FULL SEASON NAVIGATION AND CONTROL OF ULTRA-COMPACT AUTONOMOUS AG-BOT IN GPS DENIED ENVIRONMENT

BY

KARAN CHAWLA

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Aerospace Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

Adviser:

Assistant Professor Girish Chowdhary

ABSTRACT

This thesis describes perception and an autonomous navigation system for an ultra-lightweight ground robot in agricultural fields. The system is designed for reliable navigation under cluttered canopies using only a 2-D Hokuyo UST-10LX LiDAR and an RTK GPS as the primary sensors for navigation. Its purpose is to ensure that the robot can navigate through rows of crops without damaging the plants in narrow row-based and high-leaf-cover semi-structured crop plantations, such as corn (Zea mays), sorghum (Sorghum bicolor) and soybean (Glycine max). The fundamental contributions of this work are a GPS-INS based reliable localization system for the robot and a LiDAR-based navigation algorithm capable of rejecting outlying measurements in the point-cloud due to plants in adjacent rows, low-hanging leaf cover, or weeds. Finally, this work describes a behavior-based navigation architecture that enables the system to autonomously traverse a breeding field throughout the planting season.

To my parents, for their love and support.

ACKNOWLEDGMENTS

I would like to first and foremost thank my parents and family, for fostering my curiosity and having my back every time I failed. To my mother, for believing in me. To my father, for sacrificing so much to provide me with opportunities, he never had. To my advisor, Dr. Girish Chowdhary for his enthusiasm, encouragement, and support, and for giving me the opportunity to work on this project. To Hunter Young, for the development of simulation. To Nolan Replogle and Joey Byrnes, who offered their expert and candid advice during my time in the lab. To Lukas Renker and Sri Theja Vuppala, for their help with testing and debugging. And finally, to my friends who gave me the necessary distractions from my research and made my stay in Champaign memorable.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ARPA-E	Advanced Research Project Agency-Energy
CAD	Computer Aided Drawing
CMU	Carnegie Melon University
COM	Center of Mass
CSS	Corn, Sorghum and Soybean
	HIL Hardware In the Loop
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
EKF	Extended Kalman Filter
FCN	Fully Connected Network
GNSS	Global Navigation Satellite Systems
GPS	Global Positioning System
KF	Kalman Filter
PWM	Pulse-Width Modulation
RANSAC	Random Sampling Consensus
ROI	Region of Interest
ROS	Robot Operating System
PF	Particle Filter
PLA	Polylactic Acid
RGB	Red Green Blue(color-space)
RTK	Real Time Kinetic

UAV	Unmanned Aerial Vehicle
USB	Universal Serial Bus

LIST OF SYMBOLS

C_r	point cloud to the right of COM at time step k
C_l	point cloud to the left of COM at time step k
L_l	left estimated line
L_r	right estimated line
b_l	y intercept of L_l
b_r	y intercept of L_r
a	slope of the parallel lines L_l and L_r
$L_{l,k-1\|k-1}$	initial state of the left line
$L_{r,k-1\|k-1}$	Initial state of the right line
$L_{l,k\|k-1}$	predicted state of the left line
$L_{r,k\|k-1}$	predicted state of the right line
$L_{l,k\|k}$	updated state of the left line
$L_{r,k\|k}$	updated state of the right line

CHAPTER 1

INTRODUCTION

There has been an increased focus on automation in the agricultural industry for the past several years, predominantly on the expensive and enormous farm machines that sow, fertilize, and harvest commodity crops such as wheat, corn, and cotton. Specialty crops such as strawberries, broccoli, and cabbage have recently become the target of robotics researchers and startups due to the continued increase in labor costs in conjunction with the decrease in labor availability, though harvesting these crops usually requires fine manipulation tasks, and are therefore more difficult to automate.

Although the interest in smaller robots for precision agriculture has increased over the past decade, the current cost of production and implementation, unfortunately, prevents these systems from becoming more than just a line in a massive manufacturer's R&D budget [2]. Small, compact and agile ground robots are currently more useful for plant scientists and breeders, as their test and breeding fields tend to be orders of magnitude lower than typical large commercial tracts that dominate nearly 40% of land in the U.S. [1].

Reliable autonomous navigation in cluttered and unstructured canopies of current agricultural fields, where GNSS signals are unreliable and are the key to enabling low-cost, light-weight, and compact agrarian robots. This scenario is especially real for small robots that are designed to go under the canopy to obtain critical measurements of plant characteristics or perform other management activities. Present generation large and massive farm vehicles that have undergone automation operate over the canopy of the crop and use RTK GPS to navigate through the fields. Autonomous navigation is then achieved merely via waypoint following algorithms since GNSS is a reliable source for navigation above the crop canopy. However, GNSS signal strength can be unreliable under the canopy due to multi-path errors and signal attenuation.

In this thesis, we address the problem above of under-canopy autonomous navigation in unstructured modern agricultural fields. We present the development and implementation of several methods for localizing the robot globally and locally within crop rows (formed by planting crops in straight parallel lines), and navigating them accordingly. The methods described in this thesis, autonomously detect these rows and use adaptive guidance logic

to navigate the robot through unstructured fields, enabling it to collect data as it drives through the rows while maintaining a fixed distance to one of the sides. The perception and navigation stack described in this work also enables the robot to detect the end of the row and guides it to move into the next row, allowing the system to be completely autonomous.

1.1 Motivation

1.1.1 TERRA Project

Sorghum, which is 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, such as biofuel, fodder, and even alcohol for human consumption [2]. Consequently, U.S. Department of Energy ARPA-E [1] began the TERRA program 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 [3].

1.1.2 Plant Breeding and Phenotyping

The traditional approach to plant breeding involves growing a large number of plant breeds with a wide variety of traits and subsequently selecting plants that exhibit desirable characteristics, and interbreeding those varieties to produce new generations with a combination of these desirable characteristics.

Plant breeding in the 21st century is in many ways similar to the traditional process. More importantly, as a result of the genomic revolution, the past ten years has seen a five order of magnitude decrease in the cost of gene sequencing, significantly increasing the quantity of genetic data available to plant scientists [4]. In contrast, the cost and speed of detecting and quantifying the expression of genetic traits have remained unchanged, and as a consequence plant scientists cannot take the full advantage of the more comfortable access to genetic data.

To efficiently research successful genetic traits, plant scientists need to map the physiological and morphological expressions (phenotypes) of a plant genotype to specific genetic

[1]The information, data, or work presented herein was funded in part by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Award Number DE-AR0000598. The views and opinions of author expressed herein do not necessarily state or reflect those of United States Government or any agency thereof.

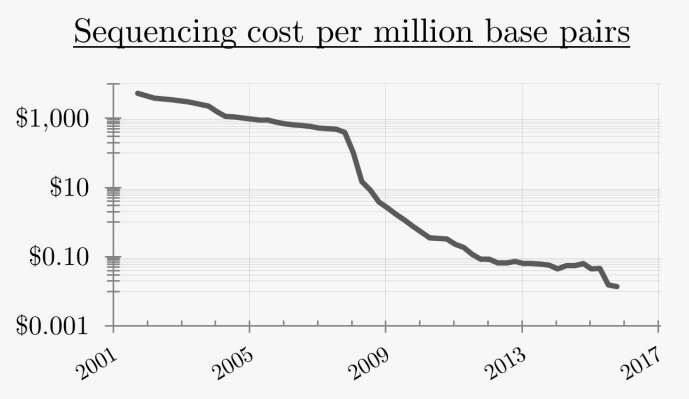


Figure 1.1: Gene sequencing costs from 2002 to 2016 [1]

markers in the plant genome. If they are successful in establishing the relationship between the phenotypes and genotypes, they reduce the need for a large number of iterations while performing phenotyping experiments to achieve desirable characteristics. Unfortunately, the rate at which plant scientists measure and analyze phenotypes is significantly lower than the speed of research. This bottleneck is well-recognized by the research community, which has termed it Phenotyping Bottleneck [4].

Research has shown that plants of the same genotype express phenotypic variation when grown in the field as compared to a greenhouse setting. The end goal of a typical breeding program is to develop seeds that will be used on farms; it follows that breeding trials need to be performed in the field under expected environmental conditions to ensure the success of the program [5].

Collecting phenotypes in the field is an arduous task, performed primarily by the plant scientists and graduate students under hot, humid and pest-ridden conditions using rudimentary tools such as calipers and yardsticks. Over the past ten years, a wide array of platforms have been developed to automate and expedite the extraction of phenotypic information from plants grown in field, including blimps [6], UAVs [7], over-row tractor platforms [8] [9], and gantry systems [10]. Unfortunately, none of these platforms have been able to provide data from beneath the crop canopy of a tall crop like bio-energy sorghum during the late season.

1.2 Related Work

Lightweight and compact agricultural ground robots address the critical niche between the heavy and over-sized agricultural equipment used in the modern day farms and human utilized tools. They enable versatile applications in modern farms and enable location-

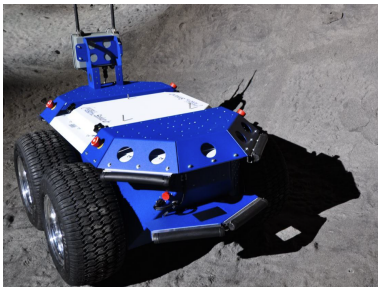
(a) Clearpath Husky [15]

(b) Robotnik Summit XL [16]

(c) The Robotanist [2]

(d) Wine Robot, Naio Tech. [18]

(e) QinetiQ TALON [19]



(f) Omron Technologie Inc. Seekur Jr. [20]

Figure 1.2: Commercial ground robot platforms

specific management and remote-sensing of crops. As such there have many ground robots that have been investigated for agricultural purposes. Robotanist [2] is used for autonomous phenotyping, Hall et al. introduces weed scouting robot AgBotII [11], Thorvald platform [12] is designed to be a multi-purpose ground robot for seeding, weeding, and harvesting. Haibo et al. present a wheat precision seeding robot [13], and Torgersen [14] provides in-depth details for the design of the NMBU Mobile Robot. ClearPath Robotics Inc. has developed a range 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 and navigating in dense forests [15]. Robotnik S.L.L has developed several ground robots that have been used to deploy sensors for precision agriculture, including the Guardian and Summit XL [16]. Rowbot Systems has developed a platform that is designed to travel between crop rows autonomously [17].

A critical difference between the robots above and the one considered for this work, is that the latter is an order of magnitude lighter(just under 7 kg) compared to the former ones (over 100 kg). While a plethora of ground robots are available in the market, no other meet the specific operative, feature, and performance requirements of the platform, nor do they offer the modularity that will enable this platform to be successfully employed throughout the life of the project.

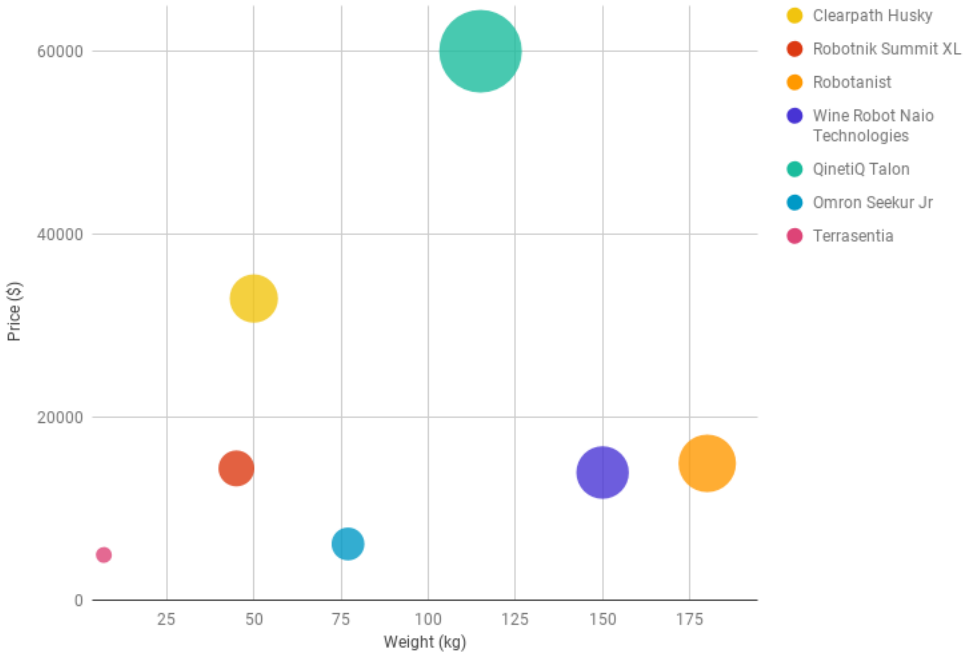


Figure 1.3: Weight-price comparison for different mobile robots available in the market with Terrasentia

A GNSS system, typically augmented with RTK differential correction, can provide highly accurate latitude and longitude information provided it has an unobstructed view of GNSS satellites. As was mentioned earlier, GNNS-based navigation systems have been used for guiding autonomous agricultural tractors([21], [22], [23], [24]) and even smaller autonomous over the canopy robots([25], [26]). However, GNSS receivers suffer from occlusion, attenuation, and multi-path errors under dense crop canopies. Additionally, GNSS based navigation unless augmented with 2D/3D LiDAR, cannot detect obstacles such as animals, humans, stubble, or other farm machines and hence provide automation, and not true autonomy. In conclusion, complete and safe autonomy in agricultural environments cannot be achieved with GNSS alone. Moreover, Rovira Mass et al. highlights that long-term navigations solution stability in agricultural fields cannot be granted as multi-path reflections and signal-attenuation are unpredictable, requiring redundancy and continuous fail-sake checking [27]. In conclusion, the aforementioned works may not have poor performance as a result of the said GNSS issues as these platforms are big enough to have the GPS receiver well above the canopy, the same cannot be expected when the vehicle goes under the canopy, such as when operating in late season corn, soybean, and sorghum as in this thesis.

Localization and navigation in unstructured environments have been the focus of significant prior research, including agricultural environments. Traditionally, LiDAR has been

used in indoor applications, e.g., industrial and warehouse automation, the past decade has brought down the cost of the LiDARs which has sparked interest in using such technology in new areas such as real-time obstacle avoidance and navigation for mobile robots such as autonomous cars, UAVs and agricultural robots. One research group from CMU used a monocular RBG camera to localize and navigate between rows of an apple orchard [28]. The same group from CMU also investigated using a 3D point cloud captured utilizing a LiDAR sensor to again localize and navigate within an apple orchard. LiDAR has the advantage of directly providing the robot with distance measurements, being less sensitive to the light conditions of the environment and possessing an extended range compared to other sensors. But when ground vegetation occludes tree trunks, Zhang et al. use a rotating 2-D LiDAR to retrieve a 3-D point cloud and then perform localization and navigation [28]. Similarly, Bell et al. use a Velodyne VLP16 to handle sparseness in structured pergola orchards [29].

Nevertheless, CSS crops significantly differ from orchards. Instead of well-defined tree trunks in groves, weeds and hanging leaves can be easily mistaken with or even occlude the stalks in these crops. Moreover, the standard spacing between the tree rows is a few meters in orchards whereas it is around 0.8m in CSS crops. Hence, it is easy to find a dense and cluttered environment in such crops, especially in later growth stages.

Hiremath et al. proposes a Particle Filter (PF) based navigation algorithm using a LiDAR as the primary sensor [30]. The PF can handle various uncertainties, for example, noise from uneven terrain and the varying colors, shapes, and sizes of the plants, and accurately estimates the robot state in the navigation frame, such as the heading and lateral deviation. However, this method is limited to crops smaller than 0.6m, which means that it does not consider the later growth stage of CSS crops which results in substantial occlusion by leaves and clutter.

Troyer et al. demonstrates a row-following platform using two single reading LiDAR sensors [31]. It showed reasonable behavior in the indoor case, but the system fails to perform well even on simulated cornrows(with PVC pipes as corn plants). Using only two distance measurements, it will be tough to navigate the cluttered environments presented by CSS crops.

Velasquez et al. present a LiDAR-based row following algorithm which is tested on simulated corn rows(potted corn plants) [32]. To generate the distance to rows, it considered LiDAR readings inside a reasonably significant ROI in front of the robot without checking if such points pertained to the row. Thus, it was prone to errors due to hanging leaves, weeds or even unexpected objects such as a person walking in the neighboring lane.

In summary, navigating rows in modern row-crop agricultural environments considered in this thesis remains a challenging problem due to the following open challenges:

- Lack of reliable GNSS under canopy: The canopy coverage greatly affects the signal reception by onboard GPS receiver due to issues of occlusion, multi-path errors and signal attenuation. For navigation purposes in crops with the standard lane width of 0.8m(30in), a navigation system with the error greater than a few centimeters may not guarantee crop safety against collisions;

- Highly cluttered rows: Presence of weeds, leaves and even fallen stalks presents the robot with a highly dynamic environment that can be erroneously perceived as part of the crop rows. Moreover, such elements may restrict movement;

- Varying plant spacing and mildly varying lane width: Although the seeding process can happen with centimeter-level accuracy and the seeds form equally spaced straight lines, some of them may not germinate leaving gaps, and the stalks may not grow straight up which leads to lodging. Thus, the lane width may not be constant through the row and might change depending on how the plants have grown over the season;

- Frequent occlusion of LiDAR: LiDAR provides information about the distances to different objects in the surrounding. Thus its occlusion hinders the robot's capability of knowing its surroundings and therefore avoiding a collision. For example, overhanging leaves that cover the sensor will cause sudden changes in distance measurements, greatly affecting the quality of available information (Fig. 1c); 2-D point cloud: Single layer LiDAR sensors are limited to a 2-D set of distance measurements. Therefore, there is a huge loss of details, which for instance could be used to detect unwanted features such as weeds and leaves.

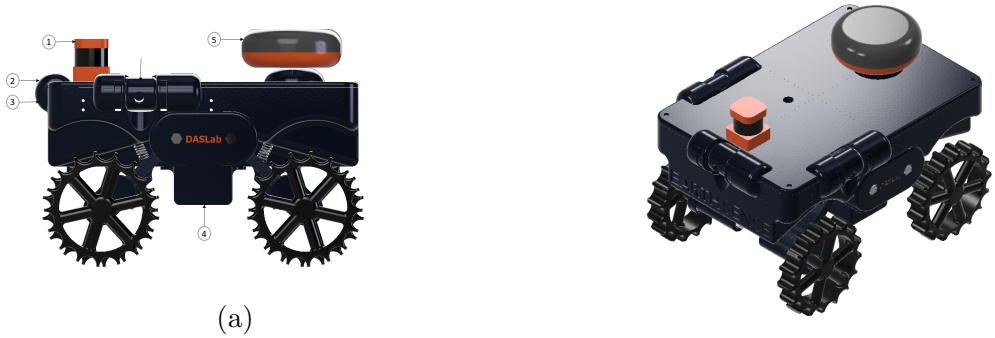
1.3 TerraSentia Robot

This section describes the low cost, ultra-compact(0.35 m wide) and ultra-light(6.6 kg) 3D printed agriculture robot TerraSentia [33]. It is a novel robot conceived to automate the measurement of plant traits for efficient phenotyping. Because of its design, it does not permanently damage plants even if it drives over them. However, having damaged plants is not a desired situation, and hence such platform requires mechanisms capable of keeping it in the clear paths. In this sense, the autonomous navigation system described in this work allows the navigation within crop rows without hitting plants, and subsequently, it is tested using the mentioned robot. Fig. 1.3a shows the CAD drawing of the robot with a

selection of available sensors and Fig. 1.4b depicts the isometric view of the system. Figure 1.3 compares the cost and weights of different autonomous platforms available in the market with Terrasentia.

A brushed 12V DC motor with a 131.24:1 metal gearbox (Pololu Corporation, USA) drives each of its four wheels - a two-motor set for left and another for the right side. Each two-motor set is driven by one of the Sabertooth Motor Controller channels, whose nominal supply current is 12A per channel. Furthermore, a two-channel Hall-effect encoder (Pololu Corporation, USA) provides 64 counts per revolution for each motor, enabling wheel velocity measurement. The self-tuning PID controller Kangaroo x2 Motion Controller(Dimension Engineering, USA) uses this information to maintain desired angular, and linear velocity commands.

In contrast with common practice in building agricultural robots ([25], [13], [12], [2]), most of the mechanical parts, including wheels, are built through additive manufacturing with PLA. The capability of controlling infill percentage enabled optimal stress distribution throughout the structure, resulting in overall ultralight robot while preserving strength and durability.

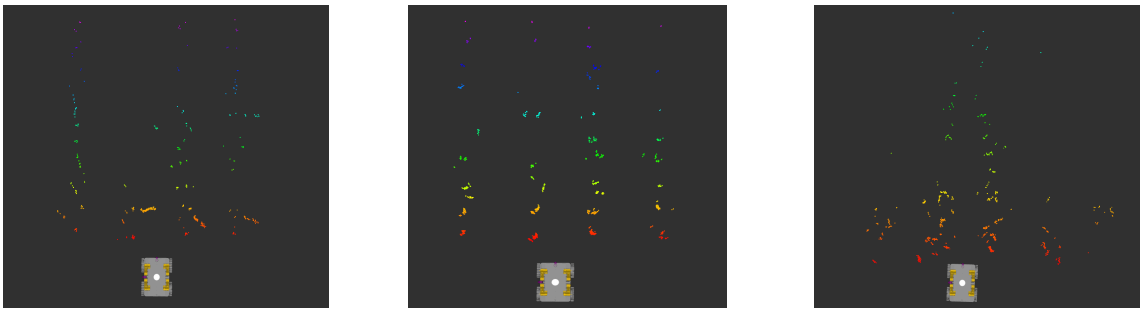


(a) (b)

Figure 1.4: TerraSentia CAD drawings with a suite of sensors. a. 1. Hukuyo UST-10LX LiDAR sensor, 2. Embedded visual sensor, 3. 2-axes camera gimbal, 4. Li-Ion Battery Bay, 5. GNSS receiver b. Isometric view of the robot

LiDAR sensor needs to be chosen based on the angular resolution, range and update rate required. Consider a stalk with a nominal width of 0.01 m. This is a reasonable minimum width for corn and sorghum, and for a robot in the middle of a 0.8 m lane width, sensor measurements of stalks are in the range 0.2 m to 0.6 m for the next sidewards readings. In this scenario, a 0.47 angular resolution provides at least two readings from the nearest stalks. In cluttered environments, a sensor would be able to get distinguishable information for only the next few meters. Nevertheless, an extended distance range would be useful for row end

maneuvers. Finally, the robot would traverse the crop with speeds not exceeding 1 m/s. At this velocity for a robot in the middle of the lane with a heading error of 30 relative to the rows, it takes around 0.6 s to hit a row. Therefore, it is reasonable to expect that sensor updates its readings at least in the order of 0.1 s. Such sensor should be suitable for outdoor applications. A sensor that complies with discussed requirements is Hokuyo UST-10LX, a 2-D LiDAR sensor which covers a 270 range with 0.25 angular resolution, maximum distance reading of 30 m and 40 Hz update rate. Anecdotal experiments show that the under USD 2K o-the-shelf Hokuyo UST-10LX 2-D LiDAR provides similar results with the proposed autonomy system. Although a 3-D LiDAR sensor would give a lot more information, its price (at least USD 8K) and computational toll required for processing incoming data make 3-D LiDAR sensor less suitable for a small and low-cost robot as TerraSentia.



(a) (b) (c)

Figure 1.5: Comparison of the lidar measurements in a Sorghum field across different seasons: (a) Early Season (b) Mid Season (c) Late Season

Given the single layer (2-D) nature of LiDAR Hokuyo UST-10LX's point-cloud, the LiDAR's vertical position must be such that the reading layer intersects the crop rows. Moreover, the sensor and other devices should be placed carefully to ensure the 270 angular field-of-view of the LiDAR is not obstructed. These are the only two constraints in positioning the LiDAR. Other than these constraints, the position of the LiDAR in front of the robot does not affect the algorithms presented here. For this work, the sensor is placed in the front part of TerraSentia: In the center of the robot's traverse axis for symmetry, 0.15 m to the front of the robot to avoid reading obstruction and 0.13 m above the soil.

An embedded 64bit minicomputer (1.2GHz quad-core Raspberry Pi 3 Model B) handles measurement acquisition from sensors and generation of output signals to other embedded devices. In particular, Raspberry Pi 3 acquires distance readings from Hokuyo UST-10LX through USB2.0 using SCIP2.0 protocol, runs the proposed perception subsystem and generates desired control signals (i.e., desired angular and linear velocities), sent to Kangaroo x2 Motion Controller as two PWM signals.

1.4 Simulation Environment

(a) (b)

Figure 1.6: Comparison of the simulation environment and the real world: (a) Terrasentia in the Gazebo simulation environment (b) Terrasentia in a corn field

To enable fast development of navigation and control software, a simulation environment was developed in Gazebo alongside ROS [34]. This not only expedited the development of perception and navigation algorithms but also helped train the vision algorithms, as the simulated environment closely matches the real fields, as shown in Fig. 1.6. A detailed description of the simulation environment is presented in Appendix 1 for completeness.

1.5 Objective and Approach

The objective of this research is to develop a robust perception and navigation system for the Terrasentia robot, discussed in the previous section so that it is capable of autonomously traversing a typical CSS crop breeding plot, planted at a row spacing of about 0.8m or greater from the initial planting through emergence of seedling until late season immediately prior to harvest.

1.6 Overview

This thesis is organized as follows; in Chapter 2 we discuss various methods for localizing the robot, both globally and concerning the rows, along with the results obtained during testing. Chapter 3 discusses the autonomous navigation architecture along with the perception and navigation algorithms that were realized as part of this research and the results obtained

during testing. 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

STATE ESTIMATION AND LOCALIZATION

Reliably traversing through the CSS crop fields can be a challenging task, even for a human on foot. At some points in a particularly dense row, one can no longer rely on their eyes to show the path; one instead has to push through the overhanging leaves and clutter under the assumption that the rows as locally planted straight. Unfortunately, the fields in which the robot navigates are continually growing and changing − we need a variety of navigation algorithms to choose from using different sensor modalities to be robust to this dynamic environment. In this chapter, we describe the development and implementation of a method for accurately estimating the robot state through different parts of the field.

2.1 State Estimation using Extended Kalman Filter

The EKF is ubiquitous in 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 [35].

The states of an actual robot operating in the real world, such as its position and orientation, can never be entirely known because physical systems are not perfect. Instead, it is more useful to think of the robot existing in possible states, *probabilistically*. The Kalman filter models these states using a Gaussian distribution where state uncertainty arises in part due to process noise, measurement noise, and inaccuracies in the dynamics model of the system, to name a few.

Using data from an array of sensors on the robot, 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 the final pose with more accuracy and certainty that would have been possible using any individual sensor [35].

2.2 Sensor Data

Wheel encoders provide a reasonable estimate of linear velocity, but the very nature of skid-steering platform requires the 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 decent indication of the real-world movement of the robot, as we lose the ability to accurately relate the rotational velocity of the robot with the linear and angular velocity of the robot induced by that wheel. As a consequence, localizing only via wheel encoders is only accurate until the wheels slip and the odometry estimate drifts rapidly after the first slip is encountered.

However, the RTK GPS fixed on the top of Terrasentia, on the other hand, provides a globally accurate position of the antenna of the robot and can accurately localize the robot in the global frame, as long as the antenna is over the canopy. This is usually possible early during the planting season, but the accuracy of RTK GPS deteriorates very quickly as soon as the robot has to go under the canopy of the crops, due to multipath errors and signal attenuation. Moreover, GPS cannot provide orientation data of the robot unless it is mounted at an offset from the center of mass of the robot, in which case the orientation becomes observable and can be predicted using EKF. GPS errors tend to manifest as discrete jumps in the position that makes finding the exact orientation of the robot for navigation difficult.

The onboard IMU provides us with three crucial pieces of information regarding the state of the robot, namely, the angular velocities along the three axes from the rate gyroscope, angular acceleration from the accelerometer and the direction from the magnetometer at a very high update rate($> 100Hz$). The accelerometer data provides globally reference angular acceleration in the yaw, pitch and roll axis, which can be integrated to obtain the angular velocity and also relative position of the robot, though significant noise limits the usefulness of dead reckoning system solely using an accelerometer. The gyroscope provides with globally referenced angular velocities in yaw, pitch and roll axis and can be integrated to provide relative position of the robot, and is frequently used in conjugation with the accelerometer to maintain a high level of confidence of the roll, pitch, and yaw angles of the robot in the short term, but is unfortunately susceptible to drift over more extended period of time. However, this drift can be corrected for by using sensor error models and estimating the bias via the EKF. The magnetometer is 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. However, magnetometers are unreliable as their accuracy depends on the magnetic field around the IMU, and since the

IMU is mounted at the very center of the robot, it experiences magnetic interference caused due to DC current passing through various electric wires around it. Due to this reason, the utility of magnetometer is limited, and it is not used for predicting the orientation of the system.

Each of these sensors provides us with an estimate of the robot position in the global frame or the local navigation frame with varying levels of confidence for different parts of our final pose estimate. All the sensor data can be fused together using a Kalman Filter to provide a more accurate estimate of the pose of the robot that can be obtained by any sensor individually. Using an EKF 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-linearized system via a first-order Taylor series expansion is sufficiently accurate, and all noise in the system is Gaussian in nature.

2.3 Kinematic Equations for a Skid Steering Robot

As was discussed in the previous section, very accurate wheel position data is provided by the Hall effect 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. However, due to slipping, uncorrected drift in position and yaw accumulates over time producing highly erroneous results. Using the kinematics of a skid-steering robot, we can integrate this wheel position data to recover the absolute position and heading in the global coordinate frame as well as the wheel linear and angular velocities.

The encoders on each side of the robot provide position data, and since the left and right wheels are not mechanically linked, the position data is first averaged, which gives us the position estimate of the robot in the world frame. One possible solution to correcting the inaccuracy in yaw caused by slipping is to weight the change in yaw and yaw rate according to the current instantaneous center of rotation. In this case, however, a constant multiplier to the yaw rate, λ, determined experimentally was used. The eventual value obtained for λ was 0.12.

Using 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 (2.1), (2.2), (2.3). Since the robot is non-holonomic, the robot cannot use its actuators to translate along the y-dimension. Therefore Δy_b will always be zero.

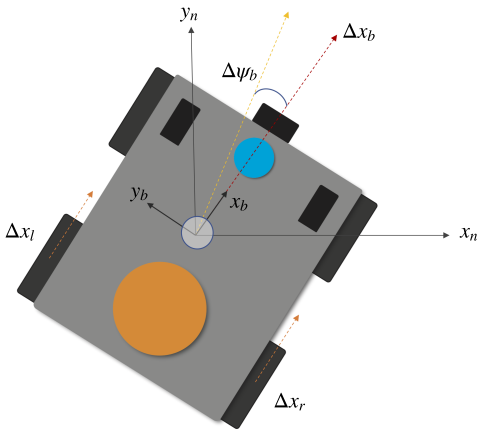


Figure 2.1: Deriving the skid steer kinematics for Terrasentia robot

$$\Delta x_b = \frac{\Delta x_l + \Delta x_r}{2} \tag{2.1}$$

$$\Delta y_b = 0 \tag{2.2}$$

$$\Delta \psi_b = \frac{\Delta x_r - \Delta x_l}{b} \cdot \lambda \tag{2.3}$$

Using the change in time between the wheel position measurements Δt we can approximate the velocity and yaw rate of the robot over that interval, as shown in equations (2.4) and (2.5).

$$\Delta v_b = \frac{\Delta x_b}{\Delta t} = \frac{\Delta x_l + \Delta x_r}{2 \cdot \Delta t} \tag{2.4}$$

$$\Delta \omega_b = \frac{\Delta \psi_b}{\Delta t} = \frac{\Delta x_r - \Delta x_l}{b \cdot \Delta t} \cdot \lambda \tag{2.5}$$

Finally, equations (2.1) $-$ (2.3) need to be transformed from the robot's frame of reference of the navigation frame (x_n, y_n and ψ_n). This is performed by moving the robot forward along an arc described by the forward velocity and yaw rate of the robot, in the navigation frame. This is shown in equations (2.6), (2.7), (2.8), where the ratio $\frac{v_b}{\omega_b}$ is known as the instantaneous radius of path curvature. These equations become computationally unstable as the yaw rate approaches zero, therefore in the current implementation we perform a check to use a different set of equations if the yaw rate is under a certain ϵ, as shown in equations

(2.9), (2.10), (2.11). The pose of the robot is updated every time sensor data is read from the encoders, i.e., at about 50 Hz on Terrasentia.

$$x_{n(i)} = x_{n(i-1)} + \frac{v_b}{\omega_b}(\sin(\triangle\psi_b - \psi_{n(i)}) - \sin(\psi_{n(i)})) \tag{2.6}$$

$$y_{n(i)} = y_{n(i-1)} + \frac{v_b}{\omega_b}(\cos(\triangle\psi_b - \psi_{n(i)}) - \cos(\psi_{n(i)})) \tag{2.7}$$

$$\psi_{n(i)} = \triangle\psi_b + \psi_{n(i-1)}) \tag{2.8}$$

$$x_{n(i)} = x_{n(i-1)} + \triangle x_b \cos(\psi) \tag{2.9}$$

$$y_{n(i)} = y_{n(i-1)} + \triangle x_b \sin(\psi) \tag{2.10}$$

$$\psi_{n(i)} = \psi_{n(i-1)} + \triangle\psi_b \tag{2.11}$$

2.4 Extended Kalman Filter

EKF linearizes the dynamics of the system by using first order Taylor expansions, whereas Kalman filter uses linear kinematics. Therefore, we will briefly discuss Kalman filter before discussing EKF. 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 covariance. Consider the following system, where $F(k)$ determines 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.

$$x(k+1) = F(x) \cdot x(k) + G(k) \cdot u(k) + v(k) y(k) = H(k) \cdot x(k) + w(k)$$

Un-modeled disturbance such as slipping of wheels or friction that affects the system dynamics is accounted for by Process noise. In the above equations, we have known input $u(k)$ and known output $y(k)$, and given these, Kalman filter determines the best estimate of the state at the next time step, given a previous estimate of the state. The derivation of

the following summary of Kalman filter is left to the reader and can be found in [35]. In the following equations. P(k) is an estimate of the error covariance, and $W(k)$ is the covariance matrix of the sensor noise.

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

The equations above make sure that the expected value of the error between $x(k)$ and $\widehat{x}(k|k)$ 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 [35]. A summary of the EKF is shown below for completeness.

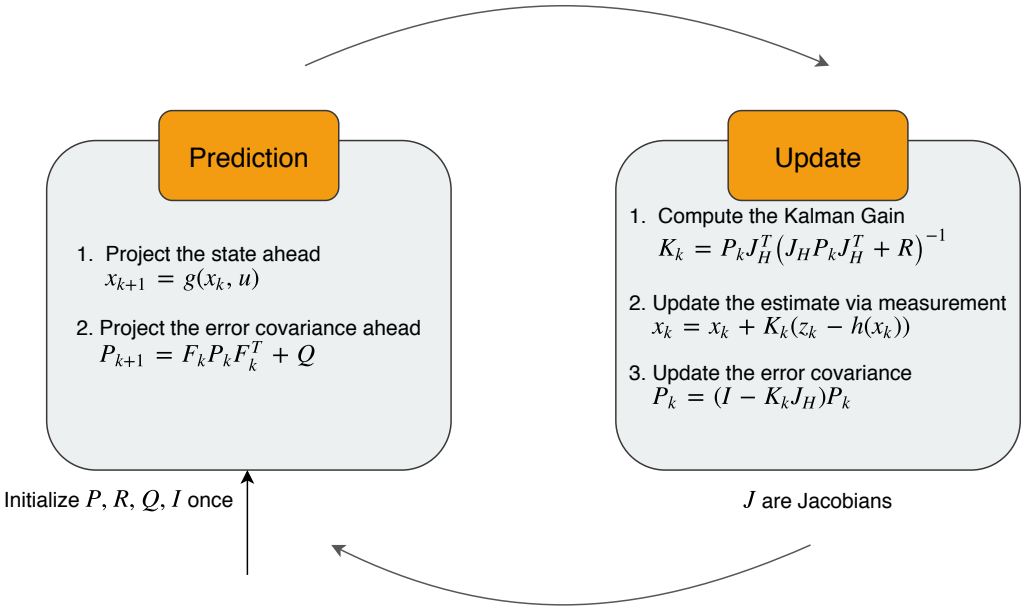


Figure 2.2: Illustration of the Prediction and Correction steps of EKF

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

2.5 Implementation and Results

The EKF algorithm is tested with a Trimble Copernicus II as the GPS and a Razor 9dof IMU which is mounted on a custom rig that aids in magnetic interference reduction and improves signal quality by considerably filtering out low-frequency noise. We have configured the algorithm to take into account the roll, pitch and yaw and their respective velocities from the IMU, velocity and heading rate is computed using the wheel encoders.

Since the GPS is mounted at a distance from the center of mass of the robot(where the IMU is mounted), we need to transform the position estimate from the GPS to the base frame of the robot, as shown in Fig. 2.1. Using the roll, pitch, and yaw(ϕ, θ, ψ) obtained in the previous section along with the GPS coordinates transformed to the UTM coordinate system, the transformation matrix (T) between the navigation frame (x_g, y_g, z_g) and the GPS coordinate frame $(x_{GPS}, y_{GPS}, z_{GPS})$ is computed using the yaw, pitch and roll (ψ, θ, ϕ) convention, as shown in Eq. (2.17) [36]. c and s designate cos and sin functions respectively, and x_0, y_0 and z_0 are the GNSS receiver offsets from the COM of the robot.

$$T = trans(t_{GPS}) \cdot R_z(\psi) \cdot R_y(\theta) \cdot R_x(\phi) \tag{2.12}$$

$$trans(t_{GPS}) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.13}$$

$$R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 & 0 \\ \sin\psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.15}$$

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.16}$$

$$T = \begin{bmatrix} c(\psi)c(\theta) & -s(\psi)c(\phi) + s(\theta)s(\phi)c(\psi) & s(\psi)s(\phi) + s(\theta)c(\psi)c(\phi) & t_x \\ s(\psi)c(\theta) & s(\psi)s(\theta)s(\phi) + c(\psi)c(\phi) & s(\psi)s(\theta)s(\phi) - s(\phi)c(\psi) & t_y \\ -s(\theta) & s(\phi)c(\theta) & c(\theta)c(\phi) & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.17}$$

The pose of the base frame of the robot is now determined by using the transformation computed in Eq. (2.12) along with the position of the robot base in GPS coordinate frame (r_p^{GPS}). Since the orientation of the GPS and the robot base are already aligned, this can be simply treated as transforming a point between two coordinate frames and applying the orientation of the GPS frame to the robot base frame as shown in Eq. (2.19) [36].

$$r_p^g = T \cdot r_p^{GPS} r_p^{GPS} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} \tag{2.18}$$

$$\begin{bmatrix} t_x + x_0 c(\psi)c(\theta) + y_0 \left(-s(\psi)c(\phi) + s(\theta)s(\phi)c(\psi)\right) + z_0 \left(s(\psi)s(\phi) + s(\theta)c(\psi)c(\phi)\right) \\ t_y + x_0 s(\psi)c(\theta) + y_0 \left(s(\psi)s(\theta)s(\phi) + c(\psi)c(\phi)\right) + z_0 \left(s(\psi)s(\theta)s(\phi) - s(\phi)c(\psi)\right) \\ t_z - x_0 s(\theta) + y_0 s(\phi)c(\theta) + z_0 c(\theta)c(\phi) \\ 1 \end{bmatrix} \tag{2.19}$$

For the current configuration of Terrasentia, the x_0, y_0 and z_0 were 0.1524 m, 0, and 0.0508 m respectively. Prediction equations are mechanized using the high rate sensor which is the IMU in this case, and the update part of the filter runs every time GPS data is updated($\approx 10Hz$). The EKF developed in the previous section was implemented from scratch in $C++$ using Eigen [37] and Boost [38] libraries for handling matrix operations and multi-threading respectively. Since the sensor data have different update rates, a blocking queue is used to solve the producer-consumer problem so that the sensor data is consumed by the EKF in the order it arrives in. Figure 2.3 and 2.4 show the comparison of the estimated position with the GPS coordinates. The data used here was obtained while traversing an uneven terrain outside DASLAB in Champaign, Illinois.

The localization error is shown in Fig. 2.5 which varies between $0 - 0.2m$ with a mean error of $0.098m$, which is within bounds for autonomous navigation. Figure 2.6 compares the other estimated states of the filter with the measured values and Fig. 2.7 shows the associated uncertainty associated with each state over 355 iterations.

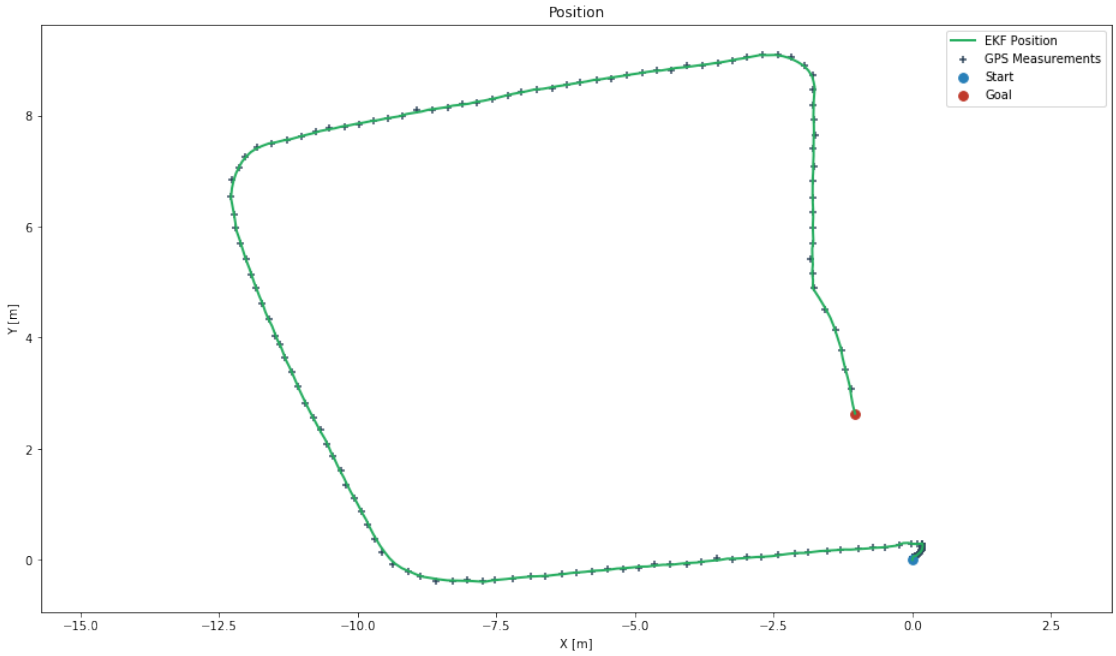


Figure 2.3: Comparison of estimated x and y position with the GPS coordinates – tested in the field

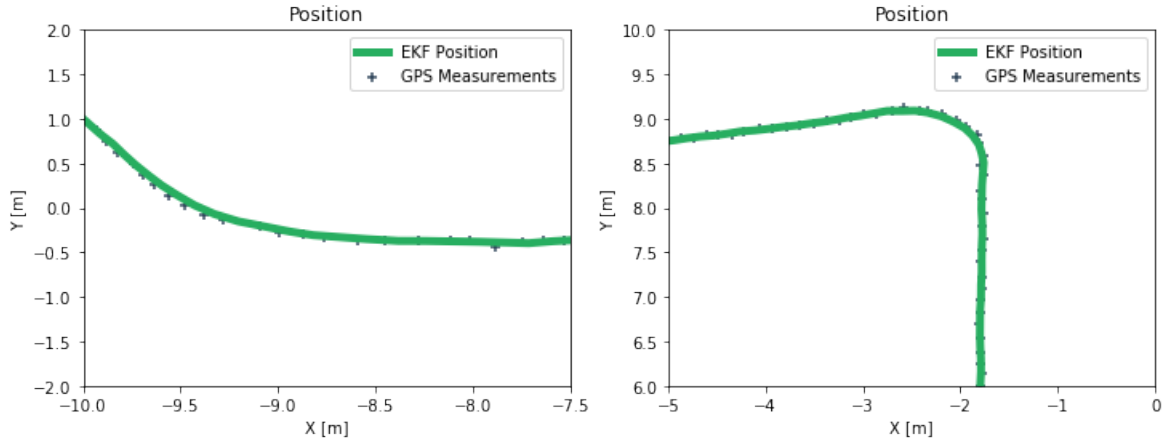
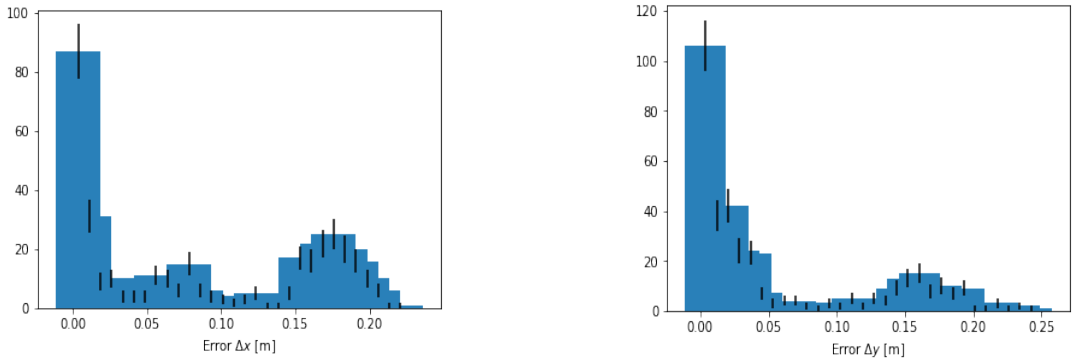


Figure 2.4: Detailed view of the estimated position with the GPS coordinates



(a) (b)

Figure 2.5: Comparison of estimated position with RTK GPS position: (a) Error in the estimated x position with RTK GPS position as reference (b) Error in the estimated y position with RTK GPS position as reference

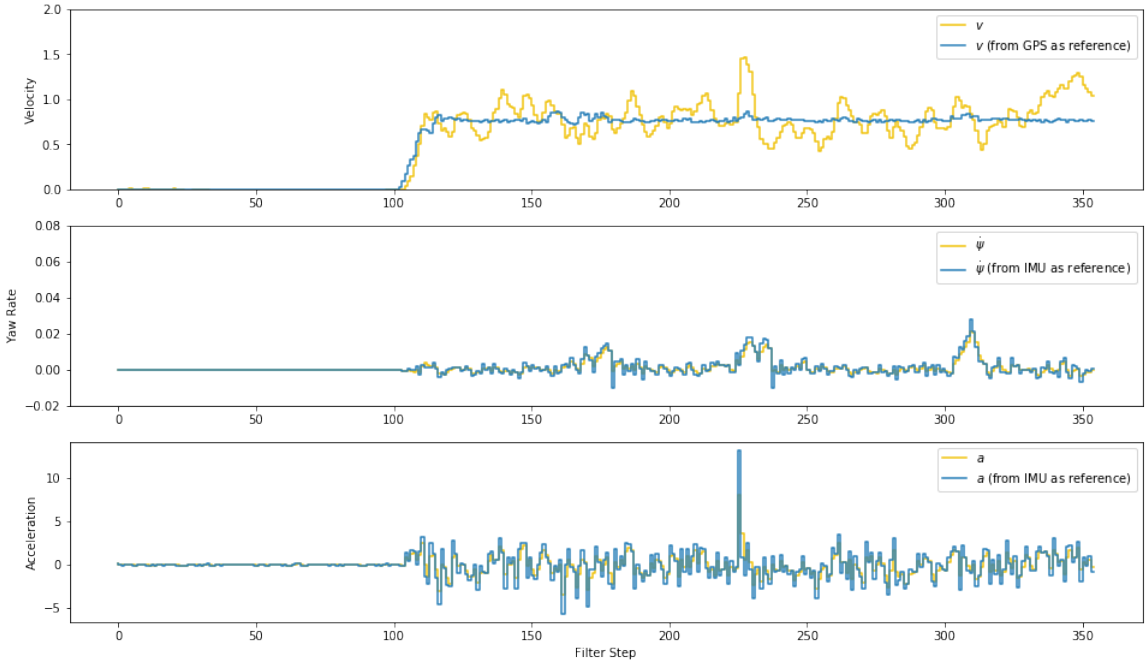


Figure 2.6: EKF state estimates

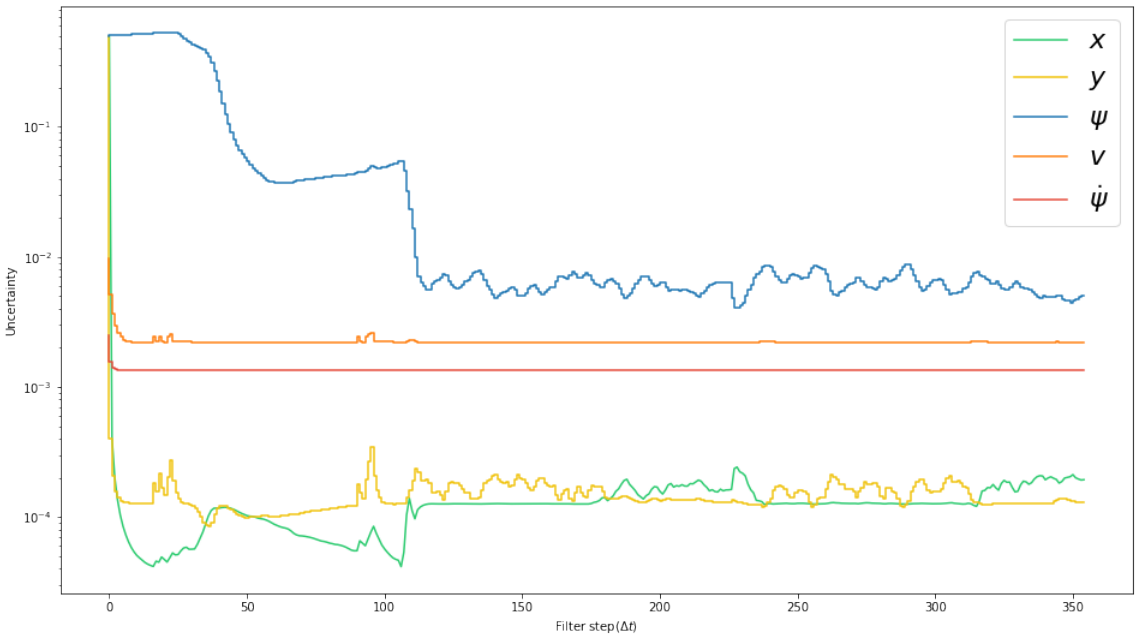


Figure 2.7: Uncertainty in state variables over 355 iterations of the EKF

CHAPTER 3

AUTONOMOUS NAVIGATION

As was mentioned in Chapter 1, reliable autonomous navigation through a typical CSS crop field is a challenging task. This can be attributed to two things: clutter due to overhanging leaves and fallen branches which makes the environment highly unstructured and dynamic and poor GNSS signal strength due to signal attenuation, occlusion and multi-path errors which make GNSS guided waypoint following unreliable. The previous chapter dealt with using an EKF to localize the robot in the field reliably, and this chapter describes the development and implementation of the autonomous navigation architecture along with several techniques to navigate either by following waypoints or by detecting the crop rows with the help of a LiDAR.

3.1 Autonomous Navigation Architecture

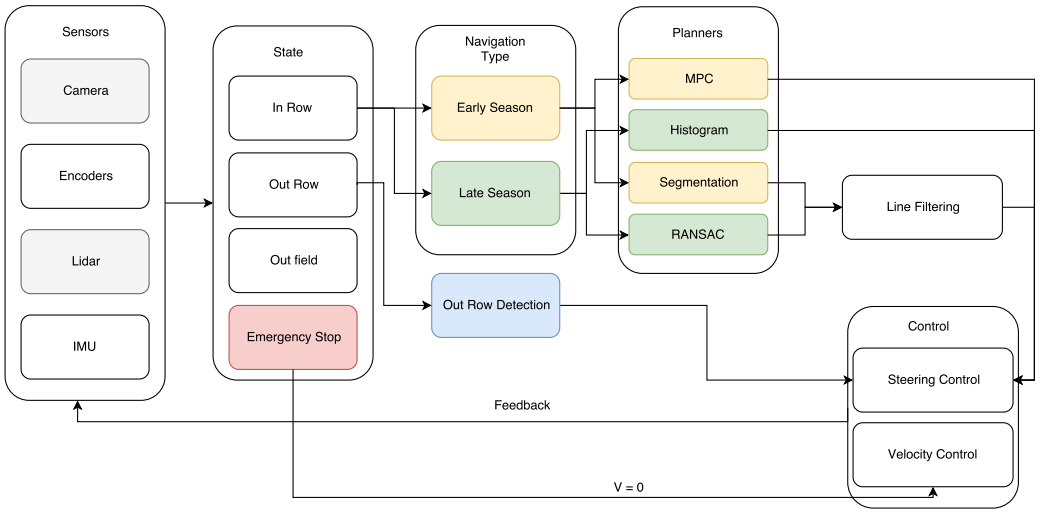


Figure 3.1: High-level Navigation Architecture

This section describes the high-level architecture which was defined for the navigation subsystem of the robot. The design takes into account the availability of GPS at different

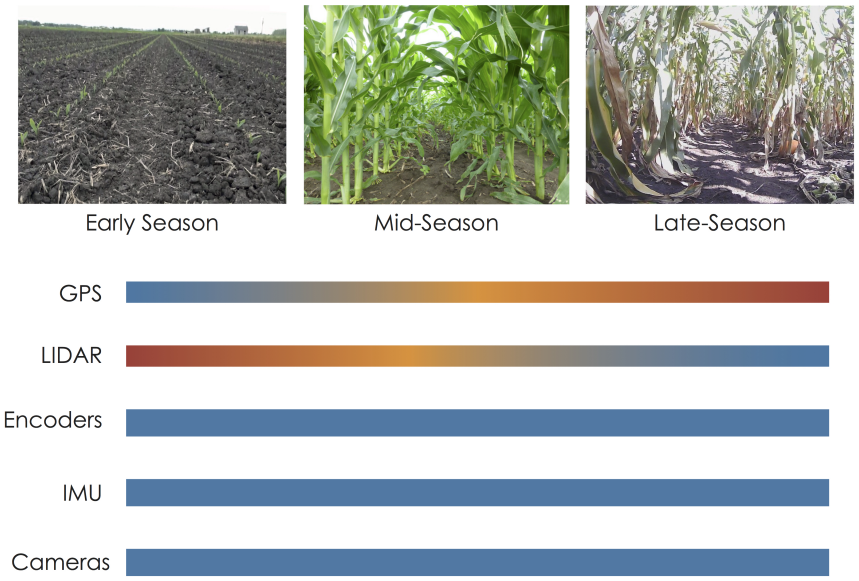


Figure 3.2: Utility of different sensors over the season

times of the season and considers the field to be divided into adjacent plots, with a minimum spacing of 0.72m between the plots. This was deemed necessary because it was essential to have a clear transition from one state to another, as the robot navigates through the field. Figure 3.3 shows the state machine where each behavior is modeled as a state that uses a specific set of algorithms for perception and navigation.

For autonomous navigation, the breeding season is classified into two categories: early-season and late-season. Early season is defined as when the plant height is lower than the height of the GPS antenna, and the GPS signals are reliable for waypoint navigation. Late season is described as when the plant height is higher than the height of the GPS antenna, and the GPS is unreliable for in row navigation. During late season, a combination of GPS waypoint navigation(in between plots) is used along with relative row following navigation for traversing rows. Figure 3.1 shows the architecture that was defined for the autonomous navigation through a CSS breeding field.

During late season, it is assumed that the robot starts in the row following state and is placed at the center of the row. Detection of point cloud by LiDAR and successful detection of parallel lines triggers the In-Row state of the system, and the robot begins traversing the CSS crop rows.

To autonomously navigate a CSS breeding plot, the robot not only has to traverse a row but also detect the end of the row and either take a turn into the adjacent row or navigate

to another row in the next breeding plot, based on the user settings.

In most modern breeding plots, the row width of all the rows is approximately the same with a variance of about $\pm 0.1\ m$. This is assumed to hold true for the end row turning algorithm described in this work.

As the robot navigates through the row, the perception system estimates the row width at each iteration and keeps track of the current average row width. Along with keeping track of the average row width, the robot also keeps track of the average number of points in the filtered point cloud scan, when it is traversing the row. The end-row state is detected using a simple heuristic rule. If the average number of points identified by the LiDAR scan decreases to $< 15\%$ of the value obtained during the In-Row state for a certain amount of time, it transitions to the Out-Row state.

If the robot is programmed to go into the adjacent row, it generates a trajectory online to navigate into the next row. However, if the robot is programmed to navigate to the row in the next plot, as shown in Fig. 3.4, it uses pre-defined waypoints along with the perception system to traverse between the plots. Since the GNSS signals are reliable in between two plots, the state estimation quickly converges, and waypoint following is used.

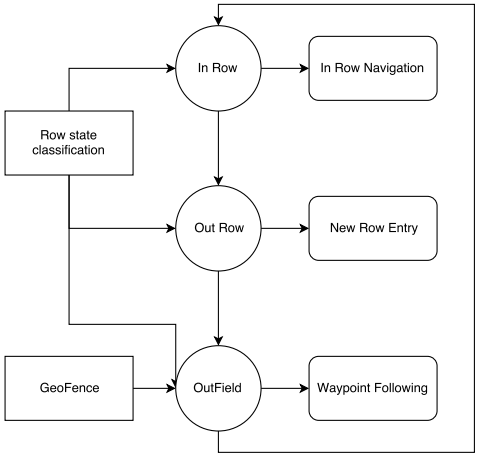


Figure 3.3: State Machine for modeling different Behaviors

3.2 Early Season Navigation

Since the crop height is lower than the RTK-GPS receiver, position data from the GNSS is used to navigate autonomously in the early season. Kayacan et al. develop a Safe Learning Terrain Parameters-Based Nonlinear Model Predictive Control (SLTPB-NMPC) algorithm for precise tracking of navigational waypoints in off-road terrain conditions [33]. A non-

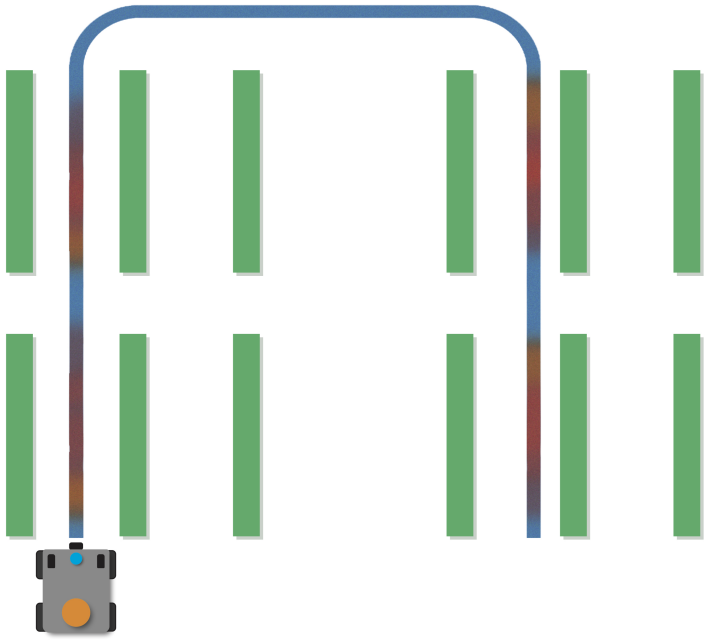


Figure 3.4: GPS signal strength as the robot navigates through the field as the season progresses, blue representing sensor reliability and red represents unreliability

linear moving horizon estimator is used to learn the terrain parameters using onboard robot sensors for precise tracking of waypoints. SLTB-NMPC algorithm is thus used for early season navigation and is mentioned here for completeness.

3.3 Late Season Navigation

3.3.1 Random Sampling Consensus

RANSAC is an iterative, non-deterministic algorithm that estimates parameters of a mathematical model from a set of observed data that may contain outliers when outliers are accorded no influence on the values of the estimates. RANSAC randomly selects points from the odometry corrected point cloud and fits two parallel lines, and then iteratively fits new lines and compares the new estimates with the previously estimated lines based on the distances of points.

RANSAC thus estimates a, b_r and b_l for the odometry corrected point cloud, which completely define the two parallel lines estimating the crop rows. The equations of the lines to be estimated can be written as follows:

$$
\begin{aligned}
L_l &: y = ax + b_l \\
L_r &: y = ax + b_r
\end{aligned}
\tag{3.1}
$$

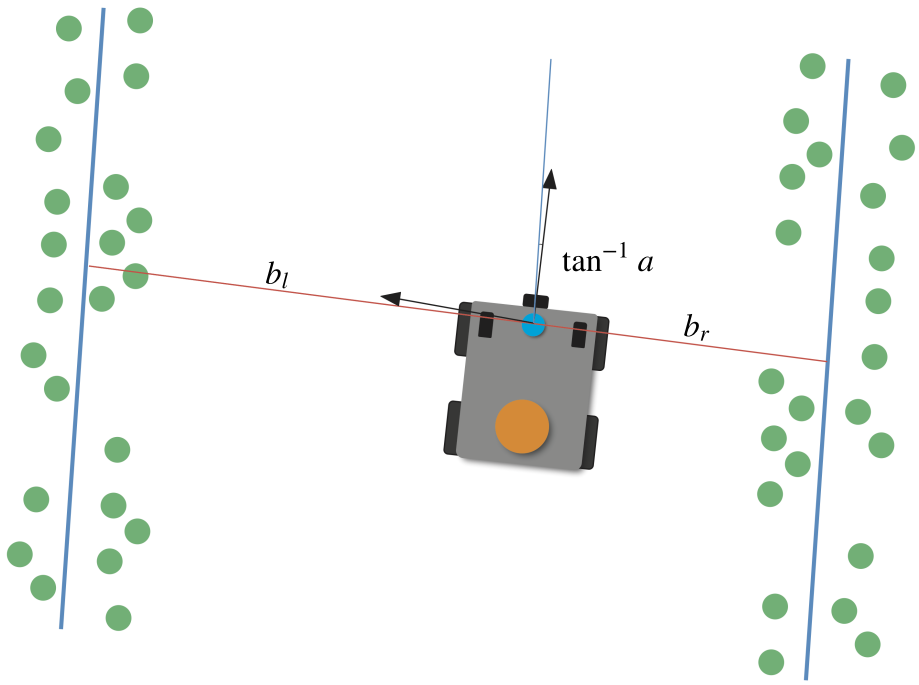


Figure 3.5: An illustration of the crop cloud points and the lines generated by RANSAC with the properties of the lines: a, b_l, and b_r

$$
R = \sum_{i \in C_l} (y_{k_i} - a x_{k_i} - b_l)^2 + \sum_{i \in C_r} (y_{k_i} - a x_{k_i} - b_r)^2 \tag{3.2}
$$

Equation 3.2 defines the function that is minimized, to estimate the best fitting lines on each side of the robot. To minimize Eq. 3.2, partial derivatives of R are taken with respect to a, b_l, b_r and set equal to zero.

$$\begin{aligned}\frac{\partial R}{\partial a} &= \sum_{i\in C_l} 2(y_{k_i} - ax_{k_i} - b_l)(-x_{k_i}) + \sum_{i\in C_r} 2(y_{k_i} - ax_{k_i} - b_r)(-x_{k_i}) = 0 \\ &= 2a\sum_{i\in C_l}(x_{k_i})^2 - 2\sum_{i\in C_l} x_{k_i}y_{k_i} + 2b_l\sum_{i\in C_l} x_{k_i} + 2m\sum_{i\in C_r}(x_{k_i})^2 \\ &\qquad -2\sum_{i\in C_r} x_{k_i}y_{k_i} + 2b_l\sum_{i\in C_r} x_{k_i} \\ \sum_{i\in C_l, C_r} x_{k_i}y_{k_i} &= a\sum_{i\in C_l, C_r}(x_{k_i})^2 + b_l\sum_{i\in C_l} x_{k_i} + b_r\sum_{iinC_r} x_{k_i}\end{aligned} \tag{3.3}$$

$$\begin{aligned}\frac{\partial R}{\partial b_l} &= \sum_{i\in C_l} 2(y_{k_i} - ax_{k_i} - c_l)(-1) + \sum_{i\in C_r} 2(y_{k_i} - ax_{k_i} - b_l)(0) = 0 \\ a\sum_{i\in C_l} x_{k_i} + b_l\sum_{i\in C_l} 1 &= b_l\sum_{i\in C_l} y_{k_i} \\ a\sum_{i\in C_l} x_{k_i} + b_l|C_l| &= b_l\sum_{i\in C_l} y_{k_i}\end{aligned} \tag{3.4}$$

$$\begin{aligned}\frac{\partial R}{\partial b_r} &= \sum_{i\in C_l} 2(y_{k_i} - ax_{k_i} - b_l)(0) + \sum_{i\in C_r} 2(y_{k_i} - ax_{k_i} - b_l)(-1) = 0 \\ a\sum_{i\in C_r} x_{k_i} + b_l\sum_{i\in C_r} 1 &= b_l\sum_{i\in C_r} y_{k_i} \\ m\sum_{i\in C_r} x_{k_i} + b_l|C_r| &= b_l\sum_{i\in C_r} y_{k_i}\end{aligned} \tag{3.5}$$

Here $|C_l|$ and $|C_r|$ represent the size of the left and right point clouds, respectively. Equations 3.3 to 3.5 can then be solved together for a, b_l and b_r. The final result is shown as follows:

$$a = \frac{\sum_{i\in C_l} y_{k_i} - b_l|C_l|}{\sum_{i\in C_l} x_{k_i}} \tag{3.6}$$

$$b_r = \frac{\sum_{i\in C_r} y_{k_i} - a\sum_{i\in C_r} x_{k_i}}{|C_r|} \tag{3.7}$$

$$b_r = \frac{\sum_{i\in C_r} y_{k_i} - \sum_{i\in C_r} x_{k_i}\frac{\sum_{i\in C_l} y_{k_i} - b_l|C_l|}{\sum_{i\in C_l} x_{k_i}}}{|C_r|} \tag{3.8}$$

Using Eq. 3.3 − 3.8 give us the following result for b_l.

$$b_l = \Big(\frac{\sum_{i\in C_l} y_{k_i} \sum_{i\in C_l, C_r} x_{k_i}^2}{\sum_{i\in C_l} x_{k_i}} + \Big(\frac{\sum_{i\in C_r} y_{k_i}}{|C_r|} - \frac{\sum_{i\in C_r} x_{k_i} \sum_{i\in C_l} y_{k_i}}{\sum_{i\in C_l} x_{k_i}|C_r|}\Big) \sum_{i\in C_r} x_{k_i} - \sum_{i\in C_l, C_r} x_{k_i} y_{k_i}\Big) \cdot \frac{1}{\frac{|C_l| \sum_{i\in C_l, C_r} x_{k_i}^2}{\sum_{i\in C_l} x_{k_i}} - \sum_{i\in C_l} x_{k_i} - \frac{\sum_{i\in C_r} x_{k_i} |C_l| \sum_{i\in C_r} x_{k_i}}{\sum_{i\in C_l} x_{k_i}|C_r|}} \tag{3.9}$$

RANSAC receives the odometry corrected point cloud and estimates the parallel line equation parameters. It performs 20 iterations wherein it divides the point cloud into left and right sides about the center of the base of the LiDAR, and selects random points from these two subsets.

As the points are selected, they are added to the set of left or right point clouds, and the summation values are logged such that the parameters could be estimated for the best fit parallel lines. If the points that are selected don't fit into the constraints set by either the left set or the right set, or at least one side does not have any points in the set, the algorithm breaks out of the loop. If this happens, a finite state machine sets the state of the machine to *turn mode*, which is discussed at the beginning of this chapter.

Once the first estimate for the left and right line equation parameters are estimated, the distance from each point in the left and right point clouds to the aforementioned best fit lines is calculated. The distance to each of the two lines is calculated as follows:

$$\begin{aligned} D_l &= \frac{-ax_i + y_i - b_l}{\sqrt{a^2+1}} \\ D_r &= \frac{-ax_i + y_i - b_r}{\sqrt{a^2+1}} \end{aligned} \tag{3.10}$$

Using these equations, the line distance is calculated and is then compared to the inlier threshold(which is determined experimentally), and if this distance is within the threshold, the point is added to the left(or right) inlier point cloud set. With the updated point cloud sets, new summation values are computed, and the line parameters are re-estimated. The summation values are only added if each point cloud has at least one point and there are a total of at least 3 points, in both the sets, together since this is the minimum number of points required to estimate the line parameters.

Now, each of the points in the left, and right inlier point clouds has its squared distance computed, from the respective line. The squared distances are summed up and divided by the number of points to achieve the mean squared distance. The minimum mean squared distance is tracked over all the iterations of the algorithm, and the line parameters corresponding to

the minimum mean squared distance is chosen as the final estimates. The line fitting method discussed above is implemented in a RANSAC algorithm, as shown in Algorithm 1.

Data: Q_k
Result: L_l and L_r
for *a certain number of iterations do* **do**
  From Q_k randomly select a set of points on the left side, C_l, and a set of points on the right side, C_r, where, $|C_l|, |C_r| \geq 1$ and $|C_l| + |C_r| \geq 3$, compute L_l and L_r based on Eq. (3.6), (3.8) and (3.9);
  Compute the distance from each point in Q_k to L_l and L_r, select a set of inliers from Q_k on the left and right sides, respectively, replace C_l and C_r with the selected inliers;
  if $|C_l|$ *and* $|C_r|$ *are both larger than a threshold* **then**
    recompute line parameters using the updated point cloud, then compute the squared distance(SD) to each point in C_l from L_l, and, from each point in C_r from L_r;
    current section becomes this one;
    if *the mean SD smaller than a threshold* **then**
      Break;
    end
  end
  Return L_l and L_r with the minimum mean SD found;
end

Algorithm 1: Line Fitting

3.3.2 Line Filtering

The fitted lines from the LiDAR point cloud contain a considerable amount of noise. To deal with the noise, an EKF is used. The EKF has two parts − a prediction step and an update step. The prediction step uses the vehicle's odometry information and guides how the line properties should change. The update step takes in a new data point, in this case, a pair of parallel lines generated by RANSAC, which is then compared to the predicted lines to obtain the filtered lines. The notation for this section is as follows:

The odometry provides the robot's position in the world reference frame. However, it is more relevant to use the robot's position with respect to the rows for the motion model of

EKF. The reasoning behind this is that the line properties(a, b_l, b_r) are computed relative to the robot's position.

$$\begin{aligned} {}^{\text{world}}\Delta x &= {}^{\text{world}}\, x_{k|k} - {}^{\text{world}}\, x_{k-1|k-1} \\ {}^{\text{world}}\Delta y &= {}^{\text{world}}\, y_{k|k} - {}^{world}\, y_{k-1|k-1} \\ {}^{\text{world}}\Delta \psi &= {}^{\text{world}}\, \psi_{k|k} - {}^{\text{world}}\, \psi_{k-1|k-1} \end{aligned} \tag{3.11}$$

The equations 3.11 represent the change in the position and attitude of the robot in the world reference frame. ${}^{world}\Delta x$ and ${}^{world}\Delta y$ can be used to calculate the distance traveled by the robot regardless of the frame of reference as shown in the Eq. 3.12.

$$\Delta dist = \sqrt{{}^{\text{world}}\Delta x^2 + {}^{\text{world}}\, \Delta y^2} \tag{3.12}$$

The robot attitude is localized with respect to the parallel rows' orientation, $\Delta dist$ needs to be transformed to that frame to be able to compute this change in heading of the robot.

$$\begin{aligned} {}^{\text{lidar}}\Delta dist_x &= \Delta dist \cdot \cos {}^{\text{world}}\, \psi \\ {}^{\text{lidar}}\Delta dist_y &= \Delta dist \cdot \sin {}^{\text{world}}\, \psi \end{aligned} \tag{3.13}$$

$$\begin{aligned} a_{k-1|k-1} &= \frac{{}^{\text{lidar}}\Delta dist_y}{{}^{\text{lidar}}\Delta dist_x} \\ \alpha &= \tan^{-1}(a_{k-1|k-1}) \end{aligned} \tag{3.14}$$

Equation (3.14) represents the angle between the computed rows and the x-axis of the robot, also referred to as the heading of the robot. It will be used to calculate the displacement of the robot perpendicular to the rows, for predicting the row properties a, b_l and, b_r in the prediction step of the filter. The transformation from the LiDAR reference frame to the row parallel coordinate system relies on this α. $\parallel$ represents the coordinate system with x-axis parallel to the tree rows and y-axis perpendicular to this. The equations that describe this transformation is shown in Eq. (3.15).

$$\begin{aligned} {}^{\parallel}\Delta dist_x &= {}^{\text{lidar}}\, \Delta dist_x \cos\alpha + {}^{\text{lidar}}\, \Delta dist_y \sin\alpha \\ {}^{\parallel}\Delta dist_y &= -{}^{\text{lidar}}\Delta dist_x \sin\alpha + {}^{\text{lidar}}\, \Delta dist_y \cos\alpha \end{aligned} \tag{3.15}$$

3.3.3 Prediction

$^{\parallel}\Delta dist_y$ is used to predict the line parameters at time step k. The prediction equations are summarized in Eq. (3.16).

$$\begin{aligned} a_{k|k-1} &= \tan(\alpha - {}^{world}\psi) \\ b_{l,k|k-1} &= b_{l,k-1|k-1} + {}^{lidar}\Delta dist_x \cdot \sin(\tan^{-1} a_{k-1|k-1}) \\ &\quad - {}^{lidar}\Delta dist_y \cdot \cos(\tan^{-1} a_{k-1|k-1}) \\ b_{r,k|k-1} &= b_{r,k-1|k-1} - {}^{lidar}\Delta dist_x \cdot \sin(\tan^{-1} a_{k-1|k-1}) \\ &\quad + {}^{lidar}\Delta dist_y \cdot \cos(\tan^{-1} a_{k-1|k-1}) \end{aligned} \tag{3.16}$$

The prediction part of the filter also updated the prediction covariance associated with the states of the filter. This is predicted using the covariance of process noise matrix, previous state's covariance matrix, and the state transition matrix. A summary of EKF filter equations is presented in Chapter 2. The state transition matrix is derived via the Jacobian of the predicted line properties in Eq. (3.16). The Jacobian thus derived is summarized in Eq. (3.17) and Eq. (3.18).

$$F_{l,k|k-1} = \begin{bmatrix} \frac{1}{\cos^2(\tan^{-1}(a_{k-1|k-1})) - {}^{world}\Delta\psi} \cdot \frac{1}{a_{k-1|k-1}^2+1} & 0 \\ \frac{{}^{lidar}\Delta dist_x - {}^{lidar}\Delta dist_y}{(a_{k-1|k-1}^2+1)^{\frac{3}{2}}} & 1 \end{bmatrix} \tag{3.17}$$

$$F_{r,k|k-1} = \begin{bmatrix} \frac{1}{\cos^2(\tan^{-1}(a_{k-1|k-1})) - {}^{world}\Delta\psi} \cdot \frac{1}{a_{k-1|k-1}^2+1} & 0 \\ \frac{-{}^{lidar}\Delta dist_x + {}^{lidar}\Delta dist_y}{(a_{k-1|k-1}^2+1)^{\frac{3}{2}}} & 1 \end{bmatrix} \tag{3.18}$$

The process noise was defined to be constant and experimentally determined, as shown in Eq. (3.19).

$$Q_{k-1} = \begin{bmatrix} 0.002 & 0 \\ 0 & 0.002 \end{bmatrix} \tag{3.19}$$

The covariance prediction equations are shown in Eq. (3.20). $P_{l,k|k-1}$ and $P_{r,k|k-1}$ are initialized as identity matrices at the beginning of the filter and are subsequently updated using these equations.

$$\begin{aligned} P_{l,k|k-1} &= F_{l,k|k-1} P_{l,k-1|k-1} F_{l,k|k-1}^T + Q_{k-1} \\ P_{r,k|k-1} &= F_{r,k|k-1} P_{r,k-1|k-1} F_{r,k|k-1}^T + Q_{k-1} \end{aligned} \tag{3.20}$$

3.3.4 Update

The next step of the EKF is − Update. Here, the innovation/residual covariance matrix is updated using the observation matrix H, the measurement noise matrix, R, and the covariance matrix, P. These equations are summarized in Eq. 3.21.

$$\begin{aligned} S_{l,k} &= H_{l,k} P_{l,k|k-1} F_{l,k|k-1}^T + R_k \\ S_{r,k} &= H_{r,k} P_{r,k|k-1} F_{r,k|k-1}^T + R_k \end{aligned} \tag{3.21}$$

After computing the Jacobian, the $H_{l,k}$ and $H_{r,k}$ are found to be $2x2$ identity matrices. The measurement noise matrix, R was computed using the specifications found for Hokuyo UST-10LX LiDAR. The specifications state an accuracy of $0.02\ m$ and the covariance matrix is therefor the square of this value,

$$R = \begin{bmatrix} 0.0004 & 0 \\ 0 & 0.0004 \end{bmatrix}$$

Finally, the Kalman gain is computed using thus derived innovation matrix, together with observation and prediction covariance matrices. It is calculated using the following equations:

$$\begin{aligned} K_{l,k} &= P_{l,k|k-1} H_{l,k}^T S_{l,k}^{-1} \\ K_{r,k} &= P_{r,k|k-1} H_{r,k}^T S_{r,k}^{-1} \end{aligned} \tag{3.22}$$

The line predictions are updated using the Eq. (3.23), where a, b_l, and b_r represent the line parameters computed via RANSAC. Finally, the line prediction covariance is updated, as shown in Eq. (3.24).

$$\begin{aligned} \begin{pmatrix} a_{k|k} \\ b_{l,k|k} \end{pmatrix} &= \begin{pmatrix} a_{k|k-1} \\ b_{l,k|k-1} \end{pmatrix} + K_{l,k} \begin{pmatrix} a - a_{k|k-1} \\ b_{l,k|k-1} \end{pmatrix} \\ \begin{pmatrix} a_{k|k} \\ b_{r,k|k} \end{pmatrix} &= \begin{pmatrix} a_{k|k-1} \\ b_{r,k|k-1} \end{pmatrix} + K_{r,k} \begin{pmatrix} a - a_{k|k-1} \\ b_{r,k|k-1} \end{pmatrix} \end{aligned} \tag{3.23}$$

$$\begin{aligned} P_{l,k|k} &= (I - K_{l,k} H_{l,k}) P_{l,k|k-1} \\ P_{r,k|k} &= (I - K_{r,k} H_{r,k}) P_{r,k|k-1} \end{aligned} \tag{3.24}$$

The Mahalanobis distance is a measure of how many standard deviations away, a point p is from the distribution's mean μ, and it is commonly used to detect outliers in linear regression models. Here, we use Mahalanobis distance to determine whether the lines should be predicted and updated or just predicted. This was useful in filtering out RANSAC lines which were very different from the previous estimate. If either the computed distances in Eq. (3.25) and (3.26) are above a certain threshold, the filter is not updated, and the new line properties are only predicted [28].

$$d_l = \begin{bmatrix} a - a_{k|k-1} & b - b_{l,k|k-1} \end{bmatrix} P_{l,k|k-1}^{-1} \begin{bmatrix} a - a_{k|k-1} \\ b - b_{l,k|k-1} \end{bmatrix} \tag{3.25}$$

$$d_r = \begin{bmatrix} a - a_{k|k-1} & b - b_{r,k|k-1} \end{bmatrix} P_{r,k|k-1}^{-1} \begin{bmatrix} a - a_{k|k-1} \\ b - b_{r,k|k-1} \end{bmatrix} \tag{3.26}$$

3.4 Control

3.4.1 In-Row Control

The perception system developed in the previous section outputs the filtered parameters for the row lines along with the estimated distances to the left and right sides of the robot. These parameters are then used to maintain the heading of the robot for autonomous navigation in Row−Following state. The cruise speed for row following is fixed and is set to an average value of 0.25 m/s. For maintaining the heading of the robot, a simple PID controller is implemented. The distances to the left and the right rows are used to compute the desired heading of the system, as shown in Eq. (3.27). In this equation, b_l and b_r are the line parameters and $\dot{\psi}$ is the current yaw rate estimate provided by state estimation described in Chapter 2.

$$\dot{\psi}_{n+1} = K_P \cdot (b_l - b_r) + K_D \cdot \dot{\psi}_n \tag{3.27}$$

To tune the PID controller, the robot's response to different K_P and K_D values was analyzed experimentally. The PID parameters can be tuned by system's response to a step function. Table 3.1 gives us some insight on how the system response changes to a step input based on different types of control inputs. From this table, it is easy to see that although we desire to minimize all of our observable parameters, it is not possible to do so. We often optimize one parameter at the expense of another. Table 3.2 shows the values that were

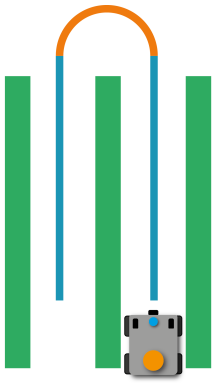


Figure 3.6: Illustration showing the Out-Row turning maneuver by Terrasentia where blue is In-Row state and orange is Out-Row state

obtained for the PID controller after tuning the system empirically.

Control Response	Rise Time	Settling Time	Overshoot	Steady State Error
K_P	Decreases	No effect	Increases	Decreases
K_D	Decreases	Increases	Increases	Eliminates
K_I	No effect	Decreases	Decreases	No effect

Table 3.1: Control Response Effects

K_P	8
K_D	0.5
K_I	0

Table 3.2: Tuned PID parameters for In-Row state

3.4.2 End-Row Turning Control

As was mentioned at the beginning of this chapter, as the robot traverses through the row, it keeps track of the average row width and an average number of points in the filtered laser scan. It uses these two parameters to generate semi-circular path at the end of the row, assuming that the rows have approximately the same width in each breeding plot. Fig. 3.6 illustrates the path generation and following during the end-row turning maneuver. A simple PD controller is used to control the turn yaw rate as shown in Eq. (3.27). The values for K_P and K_D were empirically determined to be 2 and 0.1 respectively.

3.5 Perception Subsystem Configuration

This section presents the operation of the Perception subsystem as it was deployed during the testing of the formulation developed in the previous section. It is assumed that the robot starts at the center of a CSS breeding plot row. Terrasentia robot is deployed with a LiDAR Hokuyo UST-10LX placed in the center of the front part of the robot, with the 2D cloud plane parallel to the ground. The LiDAR cloud data is filtered before it is fed to the perception subsystem with a bounding box filter to limit the readings within a rectangle of width $0.8m$ and a length of $0.35m$. Desired distance from the center of the robot is kept at zero, i.e., the robot follows the center of the row. However, the algorithm can be configured to maintain a certain distance from the center of the row. Figure 3.7a and 3.7b show the raw and filtered LiDAR point cloud of a single scan in the cornfield.

(a) (b)

Figure 3.7: Laser scan cloud pre and post-filtering (a) Raw laser scan (b) Filtered laser scan showing the ROI

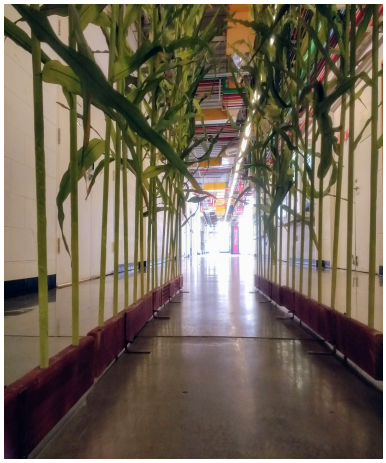
3.6 Results

Using the RANSAC and EKF formulation developed in the previous section, the robot was tested in the test environment shown in Fig. 3.8 using artificial corn. Artificial cornrows were used to test and debug the perception and autonomy algorithms as the planting season for CSS crops had not yet begun in Illinois. All the data presented here was collected online while the whole perception and navigation stack was functional. This section describes these tests.

Fig. 3.9 and 3.10 presents the accuracy of the row width using RANSAC and EKF respectively. It can be seen that the errors significantly reduce after line filtering. Most

errors are within 2 cm range around the true value. Fig. 3.11 shows the average computation time required by the perception stack on an Intel NUC with an i7 processor. On an average, it takes a little under 0.4 seconds for the complete perception stack to estimate and filter the row parameters.

Fig. 3.12 depicts the commanded yaw for row following and end-row turning. For testing the state machine architecture, a two-state, state machine was devised wherein the robot takes a turn once it has detected the end of row. For this test, the commanded velocity was set to be $0.3m/s$.



(a)

(b)

Figure 3.8: Testing environment: (a) View from the front camera (b) Testing environment setup for perception and autonomy

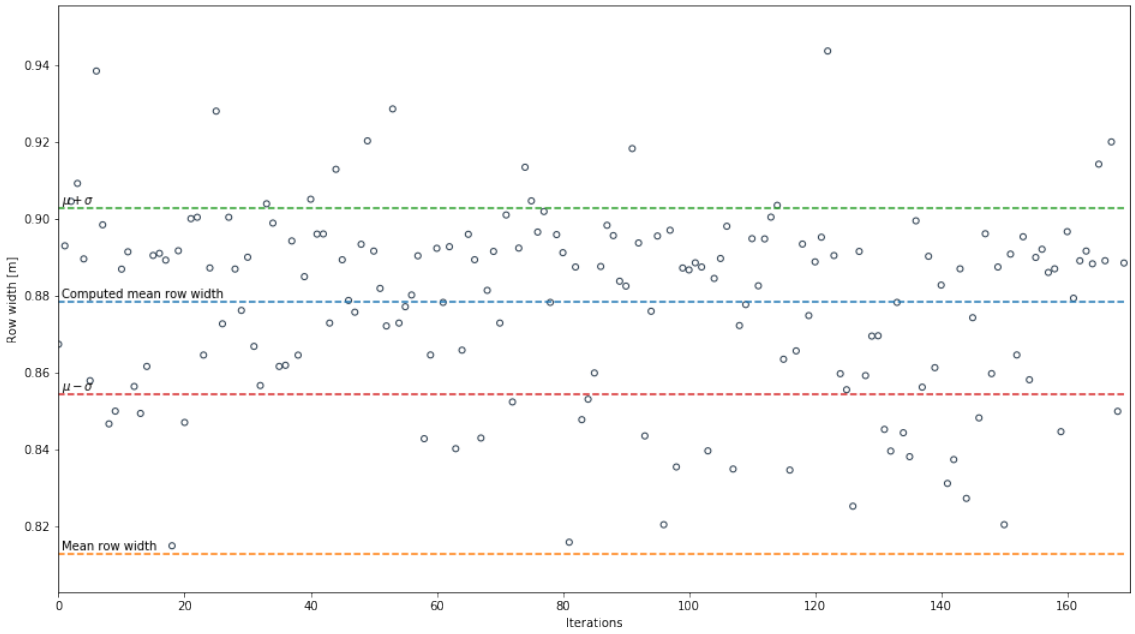


Figure 3.9: Estimated row widths with RANSAC

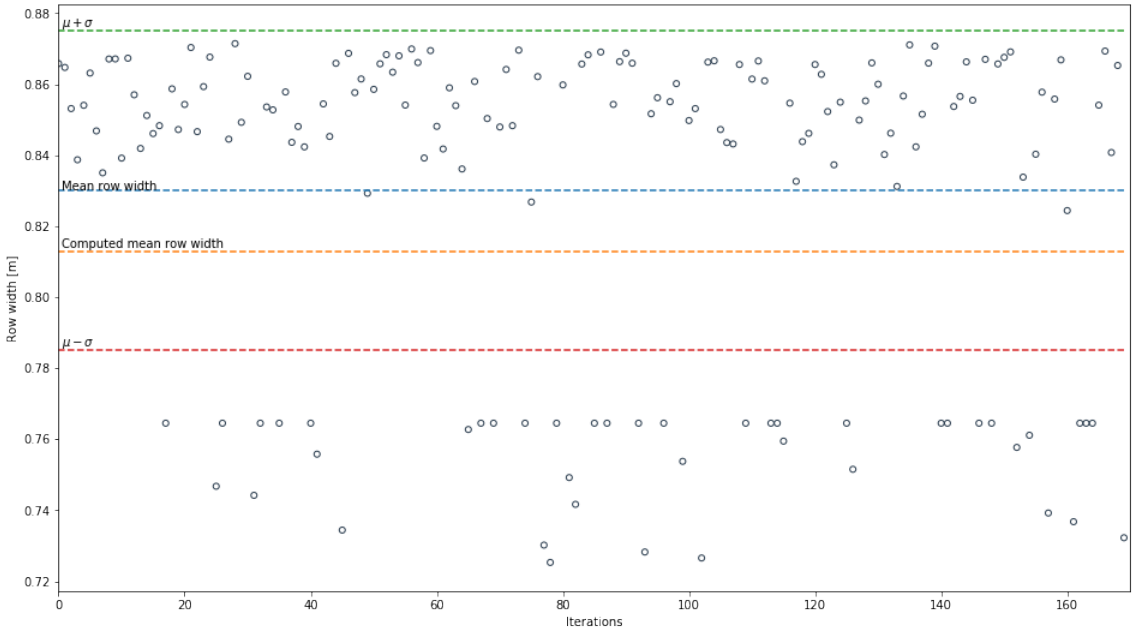


Figure 3.10: Estimated row widths after filtering with EKF

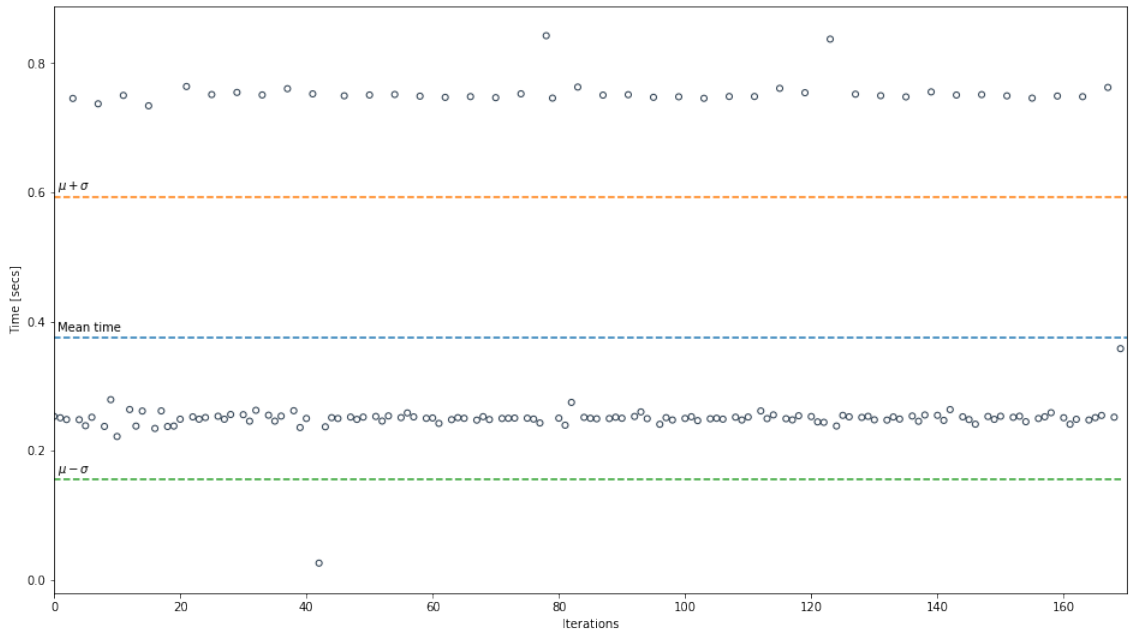


Figure 3.11: Computation time for row fitting algorithm(RANSAC & EKF)

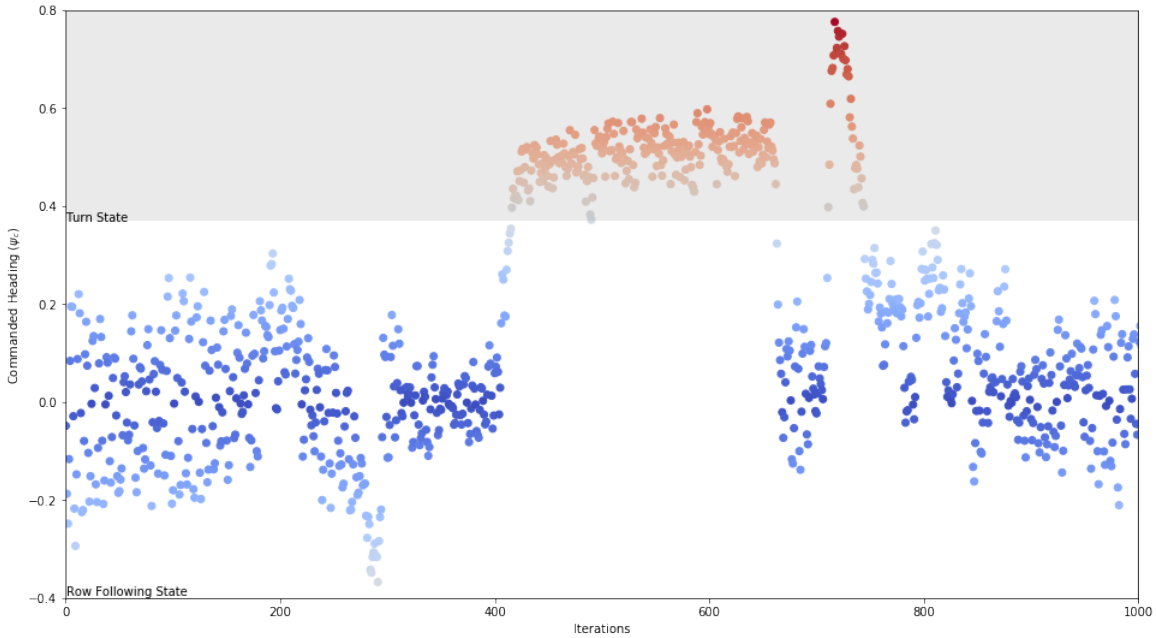


Figure 3.12: Commanded yaw rate as the robot autonomously traverses a corn row(Row Following State) and takes a turn(Turn State)

CHAPTER 4

CONCLUSION AND FUTURE WORK

The word "TerraSentia" comes from *terra* which means Earth and *sentia* which means intelligent. One can conceive a swarm of these compact, agile and autonomous robots working day and night endlessly in the agricultural fields and managing the crops. It's possible to develop adaptive algorithms that will enable such robots to make highly intelligent decisions based on the sensor data and machine learning such as informing the scientist which regions require detailed inspection, changing the swarm behavior based on the data collection over the past month, semantic mapping of the agricultural fields so that the researcher can use voice control to send commands to these robots, etc. Terrasentia does not exhibit these capabilities, but the current system architecture can accommodate all of this additional functionality.

Future work includes completely vision-based localization in the fields during the early season, and more rigorous testing and validation of the perception and navigation stack in real, cluttered CSS breeding fields. The former is essential for removing dependence from high-cost RTK GNSS during the early season. An attempt for finding row lines using a deep learning architecture − FCN was made. However, the localization errors regularly exceeded the safety limits which would result in the robot striking the crop rows. Figure 4.1 shows the output of this network during the early season. Further work towards semantic segmentation and regression needs to be done, to make this robust.

Online trajectory generation and path optimization for obstacle avoidance need to be implemented so that the robot can handle adverse situations better and get around obstacles in the field. Although some work was done to use heuristics and computer vision to differentiate between in-row and out-of-row states, these features need to be improved and unsupervised learning needs to be incorporated so that the model does not need to be retrained for every new field the robot encounters.

In this thesis, we present the work for designing an autonomous architecture for a small, compact robot − Terrasentia along with developing sensor fusion, perception, and control algorithms for autonomous navigation through the field. We outline the design process, describe state estimation, and navigation algorithms we implemented that have been tested

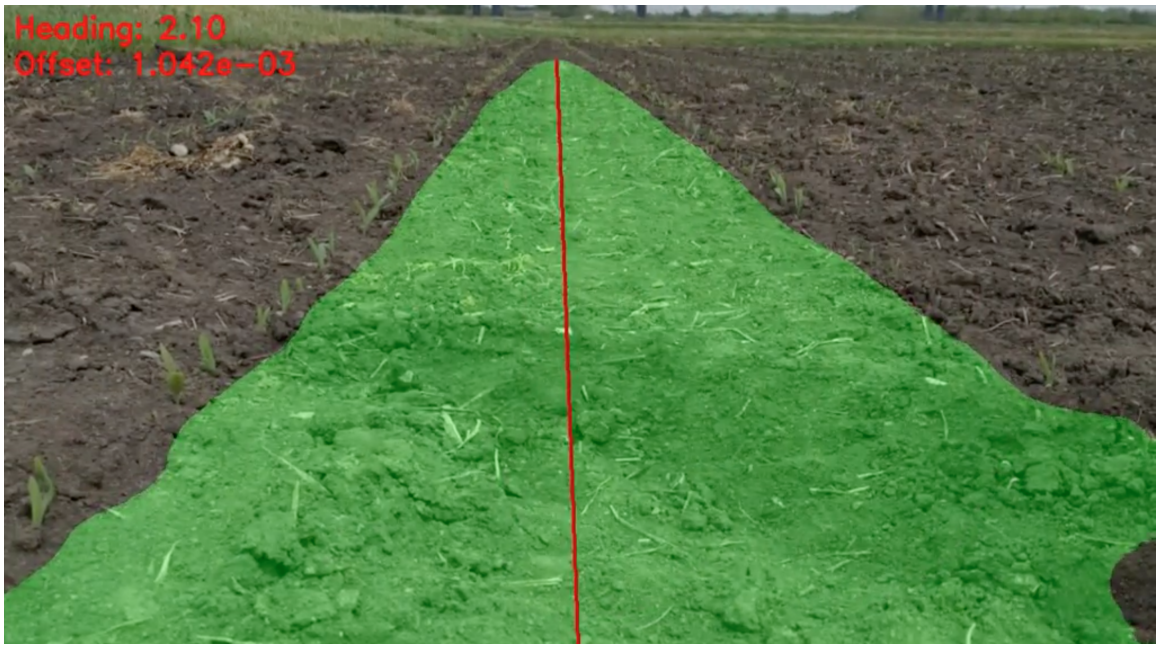


Figure 4.1: FCN detecting the row center and heading for vision based early-season navigation

and proven in both simulation and real-world testing environments.

CHAPTER 5

REFERENCES

[1] "Farms and farmland - numbers, acreage, ownership, and use," 2014. [Online]. Available: https://goo.gl/vhJv8h

[2] T. Mueller-Sim, M. Jenkins, J. Abel, and G. Kantor, "The robotanist: a ground-based agricultural robot for high-throughput crop phenotyping," M.S. thesis, 2017.

[3] "Transportation energy resources from renewable agriculture (terra)." 2015. [Online]. Available: https: //arpa-e.energy.gov/?q=arpa-e-programs/terra,

[4] R. T. Furbank and M. Tester, "Phenomics–technologies to relieve the phenotyping bottleneck," *Trends in plant science*, vol. 16, no. 12, pp. 635–644, 2011.

[5] J. K. Conner, R. Franks, and C. Stewart, "Expression of additive genetic variances and covariances for wild radish floral traits: comparison between field and greenhouse environments," *Evolution*, vol. 57, no. 3, pp. 487–495, 2003.

[6] F. Liebisch, N. Kirchgessner, D. Schneider, A. Walter, and A. Hund, "Remote, aerial phenotyping of maize traits with a mobile multi-sensor approach," *Plant methods*, vol. 11, no. 1, p. 9, 2015.

[7] C. Z. Espinoza, S. Sankaran, M. O. Pumphrey, P. N. Miklas, A. H. Carter, G. J. Vandemark, N. R. Knowles, L. R. Khot, S. Jarolmasjed, V. R. Sathuvalli et al., "Low-altitude, high-resolution aerial imaging systems for row and field crop phenotyping: A review," 2015.

[8] P. Andrade-Sanchez, M. A. Gore, J. T. Heun, K. R. Thorp, A. E. Carmo-Silva, A. N. French, M. E. Salvucci, and J. W. White, "Development and evaluation of a field-based high-throughput phenotyping platform," *Functional Plant Biology*, vol. 41, no. 1, pp. 68–79, 2014.

[9] S. C. Murray, L. Knox, B. Hartley, M. A. Méndez-Dorado, G. Richardson, J. A. Thomasson, Y. Shi, N. Rajan, H. Neely, M. Bagavathiannan et al., "High clearance phenotyping systems for season-long measurement of corn, sorghum and other row crops to complement unmanned aerial vehicle systems," in *Autonomous Air and Ground Sensing Systems for Agricultural Optimization and Phenotyping*, vol. 9866. International Society for Optics and Photonics, 2016, p. 986607.

[10] N. Virlet, K. Sabermanesh, P. Sadeghi-Tehran, and M. J. Hawkesford, "Field scanalyzer: an automated robotic field phenotyping platform for detailed crop monitoring," *Functional plant biology*, vol. 44, no. 1, pp. 143–153, 2017.

[11] D. Hall, F. Dayoub, J. Kulk, and C. McCool, "Towards unsupervised weed scouting for agricultural robotics," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5223–5230.

[12] L. Grimstad, C. D. Pham, H. T. Phan, and P. J. From, "On the design of a low-cost, light-weight, and highly versatile agricultural robot," in *Advanced Robotics and its Social Impacts (ARSO), 2015 IEEE International Workshop on*. IEEE, 2015, pp. 1–6.

[13] L. Haibo, D. Shuliang, L. Zunmin, and Y. Chuijie, "Study and experiment on a wheat precision seeding robot," *Journal of Robotics*, vol. 2015, p. 12, 2015.

[14] J. Torgersen, "Mobile agricultural robot: Independent four wheel ackerman steering," M.S. thesis, Norwegian University of Life Sciences, Ås, 2014.

[15] "Clearpath robotics, husky technical specifications." [Online]. Available: https://www.clearpathrobotics. com/husky-unmanned-ground-vehicle-robot/

[16] "Robotnik automation, sll. available mobile robots." [Online]. Available: http://www.robotnik.eu/ mobile-robots/

[17] K. Cavender-Bares and C. C. Bares, "Robotic platform and method for performing multiple functions in agricultural systems," Mar. 22 2016, uS Patent 9,288,938.

[18] "Naio technologies." [Online]. Available: https://www.naio-technologies.com/en/agricultural-equipment/vineyard-weeding-robot/

[19] "Qinetiq north america. talon." [Online]. Available: https://www.qinetiq-na.com/products/ unmanned-systems/talon/

[20] "Omron adept technologies, inc. omron adept research robots technical speci-cations." [Online]. Available: http://www.mobilerobots.com/ResearchRobots/Specifications.aspx

[21] T. Bell, "Automatic tractor guidance using carrier-phase differential gps," *Computers and electronics in agriculture*, vol. 25, no. 1-2, pp. 53–66, 2000.

[22] M. Nørremark, H. W. Griepentrog, J. Nielsen, and H. T. Søgaard, "The development and assessment of the accuracy of an autonomous gps-based system for intra-row mechanical weed control in row crops," *Biosystems Engineering*, vol. 101, no. 4, pp. 396–410, 2008.

[23] C. Zhang, N. Noguchi, and L. Yang, "Leader–follower system using two robot tractors to improve work efficiency," *Computers and Electronics in Agriculture*, vol. 121, pp. 269–281, 2016.

[24] H. Wang and N. Noguchi, "Autonomous maneuvers of a robotic tractor for farming," in *System Integration (SII), 2016 IEEE/SICE International Symposium on*. IEEE, 2016, pp. 592–597.

[25] T. Bak and H. Jakobsen, "Agricultural robotic platform with four wheel steering for weed detection," *Biosystems Engineering*, vol. 87, no. 2, pp. 125–136, 2004.

[26] T. Bakker, K. van Asselt, J. Bontsema, J. Müller, and G. van Straten, "Autonomous navigation using a robot platform in a sugar beet field," *Biosystems Engineering*, vol. 109, no. 4, pp. 357–368, 2011.

[27] F. Rovira-Más, I. Chatterjee, and V. Sáiz-Rubio, "The role of gnss in the navigation strategies of cost-effective agricultural robots," *Computers and electronics in Agriculture*, vol. 112, pp. 172–183, 2015.

[28] J. Zhang, A. Chambers, S. Maeta, M. Bergerman, and S. Singh, "3d perception for accurate row following: Methodology and results," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 5306–5313.

[29] J. Bell, B. A. MacDonald, and H. S. Ahn, "Row following in pergola structured orchards," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 640–645.

[30] S. A. Hiremath, G. W. Van Der Heijden, F. K. Van Evert, A. Stein, and C. J. Ter Braak, "Laser range finder model for autonomous navigation of a robot in a maize field using a particle filter," *Computers and Electronics in Agriculture*, vol. 100, pp. 41–50, 2014.

[31] T. A. Troyer, S. Pitla, and E. Nutter, "Inter-row robot navigation using 1d ranging sensors," *IFAC-PapersOnLine*, vol. 49, no. 16, pp. 463–468, 2016.

[32] A. Velasquez, V. Higuti, H. Guerrero, and M. Becker, "Helvis-a small-scale agricultural mobile robot prototype for precision agriculture," in *Proceedings of the 13th International Conference on Precision Agriculture (ICPA2016), St. Louis, MO, USA*, vol. 31, 2016, pp. 1–18.

[33] T. B. Y. H. Kayacan, E. and G. Chowdhary, "High precision of an ultra-compact 3d printed field robot in the presence of slip." *In International Conference on Robotics and Automation, Brisbane, Australia. IEEE. Submitted.*, 2018.

[34] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, Kobe, Japan, May 2009.

[35] S. Thrun and J. J. Leonard, "Simultaneous localization and mapping," pp. 871–889, 2008.

[36] A. Kelly, *Mobile Robotics: Mathematics, Models, and Methods*. New York, NY, USA: Cambridge University Press, 2013.

[37] G. Guennebaud, B. Jacob et al., "Eigen v3," http://eigen.tuxfamily.org, 2010.

[38] J. Siek, L.-Q. Lee, and A. Lumsdaine, "Boost random number library," http://www.boost.org/libs/graph/, June 2000.

APPENDIX A

ROS-GAZEBO SIMULATION

A ROS interfaced Gazebo simulation was created to test and prototype the sensor fusion and autonomy algorithms. Figure A.1 shows an overview of the Gazebo simulation environment. Each block in this diagram represents a set of nodes that publish necessary data over ROS topics, whereas the state machine is an Action/Client server which changes the state of the robot as it navigates in the simulation.

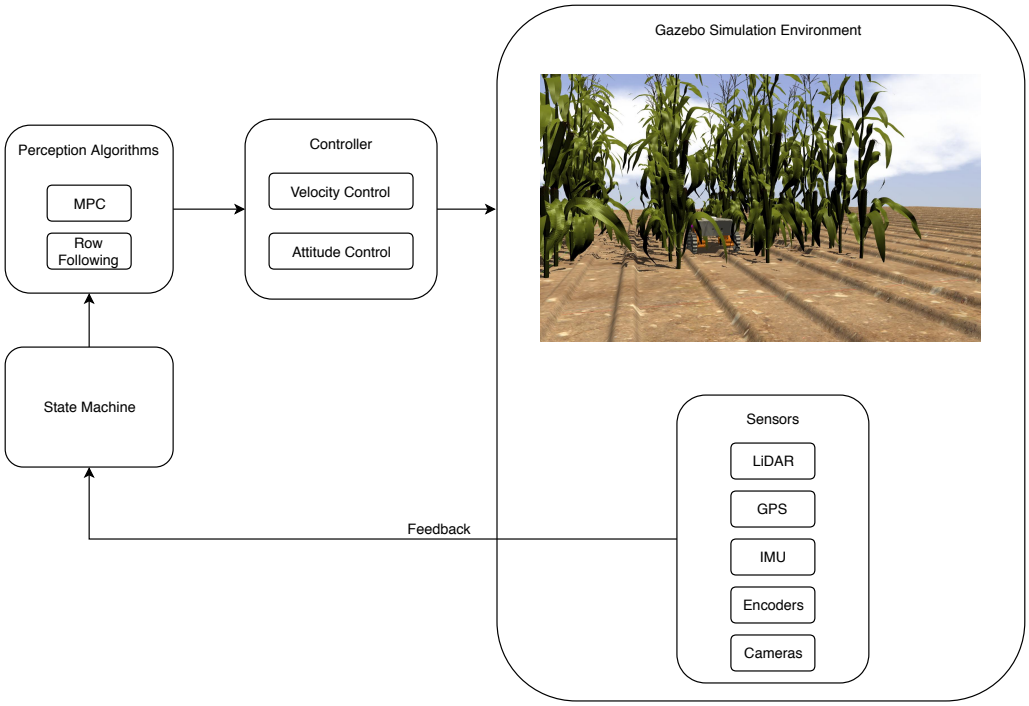


Figure A.1: An overview of ROS-Gazebo simulation environment

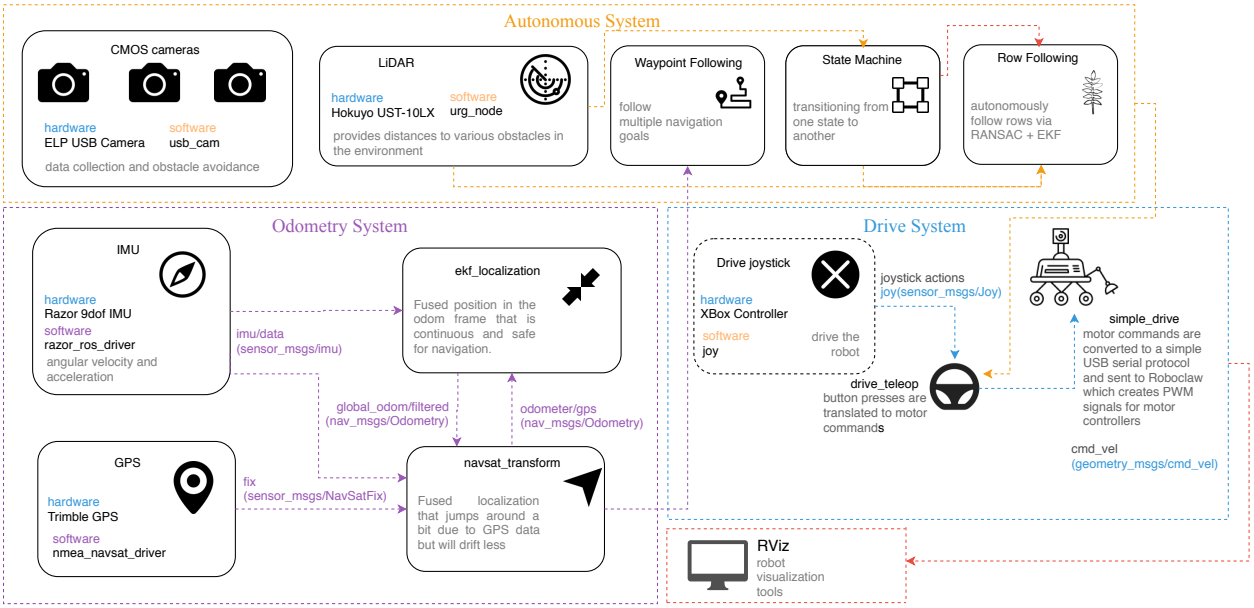


Figure A.2: ROS HIL architecture