Development of a Ground-based Robot for High-Throughput Plant Phenotyping

Tim Mueller-Sim

CMU-RI-TR-17-46

Submitted in partial fulfillment of the requirements for the degree of Masters of Science in Robotics

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA, 15213

August 2017

Thesis Committee:
George Kantor
Srinivasa Narashiman
Humphrey Hu

Abstract

The effect of the genomic revolution has had a significant impact on plant breeding – in the past ten years the cost of gene sequencing has decreased by 5 orders of magnitude. Unfortunately detecting and quantifying the expression of a genotype under field conditions is still an expensive and laborious task. Field-based robotic phenotyping can help reduce this phenotyping bottleneck, and can provide vast quantities of data that plant scientists can use to map specific desirable and undesirable traits to genetic markers. These associations can in turn be used to rapidly accelerate the plant breeding process.

This thesis describes the development of a novel robot ground-based platform capable of autonomously navigating below the canopy of row crops such as sorghum or corn while carrying a modular array of contact and non-contact sensors for high-throughput plant phenotyping. It also presents the results from several deployments to *Sorghum bicolor* breeding plots at Clemson University in South Carolina, USA.

Acknowledgements

I'd like to first and foremost thank my parents and family for fostering my curiosity all these years, for staying behind me when I faltered, and for sacrificing so much to provide me with fantastic opportunities that they never had. To my mother, for keeping my easily distracted mind busy. To my father, for putting up with all my experiments in your shop, particularly the failures. And to everyone who offered their expert and candid advice during my time here.

Contents

List of Figures

Chapter 1

Introduction

There has been a significant focus on automation in the agricultural domain for the past several decades, primarily on the expensive and massive machines that plant, fertilize, and harvest commodity crops such as wheat and corn. Specialty crops such as strawberries, apples, and grapes have recently become the target of robotics researchers due to the continued increase in labor costs along with the decrease in labor availability, though harvesting these crops usually require fine manipulation tasks and more accurate crop segmentation, and are therefore more difficult tasks to automate.

While there has been increasing interest in smaller platforms for precision agriculture, the current cost of production and implementation unfortunately prevent these systems from becoming more than just a line in a heavy equipment manufacturer's R&D budget. Small, agile platforms are currently more useful for plant scientists and plant breeders, as their test and breeding fields tend to be orders of magnitude smaller than the large commercial tracts that dominate nearly 40% of land in the U.S. [2]. The unprecedented quantity and quality of data a small robotic platform can provide to the plant breeding process will also eventually impact to the commercial market as cultivars bred for specific regions, traits, and higher yield.

In this thesis, we present our work in designing and developing a robotic platform dubbed the Robotanist, capable of autonomously traversing bio-energy sorghum breeding plots while carrying up to 50 kilograms of contact and non-contact sensors in order to collect high resolution spatial, spectral, and physical information about the plants. We also present the development and implementation of several methods for localizing globally and locally within crop rows, and traversing them accordingly. We present data obtained at various breeding plots in South Carolina, USA, during the 2016 and 2017 growing seasons.

1.1 Motivation

1.1.1 Sorghum as a bio-energy crop

Sorghum, a grain crop similar to corn, is a highly diverse plant with over 40,000 accessions. This wide inter-accession variation offers tremendous potential for sorghum to be bred for specialized purposes, including grain, biofuel, animal fodder, and even alcohol for human consumption. This large breeding stock, coupled with the lack of breeding programs in the U.S. devoted to sorghum over the past 100 years, has led the U.S. Department of Energy

Figure 1.1: The Robotanist in front of a bio-energy sorghum panel in Clemson, SC.

Advanced Research Project Agency-Energy (ARPA-E) to fund the TERRA program [3]. This program is aimed at facilitating the improvement in genetic gains of bio-energy sorghum through the development and integration of high-throughput plant breeding technologies, and it is through this program that the research presented here is funded.

1.1.2 Plant Breeding

The traditional approach to plant breeding entails growing a large number of plant breeds with a wide variety of traits, selecting those plants that exhibit desirable characteristics, and interbreeding those successful varieties to produce new lines with a combination of these desirable traits. This is a continuous process, where after each successful breeding cycle the new plant breeds exhibit ever-increasing trait qualities.

Modern plant breeding is in many ways similar to the traditional breeding process in that it still requires significant genotypic variation, successful crop selection, and crossing, though it also makes use of the plant's genetic information to inform the breeding process. Scientists attempt to find correlation between specific markers within the plant genome with favorable traits, and screen a large set of germplasms prior to the next growing cycle in order to find those plants which have a higher likelihood of exhibiting desirable traits. This process of pre-selecting a germplasm based on the probability of it exhibiting a specific trait can reduce the size of field trials and the labor and costs associated with them, while increasing the genetic gain from each trial.

The effect of the genomic revolution has had a significant impact on plant breeding; in the past 10 years gene sequencing has seen a 5 order of magnitude decrease in cost [1], vastly increasing the quantity of genetic data available to plant scientists. Unfortunately the cost and speed of detecting and quantifying the expression of those genotypes under specific environmental conditions remains unchanged, and as a result the plant breeding pipeline

cannot take full advantage of this plethora of new information.

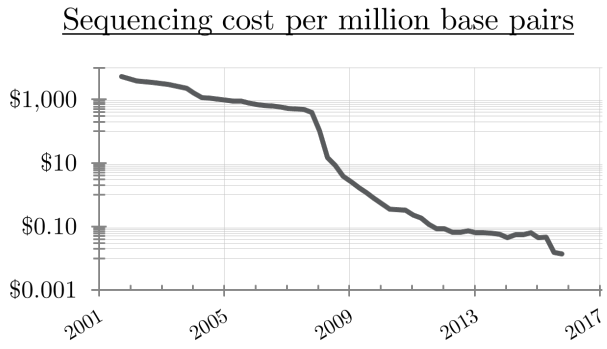


Figure 1.2: Gene sequencing costs from 2002 to 2016 [1]

1.1.3 Plant Phenotyping

In order to accurately inform a modern plant breeding program, scientists need to map the physiological and morphological expressions (phenotypes) of a plant genotype to specific genetic markers in the plant genome. If they can successfully establish these phenotype/genotype associations, they reduce the need for a large number of trials when breeding for desired traits. For example, in the case of breeding sorghum for use as a bio-energy crop, if one mapping was found that associated low lignin and cellulose to a specific set of genetic markers, and another mapping was found that associated relatively high biomass accumulation with another set of genetic markers, then plant scientists could significantly reduce the number of cycles of breeding and phenotyping required to produce high quality hybrids which exhibit both of those traits. Unfortunately, the rate at which phenotypes are measured and analyzed is significantly slower than the rate of plant genome research. This deficiency is well-recognized by the scientific community, which has deemed it the Phenotyping Bottleneck [4].

While there has been significant development of automated greenhouses which house conveyors and imaging sensors that automate the phenotyping process [5], research has shown that plants of the same genotype express phenotypic variation when grown in laboratory or greenhouse settings as opposed to the field [6]. This is due in part to the larger environmental variance and abiotic/biotic pressures present in the field when compared to controlled, indoor settings. Since the end goal of a typical plant breeding program is to develop seeds that will be used on farms throughout a specific region, it follows that breeding trials need to be performed in the field under expected environmental conditions in order to ensure success of the program.

Collecting phenotypes in the field is a labor intensive task, performed primarily by plant scientists and graduate students under hot, humid, and pest ridden environments using relatively rudimentary tools such as yardsticks and calipers. Under these conditions, the probability of incorrect data being collected due to human error increases considerably. Over the past decade, a wide variety of platforms have been developed to automate the extraction of phenotypic information from plants grown in the field, including blimps [7], UAVs [8], over-row tractor platforms [9][10], and fixed gantry systems [11].

All of these platforms offer varying levels of spatial, spectral, and temporal data. The aerial platforms have high rates of coverage, but can only carry a limited payload for relatively short periods of time, and lack the ability to penetrate the crop canopy. The over-row tractors offer a much higher payload capacity than the aerial platforms, but they have a lower rate of coverage and are limited by the height of the crop being inspected, and also generally require a human operator. The fixed gantry systems offer a moderate rate of coverage and a massive payload capacity, but they have been shown to be expensive to construct and operate, and are limited in the size of the breeding plots they can cover. Unfortunately none of these platforms have been able to provide data from beneath the canopy of a tall crop like bio-energy sorghum during the late season.

1.2 Related Work

1.2.1 High-Throughput Plant Phenotyping

LemnaTec GmbH is one of the industry leaders in automated plant phenotyping. They've developed several systems capable of continuously monitoring crops within greenhouses and laboratory settings and also in field environments. The *Greenhouse Scanalyzer* platform (figure 1.3a) consists of a conveyors belt system which transport potted plants through a modular set of non-contact sensors which can image the plant in the visible and Near-Infrared spectrum and can also perform chlorophyll fluorescence measurements. Modules available include watering and application spraying stations, weighing stations, barcode scanning stations, and multi-view imaging stations. LemnaTec GmbH provides software that can automatically extract phenotypes from the acquired data, including plant size, color, morphology, biomass, chlorophyll levels, and the effects of stress, disease, and pest pressures. Plant dimensions are currently limited to 2.5 m high by 1.0 m wide. A system installed at The University of Adelaide has been used to successfully investigate the effects of nitrogen and water deficiency on *Sorghum bicolor* [5]. While this system accomplishes automated high-throughput phenotyping on a large scale, it requires that plants be grown indoors under controlled conditions, which is not useful for a breeding program which develops cultivars for use in the field, as mentioned in section 1.1.3.

The LemnaTec GmbH *Field Scanalyzer* (figure 1.3b) is an outdoor phenotyping system that consists of a large sensor bay carried by a gantry system that can cover a breeding plot up to 10 m x 110 m. The sensor bay has a payload capacity of 500 kg, and includes a wide variety of non-contact sensors, including RGB, infrared, hyperspectral, and chlorophyll fluorescence imagers, along with a 3D laser scanner [12]. Provided software is capable of extracting plant phenotypes such as canopy height, leaf geometry, biomass, and photosynthetic efficiency. Two of these gantry systems are commercially deployed – one system is located at the University of Arizona's Maricopa Agricultural Center and another is located at Rothamsted Research in Hertfordshire, UK [11]. The system at Rothamsted Research has been used to automate the detection of the heading and flowering growth stages in 6 different wheat cultivars [13]. Unfortunately, the cost of this system is prohibitively expensive for smaller breeding programs, cannot measure below a closed canopy, and requires significant infrastructure that restricts the placement of breeding plots.

Researchers associated with the University of Arizona and the United States Department of Agriculture (USDA) developed a tractor-based platform for phenotyping Pima cotton (*Gossypium barbadense L.*) (see figure 1.4a). The system consists of sonar proximity sensors, GPS, infrared radiometers, and NIR cameras to measure canopy height, temperature and

(a) Greenhouse Scanalyzer System

(b) Field Scanalyzer System

Figure 1.3: LemnaTec GmbH Automated Plant Phenotyping Systems

reflectance [9]. However, the data collected by the system is restricted to overhead views, and a tractor's maximum boom height (in this case 1.93 m) would limit the height at which plants could be phenotyped. Texas A & M University is also developing an overhead phenotyping platform (see figure 1.4b), but the platform again does not resolve the inability of overhead systems to see into a closed canopy [10], and is also limited by the height of the crop. Neither of these systems are automated and therefore require a trained operator to function.

Aerial-based data collection is perhaps the most popular method of phenotyping. Aerial platforms are easy to deploy and can collect data over large distances in relatively short periods of time. However, they are limited by sensor resolution, payload, and flight time. A team at ETH Zurich overcame the constraints of sensor payload and flight time by deploying their system on a blimp, but they encountered difficulties measuring traits such as leaf greenness, senescence, and temperature due to the combination of high altitude and limited sensor resolution [7]. The same group at ETH Zurich developed a cable-suspended field phenotyping platform with a payload of 12 kg and a coverage of approximately 1 hectare (see figure 1.4c) [14]. This system requires significant infrastructure to implement, and restricts the placement of breeding trials to areas covered by the suspended-cable system.

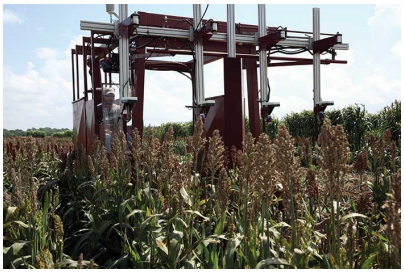
Sensing platforms on rotorcraft such as the DJI S1000 or Yamaha RMAX can yield higher resolution at lower altitudes and at significantly lower costs than permanent structures (see figure 1.4d, but flight times range from 8–120 minutes, depending on payload, and maximum payloads range from 0.8–8.0 kg [8]. Furthermore, sensors mounted on aerial vehicles are often unable to measure below a closed canopy.

1.2.2 Commercial UGVs in Agriculture

There has been extensive development of ground-based research platforms for applications such as precision pesticide spraying, soil sampling [15] and weeding [16], for a wide variety of field conditions and crops [17]. Several commercial ground-based robotic systems have also been investigated for their viability as a phenotyping platform.

Clearpath Robotics Inc. has developed a family of field tested all-terrain robots, including the Grizzly, the Husky, and the Jackal. These platforms have been used in a wide range of conditions, from mapping underground mines to mapping and navigating in dense forests [18]. Robotnik Automation S.L.L. has developed several robotic platforms that have been used to deploy sensors within an agricultural setting, including the Guardian and the

(a) University of Arizona tractor-based phenotyping platform



(b) Texas A&M tractor-based phenotyping platform

(c) ETH Zurich Field Phenotyping Platform

(d) Near Earth Autonomy drone with sensor payload

Figure 1.4: Various platforms for crop phenotyping

Summit XL [19]. QinetiQ North America sells the TALON, a small tracked robot primarily used by military and police organizations for explosive ordinance disposal and reconnaissance [20], and Omron Adept Technologies, Inc. offers the Seekur Jr., the Pioneer 3-AT, and a variety of other wheeled configurations of research robots [21]. Rowbot Systems has developed a platform [22] that is designed to travel between crop rows autonomously.

While a variety of ground-based robotic vehicles are currently available for purchase, none meet the specific functional, quality, and performance requirements of this project, nor would they offer the modularity that will enable this platform to be successfully utilized throughout the life of the project.

1.2.3 Perception, Localization, and Navigation

Localization and navigation within an agricultural setting has been the focus of significant prior research. One group from Carnegie Mellon University (CMU) used a monocular RGB camera to localize and navigate between rows of an apple orchard [23], while another group from the University of Illinois investigated the use of a variable field-of-view camera when navigating within a field of early growth corn [24]. The same group from Carnegie Mellon University has also investigated using a 3D point cloud captured using a LiDAR sensor to again localize and navigate within an apple orchard [25].

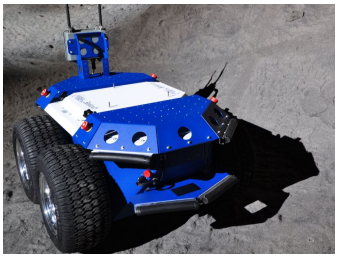
A group from Wageningen University investigated the use of an image based particle filter for navigating within maize. The team used a downward facing camera mounted on the tall mast of a small three wheeled robotic platform, and used a binary image indicating the presence of plant material determined by color thresholding in their measurement model

(a) Clearpath Husky

(b) Robotnik Sumit XL

(c) Robotnik Guardian

(d) QinetiQ TALON®



(e) Omron Adept Technologie, Ince. Seekur Jr.

(f) Rowbot Systems

Figure 1.5: Commercial robotic platforms

[26]. A group from the University of Hohenheim used a planar laser range finder in a push-broom configuration along with a RANSAC line fitting algorithm to successfully detect multiple crop rows that ranged in height from 0.15-0.65 m, in a greenhouse [27].

Most of this work was performed with monoculture crops under controlled conditions, which does not contain wide phenotypic variation inherent in sorghum breeding plots. This phenotypic variation, such as stalk height and leaf angle, causes significant visible clutter within rows (see Figure 1.6). Prior work also does not address reliable navigation from early season through to late season growth stages.

1.2.4 Conclusion

The state-of-the-art systems outlined above exhibit limitations ranging from payload capacity to geometric limitations to weather rating to cost. For this reason, we have developed our own custom intra-row autonomous mobile sensor platform - the Robotanist.

The Robotanist consists of a wheeled skid-steer, electrically powered ground vehicle that is capable of autonomously navigating sorghum rows and can travel at speeds up to 2 m/s for more than 8 hours per charge. The robot is equipped with LiDAR, RTK GPS, RGB cameras, inertial measurement units, and the computing power necessary to run perception and localization algorithms in real time. The system supports a custom manipulator capable of taking contact measurements from sorghum stalks and is capable of deploying a wide range of phenotyping sensors including a custom stereo camera synced with high power xenon/LED flash lamps and an upward facing 180° field of view camera.

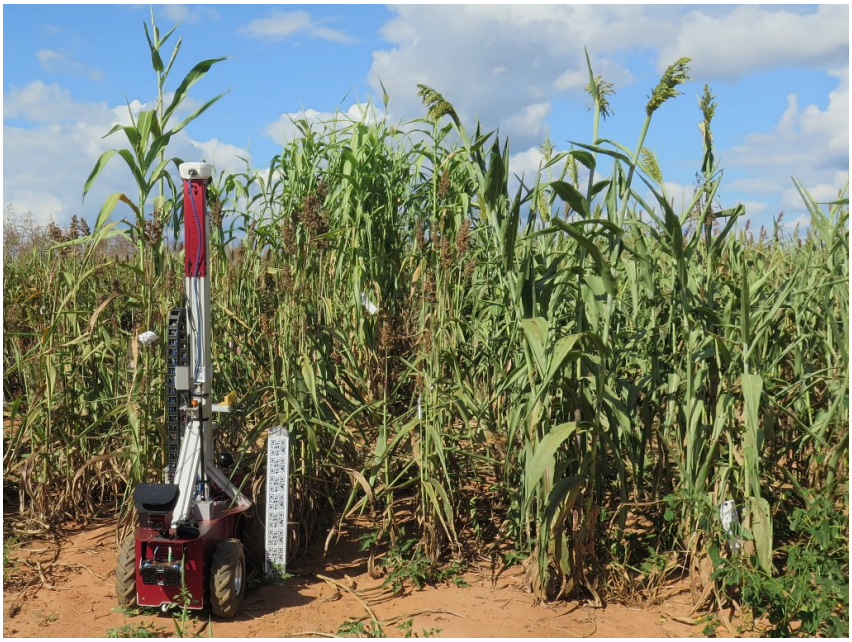


Figure 1.6: Clutter present within crop rows during late season

1.3 Objective and Approach

The objective of this research is to develop a robotic platform capable of autonomously traversing a typical sorghum breeding plot, planted at a row spacing of 30 inches or greater, from the initial planting through emergence of the seedling until late season immediately prior to harvest. The robot also needs to be able to carry a modular suite of contact and non-contact sensors, be protected from dust ingress and splashing water, and be able to operate from 5 °C to 45 °C

1.4 Overview

This thesis is organized as follows; in Chapter 2 we discuss the requirements set forth by the project and its objectives, along with the design of the robotic platform. In Chapter 3 we discuss various methods for localizing the robot, both globally and with respect to the crop rows, along with results obtained during various deployments to sorghum breeding plots in South Carolina during 2016 and 2017. Finally, Chapter 4 summarizes the results and contributions of this research, and also provides future directions for research in the field of agricultural robotics.

Chapter 2

Design of an Unmanned Ground Vehicle

Figure 2.1: Rendering of the Robotanist

Designing a robotic platform that can successfully traverse the narrow rows of a typical sorghum field with a high level of mobility while surviving the harsh environment requires significant development effort. In order to maximize our probability of designing a successful prototype we first outline the requirements set by the TERRA program and the plant breeders at Clemson University, and then we begin the design process for each subsystem while ensuring we adhere to those requirements.

2.1 Project Requirements

System requirements were driven primarily by the need to reliably traverse a typical breeding plot (1–2 hectares) within several days in order to avoid significant plant growth throughout a single measurement period. Functional requirements include the ability to autonomously transport contact and non-contact sensor payloads between rows of biofuel sorghum crop, navigate from one row to another autonomously, transmit robot state, sensor and diagnostic data to a nearby base station, and be tele-operable. Performance requirements include the ability to maintain ground speeds of 1.0 m/s or greater, operate from 5 °C to 45 °C, traverse 30,000 m^2 per day, carry a sensor payload up to 50 kg, reliably traverse the minimum 0.61 m row space available prior to harvest, wirelessly communicate with a base station up to 500 m away, localize within the

crop row with a nominal accuracy of <5 cm relative to a global coordinate frame, have a minimum turning radius of <2 m, and have an Ingress Protection (IP) rating of 54 or higher.

2.2 Design Constraints

2.2.1 Size Limitations

Sorghum breeding plots are typically planted at a row spacing of 30 inches (0.762 m) center to center. Our partners at Clemson University indicated that sorghum has a tendency to tiller, or have multiple stalks grow from the same root ball, and the largest expected radius within which stalks would emerge is 3 inches (7.5 cm). This provides us with an upper bound on the robot width of 24 inches (0.61 m), as shown in figure 2.2. Ideally, we want the robot to be as narrow as possible in order to allow for a higher localization error tolerance, but we also want to maximize the width in order to maximize the size of the base of support so as to increase stability and prevent rollover while also increasing our internal volume budget for electronic and power components. A final width of 22 inches (0.56 m) was determined as we were confident in our ability to maintain a crosstrack error of less than 1 inch (5 cm).

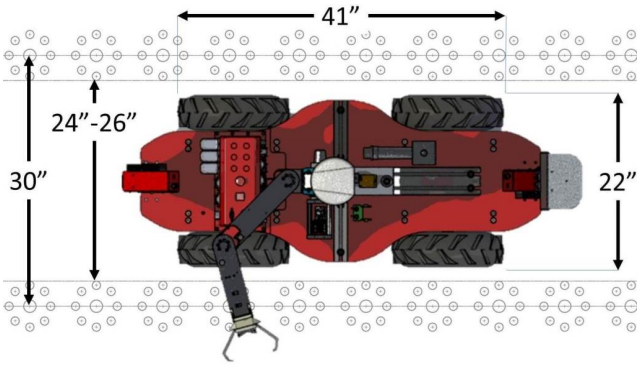


Figure 2.2: Sizing constraints on robotic platform

There were fewer constraints on the length of the robot, though our choice of drive configuration would have a significant impact on the decision process. Width constraints and added complexity would make an explicit steering configuration difficult to achieve. A skid-steer configuration was ultimately chosen, as it would simplify drivetrain design and simultaneously increase reliability.

As Apostolopoulos has shown in [28], as the length to width (L/W) ratio of a skid-steering robot increases, so does the required torque for achieving a constant turning radius. Required turning torque is very difficult to model, as it is a function of not only the geometry of the wheel but also the condition of the surface it is interacting with. The field of terramechanics is devoted to studying this interesting problem. Any effort to reduce the L/W ratio has high returns in terms of reduced drive motor torque requirements and reduced turning radius. In order to avoid a lengthy study of the expected soil type that the robot would encounter, we investigated the L/W ratios of various off-terrain wheeled robots in order to determine a rough upper limit on the L/W ratio. Several skid-steer robotic platforms of similar sizes designed at the National Robotic Engineering Center (NREC)

were investigated, including Dragon Runner and Mini-Crusher, and their L/W ratios were determined. A L/W ratio of approximately 1.5 was determined to be the upper limit for our application.

Constraints on the height of the robot are driven by the need to place the GPS antenna as high as possible above the ground. As shown in figure 2.3, the height of sorghum can vary significantly from emergence until anthesis and subsequent harvest, reaching heights of up to 20 feet. Due to significant moments generated by a 20 foot tall mast and the effect it would have on raising the center of mass of the robot, the height of the mast must be limited.

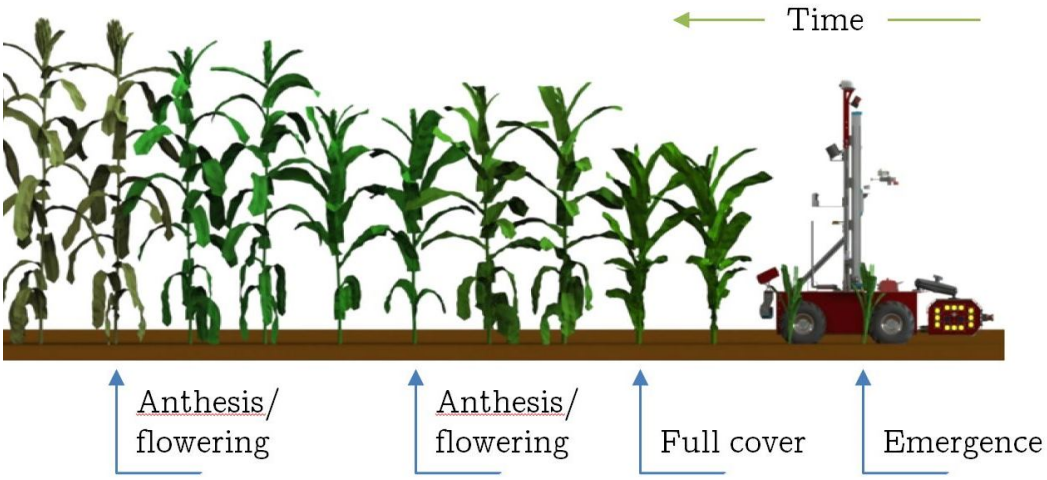


Figure 2.3: Stages of plant growth, from emergence through to maturity

2.2.2 Drive Motor Sizing

In order to determine the proper torque requirements for the drive motors, we must first determine the upper bounds on required torque for traversing a typical farm field. The lower bound for motor torque requirements involves driving forward over flat, smooth terrain, while the upper bound in our case is dictated by the torque requirements to overcome a step obstacle. We want enough motor torque to be able to overcome the highest step obstacle that wouldn't result in high centering, or roughly 5 inches.

In order to determine the lower limit on our torque requirements, we calculated the rolling resistance and acceleration required under the expected worst case soil conditions, as outlined in [29]. Rolling resistance is the horizontal force needed to compact soil as the wheel rolls over it. Using rolling resistance coefficients provided in [29], we estimate the expected torque required to overcome rolling resistance (T) under worst case expected conditions, then add that force to the acceleration force required to start the robot from a stop to a desired speed over a certain time interval (F_a). Acceleration is assumed to be a step function with a constant value. The rolling resistance is calculated as shown in equation 2.1 below. Using these calculations, we can approximate the current draw from motors while traversing

the breeding plots over flat terrain. We assume a rigid wheel model.

$$Q = \mu_T \cdot r \cdot W \tag{2.1}$$

where:

$Q \rightarrow$ wheel torque (Nm)

$\mu_T \rightarrow$ thrust coefficient

$r \rightarrow$ wheel radius (m)

$W \rightarrow$ wheel load (kN)

Next, we calculate the torque required to overcome a step obstacle, Q_s, shown in equation 2.2. Here we assume the center of gravity of the platform to be at the geometric center and quasi-static conditions. We also assume that maximum tractive torque will increase as the wheel contacts hard rock, and that only one wheel will be applying torque to the rock.

$$Q_s = \frac{r \cdot mass_{robot} \cdot wheelbase/2}{wheelbase + \sqrt{r^2 - (r - stepsize)^2}} \tag{2.2}$$

We can now approximate the upper bound on motor torque required for each wheel on our platform as follows:

$$Q_{max} = \frac{\frac{3}{4} \cdot Q + Q_s + F_{accel}}{4} \tag{2.3}$$

2.2.3 Energy Requirements

Battery sizing is driven by the power draw of all sub-systems multiplied by the desired run time of the robot before recharging is needed. Under ideal conditions, we want the robot to be able to operate for more than 8 hours at a time.

Approximating the torque required to overcome rolling resistance under ideal conditions, we can approximate the current draw the motors will pull, and multiply that by our expected operating duty cycle. Knowing this, it's a simple matter of approximating losses in any onboard voltage regulators required to power each subsystem. Our calculations indicated we'd need at least 2,125 Wh of energy to be able to run the platform and its subsystems for an entire day.

When selecting the ideal battery chemistry for our application, we have to account for more than just the specific energy it can provide. Properties like the price per unit energy, discharge cycles, and safety are also important to consider. While not as energy dense as LiPo chemistry, $LiFePO_4$ offers significant increases in battery life and safety, while still offering a better energy density than a lead-acid chemistry.

2.2.4 Computational Requirements

On-board computing is necessary to run segmentation, localization, and navigation algorithms in real-time in order to allow for autonomous operation of the robotic platform. In addition to the base platform, the attached manipulator runs computer vision algorithms in order to detect and servo to individual stalks. The non-contact sensors also require a

computer to store images and other data for offline processing. Isolating each system on it's own computer would be ideal in order to prevent conflicts during runtime from conflicting ROS nodes or saturated processors. Therefore, three computers are required in order to minimize any effects that independently developed code would have on another groups algorithms.

2.3 System Overview

2.3.1 Structural

The final dimensions of the mobile platform is 1.34 m in length, 0.56 m in width, and 1.83 m in height with the sensor mast (0.43 m without the sensor mast). The total mass of the robot is approximately 140 kg including the mast and the manipulator. The chassis of the robot is comprised of 21 separate pieces of 5052-H32 aluminum sheet metal, cut and formed to shape using CNC processes and connected using rivets and bolts. By designing the chassis out of formed sheet metal, a usable volume of 103 L for electrical hardware and wiring was obtained within the frame of the robot with a total enclosed volume of 111 L. Structural members were strategically placed within the frame of the robot to not only minimize compliance under expected loading but to also provide convenient connections for hardware mounting plates. The side panels were designed to act as shear walls, further reducing the compliance of the chassis under the majority of load cases, while simultaneously shielding the delicate hardware inside from damaging debris and moisture present on typical farms. Louvers fitted with air filters were placed on the side, front, and rear panels to act as inlets and outlets for thermostat-controlled cooling fans placed inside the chassis. The structural framework during assembly is shown in Fig. 2.4.

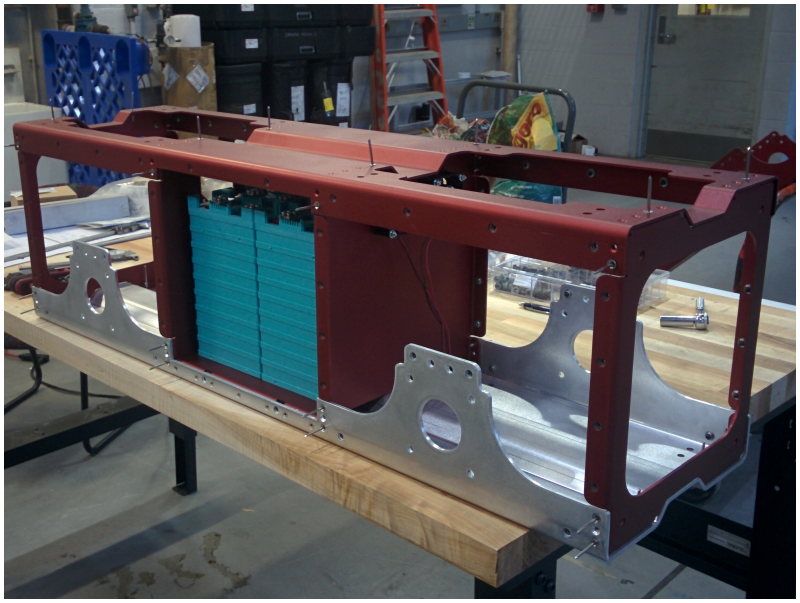


Figure 2.4: Structural framework during assembly

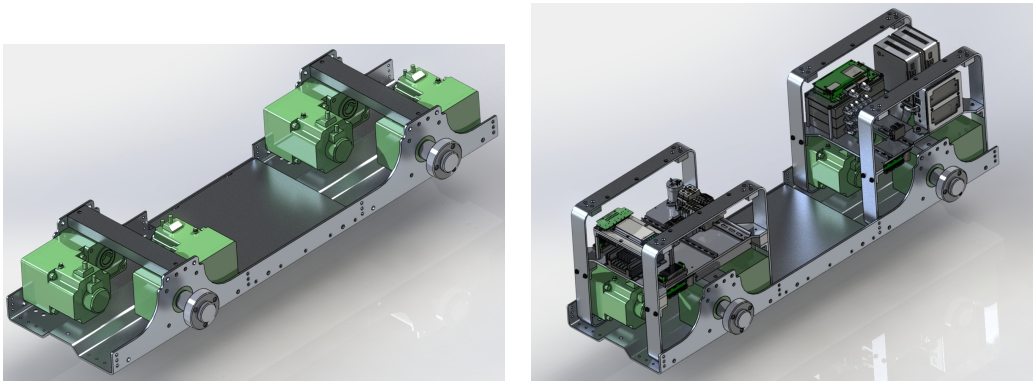
2.3.2 Drivetrain

Each wheel is independently driven by a 200 W brushless DC motor (BLVM620K-GFS) connected to a 50:1 hollow shaft gearhead (GFS6G50FR), both sourced from Oriental Motor USA Corp, and are capable of outputting a combined torque of 108 Nm and driving the robot at speeds up to 2.0 m/s. The BLDC motors are in turn controlled by custom–built HEBI Robotics motor drivers which expose data from the BLDC motors, including position data from the Hall effect sensors and current draw, through a C++ API. Motor velocity commands are sent to the HEBI motor controllers via Ethernet, which are then achieved through a PID controller on the motor driver. Attached directly to the output shaft of the gearbox via a custom driveshaft are 6 inch diameter Douglas Classic spun aluminum wheels, and the tires are Carlisle Super Lug Lawn & Garden (14X4.50–6), with a nominal diameter of 14.5 inches. Figure 2.5a shows a CAD rendering of the drivetrain, where the green components are the motor and gearbox.

2.3.3 Electrical

Battery

The energy requirements of the mobile platform were calculated using the sum of average predicted current draw of each subsystem. Along with the power requirements of individual subsystems, a conservative estimate of the rolling resistance of the wheeled robot was needed to determine the average current draw of the drive motors. This was calculated as described in 2.2.2 to be roughly 51.8 N with respect to the expected soil conditions. A battery capacity of at least 2,125 Wh was determined to be required in order for the Robotanist to be able to perform phenotyping for up to 8 hours continuously, therefore a 24V, 100 Ah $LiFePO_4$ battery pack was chosen for the power source. The battery chemistry offers a higher specific energy and volumetric energy density than a typical lead-acid battery while also offering a longer recharge cycle life and slower rate of capacity loss when compared to Lithium-ion battery chemistries.



(a) Drivetrain (b) Drivetrain and electrical components

Figure 2.5: Rendering of drivetrain and electronic mounting plates

Computing

Computing is handled using three Intel NUC Mini-PCs (NUC5i7RYH), each with a 3.4 GHz Intel i7 processor, 16 GB RAM, and a 1 TB SSD storage drive. Each computer is connected to a bank of unmanaged Gigabit network switches to provide a means of communication between themselves, various sensors, and radios. Due to its reliability and robustness, communications between various electronic components within the system are provided primarily with Ethernet via Cat5 cabling. Sensor selection opted for Ethernet variants where possible.

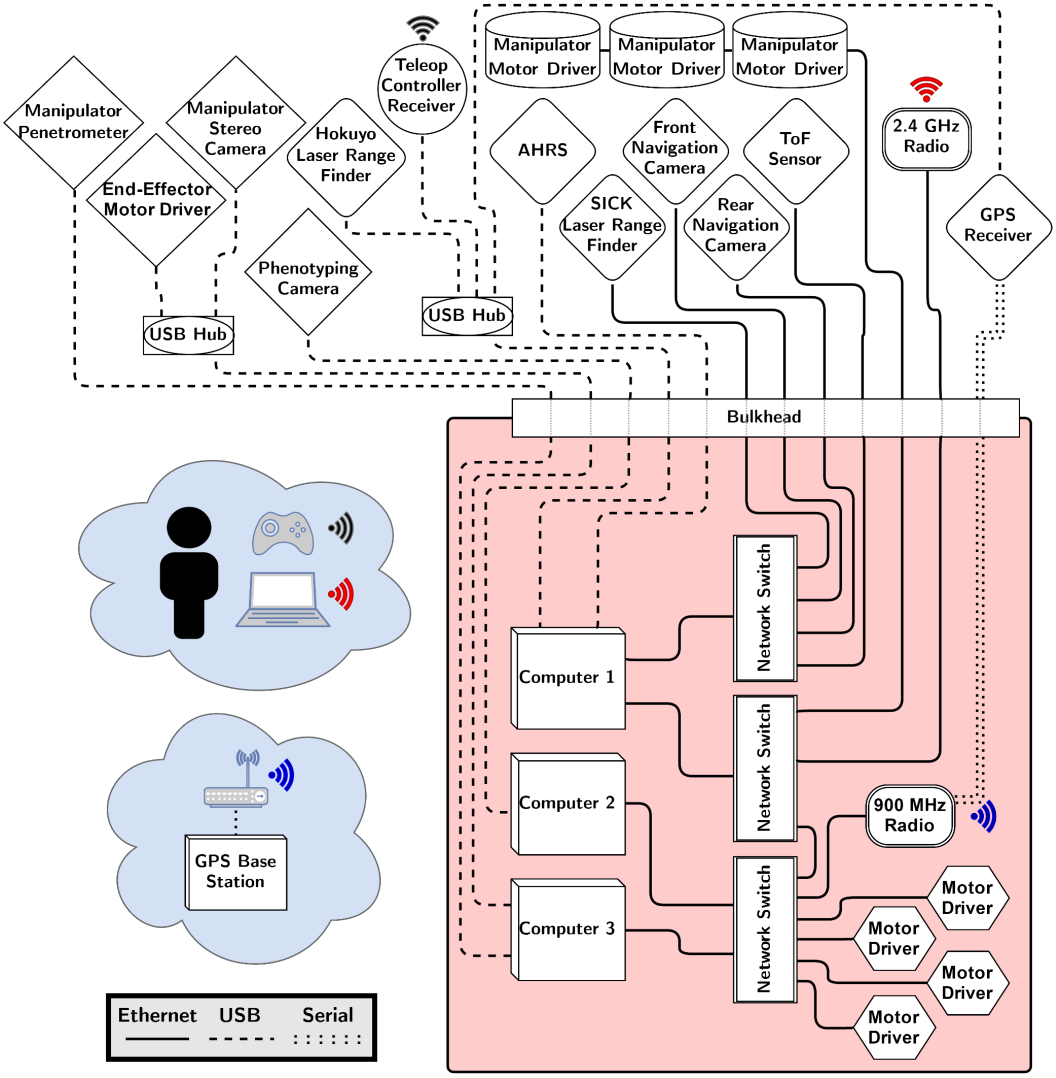


Figure 2.6: System diagram of the Robotanist

Voltage Regulation

Four DC-DC converters are utilized to provide regulated power to various subsystems. A Vicor VIPAC Array DC-DC converter (VA-D3056639) with 500 W, 12V output provides the majority of regulated power, while subsystems which require 5V and 24V regulated voltage are powered by two CUI Inc. DC-DC converters (PQA50-T). Finally, a 200 W, 12V CUI Inc. DC-DC converter (VHK200W-Q24-S12) is used to provide power solely to the phenotyping sensor payload. All four DC-DC converters can be controlled via switches present on the bulkhead near the rear of the robot.

Wireless Communication

Wireless communications are provided through the use of two onboard radios. A 900 MHz radio (Freewave FGR2–P) provides low bandwidth and long range wireless Ethernet and serial data communications with a fixed radio (Freewave FGR2–PE) connected to the GPS RTK base station. RTK correction signals are broadcast over the serial data lines, and the Ethernet ports are used for accessing diagnostic information from the robot if necessary. A 2.4 GHz wireless access point (Ubiquiti Bullet M2 Titanium) is used for short range, high bandwidth communications with the robot.

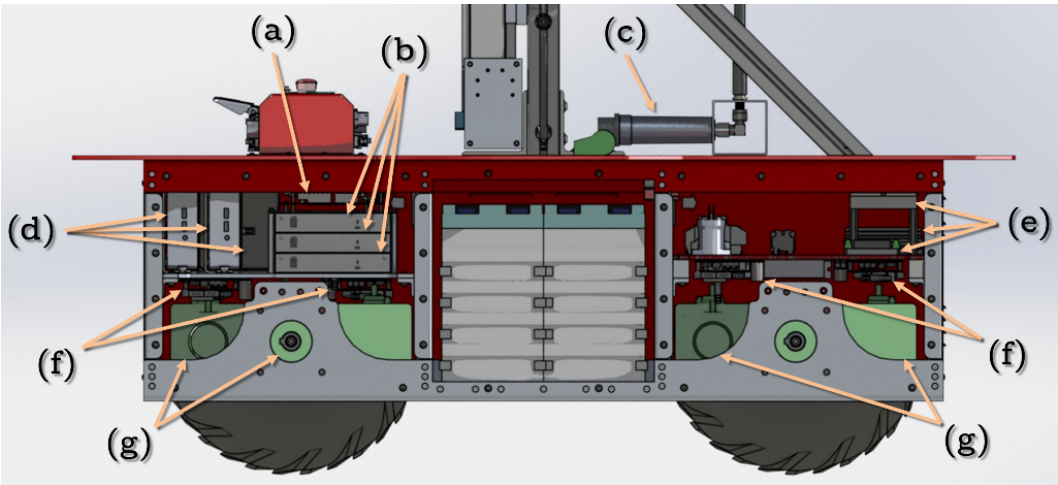


Figure 2.7: Diagram of interior electrical components: (a) Freewave 900 MHz radio; (b) 3x 8 port Gigabit network switch; (c) Ubiquiti Networks BM2-Ti 2.4 GHz Radio ; (d) 3x Intel NUC Mini PC; (e) 4x DC-DC converters; (f) 4x Hebi BLDC motor drivers; (g) 4x Oriental Motors BLVM620K-GFS and GFS6G50FR BLDC gearmotors; (k) 24V, 100 Ah $LiFePO_4$ Battery

2.3.4 Navigation Sensors

Reliably localizing, detecting obstacles, and navigating within the highly occluded and dynamic environment that exists in a typical sorghum breeding plot is a difficult task, particularly as the plant height begins to exceed the height of the GPS antenna. To attempt to

resolve this issue, a suite of perception sensors were selected that would enable modularity and provide a broad range of sensor data.

Navigation sensors are shown in Fig. 2.8. The design of the base is such that all of the navigation sensors are configurable; there are multiple mounting points for the planar LiDARs, cameras, and the time-of flight sensor on the front and rear of the robot, and the pitch angle of the camera mount is rapidly configurable to 15° increments (see figure 2.9).

A Novatel SMART6-L GPS antenna/receiver is mounted at the top of the mast of the robot so as to provide a clear view of the GPS constellation. RTK correction signals are broadcast to the Novatel receiver from a base station set up approximately 2.2 km away, allowing for a nominal horizontal accuracy of 0.010 m + 1 parts-per-million (RMS) relative to the calculated position of the base station. It provides time stamped GPS positioning data to all networked computers.

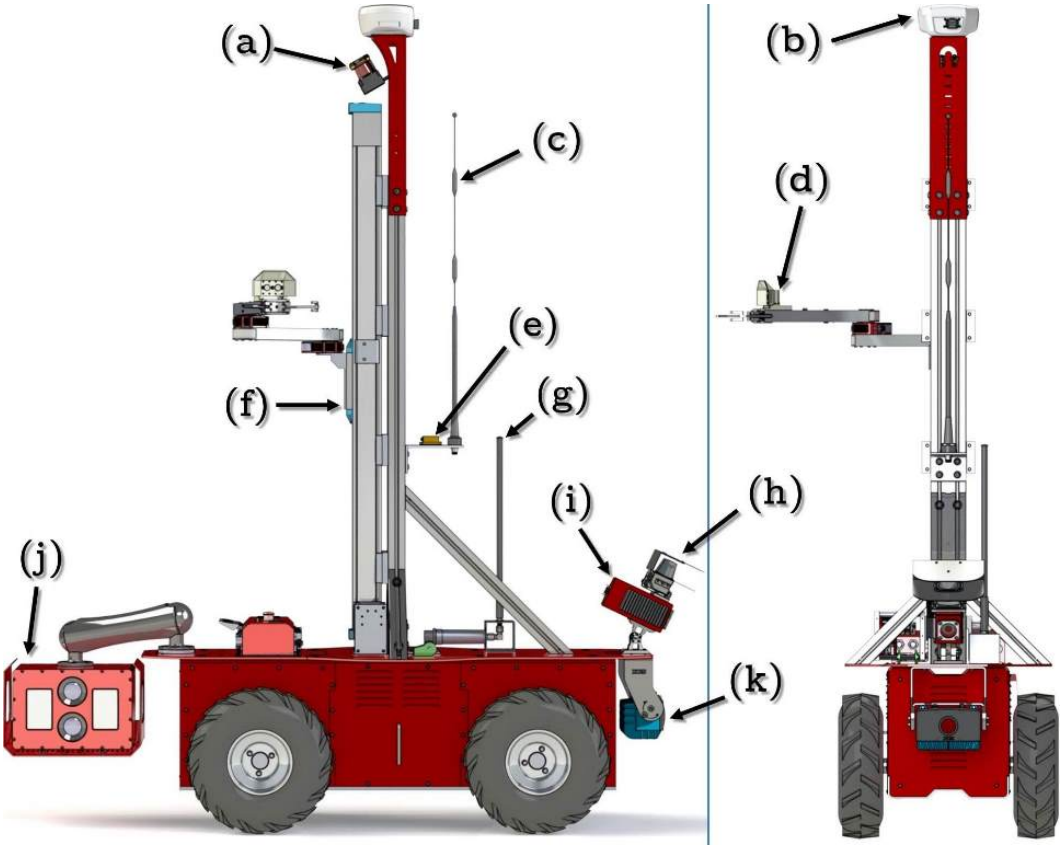


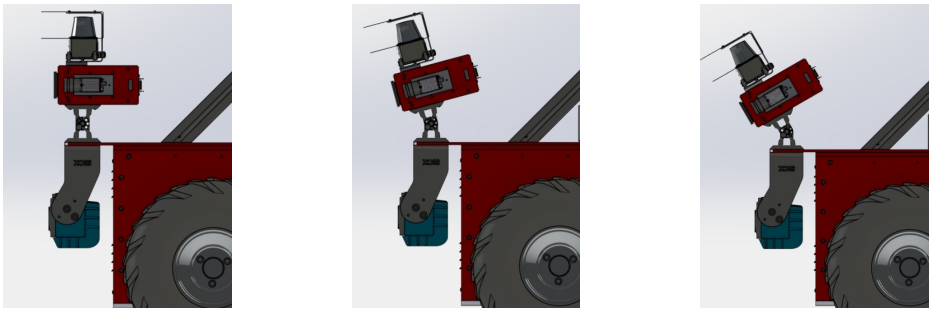
Figure 2.8: Rendering of a side view of the Robotanist: (a) Hokuyo UTM-30LX scanning laser range finder (LRF); (b) Novatel SMART6-L GPS Antenna; (c) 900 MHz radio antenna; (d) robotic manipulator; (e) Xsens MTi-30 Attitude, Heading, and Reference System (AHRS); (f) linear stage for mast sensor payload; (g) 2.4 GHz radio antenna; (h) SICK TiM561 scanning LRF; (i) housing for IDS UI-5240CP RGB camera; (j) custom stereo camera with active lighting; (k) SICK 3vistor-T time of flight camera.

Two planar LiDARs, the Hokuyo UTM-30LX and the SICK TiM561, are both mounted

in a pushbroom orientation in opposing directions of travel. The former is mounted at the top of the mast and the latter is mounted low on the frame so as to be robust to the changing canopy height as the growing season progresses. The UTM-30LX has a range of 30 m with an angular resolution of 0.25°, and a range accuracy of ±30 mm from 0.1-10 m and ±50 mm from 10-30 m. The TiM561 has a range of 10 m with an angular resolution of 0.33° and a range accuracy of ±60 mm.

A SICK 3vistor-T time-of-flight sensor is mounted in the front of the robot roughly 0.28 m above the ground. The sensor provides range, intensity, and confidence data which can be used for row following and to detect fallen stalks and other obstacles in the row beneath the canopy. The 3vistor-T has a nominal range of 7.2 m with a 69° x 56° field of fiew. The image size provided by the sensor is 176 x 144 pixels, with a sampling rate of 30 Hz.

Two IDS UI-5240CP-C-HQ cameras with Kowa LM5NCL fixed focal length (4.5mm) lenses are also fixed to the front and rear of the robot. The rear camera is removed as needed and replaced with a RAM®ball mount which is then used to attach various phenotyping sensors. This camera and lens configuration provides a field of view of 74°x62° at a pixel resolution of 1280x1024, with a sampling rate up to 30 Hz. Examples of images taken at various stages in the field are shown in figure 2.11.



(a) 0° inclination (b) 15° inclination (c) 30° inclination

Figure 2.9: Camera and LiDAR configurations

An Xsens MTi-30 Attitude and Heading Reference System (AHRS) is mounted midway up the mast and provides attitude information along with raw sensor data. The AHRS uses an Extended Kalman Filter (EKF) to fuse the inertial data from the tri-axial accelerometer and gyroscope and the tri-axial magnetometer data into an estimate of the 3D orientation of the sensor with respect to an Earth fixed coordinate frame. Data provided by Xsens states the sensor has a dynamic roll/pitch accuracy of ±0.4° and a nominal yaw accuracy of ±1.0° using the onboard EKF, and has a gyroscope drift of ±18° per hour. It can also output the raw data data from the accelerometer, gyroscope, and magnetometer for offboard sensor fusion.

Navigation Sensor Data Products

The onboard navigation sensors provide a large volume of data that can be used for localizing within the crop. Figures 2.10, 2.11, 2.12, and 2.13 show the various data products available from the onboard navigation sensors.

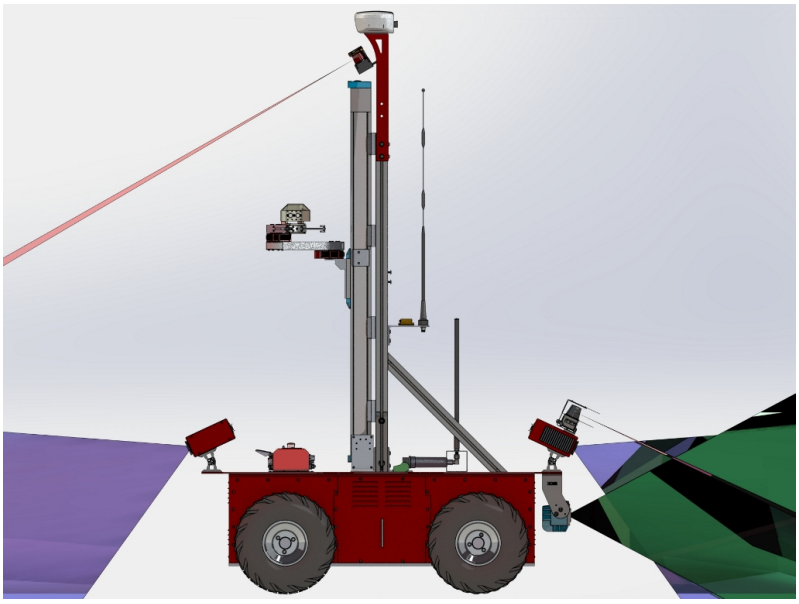


Figure 2.10: Rendering of the Robotanist showing the fields of view of various sensors

2.3.5 Software

Robot Operating System (ROS) and a combination of custom built nodes and open source packages are used for the software framework. A collection of nodes was written or sourced to allow for the operation of the robot, including nodes to communicate with sensor hardware and pass data from them into a ROS message, to communicate with the base platform remotely to query data and send commands, to perform coordinate frame transform calculations for individual sensors, to fuse sensor data and perform state estimation, to localize within and without crop rows, and to visualize sensor data.

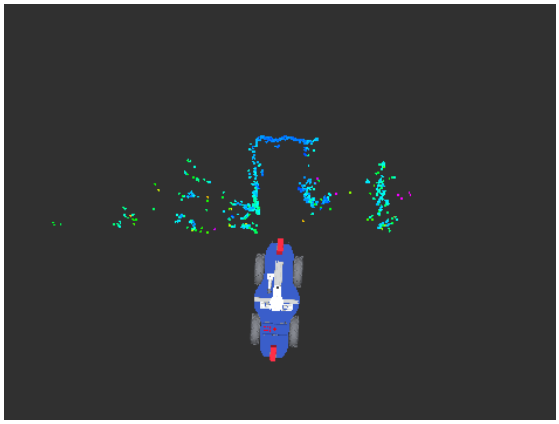
(a) 60 days after planting

(b) 90 days after planting

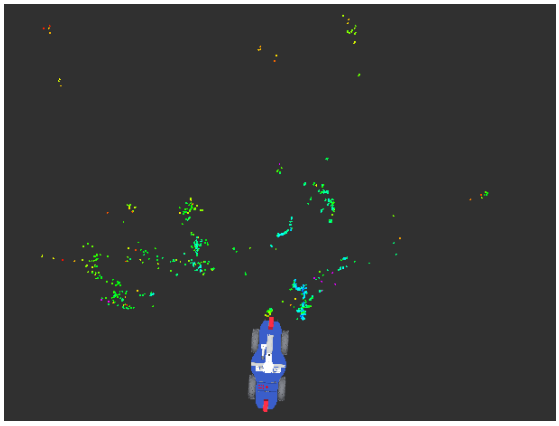
(c) 120 days after planting

Figure 2.11: Image data from front facing navigation camera at various points in the season

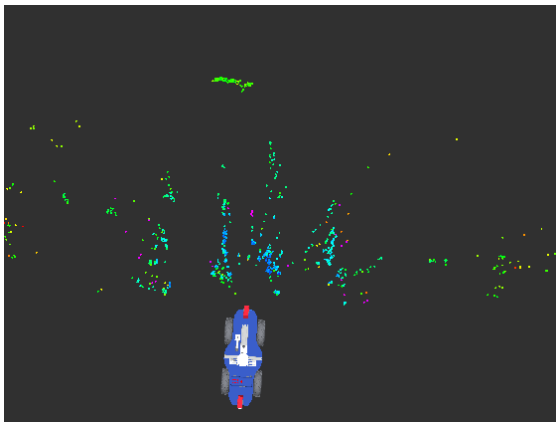
Figure 2.12: Image from 120 days after planting showing obstruction encountered while navigating within the crop rows. The Robotanist was able to drive over the blockage.



(a) Early season at 30° inclination



(b) Early season at 0° inclination



(c) Early season at 15° inclination

Figure 2.13: LiDAR data taken at various points in the season

Chapter 3

State Estimation, Localization, and Navigation

Reliably traversing a typical sorghum field can be a difficult task, even for a human on foot. At some points in a particularly dense row you can no longer rely on your eyes to show you the path is clear, you instead have to push through the foliage under the assumption that the rows are locally planted straight. The right placement of sensors can see below or above the canopy and overcome this problem. Unfortunately the fields in which we navigate are constantly growing and changing – we need a variety of navigation algorithms to choose from using different sensor modalities in order to be robust to this changing environment. In this chapter we describe the development and implementation of a method of accurately estimating the robot state along with several techniques to navigate either using previous waypoints or by detecting the crop rows themselves using a time of flight sensor.

3.1 State Estimation Using an Extended Kalman Filter

The Extended Kalman Filter (EKF) is ubiquitous in the field of robotics due to its ability to predict future states, approximate non-observable states, and filter noisy sensor data reliably and quickly. It is also relatively simple to implement and computationally efficient when compared with other probabilistic state estimation techniques; each update requires time $O(k^{2.4} + n^2)$, where n is the dimension of the state vector and k is the dimension of the measurement vector [30].

The states of an actual robot operating in the real world, such as its position and orientation, can never be perfectly known. Instead, it is more useful to think of the robot as existing in a distribution of likely states. The Kalman filter models these states as using a Gaussian distribution where state uncertainty arises in part due to process noise, measurement noise, and inaccuracies in the predicted dynamics of the system, to name a few.

Using data from a suite of sensors on our robotic platform, along with a model of the robot kinematics we can accurately predict the pose of the robot over time using a "fusion" of all available sensor and input commands, and can predict a final pose with more accuracy and certainty than would have been possible using any individual sensor.

3.1.1 Sensor Data

While wheel odometry is a good measure of linear velocity, the very nature of the skid-steer platform requires wheel slip when turning, beyond what is typically encountered when operating on loose soil. When a wheel slips during a turn, encoder data from that wheel is no longer a good indication of the actual movement of the robot, as we lose the ability to accurately relate the rotational velocity of the wheel with the linear and angular velocity of the robot induced by that wheel. As a result, localizing using wheel encoders is only useful over short intervals, as they continuously drift over time without correction.

The GPS receiver on the other hand provides a globally accurate position of the antenna on the robot, but is prone to signal errors or drop due to multi-path signal reception or loss of signal from the GPS constellation. It also cannot provide orientation data of the robot without the use of two expensive GPS units. GPS errors tend to manifest as discrete jumps in position that would be impossible due to the dynamics of the robotic platform.

The onboard inertial measurement unit provides three pieces of very important information regarding the state of the robot in its environment at a very high update rate (>100 Hz), unlike the data often provided by the GPS. The accelerometer data provides a globally referenced yaw and pitch, and its data can be integrated to obtain an estimate of the robot velocity in the local frame, which can be integrated again to estimate the relative position of the robot, though significant noise limits the usefulness of dead reckoning solely using an accelerometer. The gyroscope can be used to maintain a high level of confidence of the roll, pitch, and yaw of the robot in the short term, but is unfortunately susceptible to drift over longer periods of time. The magnetometer can be used to prevent the drift of the gyroscope from affecting the orientation estimate of the robot, and it can also be used to coarsely estimate the orientation of the robot within the global reference frame.

Each of these sensors provide us with varying levels of confidence for different parts of our final pose estimate. We can fuse this sensor data together using a Kalman Filter to provide a more accurate estimate of the state of the robot than can be obtained by any sensor individually, utilizing the most confident readings for each degree of freedom that all sensors provide. Using an Extended Kalman Filter to fuse the inputs of the robot with the measurements from available sensors, we can quickly find a best possible estimate of the state of the robot, with the important assumption that the linearization of the non-linear system via a first-order Taylor series expansion is sufficiently accurate.

3.1.2 Kinematic Equations of a Skid-steer Robot

Very accurate wheel position data is provided by the Hall encoders present on each drive motor at a high update rate (>50 Hz), which can be used to track the position of the robot in the short term, though uncorrected drift in position and yaw of the platform accumulates over time. Using the kinematics of a skid-steer robot, we can convert this wheel position data into movement of the robot relative to the base coordinate frame.

Since the left and right wheels of the robot are not mechanically linked, we first average the position that the wheels on each side has moved (Δx_{lf}, Δx_{lr}, Δx_{rf}, Δx_{rr} in figure 3.1) in order to reduce the kinematics to that of a differential robot. Unfortunately the change in yaw, $\Delta\psi_b$, is highly inaccurate due to significant wheel slip when turning. One proposed solution is to weigh that change in yaw and yaw rate according to the current instantaneous center of rotation [31]. In our instance, we applied a constant multiplier to the yaw rate, λ, determined according to trials performed in the field. The eventual value obtained for λ was 0.255.

Once we have the change in the linear position of each side of the robot (x_l and x_r), we can calculate the change in position and heading of the base in the robot coordinate frame as shown in equations 3.1, 3.2, and 3.3. Due to kinematic constraints the robot cannot use its actuators to translate along the y dimension, therefore Δy_b will always be zero. Using the change in time between wheel position measurements Δt we can approximate the velocity and yaw rate of the robot over that interval, as shown in equations 3.4 and 3.5.

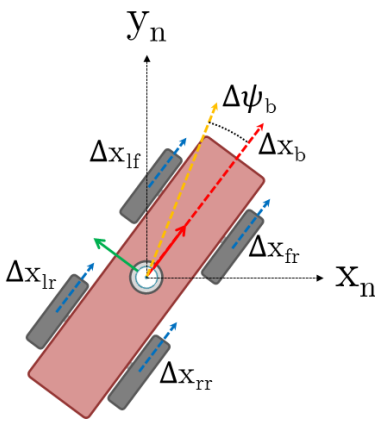


Figure 3.1: Deriving the kinematic equations of a skid-steer robot

$$\Delta x_b = \frac{\Delta x_l + \Delta x_r}{2} \tag{3.1}$$

$$\Delta y_b = 0 \tag{3.2}$$

$$\Delta \psi_b = \frac{\Delta x_l - \Delta x_r}{2} \cdot \lambda \tag{3.3}$$

$$v_b = \frac{\Delta x_b}{\Delta t} = \frac{\Delta x_l + \Delta x_r}{2 \cdot \Delta t} \tag{3.4}$$

$$\omega_b = \frac{\Delta \psi_b}{\Delta t} = \frac{\Delta x_r - \Delta x_l}{2 \cdot \Delta t} \cdot \lambda \tag{3.5}$$

We now need to perform a coordinate transform of the robot pose from the robot frame to the navigation frame (x_n, y_n, and ψ_n). We do this by moving the robot forward along an arc described by the forward velocity and yaw rate in the robot frame. This is shown in equations 3.6, 3.7, and 3.8, where the ratio $\frac{v_b}{\omega_b}$ is the instantaneous radius of path curvature. This formulation becomes computationally unstable as the yaw rate approaches 0, therefore in our implementation we perform a check to use different equations of motion if the yaw rate is below some ϵ, shown in equations 3.9, 3.10, and 3.11. This position is updated every time a message containing position data is received from the motor controllers, and the resultant odometry information is passed to the EKF node.

$$x_{n(i)} = x_{n(i-1)} + \frac{v_b}{\omega_b} \cdot (\sin(\Delta\psi_b - \psi_{n(i)}) - \sin(\psi_{n(i)})) \tag{3.6}$$

$$y_{n(i)} = y_{n(i-1)} + \frac{v_b}{\omega_b} \cdot (\cos(\Delta\psi_b - \psi_{n(i)}) - \cos(\psi_{n(i)})) \tag{3.7}$$

$$\psi_{n(i)} = \psi_{n(i-1)} + \Delta\psi_b \tag{3.8}$$

$$x_{n(i)} = x_{n(i-1)} + \Delta x_b \cdot \cos(\psi) \tag{3.9}$$

$$y_{n(i)} = y_{n(i-1)} + \Delta x_b \cdot \sin(\psi) \tag{3.10}$$

$$\psi_{n(i)} = \psi_{n(i-1)} + \Delta\psi_b \tag{3.11}$$

3.1.3 Extended Kalman Filter

Prior to discussing the EKF, we will first briefly discuss the Kalman filter, as the extended version uses linear Taylor expansions to linearize non-linear dynamics and distributions. The

Kalman filter is a basic Gaussian filter algorithm for a small subset of problems that can be adequately modeled with linear dynamics and measurement functions. For each point in time, the belief of the state of the system is represented by a mean and a covariance. We consider the following system, where $F(k)$ encodes the dynamics of the system, $x(k)$ describes the full system state, $G(k)$ describes how the inputs drive the system dynamics, $u(k)$ is the system input, $y(k)$ is the system output, $H(k)$ describes how state vectors are mapped into outputs, and $w(k)$ and $v(k)$ are zero-mean Gaussian measurement and process noise vectors, respectively.

$$\begin{aligned} x(k+1) &= F(k) \cdot x(k) + G(k) \cdot u(k) + v(k) \\ y(k) &= H(K) \cdot x(k) + w(k) \end{aligned}$$

Process noise accounts for un-modeled disturbances that affect system dynamics, such as slipping wheels or friction. In the above equations, we have known input $u(k)$ and known output $y(K)$, and given these the objective of the Kalman filter is to determine a best estimate of the state at the next time step given a previous estimate of the state. The derivation of the following summary of the Kalman filter is left to the reader, and can be found in [32]. In the following equations, $P(k)$ is an estimate of the error covariance, and $W(k)$ is the covariance matrix of the sensor noise.

$$\begin{aligned} \textbf{Prediction:} \quad \hat{x}(k+1|k) &= F(k)\hat{x}(k|k) + G(k)u(k) \\ P(k+1|k) &= F(k)P(k|k)F(k)^T + V(k) \\ \textbf{Update:} \quad \hat{x}(k+1|k+1) &= \hat{x}(k+1|k) + Rv \\ P(k+1|k+1) &= P(k+1|k) - RH(k+1)P(k+1|k) \\ \text{where:} \quad v &= y(k+1) - H(k+1)x(k+1|k) \\ S &= H(k+1)P(k+1|k)H(k+1)^T + W(k+1) \\ R &= P(k+1|k)H(k+1)^T S^{-1} \end{aligned}$$

The above equations ensure that the expected value of the error between $x(k)$ and $\hat{x}(k|k)$ is minimized at every time step k, which provides an optimal estimate of the state x. Intuitively, R can be looked at as weight that accounts for the accuracy between the predicted estimate and the measurement noise, where the confidence in the accuracy of the sensor readings increase as R increases. If R is instead a low value, the Kalman filter in turn reduces the influence of sensor data on the update of the belief of the current state. We can in turn linearize the above equations about an estimate of the current state mean and covariance. We approximate the nonlinear function with a linear function that is tangent to the function at the mean of the Gaussian. We do this using a first order Taylor expansion, calculating the Jacobian of f and h and using those matrices to calculate R and P [30]. A summary of the Extended Kalman Filter is given below.

$$
\begin{aligned}
\textbf{System:} \quad x(k+1) &= f(x(k), u(k), k) + v(k) \\
y(k) &= h(x(k), k) + w(k) \\
\textbf{Prediction:} \quad \hat{x}(k+1|k) &= f(\hat{x}(k|k), u(k), k) \\
P(k+1|k) &= F(k)P(k|k)F(k)^T + V(k) \\
\text{where:} \quad F(k) &= \frac{\partial f}{\partial x} \Big|_{x=\hat{x}(k|k)} \\
\textbf{Update:} \quad \hat{x}(k+1|k+1) &= \hat{x}(k+1|k) + Rv \\
P(k+1|k+1) &= P(k+1|k) - RH(k+1)P(k+1|k) \\
\text{where:} \quad v &= y(k+1) - h(x(k+1|k), k+1) \\
S &= H(k+1)P(k+1|k)H(k+1)^T + W(k+1) \\
R &= P(k+1|k)H(k+1)^T S^{-1} \\
H(k+1) &= \frac{\partial h}{\partial x} \Big|_{x=\hat{x}(k+1|k)}
\end{aligned}
$$

3.1.4 Results

The EKF described above was implemented using existing code from the open source ROS code *robot_localization* package [33]. The ability to quickly add and remove sensors from the state estimation provided a quick method for debugging any sensor issues that arose during testing and deployment.

Figure 3.2 shows an example of robot state estimation when using several sensors individually, as well as the robot state estimation using the fusion of multiple sensors. The red arrows are pose estimates made solely using wheel odometry, the yellow arrows are pose estimates made using the GPS data, and the blue arrows are pose estimates made using wheel encoder, IMU, and GPS sensor data fused with the Extended Kalman Filter. As we can see by the red line, wheel encoder data alone drifts significantly over time, particularly during turns. While the GPS data provides a more accurate global position of the robot, it does not directly sense the global orientation of the robot. The blue line represents the most accurate estimate of the robots current state.

Figure 3.3 shows a laser reconstruction of a sorghum breeding field during August 2016. The robot state is estimated using the EKF and wheel encoders, AHRS, and GPS, and the positions of each laser strike are transformed from the laser coordinate frame to a world frame. The coherence of the image is a testament to the effectiveness of the Extended Kalman filter as a state estimator.

3.2 Roll, pitch, and yaw correction

In order to ensure a reliable view of the GPS satellite constellation throughout the entire crop growing cycle, the ideal placement for the GPS antenna is at the top of the mast which supports the linear stage of the manipulator. While this is an ideal placement for maximizing reception of the GPS signal, the information needed is the location of the base of the robot, shown as the black dot in figure 3.4.

The GPS receiver, coupled with RTK correction signals, can provide sub-centimeter accurate positioning of the phase center of the GPS antenna. In order to transfer those

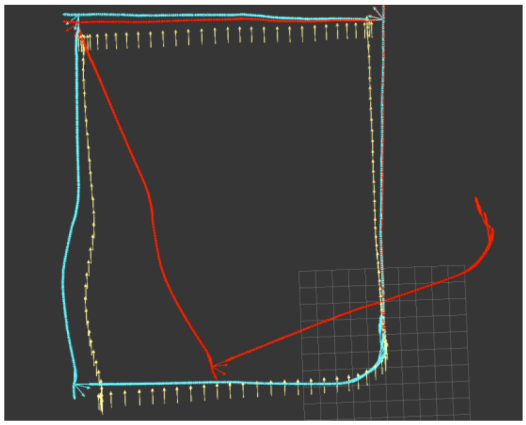


Figure 3.2: State estimation error using mobile robot – red arrows are pose estimates using wheel encoders, yellow are pose estimates using the GPS, and blue are a fusion of the wheel encoders, GPS, and AHRS

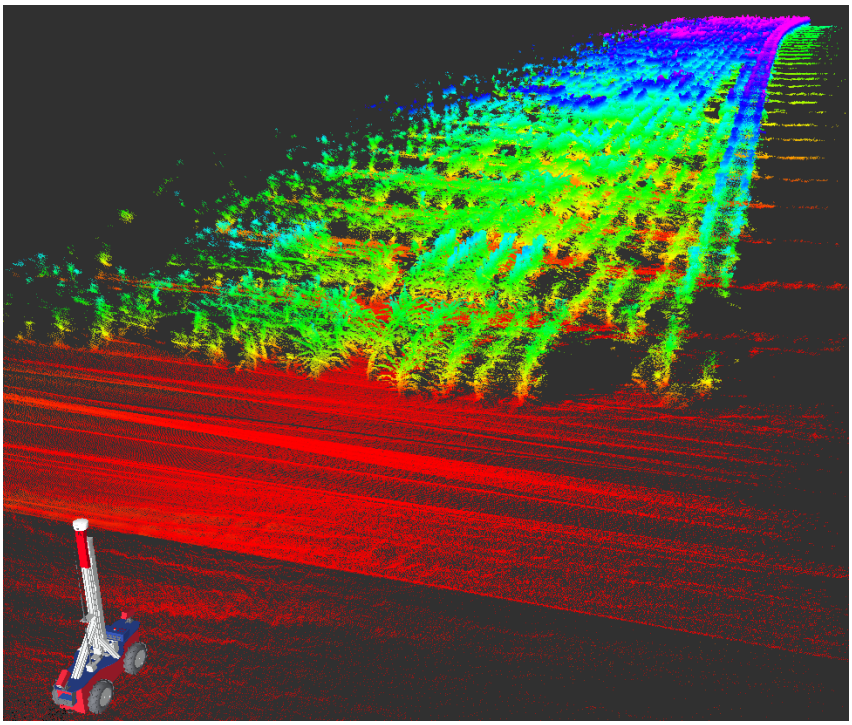


Figure 3.3: Data from planar laser scanners in push-broom configurations collected in a *Sorghum bicolor* breeding plot is transformed to an Earth referenced fixed frame using an Extended Kalman Filter to fuse inertial, magnetometer, GPS, and wheel encoder data.

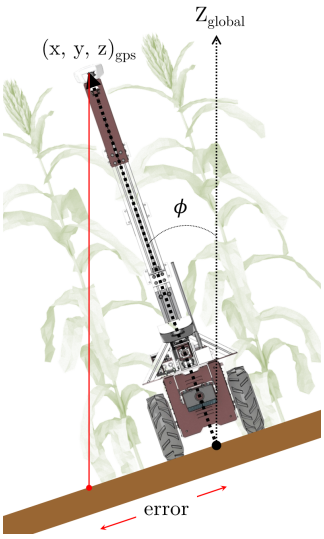


Figure 3.4: Positioning error due to uncorrected tilt of platform

local coordinates to the base of the robot, the orientation of the platform in the Earth-fixed reference frame needs to be known. In this instance the fusion of the output of three sensors to determine the global orientation of the robot is used: the inertial measurement unit (Xsense MTI-30) attached to the mast, the wheel encoders, and a brief history of the previous state of the the robot, as discussed in section 3.1.1.

Once a globally referenced orientation is obtained, this information along with the position of the GPS antenna with respect to the fixed base of the robot is used to apply a homogeneous transform to the pose of the GPS antenna and to obtain the pose of the base of the robot in the Earth-fixed frame. This is done by applying a rigid body transformation to the coordinate of the base in the GPS antenna frame.

Using the roll, pitch, and yaw (ϕ, θ, ψ) obtained in section 3.1.1 along with the GPS coordinates transformed to the Universal Transverse Mercator (UTM) coordinate system, the transformation (T^g_{GPS}) between the Earth-fixed frame (x_g, y_g, z_g) and the GPS frame (x_{GPS}, y_{GPS}, z_{GPS}) is calculated as shown in equation 3.12. Nomenclature in figure 3.5 and symbol convention in the calculations below are taken from [34], and an RPY convention for matrix multiplication is used.

$$T^g_{GPS} = \text{trans}(t_{GPS}) \bullet R_x(\phi) \bullet R_y(\theta) \bullet R_z(\psi) \tag{3.12}$$

$$\text{trans}(t_{GPS}) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 & 0 \\ \sin\psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T^g_{GPS} = \begin{bmatrix} \mathrm{c}\psi \cdot \mathrm{c}\theta & -\mathrm{c}\theta \cdot \mathrm{s}\psi & \mathrm{s}\theta & t_x \\ \mathrm{c}\phi \cdot \mathrm{s}\psi + \mathrm{c}\psi \cdot \mathrm{s}\phi \cdot \mathrm{s}\theta & \mathrm{c}\phi \cdot \mathrm{c}\psi - \mathrm{s}\phi \cdot \mathrm{s}\psi \cdot \mathrm{s}\theta & -\mathrm{c}\theta \cdot \mathrm{s}\phi & t_y \\ \mathrm{s}\phi \cdot \mathrm{s}\psi - \mathrm{c}\phi \cdot \mathrm{c}\phi \cdot \mathrm{s}\theta & \mathrm{c}\psi \cdot \mathrm{s}\phi + \mathrm{c}\phi \cdot \mathrm{s}\psi \cdot \mathrm{s}\theta & \mathrm{c}\phi \cdot \mathrm{c}\theta & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

The pose of the robot base in the Earth-fixed reference frame can now be determined. This is done using the transformation calculated in equation 3.13 along with the position of the robot base in the GPS coordinate frame (r_p^{GPS}). Since the orientation of the GPS frame and the robot frame are already aligned, determining the pose of the robot base can be simplified by treating it as the transformation of a point between two coordinate frames, and simply applying the orientation of the GPS frame to the position of the robot base (r_p^g). The phase center of the GPS antenna is also aligned directly over the center of the robot base, further simplifying calculations.

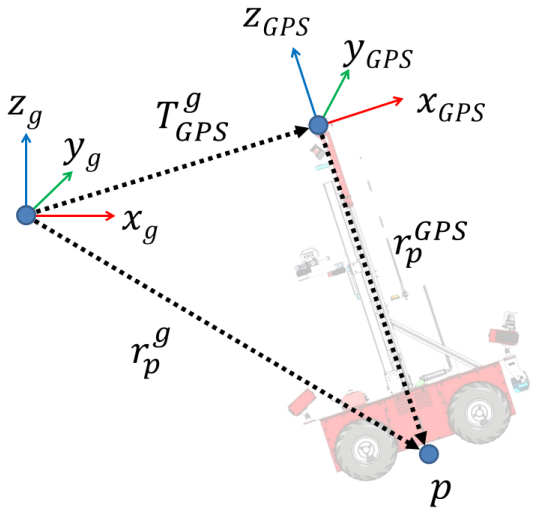


Figure 3.5: Using a rigid body transformation to find the pose at the base of the robot

$$r_p^g = T^g_{GPS} \bullet r_p^{GPS} \tag{3.14}$$

$$r_p^{GPS} = \begin{bmatrix} 0 \\ 0 \\ -\text{height} \\ 1 \end{bmatrix}$$

$$r_p^g = \begin{bmatrix} t_x - \text{height} \cdot \sin\theta \\ t_y + \text{height} \cdot \cos\theta \cdot \sin\phi \\ t_z - \text{height} \cdot \cos\phi \cdot \cos\theta \\ 1 \end{bmatrix} \tag{3.15}$$

Equation 3.14 describes the position of the robot base in the Earth-referenced frame as a function of position of the GPS antenna, the position of the base with respect to the GPS

antenna frame, and the roll and pitch of the GPS antenna with respect to the Earth-fixed frame. Interestingly, since the phase center of the GPS antenna is aligned along the Z-axis with the robot base, the position of the robot base is invariant to the yaw component. This corrected position can now be used for accurately determining the position of the robot regardless of the direction the robot is traveling down the rows.

Without roll/pitch correction of the GPS position data, the true location of the robot within rows becomes ambiguous. Due to the large wheelbase to track ratio of the robot, dynamic roll has a larger effect on error localization than pitch. Positioning error caused by rolling is also more detrimental to the operation of the robot than that caused by pitching. Positioning error parallel to the row will only result in an inaccurate location within a plot for the navigation algorithms and the collected data, and this error can be corrected through post-processing with the available stereo imaging data. Positioning error perpendicular to the row can create ambiguity as to which row the robot is currently traversing, and can also cause the robot to collide with the row itself as it performs waypoint following. With the GPS positioned at a height of 2.05 m, a roll of less than 1.419° is required to ensure the positioning error doesn't fall outside the 5.1 cm of nominal space available between the wheel and the crop row.

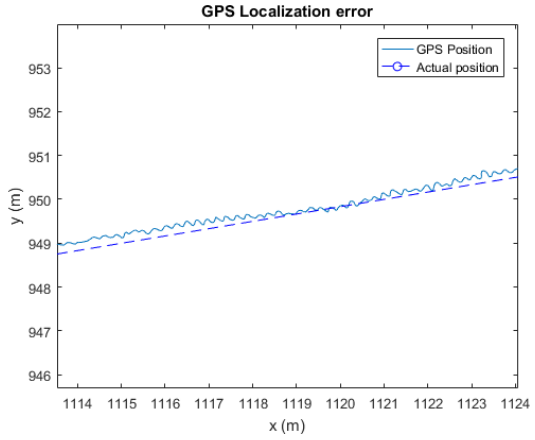


Figure 3.6: Localization error due to robot pitch and roll

Results

Figure 3.6 shows this localization error due to robot pitch and roll. The solid line is the actual position obtained by the GPS receiver, and the dashed line is the actual path of the robot. The data used here was obtained while traversing sorghum breeding plots in Clemson, South Carolina. The rolling of the robot can be seen as oscillations in the recovered position, and is due in large part to the bumpy terrain present within a typical crop row.

The perpendicular localization error due to rolling of the robot is shown in figure 3.7. The crop row corridor is approximately 0.60 - 0.66 m wide, while the robot itself is 0.56 m wide. This leaves less than 0.10 m of allowable error if a navigation algorithm is relying solely on GPS positioning to navigate. As this figure shows, sustained perpendicular errors of up to 0.25 m are present while traversing crop rows, which if fed to a waypoint following algorithm would result in many unhappy plants and plant scientists alike.

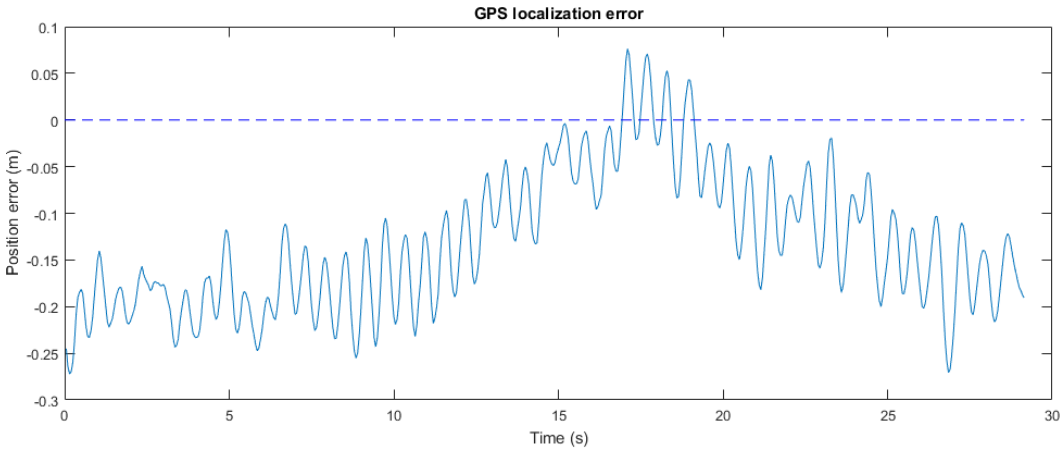


Figure 3.7: Perpendicular localization error due to robot roll

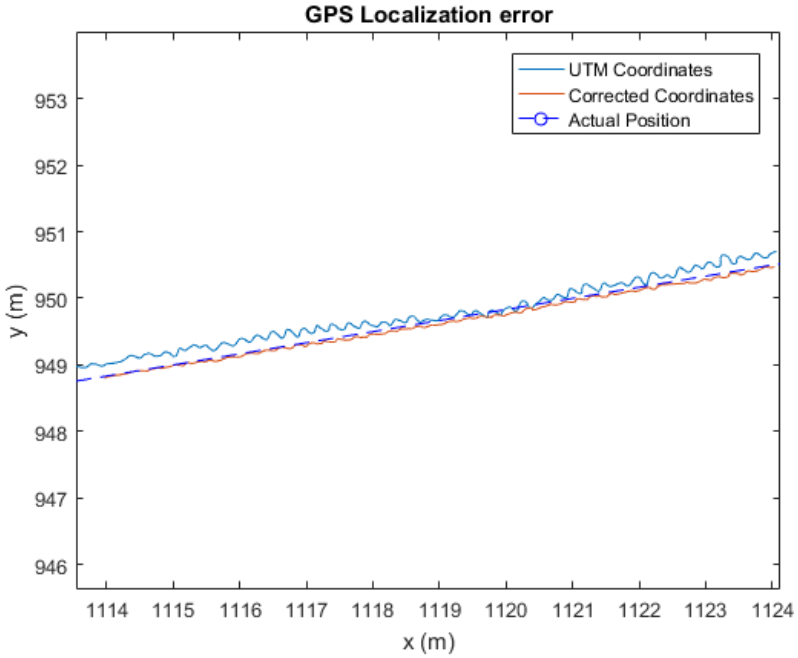


Figure 3.8: Uncorrected and corrected localization error due to robot pitch and roll

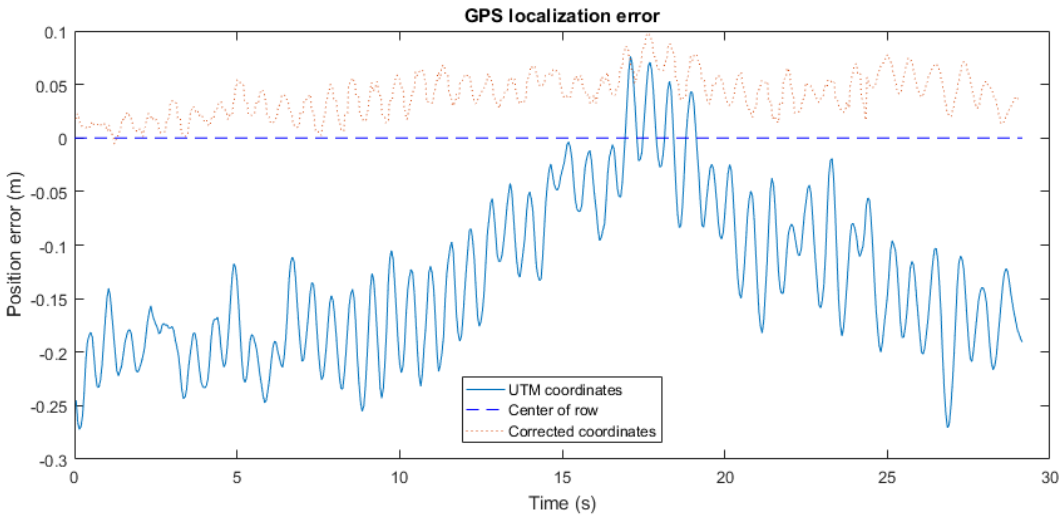


Figure 3.9: Uncorrected and corrected perpendicular localization error due to robot roll

Figures 3.8 and 3.9 show the position of the robot base as calculated above using the Earth-referenced orientation and GPS coordinates. The perpendicular localization error is now below 0.1 m, and therefore this corrected pose can now be used for waypoint navigation safely. There is still some noise and oscillation present in the positioning error, which is due in part to a misalignment of the expected position of the GPS antenna, along with time delays present when passing the orientation and position data to the ROS nodes to be calculated. Further engineering work can be performed to increase the accuracy of this pose estimate through the use of a total survey station to get a more accurate result for the translation between the antenna and the robot base, along with further optimization of code.

3.3 Pure Pursuits

Pure Pursuits is a relatively simple geometric path tracking algorithm that has been implemented on a wide variety of robotic platforms. CMUs Navlab[35] successfully used the Pure Pursuit algorithm to calculate steering angles to track a desired path, and showed that its low computational cost and ease of tuning made it a more suitable choice than more complex path following algorithms such as a quintic polynomial tracker. Giesbrecht et al.[36] have successfully used the Pure Pursuits algorithm to follow waypoints generated by a D* Lite path planner, and has shown success in following paths that constantly update due to detected obstacles.

Our implementation of Pure Pursuits path tracking algorithm follows a series of waypoints input by human operators that indicate the beginning and ends of plots. At any given instant, our algorithm attempts to track to a line drawn between the previous and the next waypoint (P_i, P_{i+1}). The distance from the robot to the next waypoint is tracked, and if this distance falls below the lookahead distance then the next set of waypoints are used to calculate the goal path, i.e. (P_{i+1}, P_{i+2}). This continues until the last waypoint is achieved,

at which point the robot is commanded to stop. This method of shifting from line to line by detecting distance to the next waypoint can lead to issues if the lookahead distance is too long or consecutive waypoints are too close to one another with a large change of heading. If the robot passes a waypoint without traveling within a lookahead distance, the robot will continue to travel along a path indefinitely. This is mitigated by ensuring the distance threshold is greater than or equal to the lookahead distance.

The output of our implementation is the difference between the heading of the robot frame and the heading from the robot position to the lookahead point (α). This error is then fed into a proportional controller which generates a yaw rate which, along with a constant forward velocity, is passed to the velocity controller driving the Robotanist's wheels. At every position update, a new lookahead point and a new heading error is calculated, and a new yaw rate is passed to the velocity controller.

3.3.1 Lookahead point and heading error

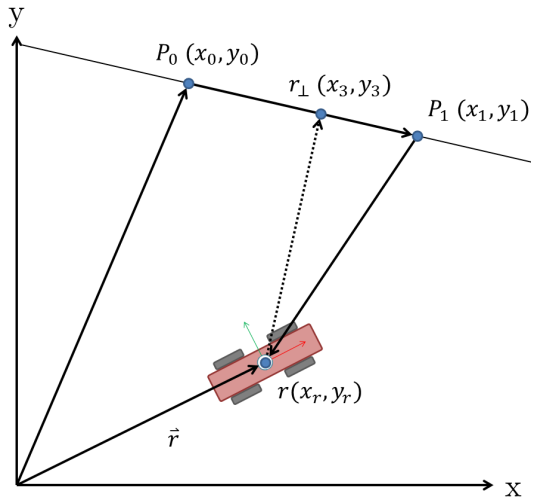


Figure 3.10: Determining point on goal path closest to the current robot pose

Our implementation is initially passed waypoints P_0 and P_1, the robot position $\vec{r}$, and the robot heading $\vec{h}$. From these data the lookahead point and subsequent heading error between the current robot pose and the calculated lookahead point can be calculated. The scalar projection of the vector between the current robot position and the next goal waypoint is first determined, which is then used to determine the position of the point on the line closest to the robot ($r_\perp$). This is visualized in figure 3.10, calculations are shown below.

$$\vec{\mathbf{P}} = < x_1 - x_0, y_1 - y_0 > \qquad \vec{\mathbf{r}}_P = < x_r - x_1, y_r - y_1 >$$

$$\hat{\mathbf{P}} = \frac{\vec{\mathbf{P}}}{\|\vec{\mathbf{P}}\|} \qquad r_{proj} = \vec{\mathbf{r}}_P \cdot \hat{\mathbf{P}}$$

$$\vec{\mathbf{r}}_\perp = \vec{\mathbf{P}}_1 + r_{proj} \cdot \hat{\mathbf{P}} \tag{3.16}$$

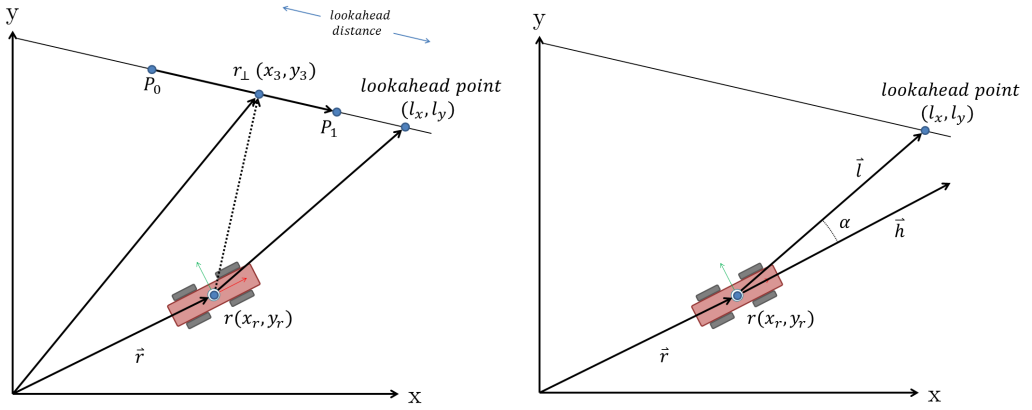
$$\vec{\mathbf{r}}_{\perp} =< x_3, y_3 >=< x_1, y_1 > +\Big((x_r - x_1) \cdot (\frac{x_1 - x_0}{\|\vec{\mathbf{P}}\|}) + (y_r - y_1) \cdot (\frac{y_1 - y_0}{\|\vec{\mathbf{P}}\|})\Big) \cdot \hat{\mathbf{P}}$$

From the closest point on the line $r_{\perp}$ to the robot, the path is then traversed one unit of lookahead distance in the direction of the next waypoint, as shown in figure 3.11a and calculated using equation 3.17. This is the lookahead point where we will attempt to drive our robot heading through the use of a proportional controller. The heading error α shown in figure 3.11b is calculated using equation 3.18. At this point the calculated heading error is unsigned. The cross product of the robot heading $\hat{h}$ and $\vec{l}$, and the

This signed heading error is then passed as an error term into a proportional controller, which will drive the heading of the robot toward the lookahead point.

$$\vec{\mathbf{r}}_l = \vec{\mathbf{r}}_{\perp} + lookahead \cdot \hat{\mathbf{P}} \tag{3.17}$$

$$\alpha = \arccos\Big(\frac{\hat{\mathbf{h}} \cdot \vec{\mathbf{l}}}{\|\vec{l}\|}\Big) \tag{3.18}$$



(a) Lookahead point on goal path

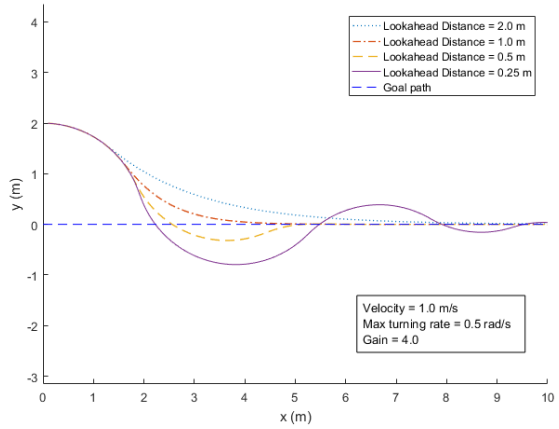
(b) Determining heading error

Figure 3.11: Determining lookahead error for Pure Pursuits

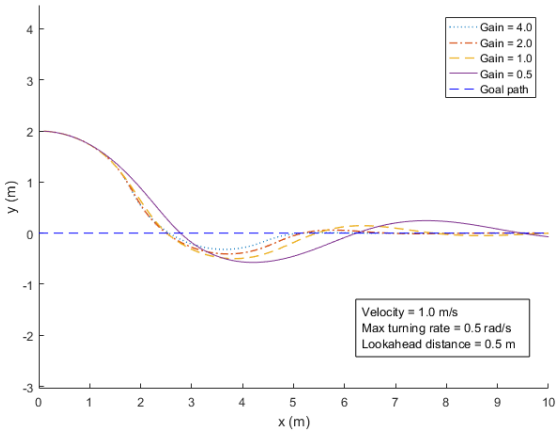
3.3.2 Simulation and implementation results

Using the Pure Pursuits formulation above, a simulation of the path planning algorithm was developed in MATLAB. A kinematic model describing the motion of a differential robot, as shown in equation 3.19, was used to generate successive robot positions. This model assumes some constant forward velocity v_f and some constant angular velocity ω over a time interval t, which defines a circular arc of length $s = v_f \cdot t$. Using the initial robot pose $(r_{x_0}, r_{y_0}, r_{\theta_0})$ a new robot pose (r_x, r_y, r_θ) is calculated over each time interval.

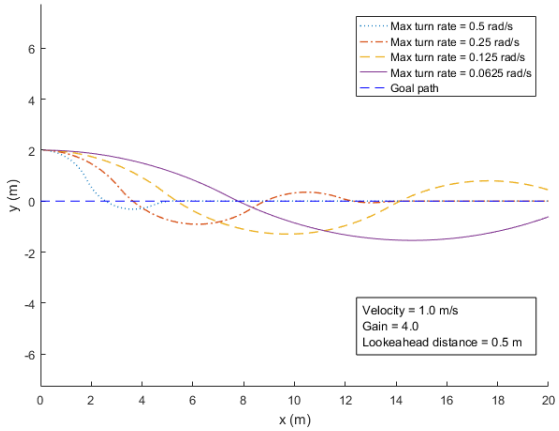
$$\begin{bmatrix} r_x \\ r_y \\ r_\theta \end{bmatrix} = \begin{bmatrix} r_{x_0} - \frac{v}{\omega}\sin(r_{\theta_0}) + \frac{v}{\omega}\sin(\omega \cdot t + r_{\theta_0}) \\ r_{y_0} - \frac{v}{\omega}\cos(r_{\theta_0}) + \frac{v}{\omega}\cos(\omega \cdot t + r_{\theta_0}) \\ r_{\theta_0} + \omega \cdot t \end{bmatrix} \tag{3.19}$$



(a) Robot path as a function of lookahead distance



(b) Robot path as a function of gain



(c) Robot path as a function of maximum turning rate

Figure 3.12: Pure Pursuits parameter tuning

This simulation enforces a limit on the angular velocity which the differential robot can be commanded, as in practice the inherent dynamics of the robot would limit the motion of robotic platform. The input parameters into the simulation are listed in table 3.1. These input parameters were used to initialize our simulation, and the parameters were tuned such that they would translate to our row following application, thus minimizing the required parameter tuning once the platform was deployed to the field.

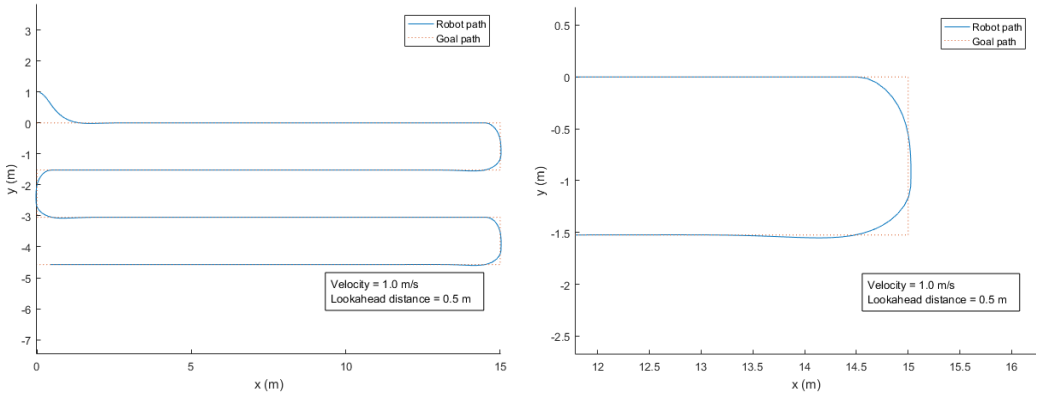
Table 3.1: Input parameters in Pure Pursuit simulation

Parameter	**Description**
$\vec{r}_0$	Initial robot pose
v_f	Forward velocity (m/s)
l_d	Lookahead distance (m)
t	Position update rate (Hz)
k	Proportional controller gain
ω_{max}	Maximum angular velocity (rad/s)
P	Waypoints
d	Waypoint distance threshold

The tunable parameters that significantly impacted the performance of our path following algorithm were the lookahead distance l_d, the gain k, and to a smaller extent the maximum angular velocity ω_{max}. As can be seen in figure 3.12a, decreasing the lookahead distance results in the robot achieving the goal path more quickly, but if this value is set too low (<0.5 m), the system will become underdamped there will be significant overshoot and oscillation about the goal path. If the lookahead is too long (>1.0 m), the system will be overdamped and will take longer to reach the goal path. From this starting robot pose, a lookahead distance between 0.5 and 1.0 m will result in the system being approximately critically damped, where the system will reach the goal path without overshoot as quickly as possible. Testing in the field indicated that a lookahead distance of 0.5 m resulted in minimal overshoot and quick acquisition of the goal path when switching waypoints.

Increasing the gain on the heading error will result in a higher angular velocity, though this will be limited by the dynamics of the robot in practice, and is limited by the maximum turn rate in this simulation (Fig. 3.12b). It was discovered during testing that if the gain is set too high, localization errors due to temporary GPS dropout or sensor failures and noise will result in significant oscillation while traversing the crop row, and could result in the platform striking the crop row due to sudden corrections. During testing we determined that a gain of 4.0 was high enough to reach the goal path quickly between waypoints without resulting in high frequency oscillation within the rows.

Decreasing the maximum angular rate results in the system becoming increasingly underdamped, as seen in figure 3.12c. As the Robotanist consists of a skid-steer drive with difficult-to-predict slip, and the terrain of a typical sorghum breeding field varies from muddy to dry, with ruts at various points throughout the field, the maximum turning rate can vary wildly. As a result, it is difficult to use our model to finely tune all parameters prior to implementation in the field. If a low turning rate is assumed, then the forward velocity needs to be reduced in order to ensure the robot acquires a new path quickly, but this results in a longer time to traverse the field. On the other hand, if the turning rate is assumed to be higher than the robot can actuate in the field then the probability of the robot impacting the crop at the start of rows increases.



(a) Following a multi-node path

(b) Waypoint transition

Figure 3.13: Pure Pursuits path following simulation

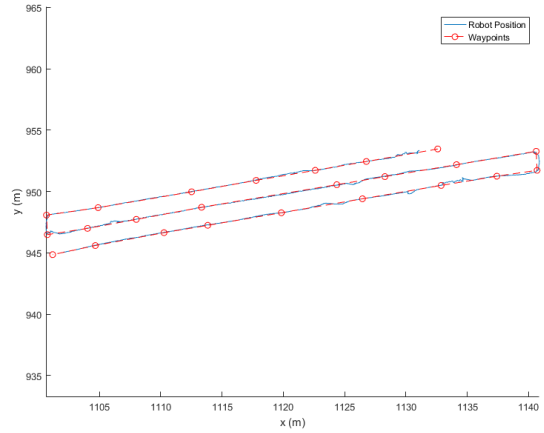
Figure 3.13 shows a simulation of our Pure Pursuits implementation. As can be seen, the robot quickly achieves a path along the line from an initial offset of 1 meter and quickly acquires the next path with little overshoot. In this instance the distance threshold for transitioning to the next waypoint was 0.5 m. The maximum angular rate was set to a value of 2.5 rad/s, which is unachievable by the robotic platform in the field. One case our model doesn't account for is a decrease in forward velocity due to slip of the wheels when attempting to turn too quickly. This is due to the commanded rotational velocity saturating the wheel motors and preventing the commanded forward velocity from being achieved.

Figure 3.14 shows the implementation of our Pure Pursuits algorithm on the Robotanist while traversing through several rows of a sorghum breeding plot in Clemson, South Carolina, during deployment in July, 2017. With a little experimentation, the final parameters that were determined to be optimal were a lookahead distance of 0.5 m, a gain of 4.0, and a forward velocity of 1.0 m/s. As we can see in figure 3.14c, the robot reliably follows the goal path once acquired, with slight oscillations due to a combination of the large gain and errors in the state estimation used for roll/pitch correction. Transitioning from one set of waypoints to the next is relatively smooth, and the robot achieves the new goal path relatively quickly, as can be seen in figure 3.14b. There was a rut at the first waypath shown along the path, and the robot performed a zero-radius turn due to significant slip along the left wheels.

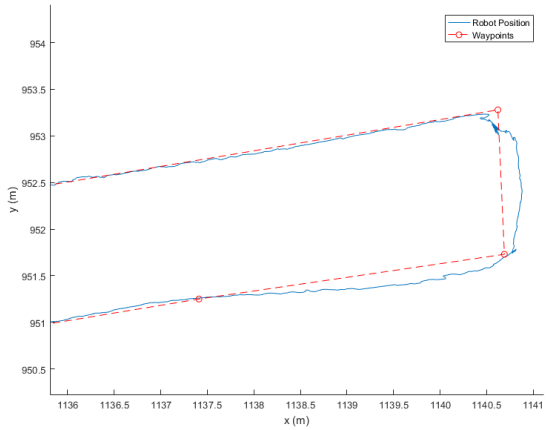
Throughout testing in the field, while the state estimate of the robot was accurate, our implementation of the Pure Pursuits algorithm resulted in reliable row following even through dense vegetation and rows with significant sloping due to ruts caused by farm implementations.

3.4 Localization under the canopy

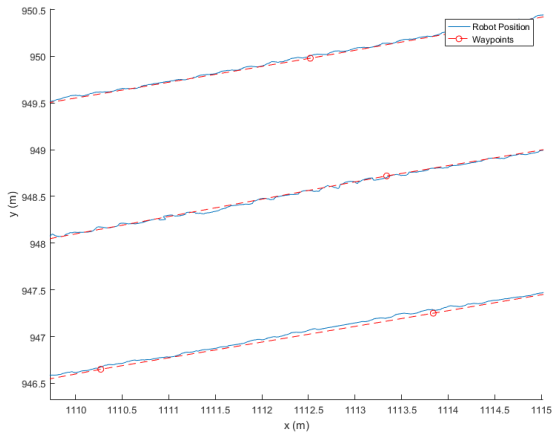
Path following using absolute positioning provided by the GPS receiver as outlined in section 3.3 is a very useful technique that is sufficient for navigating within a known breeding plot, so long as no sensor failure occurs, the RTK correction is available, and a sufficient view of the GPS satellite constellation is available. These first two issues can be mitigated through



(a) Path following between waypoints



(b) Close up of typical U-turn (path is clockwise)



(c) Straight line path following

Figure 3.14: Pure Pursuits implementation in sorghum breeding plots

the use of redundant sensors and proper infrastructure, but the latter is limited by the height of the plants themselves. Sorghum bred for bio-energy can reach heights upwards of 20 feet tall, and utilizing a mast that can extend above that height is unfeasible with our developed platform. This necessitates the development of some other method to allow for reliable navigation within the breeding plots throughout the entirety of the growing season. To this end, we investigate the use of data from the Sick Visionary-T sensor placed on the front of the robot and shown in figure 3.15.

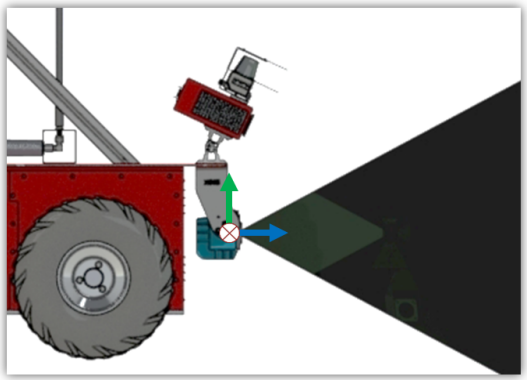


Figure 3.15: Sick Time of Flight sensor coordinate frame and field of view

3.4.1 Row following with a Time-of-Flight sensor

The Visionary-T is a sensor that provides range data (bottom of *a)* in figure 3.17) with corresponding intensity and confidence values (top and middle of *a)* in figure 3.17), and is placed on the front of the robot providing a view of the row from beneath the crop canopy. This sensor utilizes infrared light sources to obtain range data, and as a result the noise present in these data can vary wildly depending on the level of ambient light. Fortunately we will be using this data to navigate while primarily under a crop canopy, which lowers the intensity of ambient lighting in the scene and will subsequently lower the amount of noise present in the the point clouds.

The rows within a typical sorghum breeding plot are cluttered and frequently occluded. Due to the variation in the plant genotype (and its expression) from plot to plot, the corridor between these crop rows can change significantly within each plot. This makes utilizing common features among all breeds in the field a difficult task, as often times features present in one plot will be absent or occluded in the next, e.g. a short, bushy grain sorghum breed will be placed next to a tall variety whose lower leaves senesce.

With this in mind, we developed a point cloud processing pipeline that attempts to localize within the two plant rows by detecting the rows of sorghum stalks immediately to the left and right of the robot and feeding that crosstrack error into a velocity controller. This proportional controller drives the platform towards the center of the row. The pipeline, shown in figure 3.16, begins by removing extraneous points that are outside the range of the expected adjacent rows (spatial filter 1), shown below in the sensor coordinate frame.

$$
\begin{aligned}
-0.75m \leq \quad & x \quad \leq 0.75m \\
-0.05m \leq \quad & y \quad \leq 0.50m \\
0.40m \leq \quad & z \quad \leq 5.00m
\end{aligned}
$$

Statistical analysis is then performed on each point in the output of this initial filtering to remove noise and outliers from the point cloud. This is done by first calculating the distance from each point to all other points, then calculating the mean and standard deviation to all k-nearest neighbors for all points. A point is then considered an outlier if the mean distance for it's k-nearest neighbors is some multiple of a standard deviation away from the global mean for all points.

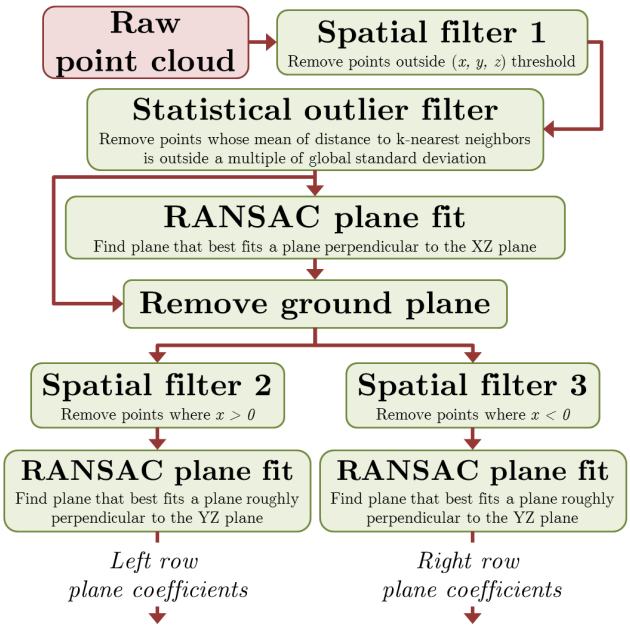
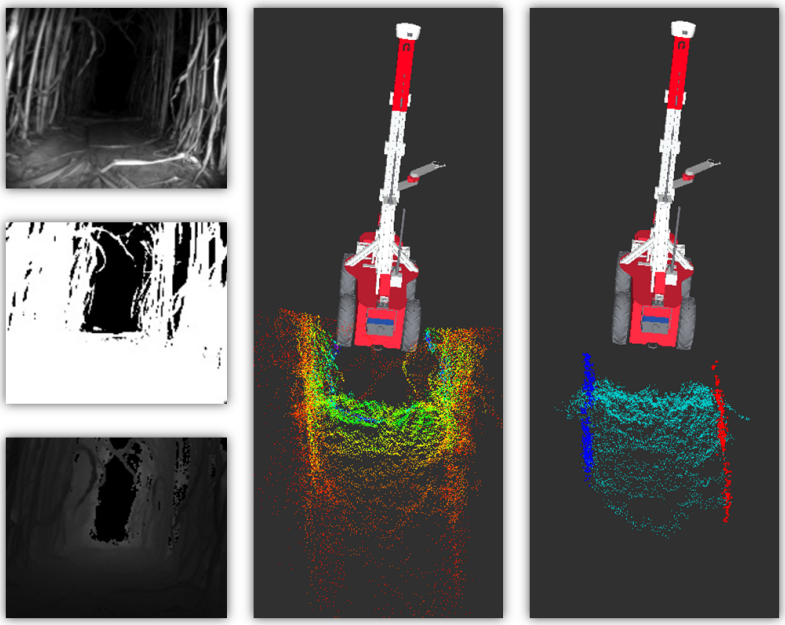


Figure 3.16: Pipeline for detecting left and right crop rows

Once these outliers have been removed, we should obtain a relatively clean point cloud which consists primarily of the sorghum stalks, leaves, and the ground. The cleanliness of this point cloud is a function of the genotype and age of the plant. For low, bushy plants there may be significant occlusion from leaves low and in the center of the row which prevent the sensor from detecting the sorghum stalks and which introduce leaves into the filtered point cloud. For taller plants whose lower leaves have senesced, we will obtain point clouds which look similar to that which is displayed in b) in figure 3.17.

At this point in the pipeline, we attempt to fit a plane to points in the filtered point cloud which correspond to the ground. This is done using the random sample consensus algorithm (RANSAC), a popular non-deterministic method for estimating model parameters in noisy data. The algorithm attempts to find the parameters of a plane ($ax + by + cz + d = 0$) which fits the given 3D points in the point cloud as closely as possible by attempting to maximize the number of points within some Euclidean distance threshold. This is performed as follows:

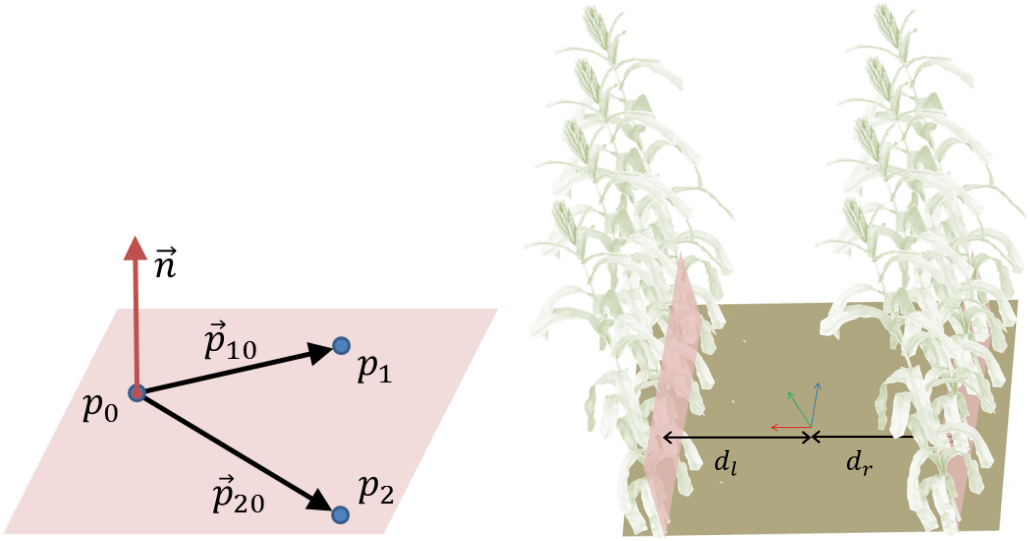


a) b) c)

Figure 3.17: SICK Time of Flight sensor data and detected rows

1. Three points are randomly chosen from the input cloud, with a check to ensure they aren't collinear.
2. A plane is fit to those three points, as shown in figure 3.18a, where $\vec{n} = \vec{p}_{10} \times \vec{p}_{20}$.
3. If the normal angle of this plane is greater than an epsilon away from the desired plane normal angle, go to 1.).
4. For all points in the point cloud, calculate the normal distance to this plane.
5. If the distance is less than some threshold, consider that point an inlier.
6. If the number of inliers is greater than the number of inliers in the previous best model, store this new model.
7. Repeat from *1.)* for a specified number of iterations.
8. Re-calculate plane model parameters using the inliers found with the best model above, using Least-Squares regression.
9. Put model in Hessian Normal form, where $\hat{n} \cdot \vec{x} = -p$.

Once the ground plane has been detected using the RANSAC plane fitting algorithm shown above, with the desired plane normal being perpendicular to the the points detected within the ground plane are removed from the filtered point cloud. At this point we've isolated the leaves and stalks of the sorghum, and can start reasoning about the placement of each row with respect to the sensor.



(a) Calculating plane with 3 points

(b) Calculating crosstrack heading error

Figure 3.18: RANSAC plane fitting for detecting crosstrack error

To detect the left row, we first filter out all points where $x < 0$, and then perform RANSAC plane fitting on the remaining points, with the constraint that the plane is roughly perpendicular to the YZ plane. Detecting the right row is performed similarly to the left, except that we apply a spatial filter which removes all points where $x > 0$. Once we have the parameters describing the planes passing through the left and right rows, we can calculate the crosstrack error of the sensor. As the plane model parameters are in Hessian Normal form, the distance to the plane from the origin is simply the final coefficient, d. Once we obtain the distance to the left plane and the right plane, d_l and d_r respectively, we can estimate the signed crosstrack error of the robot by taking the difference of the distance from the right plane to the distance to the left plane, $e_c = d_r - d_l$, shown in figure 3.18b. We can now use this crosstrack error as an input into a velocity controller to keep the robot in the center of the detected row.

3.4.2 Results

The above algorithm was implemented on the Robotanist in C++ using the ROS framework and the Point Cloud Library [37]. The navigation algorithm was tested across multiple rows of sorghum breeding plots in Clemson, South Carolina, during the Summer of 2017. The platform was driven between rows of sorghum planted at row widths of 0.762 meters, and was driven between multiple plots with approximately 1-3 meter gaps between them. An example of sensor data from these experiments are shown in figure 3.17, where a) contains (top to bottom) intensity, confidence, and depth map data, b) shows the unfiltered point cloud along with the robot model, and c) shows the detected ground plane (cyan) and the detected left and right rows (red and blue, respectively).

As mentioned above, the crosstrack error signal is sent to a proportional velocity controller that drives the angular velocity of the robot in the direction of the detected crosstrack

error. The forward velocity of the robot was fixed at 0.5 m/s. The tuning of the proportional controller was straightforward: too low and the applied angular velocity would not affect the heading of the robot, too high and the robot would collide with the opposite row before an updated crosstrack error could re-correct.

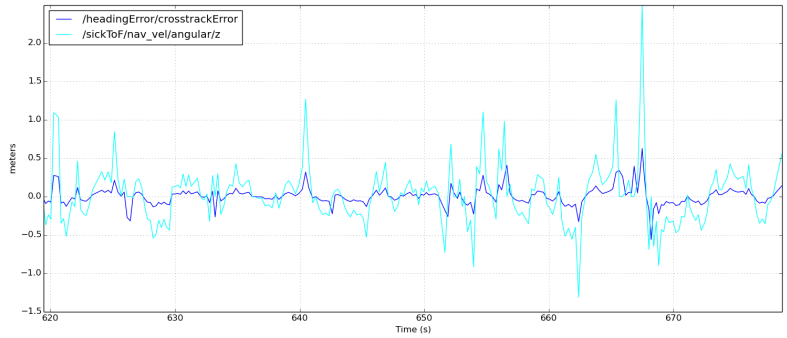


Figure 3.19: Detected heading error and corresponding angular velocity commands on successful navigation of crop row

Figure 3.19 shows a brief portion of the crosstrack error and its corresponding commanded angular velocity from a successful autonomous run through a row of sorghum. We used a qualitative metric to measure the success of the algorithm; the robot was placed in autonomous mode and allowed to traverse between rows while being followed by a human operator. For plots where there wasn't significant occlusion of the sorghum stalks due to low, bushy varieties, the in-row localization performed well, with no input from the operator required. When short varieties were encountered, the algorithm would not detect any rows and therefore the robot would be commanded to maintain its current heading. This would often result in a potential for collision with a plant, at which point the operator would override control of the robot and teleoperate the platform through that particular plot. As we can see, reliable navigation with any one sensor is not guaranteed, therefore future work should involve fusing multiple sensors and navigation algorithms, utilizing the current sensors with the highest confidence.

Chapter 4

Conclusion

One can envision a fully autonomous agricultural robot able to autonomously deploy itself from a docking station to breeding plots and begin traversing between rows. It is possible to develop algorithms that will enable such a robot to make data collection decisions in real-time, such as informing researchers which regions of a field require further inspection. The Robotanist does not yet exhibit these capabilities, but the current system architecture can accommodate this additional functionality.

Future work includes utilizing the availabe navigation sensors to localize within the crop rows. Some preliminary work has been performed using the forward facing camera to localize within the crop rows by detecting the edges where the crop stalks meet the soil, fitting lines to those edges using a Hough line transform, determining the vanishing point of adjacent rows, and determining the heading error from their intersection point. Unfortunately the wide variation in crop cover, color, and lighting caused significant error when detecting these edges. Localization errors regularly exceeded the limits which would result in the platform striking the crop rows. Future attempts may make the use of the current state of the art in deep learning for computer vision to accomplish this task.

Some preliminary work was also performed on range data obtained using the forward facing LiDAR. A particle filter algorithm was implemented using a map of the expected crop rows and implemented in C++ with the aid of the OpenCV library. Figure 4.2 shows the histogram of range data from laser returns along a 100 m straight section of the sorghum breeding plots during the early season. Using a multi-modal Gaussian sensor model, we tested the particle filter on data from several stages in the growing season. While the results of localization were promising, with a localization accuracy of ± 0.04 m, there error was still too large to be used reliable for navigation purposes.

In this thesis we present the problem of designing a platform capable of reliably navigating within a typical sorghum breeding plot while carrying a modular array of contact and non-contact sensors. We outline the design process for the platform, justify our design decisions, and describe state estimation, path following, and navigation algorithms we implemented which have been subsequently proven during the Summer growing season of 2016 and 2017 at Clemson University sorghum breeding fields in Clemson, South Carolina. Field validation of the developed hardware is presented.

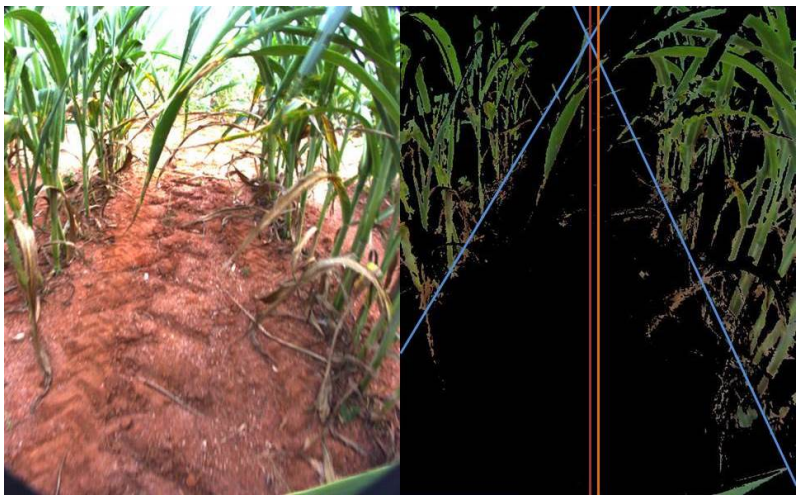


Figure 4.1: Detected crop rows with lines fit using a Hough transform (blue), the detected vanishing point (red), and the heading of the camera (orange)

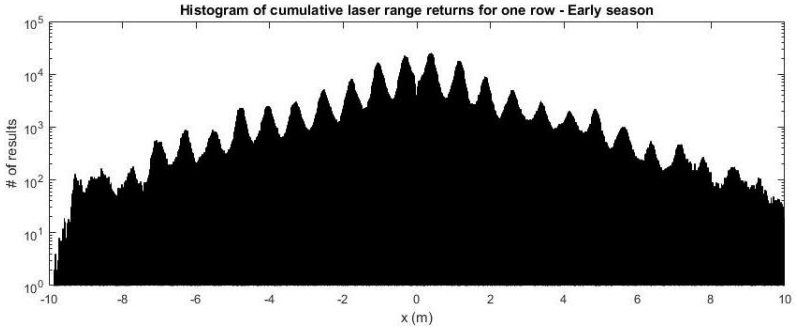


Figure 4.2: Histogram of laser returns along a 100 m straight section of sorghum during the early season

Bibliography

[1] Wetterstrand KA. DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP). `https://www.genome.gov/sequencingcostsdata`, 2016. [Online; accessed 15-July-2017].

[2] Farms and Farmland - Numbers, Acreage, Ownership, and Use. `https://www.agcensus.usda.gov/Publications/2012/Online_Resources/Highlights/Farms_and_Farmland/Highlights_Farms_and_Farmland.pdf`, September-2014. [Online; accessed 20-July-2017].

[3] Transportation Energy Resources from Renewable Agriculture (TERRA). `https://arpa-e.energy.gov/?q=arpa-e-programs/terra`, 18-June-2015. [Online; accessed 20-July-2017].

[4] Robert T. Furbank and Mark Tester. Phenomics – technologies to relieve the phenotyping bottleneck. *Trends in Plant Science*, 16(12):635–644, 2011.

[5] Elizabeth Heather Neilson, AM Edwards, CK Blomstedt, B Berger, B Lindberg Møller, and RM Gleadow. Utilization of a high-throughput shoot imaging system to examine the dynamic phenotypic responses of a c4 cereal crop plant to nitrogen and water deficiency over time. *Journal of experimental botany*, 66(7):1817–1832, 2015.

[6] Jeffrey K Conner, Rachael Franks, and Christy Stewart. Expression of additive genetic variances and covariances for wild radish floral traits: comparison between field and greenhouse environments. *Evolution*, 57(3):487–495, 2003.

[7] F. Liebisch *et al*. Remote, aerial phenotyping of maize traits with a mobile multi-sensor approach. *Plant methods*, 11(1):1–20, 2015.

[8] S. Sankaran *et al*. Low-altitude, high-resolution aerial imaging systems for row and field crop phenotyping: A review. *European Journal of Agronomy*, 70:112–123, 2015.

[9] P. Andrade-Sanchez *et al*. Development and evaluation of a field-based high-throughput phenotyping platform. *Functional Plant Biology*, 41(1):68–79, 2014.

[10] S. C. Murray *et al*. High clearance phenotyping systems for season-long measurement of corn, sorghum and other row crops to complement unmanned aerial vehicle systems. In *Proc. SPIE*, volume 9866, pages 1–8, 2016.

[11] Nicolas Virlet, Kasra Sabermanesh, Pouria Sadeghi-Tehran, and Malcolm J Hawkesford. Field scanalyzer: An automated robotic field phenotyping platform for detailed crop monitoring. *Functional Plant Biology*, 44(1):143–153, 2017.

[12] Lemnatec GmbH. Field Scanalyzer Systems. http://www.lemnatec.com/products/field-phenotyping/field-scanalyzer/, 2016. [Online; accessed 21-July-2017].

[13] Pouria Sadeghi-Tehran, Kasra Sabermanesh, Nicolas Virlet, and Malcolm J Hawkesford. Automated method to determine two critical growth stages of wheat: Heading and flowering. *Frontiers in Plant Science*, 8, 2017.

[14] Norbert Kirchgessner, Frank Liebisch, Kang Yu, Johannes Pfeifer, Michael Friedli, Andreas Hund, and Achim Walter. The eth field phenotyping platform fip: a cable-suspended multi-sensor system. *Functional Plant Biology*, 44(1):154–168, 2017.

[15] Thomas Bak and Hans Jakobsen. Agricultural robotic platform with four wheel steering for weed detection. *Biosystems Engineering*, 87(2):125–136, 2004.

[16] S. Bonadies *et al*. A survey of unmanned ground vehicles with applications to agricultural and environmental sensing. In *Proc. SPIE*, volume 9866, pages 1–14, 2016.

[17] M. Bergerman *et al*. Results with autonomous vehicles operating in specialty crops. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1829–1835, May 2012.

[18] Clearpath Robotics. Husky Technical Specifications. https://www.clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/. [Online; accessed 25-July-2017].

[19] Robotnik Automation, SLL. Available Mobile Robots. http://www.robotnik.eu/mobile-robots/. [Online; accessed 25-July-2017].

[20] QinetiQ North America. TALON®. https://www.qinetiq-na.com/products/unmanned-systems/talon/. [Online; accessed 25-July-2017].

[21] Omron Adept Technologies, Inc. Omron Adept Research Robots Technical Specifications. http://www.mobilerobots.com/ResearchRobots/Specifications.aspx. [Online; accessed 25-July-2017].

[22] K. Cavender-Bares and J.B. Lofgren. Robotic platform and method for performing multiple functions in agricultural systems, February 23 2016.

[23] J. Zhang *et al*. Monocular visual navigation of an autonomous vehicle in natural scene corridor-like environments. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 3659–3666. IEEE, 2012.

[24] J. Xue *et al*. Variable field-of-view machine vision based row guidance of an agricultural robot. *Computers and Electronics in Agriculture*, 84:85–91, 2012.

[25] Ji Zhang, Andrew Chambers, Silvio Maeta, Marcel Bergerman, and Sanjiv Singh. 3d perception for accurate row following: Methodology and results. In *Proc.IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 5306–5313. IEEE, 2013.

[26] Santosh Hiremath, Frits K Van Evert, Cajo ter Braak, Alfred Stein, and Gerie van der Heijden. Image-based particle filtering for navigation in a semi-structured agricultural environment. *Biosystems Engineering*, 121:85–95, 2014.

[27] David Reiser, Garrido Miguel, Manuel Vázquez Arellano, Hans W Griepentrog, and Dimitris S Paraforos. Crop row detection in maize for developing navigation algorithms under changing plant growth stages. In *Robot 2015: Second Iberian Robotics Conference*, pages 371–382. Springer, 2016.

[28] Dimitrios Apostolopoulos. Analytic configuration of wheeled robotic locomotion. 2001.

[29] M Saarilahti et al. Soil interaction model. Technical report, 2002.

[30] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[31] Jingang Yi, Hongpeng Wang, Junjie Zhang, Dezhen Song, Suhada Jayasuriya, and Jingtai Liu. Kinematic modeling and analysis of skid-steered mobile robots with applications to low-cost inertial-measurement-unit-based motion estimation. *IEEE transactions on robotics*, 25(5):1087–1097, 2009.

[32] Howie M Choset. *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[33] Thomas Moore and Daniel Stouch. A generalized extended kalman filter implementation for the robot operating system. In *Intelligent Autonomous Systems 13*, pages 335–348. Springer, 2016.

[34] Alonzo Kelly. *Mobile Robotics: Mathematics, Models, and Methods*. Cambridge University Press, 2013.

[35] Omead Amidi. Integrated mobile robot control. Technical Report CMU-RI-TR-90-17, Pittsburgh, PA, May 1990.

[36] J. Giesbrecht et al. Path tracking for unmanned ground vehicle navigation: Implentation and adaptation of the pure pursuit algorithm. Technical Memorandum, 2005.

[37] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.