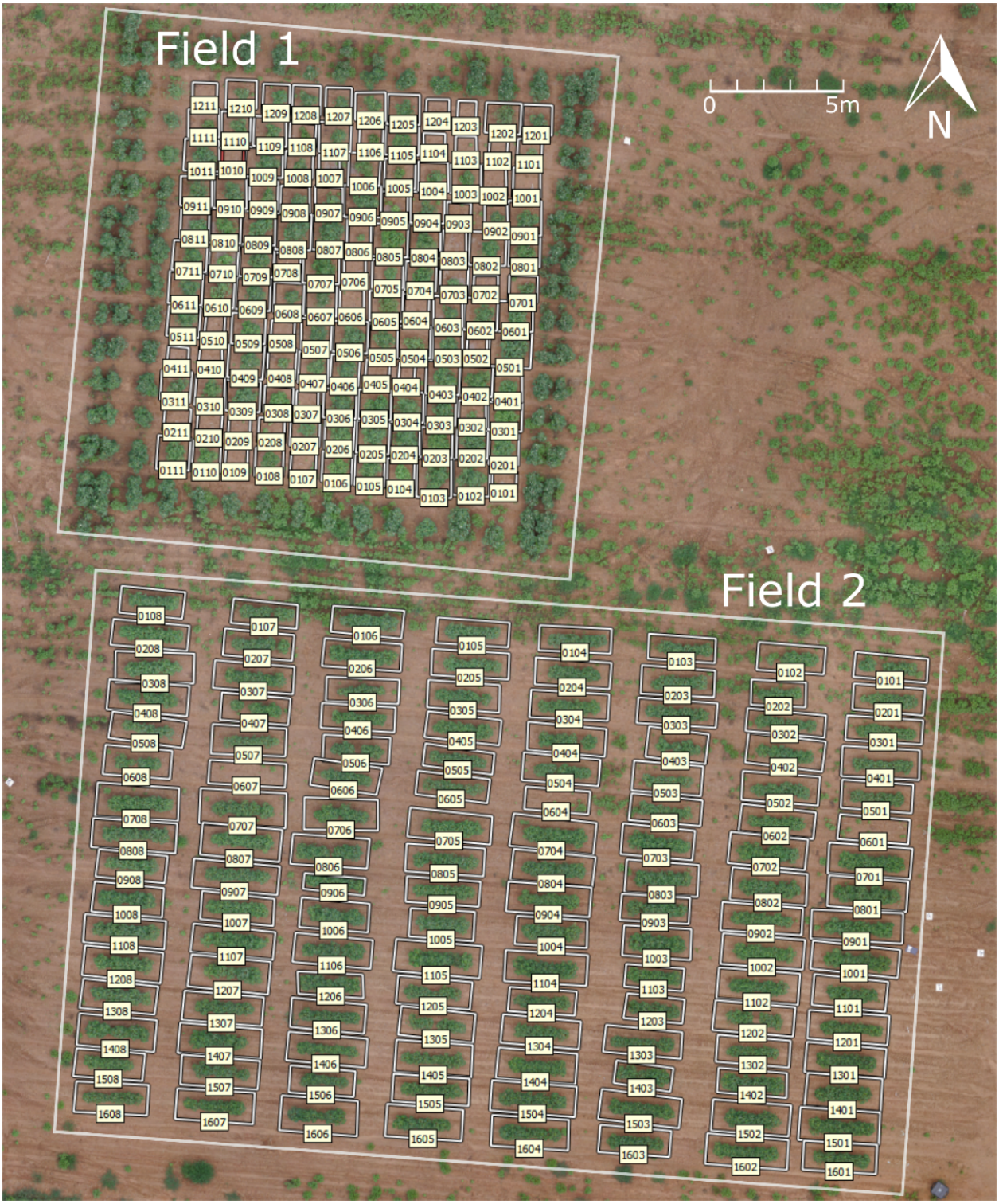

Figure 5.1: Plot layout of the two test fields.

5.2.2 Data collection

Aerial color images of the test fields were collected using an octocopter (s1000+, DJI, China) with a color camera (Lumix DMC-G6, Panasonic, Japan) on August 12, August 19, August 26, and September 9, 2016 (Table 5.1). The color camera was directly mounted on the bottom of the drone with the lens facing downward. An inertial measurement unit (IMU) with GPS was mounted on the drone to record the location and orientation of the camera. Raspberry Pi 2 was used to trigger the camera and record the IMU/GPS measurement. During the flight, the camera took images at a frequency of 1 Hz. The exposure time and aperture were manually set for the color camera according to the light conditions of the field, and auto-settings were used for other parameters. The drone was flown at a height of 15 m above ground level (AGL) to reduce the ground sample distance (GSD). One flight was used to collect data from both of the test fields.

Table 5.1: Data collection summary

Date	**Time (p.m.)**	**Flight height (m)**	**Flight speed (m/s)**	**Focal length (mm)**	**Ground pixel size (mm)**
8/12/2016	12:09	15	2.5	18	3.17
8/19/2016	1:43	15	1.5	20	2.23
8/26/2016	1:35	15	3	20	2.69
9/9/2016	1:12	15	2.5	22	2.36

The number of blooms in each plot was manually counted as a reference for the image method. To count the blooms in each plot, only white flowers were counted in each plot on the same day that aerial images were collected. Although manual counting is reliable overall, it is possible for some blooms to be counted multiple times or not counted. The judgement

of whether a flower is white is subjective and varies from person to person, which could cause human counting errors. For field 1, blooms in all of the plots were counted on all four days. For field 2, blooms were counted in all plots on August 12, and in 32 random plots on the other three days.

5.2.3 Data processing pipeline

The overall data processing pipeline for bloom counting can be divided into four key steps: 1) dense point cloud generation, 2) plot images extraction, 3) bloom detection, and 4) bloom registration (Figure 5.2). The final output of the pipeline is the bloom count for each plot. Other information such as bloom position can be obtained too.

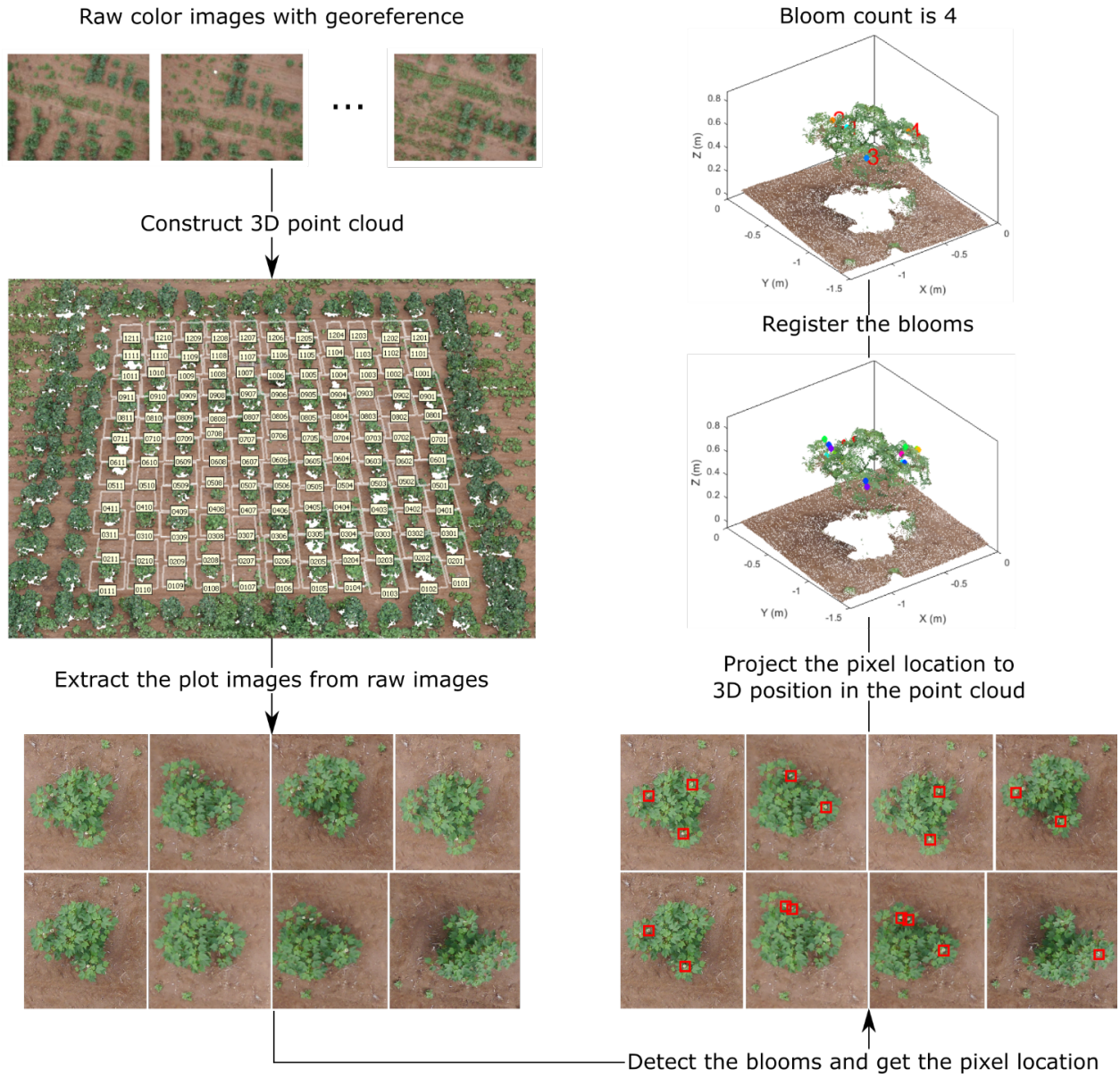

Figure 5.2: Overall flowchart of the bloom detection algorithm, demonstrating the output of each step for field 1.

Dense point cloud generation

A dense point cloud of each test field was generated in PhotoScan (PhotoScan Professional 1.3.1, Agisoft, Russia) using the raw color images and the IMU/GPS measurements. The high accuracy and reference preselection settings were used for photo alignment, the high-

quality setting for dense point cloud build, and default for other settings. After photo alignment, PhotoScan constructed a tie point cloud—often called a sparse point cloud in other software—from the feature points detected in the images. Before constructing the dense point cloud, the tie point cloud was used to calculate the accuracy of the dense point cloud. After building the dense point cloud, it was found that field 1 was not fully covered by the images collected on August 19 and those uncovered plots were excluded from further data analysis. Additionally, some plots had an incomplete point cloud for the canopy due to low side-overlap on August 19, August 26, and September 9, when the focal length was enlarged to reduce the ground pixel size. Point cloud coverage, which is the percentage of the plot constructed with valid point cloud, was used to evaluate the completeness of the point cloud. The plots with point cloud coverage less than 80% were excluded from further analysis. The method to calculate point cloud coverage is explained in section 2.5.

Pot images extraction

The main purpose of the plot images extraction step is to export plot images from the raw images such that one plot image contains only one plot or part of the plot. Since one plot can be imaged several times, one plot has multiple plot images from different raw images. This is helpful for detecting blooms inside the canopy because different raw images provide views of each plot from different angles, which greatly improves the chance of one bloom being imaged compared to the ortho-image, which only shows the top view of the canopy. To extract the plot images, plot boundaries were first manually drawn for each plot and stored as quadrangle shapes in PhotoScan. Then the four vertices of each shape were projected

to each raw image to get the pixel location of the vertices in each image. One image was considered to cover part of the plot if two or three vertices were in the image (pixel location is within the boundary of the image), and the whole plot if four vertices were in the image. The pixel location of the four vertices was recorded and plot images were extracted from the raw image accordingly. The step was processed in PhotoScan using built-in Python functions. The image file name, plot number, and pixel locations of the four vertices were saved in a text file and imported into MATLAB (MATLAB 2017a, MathWorks, USA) to extract plot images from raw images.

Bloom detection

The bloom detection process contains two steps. The first step is to screen out the locations that most likely contain cotton blooms based on the fact that cotton blooms usually have the highest pixel intensity because of their white color. The plot image was first transferred into CIELAB color space and then the screening was performed using the L channel. A threshold of 0.75 on the normalized L channel (normalized by the maximum value in L channel) was used to extract bright objects. The number of pixels for each object was counted and noisy small objects less than 15 pixels were removed. Objects separated by a distance of less than the diameter of a flower—which is about 20 pixels for our dataset—were combined into one object because some flowers were split into two objects by the leaves. Images with a size of 36 by 36 pixels around the center of the remaining objects were extracted from the plot image. Those images were treated as potential bloom images.

The second step is to classify the potential bloom images into bloom and non-bloom class using the pre-trained CNN (detailed information on the CNN and training process is described in section 2.4). After classification, the images classified as bloom class were kept and the pixel location of those images in the raw color images was recorded.

Bloom registration

Since the same bloom can be detected in several images, it is necessary to register the bloom before counting the blooms to prevent counting the same bloom more than once. Registration methods based on image features were not useful because very few features could be detected from the bloom images due to the resolution limit. Therefore, we first projected the pixel location of each detected bloom into the 3D location in the dense point cloud, and then clustered the blooms based on the locations because the same bloom should have the same 3D location in the dense point cloud. The projection was done in PhotoScan. Because each pixel in the image has a corresponding image ray, any point in the image ray will appear as the same pixel in the image, and the 3D location of that pixel in the point cloud is the intersection between the image ray and the point cloud. If there is no intersection between the image ray and the point cloud, PhotoScan will return the closest point to the image ray.

Due to the accuracy of the point cloud and pixel location, the 3D location of the bloom may deviate from the true location, which increases the chance of incorrect bloom registration solely based on the location. However, since multiple blooms detected in the same image are different blooms and cannot be in the same cluster, the bloom registration based on the 3D position can be generalized as a constrained clustering problem as follows:

Given the set of data points and the set of their corresponding classes, form the data points into clusters so that no data points in the same class will be in the same cluster and the distance between any two clusters are not smaller than a threshold.

In the case of bloom registration, the data point is the bloom position and the class is the image number to indicate in which image the bloom was detected. Although existing constrained clustering algorithms can solve the problem, most of the algorithms are for general clustering problems and are not efficient for this specific problem. Therefore, inspired by hierarchical clustering, a deterministic clustering algorithm was designed specifically for the bloom registration problem. The algorithm initializes with each data point as one cluster. The algorithm involves cluster selection and merging. First, for each cluster i, a set of clusters, $\mathcal{S}$, was selected from all of the clusters such that no data points in $\mathcal{S}$ had the same class as any data point assigned to cluster i. Second, the distance between cluster i and every cluster in $\mathcal{S}$ was calculated and cluster i was merged to the closest cluster in $\mathcal{S}$ if their distance was smaller than the threshold λ. The algorithm repeats the selection and merging until no merging happens. The algorithm has a similar effect as the regular hierarchical clustering algorithms when the distance between two clusters is measured using equation 1.

$$dist\left(\mu_i, \mu_j\right) = \begin{cases} dist'\left(\mu_i, \mu_j\right), & \text{if } E_i \cap E_j = \varnothing \\ +\infty, & otherwise \end{cases} \tag{5.1}$$

where μ_i and μ_j are the center of the cluster i and j, $dist'(\mu_i, \mu_j)$ is the distance metric used in the hierarchical clustering algorithms, and E_i and E_j are the set of classes of the data points assigned to cluster i and j. Compared to the regular hierarchical clustering algorithms, the efficiency of our algorithm is improved because the distance calculation is only performed on a subset of the clusters.

Euclidean distance was used to measure the cluster dissimilarity. However, because of the large error of the 3D position projection on the z-axis, the Euclidean distance was modified by adding a weight term on the z-axis to adjust the influence of z-axis on the distance. The weighted Euclidean distance between $a = (a_x, a_y, a_z)$ and $b = (b_x, b_y, b_z)$ was calculated using equation 5.2. The final program used 0.5 for w.

$$dist(a, b) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2 + w^2 (a_z - b_z)^2} \tag{5.2}$$

Bloom registration algorithm

Input: Set of data points $\mathcal{D} = \{x_i\}_i^n$, class $E = e_i{}_i^n$, and distance threshold λ.
Output: The assignment variables for the data points, $z = [z_1, z_2, \ldots z_n]$.
Initialization: each point is assigned as one cluster, $z_1 = 1, z_2 = 2, \cdots, z_n = n$
do
  for $i = 1 : n$ **do**
    if *cluster i exists* **then**
      Find cluster i and its center μ_i
      $\mathcal{S} \leftarrow \varnothing$ // initialize a set to store the center of selected clusters
      Given the current assignment of points, find the set of classes of the data points assigned to cluster i, $E_i = \{e_k \in E | z_k = i\}$
      for $j = 1 : n$ **do**
        if *cluster j exists* **then**
          Find cluster j and its center μ_j
          Given the current assignment of points, find the classes of the data points assigned to cluster j, $E_j = \{e_k \in E | z_k = j\}$
          if $E_i \cap E_j = \varnothing$ **then**
            //Add current cluster's center to $\mathcal{S}$.
            $\mathcal{S} \leftarrow \{\mathcal{S} \cup \mu_j\}$
          end
        end
      end
      $[d_{min}, i_{min}] \leftarrow min\left\{dist\left(\mu_i, \mathbf{s_1}\right), \cdots, dist\left(\mu_i, \mathbf{s}_{|\mathcal{S}|}\right)\right\}$
      if $d_{min} < \lambda$ **then**
        //Merge cluster i with i_{min} and update the assignment variables
        Assign i_{min} to the assignment variables for the data points in cluster i
      end
    end
  end
  for $k = 1 : n$ **do**
    Given the current assignment of points, find the set of points assigned to cluster k, $\mathcal{D}_k = \{x_i \in \mathcal{D} | z_i = k\}$
    if $|\mathcal{D}_k| > 0$ **then**
      $\mu_k \leftarrow \frac{\sum_{\mathbf{x}_i \in \mathcal{D}_k} \mathbf{x}_i}{|\mathcal{D}_k|}$
    else
      Remove the cluster and apply the changes to the related variables
    end
  end
while *clusters merged*

5.2.4 Convolutional neural network

The convolutional neural network was used to classify potential bloom images. The network had seven layers, one input layer, two convolutional layers, two max pooling layers, and two fully connected layers (Figure 5.3). Since the classification was performed on the potential bloom images, the potential bloom images were first extracted from individual data sets and manually labeled into bloom and non-bloom classes. Then the labeled potential bloom images were used to construct the image database. In total, 14,000 images for the bloom class and 60,000 images for the non-bloom class were extracted. Two thirds of the bloom images (roughly 9,000 images) and the same number of non-bloom images were randomly selected as training samples. The remaining images were used as testing samples. The CNN was trained for 30 epochs. The learning rate was set to 0.01 at the first epoch and decreased by a factor of 10 every 10 epochs. The mini batch size was set to 256. The regularization factor was set to 0.01 to prevent overfitting.

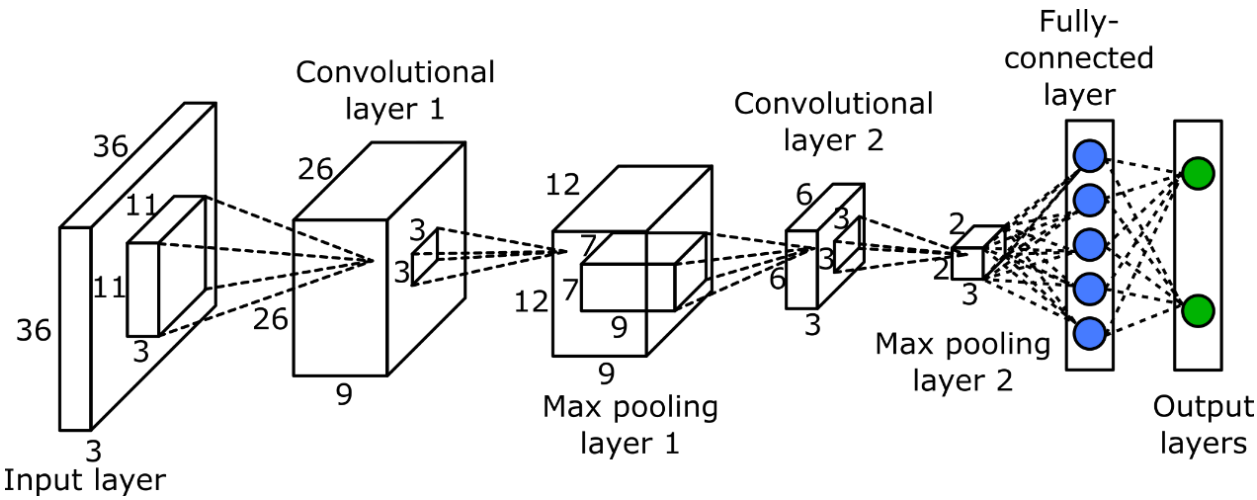

Figure 5.3: Structure of the convolutional neural network.

5.2.5 Quality of the dense point cloud

The quality of the dense point cloud of the canopy greatly affects the accuracy of the 3D position of each bloom, and thus the clustering results, and ultimately the bloom numbers. The quality of the dense point cloud can be evaluated from two aspects—the accuracy and the completeness of the dense point cloud.

The position accuracy of each point in the point cloud is difficult to calculate because the geometry of the canopy is unknown. Instead, the projection and reprojection error were calculated for the tie points to estimate the overall accuracy of the dense point cloud. The projection error was calculated as the distance between the projection of the feature points to the point cloud and their corresponding tie points. The reprojection error was calculated as the pixel distance between the projection of the tie points on the images and their original corresponding feature points. The projection error was analyzed on each single axis (easting, northing, and elevation) and the summation of three axes.

The completeness of the dense point cloud quantifies how completely the point cloud represents the real object or scene, and can be measured by the density of the point cloud. The completeness of the point cloud on the easting-northing plane was calculated since the camera only captures the top view. First, the point cloud was rasterized into a 2D elevation map using grid steps of 1.3 cm without interpolating the empty cells. The elevation map was divided into each plot. For each plot, the ground surface was calculated using the Maximum Likelihood Estimation SAmple Consensus (MLESAC) [136]. The plot was divided into ground and canopy using the elevation map based on the distance to the ground surface

with a threshold of 0.1 m. The point cloud coverage (PCC) for the canopy is defined in equation 5.3.

$$PCC = \frac{Area\ of\ the\ canopy}{Area\ of\ the\ canopy + Area\ of\ the\ empty\ cells} \tag{5.3}$$

The area of the empty cells was included in the denominator because the empty cells were usually part of the canopy. The PCC estimates the completeness with which the canopy is represented by the dense point cloud, which can greatly affect the bloom registration result.

5.2.6 Evaluation of the bloom registration algorithm

To evaluate the efficiency and accuracy of the bloom registration algorithm, artificial data was generated in three steps. First, a data point with three dimensions was generated by randomly drawing integers from 0 to 9 for each axis. This step was repeated until 125 different data points were generated. The same class number was assigned to the 125 data points to simulate that they were from the same image. Second, the first step was repeated 10 times, generating in total 1,250 data points. A new class number was assigned to the 125 data points at each repeat to simulate that the data points from each repeat were from different images. The data points with the same coordinates were set in the same cluster, which is the true clustering result. Third, a random error that followed $\mathcal{N}\left(0, \frac{\sqrt{3}}{3}\sigma\right)$ was added to each axis of each data point, thus the position error (vector summation of the errors of the three axis) followed $\mathcal{N}(0, \sigma)$. Different σ was used to simulate the noise level of the 3D position of

the bloom. The 1,250 data points were clustered using the bloom registration algorithm and compared with the true clustering result.

Since the clustering result will depend on the noise level of the data and threshold for the clustering algorithms, 10 different σ values from 0.1 to 0.5 with an interval of 0.05, and 5 threshold values were tested. The threshold values were σ, 1.5σ, 2σ, 2.5σ and 3σ and they covered from 68% to 99.7% of the position error. To acquire statistics of the result, the third step of generating artificial data was repeated 10 times to generate 10 sets of artificial data for each noise level and threshold. The mean and standard deviation of the clustering error and runtime on the 10 data sets were analyzed. The misclustering rate was defined using equation 5.4.

$$misclustering\ rate = \frac{number\ of\ misclustered\ points}{total\ number\ of\ points} \tag{5.4}$$

Since it is difficult to relate the clustering result with the ground truth, the misclustered points were considered as the points that were in the same cluster from the clustering result but not in the same cluster from the ground truth. The number of misclustered points was calculated with two approaches. The first approach—referred to as misclustering rate by reference—used the true clustering result as the ground truth, and the second approach—referred to as misclustering rate by result—used the clustering results as the ground truth. The first approach was good at evaluating the clustering error when the clustering algorithm over clustered the data points, while the second approach was suitable for under-clustering errors.

5.3 Result

5.3.1 Quality of the dense point cloud

The point cloud accuracy was analyzed using the dataset collected on 8/12. More than 99% of the reprojection errors of the tie points were less than 1.7 pixels and only a few were larger than 2 pixels (Figure 5.4). The mean reprojection error was 0.5 pixels. The tie points had overall larger projection error on the elevation than the easting and northing (Figure 5.5). The mean projection error for the easting and northing was 0 m, whereas the mean projection error for the elevation was 0.014 m, which is reasonable since the depth generated from multi-view stereo is usually the least accurate. The elevation also had larger variation than the easting and northing. The mean projection error for the three axes combined was 0.022 m, and 99% of the errors were between 0 m and 0.127 m (Figure 5.5d). The large projection error on the elevation validates the rationale to assign a weight to the elevation when calculating the cluster distance in bloom registration.

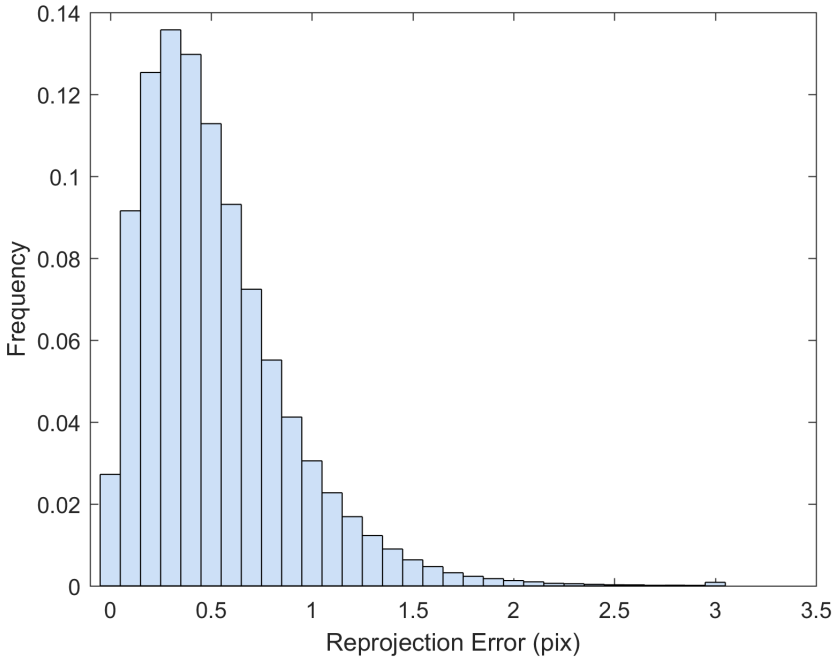

Figure 5.4: Histogram of the reprojection error for the tie points generated from 8/12/2016 dataset.

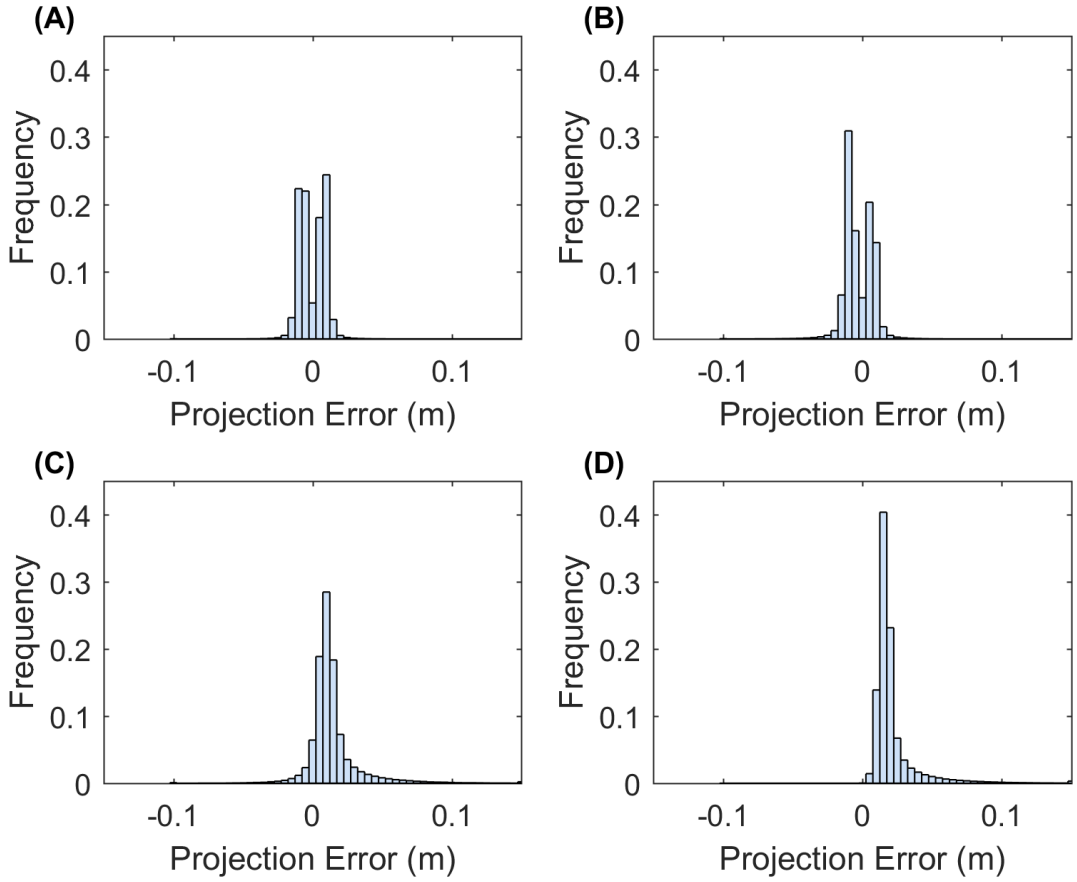

Figure 5.5: Histogram of the projection error for the tie points generated from the 8/12/2016 dataset. A) Error histogram for easting. B) Error histogram for northing. C) Error histogram for elevation. D) Error histogram for the summation of the three axes.

In the event of an incomplete dense point cloud for the canopy, some blooms may not be able to get 3D point positions and thus cannot be counted. If the blooms equally distribute over the canopy, the point cloud coverage can be considered as the probability of a bloom having a valid point in the dense point cloud. Assuming a canopy has n blooms and point cloud coverage of p, then k ($k = 1, 2, \ldots, n$) blooms not having valid points (which cannot be counted using the imaging method) follow a binomial distribution $B\,(n,\ 1-p)$. The mean

count error is $n\,(1-p)$ and the relative error is $(1-p)$. Therefore, the point cloud coverage should be close to 1 to make the counting error small.

Figure 5.6 showed that the 8/12 dataset had good point cloud coverage (> 0.9) on most of the plots for both test fields. However, the other three datasets showed low point cloud coverage on both fields, except that the 8/19 dataset had some plots with good point cloud coverage. The low point cloud coverage was mainly due to the insufficient image side-overlap ($<60\%$); the PhotoScan was unable to construct valid 3D points for areas that were covered by less than three images. Therefore, increasing the image overlap can improve the point cloud coverage. The depth filter inside the PhotoScan removed noisy points due to the movement of the plants and image noise, which was another reason for the low coverage. To achieve low image counting error, the plots with point cloud coverage lower than 0.8 were excluded from data analysis, which removed 16 plots in field 1 for the 8/12 dataset, 18 plots in field 1 and 103 plots in field 2 for the 8/19 dataset, all the plots in field 1 and 112 plots in field 2 for the 8/26 dataset, and 127 plots in field 1 and 97 plots in field 2 for the 9/9 dataset.

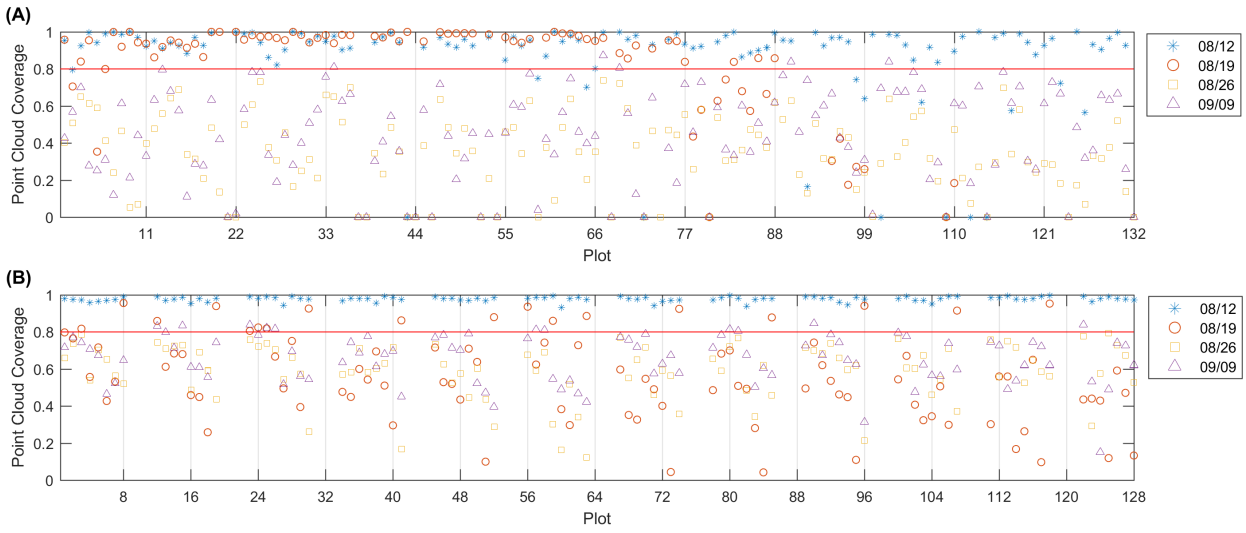

Figure 5.6: Point cloud coverage for field 1 (A) and field 2 (B). The red line indicates the 0.8 threshold.

5.3.2 Training result of the convolutional neural network (CNN)

With more than 28,000 training images and mini batch size of 256, the CNN was trained about 114 iterations every epoch. Because of the large training sample size and relatively simple structure, the training process converged quickly and the training accuracy reached 0.94 in the first epoch. As shown in Figure 5.7, the training accuracy for bloom and non-bloom class increased in the first few epochs. The training accuracy for the non-bloom class reached a maximum on the 8th epoch, and the training accuracy for the bloom class reached a maximum on the 12th epoch. The accuracy for both classes decreased after reaching the maximum value, fluctuated a bit, but then reached a stable accuracy with small variation after 20 epochs. The training loss decreased quickly over the first few epochs, which indicates the quick convergence of the training. After 10 epochs, where the learning rate changed from 0.01 to 0.001, the training loss kept decreasing but the change rate was small. After 20 epochs,

the training loss reached the minimum with a certain level of oscillation, which indicates that the training process should stop at the 20th epoch because no further improvement of the CNN can be gained from the last 10 epochs. Therefore, the training result at the 20th epoch was used to classify the potential bloom images. The testing accuracy was smaller than the training accuracy for both classes after 7 epochs but the difference between them was less than 0.01, which showed the CNN worked well on both the training and testing sets.

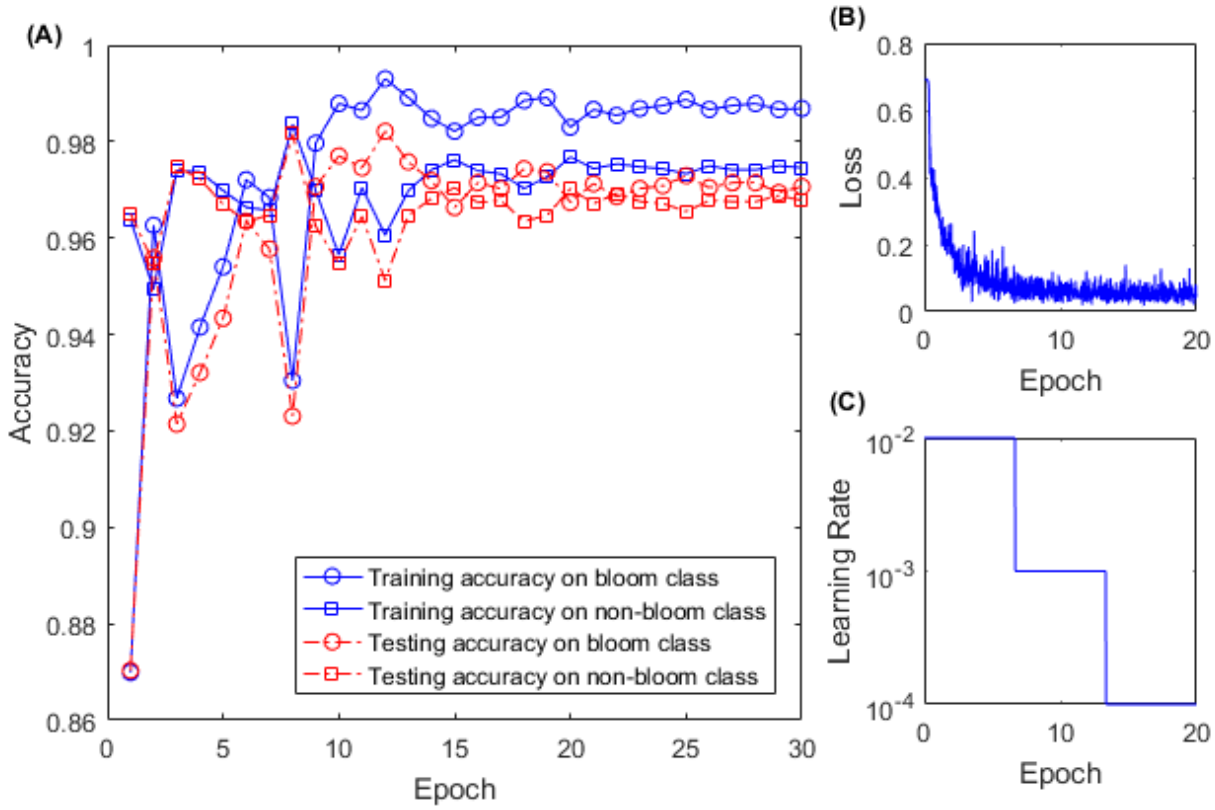

Figure 5.7: CNN training setting and result. A) Training and testing accuracy of the CNN on two classes over training epoch. B) Training loss over epoch. C) Learning rate over epoch.

The classification result for each dataset using the trained CNN varied due to the different lighting conditions and flowering stages (Table 5.2). The classification result showed high precision (>0.9) for both classes across all datasets. However, the recall (true positive rate) for

the bloom class was low for the 8/19 and 9/9 datasets. The CNN can cause overestimation or underestimation when comparing the number of predicted blooms and the number of actual blooms. For example, in the 8/12 dataset, the number of predicted bloom images was 789, but the actual number of bloom images was 719, thus the classification result overestimated the number of blooms by 10%. Similarly, the classification result overestimated the number of blooms by 23% and 29% in the 8/19 and 9/9 datasets, and underestimated the number of blooms by 2% in the 8/26 dataset. Objects such as cotton bolls, specular highlights on leaves, and pink flowers were misclassified as blooms because their shape and color appeared like a bloom in the aerial image due to the limited resolution (Figure 5.8). The misclassified blooms caused by objects (e.g. label sticks) on the ground could be removed based on the height from the ground, but the misclassified blooms caused by the plants (e.g. leaves, pink flowers, and cotton bolls) were difficult to eliminate using the height. Small blooms or partly hidden blooms can easily be misclassified as non-blooms because of their size. Blooms in the shade can also be misclassified as non-blooms because their intensity is reduced. Including those misclassified images into the training set may further improve the CNN performance.

Table 5.2: Classification result of the potential bloom images extracted from individual dataset for field 1.

		8/12/2016				**8/19/2016**			
Output class		**Bloom**	**Non-bloom**	**Precision**	**Recall**	**Bloom**	**Non-bloom**	**Precision**	**Recall**
Actual class	Bloom	709	10	0.90	0.90	813	37	0.96	0.78
	Non-bloom	80	4876	0.98	0.997	234	23208	0.99	0.998
		8/26/2016				**9/9/2016**			
Output class		**Bloom**	**Non-bloom**	**Precision**	**Recall**	**Bloom**	**Non-bloom**	**Precision**	**Recall**
Actual class	Bloom	731	82	0.90	0.91	558	28	0.95	0.74
	Non-bloom	68	8543	0.99	0.99	191	3946	0.95	0.99

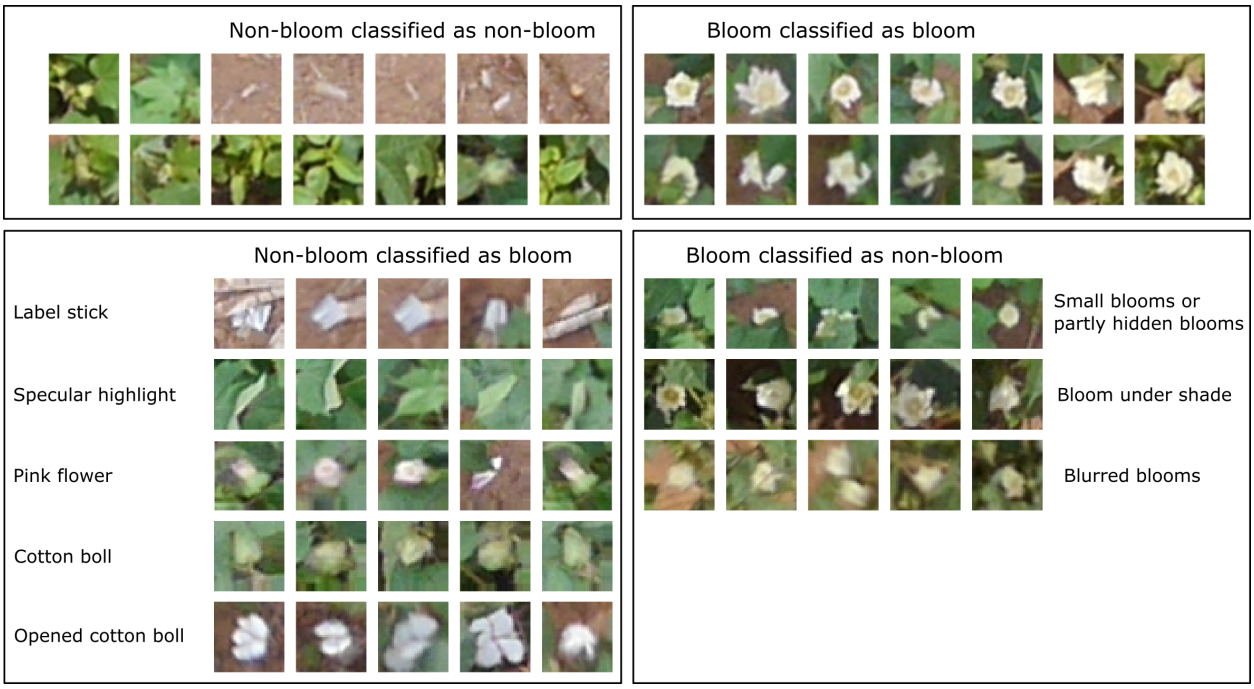

Figure 5.8: Example images of the classification result.

5.3.3 Efficiency and accuracy of the bloom registration algorithm

To cluster the 1,250 data points, the bloom registration algorithm used 13.3 to 48.4 seconds, which is 0.0106 to 0.0387 seconds per data point (Figure 5.9a). Using a larger threshold and larger noise level reduced the runtime because they resulted in a smaller number of clusters and thus the number of distance calculations. The bloom registration algorithm produced a near-zero misclustering rate by result at smaller noise level no matter what threshold was used because the distance between two reference clusters was still larger than the spread of the data points due to the noise (Figure 5.9b). As the noise level increased, the misclustering rate by result increased and larger threshold values had larger misclustering rates. This was mainly because of over-clustering when the noise of the points became large enough that certain portions of two clusters overlapped with each other and the algorithm could cluster

them into one cluster. The misclustering rate by reference did not change significantly as the noise level increased (Figure 5.9c). Smaller thresholds had larger misclustering rates by reference.

At low noise level, the under-clustering takes the main effect because the bloom registration algorithm can split one cluster into smaller clusters at low noise level, generating a larger number of clusters compared to the real cluster number (Figure 5.9d). As the noise increased, a smaller number of clusters than the real number of clusters was generated, resulting in over-clustering. The bloom registration algorithm had a smaller cluster number with larger threshold at the same noise level, indicating that it is more prone to under-clustering but less prone to over-clustering.

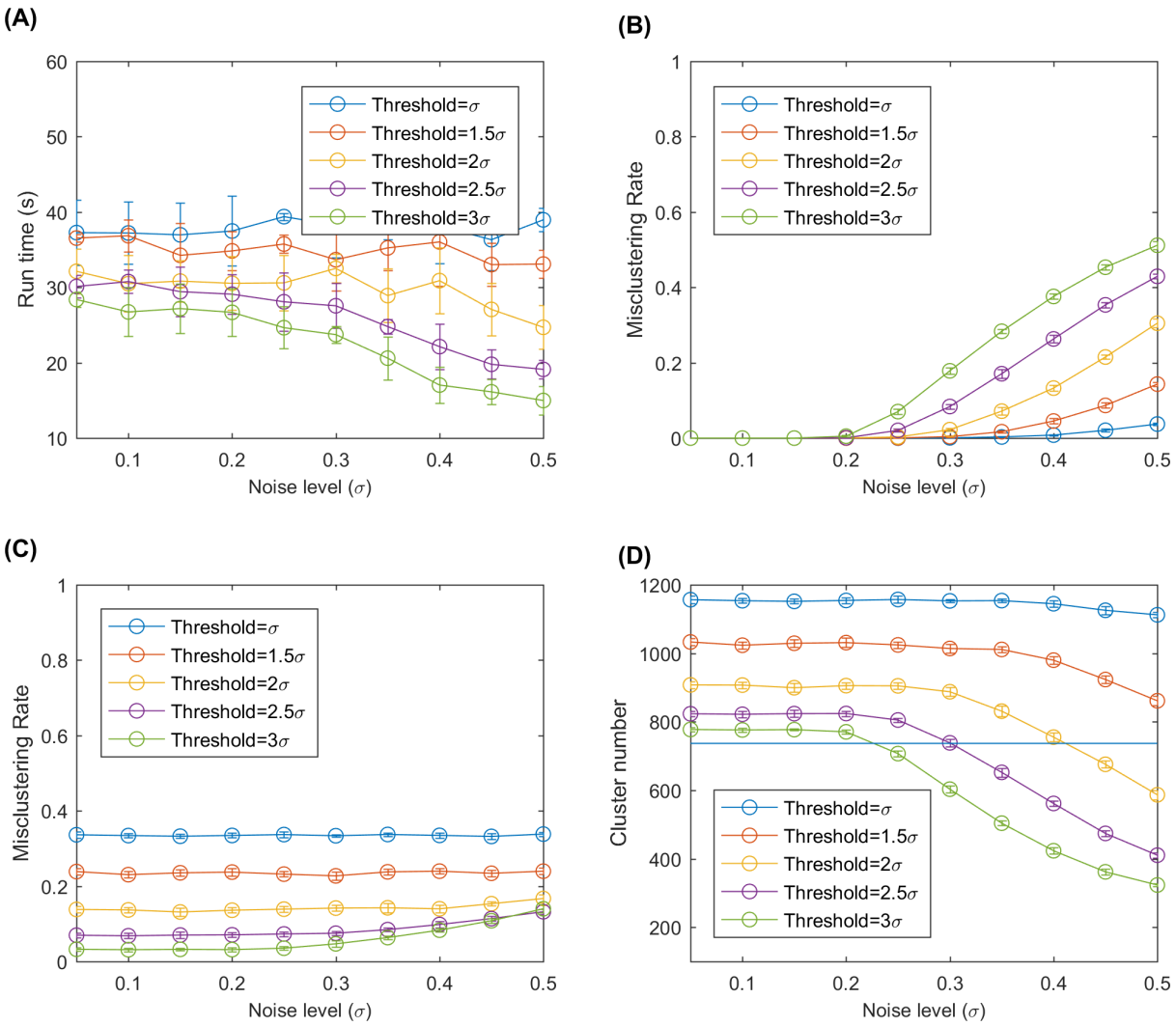

Figure 5.9: Test results of the bloom registration algorithm. A) Run time of the bloom registration algorithm on the simulation data. B) Misclustering rate by result. C) Misclustering rate by reference. D) Number of clusters using modified hierarchy clustering. The horizontal line is the true number of clusters.

5.3.4 Bloom count result

This section shows the bloom count results after removing the plots with point cloud coverage less than 0.8. The image count and manual count had the same trend for both fields (Figure 5.10). The error of the image count was between -4 and 3 for field 1, and the between -10

and 5 for field 2, showing that field 2 had more underestimated plots than field 1 (Figure 5.11). This may be due to the single plant per plot layout in field 1. The single plant per plot layout allows the plant to be seen from all directions without being blocked by other plants in the plot. Therefore, blooms in field 1 were more likely to be captured in the aerial images.

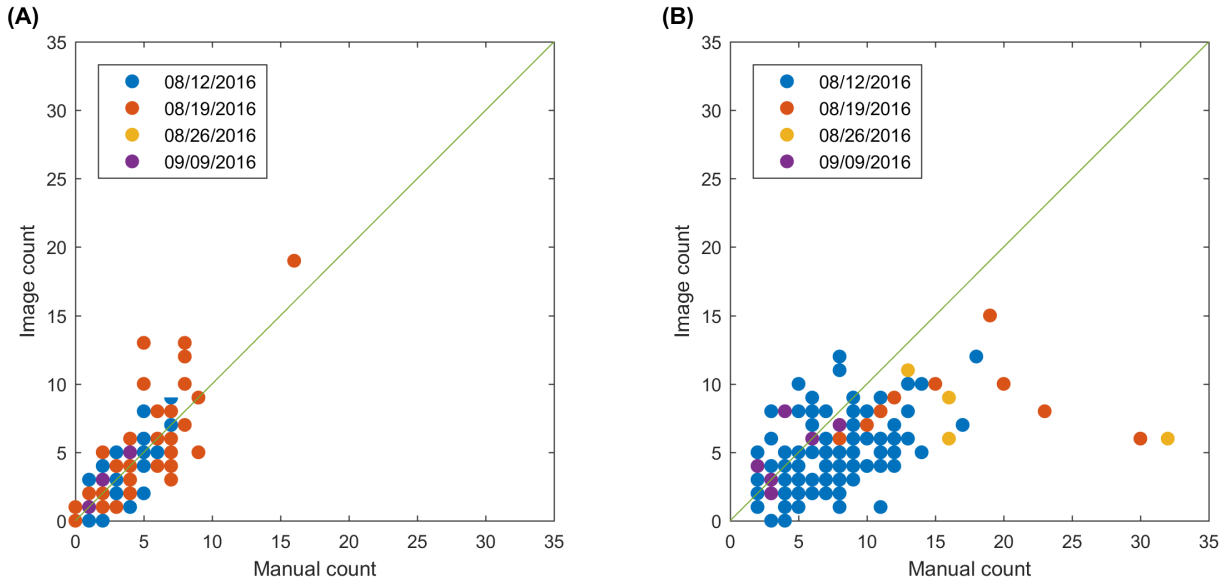

Figure 5.10: Comparison between image count and manual count for field 1 (A) and field 2 (B) after removing plots with point cloud coverage less than 0.8.

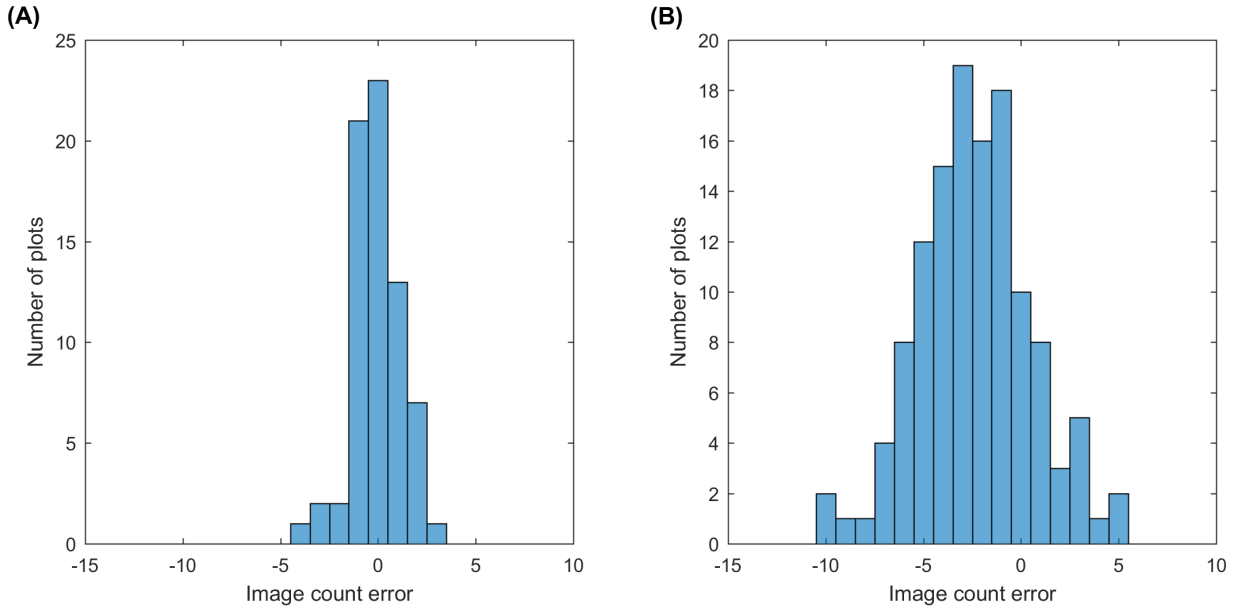

Figure 5.11: Histogram of the image count error for field 1 (A) and field 2 (B).

The overestimation was caused for two major reasons. One reason is the classification error of the CNN. For example, some leaves with specular highlights were classified as blooms (Figure 5.12a). The other reason is the quality of the point cloud, which causes the error of the bloom's 3D position and thus make the bloom registration incorrect (the same bloom from different images was registered as different blooms) (Figure 5.12b). The underestimation was caused by hidden blooms that were not shown in the aerial images, or blooms that were shown in the images but were not classified correctly by the CNN.

With enough datasets, it is possible to monitor the development of the flowers over time, which is one of the advantages of the proposed method. Figure 5.13 demonstrates the trend of the bloom development. It shows that the number of blooms was low at the early flowering stage, reached the peak in the middle flowering stage, and then decreased at the late flowering stage.

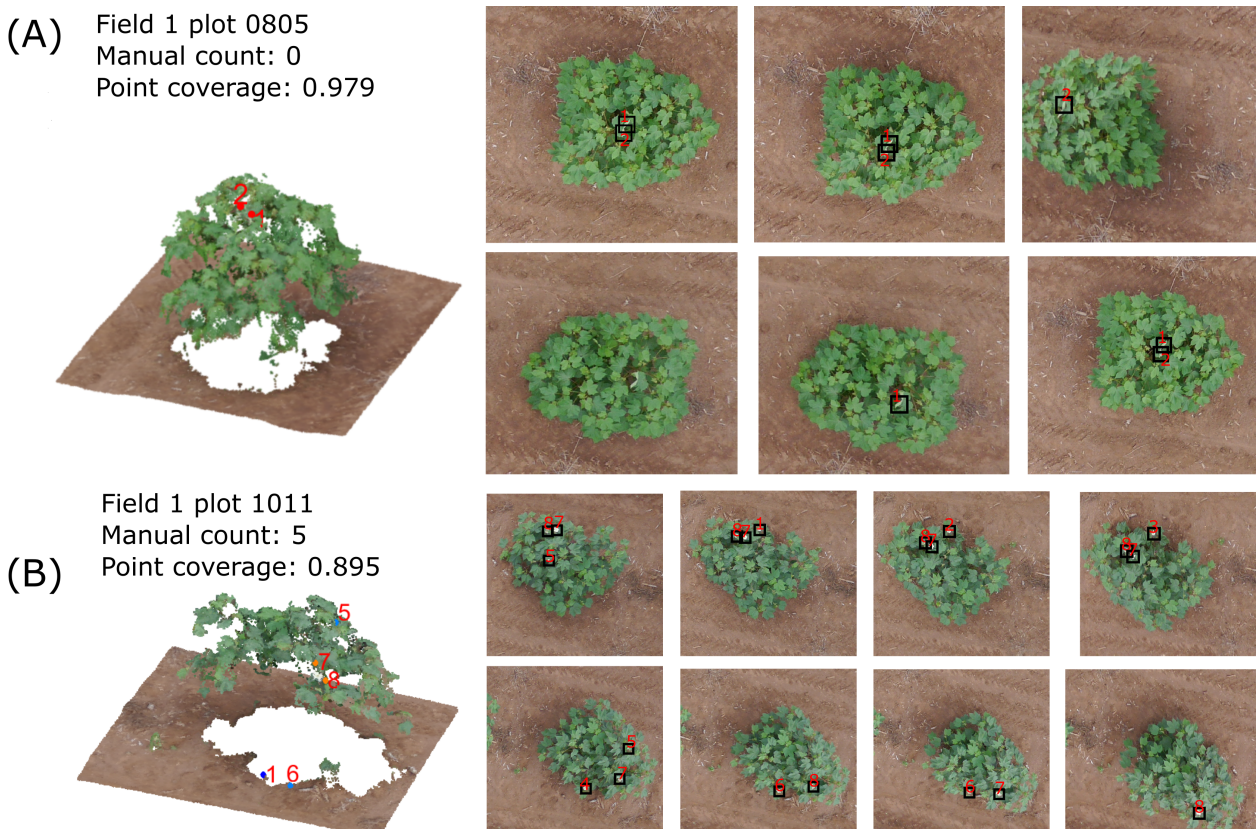

Figure 5.12: The bloom detection results for plot 0110 (A) and plot 1011 (B) in field 1 on 8/12/2016 dataset. The left images show the point cloud and detected blooms and right images show the corresponding blooms in the raw images.

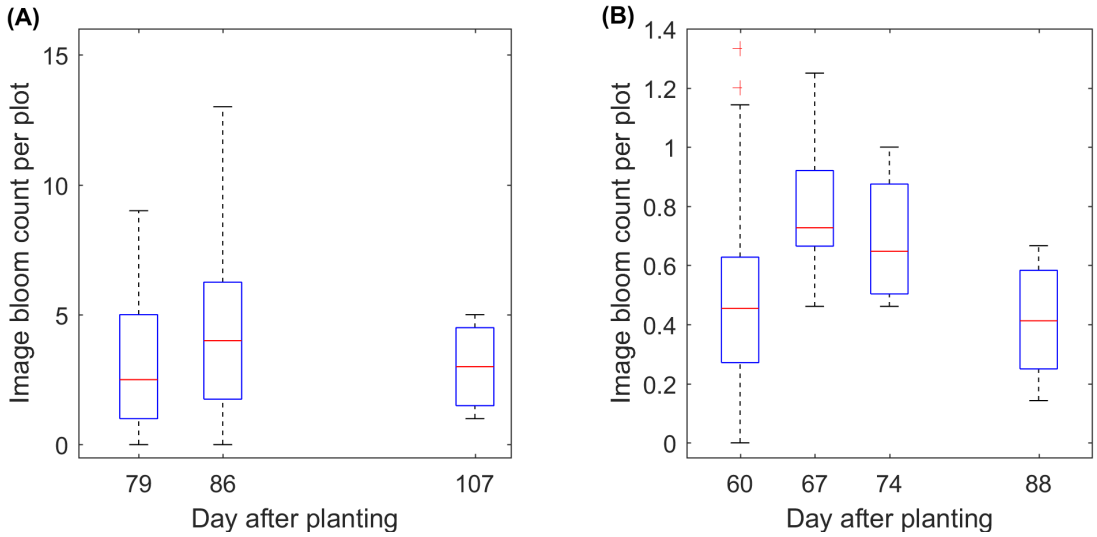

Figure 5.13: Boxplot of the image count per plot over time for field 1 (A) and field 2 (B).

5.4 Discussion

This paper demonstrated a high-throughput methodology to count cotton blooms using aerial color images. Continuously monitoring the cotton blooms over time can provide information about the cotton growth status, such as the flowering time and peak flowering time, which can be used for production management and yield estimation. It is also helpful for breeding programs to identify short season or long season genotypes.

Unlike sorghum or corn, whose flowers open at top of the stem, cotton flowers open from the bottom and progress up the plant. At the early cotton flowering stage, blooms at the bottom can be covered by the leaves and may not be able to be captured by the aerial images. Therefore, it is expected that image counting can underestimate the real bloom count. Instead of using the ortho-images which only take the top view of the canopy, we utilized all the raw images that take different views of a plot to get the bloom count in order to improve the underestimation to a certain extent. The inability to detect hidden blooms inside the canopy from aerial images is the major limitation of the proposed methodology. The underestimation could be improved at the middle and later flowering stages since the flowers that open at the middle and top of the canopy are more likely to be imaged in the aerial images. This issue could be addressed by using oblique aerial images or ground side-view images.

Besides the inability to detect hidden flowers, the proposed bloom counting method can generate errors from two aspects: the bloom detection error and the bloom registration error. The bloom detection error was affected by the image quality, the threshold to select the potential bloom images, and the accuracy of the CNN. The image quality was affected by

the illumination condition at the time of data collection. Ideally, the best image quality can be obtained when the illumination is uniform from all directions so there are no shadows or specular highlights. The ideal condition is hard to obtain in the field, but the approximate ideal condition can be obtained on a cloudy day. When the sunlight directly shines on the canopy without being scattered by clouds, shadows and specular highlights can be found in the aerial images, which cause non-uniformed intensity changes of the plot. Those changes will affect the selection of the potential bloom positions.

The threshold to select the potential bloom positions was arbitrary and mainly based on the images. A high threshold can eliminate some blooms that have low intensity due to shadows. A low threshold can include more non-bloom objects (such as the specular highlights), which can increase the processing time and be misclassified by the CNN. The classification of the potential bloom images relied on the CNN but misclassifying a bloom image as a non-bloom image affected the result differently from misclassifying a non-bloom image. When a bloom was classified as a non-bloom in one image, it was possible to be correctly classified as a bloom in other images given a different perspective of the flower; therefore, this flower could be included in the final result. However, when a non-bloom image was classified as a bloom, this false bloom was counted and there was no approach to remove it in the current method. The training samples for the CNN were selected from images collected on only four different days, so the trained CNN may not be suitable for data collected in different growth stages, especially when the color of the cotton leaves changes over the season. Therefore, including more training samples from different dates over the growth season could improve the accuracy and robustness of the CNN. The different appearances of the cotton

flower caused by genotype differences—for instance, the Pima cotton has yellow bloom in contrast with white bloom for some upland cotton varieties—should also be considered when constructing the training samples.

The accuracy of the bloom registration was affected by the 3D position of the bloom and the registration algorithm. The 3D position of the bloom was affected by the pixel location and the dense point cloud. Therefore, the accuracy of the dense point cloud has an important impact on the accuracy of a bloom's 3D position, and thus affects the bloom registration results. A larger error of the 3D position can cause the bloom registration to under-cluster or over-cluster the blooms. Under-clustering can count the same bloom more than once and over-clustering may count several blooms as one bloom, making the bloom count unreliable. The completeness of the point cloud also affects the registration result. If the dense point cloud cannot cover the whole canopy, some blooms that are detected in the images may not have a valid projection on the dense point cloud. Those blooms will not be registered and underestimation of the bloom count will occur. Therefore, adequate image overlap is critical in data collection to capture the whole canopy. Increasing the image overlap can improve the completeness, but also requires more data collection and processing time. Oblique imagery can also improve the completeness of the point cloud by providing more views of a plot and potentially can image more occluded flowers.

Although the proposed bloom counting methodology usually provides an underestimated bloom count compared to manual counting, it has the advantage in throughput over manual counting. It saves manual labor and makes continuously monitoring the flowering possible. Without such throughput, it is impossible to continuously monitor the flowering stage and

determine the flowering time, peak flowering time, and seasonal bloom count. For farmers and breeders, it is helpful to estimate the fiber yield because some studies have shown the fiber yield is correlated with seasonal bloom count [163, 164]. However, additional studies on how well the proposed methodology can estimate fiber yield are needed. The methodology is also helpful for differentiating the growth behavior among different genotypes, which can be used to select certain genotypes, such as short-season or long-season genotypes. Compared to other flower detection methods that are based on the percentage of flower pixels, the method proposed in this study can directly provide flower count without exploring the correlation between pixel percentage and flower count [165]. The proposed method also can provide the locations of flowers as byproducts, which could be used to correlate with the cotton bolls (Figure 5.14).

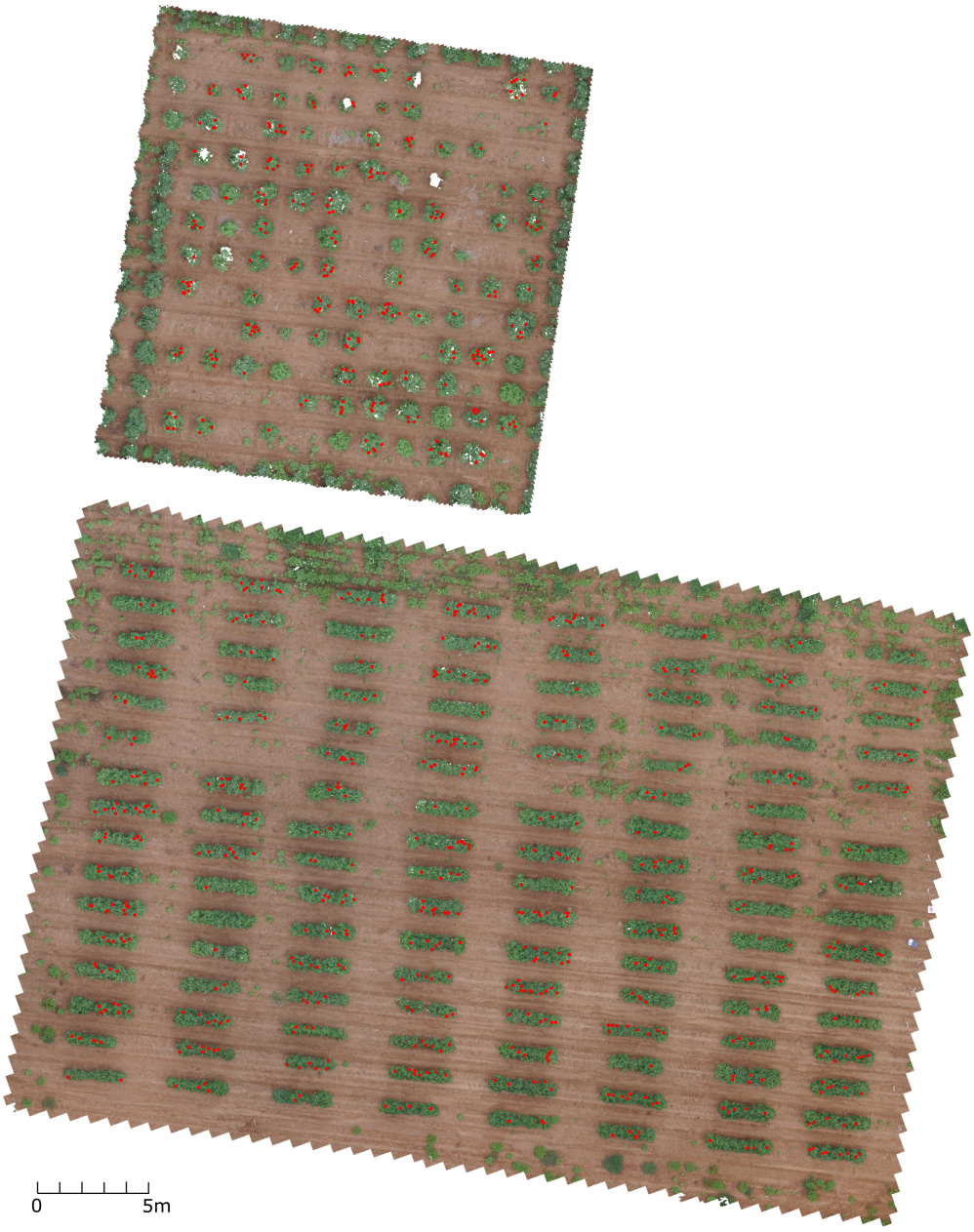

Figure 5.14: Top view of the two test fields with red dots indicating the flower locations using 8/12/2016 dataset. The image was rendered from dense point cloud.

To implement the proposed method, farmers and breeders can use commercial aerial photogrammetry systems or build custom systems to collect aerial images. The data processing pipeline can be used as long as the image quality (such as the ground resolution) meets

the requirements. This study used small fields of 920 m^2 (0.22 acre) and the data collection throughput is not enough for commercial farms or breeding programs with large fields. Although the throughput can be increased by increasing the flight altitude, the reduced ground resolution may not meet the requirement for the pipeline to correctly recognize cotton flowers. Alternative solutions include using high-resolution cameras to maintain the ground resolution when imaging at higher altitude.

5.5 Conclusion

This study developed a high throughput methodology for cotton bloom detection using aerial images, which can be potentially used to monitor cotton flowering over the season for cotton production management and yield estimation. The method generally underestimated the bloom count due to the inability to count hidden flowers, but the bloom count for the single plant layout was less likely to be underestimated than the multiple plant layout. The accuracy and completeness of the dense point cloud has an impact on the bloom count result, so generating a good dense point cloud can improve the results significantly. The bloom registration algorithm developed in this study was efficient in terms of runtime and was more prone to under-clustering but less prone to over-clustering. The trained CNN correctly classified more than 97% of the training and test images, and more than 90% of the potential flowers extracted from individual datasets. Since the false classification from the CNN can result in false bloom count, designing a robust CNN that can handle images taken under different field illumination conditions and cotton growth stages will be included in future

studies. In addition, oblique imagery will be explored to improve the quality of the dense point cloud.

Chapter 6

Development of The Modular Agricultural Robotic System[1]

[1]Rui Xu and Changying Li. To be submitted to *Journal of Field Robotics*.

Abstract

Increasing global population and climate change pose significant challenges for meeting food and fiber demand, and the agricultural robot is a promising solution to these challenges. This paper presents the modular agricultural robotic system (MARS) from concept design to real implementation. The MARS is a low-cost, multipurpose agricultural robot that uses a modular design for the hardware and software. There are five essential hardware modules (wheel module, connection module, robot controller, robot frame, and power module) and three optional hardware modules (actuation module, sensing module, and smart attachment). Various combinations of the hardware modules can create different robot configurations for specific agricultural tasks. The software was designed using the Robot Operating System (ROS) with three modules: control module, navigation module, and vision module. A robot localization method using dual Global Navigation Satellite System (GNSS) antennas was developed. Two line-following algorithms were implemented as the local planner for the ROS navigation stack. Based on the MARS design concept, two MARS designs were implemented: a low-cost, lightweight robotic system named MARS mini and a heavy-duty robot named MARS X. The MARS X was tested for its performance and navigation accuracy, achieving a high accuracy over a 537 m long path with 15% of the path having an error larger than 0.05 m. The MARS mini and MARS X were shown to be useful for plant phenotyping through two field tests. The modular design makes

the robots easily adaptable to different agricultural tasks and the low-cost makes it affordable for researchers and growers.

6.1 Introduction

The projected global population will reach 9 billion by 2050, which will require the current food production to double to feed the global population [1]. Modern agriculture also faces challenges in climate change, farmland loss, and labor shortage. Agricultural scientists and engineers strive to find solutions to increase production with higher quality in a sustainable way, and some have achieved limited success. For example, agricultural machinery has improved dramatically the productivity and efficiency of many agricultural tasks, making managing large farmland possible with only a few workers. Another successful example is found in precision agriculture (or digital agriculture), which integrates sensors, controls, and information technologies to continuously monitor and manage crops in order to increase production with fewer resources. Similarly, high-throughput phenotyping requires collecting phenotypical data for plants at a large spatial scale and high temporal resolution. Agricultural robots can play an important role to automate these processes [172].

Agricultural robots have several advantages over traditional agricultural machinery. Agricultural robots can be lightweight, so they create little or no soil compaction and are less limited by field conditions than heavy tractors. Agricultural robots can be autonomous and intelligent, so they are less dependent on labor. They are also more suitable for some agricultural tasks and crops that require a certain level of intelligence, such as harvesting

specialty crops [173, 174, 175]. For irrigation and weeding tasks, agricultural robots can control precisely the irrigation and weeding spots to reduce water and pesticide usage [176, 54, 33]. Several robots can form a robot swarm to cover a large field and can reduce the chance of disruptions resulting from a single machine failure that might occur in traditional agricultural machinery [177, 178].

The adoption of robotics in agriculture has been generally slower than in other industries, primarily because of the complexity of the tasks in uncontrolled agricultural environments. Unlike industrial robots in a controlled environment, most agricultural robots need to work in an uncontrolled environment or semi-structured environment, thus requiring one or multiple sensors to achieve intelligence and autonomy. This can make the robot costly. Thanks to recent advancements in sensing technologies, sensors are becoming less expensive and more robust, allowing researchers and companies to create affordable robots. Some agricultural robots have achieved limited intelligence by combining computer vision and artificial intelligence, a process that has demonstrated considerable potential for harvesting robots and weeding robots. Most robots can be driven autonomously using Global Navigation Satellite System (GNSS) and Inertial Measurement Unit (IMU), and avoid obstacles using the imaging sensors or LiDAR.

Many agricultural robots have been developed by both research labs and technology companies. For example, the Australian Centre for Field Robotics has developed several agricultural robots for weeding and phenotyping, including Ladybird, Shrimp, and RIPPA [28, 26, 179]. Another well-known weeding robot is AgBotII, which can detect and classify weed using computer vision and remove the weed using a mechanical or spray weeding module [33].

Companies such as Blue River, EcoRobotix, F Poulsen Engineering, and Naio Technologies have also developed commercial weeding robots. Harvesting robots is an important research area, and some robots have been developed in academia to harvest apples [180], strawberries [51, 173, 181], sweet peppers [174], tomatoes [182, 183], and kiwi fruit [175]. In parallel, some companies have developed robot harvesters. For example, the Agrobot and Harvest CROO Robotics both developed the strawberry harvester, Energid Technologies developed a robotic citrus harvesting system, and Abundant Robotics developed an apple harvester. Phenotyping robots is another important research area in recent years. Robotanist is a skid-steering robot that can navigate autonomously within sorghum fields using abundant sensors equipped on the robot [27]. Vinobot and Vinoculer are two robotic platforms where Vinoculer is a mobile observation tower that can identify specific plants for further inspection by the ground robot Vinobot [24]. Phenobot is a robot for sorghum plant phenotyping [50]. TerraSentia is a low-cost, 3D printed field robot that can count corn stands using deep learning methods [23]. An autonomous mobile robot was developed for plant phenotyping using a LiDAR sensor and soil sensing with a multipurpose toolhead on a robotic arm [81, 31].

The above-mentioned agricultural robots are task-based and can perform specific tasks for specific field arrangements, but they are not suitable for other tasks. When considering the complexity of agricultural environments resulting from diversity in farm size, farm topology, and crops, growers must use different machines for different crops and production methods. This method is not cost-effective, especially for small farms. To address this issue, some researchers and companies have developed multi-purpose robotic platforms that can be used for different production methods. Some multi-purpose robotic platforms can carry different

attachments, but the physical appearance is fixed, such as BoniRob (BOSCH, German), Robotti (AGROINTELLI, Danish), and PUMAgri (SITIA, France). BoniRob is an adaptable, multi-purpose robotic platform that can be adapted to different tasks using exchangeable application modules [34]. Armadillo is another multi-purpose robotic platform that can carry different tools [35]. The robot is equipped with multiple sensors for row detection and navigation and has been used for weeding, soil measuring, and phenotyping [176, 59, 178]. Some robots have adopted modular designs to enable the robot's reconfiguration for different environments and tasks. One well-known example is the Thorvald II agricultural robotic system, which consists of several standardized robot modules that can form different robot designs [32]. The system also used modular design for its software using the Robot Operating System (ROS) [184]. This modular design creates great flexibility in creating robot designs for various environments, such as the greenhouse, polytunnel, and open field [53, 52].

Modularity can bring several benefits when incorporated into the design of agricultural robots. The biggest advantage is that the robot can be easily reconfigured and customized using different modules for various tasks. Because the modules can be reused for different robot configurations, the total cost of reusing the modules to perform different tasks would be much lower than the cost of several task-based robots. Furthermore, harsh field environments such as humidity, dust, high temperature, and rain can cause metal erosion and electrical problems, which can increase maintenance costs. With a modular design, individual parts can be replaced and repaired, making maintenance easier and cheaper.

The current commercial agricultural robots, such as the Thorvald II system, are still expensive for most small farms and research labs. In addition, their design is proprietary

so integrating third-party components and optimizing the design for specific tasks can be difficult. Therefore, we aim to develop a low-cost Modular Agricultural Robotic System (MARS) that is affordable for growers and researchers. There were four objectives for this study. The first objective was to describe the design concept and requirements for MARS. The second objective was to implement two types of MARS, a low-cost and lightweight MARS mini and a high payload MARS X, based on the design concept. The third objective was to develop software modules that can perform basic autonomous and intelligent tasks, such as autonomous navigation and object detection. The fourth objective was to test the robots' performance in an agricultural field and demonstrate their application for high-throughput phenotyping.

6.2 Design Concept

The Modular Agricultural Robotic System (MARS) is designed around hardware and software modularity (Figure 6.1). The modularity design provides maximum flexibility and reusability of the system by allowing users to reconfigure the system to their specific needs with little effort. Each module serves specific functions and, combining different modules can enhance the functionality of the entire robot.

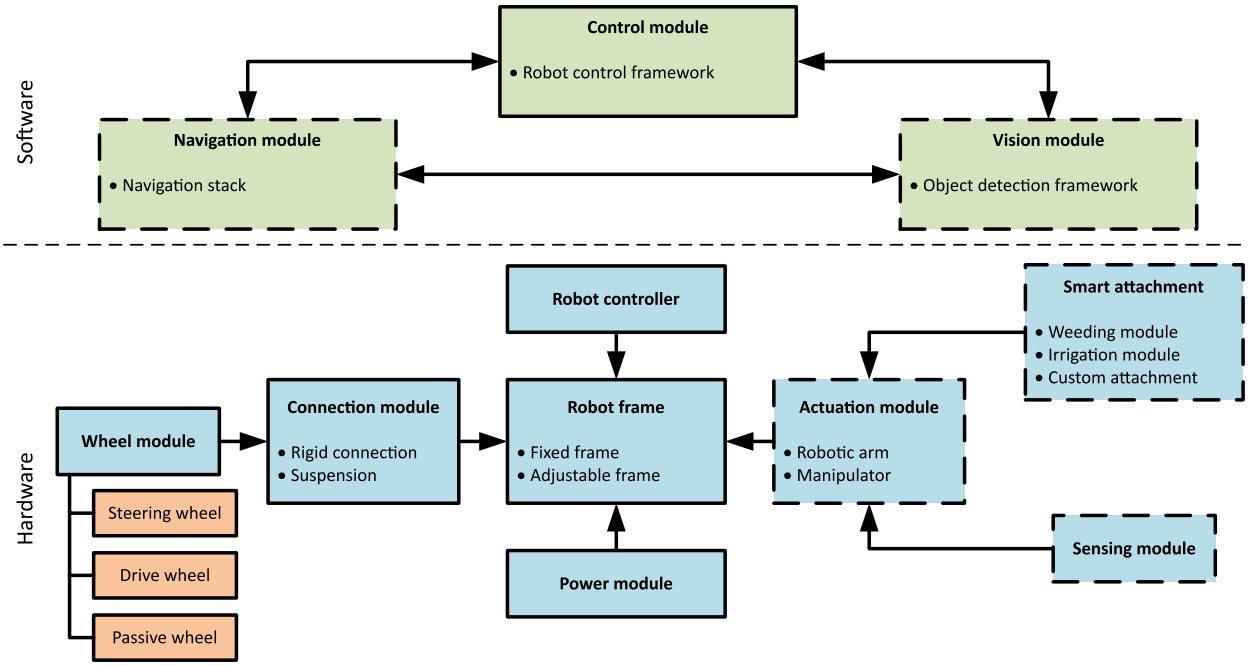

Figure 6.1: System diagram of the MARS. The rectangular boxes with the dashed line frame are optional modules. Arrows indicate how the modules are connected. The blue rectangles indicate the hardware modules and their components. The orange rectangles indicate the sub-modules.

6.2.1 Hardware modules

The MARS contains eight hardware modules, and each module includes several variations (Figure 6.1). The wheel module, connection module, robot frame, robot controller, and power module are necessary and can provide basic functionality. The actuation module, sensing module, and smart attachment are optional and can expand the robot's functionality. All the hardware modules have the same mechanical interface for easy assembling.

Robot frame

The robot frame is used to combine all other modules and provides the main structure of the robot. It provides the mount points for other modules and how other modules connected

to the robot frame governs the robot's basic functionality and structure. Because the robot can fit for various agricultural environments and applications, the robot frame can differ in width, height, and length. The robot frame should be designed using mechanical structures that are easy to assemble and modify to accommodate the robot frame's customizability. It should also sustain the robot's static weight and the dynamic force generated by the robot motion. The robot frame can be fixed or adjustable. The fixed frame has a rigid connection, and the overall dimension is not changeable. The adjustable frame can change the frame dimension, such as width and height, via slide mechanisms.

Wheel module

The wheel module consists of three sub-modules: steering wheel, drive wheel, and passive wheel. The steering wheel has a motor to rotate the wheel on the horizontal plane, which can change the wheel's drive direction. The drive wheel has a motor to propel the robot, while the passive wheel has a free rotation wheel that is mainly used to support the robot. The drive wheel or passive wheel can be attached to a steering wheel to form a steerable drive wheel or a steerable passive wheel. They also can be used directly as non-steerable drive wheels or non-steerable passive wheels. The wheel module is connected to the robot frame through the connection module. Because the steering wheel and drive wheel use motors, the related electronic components are also included in the wheel module (Figure 2). The electronic components include the motor driver and motor controller. They are powered by the power module and can be controlled by the robot controller through certain unified bus interface, such as Controller Area Network (CAN).

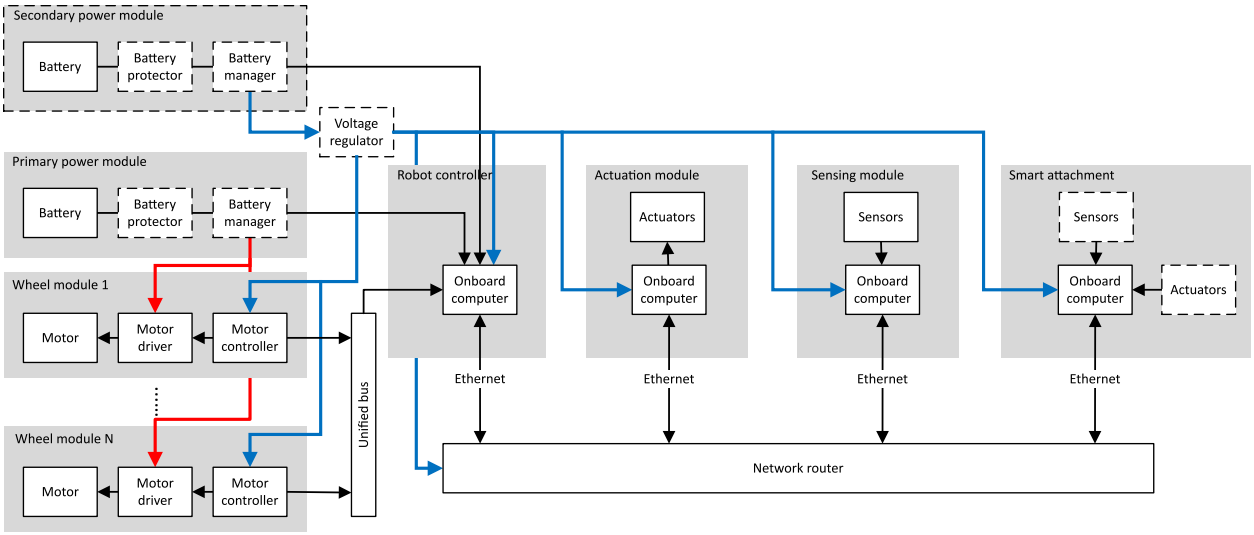

Figure 6.2: Diagram of the electronic connections for MARS.

A robot can have multiple wheel modules. Using more steering wheels can increase the maneuverability of the robot, and using more drive wheels can increase the payload of the robot. For example, a four-wheel steering four-wheel driving robot uses four steering wheels and four drive wheels (Figure 6.3), enabling the robot to follow any trajectory in its workspace.

Connection module

The connection module is used to connect the wheel module with the robot frame. It has two variations: rigid connection and suspension. The rigid connection fixes the wheel module on the robot frame. The suspension can increase the robot's ability of keeping the wheel in contact with the ground, which is useful moving through uneven terrain.

Robot controller

The robot controller is used to control the motion of the robot. The robot controller uses an onboard computer to control the motor in the wheel modules (Figure 6.2). It can communicate with other modules, such as the actuation module, the sensing module, and the smart attachment. The robot controller hosts the software control module.

Power module

The power module provides power for the entire robotic system. The power module includes the battery, battery manager, battery protector, and other electronic components that protect the battery and load. It may have a voltage regulator to regulate voltage to power other modules. The battery status can be monitored by the robot controller. A robot can have multiple power modules. For example, a robot can use one power module to power the wheel modules and one power module to power other modules. Having separable power sources and modules can protect them from the noise and back electromotive force induced by the motors.

Actuation module

The actuation module is an optional module that can be added to the robot for specific tasks, such as weeding, that require motion within the robot. The actuation module can be any robot arm and manipulator. The actuation module should have an independent electronic system and the controller can be added easily to the robot system.

Sensing module

The sensing module is optional and is used primarily to acquire information through sensors and cameras. The acquired information can be used to expand the robot's functionality and provide information for other modules. For example, in weeding tasks, the sensing module can have a color camera to identify and localize the weed and pass the information to the weeding module to remove the weed. The sensing module can be attached to the actuation module or directly attached to the robot frame.

Smart attachment

Smart attachments are optional modules that can expand significantly the robot's capacity to perform various agricultural tasks, including weeding, irrigation, and other customized tasks. The user can design their customized attachment for their specific tasks. The smart attachment is a standalone module that has an independent electronic system and controller. The smart attachment can communicate with and be controlled by the robot controller.

A complicated agricultural task may need coordination between the actuation module, sensing module, and the smart attachment. For example, the weeding task uses the sensing module to provide the location of the weeds, uses the actuation module to move the weeding module to the weed location, and uses the weeding module to remove the weed.

6.2.2 System configuration

The user can create different robot configurations using the hardware modules for different agricultural environments and applications. The required modules are the wheel module, connection module, robot frame, robot controller, and power module. Figure 6.3 demonstrated several robot configurations using the hardware modules. These configurations can be used for different agricultural environments and applications. Configuration 1 is a four-wheel-steering four-wheel driving robot with high maneuverability and can run on uneven terrain. Configuration 2 and configuration 3 are three-wheel robots, but configuration 3 has two drive wheels, giving more payload. They use fewer modules than other configurations so that the total cost will be cheaper. They can be used in the greenhouse. Configuration 4 is a four-wheel skid-steering robot, which can run between crop rows.

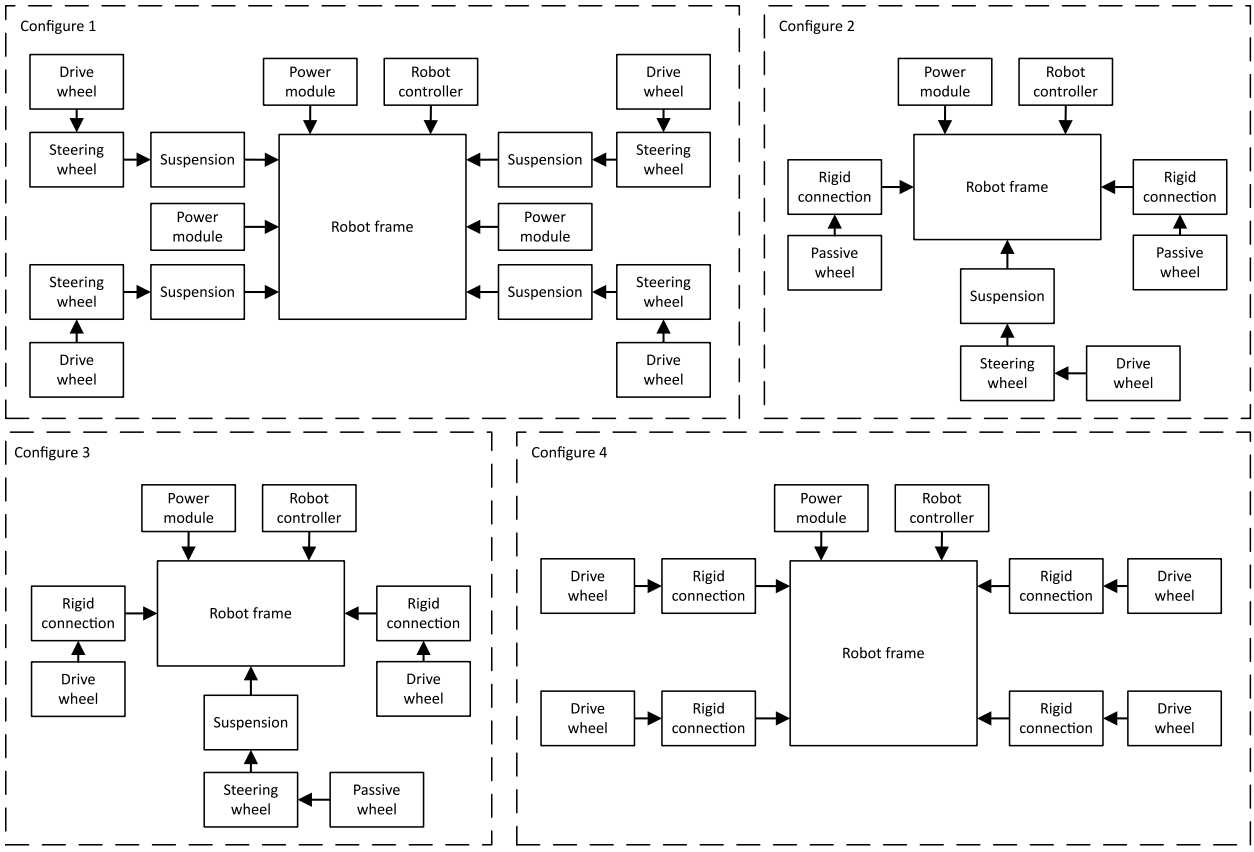

Figure 6.3: Example configurations of MARS. Configuration 1: four-wheel steering four-wheel driving robot. Configuration 2: front wheel steering front-wheel driving with two back support wheels. Configure 3: one wheel steering on the front and two drive wheels on the back. Configure 4: four-wheel skid- steering robot.

6.2.3 Software modules

The software modules include the control module, the navigation module, and the vision module, covering the most useful functions for most agricultural tasks. The software modules control the robot as a whole, and one software module can control one or more hardware modules. Each software module provides specific functions, but they can coordinate with each other to provide more complex functions, such as vision-guided autonomous navigation.

The software is implemented in ROS because ROS is modular and flexible. Several ROS packages were designed to implement the software modules. Specifically, a robot control framework was designed to enable the control module to control the robot's motion according to the robot's configuration, and an object detection framework was designed for the vision module to enable the detection and localization of objects in the image. For the navigation module, a GNSS-guided autonomous navigation module was developed using the ROS navigation stack.

Control module

The control module's main function is to control the motion of the robot, i.e., control the rotation of the steering wheel and the speed of the drive wheel. The control module is hosted by the robot controller. Different robot configurations have different kinematics. For example, a four-wheel steering, four-wheel driving robot has different kinematics from a skid-steering robot. The wheel module can use different motor and motor controllers for different agricultural applications, so the motor controller's ROS driver can vary. Therefore, inspired by the *move_base* package that uses the ROS *pluginlib* to create a reconfigurable navigation node, we use a similar idea to develop a robot control framework, named *robot_drive*, which can configure different kinematic model and motor drivers according to the robot configuration. A universal *motion_controller*, named *basic_motion_controller*, was implemented for any robot configurations.

Robot control framework The robot control framework consists of *motion_controller*, *wheel_driver*, which are managed by *robot_drive* (Figure 6.4). The motion_controller is a

plugin that models the kinematics and inverse kinematics of the robot. The *wheel_driver* is a plugin that interfaces with the wheel module. The *robot_drive* provides an ROS interface for configuring, running, and interacting with the robot control framework. It subscribes to the *cmd_vel* topic, which defines the velocity command (turning speed and rotation speed) of the robot and converts it to the command of the wheel module, which is the rotation of the steering wheel and the speed of the drive wheel. These commands are passed to the *wheel_driver* to be executed by the wheel module. The *robot_drive* can also calculate the odometry of the robot through the *motion_controller* using the status of the wheel modules retrieved by the *wheel_driver*. The odometry is then published as the *odom* topic. Because the *motion_controller* and *wheel_driver* are plugins, the users can implement their own *motion_controller* and *wheel_driver* based on their robot configuration. We provided BaseRobotController and BaseWheelDriver class to define the interfaces for the plugins. The users need to implement the interfaces for their *motion_controller* and *wheel_driver*.

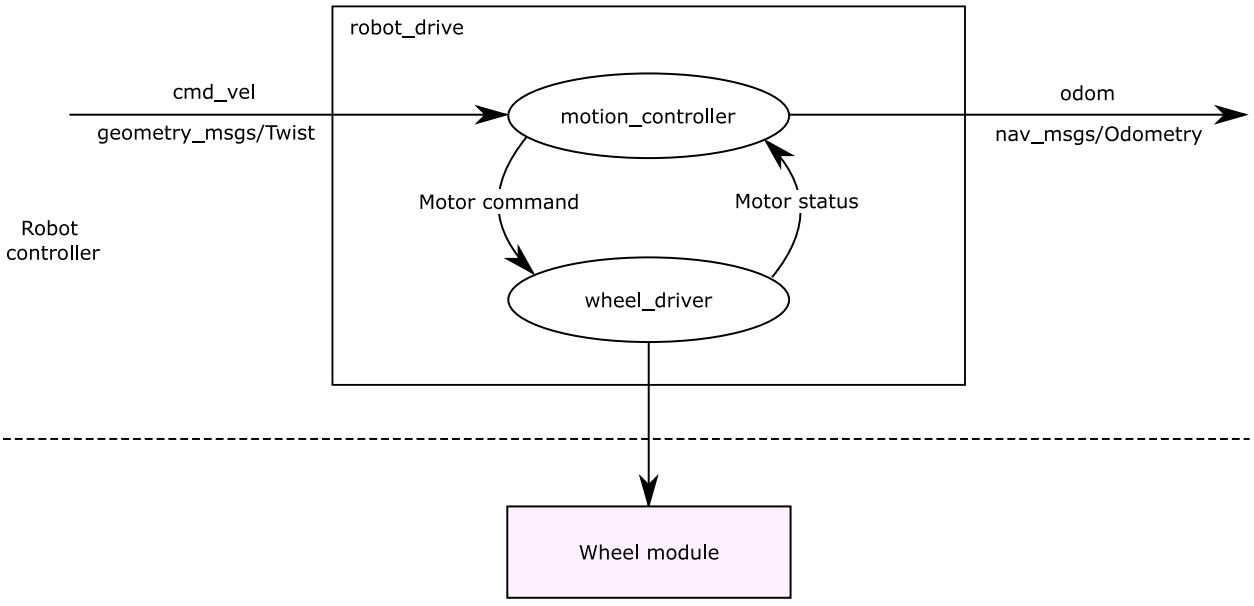

Figure 6.4: Diagram of the robot control framework.

Control the motion of the robot The *basic_motion_controller* used a simple kinematic model for the robot. It treats the motion of the robot as an instantaneous rotation around a time-varying point called the instantaneous center of rotation (ICR) (Figure 6.5). We assume the robot moves on a plane with linear velocity $\boldsymbol{v} = \begin{bmatrix} v_x & v_y & 0 \end{bmatrix}^T$ in the robot frame and rotates with an angular velocity $\boldsymbol{\omega} = \begin{bmatrix} 0 & 0 & \omega \end{bmatrix}^T$. Then the control input to the robot is $\eta = \begin{bmatrix} v_x & v_y & \omega \end{bmatrix}^T$, which is issued to the center of the robot (COR) (not necessarily the robot's center of mass (COM)). For each wheel, the velocity vector of the wheel's contact point with the ground ($v_i = \begin{bmatrix} v_{ix} & v_{iy} \end{bmatrix}$) is orthogonal to the straight line joining the point and the ICR.

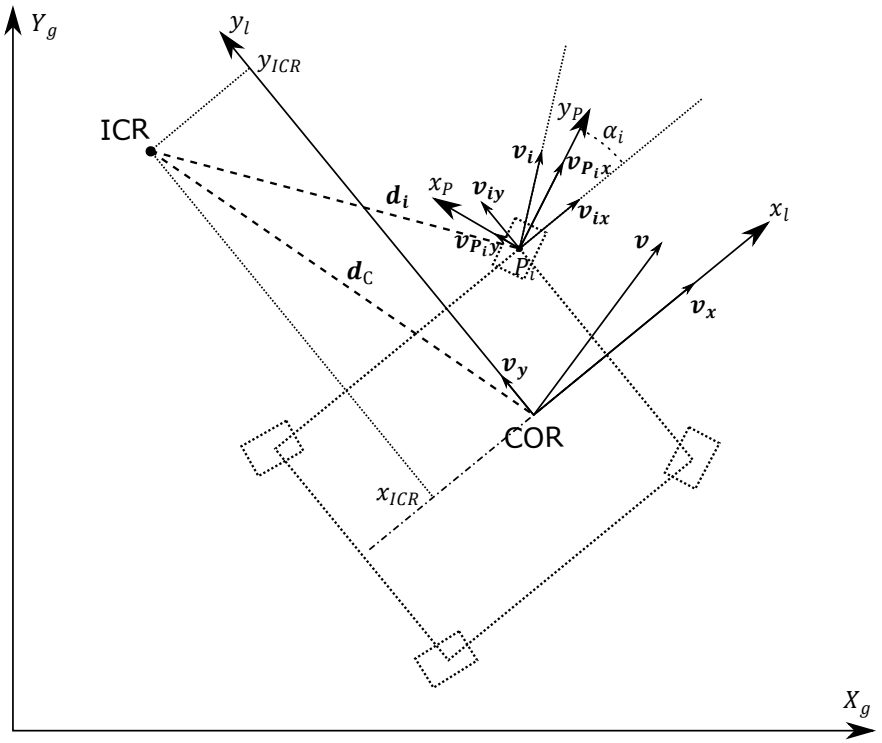

Figure 6.5: Wheel velocities. COR is the center of the robot. ICR is the instantaneous center of rotation. The velocity is expressed in the robot frame. $X_g - Y_g$ is the global frame. $x_l - y_l$ is the robot frame. $x_P - y_P$ is the wheel frame. The robot frame uses the COR as the origin.

In Figure 6.5, the radius vectors $\boldsymbol{d}_i = \begin{bmatrix} d_{ix} & d_{iy} \end{bmatrix}$ and $\boldsymbol{d}_C = \begin{bmatrix} d_{Cx} & d_{Cy} \end{bmatrix}$. Based on the geometry of Figure 6.5, the following expression can be derived:

$$\frac{\|\boldsymbol{v}_i\|}{\|\boldsymbol{d}_i\|} = \frac{\boldsymbol{v}}{\boldsymbol{d}_C} = |\omega| \tag{6.1}$$

or,

$$\frac{v_{ix}}{-d_{iy}} = \frac{v_{iy}}{d_{ix}} = \frac{v_x}{-d_{Cy}} = \frac{v_y}{d_{Cx}} = \omega \tag{6.2}$$

where the symbol $\|\cdot\|$ denotes the Euclidean norm.

If we define the coordinate of the ICR in the robot frame as $ICR = (x_{ICR}, y_{ICR}) = (-d_{Cx}, -d_{Cy})$, then we can rewrite equation 6.2 to get the following relationship:

$$x_{ICR} = -\frac{v_y}{\omega}, y_{ICR} = \frac{v_x}{\omega} \tag{6.3}$$

When the angular velocity is zero, the ICR is at an infinite position. If the position of the wheel in the robot frame is denoted as (P_{ix}, P_{iy}), then the vector $\boldsymbol{d}_i$ can be calculated by the following equations:

$$d_{ix} = P_{ix} - x_{ICR} = P_{ix} + \frac{v_y}{\omega} \tag{6.4}$$

$$d_{iy} = P_{iy} - y_{ICR} = P_{iy} - \frac{v_x}{\omega} \tag{6.5}$$

For the i-th wheel module, the velocity vector of the wheel's contact point with the ground ($\boldsymbol{v}_i$) is orthogonal of the straight line joining the point and the ICR. Combining equation 6.2,

6.4 and 6.5, $\boldsymbol{v}_i$ can be calculated as the following equation:

$$v_{ix} = v_x - \omega P_{iy} \tag{6.6}$$

$$v_{iy} = v_y - \omega P_{ix} \tag{6.7}$$

For each wheel, the velocity vector of the ground contact point is decomposed as the velocity along the wheel direction (longitudinal velocity) and perpendicular to the wheel direction (lateral velocity). Based on the geometry of Figure 4, the longitudinal velocity (v_{P_ix}) and lateral velocity (v_{P_iy}) can be calculated using the following equations:

$$v_{P_ix} = v_{ix}\cos\alpha_i + v_{iy}\sin\alpha_i \tag{6.8}$$

$$v_{P_iy} = -v_{ix}\sin\alpha_i + v_{iy}\cos\alpha_i \tag{6.9}$$

or, after combining with equation 6.6 and 6.7,

$$v_{P_ix} = v_x\cos\alpha_i + v_y\sin\alpha_i + (P_{ix}\sin\alpha_i - P_{iy}\cos\alpha_i)\omega \tag{6.10}$$

$$v_{P_iy} = -v_x\sin\alpha_i + v_y\cos\alpha_i + (P_{ix}\cos\alpha_i + P_{iy}\sin\alpha_i)\omega \tag{6.11}$$

where α_i is the wheel direction.

The velocity command for the drive wheel ω_P is the longitudinal velocity v_{P_ix} divided by the wheel radius r_i, which is

$$v_{P_iy} = \frac{v_x \cos\alpha_i + v_y \sin\alpha_i + (P_{ix} \sin\alpha_i - P_{iy} \cos\alpha_i)\omega}{r_i} \tag{6.12}$$

The lateral velocity can cause lateral slip for the wheel and we usually want to make it as small as possible. For a non-steerable wheel, α_i is a constant, so it's not possible to change α_i to reduce v_{P_iy}. However, for a steerable wheel, we can change α_i to reduce v_{P_iy} to zero. In this case, α_i is calculated using the following equation:

$$\omega_i = \text{atan2}(\omega P_{ix} + v_y, v_x - \omega P_{iy}) \tag{6.13}$$

where $\text{atan2}(y, x)$ is the four-quadrant inverse tangent.

Using equation 6.11, we can get the steering command α_i for the steering wheel, which is essential to align the wheel direction perpendicular to the radius vector $\boldsymbol{d}_i$.

Estimate the motion of the robot When calculating the longitudinal velocity and lateral velocity for all the wheels, we can write equation 6.8 and 6.9 into matrix format, as

following:

$$\begin{bmatrix} v_{P_1x} \\ \vdots \\ v_{P_Nx} \\ v_{P_1y} \\ \vdots \\ v_{P_1y} \end{bmatrix} = \begin{bmatrix} \cos\alpha_1 & \sin\alpha_1 & P_{1x}\sin\alpha_1 - P_{1y}\cos\alpha_1 \\ \vdots & \vdots & \vdots \\ \cos\alpha_N & \sin\alpha_N & P_{Nx}\sin\alpha_N - P_{Ny}\cos\alpha_N \\ -\sin\alpha_1 & \cos\alpha_1 & P_{1x}\cos\alpha_1 + P_{1y}\sin\alpha_1 \\ \vdots & \vdots & \vdots \\ -\sin\alpha_N & \cos\alpha_N & P_{Nx}\cos\alpha_N + P_{Ny}\sin\alpha_N \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} \tag{6.14}$$

where N is the number of wheel modules. Equation 6.14 can also be used to estimate the linear speed and angular speed of the robot from the wheel speed. First, we need to make several assumptions: 1) the wheel speed is measured by wheel encoders, which means v_{P_ix} is known, 2) the wheel direction is provided by the steering wheel or fixed, which means α_i is also known and 3) the wheel direction is perpendicular to the line joining the ICR and the wheel's contact point with the ground, which means v_{P_iy} is zero. Assumptions 1 and 2 usually hold because the drive module and steering module have encoders to measure the wheel's speed and direction. Assumption 3 can hold for the steerable wheel module and for the non-steerable passive wheel if the passive wheel's direction is perpendicular to the line joining the COR and the wheel's contact point with the ground. Therefore, we can restrict the COR position and the robot's configuration to make assumption 3 valid for the non-steerable passive wheel. Assumption 3 is not always valid for the non-steerable drive wheel, and thus we can remove the lateral velocity equation from equation 6.14.

With the above assumptions, equation 6.14 can be rewritten as equation 6.15.

$$V_{(K+M)\times 1} = \begin{bmatrix} v_{P_1x} \\ \vdots \\ v_{P_Kx} \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} \cos\alpha_1 & \sin\alpha_1 & P_{1x}\sin\alpha_1 - P_{1y}\cos\alpha_1 \\ \vdots & \vdots & \vdots \\ \cos\alpha_K & \sin\alpha_K & P_{Kx}\sin\alpha_K - P_{Ky}\cos\alpha_K \\ -\sin\alpha_1 & \cos\alpha_1 & P_{1x}\cos\alpha_1 + P_{1y}\sin\alpha_1 \\ \vdots & \vdots & \vdots \\ -\sin\alpha_M & \cos\alpha_M & P_{Mx}\cos\alpha_M + P_{My}\sin\alpha_M \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} = M_{(K+M)\times 3} \cdot \eta \tag{6.15}$$

where K is the number of drive wheels and M is the total number of steerable and non-steerable passive wheels.

We can use linear least squares to estimate η from equation 6.13, which is $\eta = (M^T M)^{-1} M^T V$.

Navigation module

The navigation module uses the ROS navigation stack to implement autonomous navigation. The navigation module uses the sensing module to provide localization information. The sensing module can contain various sensors, and different combinations of the sensors can provide different capacities for navigation (Table 6.1). The user can configure the sensor based on the capacities that the navigation task requires. For example, a basic setting for autonomous navigation is using the GNSS and IMU to achieve waypoint navigation. If the user wants to add obstacle avoidance capacity to the navigation, the user can add the LiDAR sensor on top of the GNSS/IMU. In addition to the required sensors, the navigation module

also needs the results from the vision module to achieve certain capacities, such as selective obstacle avoidance and crop row following. In the agricultural environment, the robot can run over many obstacles, such as plant leaves and weeds. The selective obstacle avoidance can detect the obstacles and determine whether the robot can run over it, so it requires the vision module to tell what the obstacle is. For crop row following, the navigation module relies on the vision module to detect the crop row and correct the robot's heading accordingly. Currently, we have achieved waypoint navigation and obstacle avoidance using GNSS, IMU, and LiDAR. The selective obstacle avoidance and crop row following will be developed in future work.

Table 6.1: Navigation capacities for different sensor combinations. Y: the capacity can be achieved. V: the capacity can be achieved with the vision module.

Sensors	**Capacities**			
	Waypoint navigation	**Obstacle avoidance**	**Selective obstacle avoidance**	**Crop row following**
GNSS/IMU	Y			
LiDAR		Y		V
Color camera				V
GNSS/IMU+LiDAR	Y	Y		
LiDAR+Color camera		Y	V	V
GNSS/IMU+LiDAR+Color camera	Y	Y	V	V

Robot localization using dual GNSSs To localize the robot, the combination of GNSS and IMU typically is used to measure the robot's location and heading. Since the IMU is sensitive to magnetic interference from the surrounding environment, the IMU heading can be off from the true north and drift over time, which requires constant calibration and compensation. To avoid calibration, dual GNSSs can be used to measure the localization and heading simultaneously without using an IMU. Two GNSS antennas were mounted on

the robot with known positions in the robot frame, which are $g_1 = (x_1, y_1)$ and $g_2 = (x_2, y_2)$, respectively (Figure 6.6). The GNSSs also report their measured positions in the global frame, which are $G_1 = (X_1, Y_1)$ and $G_2 = (X_2, Y_2)$, respectively. Assuming the robot runs on a 2D flat surface, the transformation between the robot frame to the global frame can be calculated using the two antennas' position. Equation 6.16 and 6.17 showed the transformation of the position of the GNSSs from the robot frame to the global frame, which includes a scale factor c, a rotation matrix with rotation angle θ and a translation vector $\begin{bmatrix} X_C & Y_C \end{bmatrix}^T$. The scale factor c is included to compensate the discrepancy of the two antennas' distance in the global frame and local frame due to the measurement error. The heading of the robot in the global frame is same as the rotation angle θ and the position of the center of robot is $COR = (X_C, Y_C)$.

$$\begin{bmatrix} X_1 \\ Y_1 \end{bmatrix} = c \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} X_C \\ Y_C \end{bmatrix} \tag{6.16}$$

$$\begin{bmatrix} X_2 \\ Y_2 \end{bmatrix} = c \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} + \begin{bmatrix} X_C \\ Y_C \end{bmatrix} \tag{6.17}$$

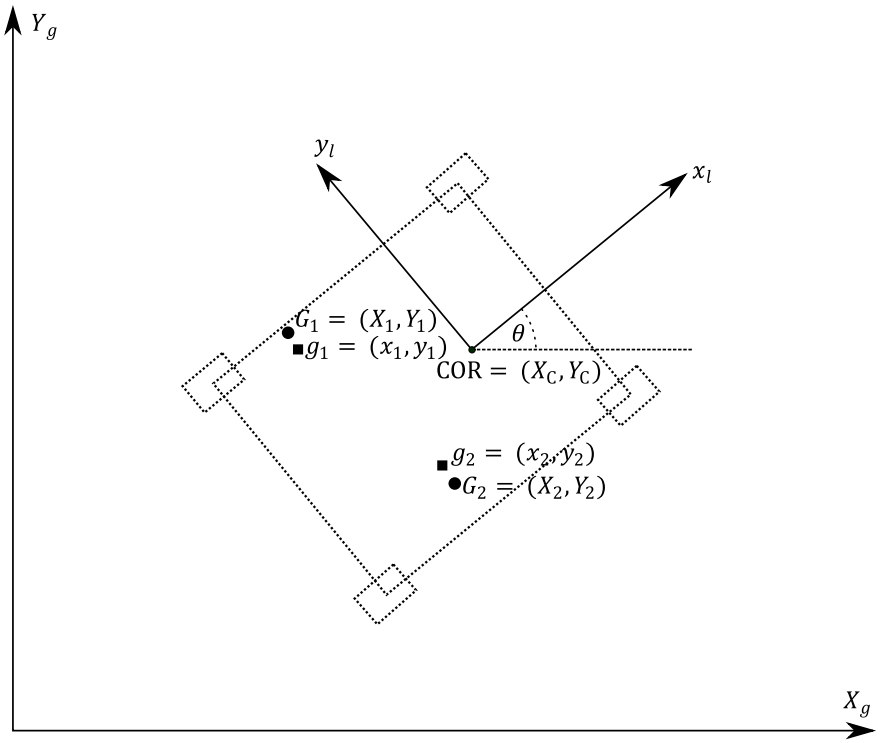

Figure 6.6: Dual GNSS configuration. COR is the center of the robot. $X_g - Y_g$ is the global frame. $x_l - y_l$ is the robot frame. θ is the heading of the robot. The robot frame uses the COR as the origin. The circles indicate the measured global positions of the antennas and the squares indicate the local positions. They do not exactly match with each because of the measurement error of the GNSSs.

From equation 6.16 and 6.17, the rotation matrix and translation vector $\begin{bmatrix} t_x & t_y \end{bmatrix}^T$ can be derived.

$$c = \frac{D}{d} \tag{6.18}$$

$$\cos\theta = \frac{d_x d_X + d_y d_Y}{d^2 c} \tag{6.19}$$

$$\sin\theta = -\frac{-d_y d_X + d_x d_Y}{d^2 c} \tag{6.20}$$

$$X_C = -\frac{x_2 d_x + y_2 d_y}{d^2} X_1 + \frac{x_1 d_x + y_1 d_y}{d^2} X_2 + \frac{x_1 y_2 - x_2 y_1}{d^2} d_Y \tag{6.21}$$

$$Y_C = \frac{x_2 y_1 - x_1 y_2}{d^2} d_X - \frac{x_2 d_x + y_2 d_y}{d^2} Y_1 + \frac{x_1 d_x + y_1 d_y}{d^2} Y_2 \tag{6.22}$$

Where $d_x = x_1 - x_2$, $d_y = y_1 - y_2$, $d = \sqrt{d_x^2 + d_y^2}$, $d_X = X_1 - X_2$, $d_Y = Y_1 - Y_2$, $D = \sqrt{d_X^2 + d_Y^2}$.

Let $a = \cos\theta$ and $b = \sin\theta$, then $\theta = \text{atan2}(b, a)$. The variance of the robot's heading σ_θ^2 can be estimated using the propagation of uncertainty.

$$\sigma_\theta^2 \approx \left|\frac{\partial\theta}{\partial a}\right|^2 \sigma_a^2 + \left|\frac{\partial\theta}{\partial b}\right|^2 \sigma_b^2 + 2\frac{\partial\theta}{\partial a}\frac{\partial\theta}{\partial b}\sigma_{ab} \tag{6.23}$$

$$\frac{\partial\theta}{\partial a} = -\frac{b}{a^2 + b^2} = -b \tag{6.24}$$

$$\frac{\partial\theta}{\partial b} = \frac{a}{a^2 + b^2} = a \tag{6.25}$$

where σ_a^2 and σ_b^2 is the variance of a and b, and σ_{ab} is the covariance between a and b. Assuming the easting and northing reported by the GNSS are independent, which means $\sigma_{X_1Y_1} = 0$ and $\sigma_{X_2Y_2} = 0$, then σ_a^2, σ_b^2 and σ_{ab}^2 can be calculated using the following equations.

$$\sigma_a^2 = \frac{d_x^2\left(\sigma_{X_1}^2 + \sigma_{X_2}^2\right) + d_y^2\left(\sigma_{Y_1}^2 + \sigma_{Y_2}^2\right)}{d^4 c^2} \tag{6.26}$$

$$\sigma_b^2 = \frac{d_y^2\left(\sigma_{X_1}^2 + \sigma_{X_2}^2\right) + d_x^2\left(\sigma_{Y_1}^2 + \sigma_{Y_2}^2\right)}{d^4 c^2} \tag{6.27}$$

$$\sigma_{ab} = \frac{\sigma_{Y_1}^2 - \sigma_{X_1}^2 + \sigma_{Y_2}^2 - \sigma_{X_2}^2}{d^4 c^2} \tag{6.28}$$

If the two GNSS has the same variance for the easting and northing, σ_{ab} can be treated as 0. The variance of the position of the robot in the global frame, $\sigma_{X_C}^2$ and $\sigma_{Y_C}^2$, can be calculated using the following equations.

$$\sigma_{X_C}^2 = \frac{(x_2 d_x + y_2 d_y)^2 \sigma_{X_1}^2 + (x_1 d_x + y_1 d_y)^2 \sigma_{X_2}^2 + (x_1 y_2 - x_2 y_1)^2 (\sigma_{Y_1}^2 + \sigma_{Y_2}^2)}{d^4} \tag{6.29}$$

$$\sigma_{Y_C}^2 = \frac{(x_2y_1 + x_1y_2)^2(\sigma_{X_1}^2 + \sigma_{X_2}^2) + (x_2d_x + y_2d_y)^2\sigma_{Y_1}^2 + (x_1d_x + y_1d_y)^2\sigma_{Y_2}^2}{d^4} \tag{6.30}$$

Autonomously line following Many agricultural tasks require the agricultural robot autonomously follow a preset path to perform the task. In most cases, the path is the crop row, which is usually a line. For the navigation module, we implemented two simple line-following algorithms. The first algorithm is the pure pursuit, which controls the robot constantly to pursue a virtual goal on the path with a lookahead distance [74]. The second algorithm (way-line controller) is described by [28], which only works for omnidirectional robot configurations, such as configuration 1 in Figure 6.3. The second algorithm can control the heading of the robot and path-following independently. The heading controller controls the turning rate to minimize the yaw error between the robot heading and target heading. The path-following controller controls the instantaneous translational motion vector angle to minimize the cross-track error (distance to the following line). Both algorithms were implemented as *base_local_planner* for the *move_base* package.

Vision module

The vision module mainly uses machine vision and machine learning to process images for object detection and localization and to provide information for various functions, such as visual-servoing navigation, weeding, and harvesting. To make the vision module flexible, we developed an object detection framework using ROS *pluginlib* that can be reconfigured for different object detection tasks.

The main goal of the object detection framework is to simplify the development of ROS nodes in order to detect objects from image topics. The core of the frame is the *object_detector*, which is a ROS plugin that can be dynamically loaded (Figure 6.7). By changing different *object_detector*, the user can customize the object detection methods according to the specific task. The *detect_object* provides a ROS interface for configuring, running, and interacting with the object detection framework. The *detect_object* also provides a localization function to estimate the 3D position of the object in the world frame using additional sensor sources, such as depth image from a stereo camera or point cloud from LiDAR.

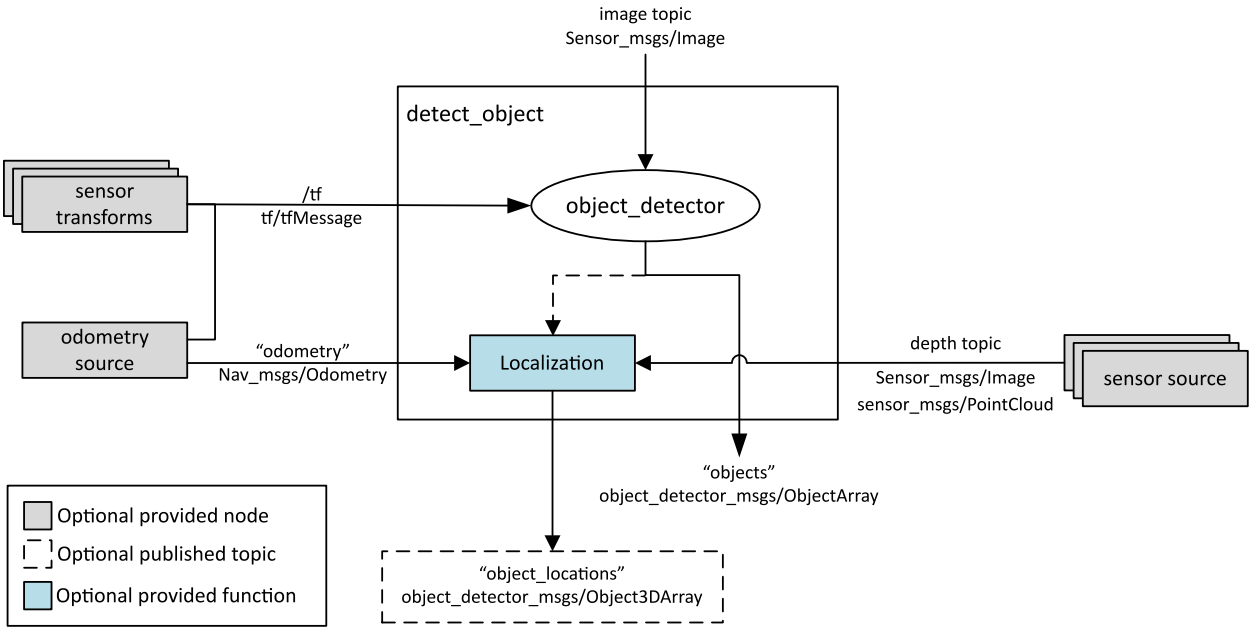

Figure 6.7: Diagram of the object detection framework.

The *object_detector* takes an image as input and outputs the bounding box of the detected objects in the image. The object detector also can output masks for the objects if necessary. The object detector is a class inherited from the base class *object_detector_core::BaseDetector* and *object_detector_core::BaseDetectorROS* (Figure 6.8). The *object_detector_core::BaseDetector*

class defines the interfaces that can be called by the *detect_object*. The user needs to override the two virtual functions: *void initialize(std::string name)* and *int detect(const cv::Mat & image)*. The initialization function is to initialize the *object_detector*, such as reading ROS parameters from the parameter server and configuring *object_detector*. The detect function is to run the detection algorithm and save the result as an *object_detector_msgs::ObjectArray* object. The *detect_object* will load the *object_detector* class and call the initialization function to initialize the class. Upon receiving an image message, the *detect_object* calls the detect function to get detection results. The detection result is accessible through the *getObjectArray* and *getObject* functions.

Currently, the object detection framework implemented two object detectors. The first detector is a simple thresholding detector that was implemented using OpenCV (OpenCV 3) [185]. The second detector is a deep learning detector implemented using the TensorFlow API [186]. It can accept any pre-trained model that complies with Tensorflow. With the deep learning detector, most of the object detection tasks can be achieved using an appropriate convolutional neural network.

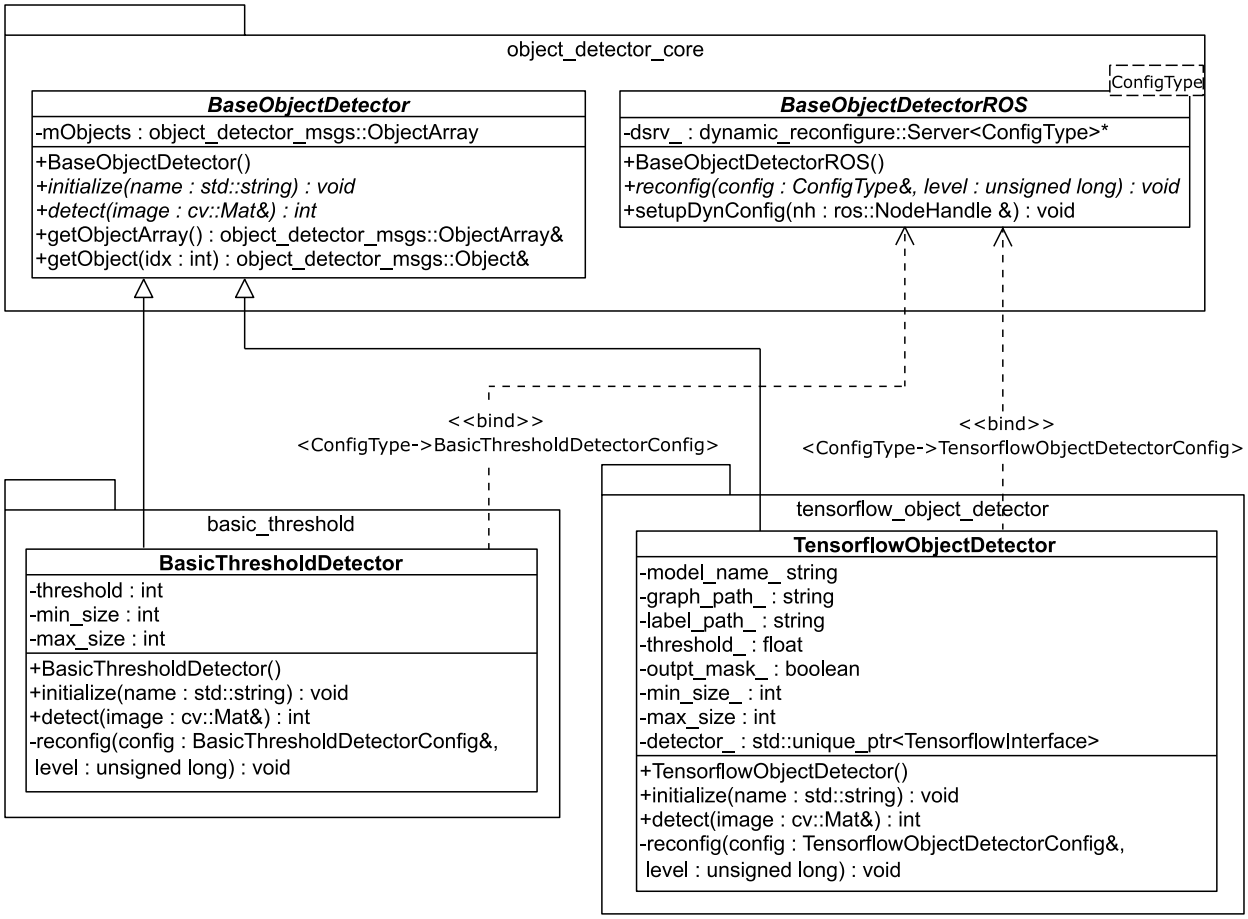

Figure 6.8: Class diagram of the *object_detector*.

6.3 Design Implementation

Using the design concept of MARS, we implemented two MARS designs: MARS mini and MARS X. The MARS mini is a lightweight and low-cost design best suited for light-duty applications. The MARS X is a four-wheel steering, four-wheel driving robot with a high clearance robot frame that can be used for heavy-duty applications.

6.3.1 MARS mini

The design goal of the MARS mini was to create a set of hardware modules and robot configurations using low-cost, off-the-shelf parts and 3D printed parts. Figure 6.9 shows all currently available hardware modules, and more modules will be designed and released through GitHub (https://github.com/UGA_BSAIL/MARS_Mini). The colored parts (except for the passive wheel) can be 3D printed. With 3D printing, the user can easily create customized parts for specific purposes. The robot frames are made of aluminum extrusions, making them easy to modify and attach to other modules. Figure 6.10 shows four possible robot configurations using the hardware modules, and Figure 6.11 shows two robot configurations that have been fabricated. All the 3D printed parts are printed in polylactide (PLA) using a 3D printer (Ultimaker S5, Ultimaker B.V., Netherlands).

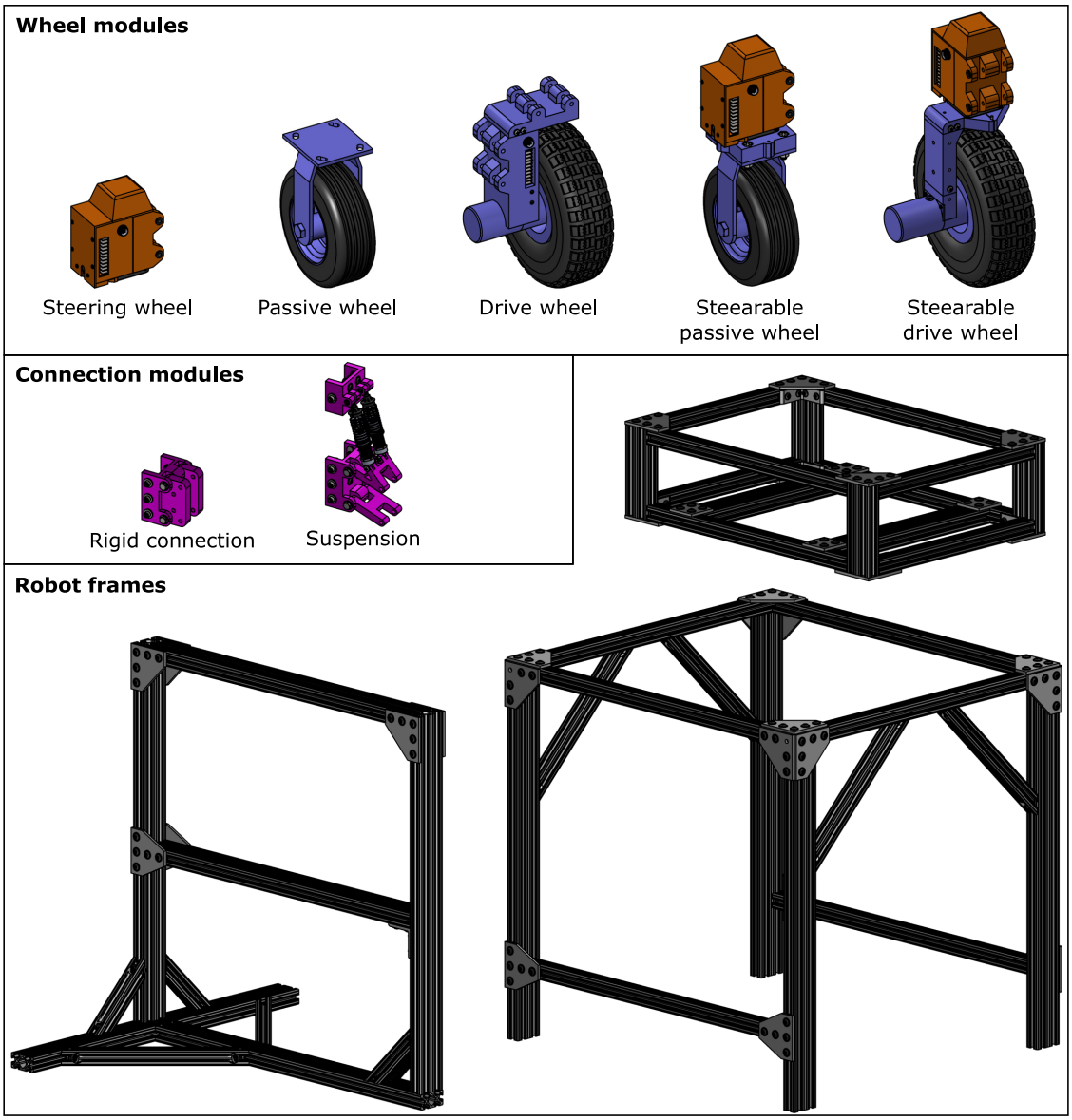

Figure 6.9: Modules of the MARS mini.

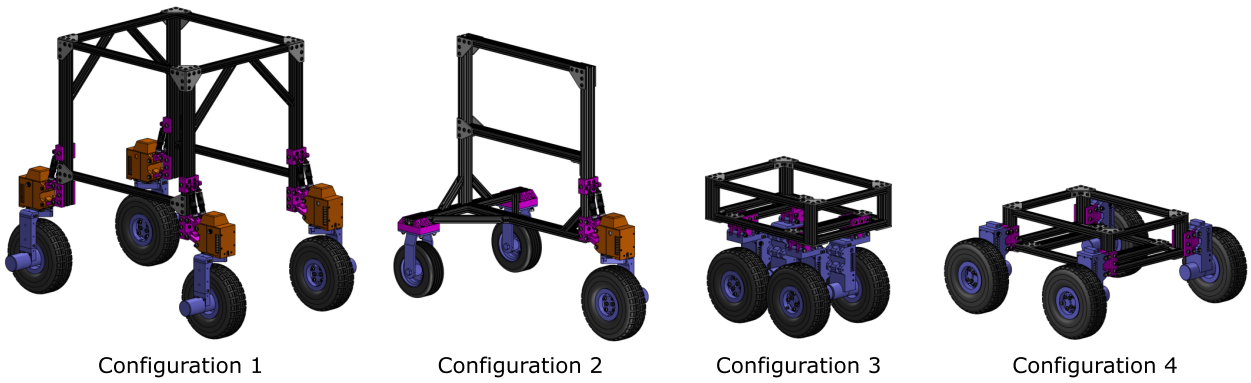

Figure 6.10: Four robot configurations of the MARS mini. Configuration 1: four-wheel steering four-wheel driving robot. Configuration 2: front wheel driving front wheel steering tricycle robot. Configuration 3: four-wheel skid steering robot with narrow width. Configuration 4: four-wheel skid steering robot with wide width.

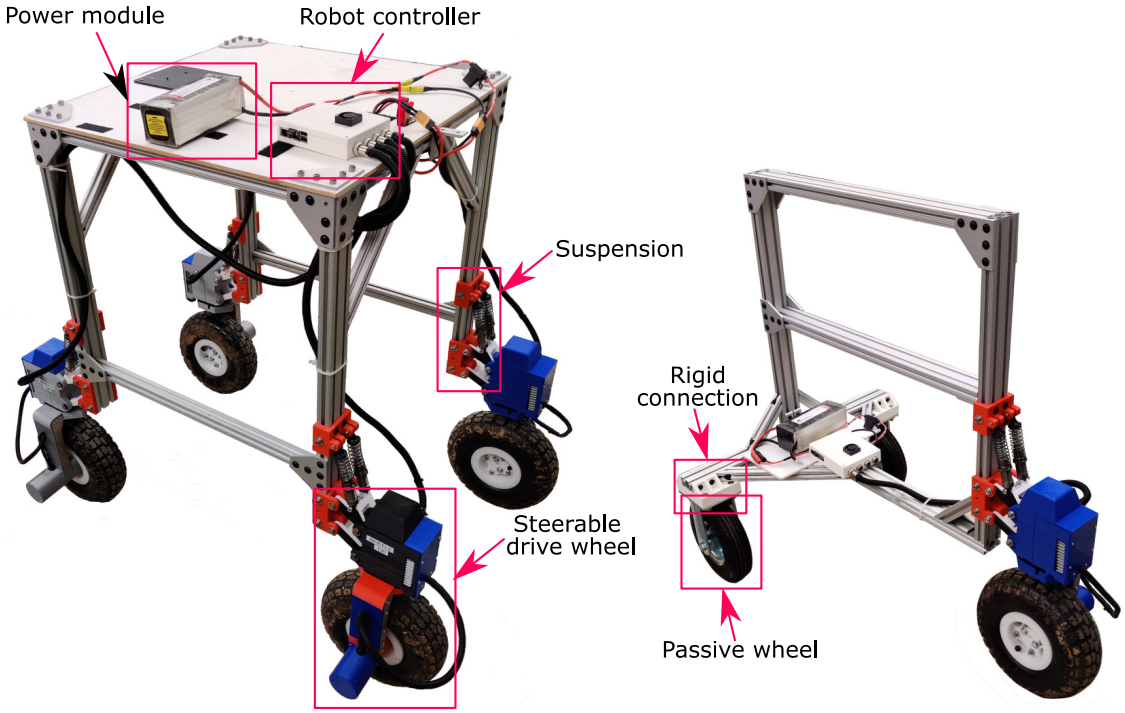

Figure 6.11: Two fabricated robot configurations from figure 6.10. Some parts are slightly different because the design of some parts was changed after printing.

The steering wheel uses a 24 V DC motor and reduction gearbox with a reduction ratio of 504. A magnetic encoder is attached to the motor to measure the motor speed. Similarly, the drive wheel uses the same setup but with a reduction ratio of 84, which gives a top speed of

1 m/s with a 10-inch wheel. Both motors are driven by a Sabretooth dual 12A motor driver (Dimension Engineering, OH, USA), and the motor driver was controlled with a Kangaroo x2 motion controller (Dimension Engineering, OH, USA). We used one Sabretooth dual 12A motor driver and one Kangaroo x2 motion controller to control two motors in the steering wheel and driving wheel when they form a steerable driving wheel.

The wheel module has a 5-pin, circular connector for electronic connection, with two pins for power and three pins for serial communication. The wheel modules can connect to the same serial port for communication. A Raspberry Pi 3 was used as the robot controller and a 24 V 16 Ah lithium-ion polymer battery (LiPo) pack as the power module.

The MARS mini uses the robot control framework to control the motion of the robot. A *wheel_driver* was implemented to interface with the Kangaroo x2 motion controller.

6.3.2 MARS X

Unlike the MARS mini, the MARS X is a heavy-duty robot that is made from metal. The MARS X was designed to have a high payload so that it can be used to carry heavy sensors and equipment. Similar to robot configuration 1 in Figure 6.10, the robot has four steerable drive wheels. The robot frame is made of aluminum extrusions with an opening in the middle so the robot can inspect the crop from the top (Figure 6.12). The MARS X uses suspension to connect the wheel module with the robot frame so the robot can run on uneven terrain. The total weight of MARS X is estimated to be about 500 kg, which is less than 1/10 of the average weight of a tractor.

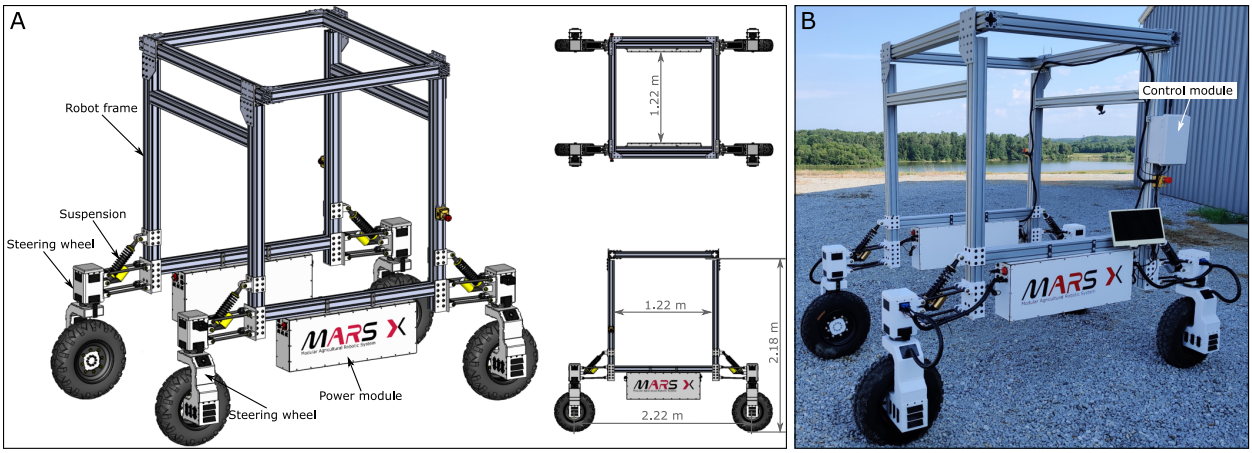

Figure 6.12: Mechanical structure of the MARS X. A) Rendering image. B) Real robot.

The wheel module uses two high power, brushless servomotors (SM34165DT, Moog Animatics, CA, US) for steering and driving. The servomotor can provide 1.45 N m continuous torque and has a built-in motor driver and motor controller. The servomotor connects to a gearhead (CSF-32-XX-GH, Harmonic Drive LLC, MA, US) to reduce the speed and increase the torque. The reduction ratio is 100 for the steering wheel and 80 for the drive wheel. The robot uses two power modules with each module on each side of the frame (Figure 6.13). The power module is made of 12 60-Ah lithium-iron-phosphate battery cells (LFP-G60, AA Portable Power Corp., CA, US) and a battery management system (BMS1060A, Roboteq, Inc., AZ, US). The battery cells are connected in series to provide a power source with 38.4 V. A DC-DC converter is used to provide 24 V source for the servomotor and other electronic components.

The robot uses a Jetson Xavier to control the robot. Because the servomotor uses RS232 for control and communication, an RS232 to USB converter was used to connect the servomotor in the wheel module with the Jetson Xavier. For better cable management, the

RS232 to USB converter first connects to a USB hub, then connects to the Jetson Xavier (Figure 6.13).

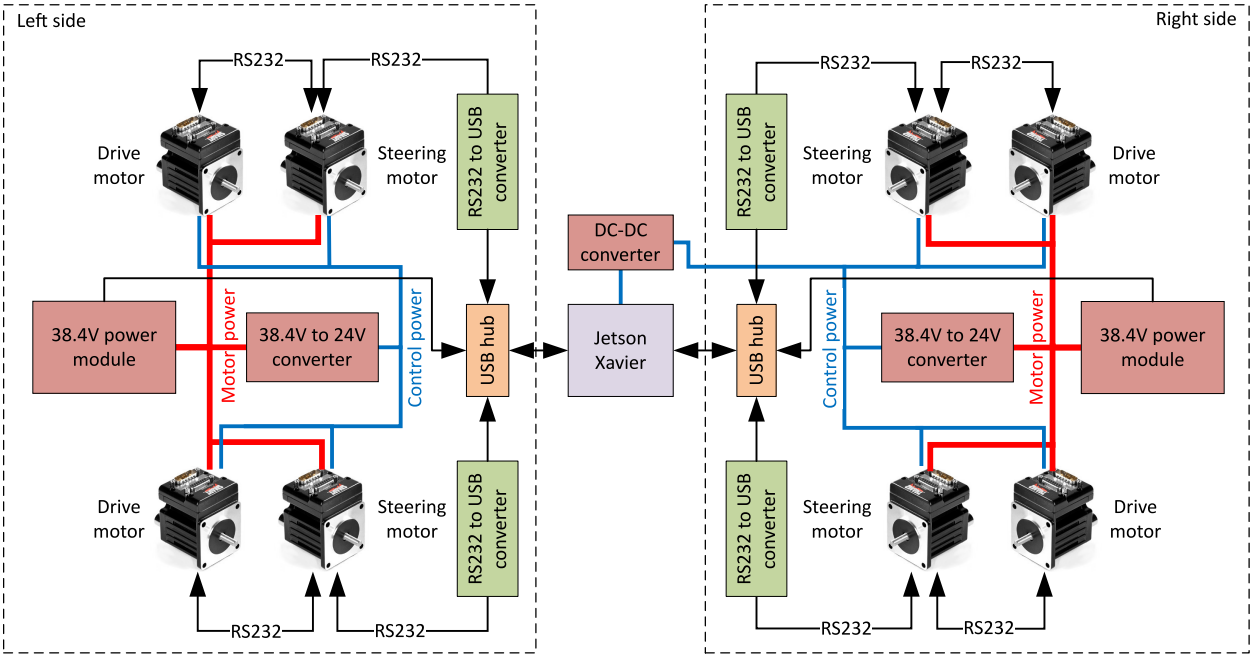

Figure 6.13: Electronic diagram of the MARS X.

The MARS X uses the same software setup as the MARS mini, except that a new *wheel_driver* was developed for the different motor used in MARS X.

6.4 Field Test

6.4.1 Mobility test

Two mobility tests were performed for the MARS mini and MARS X configuration 1 robot. The first test looked for how well the robots can pass obstacles, and the second tested the ability to climb an incline. The two tests were performed at a research farm and recorded by a static camera (videos are in Supplementary Materials).

Passing over obstacle

Several wood blocks were placed on a concrete floor in sequence to simulate the obstacles in the field (Figure 6.14). The robots were teleoperated to run through the obstacles at a speed of 0.2 m/s. The robots moved forward and backward to pass the obstacles twice. A camera was used to take the video of the test.

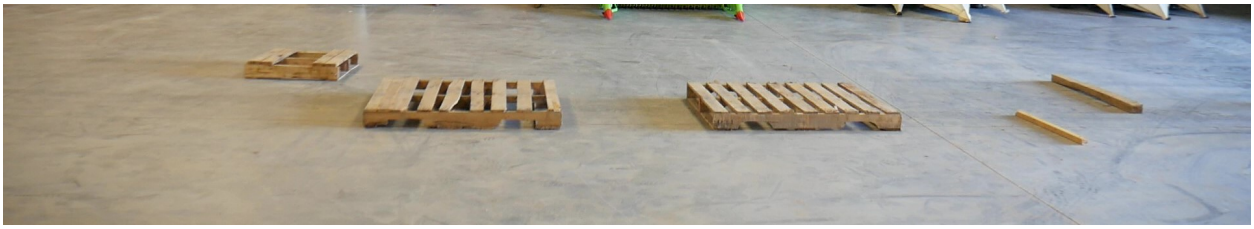

Figure 6.14: Obstacles used in the test.

Incline test

Each robot was driven manually (teleoperated) to a 30 degree step incline and stopped on the incline. Then the robot was driven manually to reach the top of the incline. The incline was depicted in Figure 6.15.

Figure 6.15: Incline test for MARS X.

6.4.2 Navigation test

The two implemented path-following algorithms were tested in a cotton field using MARS X because the MARS X is wide enough to traverse over the cotton plants at the late growth stage. The navigation performance of the MARS mini configuration 1 robot was evaluated in one high-throughput phenotyping test (Section 6.4.3). The cotton field was located at a research farm in Watkinsville, GA. It has 11 crop rows, and each row contains 8 plots. Each plot is about 3 m long, and there is 1.5 m long alley between the plots. The distance between the crop rows is 1.8 m. The robot was set to traverse over the crop row. Two RTK-GNSSs (Smart6L, Novatel, Canada) were mounted on the robot for localization. The

robot_localization ROS package was used to fuse the robot's location and heading from the dual RTK-GNSSs and the local speed of the robot from the motor.

Twenty-four waypoints were set so that the robot can go through all the crop rows with a total travel length of 537 m. The lookahead distance was set to 1 m for the pure pursuit controller. The cross-track proportion and yaw error proportion for the way-line controller was set to 1.5 and 0.05, respectively. The navigation tests were recorded using *rosbag*, and the yaw heading error and cross-track error were calculated to evaluate the navigation accuracy. The maximum translational speed for the robot was set to 0.3 m/s. The goal tolerance was set to 0.1 m, and the yaw tolerance was set to 0.1 rad.

6.4.3 High-throughput phenotyping tests

We evaluated the robot's performance in field data collection for high-throughput phenotyping through two tests. The first test was collecting videos for cotton seedling counting using the MARS mini configuration 1 robot. The second test used MARS X to collect hyperspectral images for a cotton field.

Cotton seedling counting

The MARS mini configuration 1 robot has a high clearance that can be used to enables the robot to traverse over the crop rows, making it suitable for collecting data on small crops, so we used it to collect videos for cotton seedling counting. Two color cameras were mounted on the robot, with one camera on the top front looking at the top of the cotton and one camera

on the bottom side looking at the side of the cotton. An RTK-GNSS (Smart6L, Novatel, Canada) and IMU (VN100, VectorNav, TX, USA) were used for robot localization.

The data collection was performed 14 days after planting at the same cotton field used by the navigation test. One crop row (8 plots) was selected, and the robot was set to autonomously traverse over the cotton row using the pure pursuit controller at a maximum speed of 0.5 m/s. First, the collected video was split for each plot and then analyzed using the DeepSeedling pipeline [187]. Part of the video was annotated to retrain the FasterRCNN used by the DeepSeedling to adapt the network to the current data. In total, 610 frames were annotated, 400 frames were used for training, and 210 frames for validation. The retrained model was used to detect cotton for the rest of the data. The number of cotton plants was manually counted in the field as ground truth.

Hyperspectral imaging

The hyperspectral camera is a heavy sensor (about 10 kg) that can be carried by the MARS X. A push-broom hyperspectral camera (MSV-500, Middleton Spectral Vision, WI, USA) was mounted on the MARS X to image the cotton plots in the same cotton field that was used as the navigation test (Figure 6.16). A spectrometer (USB2000+, Ocean Optics, FL, USA) was mounted on top of the robot to measure the spectrum of the downwelling sunlight. Dual RTK-GNSSs were used for the localization of the robot. The robot was set to navigate over a crop row autonomously using the way-line controller at a speed of 0.3 m/s. A separate laptop was used to record the hyperspectral image.

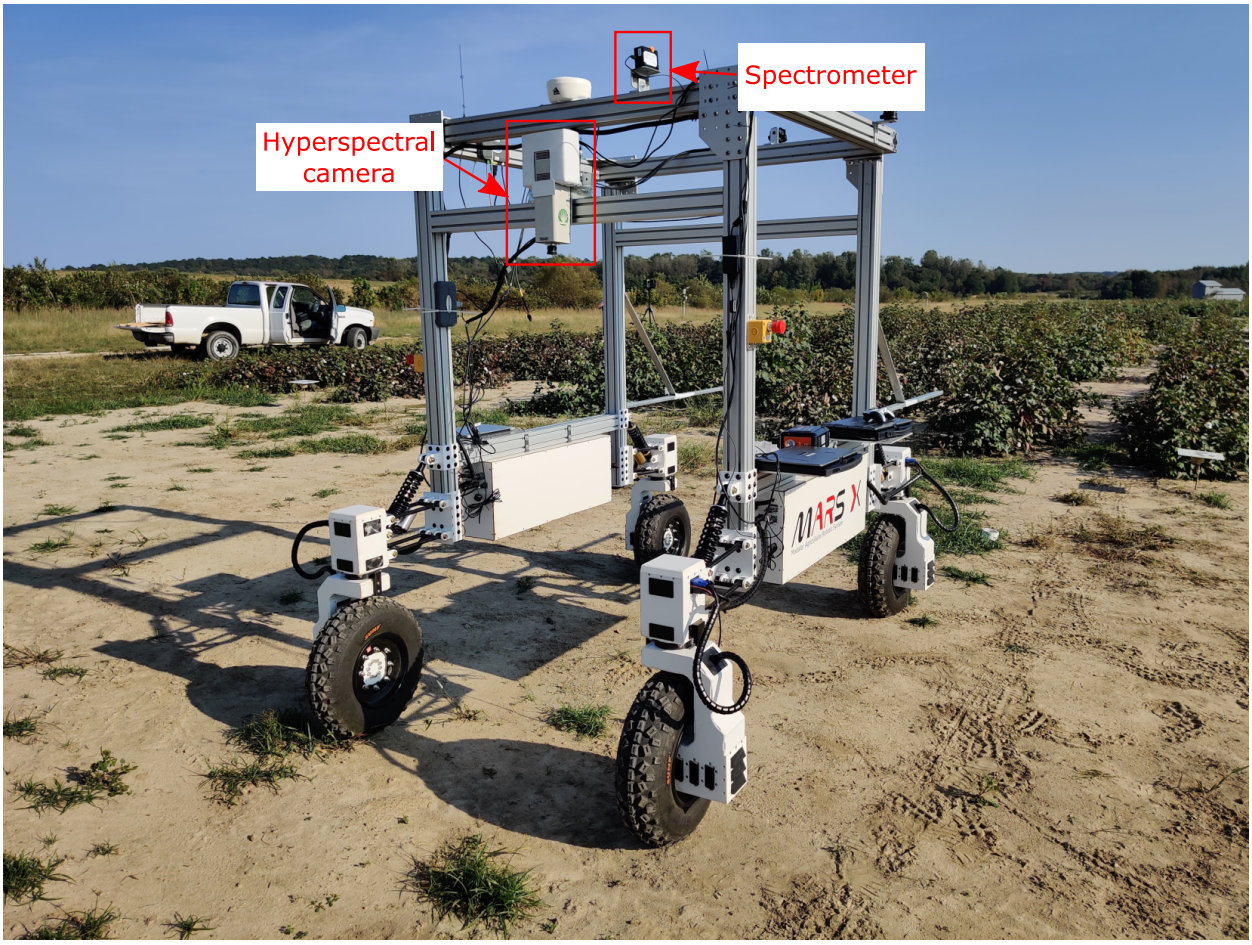

Figure 6.16: MARS X with the hyperspectral camera.

6.5 Results

6.5.1 Mobility test

Thanks to the suspension, both robots were able to pass all the obstacles without losing contact with the ground, which can prevent losing traction (Figure 6.17). During the test, we noticed that when one wheel is on the obstacles, the other three wheels' suspensions were also compressed, making the robot tilt toward one side. The MARS X was able to climb the

incline thanks to the drive motor's high torque. However, because MARS X has a relatively high center of gravity (0.423 m high), there is a potential risk that the robot could flip over on a very steep incline. We have provided all the test videos as supplementary materials.

Figure 6.17: Example video frames of the obstacle test.

6.5.2 Navigation test

The robot took 36 minutes to finish the navigation at a speed of 0.3 m/s. Both the path-following algorithms can track the path with high accuracies (Figure 6.18). The mean cross-track error is 0.035 m with a standard deviation of 0.048 m for the pure pursuit controller. The way-line controller had a smaller cross-track error with a mean error of 0.027 m and a standard deviation of 0.036 m. Over the path of 537 m, the robot can follow the path within the error of 0.05 m for most of the time and only 15% of the path has errors larger than 0.05 m for the pure pursuit controller. The way-line controller has an error larger than 0.05 m on 18% of the path. The large tracking error usually happened when the robot started to track a new line because the line following algorithms took time to converge. Both path-following

algorithms had small heading errors (Figure 6.19). After the robot turns to the desired heading, it can maintain the heading with an error between $-3°$ and $2°$ for the pure pursuit controller and between $-2°$ and $2°$ for the way-line controller.

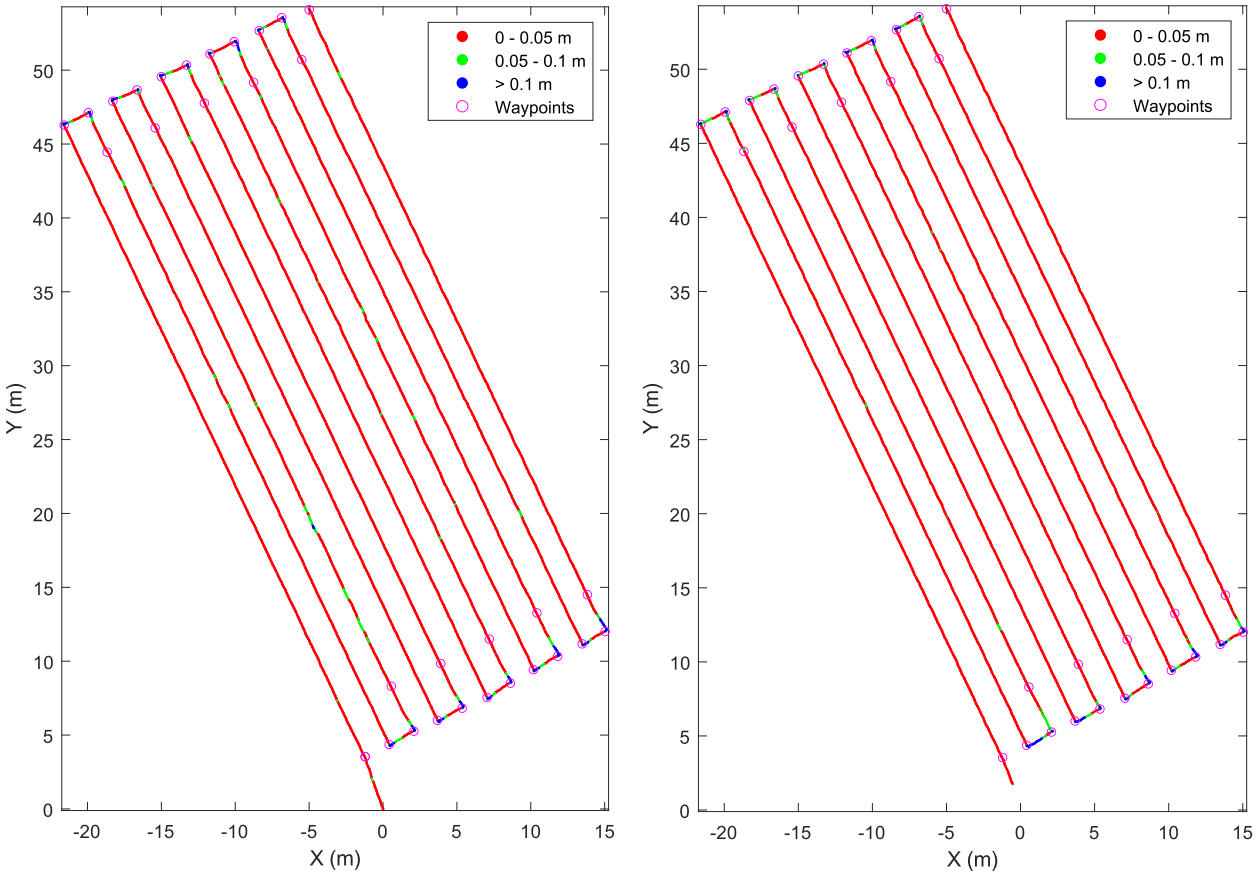

Figure 6.18: Cross-track error of the navigation test for the pure pursuit controller (left) and way-line controller (right).

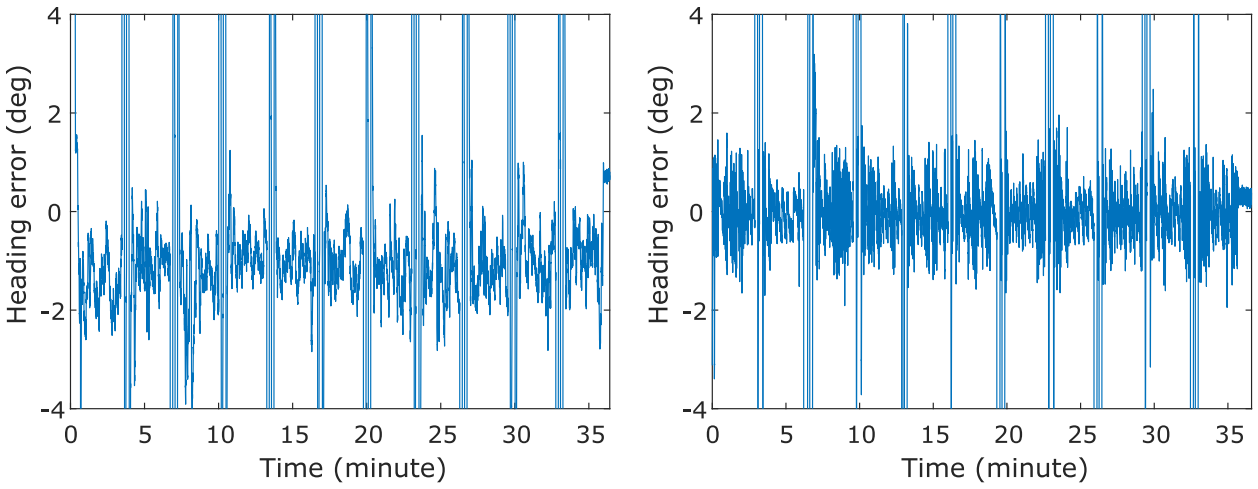

Figure 6.19: Heading error of the navigation test for the pure pursuit controller (left) and way-line controller (right).

6.5.3 Cotton seedling counting

With the pure pursuit controller, the robot was able to maintain a straight line with a deviation from −0.13 m to 0.19 m (Figure 6.20). The robot's deviation could have been caused by a wheel being made slippery by the ground terrain. Because the GNSS was mounted on top of the robot with a certain height from the ground, the robot's tilt will also change the GNSS's position, which affects the localization of the robot. The low weight of the MARS mini robot made it more susceptible to deviations resulting from unevenness of the ground. Therefore, it is recommended to use the robot on relatively even ground.

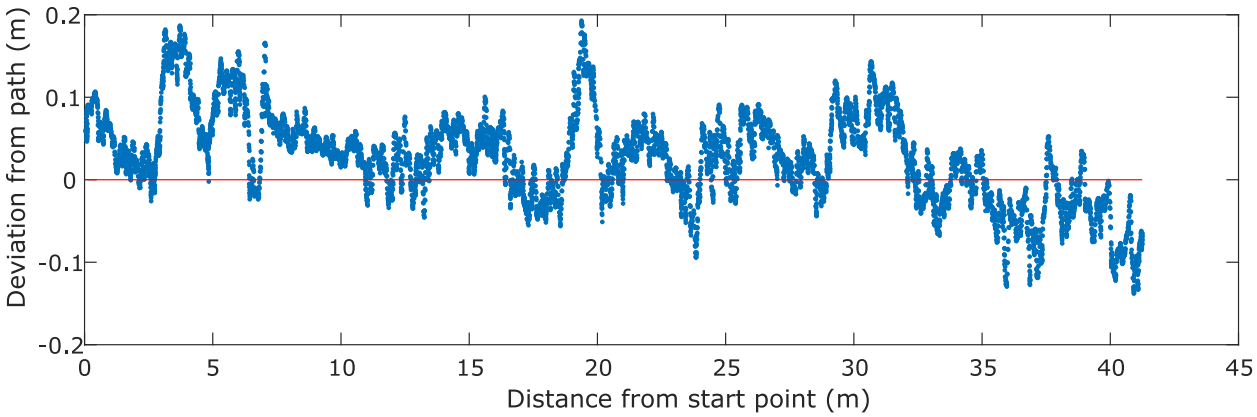

Figure 6.20: Error of the line tracking of the pure pursuit controller with MARS mini configuration 1 robot.

The MARS mini robot is able to collect stable videos to count cotton seedlings and a high counting accuracy with an error ranging from -2 to 6 was achieved (Figure 6.21). The network can accurately detect the cotton plants when the plants are spaced sparsely (Figure 6.22A). However, because the video was taken 14 days after planting, some plants already developed true leaves, which makes it difficult for the FasterRCNN to detect all the overlapping cotton seedlings (Figure 6.22B). In some cases, two overlapped cotton plants were detected as one plant (Figure 6.22), while in other cases, the network detected false plants whose leaves were from two plants. Therefore, it is better to collect data before the plants develop true leaves.

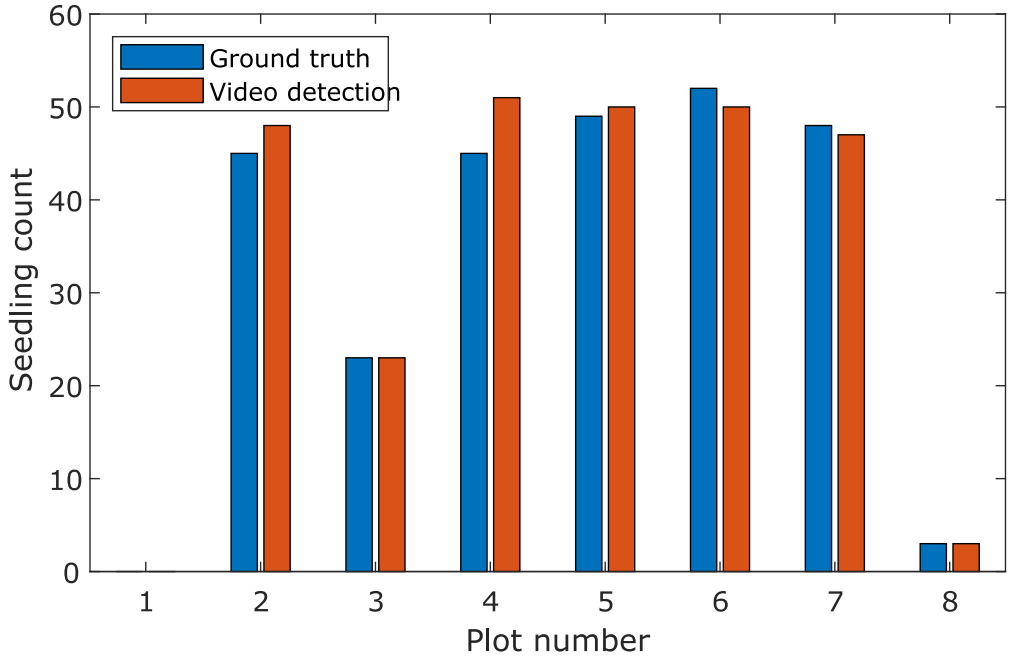

Figure 6.21: Results of the cotton seedling counting.

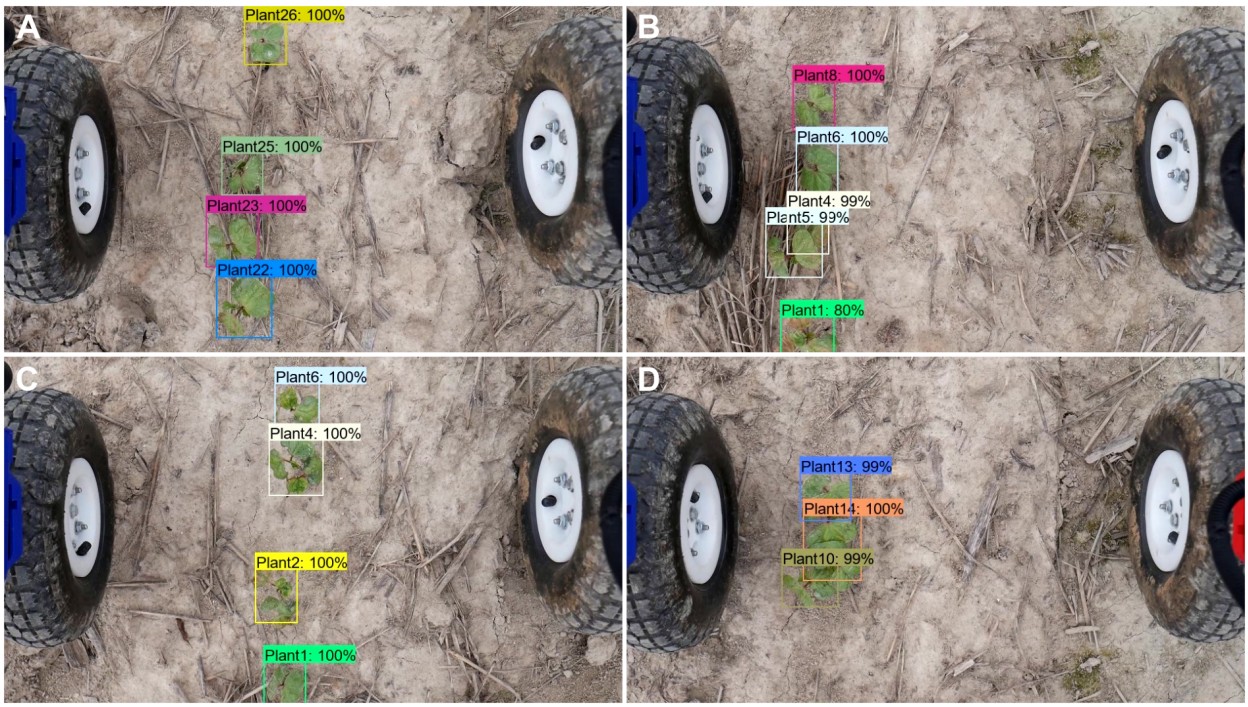

Figure 6.22: Example frame of the cotton seedling detection. A) All the cotton plants were detected. B) Two overlapped plants were detected as two plants (Plant 4 and plant 5). C) Two overlapped plants were detected as one plant (Plant 4). D) False detection of a cotton plant (Plant 14) whose leaves were from two plants.

6.5.4 Hyperspectral imaging

Because the robot could maintain a straight line at a constant speed, the scan lines of the hyperspectral camera can be directly stitched without clear geometric distortion (Figure 6.23). When the robot deviated from the navigation line or ran over a bump, the image can be distorted by the deviation or the tilt of the camera caused by the bump (Figure 6.23B). It is possible to correct the distortion using the location of the robot to calculate the locations of the scan lines and stitch the scans according to the locations. One limitation of the current design is that the shadow of the robot and disturbance of the plants caused by the robot can affect hyperspectral imaging. This limitation can be addressed by extending the hyperspectral camera outside of the robot frame so the camera's field view will not cover the shadow.

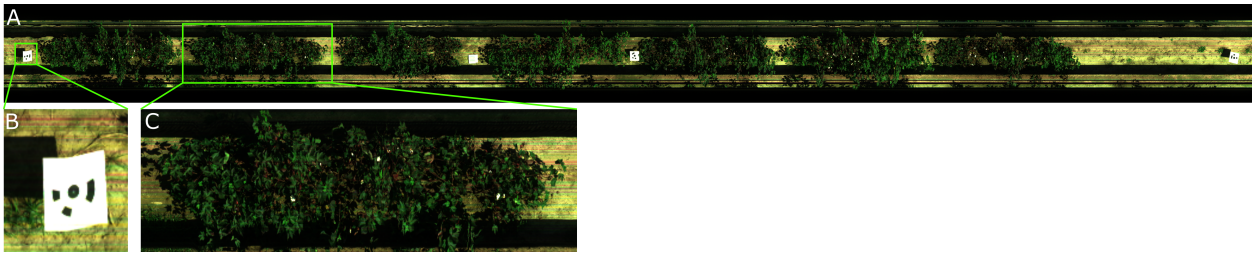

Figure 6.23: Hyperspectral data collected by MARS X. A) Pseudo color image of the cotton row. B) Zoom-in image of the ground target. The target is a square but appeared distorted in the image. C) Zoom-in image of one plot.

6.6 Discussion

Agricultural robots have shown promising potential to solve labor shortages in the agricultural industry. However, developing a robust robotic system that can reliably operate in

unstructured agricultural environments has been challenging and costly. Although some progress has been made in recent years, agricultural robots are still not robust enough and cost-effective to replace traditional agricultural machinery, which limits robots' wide use. In this study, we proposed a modular robot design to lower the costs in both aspects. First, we used low-cost components and manufacturing methods, such as 3D printing and extrusion aluminum. Second, the robot modules can be reused for different robot configurations, reducing the total cost of performing different agricultural tasks using several task-based robots. The modality also reduces the maintenance cost. The MARS mini is designed primarily as a low-cost way to provide a robotic platform for researchers and growers with small farms. We have demonstrated the application of MARS mini for cotton seedling detection. The MARS mini is ideal for phenotyping research because its light weight will not create soil compaction. The downside of its light weight is that the robot is not stable on a bumpy ground, which means that the data quality may suffer when the robot vibrates. Therefore, the MARS mini should be used on a relatively flat field to get quality data. The MARS X, on the contrary, is designed for heavy-duty tasks. Its weight is large enough to keep the robot stable on the bumpy ground but is not too heavy to create soil compaction. Both MARS mini and MARS X have a high payload that can carry heavy sensors (such as the hyperspectral camera) and equipment, but the payload of the MARS X is much higher (up to several hundreds of kilograms).

Several aspects can be improved for the current design. First, the MARS mini's mechanical strength can be increased by improving the 3D printed parts, including adding stiffeners, changing the printing settings, and using stronger material such as acrylonitrile butadiene

styrene (ABS). Because of the strength of plastic and the natural degradation in harsh environments, the payload and lifespan of MARS mini robots are limited. Therefore, a metal version using aluminum may be worth developing for the MARS mini in the future. Second, the kinematics of the robot can be improved further. Currently, the kinematics assumes that the wheels' direction and speed can be changed instantly to stay synchronized, but in reality, they are not always synchronized because the wheels vary in the time they take to change to the desired direction and speed. The asynchronization of the wheels results in an invalid ICR, which creates torques for the robot frame, especially for the four-wheel steering four-wheel driving robot. Based on our test, the asynchrnonization is negligible at low speeds (0 to 0.3 m/s) but becomes more apparent with increased speed. A possible solution for this issue could be to lower the speed temporarily when an invalid ICR is detected and restore the speed after all the wheels are synchronized.

As a robot platform, the MARS mini and MARS X can be adjusted to perform different tasks. The extrusion-based robot frame also makes it easy to mount other modules on the robot, as demonstrated in the high-throughput phenotyping tests. The navigation test demonstrated that the robot could follow the preset path with considerable accuracy, enabling the robot to perform tasks automatically without human intervention and without damaging the crops.

6.7 Conclusion

This paper presented the design concept and implementation details of MARS. Several hardware modules and software modules were proposed and designed. Based on the design concept, two types of robot, MARS mini and MARS X, were implemented. The MARS mini is a low-cost, and lightweight robot made of low-cost components and 3D printed parts. Several robot configurations of the MARS mini were designed. MARS X is a heavy-duty robot that can carry a large payload. The performance test showed that MARS X could pass obstacles and climb the incline. The software modules of MARS were implemented using ROS, and we designed a robot control framework and object detection framework. We implemented two path-following algorithms, and the field test showed high path-following accuracy. We have demonstrated the usage of the robots for plant phenotyping in two field data collection tests. For future work, we will implement more hardware modules to expand the current functionality of the MARS mini and MARS X. We will implement more navigation algorithms, including visual servoing and LiDAR-based navigation methods.

Chapter 7

Development of Next Generation Berry Impact Recording Device

7.1 Introduction

The Blueberry industry is an important contributor to the US agricultural economy. The US is the largest blueberry producer worldwide, with a production of 680 million pounds in 2019 (USDA). The US also has the largest blueberry market, estimated to be $758 million in 2019. Fresh blueberries have a much higher price than processed blueberries. In 2019, the market price for cultivated fresh blueberries was $2.03 per pound, while the price for processed blueberries was $0.50 per pound (USDA). Because blueberries are prone to bruise damage, fresh blueberries need to be handled carefully during harvesting, packing,

and transportation to ensure good quality. Bruised fruit is not suitable for the fresh market and results in great economic losses for the farmers. The fruit can be bruised by mechanical impact during harvesting, packing, and transportation. It is estimated that 78% of the mechanically harvested blueberries are severely bruised, so most of the blueberries destined for the fresh market are hand-harvested [188]. However, because machine harvesting cost (estimated to be about $0.12/lb) is much lower than hand harvesting ($0.50/lb to $0.70/lb), more growers are harvesting blueberries using mechanical harvesters, especially in light of the labor shortage in recent years [189].

Although severely damaged blueberries can be sorted out by packing lines, blueberries with internal bruises cannot be sorted because internal bruises are not visible from the outside and do not appear immediately after impact, but rather develop over time. The packing line itself also can cause bruises since it has multiple transition points with vertical drops that can cause impact when fruits go through. Therefore, to improve the machine harvester and packing lines and reduce bruising damage caused by mechanical impact, it is important to know the quantity and location of the impacts during the harvesting and packing process.

An instrumented sphere, also known as "pseudo fruit," can be used to measure the mechanical impacts encountered by the fruit. Various instrumented spheres have been developed in the past [190, 191, 192, 193, 194, 195]. For example, the well-known IS100 is an instrumented sphere that uses a tri-axial accelerometer to measure dynamic impact. The IS100 was later commercialized as the Impact Recording Device (IRD) (Techmark Inc., Lansing, MI, USA). Another example is the PMS-60, which can record both static load and dynamic impact using pressure sensors [193]. A wireless instrumented sphere that uses

both accelerometer and load cell was developed to measure the impact and compression in real-time [194].

The existing instrumented spheres are designed for large fruits and vegetables, such as apples and potatos, so they are not suitable for smaller fruits and vegetables like blueberries. Therefore, the Berry Impact Recording Device (BIRD) and its second generation (BIRD II) were developed [196, 197]. The first generation BIRD (BIRD I) is a 1-inch sphere with a weight of 14 g and sensing range of $\pm 500\,\boldsymbol{g}$ in each orthogonal axis. The BIRD II reduced the diameter to 21 mm and weight to 6.9 g. However, the sensing range of BIRD II is only $\pm 200\,\boldsymbol{g}$ in each orthogonal axis. Both BIRD I and BIRD II are not waterproof, however, and need to download data through a wired connection. In this study, we sought to design the next generation Berry Impact Recording Device (BIRD Next) to improve the performance of the sensor. Specific objectives include: 1) increasing the sensing range and recording frequency, 2) making the sensor waterproof by using wireless communication and wireless charging, and 3) developing a mobile app to interface with the sensor. The BIRD Next can be used for fruits such as cherries whose processing involves water.

7.2 Hardware Design

One key design challenge of BIRD Next is to keep the sensor size comparable with BIRD I and BIRD II even the BIRD Next has a more complicated circuit than BIRD I and BIRD II sensors. Therefore, to address the challenge, we split the circuit of BIRD next into six circuit boards, and formed the six boards into a cube with the battery inside the cube (Figure

7.1). With such a setup, the total usable PCB area to hold electronic components increased five times over a single circuit board while maintaining the same sensor size. Each circuit board implemented specific functions. Three single-axial accelerometers were placed on three orthogonal boards to measure the acceleration of X, Y, and Z axes. Two circuit boards were used as the main controller board, which has a wireless microcontroller that implements wireless communication. One circuit board was used to implement wireless charging and provide regulated power for other boards. The size of circuit boards is $12.6 \times 12.6\,\mathrm{mm}^2$ and six circuit boards form a $12.6 \times 12.6 \times 12.6\,\mathrm{mm}^3$ cube.

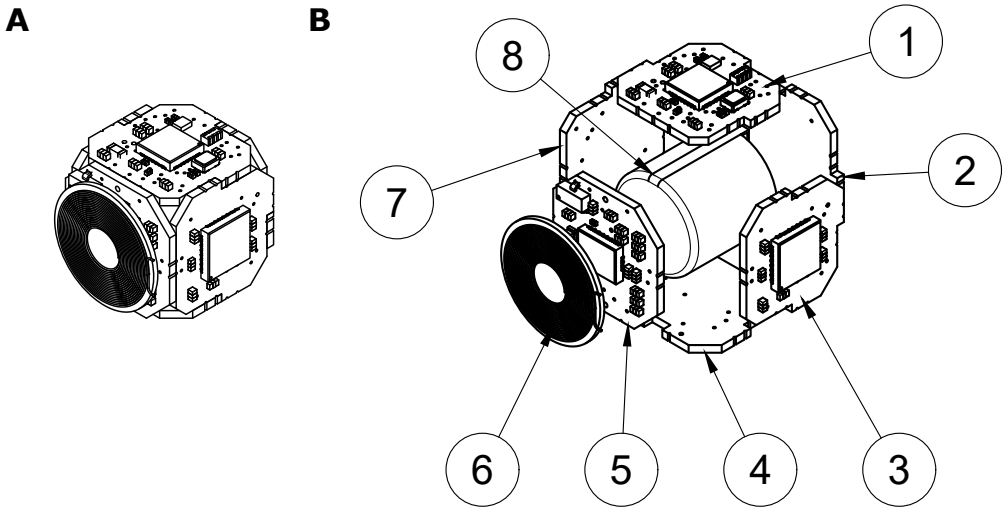

Figure 7.1: Normal view (A) and explored view (B) of the assembly of the BIRD Next sensor. 1) Main controller board. 2) Antenna board. 3) X axial accelerometer board. 4) Z axial accelerometer board. 5) Wireless charging board. 6) Charging coil. 7) Y axial accelerometer board. 8) Lithium-ion polymer battery.

7.2.1 Accelerometer board

The accelerometer board consists of a single-axial accelerometer (ADXL1004, Analog Devices, MA, USA) and a low pass filter (LPF) with a cutoff frequency of 5 kHz to suppress the out-of-band noise and signal as suggested by the manufacture (Figure 7.2). The accelerometer can

measure acceleration up to 500 $\boldsymbol{g}$, which can measure up to 866 $\boldsymbol{g}$ with three axes combined. The current assumption is 1.0 mA at normal operation and 255 µA at standby mode.

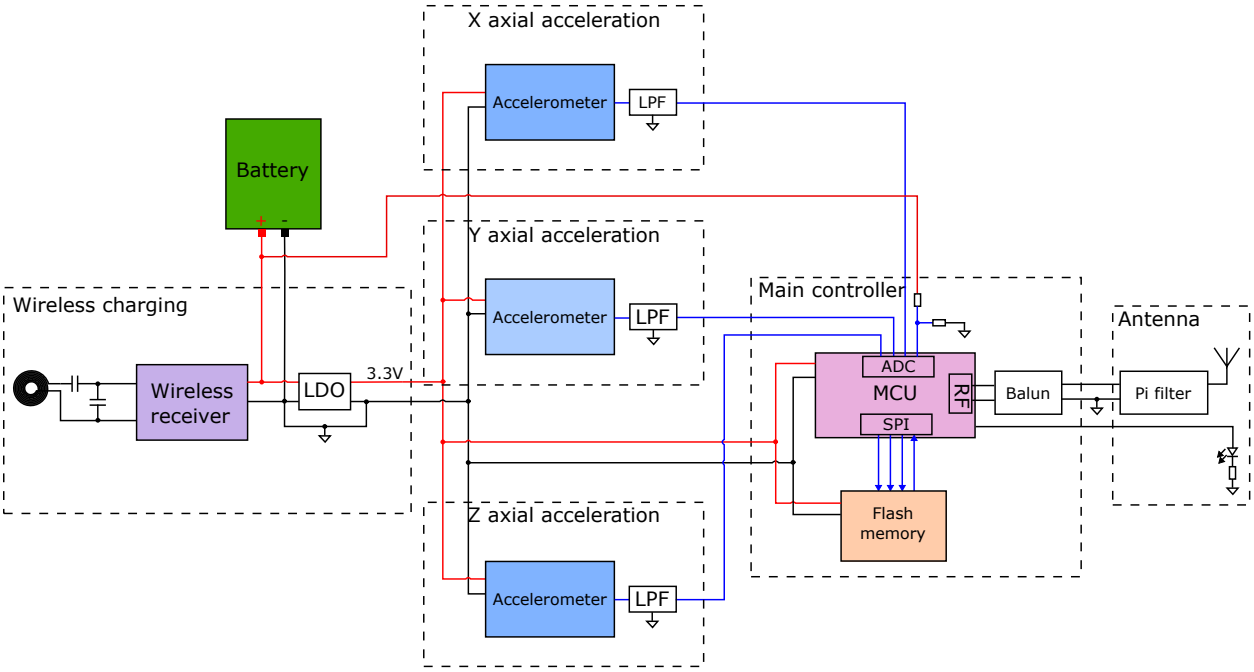

Figure 7.2: Circuit schematic.

7.2.2 Main controller board

The main controller for the sensor was a wireless microcontroller (CC2640R2F, Texas Instruments, TX, USA) that implemented Bluetooth 5.1 Low Energy (BLE). Bluetooth was chosen for its low power and wide usage in mobile devices and PCs, which makes it convenient for users to interface with the sensor using mobile devices. Bluetooth 5.1 has lower current consumption and a longer transfer range than previous Bluetooth versions. The microcontroller had an Advanced RISC Machine (ARM) Cortex-M3 core as the main CPU, a Radio Frequency (RF) core to interface the analog RF and handles Bluetooth low energy protocol, and a Sensor Controller (SC) that can independently control the peripherals such

as General-Purpose Input/Output (GPIO), Analog-to-Digital Converter (ADC) and Serial Peripheral Interface (SPI). The SC can run autonomously in the background to offload the main CPU and reduce power consumption, and is used to run the ADC conversions for the sensor. An ultra-low-power flash memory (MX25R1635F, Macronix International, Taiwan) with 16 Mbit capacity was chosen to store sensor measurements. The main controller used SPI to interface the flash memory.

Differential input and internal bias were used as the RF front-end configuration for CC2640R2F. The RF front-end circuitry consisted of a balun, a Pi-filter, and a 2.4 GHz chip antenna. Because of the size limitations of the circuit board, the Pi-filter and antenna were placed on another circuit board. A chip antenna was chosen for its small size. The balun can convert the differential signal to a single-ended signal and match the input impedance of CC2640R2F to 50 Ω. The main purpose of the Pi-filter was to fine-tune the impedance of the antenna to match the input impedance of the balun.

7.2.3 Wireless charging board

Wireless charging technology has been used widely in applications in which wired power transfer is not desirable, such as electric toothbrushes. There are two types of wireless charging technologies: inductive charging and resonant charging. The inductive charging, implied by its name, uses electromagnetic induction to transfer energy. An alternating current is run through an induction coil in the charging station and generates an alternating magnetic field, which induces an alternating current in the secondary coil. It is then converted to direct current through a rectifier to charge the battery. The inductive charging requires

two coils closely coupled as the transfer efficiency reduces significantly when the distance between the coils increases. Therefore, the charging distance is limited to a few centimeters. The resonant charging transfers power through two coils (a transmitter and a receiver) with identical resonant frequencies. When the transmitter operates at the resonant frequency, the receiver resonates and thus receives power from the transmitter. Resonant charging is less restricted by the spatial arrangement of the two coils, which makes the charging distance is higher than inductive charging but with poor efficiency. In this project, inductive charging was chosen because this is a matured technology and efficiency is not an important factor for our application. We chose a wireless receiver chip (BQ51050B, Texas Instruments, TX) to implement the wireless charging. The BQ51050B complies with the Qi standard and can achieve wireless charging with minimal external components. The charging current was programmed to 100 µA. BQ51050B has a built-in lithium-ion battery charger to manage the charging process.

A cylindrical Lithium-ion polymer battery was chosen. The battery is 10 mm in diameter and 11 mm in height and can fit inside the sensor cube. The voltage of the battery was 3.7 V, and the capacity was 60 mA h. The battery is connected directly to BQ51050B. A Low-dropout (LDO) regulator (MCP1700, Microchip, AZ, USA) was used to convert the battery voltage to 3.3 V to power the other circuit boards.

7.2.4 Sensor assembly

After the individual circuit board was assembled, all six circuits were combined to make a cube. First, the three accelerometer boards, the main controller board, and the antenna

board were first assembled (Figure 7.3). The tabs on each circuit board allow the boards to align perpendicular to each other, and the boards were connected through wires at the edge (Figure 7.4). Second, the wireless circuit board was connected with the battery first and then connected with other boards to form the sensor cube. Third, the sensor cube was encapsulated into a sphere with a diameter of 20 mm using transparent epoxy (Figure 7.5). This severed as the inner layer sphere to protect the circuit boards and to make the sensor weight evenly distributed. Last, an outer sphere was made to encapsulate the inner sphere. The outer sphere is mainly used to mimic the surface properties of the fruit. Therefore, the material and size of the outer sphere could be adjusted based on the target fruit. Following previous research, we used silicon rubber (Mold Max 27T, Smooth-On, Inc., PA, USA) for the outer sphere to mimic the surface properties of blueberry and make the diameter to 25.4 mm (1 inch).

Figure 7.3: Sensor assembling flowchart. 1) Assemble the accelerometer boards and the controller boards. 2) Assemble the wireless charging board with battery. 3) Connect the wireless charging board with other boards to form the sensor cube. 4) Encapsulate the sensor cube into the inner sphere using transparent epoxy. 5) Encapsulated the inner sphere into the outer sphere using translucent silicon rubber.

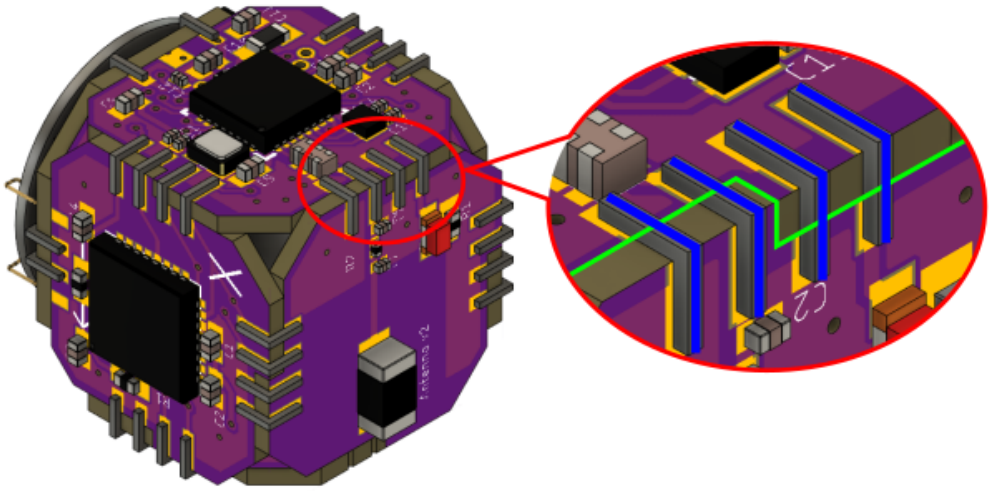

Figure 7.4: Illustrations of the connection between circuit boards (highlighted by blue lines) and the tab to align the circuit board (highlighted by green lines).

Figure 7.5: Inner (left) and outer sphere (right) of the BIRD Next sensor.

7.3 Software Design

7.3.1 Sensor program

The sensor program was designed using a finite-state machine (Figure 7.6). There are three states (record, idle, and upload) and each state implements specific functions. The program

stays in the idle state when the sensor is not recording or uploading data. When the user starts the recording, the program enters the record state to take measurements of the acceleration and save them to the flash memory. The program changes from record state to idle when the user stops recording or the flash memory is full. When the user requests to download data from the sensor to mobile devices or PCs, the program enters the upload state to send data. The program goes back to the idle state when the user stops downloading or all the data were sent.

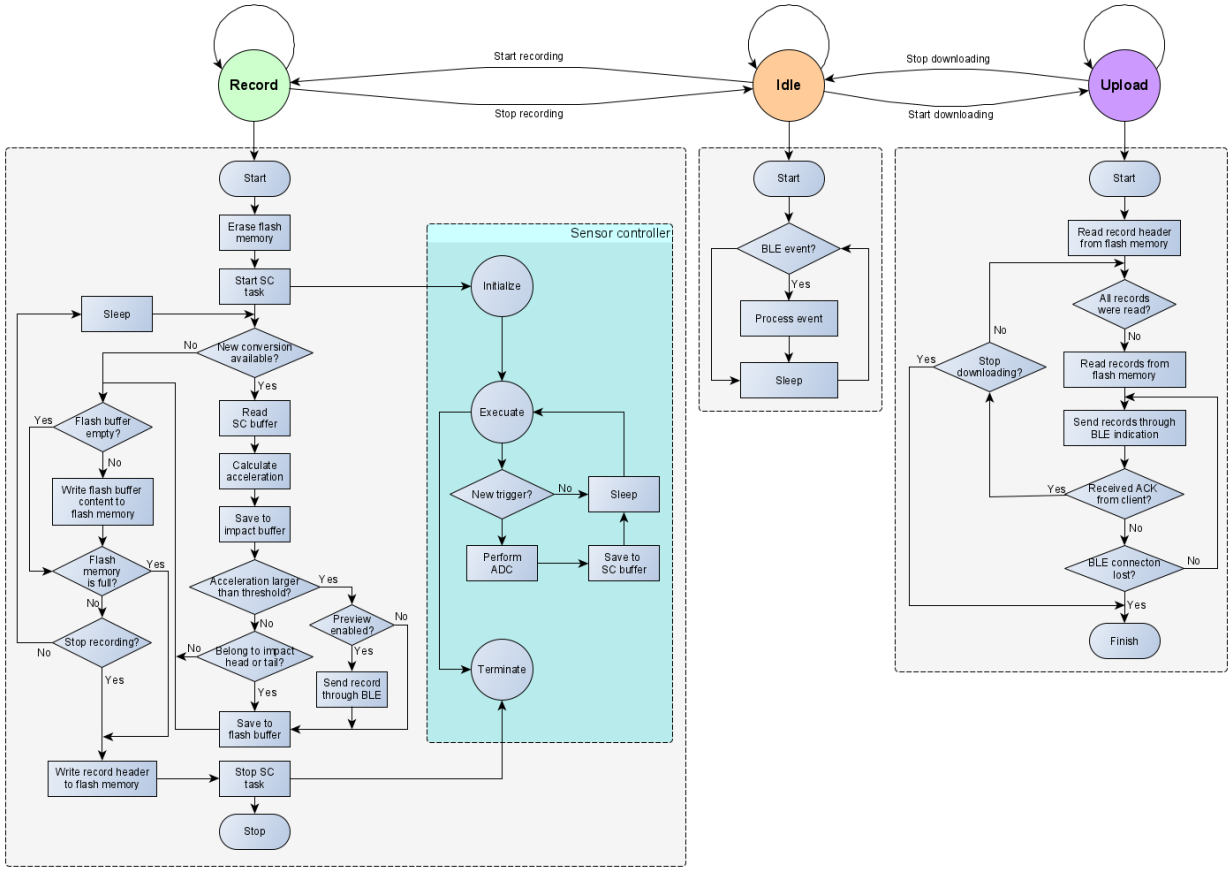

Figure 7.6: Finite-state machine of the sensor program. The circle indicates the state.

7.3.2 Bluetooth communication

The CC2640R2F has an independent RF core to process Bluetooth protocols, and Texas Instrument provided Application Programming Interface (API) to program with the RF core. We used those APIs to implement the Bluetooth communication between the sensor and other devices.

Bluetooth 5.1 uses the Generic Attribute Profile (GATT) in the application layer for data communication between two connected devices. Data are passed and stored as characteristics that are stored onto the memory of the Bluetooth device. When the sensor is connected with a mobile device, the sensor is served as a GATT server that contains characteristics that can be read or written by the GATT client, i.e., mobile device. The GATT clients (mobile devices) initiate GATT commands and requests, and the GATT server (BIRD Next sensor) receives the commands and requests and return responses. The characteristics are the data values transferred between the server and the client. A collection of related characteristics is called a service. In the sensor program, several characteristics are used for sensor configuration and data downloading, and those characteristics are combined as the sensor service (Table 7.1). For example, the SensorState characteristic is used to configure the state of the program, where the user can start or stop the data recording by changing the value of this characteristic. The FlashData characteristic is used for uploading the recorded data.

A characteristic has certain properties that indicate when certain operations are permissible. A GATT server can push messages to the client actively in the forms of GATT notification (requires no acknowledgment from the client) or GATT indication (requires an

acknowledgment from the client). This mechanism can greatly reduce power usage for the server if only newly-generated data is pushed to the client compared to when the client constantly polls for new data. In the sensor program, we used the PreviewData characteristic to push the real-time acceleration data to the client in the form of GATT notifications. The GATT indication was used for uploading the recorded data to the client using the FlashData characteristic to ensure the data integrity. The SensorState characteristic also has a notification property so the sensor can inform the client once the program state is changed.

Table 7.1: Characteristics of the sensor service.

Characteristic	**Properties**	**Description**
PreviewEnabler	Read | Write	Use to enable/disable preview
PreviewData	Notification	Used to send preview data
SensorState	Read | Write | Notification	Used to change and retrieve sensor state
TimeStamp	Write	Used to synchronize timestamp of the recording with mobile App
FlashData	Indication	Used to send data in the flash memory
SensorRange	Read	Used to read sensor range
BatteryVoltage	Read	Used to read sensor battery voltage
ImpactThreshold	Read | Write	Used to read and set the impact threshold
RecordingFrequency	Read | Write	Used to read and set recording frequency

7.3.3 Data recording

An impact occurs when the sensor collides with other objects. The BIRD Next sensor captured the impacts by constantly measuring the acceleration at a fixed frequency and detecting large accelerations that exceeded a threshold. An impact curve (acceleration over time) is usually a bell curve and can last several million seconds, depending on the properties of the collision surface. Therefore, the higher the frequency of the sensor measuring the acceleration, the more details of the impact curve are captured. The impact threshold and recording frequency can be configured by the user using the ImpactThreshold and RecordingFrequency characteristic.

Once a impact was detected, each data point of the impact was saved to the flash memory along with its timestamp. The relative timestamp of the acceleration was stored by the sequence number of the measurement. The absolute timestamp of the first measurement was recorded and used to calculate the absolute timestamp of the rest measurements. We used Unix timestamp (seconds after 00:00:00 UTC, January 1970) to record the absolute timestamp for the first measurement. The timestamp of the first measurement is synchronized with mobile devices through the TimeStamp characteristic. The flash memory was divided into two sections. The first 256 bytes were used to store the record header that includes the information of the recording settings (such as recording frequency and impact threshold) and the number of records. The remaining space was used to store the measurements.

When the user starts the recording, the sensor program changes to the record state. The data recording uses ADC to convert the output of the accelerometer to a digital value and SPI to save data to the flash memory. The ADC conversion was done in the sensor controller so it can run in the background. The sensor controller has a task-based programming flowchart. Each task has three states – initialization, execution, and termination. When the sensor controller starts to run a task, it first runs the initialization code, then runs the execution code repeatedly, and finally runs the termination code after the task is terminated. In the sensor program, we set up an ADC conversion task and used a timer to generate triggers at the desired frequency to trigger the ADC conversions. The conversion results were saved to a buffer memory and passed to the main program.

Because the writing speed of the flash memory is significantly slower than the processing speed of the main program and the sensor controller, three buffers were used to cache the

data. First, A SC buffer was used to store the conversion results in the sensor controller. Second, an impact buffer was used to store the acceleration data calculated from the ADC conversions. Third, a flash buffer was used to store the acceleration data that needed to be stored on the flash memory. With such arrangement, the main program did not need to wait for the flash memory to finish writing. The sensor program can achieve a maximum recording frequency of 10 kHz.

Since the flash memory needs to be erased before writing, the entire flash memory is erased before the recording. Then, the main program starts the sensor controller task to start the ADC conversion. When a new conversion is complete, the sensor controller generated an interrupt to inform the main program, and the main program calculated the acceleration from the digital value. Similar to BIRD I and BIRD II, accelerations higher than the impact threshold were recorded, as well as the leaders are trailer of an impact [196]. If the flash buffer is not empty and the flash memory is ready for writing, the content of the flash buffer is written to the flash memory. When the data recording was finished because of termination by the user or the flash memory was full, the recording header was saved to the flash memory.

7.3.4 Data uploading

After the data recording was completed, the user could download data to mobile devices by changing the state of the sensor program to the upload state. In the upload state, the program first reads the record header from the flash memory to find the number of records in the flash memory. The sensor only reads and uploads the records in the flash memory to save uploading time. The data uploading was done through the FlashData characteristic

in the form of GATT indication. Because the Bluetooth limits the length of the packet (controlled by the Maximum Transmission Unit (MTU)) that can be transferred each time, the measurements were split into multiple transfers, and each transfer was re-sent when it did not receive an acknowledgment from the mobile devices. This helped to prevent from losing data during transfer. To maximize the data transfer throughput, the sensor program used the maximum supported MTU, achieving an uploading speed of 8 kB/s.

7.4 Mobile App

An Android App, called Bluebird, was developed to interface with the sensor through Bluetooth (Figure 7.7). The Bluebird App can configure the sensor, start/stop recording, download data, and plot recorded data. The App has two main pages: the sensor page and the record page. The App shows the sensor page by default on start and scans the nearby Bluetooth devices. It lists the Bluetooth devices with their name and signal strength. The user can select the BIRD Next sensor, and the App will show the sensor information and configuration. The user also can configure the recording frequency and impact threshold in the "Sensor Configure" section, as well as start/stop recording and download data. The live sensor data can be enabled/disabled through the "Live Data" toggle button. When the user starts downloading data, a progress bar will show the downloading progress. The user can stop the downloading with the stop button next to the progress bar. The downloaded data will be shown on the record page with the record's time as the default name, which can be renamed by the user. The sensor data were saved as a binary file but can be shared

as a text file to other Apps, such as email. The App can plot the recorded data as the "Acceleration over time" plot and "PeakG vs Velocity Change (VC)" plot. The PeakG of an impact is the maximum acceleration within the impact, and the velocity change is the area under the impact curve. The "PeakG vs Velocity Change (VC)" plot is particularly useful to differentiate the impact of different collision surfaces. The "acceleration over time" plot can be focused on an individual impact curve.

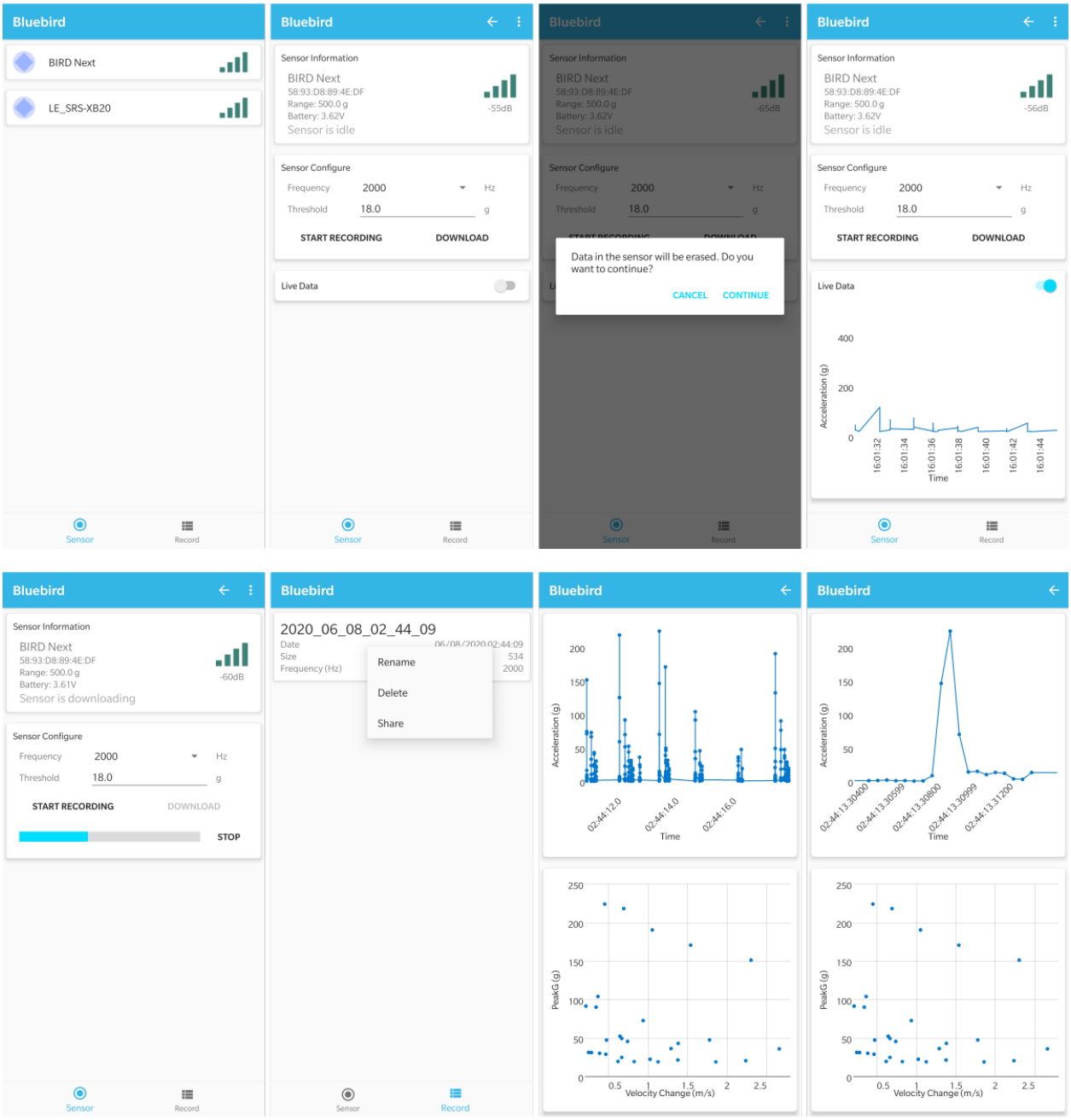

Figure 7.7: User interface of the Bluebird App.

7.5 Calibration and Characterization

7.5.1 Sensor calibration

The sensor was calibrated using a centrifuge (Centrifuge 5430R, Eppendorf, German). To cover the range of the sensor (0 $\boldsymbol{g}$ to 500 $\boldsymbol{g}$), eight rotational speeds at 970, 1370, 1680, 1940, 2170, 2380, 2570, 2750, 2920 and 3070 rotation per minutes (RPM) were used to create eight accelerational values of of 49.80 $\boldsymbol{g}$, 99.34 $\boldsymbol{g}$, 149.39 $\boldsymbol{g}$, 199.20 $\boldsymbol{g}$, 249.24 $\boldsymbol{g}$, 299.81 $\boldsymbol{g}$, 349.60 $\boldsymbol{g}$, 400.27 $\boldsymbol{g}$, 451.29 $\boldsymbol{g}$, and 498.85 $\boldsymbol{g}$, respectively. The centrifuge speed was increased from the lowest speed to the highest and then decreased to the lowest. Each speed was kept for a few seconds to allow the sensor to record enough data points, and 600 data points were selected as one replicate. The increase and decrease of the centrifuge speed created one replicate for 451.29 $\boldsymbol{g}$ and two replicates at other accelerations. The sensor was configured to record data at 100 Hz. The mean value of each replicate was used as the measured acceleration. A linear regression analysis between the reference acceleration and the measured accelerations was performed, and the best-fit line was used to calibrate the sensor. After calibration, the sensor's precision and accuracy were evaluated as the standard deviation of each replicate and the deviation of the measured acceleration from the reference acceleration, respectively.

7.5.2 Surface uniformity test

The impact test using a pendulum was conducted to test the surface uniformity of the BIRD Next, which was expected to be improved than the BIRD II with the two-layer design.

The sensor was fixed at the end of the pendulum so that the collision point can be controlled (Figure 7.8). Six collision points were tested, and each collision point was tested 35 times. The sensor was released at the same height for the six collision points and impacted on an aluminum wall. There were multiple rebounds but only the first impact was considered for the data analysis. The impact threshold and recording frequency were set to 18 g and 10KHz, respectively. ANOVA test was performed on the PeakG of the impacts to test whether there were statistical differences among the impact values.

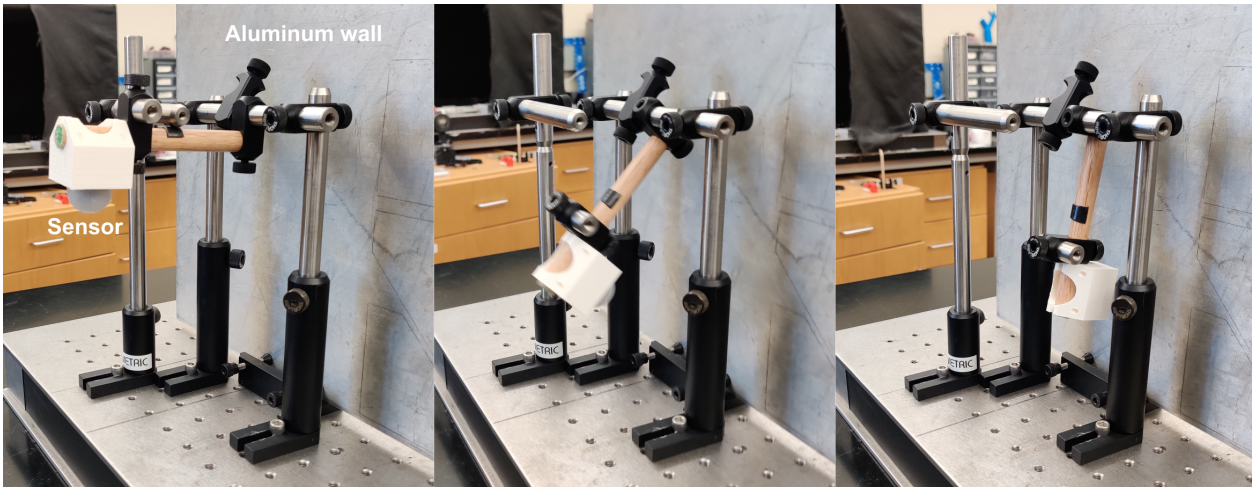

Figure 7.8: Setup of the uniformity test. The sensor was mounted on a pendulum and impacted on a aluminum wall.

7.6 Results and Discussion

7.6.1 Sensor calibration

The measured acceleration showed strong linear relationship with the centrifuge acceleration ($R^2 = 1$) (Figure 7.9). The accuracy of the sensor is within $\pm 0.6\,\boldsymbol{g}$ at different accelerations and the precision ranged from $0.73\,\boldsymbol{g}$ to $1.02\,\boldsymbol{g}$ (Figure 7.10). The largest difference between

different replicate at the same acceleration is 0.85 $\boldsymbol{g}$ at 451.29 $\boldsymbol{g}$ acceleration, indicating a good repeatability.

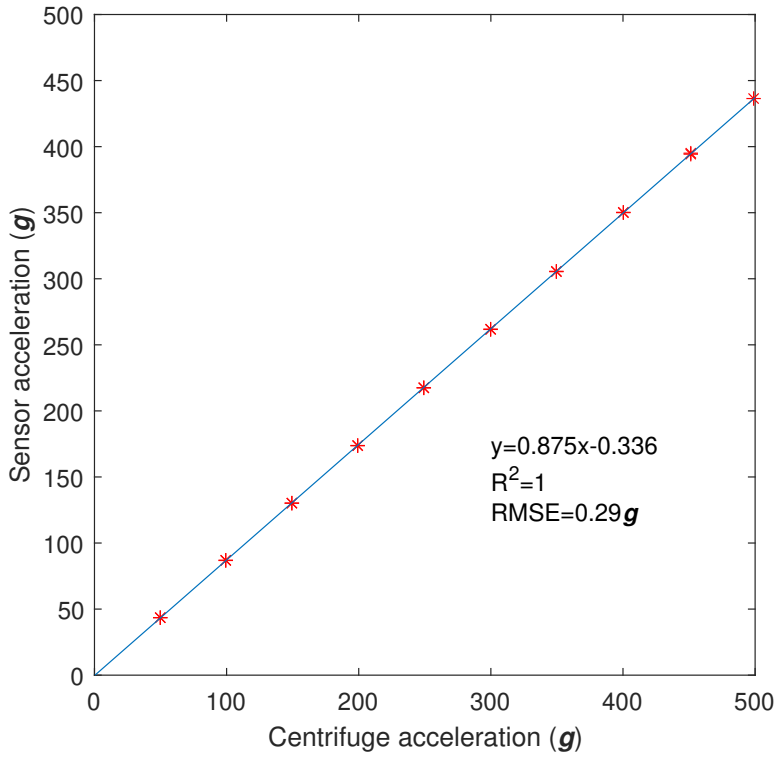

Figure 7.9: Linear regression between the centrifuge acceleration and the sensor acceleration.

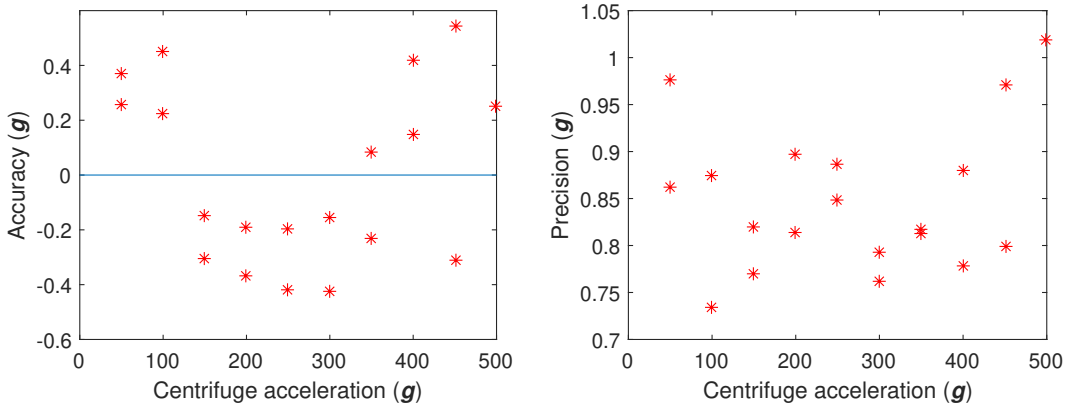

Figure 7.10: Accuracy and precision of the BIRD Next

7.6.2 Surface uniformity test

Figure 7.11 shows the BIRD Next recorded different PeakG values at different collision points, which is similar to the BIRD II sensor. The ANOVA test also showed a significant variance among different collision points. This result contradicted our expectation that the two-layer housing design would improve surface uniformity. A possible reason is that the uneven thickness of the outer layer caused by the fabrication process created different responses to the same impact at different collision points.

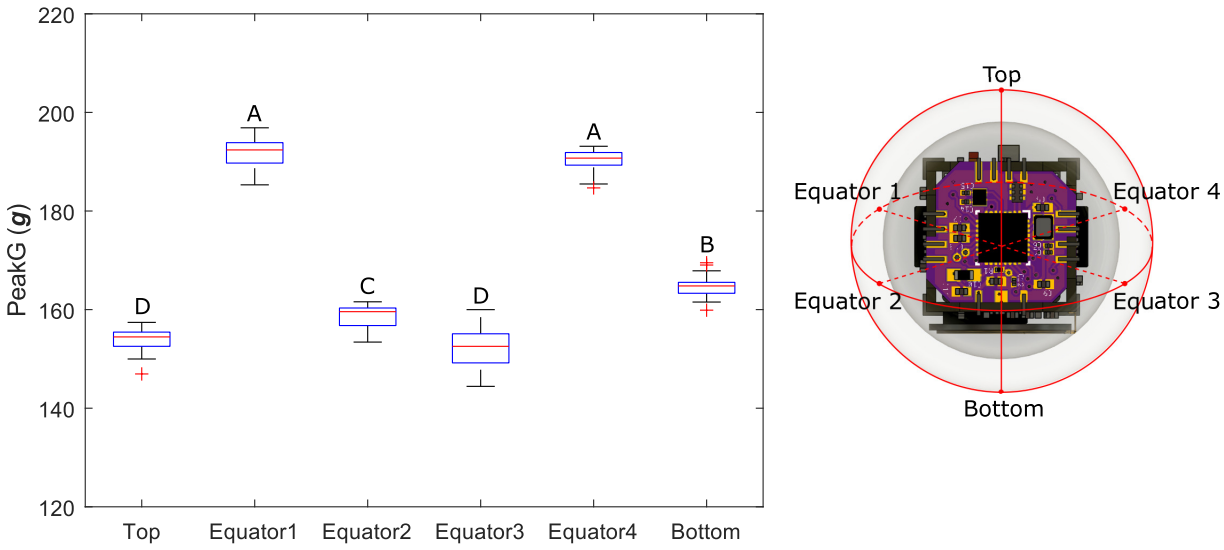

Figure 7.11: Uniformity test result. The same letter showed no significant statistical difference between the collision points. The collision points were shown in the right figure.

7.6.3 Comparison with BIRD II

The BIRD Next offers several improvements over the BIRD II. First, the BIRD Next implemented wireless communication and wireless charging, so no wired connection was

needed. This makes it possible to enclose the entire electronics with epoxy, making it waterproof. Second, the two-layer design makes it much easier to change the sensor's surface property, weight, and size by changing the outer sphere when compared to the BIRD II. Third, the sensing range and frequency of BIRD Next were significantly improved. The sensing range of BIRD next is $\pm 500\,\boldsymbol{g}$ for each axis, equivalent to BIRD I and 2.5 times that of BIRD II ($\pm 200\,\boldsymbol{g}$). The maximum sensing frequency was 10 kHz, which is five times that of BIRD II (2 kHz). Fourth, the accuracy the BIRD Next was increased because of a better ADC in the main controller, but precision decreased. Lastly, the data recording capacity was significantly increased because of the large flash memory used for the BIRD Next.

The improvements offered by BIRD Next cost performance in some aspects. The added functions of BIRD Next made the circuitry complicated, which increased the sensor size to 25.4 mm (1 inch). The increased size requires more housing material and makes the sensor heavier. The weight was increased to 10 gram compared to BIRD I (3.9 gram.

7.7 Conclusion

This chapter describes the development of BIRD Next, which is an improved version of BIRD II. The BIRD Next implemented wireless communication and wireless charging and is waterproof. An Android App was developed to interface with the sensor through Bluetooth. The sensing range and frequency of BIRD Next were significantly improved, but the sensitivity and precision were reduced when compared to BIRD II. The wireless communication enabled users to monitor the sensor data in real-time. The waterproof capacity allows the sensor to

be used for fruits, such as cranberries and strawberries, whose harvesting and postharvest processing involves water.

Chapter 8

Conclusion and Future Work

This dissertation described the development of an Unmanned Aerial System (UAS) for field-based high-throughput phenotyping, a Modular Agricultural Robotic System (MARS), and the next generation Berry Impact Recording Device (BIRD Next). The UAS can be used to measure phenotypic traits at the plot level, including canopy height, canopy volume, canopy cover, canopy temperature, canopy vegetation indices and cotton bloom count. The MARS robot's performance was tested in agricultural settings, and its application in phenotyping research was demonstrated through two field tests in a cotton field. The UAS and MARS developed herein provided useful research tools for breeding programs, digital farming and other agricultural studies. The BIRD Next implemented wireless communication and wireless charging and is waterproof. Its sensing range and recording frequency were significantly improved over that of the previous version.

Because of the time and scope of the dissertation, several limitations were identified in this dissertation:

1. The field calibration methods for the multispectral and thermal cameras on the UAS need to be tested and validated with more data samples. The LiDAR data were not processed because of a lack of accurate position and pose measurement.

2. The cotton bloom detection used traditional image processing methods for selecting potential flowers and a simple CNN for bloom classification. An end-to-end CNN design can be used to streamline the data processing.

3. The design of MARS robots can be optimized in the mechanical structure, electronic design, and control algorithms. More field tests should be conducted to evaluate its robustness in agricultural environments.

4. The surface of the BIRD Next is not uniform, which can result in different measurements for the same impact. Therefore, adequate replicates should be collected to reduce measurement variation.

Future studies can be conducted to address the limitations and issues identified.

1. Improve the design of the UAS by integrating RTK-GNSS and IMU to improve the localization accuracy so that the LiDAR data can be correctly processed.

2. Advanced data processing methods, including deep learning, can be explored for extracting phenotypical traits from the multimodal data collected by the UAS developed.

3. It is worth to explore recently developed deep learning methods for cotton bloom detection and counting.

4. The MARS robots developed are expected to be used in many agricultural tasks, including plant phenotyping, weeding, irrigation and sowing. Developing attachments for those tasks can significantly expand the functionality of the MARS robots.

5. Real-time data processing algorithms using edge computing can be explored for the UAS and MARS robots.

6. Collaboration between the UAS and MARS and coordination of multiple MARS robots can be explored to improve efficiency.

7. The surface uniformity of the BIRD Next can be improved by improving the fabrication process.

Appendix A

Supplementary Data for Chapter 4

We performed multiple comparisons among genotypes for the maximum height and manually measured maximum height (ANOVA statistical test). The test was performed for each dataset. Because of small data sample (8 samples per genotype for dataset 09/30 and 10/07 and 4 samples per genotype for dataset 10/16, 10/19, 10/23 and 10/30), the calculated maximum height and manually measured maximum height did not consistently have the same mean separation for all the datasets. However, it is clear that manual measurement consistently showed statistical difference between genotype 2 and 5 in all data sets except one (09/30). Our UAV based system also consistently showed the difference between the two genotypes.

Table A.1: Multiple comparison tests among genotypes for each dataset's calculated maximum height and manually measured maximum height. ∗indicates a significant statistical difference ($p<0.05$), otherwise, no significant statistical difference between the two genotypes was found.

	Calculated maximum height						**Manually measured maximum height**							**Calculated maximum height**						**Manually measured maximum height**					
Genotype	**1**	**2**	**3**	**4**	**5**	**6**	**1**	**2**	**3**	**4**	**5**	**6**		**1**	**2**	**3**	**4**	**5**	**6**	**1**	**2**	**3**	**4**	**5**	**6**
	9/30													**10/19**											
1		∗													∗										
2	∗		∗											∗		∗		∗						∗	
3		∗													∗										
4																									
5															∗						∗				
6																									
	10/07													**10/23**											
1		∗						∗																	
2	∗		∗		∗		∗		∗		∗					∗		∗						∗	
3		∗						∗							∗										
4																									
5		∗						∗							∗						∗				
6																									
	10/16													**10/30**											
1		∗													∗										
2	∗		∗	∗	∗						∗			∗		∗	∗	∗				∗		∗	
3		∗													∗						∗				
4		∗													∗										
5		∗						∗							∗						∗				
6																									

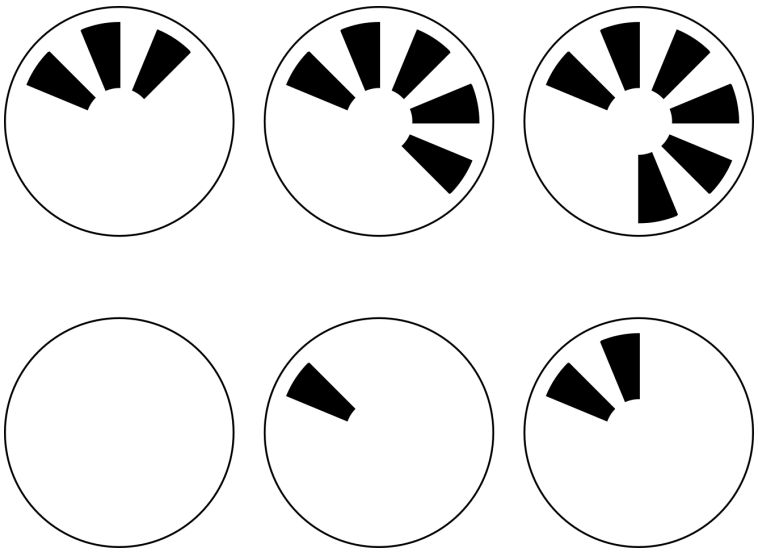

Figure A.1: Ground calibration target patterns.

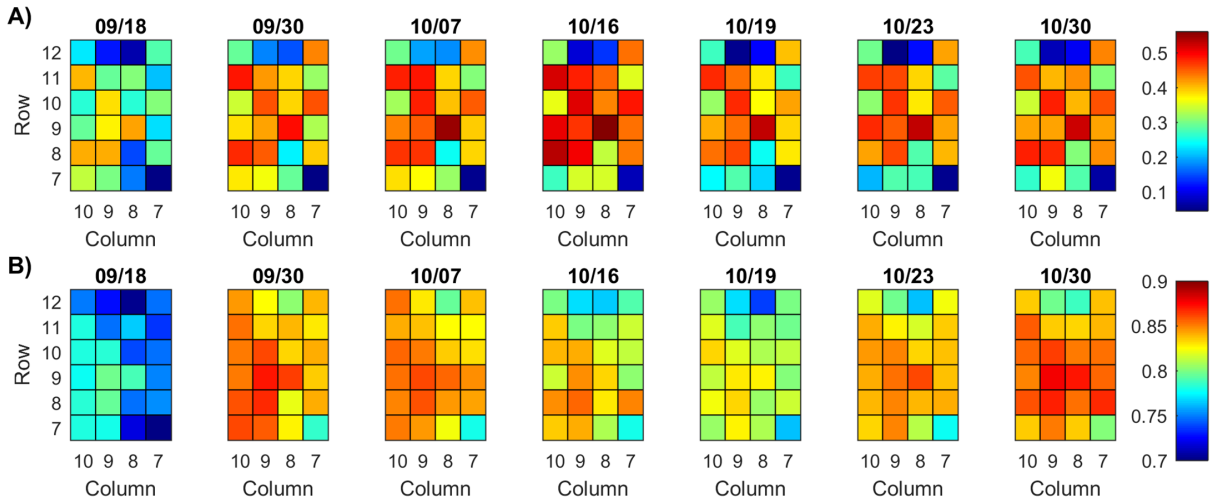

Figure A.2: Canopy cover (A) and NDVI (B) for plots in rows 7 to 12 on different dates.

Appendix B

Supplementary Data for Chapter 5

Figure B.1 was generated from preliminary data using the plots from field 1 (single plot layout) that had both manual and image count. There are 70, 41 and 3 plots on 79, 86 and 107 days after planting, respectively. The overall trend from the image count matches with that from the manual count.

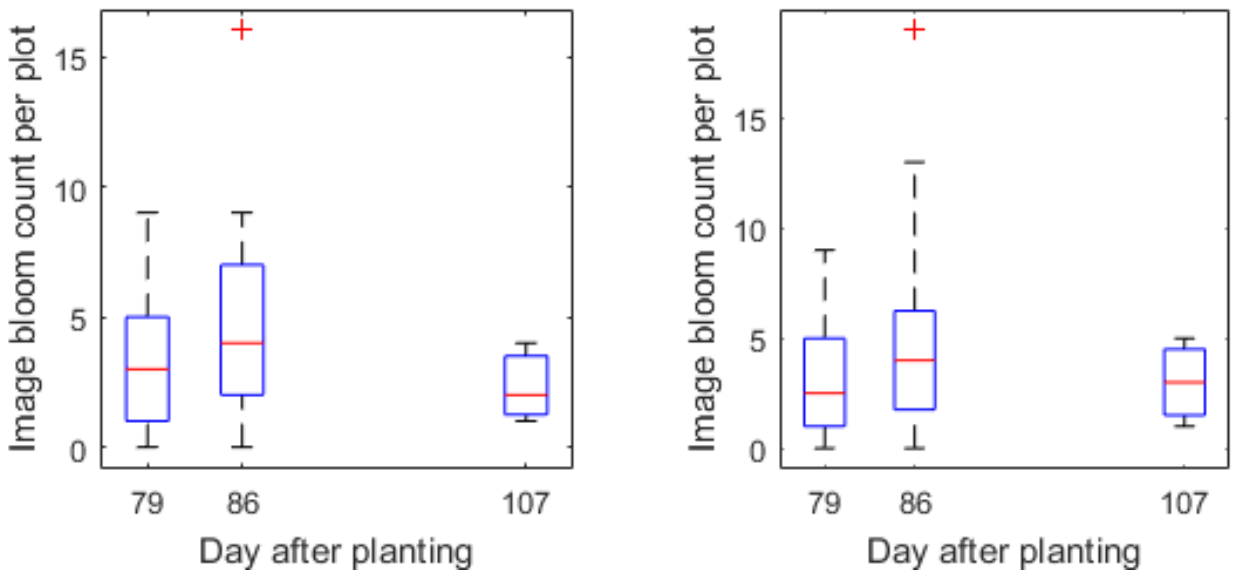

Figure B.1: Boxplot of the flower count over time for field 1.

Field 2 has four genotypes and each genotype has 64 plots. The four genotypes are GA2011158 (genotype 1), GA2009037 (genotype 2), GA2010074 (genotype 3), Americot conventional (genotype 4). Figure B.2 to Figure B.5 were generated from the preliminary data using plots from four genotypes in field 2 (10-feet plot) that had both manual and image count of blooms. There are 125, 9, 4 and 6 plots on 60, 67, 74 and 88 days after planting, respectively. The overall trend from the image count matches with that from the manual count. With limited plots, however, the underestimation could affect the trend. For example, image count on 74 days after planting has large underestimation, which makes the trend different from manual count on genotype 3.

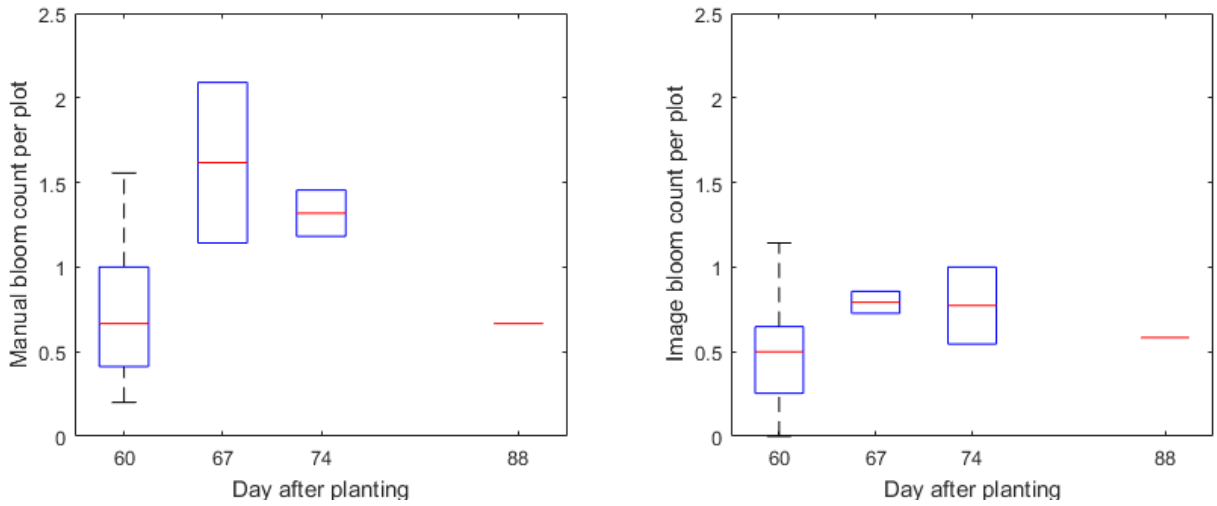

Figure B.2: Boxplot of the flower count over time for field 2 for genotype 1. Sample size for DAP 60, 67, 74, 88 are 29, 2, 2, and 1, respectively.

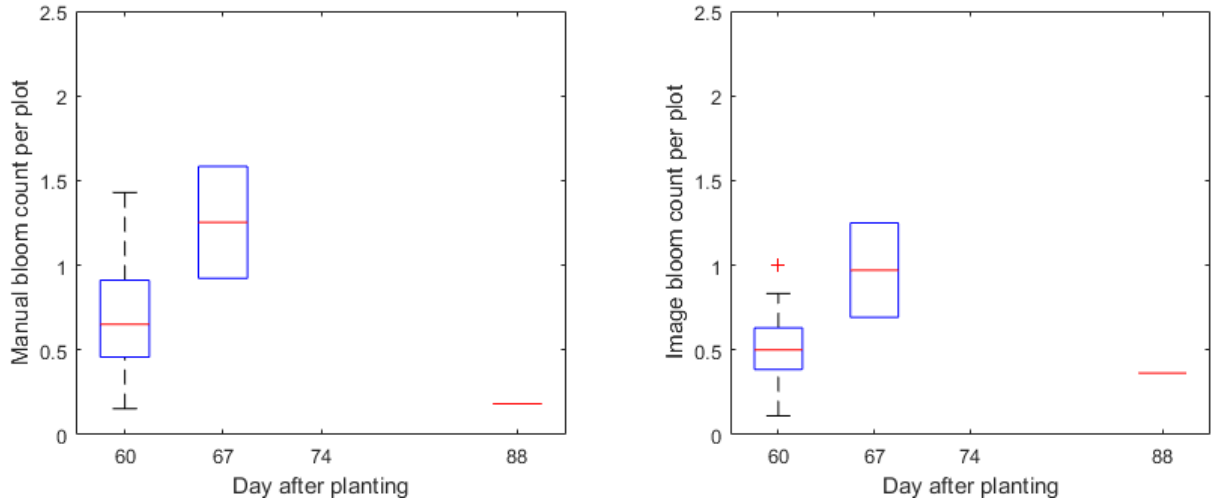

Figure B.3: Boxplot of the flower count over time for field 2 for genotype 2. Sample size for DAP 60, 67, 88 are 32, 2, and 1, respectively.

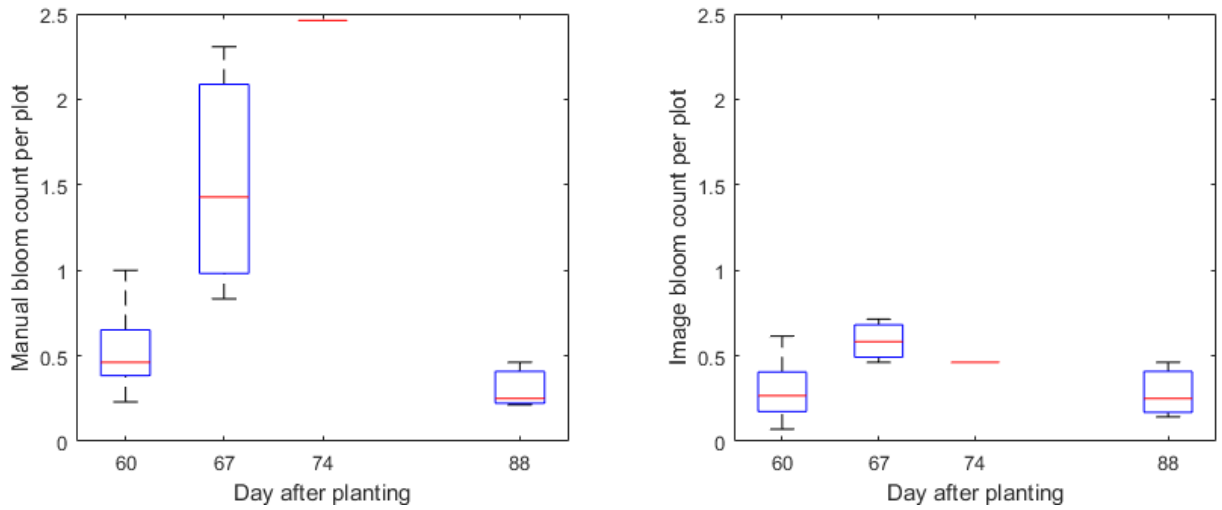

Figure B.4: Boxplot of the flower count over time for field 2 for genotype 3. Sample size for DAP 60, 67, 74, 88 are 32, 3, 1, and 3, respectively.

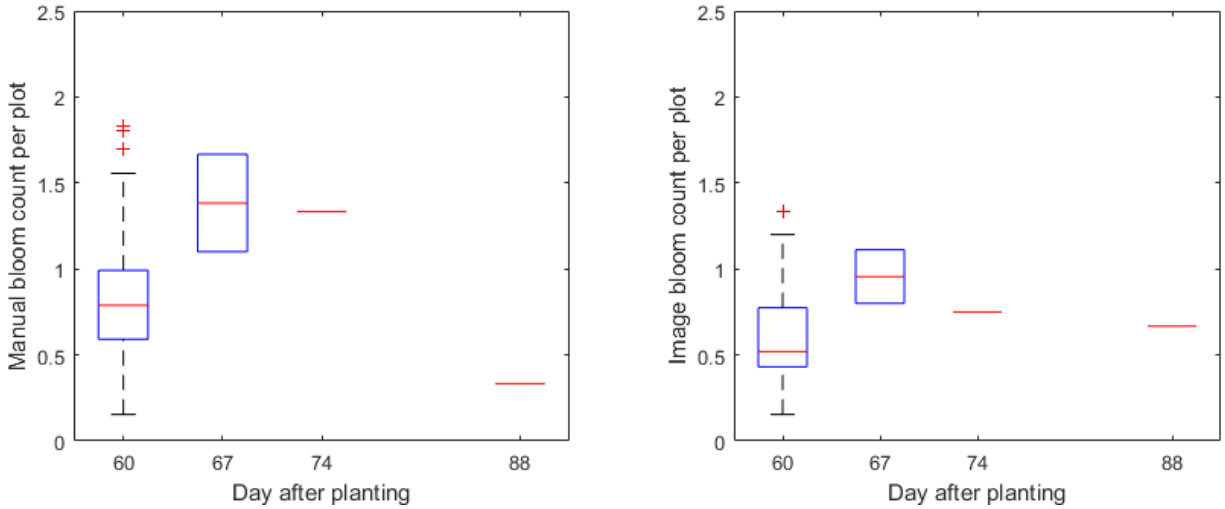

Figure B.5: Boxplot of the flower count over time for field 2 for genotype 4. Sample size for DAP 60, 67, 74, 88 are 32, 2, 1, and 1, respectively.

References

[1] David Tilman et al. "Global food demand and the sustainable intensification of agriculture". In: *Proceedings of the National Academy of Sciences* 108.50 (2011), pp. 20260–20264. ISSN: 0027-8424. DOI: 10.1073/pnas.1116437108. eprint: https://www.pnas.org/content/108/50/20260.full.pdf. URL: https://www.pnas.org/content/108/50/20260.

[2] Ann Steensland. *2020 Global Agricultural Productivity Report: Productivity in a time of pandemics*. Virginia Tech College of Agriculture and Life Sciences, 2020.

[3] Xiuliang Jin et al. "High-throughput estimation of crop traits: A review of ground and aerial phenotyping platforms". In: *IEEE Geoscience and Remote Sensing Magazine* (2020), pp. 1–33.

[4] Wanneng Yang et al. "Crop phenomics and high-throughput phenotyping: past decades, current challenges, and future perspectives". In: *Molecular Plant* 13.2 (2020), pp. 187–214.

[5] Lucas Busemeyer et al. "BreedVision—A multi-sensor platform for non-destructive field-based phenotyping in plant breeding". In: *Sensors* 13.3 (2013), pp. 2830–2847.

[6] Pedro Andrade-Sanchez et al. "Development and evaluation of a field-based high-throughput phenotyping platform". In: *Functional Plant Biology* 41.1 (2014), pp. 68–79.

[7] Bablu Sharma and Glen L Ritchie. "High-throughput phenotyping of cotton in multiple irrigation environments". In: *Crop Science* 55.2 (2015), pp. 958–969.

[8] Geng Bai et al. "A multi-sensor system for high throughput field phenotyping in soybean and wheat breeding". In: *Computers and Electronics in Agriculture* 128 (2016), pp. 181–192.

[9] Anna Kicherer et al. "Phenoliner: A new field phenotyping platform for grapevine research". In: *Sensors* 17.7 (2017), p. 1625.

[10] Nico Higgs et al. "ProTractor: a lightweight ground imaging and analysis system for early-season field phenotyping". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 0–0.

[11] Yu Jiang et al. "GPhenoVision: A ground mobile system with multi-modal imaging for field-based high throughput phenotyping of cotton". In: *Scientific reports* 8.1 (2018), pp. 1–15.

[12] Jared L Crain et al. "Development and deployment of a portable field phenotyping platform". In: *Crop Science* 56.3 (2016), pp. 965–975.

[13] Alison L Thompson et al. "Deploying a proximal sensing cart to identify drought-adaptive traits in upland cotton for high-throughput phenotyping". In: *Frontiers in plant science* 9 (2018), p. 507.

[14] Dhananjay Kumar et al. “Affordable Phenotyping of Winter Wheat under Field and Controlled Conditions for Drought Tolerance”. In: *Agronomy* 10.6 (2020), p. 882.

[15] Katherine Meacham-Hensold et al. “Plot-level rapid screening for photosynthetic parameters using proximal hyperspectral imaging”. In: *Journal of experimental botany* 71.7 (2020), pp. 2312–2328.

[16] Alison L Thompson et al. “Professor: A motorized field-based phenotyping cart”. In: *HardwareX* 4 (2018), e00025.

[17] Nicolas Virlet et al. “Field Scanalyzer: An automated robotic field phenotyping platform for detailed crop monitoring”. In: *Functional Plant Biology* 44.1 (2017), pp. 143–153.

[18] Katia Beauchêne et al. “Management and characterization of abiotic stress via PhénoField®, a high-throughput field phenotyping platform.” In: *Frontiers in plant science* 10 (2019), p. 904.

[19] Norbert Kirchgessner et al. “The ETH field phenotyping platform FIP: a cable-suspended multi-sensor system”. In: *Functional Plant Biology* 44.1 (2017), pp. 154–168.

[20] Geng Bai et al. “NU-Spidercam: A large-scale, cable-driven, integrated sensing and robotic system for advanced phenotyping, remote sensing, and agronomic research”. In: *Computers and Electronics in Agriculture* 160 (2019), pp. 71–81.

[21] Yu Jiang et al. “Quantitative analysis of cotton canopy size in field conditions using a consumer-grade RGB-D camera”. In: *Frontiers in plant science* 8 (2018), p. 2233.

[22] Xiaodong Zhou and Shusheng Bi. "A survey of bio-inspired compliant legged robot designs". In: *Bioinspiration & biomimetics* 7.4 (2012), p. 041001.

[23] Zhongzhong Zhang et al. "High precision control and deep learning-based corn stand counting algorithms for agricultural robot". In: *Autonomous Robots* (2020), pp. 1–14.

[24] Ali Shafiekhani et al. "Vinobot and vinoculer: Two robotic platforms for high-throughput field phenotyping". In: *Sensors* 17.1 (2017), p. 214.

[25] Sergio Cubero et al. "RobHortic: A Field Robot to Detect Pests and Diseases in Horticultural Crops by Proximal Sensing". In: *Agriculture* 10.7 (2020), p. 276.

[26] Madeleine Stein, Suchet Bargoti, and James Underwood. "Image based mango fruit detection, localisation and yield estimation using multiple view geometry". In: *Sensors* 16.11 (2016), p. 1915.

[27] Tim Mueller-Sim et al. "The Robotanist: a ground-based agricultural robot for high-throughput crop phenotyping". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 3634–3639.

[28] James Underwood et al. "Efficient in-field plant phenomics for row-crops with an autonomous ground vehicle". In: *Journal of Field Robotics* 34.6 (2017), pp. 1061–1083.

[29] Adam Stager, Herbert G Tanner, and Erin E Sparks. "Design and Construction of Unmanned Ground Vehicles for Sub-Canopy Plant Phenotyping". In: *arXiv preprint arXiv:1903.10608* (2019).

[30] R Guzmán et al. "Autonomous hybrid GPS/reactive navigation of an unmanned ground vehicle for precision viticulture-VINBOT". In: *62nd German Winegrowers Conference At: Stuttgart*. 2016.

[31] Jawad Iqbal et al. "Development of a Multi-Purpose Autonomous Differential Drive Mobile Robot for Plant Phenotyping and Soil Sensing". In: *Electronics* 9.9 (2020), p. 1550.

[32] Lars Grimstad and Pål Johan From. "The Thorvald II agricultural robotic system". In: *Robotics* 6.4 (2017), p. 24.

[33] Owen Bawden et al. "Robot for weed species plant-specific management". In: *Journal of Field Robotics* 34.6 (2017), pp. 1179–1199.

[34] Arno Ruckelshausen et al. "BoniRob–an autonomous field robot platform for individual plant phenotyping". In: *Precision agriculture* 9.841 (2009), p. 1.

[35] Kjeld Jensen et al. "A low cost, modular robotics tool carrier for precision agriculture research". In: *Proc Int Conf on Precision Agriculture*. 2012.

[36] Anna Kicherer et al. "An automated field phenotyping pipeline for application in grapevine research". In: *Sensors* 15.3 (2015), pp. 4823–4836.

[37] Taylor L. Tuel. "A Robotic Proximal Sensing Platform for In-Field High-Throughput Crop Phenotyping". English. PhD thesis. 2019, p. 51. ISBN: 9781088352618. URL: http://proxy-remote.galib.uga.edu:80/login?url=https://www.proquest.com/docview/2311651907?accountid=14537.

[38] Maria G Salas Fernandez et al. “A high-throughput, field-based phenotyping technology for tall biomass crops”. In: *Plant physiology* 174.4 (2017), pp. 2008–2022.

[39] Sierra N Young, Erkan Kayacan, and Joshua M Peschel. “Design and field evaluation of a ground robot for high-throughput phenotyping of energy sorghum”. In: *Precision Agriculture* 20.4 (2019), pp. 697–722.

[40] Joshua N Murman. “Flex-Ro: A Robotic High Throughput Field Phenotyping System”. In: (2019).

[41] Carlos M Lopes et al. “Vineyard yeld estimation by VINBOT robot-preliminary results with the white variety Viosinho”. In: *Proceedings 11th Int. Terroir Congress. Jones, G. and Doran, N.(eds.), pp. 458-463. Southern Oregon University, Ashland, USA*. Jones, G.; Doran, N.(eds.) 2016.

[42] James P Underwood et al. “Mapping almond orchard canopy volume, flowers, fruit and yield using lidar and vision sensors”. In: *Computers and Electronics in Agriculture* 130 (2016), pp. 83–96.

[43] Suchet Bargoti and James Underwood. “Deep fruit detection in orchards”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 3626–3633.

[44] Harjatin Singh Baweja et al. “Stalknet: A deep learning pipeline for high-throughput measurement of plant stalk count and stalk width”. In: *Field and Service Robotics*. Springer. 2018, pp. 271–284.

[45] Ali Shafiekhani, Felix B Fritschi, and Guilherme N DeSouza. "Vinobot and vinoculer: from real to simulated platforms". In: *Autonomous Air and Ground Sensing Systems for Agricultural Optimization and Phenotyping III*. Vol. 10664. International Society for Optics and Photonics. 2018, 106640A.

[46] Anwesa Choudhuri and Girish Chowdhary. "Crop stem width estimation in highly cluttered field environment". In: *Proceedings of the Computer Vision Problems in Plant Phenotyping (CVPPP 2018), Newcastle, UK* (2018), pp. 6–13.

[47] Wyatt McAllister et al. "Agbots: Weeding a field with a team of autonomous robots". In: *Computers and Electronics in Agriculture* 163 (2019), p. 104827.

[48] Vitor AH Higuti et al. "Under canopy light detection and ranging-based autonomous navigation". In: *Journal of Field Robotics* 36.3 (2019), pp. 547–567.

[49] David Hall et al. "Towards unsupervised weed scouting for agricultural robotics". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 5223–5230.

[50] Yin Bao et al. "Field-based robotic phenotyping of sorghum plant architecture using stereo vision". In: *Journal of Field Robotics* 36.2 (2019), pp. 397–415.

[51] Ya Xiong et al. "Development and field evaluation of a strawberry harvesting robot with a cable-driven gripper". In: *Computers and electronics in agriculture* 157 (2019), pp. 392–402.

[52] Tuan D Le et al. "A low-cost and efficient autonomous row-following robot for food production in polytunnels". In: *Journal of Field Robotics* 37.2 (2020), pp. 309–321.

[53] Lars Grimstad et al. “A novel autonomous robot for greenhouse applications”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1–9.

[54] James P Underwood et al. “Real-time target detection and steerable spray for vegetable crops”. In: *Proceedings of the International Conference on Robotics and Automation: Robotics in Agriculture Workshop, Seattle, WA, USA*. 2015, pp. 26–30.

[55] Xuyong Tu, Jingyao Gai, and Lie Tang. “Robust navigation control of a 4WD/4WS agricultural robotic vehicle”. In: *Computers and Electronics in Agriculture* 164 (2019), p. 104892.

[56] Gerrit Polder and Jan Willem Hofstee. “Phenotyping large tomato plants in the greenhouse using a 3D light-field camera”. In: *2014 Montreal, Quebec Canada July 13–July 16, 2014*. American Society of Agricultural and Biological Engineers. 2014, p. 1.

[57] Johann Christian Rose et al. “Towards automated large-scale 3D phenotyping of vineyards under field conditions”. In: *Sensors* 16.12 (2016), p. 2136.

[58] Tianshuang Gao et al. “A novel multirobot system for plant phenotyping”. In: *Robotics* 7.4 (2018), p. 61.

[59] C Scholz et al. “Automatic soil penetrometer measurements and GIS based documentation with the autonomous field robot platform bonirob”. In: *12th International Conference of Precision Agriculture*. 2014.

[60] Sebastian Haug and Jörn Ostermann. “A crop/weed field image dataset for the evaluation of computer vision based precision agriculture tasks”. In: *European Conference on Computer Vision*. Springer. 2014, pp. 105–116.

[61] Nived Chebrolu et al. “Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields”. In: *The International Journal of Robotics Research* 36.10 (2017), pp. 1045–1052.

[62] Alberto Pretto et al. “Building an Aerial-Ground Robotics System for Precision Farming: An Adaptable Solution”. In: *IEEE Robotics & Automation Magazine* (2020).

[63] Tokihiro Fukatsu, Gen Endo, and Kazuki Kobayashi. “Field Experiments with a Mobile Robotic Field Server for Smart Agriculture”. In: *Proc. of WCCA-AFITA2016, OS6-2* (2016), pp. 1–4.

[64] Pablo Gonzalez-De-Santos et al. “Unmanned Ground Vehicles for Smart Farms”. In: *Agronomy*. IntechOpen, 2020.

[65] Francisco Yandun Narvaez et al. “A survey of ranging and imaging techniques for precision agriculture phenotyping”. In: *IEEE/ASME Transactions on Mechatronics* 22.6 (2017), pp. 2428–2439.

[66] Xinyu Gao et al. “Review of Wheeled Mobile Robots’ Navigation Problems and Application Prospects in Agriculture”. In: *IEEE Access* 6 (2018), pp. 49248–49268.

[67] Bernard Benet et al. “Development of autonomous robotic platforms for sugar beet crop phenotyping using artificial vision”. In: 2018.

[68] Giuseppe Quaglia et al. “Design of a UGV Powered by Solar Energy for Precision Agriculture”. In: *Robotics* 9.1 (2020), p. 13.

[69] Yin Bao, Lie Tang, and Dylan Shah. “Robotic 3d plant perception and leaf probing with collision-free motion planning for automated indoor plant phenotyping”. In: *2017 ASABE Annual International Meeting*. American Society of Agricultural and Biological Engineers. 2017, p. 1.

[70] Tsunghan Han and Changying Li. “Developing a High Precision Cotton Boll Counting System Using Active Sensing”. In: *2019 ASABE Annual International Meeting*. American Society of Agricultural and Biological Engineers. 2019, p. 1.

[71] Morgan Quigley et al. “ROS: an open-source Robot Operating System”. In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan. 2009, p. 5.

[72] Kjeld Jensen et al. “Towards an open software platform for field robots in precision agriculture”. In: *Robotics* 3.2 (2014), pp. 207–234.

[73] Avital Bechar and Clément Vigneault. “Agricultural robots for field operations. Part 2: Operations and systems”. In: *Biosystems engineering* 153 (2017), pp. 110–128.

[74] R Craig Coulter. *Implementation of the pure pursuit path tracking algorithm*. Tech. rep. Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

[75] Wenyu Zhang et al. “Double-DQN based path smoothing and tracking control method for robotic vehicle navigation”. In: *Computers and Electronics in Agriculture* 166 (2019), p. 104985.

[76] David Ball et al. “Vision-based obstacle detection and navigation for an agricultural robot”. In: *Journal of field robotics* 33.8 (2016), pp. 1107–1130.

[77] Zhiqiang Zhai et al. “Multi-crop-row detection algorithm based on binocular vision”. In: *Biosystems engineering* 150 (2016), pp. 89–103.

[78] M Kise, Q Zhang, and F Rovira Más. “A stereovision-based crop row detection method for tractor-automated guidance”. In: *Biosystems engineering* 90.4 (2005), pp. 357–367.

[79] Marianne Bakken, Richard JD Moore, and Pål From. “End-to-end Learning for Autonomous Crop Row-following”. In: *IFAC-PapersOnLine* 52.30 (2019), pp. 102–107.

[80] Flavio BP Malavazi et al. “LiDAR-only based navigation algorithm for an autonomous agricultural robot”. In: *Computers and Electronics in Agriculture* 154 (2018), pp. 71–79.

[81] Jawad Iqbal et al. “Simulation of an Autonomous Mobile Robot for LiDAR-Based In-Field Phenotyping and Navigation”. In: *Robotics* 9.2 (2020), p. 46.

[82] Oscar C Barawid Jr et al. “Development of an autonomous navigation system using a two-dimensional laser scanner in an orchard application”. In: *Biosystems Engineering* 96.2 (2007), pp. 139–149.

[83] Santosh A Hiremath et al. “Laser range finder model for autonomous navigation of a robot in a maize field using a particle filter”. In: *Computers and Electronics in Agriculture* 100 (2014), pp. 41–50.

[84] Pieter M Blok et al. “Robot navigation in orchards with localization based on Particle filter and Kalman filter”. In: *Computers and Electronics in Agriculture* 157 (2019), pp. 261–269.

[85] Jorge Miguel Mendes et al. “Localization based on natural features detector for steep slope vineyards”. In: *Journal of Intelligent & Robotic Systems* 93.3-4 (2019), pp. 433–446.

[86] Lars Grimstad and Pål J From. “Software Components of the Thorvald II Modular Robot”. In: (2018).

[87] Mostafa Sharifi et al. “Mechatronic design and development of a non-holonomic omnidirectional mobile robot for automation of primary production”. In: *Cogent Engineering* 3.1 (2016), p. 1250431.

[88] Novian Habibie et al. “Fruit mapping mobile robot on simulated agricultural area in Gazebo simulator using simultaneous localization and mapping (SLAM)”. In: *2017 International Symposium on Micro-NanoMechatronics and Human Science (MHS)*. IEEE. 2017, pp. 1–7.

[89] Brian P Gerkey et al. “Most valuable player: A robot device server for distributed control”. In: *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*. Vol. 3. IEEE. 2001, pp. 1226–1231.

[90] O Michel. “WebotsTM: Professional mobile robot simulation, 2004”. In: *arXiv preprint cs* 412052 ().

[91] Nathan Koenig and Andrew Howard. "Design and use paradigms for gazebo, an open-source multi-robot simulator". In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. Vol. 3. IEEE. 2004, pp. 2149–2154.

[92] Eric Rohmer, Surya PN Singh, and Marc Freese. "V-REP: A versatile and scalable robot simulation framework". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 1321–1326.

[93] Redmond R Shamshiri et al. "Simulation software and virtual environments for acceleration of agricultural robotics: Features highlights and performance comparison". In: (2018).

[94] Patricio Nebot, Joaquín Torres-Sospedra, and Rafael J Martínez. "A new HLA-based distributed control architecture for agricultural teams of robots in hybrid applications with real and simulated devices or environments". In: *Sensors* 11.4 (2011), pp. 4385–4400.

[95] Naoum Tsolakis, Dimitrios Bechtsis, and Dionysis Bochtis. "Agros: A robot operating system based emulation tool for agricultural robotics". In: *Agronomy* 9.7 (2019), p. 403.

[96] Ulrich Weiss and Peter Biber. "Plant detection and mapping for agricultural robots using a 3D LIDAR sensor". In: *Robotics and autonomous systems* 59.5 (2011), pp. 265–273.

[97] Erkan Kayacan, Zhong-Zhong Zhang, and Girish Chowdhary. "Embedded High Precision Control and Corn Stand Counting Algorithms for an Ultra-Compact 3D Printed Field Robot." In: *Robotics: Science and Systems*. 2018.

[98] Qian Zhang et al. "Applications of Deep Learning for Dense Scenes Analysis in Agriculture: A Review". In: *Sensors* 20.5 (2020), p. 1520.

[99] Yu Jiang, Changying Li, et al. "Convolutional Neural Networks for Image-Based High-Throughput Plant Phenotyping: A Review". In: *Plant Phenomics* 2020 (2020), p. 4152816.

[100] Asher Bender, Brett Whelan, and Salah Sukkarieh. "A high-resolution, multimodal data set for agricultural robotics: A Ladybird's-eye view of Brassica". In: *Journal of Field Robotics* 37.1 (2020), pp. 73–96.

[101] José Luis Araus and Jill E Cairns. "Field high-throughput phenotyping: the new crop breeding frontier". In: *Trends in plant science* 19.1 (2014), pp. 52–61.

[102] Guijun Yang et al. "Unmanned aerial vehicle remote sensing for field-based crop phenotyping: current status and perspectives". In: *Frontiers in plant science* 8 (2017), p. 1111.

[103] Sindhuja Sankaran et al. "Low-altitude, high-resolution aerial imaging systems for row and field crop phenotyping: A review". In: *European Journal of Agronomy* 70 (2015), pp. 112–123.

[104] Chuanqi Xie and Ce Yang. "A review on plant high-throughput phenotyping traits using UAV-based sensors". In: *Computers and Electronics in Agriculture* 178 (2020), p. 105731.

[105] Songyang Li et al. "Combining color indices and textures of UAV-based digital imagery for rice LAI estimation". In: *Remote Sensing* 11.15 (2019), p. 1763.

[106] Bo Li et al. "The estimation of crop emergence in potatoes by UAV RGB imagery". In: *Plant methods* 15.1 (2019), p. 15.

[107] Rui Xu et al. "Aerial images and convolutional neural network for cotton bloom detection". In: *Frontiers in plant science* 8 (2018), p. 2235.

[108] Juliane Bendig et al. "Estimating biomass of barley using crop surface models (CSMs) derived from UAV-based RGB imaging". In: *Remote sensing* 6.11 (2014), pp. 10395–10412.

[109] Juliane Bendig et al. "Combining UAV-based plant height from crop surface models, visible, and near infrared vegetation indices for biomass monitoring in barley". In: *International Journal of Applied Earth Observation and Geoinformation* 39 (2015), pp. 79–87.

[110] Ramón A. Díaz-Varela et al. "High-resolution airborne UAV imagery to assess olive tree crown parameters using 3D photo reconstruction: application in breeding trials". In: *Remote Sensing* 7.4 (2015), pp. 4213–4232.

[111] Fenner H Holman et al. “High throughput field phenotyping of wheat plant height and growth rate in field plot trials using UAV based remote sensing”. In: *Remote Sensing* 8.12 (2016), p. 1031.

[112] Simon Madec et al. “High-throughput phenotyping of plant height: comparing unmanned aerial vehicles and ground LiDAR estimates”. In: *Frontiers in plant science* 8 (2017), p. 2002.

[113] E Raymond Hunt et al. “Acquisition of NIR-green-blue digital photographs from unmanned aircraft for crop monitoring”. In: *Remote Sensing* 2.1 (2010), pp. 290–305.

[114] Mainassara Zaman-Allah et al. “Unmanned aerial platform-based multi-spectral imaging for field phenotyping of maize”. In: *Plant methods* 11.1 (2015), pp. 1–10.

[115] Aaron Patrick et al. “High throughput phenotyping of tomato spot wilt disease in peanuts using unmanned aerial systems and multispectral imaging”. In: *IEEE Instrumentation & Measurement Magazine* 20.3 (2017), pp. 4–12.

[116] Rui Xu, Changying Li, and Andrew H Paterson. “Multispectral imaging and unmanned aerial systems for cotton plant phenotyping”. In: *PloS one* 14.2 (2019), e0205083.

[117] Mengjiao Yang et al. “Assessment of water and nitrogen use efficiencies through UAV-based multispectral phenotyping in winter wheat”. In: *Frontiers in plant science* 11 (2020), p. 927.

[118] RD Jackson, RJ Reginato, and SB Idso. "Wheat canopy temperature: a practical tool for evaluating water requirements". In: *Water resources research* 13.3 (1977), pp. 651–656.

[119] Ray D Jackson et al. "Canopy temperature as a crop water stress indicator". In: *Water resources research* 17.4 (1981), pp. 1133–1138.

[120] Victoria Gonzalez-Dugo et al. "Using high resolution UAV thermal imagery to assess the variability in the water status of five fruit tree species within a commercial orchard". In: *Precision Agriculture* 14.6 (2013), pp. 660–678.

[121] David Gómez-Candón et al. "Field phenotyping of water stress at tree scale by UAV-sensed imagery: new insights for thermal acquisition and calibration". In: *Precision agriculture* 17.6 (2016), pp. 786–800.

[122] Liyuan Zhang et al. "Maize canopy temperature extracted from UAV thermal and RGB imagery and its application in water stress monitoring". In: *Frontiers in plant science* 10 (2019), p. 1270.

[123] Victoria Gonzalez-Dugo et al. "Using high-resolution hyperspectral and thermal airborne imagery to assess physiological condition in the context of wheat phenotyping". In: *Remote Sensing* 7.10 (2015), pp. 13586–13605.

[124] Micol Rossini et al. "Assessing canopy PRI from airborne imagery to map water stress in maize". In: *ISPRS journal of photogrammetry and remote sensing* 86 (2013), pp. 168–177.

[125] Maitiniyazi Maimaitijiang et al. “Unmanned Aerial System (UAS)-based phenotyping of soybean using multi-sensor data fusion and extreme learning machine”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 134 (2017), pp. 43–58.

[126] Aijing Feng et al. “Yield estimation in cotton using UAV-based multi-sensor imagery”. In: *Biosystems Engineering* 193 (2020), pp. 101–114.

[127] Julia Kelly et al. “Challenges and best practices for deriving temperature data from an uncalibrated UAV thermal infrared camera”. In: *Remote Sensing* 11.5 (2019), p. 567.

[128] Daniel B Goldman. “Vignette and exposure calibration and compensation”. In: *IEEE transactions on pattern analysis and machine intelligence* 32.12 (2010), pp. 2276–2288.

[129] Baabak Mamaghani and Carl Salvaggio. “Multispectral sensor calibration and characterization for sUAS remote sensing”. In: *Sensors* 19.20 (2019), p. 4453.

[130] Rubén Usamentiaga et al. “Infrared Thermography for Temperature Measurement and Non-Destructive Testing”. In: *Sensors* 14.7 (2014), pp. 12305–12348. ISSN: 1424-8220. DOI: 10.3390/s140712305. URL: https://www.mdpi.com/1424-8220/14/7/12305.

[131] W Minkina and D Klecha. “1.4-Modeling of Atmospheric Transmission Coefficient in Infrared for Thermovision Measurements”. In: *Proceedings IRS² 2015* (2015), pp. 903–907.

[132] Gaetano Messina and Giuseppe Modica. “Applications of UAV Thermal Imagery in Precision Agriculture: State of the Art and Future Research Outlook”. In: *Remote Sensing* 12.9 (2020), p. 1491.

[133] Chiachung Chen. “Determining the leaf emissivity of three crops by infrared thermometry”. In: *Sensors* 15.5 (2015), pp. 11387–11401.

[134] Robert P Madding. “Emissivity measurement and temperature correction accuracy considerations”. In: *Thermosense XXI*. Vol. 3700. International Society for Optics and Photonics. 1999, pp. 393–401.

[135] Sascha Heinemann et al. “Land Surface Temperature Retrieval for Agricultural Areas Using a Novel UAV Platform Equipped with a Thermal Infrared and Multispectral Sensor”. In: *Remote Sensing* 12.7 (2020), p. 1075.

[136] Philip HS Torr and Andrew Zisserman. “MLESAC: A new robust estimator with application to estimating image geometry”. In: *Computer vision and image understanding* 78.1 (2000), pp. 138–156.

[137] Georg Bareth et al. “A comparison of UAV-and TLS-derived plant height for crop monitoring: using polygon grids for the analysis of crop surface models (CSMs)”. In: *Photogrammetrie-Fernerkundung-Geoinformation* 2016.2 (2016), pp. 85–94.

[138] Shangpeng Sun et al. “In-field high throughput phenotyping and cotton plant growth analysis using LiDAR”. In: *Frontiers in plant science* 9 (2018), p. 16.

[139] Miguel S Torino et al. “Evaluation of vegetation indices for early assessment of corn status and yield potential in the Southeastern United States”. In: *Agronomy Journal* 106.4 (2014), pp. 1389–1401.

[140] Yeyin Shi et al. “Unmanned aerial vehicles for high-throughput phenotyping and agronomic research”. In: *PloS one* 11.7 (2016), e0159781.

[141] Erich Seifert et al. “Influence of drone altitude, image overlap, and optical sensor resolution on multi-view reconstruction of forest images”. In: *Remote sensing* 11.10 (2019), p. 1252.

[142] Sen Cao et al. “Radiometric calibration assessments for UAS-borne multispectral cameras: Laboratory and field protocols”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 149 (2019), pp. 132–145.

[143] Ronald L Phillips. “Mobilizing science to break yield barriers”. In: *Crop Science* 50 (2010), S–99.

[144] Llorenç Cabrera-Bosquet et al. “High-throughput phenotyping and genomic selection: The frontiers of crop breeding converge F”. In: *Journal of integrative plant biology* 54.5 (2012), pp. 312–320.

[145] Alexis Comar et al. “A semi-automatic system for high throughput phenotyping wheat cultivars in-field conditions: description and first results”. In: *Functional Plant Biology* 39.11 (2012), pp. 914–924.

[146] David J Mulla. "Twenty five years of remote sensing in precision agriculture: Key advances and remaining knowledge gaps". In: *Biosystems engineering* 114.4 (2013), pp. 358–371.

[147] Jorge Gago et al. "UAVs challenge to assess water stress for sustainable agriculture". In: *Agricultural water management* 153 (2015), pp. 9–19.

[148] Francisco Garcia-Ruiz et al. "Comparison of two aerial imaging platforms for identification of Huanglongbing-infected citrus trees". In: *Computers and Electronics in Agriculture* 91 (2013), pp. 106–115.

[149] Pablo J Zarco-Tejada, Victoria González-Dugo, and José AJ Berni. "Fluorescence, temperature and narrow-band indices acquired from a UAV platform for water stress detection using a micro-hyperspectral imager and a thermal camera". In: *Remote sensing of environment* 117 (2012), pp. 322–337.

[150] Jose AJ Berni et al. "Thermal and narrowband multispectral remote sensing for vegetation monitoring from an unmanned aerial vehicle". In: *IEEE Transactions on geoscience and Remote Sensing* 47.3 (2009), pp. 722–738.

[151] Javier Baluja et al. "Assessment of vineyard water status variability by thermal and multispectral imagery using an unmanned aerial vehicle (UAV)". In: *Irrigation Science* 30.6 (2012), pp. 511–522.

[152] R.D. Jackson et al. "Estimation of daily evapotranspiration from one time-of-day measurements". In: *Agricultural Water Management* 7.1-3 (1983), pp. 351–362.

[153] Yanbo Huang et al. “Cotton yield estimation using very high-resolution digital images acquired with a low-cost small unmanned aerial vehicle”. In: *Transactions of the ASABE* 59.6 (2016), pp. 1563–1574.

[154] Tianxing Chu et al. “Cotton growth modeling and assessment using unmanned aircraft system visual-band imagery”. In: *Journal of Applied Remote Sensing* 10.3 (2016), p. 036018.

[155] Duli Zhao et al. “Canopy reflectance in cotton for growth assessment and lint yield prediction”. In: *European Journal of Agronomy* 26.3 (2007), pp. 335–344.

[156] Chih-Chung Chang and Chih-Jen Lin. “LIBSVM: A library for support vector machines”. In: *ACM transactions on intelligent systems and technology (TIST)* 2.3 (2011), pp. 1–27.

[157] Dehua Zhao et al. “A comparative analysis of broadband and narrowband derived vegetation indices in predicting LAI and CCD of a cotton canopy”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 62.1 (2007), pp. 25–33.

[158] Jorge Torres-Sánchez et al. “High-throughput 3-D monitoring of agricultural-tree plantations with unmanned aerial vehicle (UAV) technology”. In: *PloS one* 10.6 (2015), e0130479.

[159] Steven J Franks. “The unique and multifaceted importance of the timing of flowering”. In: *American journal of botany* 102.9 (2015), pp. 1401–1402.

[160] Pablo J Zarco-Tejada et al. "Tree height quantification using very high resolution imagery acquired from an unmanned aerial vehicle (UAV) and automatic 3D photo-reconstruction methods". In: *European journal of agronomy* 55 (2014), pp. 89–99.

[161] Randy Wells and William R Meredith Jr. "Comparative Growth of Obsolete and Modern Cotton Cultivars. III. Relationship of Yield to Observed Growth Characteristics 1". In: *Crop science* 24.5 (1984), pp. 868–872.

[162] William T Pettigrew. "Source-to-sink manipulation effects on cotton lint yield and yield components". In: *Agronomy journal* 86.4 (1994), pp. 731–735.

[163] James J Heitholt. "Cotton boll retention and its relationship to lint yield". In: *Crop science* 33.3 (1993), pp. 486–490.

[164] James J Heitholt. "Cotton flowering and boll retention in different planting configurations and leaf shapes". In: *Agronomy journal* 87.5 (1995), pp. 994–998.

[165] FJ Adamsen et al. "Method for using images from a color digital camera to estimate flower number". In: *Crop Science* 40.3 (2000), pp. 704–709.

[166] Fadzilah Siraj, Muhammad Ashraq Salahuddin, and Shahrul Azmi Mohd Yusof. "Digital image classification for malaysian blooming flower". In: *2010 Second International Conference on Computational Intelligence, Modelling and Simulation*. IEEE. 2010, pp. 33–38.

[167] Tzu-Hsiang Hsu, Chang-Hsing Lee, and Ling-Hwei Chen. "An interactive flower image recognition system". In: *Multimedia Tools and Applications* 53.1 (2011), pp. 53–73.

[168] Balvant V Biradar and Santosh P Shrikhande. "Flower detection and counting using morphological and segmentation technique". In: *Int. J. Comput. Sci. Inform. Technol* 6 (2015), pp. 2498–2501.

[169] Marco Seeland et al. "Description of flower colors for image based plant species classification". In: *Proceedings of the 22nd German Color Workshop (FWS). Zentrum für Bild-und Signalverarbeitung eV, Ilmenau, Germany.* 2016, pp. 145–1154.

[170] KR Thorp et al. "Lesquerella seed yield estimation using color image segmentation to track flowering dynamics in response to variable water and nitrogen management". In: *Industrial Crops and Products* 86 (2016), pp. 186–195.

[171] Yuanyuan Liu et al. "Flower classification via convolutional neural network". In: *2016 IEEE International Conference on Functional-Structural Plant Growth Modeling, Simulation, Visualization and Applications (FSPMA).* IEEE. 2016, pp. 110–116.

[172] Redmond R Shamshiri et al. "Research and development in agricultural robotics: A perspective of digital farming". In: *International Journal of Agricultural and Biological Engineering* (2018).

[173] Ya Xiong et al. "An autonomous strawberry-harvesting robot: Design, development, integration, and field evaluation". In: *Journal of Field Robotics* 37.2 (2020), pp. 202–224.

[174] Boaz Arad et al. "Development of a sweet pepper harvesting robot". In: *Journal of Field Robotics* (2020).

[175] Henry AM Williams et al. “Robotic kiwifruit harvesting using machine vision, convolutional neural networks, and robotic arms”. In: *biosystems engineering* 181 (2019), pp. 140–156.

[176] Christian Scholz, Maik Kohlbrecher, and Arno Ruckelshausen. “Camera-based selective weed control application module (“Precision Spraying App”) for the autonomous field robot platform BoniRob”. In: *International Conference of Agricultural Engineering, Zurich*. 2014.

[177] Timo Blender et al. “Managing a mobile agricultural robot swarm for a seeding task”. In: *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*. IEEE. 2016, pp. 6879–6886.

[178] David Ball et al. “Robotics for sustainable broad-acre agriculture”. In: *Field and service robotics*. Springer. 2015, pp. 439–453.

[179] Robert Bogue. “Robots poised to revolutionise agriculture”. In: *Industrial Robot: An International Journal* (2016).

[180] Weikuan Jia et al. “Apple harvesting robot under information technology: A review”. In: *International Journal of Advanced Robotic Systems* 17.3 (2020), p. 1729881420925310.

[181] Yang Yu et al. “Real-Time Visual Localization of the Picking Points for a Ridge-Planting Strawberry Harvesting Robot”. In: *IEEE Access* 8 (2020), pp. 116556–116568.

[182] Hiroaki Yaguchi et al. "Development of an autonomous tomato harvesting robot with rotational plucking gripper". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 652–657.

[183] Wang Lili et al. "Development of a tomato harvesting robot used in greenhouse". In: *International Journal of Agricultural and Biological Engineering* 10.4 (2017), pp. 140–149.

[184] Lars Grimstad and Pal Johan From. "A configuration-independent software architecture for modular robots". In: *2018 International Conference on Reconfigurable Mechanisms and Robots (ReMAR)*. IEEE. 2018, pp. 1–8.

[185] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[186] Martín Abadi et al. "Tensorflow: A system for large-scale machine learning". In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 2016, pp. 265–283.

[187] Yu Jiang et al. "DeepSeedling: deep convolutional network and Kalman filter for plant seedling detection and counting in the field". In: *Plant methods* 15.1 (2019), p. 141.

[188] GK Brown et al. "Estimates of mechanization effects on fresh blueberry quality". In: *Applied engineering in agriculture* 12.1 (1996), pp. 21–26.

[189] Fumiomi Takeda et al. "Assessment of the V45 blueberry harvester on rabbiteye blueberry and southern highbush blueberry pruned to V-shaped canopy". In: *HortTechnology* 18.1 (2008), pp. 130–138.

[190] Richard Clark Rider, RB Fridley, and M O'Brien. "Elastic behavior of a pseudo-fruit for determining bruise damage to fruit during mechanized handling". In: *Transactions of the ASAE* 16.2 (1973), pp. 241–0244.

[191] BR Tennes et al. "Self-contained impact detection device: calibration and accuracy". In: *Transactions of the ASAE* 31.6 (1988), pp. 1869–1874.

[192] HR Zapp et al. "Advanced instrumented sphere (IS) for impact measurements". In: *Transactions of the ASAE* 33.3 (1990), pp. 955–0960.

[193] B Herold et al. "A pressure measuring sphere for monitoring handling of fruit and vegetables". In: *Computers and electronics in agriculture* 15.1 (1996), pp. 73–88.

[194] Ivan Muller et al. "Wireless instrumented sphere for three-dimensional force sensing". In: *2009 IEEE Sensors Applications Symposium*. IEEE. 2009, pp. 153–157.

[195] Yull Heilordt Henao Roa, Fabiano Fruett, and Marcos David Ferreira. "Real time measurement system based on wireless instrumented sphere". In: *SpringerPlus* 2.1 (2013), p. 582.

[196] Pengcheng Yu et al. "Development of the Berry Impact Recording Device sensing system: Hardware design and calibration". In: *Computers and electronics in agriculture* 79.2 (2011), pp. 103–111.

[197] Rui Xu and Changying Li. "Development of the second generation berry impact recording device (BIRD II)". In: *Sensors* 15.2 (2015), pp. 3688–3705.