SANTA CLARA UNIVERSITY

School of Engineering

Date: September 8th, 2024

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Ryan Shiau Lam

ENTITLED

Reactive In-Row LiDAR-Based Vineyard Navigation

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN
ROBOTICS AND AUTOMATION

Christopher Kitts — 9/10/2024

Thesis Advisor and Director of the Robotics and Automation Program
Dr. Christopher Kitts — Date

2024-09-09

Thesis Reader
Dr. Manoj Sharma — Date

Michael Neumann — 2024-09-10

Thesis Reader
Dr. Michael Neumann — Date

Reactive In-Row LiDAR-Based Vineyard Navigation

By

Ryan Shiau Lam

MASTER THESIS

Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Robotics and Automation
in the School of Engineering at
Santa Clara University
Spring 2024

Santa Clara, California

Abstract

Agricultural robotics plays a critical role in addressing contemporary societal challenges, including food security, sustainable agricultural practices, and labor shortages. Effective navigation is essential for agricultural robots to traverse intricate environments and execute tasks with precision. This thesis introduces two innovative methods for in-row vineyard navigation: one leveraging branch features and the other harnessing trunk features of the vine. Through a comprehensive analysis of the two methods of in-row vineyard navigation, the study demonstrates the superior performance of the trunk detection method over the branch detection algorithm, notably surpassing current research standards in terms of cross-track error performance. By emphasizing algorithmic innovation, this research contributes to advancing precision agriculture practices and addressing the evolving complexities of modern agriculture. The findings emphasize the importance of robust navigation systems in enhancing the sustainability and productivity of agricultural systems, thus highlighting the potential of agricultural robotics to drive positive societal change.

Acknowledgments

I would like to express my thanks and gratitude to Dr. Christopher Kitts and Dr. Manoj Sharma for their persistent and continued guidance throughout not only this project, but my during my academic career as well. They have both been significant components towards my development as an engineer and I am forever grateful.

I am extremely thankful for my family: Janie Shiau, Rudy Lam, and Nathan Lam for their unconditional love and support throughout the years. Their immense sacrifice has allowed me to blossom into the person I am today.

Finally, I'd like to thank everyone who has assisted and uplifted me through the completion of of this project, most notably Meilein Rose Fajardo and Matthew Kiyama.

Ryan Shiau Lam

Santa Clara University

Date: June 15, 2024

Contents

Abstract iii

Acknowledgments iv

Contents v

List of Figures vii

List of Tables viii

Nomenclature ix

1 Introduction 1
- 1.1 Background and Motivation . . . 1
  - 1.1.1 Labor Shortages . . . 2
  - 1.1.2 Food Security . . . 2
  - 1.1.3 Environmental Sustainability . . . 3
- 1.2 A Survey of Related Work . . . 3
  - 1.2.1 Vineyard Navigation . . . 4
  - 1.2.2 Related Work on Map Based Navigation . . . 4
  - 1.2.3 Related Work on Reactive Based Navigation . . . 6
- 1.3 Problem Statement . . . 8
- 1.4 Thesis Contributions . . . 9
- 1.5 Reader's Guide . . . 10

2 AgBot System Specifications 11
- 2.1 AgBot: Hardware Design . . . 12
  - 2.1.1 Drive Unit . . . 12
  - 2.1.2 Rover System . . . 13
- 2.2 Agbot: Software Motion Control and Design . . . 15

3 Branch Detection Based Steering System 16
- 3.1 Branch-Based Navigation Implementations . . . 17
  - 3.1.1 Pointcloud Acquisition . . . 17
  - 3.1.2 Branch Detection and Scene Comprehension . . . 19
  - 3.1.3 Rover Position and Orientation Derivation . . . 20

3.1.4 Error Computation 23
3.1.5 Rover Control 24
3.2 Performance Verification System 26
3.2.1 Experimental Process 27
3.3 Results 28
3.3.1 System Performance Analysis 29
3.4 Summary 31

4 Trunk Detection Base Navigation System **32**
4.1 Trunk-Based Navigation Implementation 33
4.1.1 Pointcloud Acquisition 33
4.1.2 Trunk Detection and Scene Comprehension 34
4.1.3 Rover Position and Orientation Derivation 35
4.1.4 Error Computation 37
4.1.5 Rover Control 38
4.2 Performance Verification System 41
4.2.1 Experimental Process 42
4.3 Results 43
4.3.1 System Performance 44
4.4 Summary 46

5 Comparison and Conclusion **47**
5.1 Comparative Analysis 47
5.2 Conclusion 48

List of Figures

1.1 Performance OF model D-42 against state-of-the-art. 5
1.2 Performance of Jackal's ability to navigate a simulated linear vineyard row. 7

2.1 AgBot: Deployed Rover in the Vineyard. 11
2.2 AgBot: Modular Connectivity Illustration. 12
2.3 AgBot: CAD Design with 2 Drive Units 13
2.4 AgBot: Robotic Operating System (ROS) System Sketch for Rover Motion. 14

3.1 Perception Based Vineyard Navigation System Overview 16
3.2 Branch Detection: Perception and Scene Comprehension Block. 18
3.3 Branch Detection: A Pointcloud of the Vineyard. 18
3.4 Branch Detection: Rover Diagram 20
3.5 Branch Detection: Process Flow Chart. 21
3.6 Branch Detection: Cross-track and Heading Error Computation Block . . . 23
3.7 Branch Detection: Control Example Diagram 25
3.8 Branch Detection: Performance Verification System 27
3.9 Branch Detection: System Perceived Performance 30
3.10 Branch Detection: Ground Truth Performance 30

4.1 Perception Based Vineyard Navigation System Overview 32
4.2 Trunk Detection: Perception and Scene Comprehension Block. 34
4.3 Trunk Detection: A Pointcloud of the Vineyard. 35
4.4 Trunk Detection: Rover Diagram 36
4.5 Trunk Detection: Cross-track and Heading Error Computation Block 37
4.6 Trunk Detection: Process Flow Chart 39
4.7 Trunk Detection: Control Example Diagram 40
4.8 Trunk Detection: Performance Verification System 42
4.9 Trunk Detection: System Performance 45
4.10 Trunk Detection: Ground Truth Performance 45

List of Tables

3.1 Branch Detection: Performance Metrics at Steady State 29

4.1 Trunk Nav: System Performance Values at Steady State 44

5.1 Comparative Analysis: Branch vs. Trunk 47
5.2 Comparative Analysis: Trunk compared to other algorithms. 48

Nomenclature

APOM Augmented Perception Obstacle Map

LiDAR Light Detection and Ranging Sensor *(also commonly known as 'lidar' or a 'laser range scanner')*

GPS Global Positioning System

GNSS Global Navigation Satellite System

ROS Robotic Operating System

SCU Santa Clara University

SLAM Simultaneous Localization and Mapping

INS Inertial Navigation System

IMU Inertial Measurement Unit

RMSE Root Mean Squared Error

TEB Timed Elastic Band

PCA Principal Component Analysis

DU Drive Unit

Chapter 1

Introduction

1.1 Background and Motivation

Agriculture indeed stands as the backbone of civilization, providing sustenance, economic stability, and cultural identity to societies throughout history. In the contemporary context, the integration of agricultural robots emerges as a transformative force, aimed at revolutionizing the way we cultivate the land and nourish our communities. The transition from classical farming tools to robots equipped with advanced sensors and intelligent software, signifies a necessary paradigm shift in farming practices by promising increased efficiency, sustainability, and resilience in the face of global challenges. Through the automation of tasks ranging from planting and monitoring crops to harvesting and data analysis, agricultural robotics addresses critical societal issues such as labor shortages, environmental degradation, and food insecurity. Furthermore, the growing field of robotics democratizes access to agricultural technology, empowering farmers of all scales to optimize production, conserve resources, and adapt to a changing climate. The evolution of agricultural robotics not only represents itself as a technological marvel, but also fundamentally reimagines societies relationship with the land and the systems that sustain us.

The progression of time and societal evolution has introduced the realm of agriculture to many intertwined complex challenges, which include labor shortages, food scarcity and environmental degradation. As the population grows and humans continue to shift towards accelerated urbanization, many traditional farming practices struggle to keep pace with

the increasing demands for food production. Additionally, factors such as climate change, soil depletion, and water scarcity have compounded these challenges, further exacerbating issues of sustainability and resilience in agricultural systems.

1.1.1 Labor Shortages

The shortage of labor is one of the primary issues currently plaguing the agricultural sector. This issue is especially prevalent in regions with aging populations and declining interests in agricultural work among younger generations. Since 1950, the average annual employment of hired farm workers has decreased a staggering 51% [1]. As rural communities continue to shrink and urban migration accelerates, the availability of manual labor for farming operations becomes increasingly scarce. Agricultural robots address this issue by automating repetitive and labor-intensive tasks such as planting, weeding, and harvesting, reducing the reliance on human labor and mitigating the impacts of labor shortages on farm productivity. By freeing up human labor for more skilled and strategic tasks, robots enable farmers to operate more efficiently and sustainably while also improving working conditions and safety on the farm.

1.1.2 Food Security

Another primary issue plaguing modern agriculture is food scarcity. With population projected to increasing to nearly 10 billion individuals by 2050, ensuring an adequate and nutritious food supply in such a short amount of time presents an extremely daunting challenge [2]. Traditional farming methods often struggle to meet the demands of a growing population while simultaneously contending with the impacts of climate change and resource constraints. However, agricultural robots offer a promising solution to this dilemma by enhancing productivity, optimizing resource usage, and minimizing crop losses due to pests, diseases, and adverse weather conditions. Through precision farming techniques facilitated by robots, farmers can achieve higher yields with minimal resource consumption, thereby increasing food production and improving food security on a global scale.

1.1.3 Environmental Sustainability

Environmental degradation is perhaps one of the most pressing challenges confronting modern agriculture, with conventional farming practices contributing to soil erosion, water pollution, deforestation, and greenhouse gas emissions [3]. Agricultural automation offer a pathway to mitigating these environmental impacts by promoting precision agriculture techniques that minimize the use of agrochemicals, reduce soil compaction, and optimize water and nutrient management. Through the deployment of sensors, drones, and autonomous machinery, farmers can monitor soil health, detect crop diseases, and implement targeted interventions that minimize environmental harm while maximizing agricultural efficiency. By fostering more sustainable and regenerative farming practices, agricultural robots play a vital role in preserving natural ecosystems, mitigating climate change, and safeguarding the long-term viability of agricultural practices for future generations.

1.2 A Survey of Related Work

From the early days of irrigation to the introduction of basic farming tools like the hoe and plow, each era saw advancements in agricultural capabilities. The introduction of modern day mechanized machinery such as the tractor and thrasher heralded a new era of innovation. Each agricultural revolution not only showcased human ingenuity and resilience, but were also driven by the common goal of improving crop yields and farming productivity to meet the increasing demands of society.

With challenges such as a population projected to climb to record numbers, and arising issues such as climate change and environmental degradation that'll further hinder our ability to effectively farm, it is evident that that society is due for another agricultural revolution. To resolve the issues in the coming generations, it is no longer reasonable to allocate more resources as a means to a solution. Instead, society must approach the issue in a mindful manner. Enabling methods of resource consumption minimization and product yield maximization, it is clear that collaborative robotics are the next step in agricultural progression.

This thesis aims to delve into a fundamental aspect of advanced automated farming

systems: navigation. Autonomous navigation plays a central role in enabling mobile units to navigate efficiently across diverse agricultural landscapes, overcoming challenges with intelligence and precision.

1.2.1 Vineyard Navigation

A subset of agricultural and orchard navigation is vineyard navigation. As a result of the very specific environmental conditions necessary to conduct viticulture, many vineyards struggle to obtain reliable Global Navigation Satellite System (GNSS) data due to occlusion from nearby woods or dense foliage.

Consequently, this limits methods of vineyard navigation to localized on board perception-based techniques. Fortunately, vineyard rows typically exhibit distinctive visual cues that can be leveraged to facilitate accurate in-row navigation. Perception-based navigation is often categorized in one of two approaches: map-based techniques and reactive methods; each offering its own set of advantages.

This section surveys related work that has been conducted in the domain of perception-based vineyard navigation.

1.2.2 Related Work on Map Based Navigation

Map-based navigation leverages the use of environmental maps, which are created through the process of mapping, utilizing sensors such as Light Detection and Ranging Sensor (LiDAR) and cameras. These maps provide the system to obtain prior knowledge of the environment; resulting in capabilities such as global planning, optimal pathfinding, and enhanced predictability. Consequently, depending on the tasks, map-based approaches are adept at navigating complex and relatively stable environments, utilizing techniques such as Simultaneous Localization and Mapping (SLAM)[4]. This reliability is especially valuable in sectors like agriculture, where consistent navigation in dynamic but largely static environments is crucial for tasks such as precision farming [5].

Abdelaziz et al.[6] addresses the issue of GNSS-denied environments, such as the vineyard used to test our algorithm, with a LiDAR and Visual-SLAM aided Inertial Naviga-

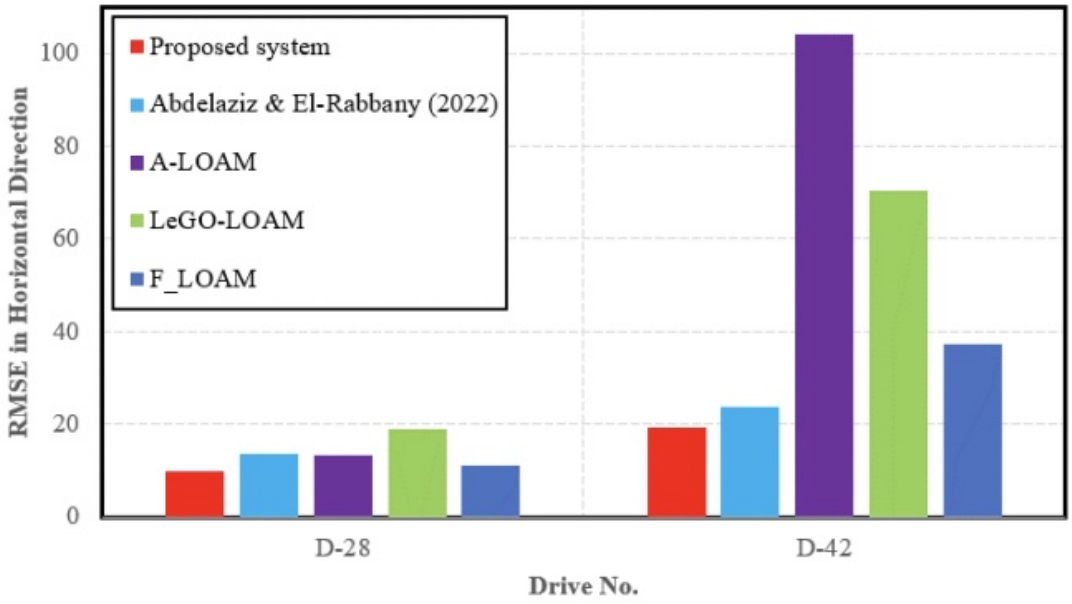

Figure 1.1 – Performance of Abdelaziz et al.[6] model[D-42] against state-of-the-art.

tion System (INS). Conducting sensor fusion, using an Inertial Measurement Unit (IMU) with the generated extended Kalman filter trajectories from the Visual SLAM and LiDAR SLAM, researchers were able to develop an INS. To test their model, Abdelaziz et al. employed their system on two datasets with a simulated GNSS outage. These datasets included a residential environment and a highway environment. Their highway experiment displayed that their sensor fusion system was able to significantly outperform state-of-the-art SLAM algorithms, yielding horizontal Root Mean Squared Error (RMSE) of 20.45 in position error. Although their model displayed an impressive 95% improvement on the state-of-the-art, the computed accuracy was only deployed in a simulated environment.

Gim et al.[7] proposed a method of autonomous navigation system using a 2.5D map generated with a point cloud. The research conducted describes a method of navigation with obstacle avoidance in a dynamic environment. In order to obtain a 2.5D map of the environment, the researchers obtained a combined a 2D grid map with 3D geometry from a LiDAR point cloud. The resulting map generation was comprised of the static points, where as dynamic points were detected by tracking the changes over time. Experimental analysis by the researchers demonstrated that this approach reduces computation costs, allowing for real-time obstacle avoidance and simultaneous SLAM processing with incoming 3D point

cloud data.

Li et al.[8] addressed the issue of autonomous orchard navigation using a 3D-point cloud map. Their approach involved filtering the point cloud map through pass-through filtering and Principal Component Analysis (PCA), followed by clustering and segmentation of tree row orientation to create a local map. Based on the map, researchers employed global path planning using the combined efforts of least squares and A*. Finally, the Timed Elastic Band (TEB) algorithm is used to dynamically compute and adjust the robots local orientation. Experimental analysis shows their system was able to obtain an average longitudinal deviation of 0.267m with an average heading angle deviation of 4.09° in a real orchard.

1.2.3 Related Work on Reactive Based Navigation

Reaction-based navigation is a method of autonomously guiding vehicles and robots strictly on reactive responses to environmental stimuli. Unlike other approaches that rely on a pre-mapped environment or a pre-planned Global Positioning System (GPS) coordinates, reactive systems are able to respond to obstacles or terrain variation in real-time. This approach is particularly beneficial in environments such as vineyards, where the complex terrain and obstacles may significantly impede on the performance of other methods of navigation.

Milburn et al.[9] exemplified the ability of reactive-navigation in a vineyard on a quadruped robot. The robot in this paper is equipped with an arm as well as vision sensors. Researchers were able to automate navigation throughout the vineyard and autonomous pruning through the implementation of a vision-based grapevine detection system utilizing a ZED stereo camera. The robot derived waypoints based on the detected grapevine clusters and was able to generate a trajectory. Although the performance of the robot navigation produced a mean positional error of $\sim 0.215m$, the trained model succumbed to difficulties when generalizing grapevines at different angles. Because the system depended on a computer vision model to accurately detect grape clusters, the system struggled to adapt to untrained features such as season variation and different stages of plant growth progression.

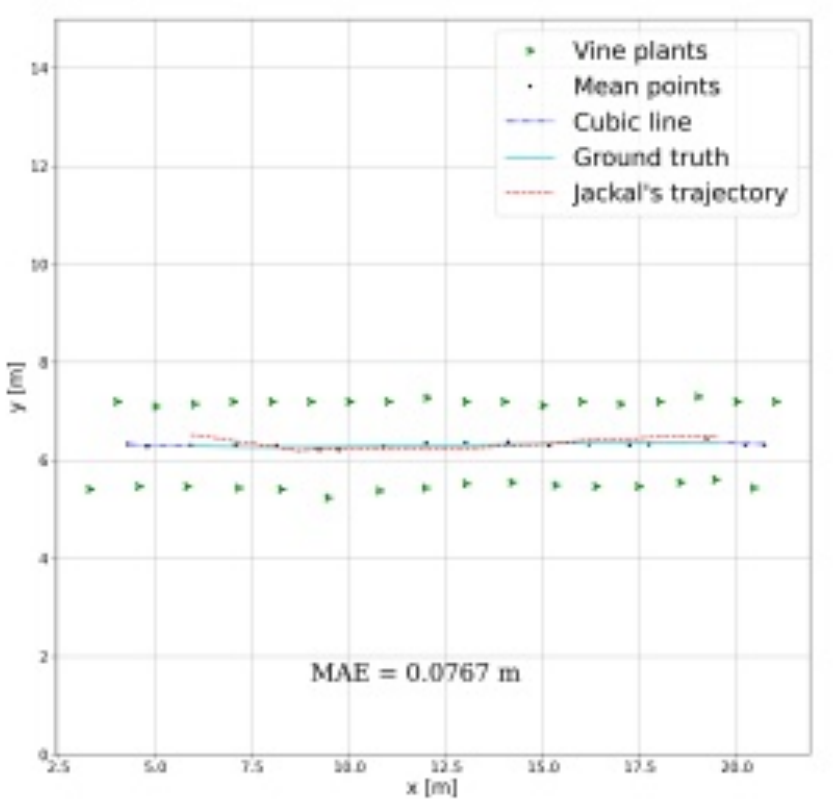

Figure 1.2 – Performance of Jackal's ability to navigate a simulated linear vineyard row [10].

Aghi et al.[10] integrates a novel and relatively affordable method of autonomous vineyard navigation, leveraging a custom trained segmentation network coupled with a low-range depth camera. Based on the extracted semantic information of the incoming scene, the system was able to extrapolate a trajectory by taking the center-point of segmented clusters to compute a resulting linear and angular velocity. Their system was able to achieve a remarkable mean absolute error of 0.0767 m in a simulated environment.

Aghi et al.[11] present in-row navigation based on two states. The first state utilizes a RGB-D camera to acquire a depth map, used to fit a rectangular area to the furthest pixel in the depth map, indicating the end of a row. If this process fails, the algorithm defaults to leveraging a neural network to re-correct the orientation of the rover if necessary. In their findings, the researchers achieved a CNN F1-Score of 0.94, however, no results of their navigation accuracy have been reported.

The in-row algorithm proposed in [12] uses a LiDAR to detect a conic region in front of the rover. Using this conic region, the robot is able to calculate the angular offset between

the cone center line and robot center line. To determine the lateral offset, the algorithm progressively increases the size of two rectangles until a select number of points in the conic edges fall within the bounds of the rectangles. This method achieved a mean center displacement error of 0.372m in a real environment.

In [13], a multi-sensor navigation system designed for in-row guidance is introduced. The proposed Augmented Perception Obstacle Map (APOM) allows the system to assess the data obtained from a RGB-D camera, LiDAR, and a number of ultrasonic sensors. Subsequently, the map is analyzed to find scenarios indicative of an in-row state, in which case, the robot is able to generate a navigation trajectory. Based on their experimentation, the average deviation from the centerline was measured to be 0.076m.

Mengoli et al.[14] propose a Hough Transform-based approach to conduct autonomous navigation within an orchard. The system works by transforming a 3D point cloud of the orchard to a 2D slice in order to extract distinct features of the orchard. A Hough Transformation is then applied to the 2D projection of the orchard to estimate the rows. Using the information, the robot was able to navigate in-row and perform row-change maneuvers, garnering an RMS error of 0.3429m and 0.5840m, respectively.

Saramento et al.[15] implements a novel navigation algorithm that employs trunk detection using a neural network and exploits the concept of a vanishing point to obtain a center-line trajectory. The two components of the proposed method are an estimation of the vanishing point and a positional averaged obtained by the two closest base trunks. Using a PID-controller, the robot followed the generated trajectory in a simulated straight vineyard row. For a straight-line configuration without disturbances, the researchers achieved an RMS error of 0.0117m.

1.3 Problem Statement

Viticulture robotics and automation represents the cutting edge of agricultural technology, reimagining the landscape of vineyard management. These state-of-the-art systems are engineered to automate tasks ranging from remedial pruning to automated vine disease detection. At the core of of vineyard automation lies the issue of achieving autonomous

navigation to allow mobile units to safely traverse vineyard rows.

This research presents two novel algorithms to combat GPS-degraded vineyards environments due to signal occlusion as a result of dense foliage. The objective of this research is to enable vineyard navigation at Kings Mountain Vineyard where the nearby woods and the vine growth causes GPS-signal degradation. Our proposed solution displays two methods of on-board perception-based in-row navigation of vineyards through the augmentation of one of our existing rovers by adding a LiDAR. The first method leverages the features of the vineyard branches to construct a boundary for in-row navigation. The second method proposes a season-invariant means of LiDAR based navigation that uses the trunks to construct a similar boundary. Both the branch-based detection and trunk-based detection methods were run through a gauntlet of experimentation to validate the performance of the navigation strategies, resulting in an RMS Error of 0.1877m and 0.0801m respectively.

1.4 Thesis Contributions

This thesis explores the integration of two novel LiDAR-based perception methods for row detection. These two methods are as follows:

- Development and performance analysis of branch-based in-row navigation, leveraging minimum distances to determine rover heading and cross-track error.
- Development and performance analysis of trunk-based season-invariant in-row navigation, leveraging clustering and line fitting to determine rover heading and cross-track error.

The resulting heading and cross-track error from the perception block are funneled into a PI-controller to enable in-row navigation based on the computed angular-velocity.

To the author's best knowledge, at the time of writing this thesis, the performance from the trunk-based detection method exceeds the current research standard in the scope of season invariant vineyard in-row navigation. Additionally, as this is an exploration on the two perception systems and detection algorithms, the functionality can be easily adapted to

any rover equipped with a 3D LiDAR, so long as it utilizes heading and cross-track error to determine its control of angular velocity. The following content also provides insight into the advantages and pitfalls of the two methods, discovered through rigorous real-world experimentation.

1.5 Reader's Guide

The following thesis is organized as follows. Chapter 2 provides a system overview on the AgBot, a modular rover used for agricultural applications at Santa Clara University (SCU). Chapter 3 proposes and analyzes a in-row navigation algorithm based on detecting the branches and the vines. Chapter 4 introduces and examines a season-invariant trunk detection algorithm used for in-row vineyard navigation. Chapter 5 conducts a comparison analysis on the two proposed methods, as well as, discusses possible paths for future work.

Since chapters 3 and 4 each delve into distinct techniques for in-row vineyard navigation, they are designed to stand alone. As a result, some content may be duplicated between them.

Chapter 2

AgBot System Specifications

The progression of technological advancements in robotics and its adjacent research fields has contributed to the modern automated agricultural revolution. One of these advancements is improvements in both the hardware and software of deployed agricultural mobile robots. These robots are often equipped with advanced sensors, such as LiDAR, GPS and cameras to permit enhanced perception of the environment [16]. In line with the trend, researchers at SCU have designed a modular rover for agricultural applications, named the AgBot. The rover, as seen in Figure 2.1, serves as a test bed for research and development aimed towards agricultural applications. The following sections describe the hardware and software architecture that compose of the rover.[1]

Figure 2.1 – AgBot: Deployed Rover in the Vineyard.

[1] With their approval, this chapter has been heavily adapted from the paper "Modular and Reconfigurable Multiple Drive-Unit Based Rover - Design and Control" by Dr. Sharma and Dr. Kitts from SCU, discussing the system architecture and design of the agriculture rovers used in this thesis [17].

2.1 AgBot: Hardware Design

This section discusses the hardware implementation of the modular rover used for agricultural applications at SCU Robotic Systems Laboratory.

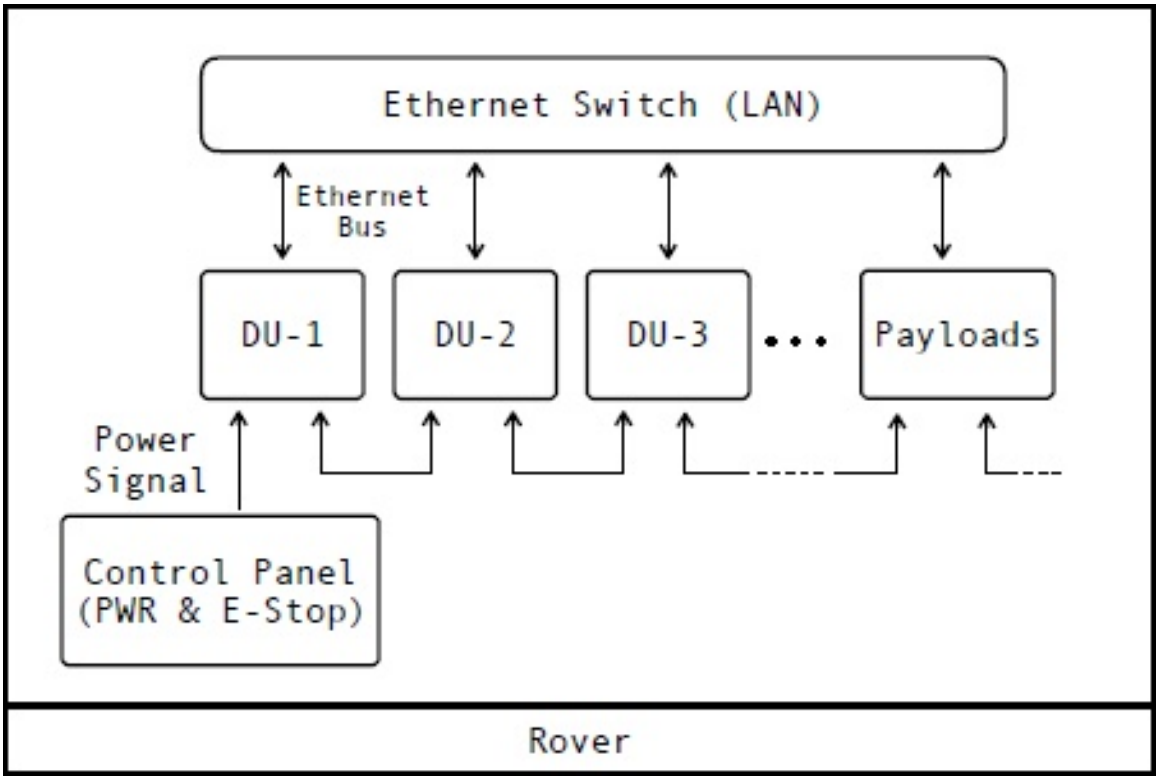

Figure 2.2 – AgBot: Modular Connectivity Illustration.

2.1.1 Drive Unit

The Drive Unit (DU) is a self-contained unit comprised of a electrical power sources (batteries), actuators, compute units (typically a Raspberry Pi or Nvidia Jetson Orin), an optional customizable sensor suite, a network switch, and any additional electrical components used to maintain control of the rover. Figure 2.2 illustrates the design and communication architecture between the drive units and the rover payload. Each DU follows the aforementioned template, however, other components can be equipped to allow for additional functionality of the rover. Such a configuration can be seen in Figure 2.3 where the drive unit is equipped with a sensor suite consisting of differential RTK-GPS antenna and

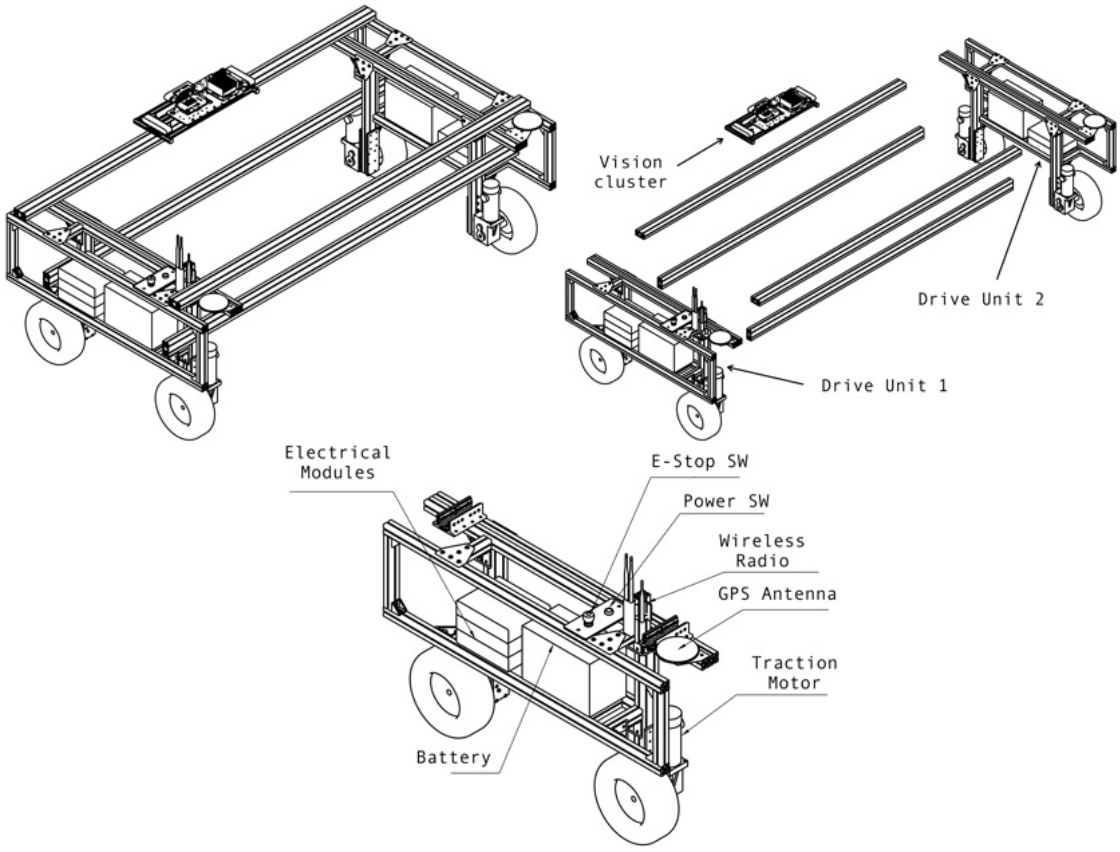

Figure 2.3 – AgBot: CAD Design with 2 Drive Units

laser sensors, which assists with various functionality such as GPS-based navigation and docking. Additionally, each DU is equipped with a closed-loop controller, responsible for designating movement.

2.1.2 Rover System

As observed in Figure 2.3, the module design of the rover system consists two or more DUs and is designed to allow the attachment of optional sensor suites and external tools. The main chassis is constructed using T-slotted frame rails, connecting the drive units to one another. The decision to utilize T-slot rails for the main chassis is primarily driven by the flexibility and structural integrity supported by the aluminum material. Further the design allows the rover to be reconfigured to meet the width and height requirements of varying agricultural environments. Additionally, the modular and easily re-configurable design greatly assists with rapid experimentation. The average weight of a rover with a two

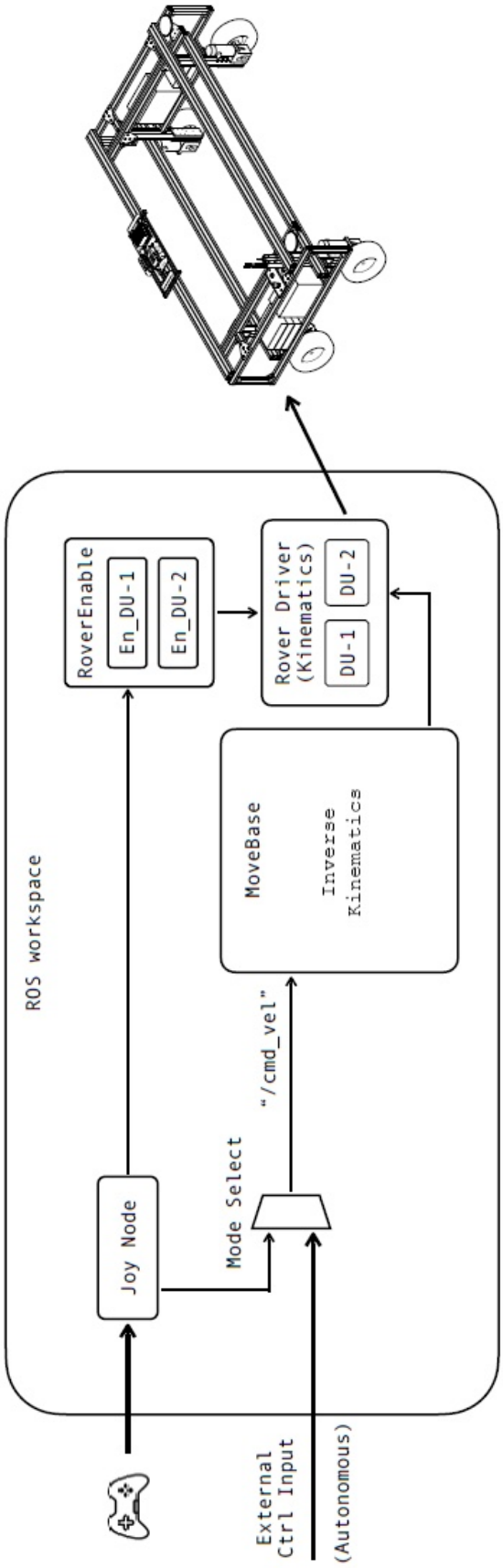

Figure 2.4 – AgBot: ROS System Sketch for Rover Motion.

DU configuration weights is approximately $100kg$ and is capable of carrying an additional $200kg$. For computing purposes, one DU is designated as the master which contains the rover's inverse kinematic transform to determine wheel speed set points based on rover linear and angular velocity commands.

2.2 Agbot: Software Motion Control and Design

The AgBot software is modular and is implemented through the use of ROS 2. ROS is a data communication framework commonly used in robotics and provides data distribution and communication between different software modules that may be executed on various computing platforms across a network [18]. For this work, the AgBot software implements the monitoring of joystick inputs by either a realtime pilot or supervisory operator, the selection between different motion control inputs (such as a joystick or an external controller), the transformation of robot level velocity commands to drive unit speed commands, and interface functions between various sensors and motors. Other more sophisticated functions have been developed for the AgBot but were not used for the work presented here.

As seen in Figure 2.4, a joystick or external controller specifies set points for linear velocity v_x and an angular velocity ω_z. Controller buttons are used to specify this selection and to enforce a safety enable for motion. When in manual control mode, the controller's joysticks are responsible for setting the velocity set points for the rover. Given the active linear and angular velocity commands, the MoveBase function performs the vehicle's inverse kinematic transform to determine the wheel speed setpoints for each DU.

The LiDAR-based vehicle controller developed for this project is implemented as an external controller. Its objective is to slowly drive the rover down the center of a vineyard lane. Given this, it uses a simple linear velocity control approach consisting of a discrete on/off selection between a command of 0.3 m/s, indicating slow forward motion, or 0 m/s to cease rover movement. Steering down the center of the lane is controlled by specifying the angular velocity command, ω_z. This control capability depends on the ability to perceive the position and orientation of the rover within the lane. The methods explored for doing this are detailed in chapters 3 and 4.

Chapter 3

Branch Detection Based Steering System

The general structure of both branch and trunk detection perception methods of navigation explored in this thesis follow the process shown in Figure 3.1 and again in Figure 4.1. First, LiDAR data is acquired and filtered to perceive the local environment around the rover. Next, rover position and orientation data is extracted to obtain a cross-track error, ε_{ct}, and a heading error, ε_{ψ}, with respect to the estimated centerline, which can be seen in Figure 3.4. Lastly, a PI controller uses the errors to specify a realtime setpoint for the rover's angular velocity, ω_z, with the objective of driving the rover down the centerline of the row. The content in this chapter encompasses the topic of branch detection based vineyard navigation. Section 3.1 guides the reader through the intuition and integration process for branch based vineyard navigation. Section 3.2 describes the experimental process for system performance verification. Section 3.3 provides an in-depth analysis on the system performance based on metrics obtained in experimentation.

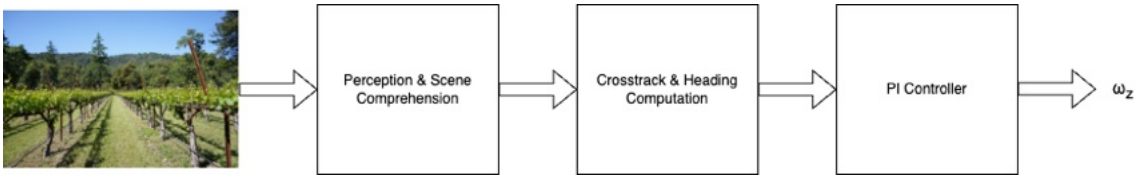

Figure 3.1 – Perception Based Vineyard Navigation System Overview

3.1 Branch-Based Navigation Implementations

As the name implies, this approach uses the measured distances of the foliage and the branches to safely traverse the row of a vineyard. To do this, we position our LiDAR sensor such that the base of our sensor is the same height as the branches of the plants. This allows us to convert the pointcloud to a *LaserScan* of only the region containing the branches and some additional foliage. Using the *LaserScan*, we extract conic regions of interests on the left and right side of the rover to determine the rover position and orientation. The estimated rover position and orientation are used to derive the linear and angular velocity commands to conduct in-row vineyard navigation.

There are some pitfalls to this method of navigation. Due to the nature of this technique, the performance of the algorithm is heavily susceptible to factors such as variations in seasonal growth and differences in branch densities. Additionally, because this method of navigation uses the measured shortest distances from the rover to the branches, it also assumes the rover is oriented within $\pm 45^{\circ}$ with respect to the vineyard. This operating range of $\pm 45^{\circ}$ allows us to ensure both the left and right rows are within view for our algorithm to perform correctly.

3.1.1 Pointcloud Acquisition

In the initial stage of scene comprehension, depicted in Figure 3.2, the process begins with capturing the scene as a point cloud at a rate of 10 Hz. The point cloud, as observed in Figure 3.3, is a 3D map made of points in space, often created using laserscanning technology like LiDAR. Each point represents a specific location at which the LiDAR sees a portion of a vine, forming a detailed digital representation of the environment. Utilizing the ROS 2 package, *pointcloud_to_laserscan*, the system translates the three-dimensional snapshot of the environment into a two-dimensional slice, referred to as a *LaserScan*.

The algorithm uses polar coordinates presented by the *LaserScan* to delineate two conic regions of interests. The structure of the *LaserScan* can be seen in equation 3.1 where δ_θ is the distance of the nearest obstacle from the LiDAR to a point at a given angle. In this setup, the 0th index of the *LaserScan*, corresponding to 0°, is directed toward the rear of

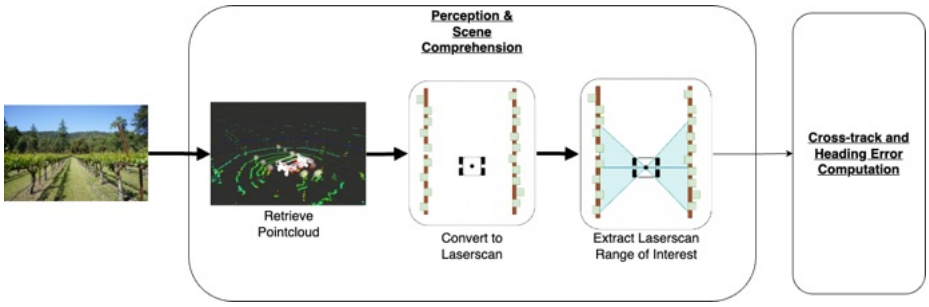

Figure 3.2 – Branch Detection: Perception and Scene Comprehension Block.

the rover, while the 180th index, corresponding to 180°, faces the front of the rover.

$$LaserScan = \{\delta_\theta \mid \theta \in [0,\ 359]\} \tag{3.1}$$

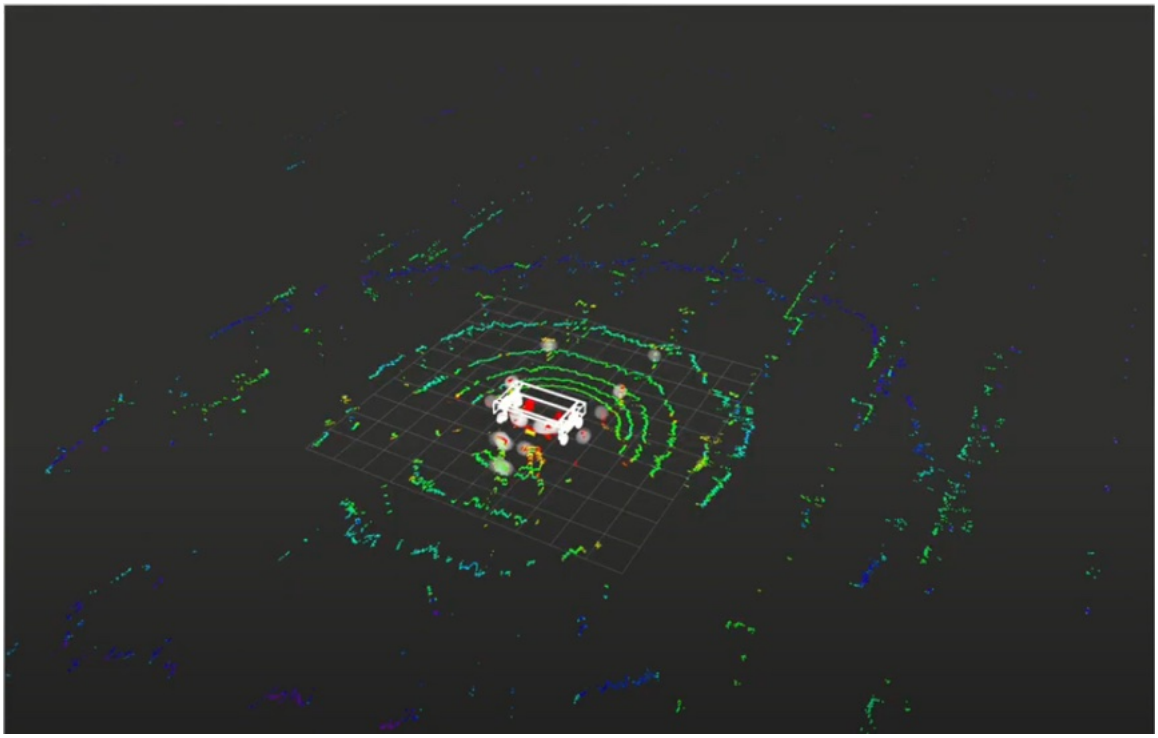

Figure 3.3 – Branch Detection: A Pointcloud of the Vineyard.

In the following step, the algorithm takes a subset of this structure to derive the location of the corresponding rows.

3.1.2 Branch Detection and Scene Comprehension

As observed in the diagram in Figure 3.4, the system initiates scene comprehension by partitioning the *LaserScan* set into two subsets, representing the left and right sides of the rover. As shown in the diagram, a row reference frame is defined with its origin at the center of one end of the row, and it's X unit vector pointed down the centerline between the left and right rows. Additionally, we define the width of the vineyard row as W. In Figure 3.4 we have chosen the world frame to define the vineyard location. The rover frame is attached to the LiDAR (which is also at the centroid of the rover).

The assignment of the frame and definition of width is done assuming that the centerline is straight and the width is constant; of course, there is variation of both of these parameters in a real field. For the vineyard in which testing was done, this was on the order of centimeters.

$$Range_l = \{\delta_\theta \in LaserScan \mid 45^\circ \leq \theta < 135^\circ\} \tag{3.2}$$

$$Range_r = \{\delta_\theta \in LaserScan \mid 225^\circ \geq \theta > 315^\circ\} \tag{3.3}$$

$$Range_{aggregate} = [Range_l,\ Range_r] \tag{3.4}$$

In equations 3.2 and 3.3, the dimensions of $Range_l$ and $Range_r$ are both 1×90. $Range_{aggregate}$ is defined by concatenating $Range_l$ and $Range_r$, resulting in a dimension of 1×180. By employing a majority voting mechanism, the system assesses the density of each subset to determine the rover's status. The system states are *in-row* or *out-of-row*. Should the majority of distance values surpass a predefined threshold $\delta_{\text{threshold}}$, it is concluded that the rover has strayed from the vineyard row, prompting a cessation of movement. This decision-making process is elaborated in equation 3.5 and visualized in the system flow chart depicted in Figure 3.5.

$$roverstate = \begin{cases} \text{in-row} & \frac{len(Range_{aggregate})}{2} < len(\{\delta_\theta \in Range_{aggregate} \mid \delta_\theta \leq \delta_{threshold}\}) \\ \text{out-of-row} & else \end{cases} \tag{3.5}$$

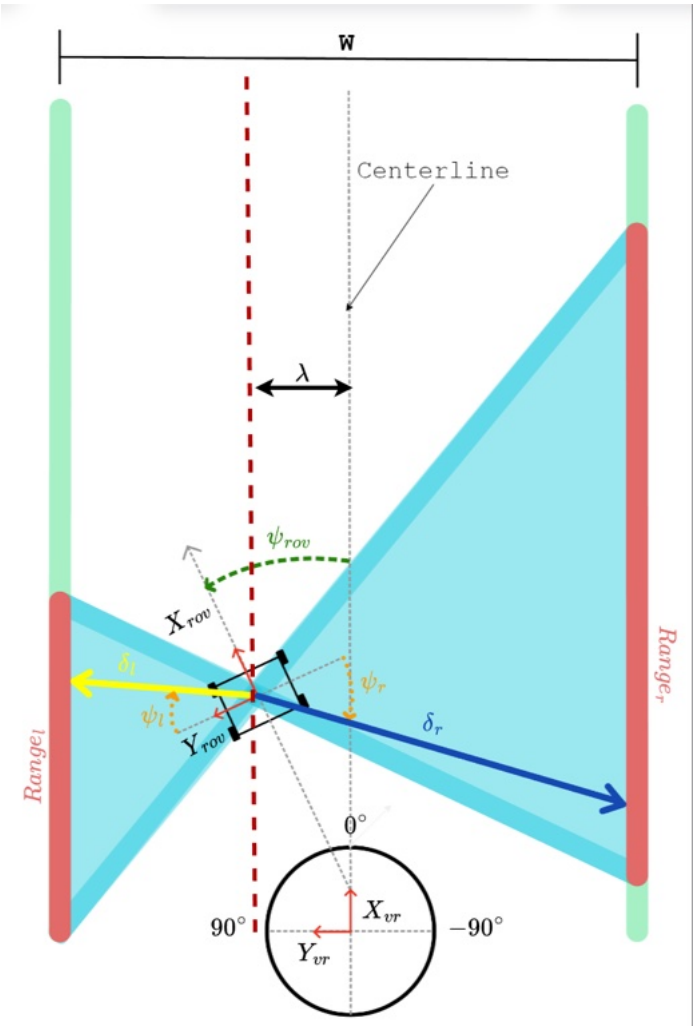

Figure 3.4 – Branch Detection: Rover Diagram

3.1.3 Rover Position and Orientation Derivation

To estimate the position and orientation of the rover in relation to the vineyard, the algorithm relies on informed assumptions based on minimum lateral distances.

In Figure 3.4, we see a rover positioned to the left of the centerline in a vineyard of arbitrary width W. The lateral offset of the rover with respect to the vineyard centerline is denoted λ. We can also see the LiDAR detection range indicated by the shaded blue region. Within each conic region, we see the queried estimated minimum distance, denoted as δ_l and δ_r. Rover heading, ψ_{rov}, is the angle measured in the counterclockwise direction between the rover's longitudinal axis, X_{rov}, and the X_{vr}.

From Figure 3.4, we see the rover querying the detected minimum distances, δ_l and δ_r,

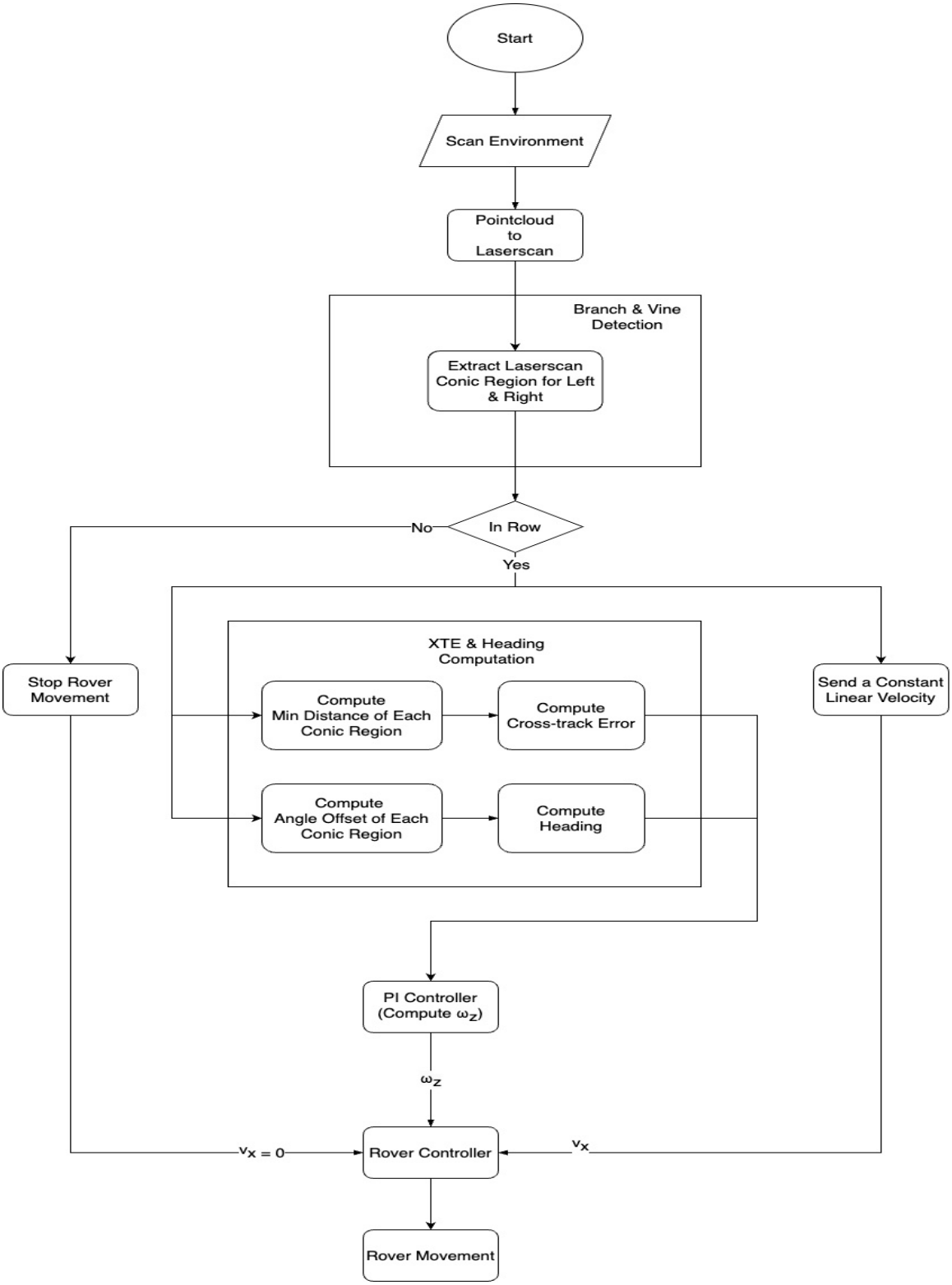

Figure 3.5 – Branch Detection: Process Flow Chart.

from the respectively from the range sets $Range_l$ and $Range_r$. The two scalar values δ_l and δ_r, allow us to compute the estimate lateral position of the rover relative to the vineyard row. In an ideal scenario, the sum of δ_l and δ_r should equate to the width of the vineyard row, W. While the two distances may not always represent the true minimum distances from the rover to the respective left and right rows, the queried measurements are sufficient for determining the rover's lateral offset from the centerline.

$$\delta_l = min(Range_l) \tag{3.6}$$

$$\delta_r = min(Range_r) \tag{3.7}$$

Using the distances, the lateral position of the rover, λ, is defined in the row reference frame. This is the distance of the rover frame's origin from the centerline measured along the $+Y_{vr}$ direction.

$$\lambda = \frac{\delta_r - \delta_l}{2} \tag{3.8}$$

We know the *LaserScan* is a vector of 360 elements where each index corresponds to degree at which a distance is measured. In the ideal scenario, the rover is pointed directly forward at ($\psi_{rov} = 0^\circ$) and the minimum distance measured on the left side should be the 45th element of $Range_l$ (also the 90th element of *LaserScan*). The *argmin* of $Range_l$ provides the index of the minimum element of the vector. The rover heading based on the left area of interest, $Range_l$, is computed as ψ_l. Similarly, we compute ψ_r for the right area of interest, $Range_r$. ψ_l and ψ_r are independently computed using the following equations.

$$\psi_l = argmin(Range_l) - 45^\circ \tag{3.9}$$

$$\psi_r = argmin(Range_r) - 45^\circ \tag{3.10}$$

We assume the rover is oriented no greater than 45° and no less than -45°. We also assume both rows are present, and therefore, we compute the rover heading, ψ_{rov} as an average of ψ_l and ψ_r. However, in the future, our control scheme can rely on only one of

the sides with speed constraints.

$$\psi_{rov} = \frac{(\psi_l + \psi_r)}{2} \tag{3.11}$$

3.1.4 Error Computation

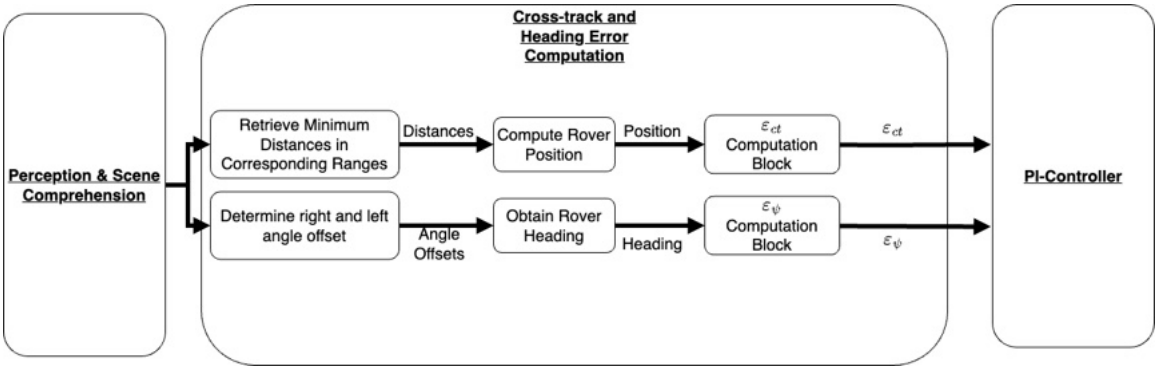

Figure 3.6 – Branch Detection: Cross-track and Heading Error Computation Block

To enable closed-loop control of a system, it is essential to quantify an error. Consequently, as observed in Figure 3.6, we utilize the cross-track and heading errors of our system to accurately guide the rover in its intended direction. To compute the cross-track error, we assume the rover is laterally centered in the vineyard when the left and right distances are equal. Under this condition, our ideal rover position should be 0 m from the center of the vineyard row. Hence, we calculate the cross-track error using equation 3.12. Due to hardware limitations of the LiDAR deadzone, assume a maximum ε_{ct} of 0.5 m.

$$\varepsilon_{ct} = 0 - \lambda \tag{3.12}$$

Similarly, in the ideal scenario where the rover is pointed directly forward, our computed heading should align with $\psi_{rov} = 0^{\circ}$. Consequently, our heading error is computed using the following equation.

$$\varepsilon_{\psi} = 0 - \psi_{rov} \tag{3.13}$$

Note that the rover position is positive in magnitude when the rover is positioned to the

left side of the centerline, and therefore, from equation 3.12, the crosstrack error, ε_{ct}, is negative when the rover is positioned on the left of the centerline, and positive when the rover is positioned to the right side of the centerline. Furthermore, as described in equation 3.13, the heading increases as the rover rotates counterclockwise, resulting in a negative heading error, ε_{ψ}, when the rover is oriented to the left, and a positive heading error when the rover is oriented to the right.

3.1.5 Rover Control

As shown in equation 3.14, the control law for linear velocity is a discrete decision, either stopped when not in a row or moving forward at a constant value when the rover is within the vineyard row. For the experiments performed in this thesis, a linear velocity of 0.3 m/s was used.

$$\upsilon_x = \begin{cases} \upsilon_{constant} & \text{rover in-row} \\ 0 & \text{rover out-of-row} \end{cases} \tag{3.14}$$

Based on prior work on mobile robots by Kitts et al. [19], the control scheme is defined in this section and visualized in Figure 3.7. In the diagram, we have chosen to define the vineyard using the row reference frame, which is placed at the beginning and center of the row. Using our branch detection method, we define the reference path as the centerline of the vineyard row. To the right of the centerline, we see the rover is laterally offset by our cross-track error denoted as ε_{ct}. The rover heading is also offset by ψ_{rov} *degrees*. To minimize the cross-track error, our rover must steer ψ_{Des} *degrees*, which is the product of our cross-track error, ε_{ct}, measured in *meters*, and a gain, k_{ct}, expressed in deg/m.

$$\psi_{Des} = k_{ct} \cdot \varepsilon_{ct} \tag{3.15}$$

We add ε_{ψ} *degrees* to account for the orientation offset of our rover. This provides us

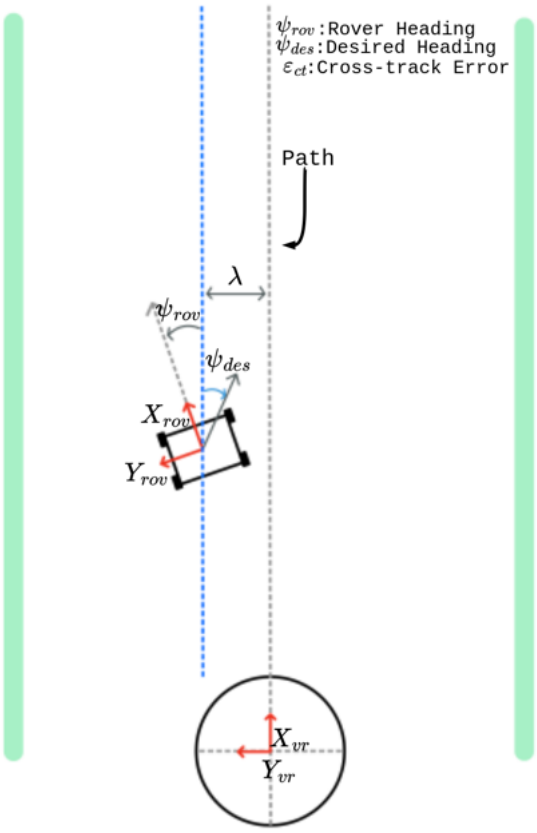

Figure 3.7 – Branch Detection: Control Example Diagram

with the error term, ε, expressed in degrees. We define our error term, ε, below.

$$\varepsilon = (\varepsilon_{\psi} + \psi_{Des}) \cdot (\frac{\pi}{180}) \tag{3.16}$$

Using a PI-controller, we control the angular velocity of our rover, ω_z, expressed in rad/s. This allows the rover to steer towards the defined path if it is not already on it. We define our controller in equation 3.17.

$$\omega_z = k_p(\varepsilon) + k_i \int (\varepsilon) dt \tag{3.17}$$

Given our operating range of a maximum heading error of $\pm 45^{\circ}$ and a maximum cross-track error $\pm 0.5\ m$, the gains are selected such that the control input, ω_z, is within the range of $\pm 0.3\ rad/s$. This is done with the intent that the controller is typically not saturated during operation. Consequently, we have chosen a proportional gain of $kp = 0.573/s$, an

integral gain of $k_i = 0.458/s^2$, and a $k_{ct} = 80\ deg/m$.

The steering control law in equation 3.17 results in being able to eliminate the cross-track error, while pointing the rover in the proper direction down the row.

3.2 Performance Verification System

To prepare our experimental environment, we positioned a 0.625 *cm* wide red strap approximately centered between the two vineyard rows. Although we positioned the strap to the best of our ability, because the position of the strap is approximate, the accuracy may vary by $\pm$ a few centimeters due to the unevenness of the terrain. This red strap is strictly used as an independent truth measure of the errors for evaluating the perception and control system. Based on averaging 10 measurements taken at about 10 *m* intervals, the row width was estimated to be 3.4 *m*

By leveraging this setup, we can establish a ground truth for the cross-track error of our vineyard. This involves positioning the depth sensing camera at the center of the rover and measuring the distance between it and the red nylon strap. This distance serves as a reliable indicator of how accurately the rover follows the desired path between the vineyard rows.

Our performance verification consists of a NVIDIA Jetson Orin and a Zed stereo camera mounted precisely at the center of the rover. With the Zed stereo camera angled to focus on the ground directly beneath the center of the rover, as depicted in Figure 3.8, we capture depth details of the terrain as the rover moves. This configuration grants us access to highly accurate real-time depth information of both the red strap and the location directly under the rover. Retrieving the depth information, in *meters*, of the two areas on the field, allows us to map the lateral pixel offset to a measurement in *meters*. Furthermore, we assume the position of the red strap and the point directly beneath the center of the rover are in a perpendicular plane to the camera direction. Although this may not always be the case, we found the approximation to yield consistent results.

Our ROS 2-based verification system's software stack begins by detecting all red pixels within the image, creating a bit mask where red-hued pixels are white and all others are black. Using tools from the OpenCV library, the mask undergoes a Gaussian blur to

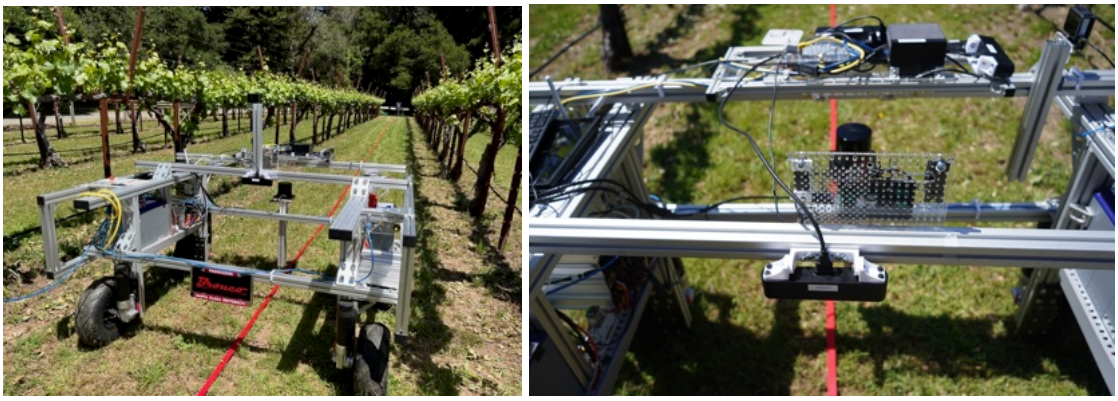

Figure 3.8 – Branch Detection: Performance Verification System

reduce image noise, thereby enhancing accuracy and simplifying computations along the verification pipeline.

After frame preprocessing, the system pinpoints the white pixel nearest to the frame's center, determining the red strap's location. By then querying the depth of the rover's central point and the detected white pixel, we calculate the distance between the two points in *meters*, establishing the rover's position relative to the strap. This information allows us to assess the rover's deviation from the vineyard row's center, providing us with a ground truth cross-track error based on the red strap's location. The reported cross-track error is published on the ROS 2 network and recorded in a rosbag during an experimental run.

3.2.1 Experimental Process

For our experimentation, we set up our system to operate autonomously within the vineyard row, while we closely monitored its performance in case of any anomalous behavior. The rover navigated along the length of a single row, which typically extended approximately 100 *meters*.

Throughout each run, we recorded data on the rover's reported cross-track error, heading error, and corresponding velocities, alongside the ground truth cross-track error. This comprehensive dataset allowed us to analyze the rover's performance in real-time and assess its alignment with the desired trajectory.

Following each run, we plotted the results to ensure the fidelity of our data. This visual and quantitative analysis provided valuable insights into the rover's behavior and perfor-

mance within the vineyard environment.

In an attempt to optimize the rover's navigation capabilities, we iteratively tuned the gains of the PI-Controller after each run.

3.3 Results

The key performance metrics of our system include mean cross-track error, root mean square error (RMSE), and standard deviation. These metrics allow us to quantify the accuracy and reliability of the navigation system by providing valuable insights into the system's ability to maintain the desired trajectory within the vineyard rows. We have computed the following metrics reported by both our navigation system and our performance verification system.

Our metrics for performance evaluation, measured in *meters*, are described below.

- **Mean Cross-track Error**: Mean cross-track error gives us the average measured lateral displacement of the rover with respect to the centerline. A nonzero mean error suggests a consistent bias in one direction.

$$\mu_{ct} = \frac{1}{N}\sum_{i=1}^{N} \varepsilon ct_i \mid N = \#\text{ of error points} \tag{3.18}$$

- **Root Mean Square Error (RMSE)**: RMSE provides an overall measure of accuracy by calculating the average size of errors, regardless of their direction. A lower RMSE indicates better accuracy and precision in following the desired path within the vineyard rows.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (\varepsilon ct_i)^2} \mid N = \#\text{ of error points} \tag{3.19}$$

- **Standard Deviation**: Standard deviation shows us how much the rover's position varies around the mean error. A higher standard deviation means more variability in performance, while a lower standard deviation indicates more consistency in maintaining the desired trajectory.

$$Std = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\varepsilon ct_i - \mu_{ct})^2} \mid N = \#\text{ of error points} \tag{3.20}$$

3.3.1 System Performance Analysis

Table 3.1 displayed the performance metrics, mean cross-track error, RMSE, and standard deviation, as measured by both the navigation system and the performance verification system. Although, we conducted various experiments to allow us to tune and ensure consistent performance, the table presented below showcases the results from our most notable run.

Table 3.1 – Branch Detection: Performance Metrics at Steady State

Branch Navigation: System Reported Performance			
	Mean Error (m)	**RMSE (m)**	**Standard Deviation (m)**
Sys Perceived	-0.0734	0.1898	0.1751
Sys Truth	-0.1727	0.1965	0.0939

As indicated by the mean cross-track errors, the performance verification system reports a higher bias to the right side of the vineyard, than what is perceived by the navigation system. The similar measurements between the RMSE values reported by the two systems indicates the rover is able to maintain an overall cross-track accuracy of 0.19 m. Lastly, the verification system indicates lower variability in performance, with a measured standard deviation of 0.0939 m, compared to the navigation system's perceived standard deviation of 0.1751 m.

After reviewing the performance of existing reactive navigation algorithms, we believe a RMSE of 0.19 m is on par with current research standard. A major drawback of this system, however, is the significant amount of noise introduced by the gaps in the vineyard foliage. The algorithm greatly depends on the existence of foliage, and any gaps and sparseness often results in incorrect minimum lateral distance measurements. This directly impacts our control of the rover. Furthermore, we attribute the better performance reported by the verification system, to the fact that there is less noise when measuring the terrain, compared to the performance metrics determined by our navigation system through the

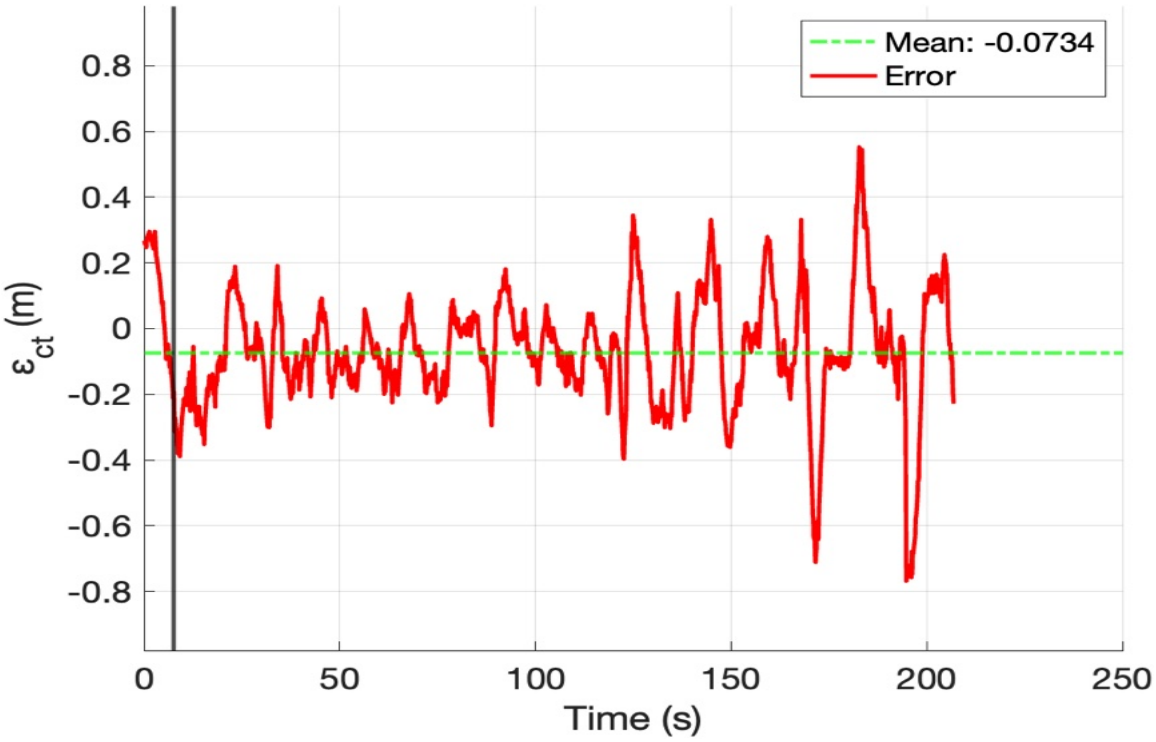

Figure 3.9 – Branch Detection: System Perceived Performance

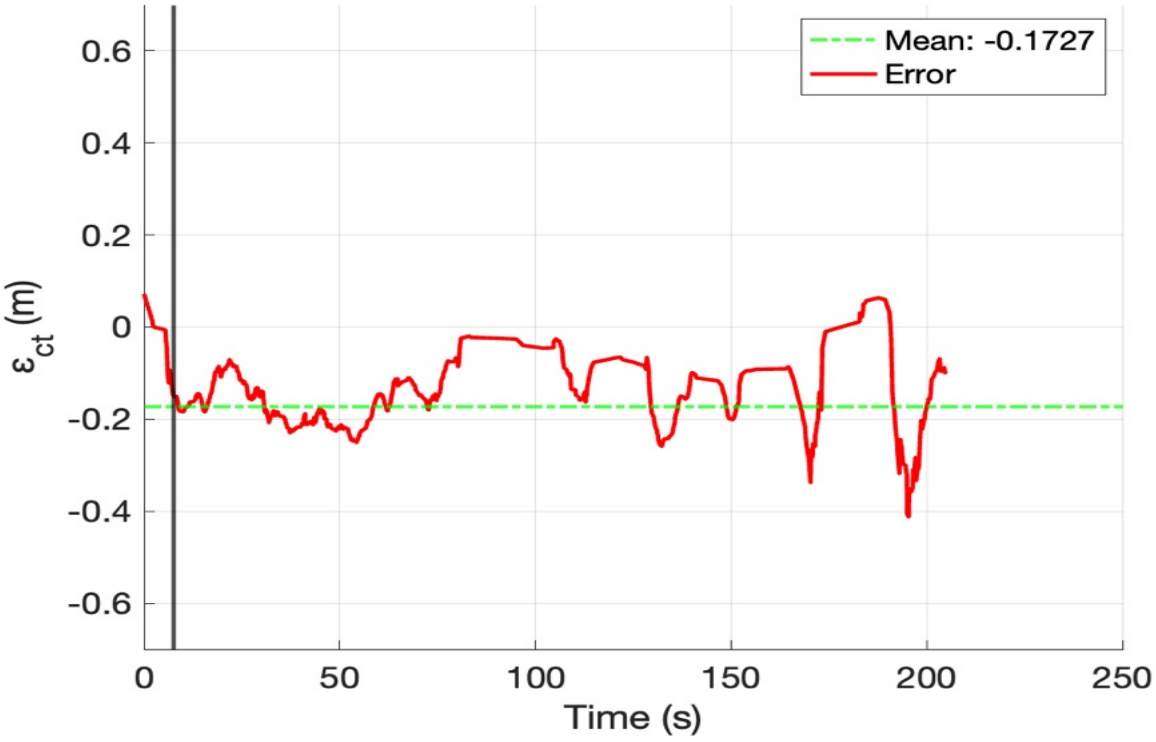

Figure 3.10 – Branch Detection: Ground Truth Performance

measurement of the sparse foliage. The consistency of the terrain directly leads to a more consistent performance measurement of our rover.

3.4 Summary

In chapter 3, we introduce a novel method of reactive in-row navigation that leverages the vineyard foliage to steer the rover along the centerline. We also provide insight to our experimentation process for validating the navigation algorithm. In section 3.3.1, we present the results from our most notable run. We found our overall navigation accuracy of $0.19\ m$ to be on par with current research standard. In the same section, we also address the greatest flaw of our system, which is leveraging the noisy foliage to steer our rover along the centerline.

Chapter 4

Trunk Detection Base Navigation System

The content in this chapter encompasses the topic of trunk detection based vineyard navigation. Section 4.1 guides the reader through the intuition and integration process for trunk based vineyard navigation. Section 4.2 describes the experimental process for system performance verification. Section 4.3 provides an in-depth analysis on the system performance based on metrics obtained in experimentation.

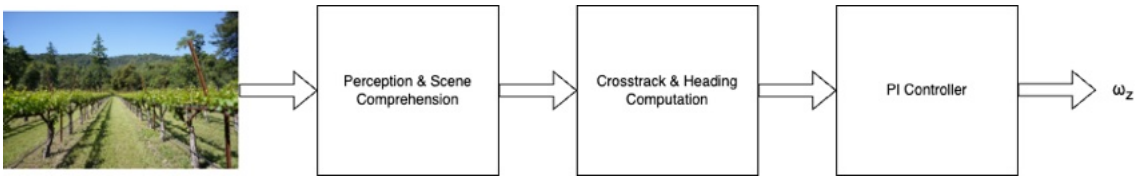

Figure 4.1 – Perception Based Vineyard Navigation System Overview

The general structure of both branch and trunk detection perception methods of navigation explored in this thesis follow the process shown in Figure 4.1 and previously in Figure 3.1. First, LiDAR data is acquired and filtered to perceive the local environment around the rover and to estimate the location of the lines along which the vines are located. Next, rover position and orientation data is extracted to obtain a cross-track error, ε_{ct}, and a heading error, ψ_{rov}, with respect to the estimated centerline. Lastly, a PI controller uses the error to specify a realtime setpoint for the rover's angular velocity, ω_z, with the objective of driving the rover down the centerline of the row.

4.1 Trunk-Based Navigation Implementation

This method of navigation uses the trunks of the plants to construct a boundary for the rover to safely traverse the vineyard row. First, the LiDAR is positioned at a constant height of approximately 0.6 *m* above the ground to capture trunks of the plants. We then convert the pointcloud emitted by the LiDAR, to a 2D *LaserScan* of the trunks using the "pointcloud_to_laserscan" ROS package. Next, we compute two independent clusters to detect the vineyard rows on either side of the rover. A line is fitted to its respective cluster to estimate the vineyard rows, allowing us to approximate the rover's position and orientation based on the lateral distances from the rover to the lines and the slopes of those lines, respectively. The rover position and orientation are used to derive the linear and angular velocity commands to conduct in-row vineyard navigation.

4.1.1 Pointcloud Acquisition

In the initial stage of scene comprehension, depicted in Figure 4.2, the process begins with capturing the scene as a point cloud at a rate of 10 Hz. The point cloud, as observed in Figure 4.3, is a 3D map made of points in space, often created using laserscanning technology like LiDAR. Each point represents a specific location at which the LiDAR detects a portion of a vine, forming a detailed digital representation of the environment. In this particular scenario, the points represent the trunks of the plants which we assume are organized in a line for each row. We understand there exists a variation of objects that may be misinterpreted as a point in a vineyard environment. As a result, additional filtering of the points may improve the performance of the line fitting algorithm discussed in the later sections, however, this is yet to be implemented and tested. Utilizing the ROS 2 package, *pointcloud_to_laserscan*, the system translates the three-dimensional snapshot of the environment into a two-dimensional slice, referred to as a *LaserScan*.

Following this capture, the *LaserScan* undergoes a refinement process. The *LaserScan* is sifted to extract pairs of (θ, δ_θ), where the distance, δ_θ at a particular angle, θ, falls within predefined distance threshold bounds, as delineated in equation 4.1. In this thesis, we used a lower distance threshold, δ_{lower} of 1 *meter*, and a upper distance threshold, δ_{upper},

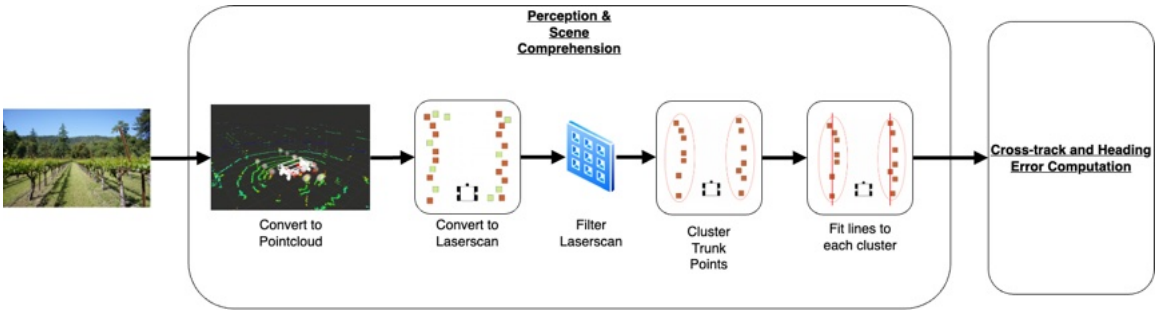

Figure 4.2 – Trunk Detection: Perception and Scene Comprehension Block.

of 3.5 *meters*. In this setup, the 0th index of the *LaserScan*, corresponding to 0°, is directed toward the rear of the rover, while the 180th index, corresponding to 180°, faces the front of the rover.

$$LaserScan = \{\delta_\theta \mid (\theta \in [0,\ 359]) \wedge (\delta_{lower} \leq \delta_\theta \leq \delta_{upper})\} \tag{4.1}$$

Given that the points are in polar coordinates, represented as $(\theta,\ \delta_\theta)$, we convert these points to Cartesian coordinates $(x,\ y)$ using the following equations.

$$x = \delta_\theta \cos\theta, \tag{4.2}$$

$$y = \delta_\theta \sin\theta \tag{4.3}$$

This transformation allows us to express the positions of the points in a two-dimensional xy-coordinate plane, necessary for fitting a line to the points as seen in Figure 4.4.

4.1.2 Trunk Detection and Scene Comprehension

As shown in Figure 4.4, a row reference frame is defined with its origin at the center of one end of the row, and its X unit vector pointed down the centerline betwen the left and right rows. This reference frame has been chosen to define the vineyard location. Additionally, we define the width of the vineyard row as W.

The assignment of the frame and definition of width is done assuming that the centerline is straight and the width is constant; of course, there is variation of both of these parameters in a real field. For the vineyard in which testing was done, this was on the order

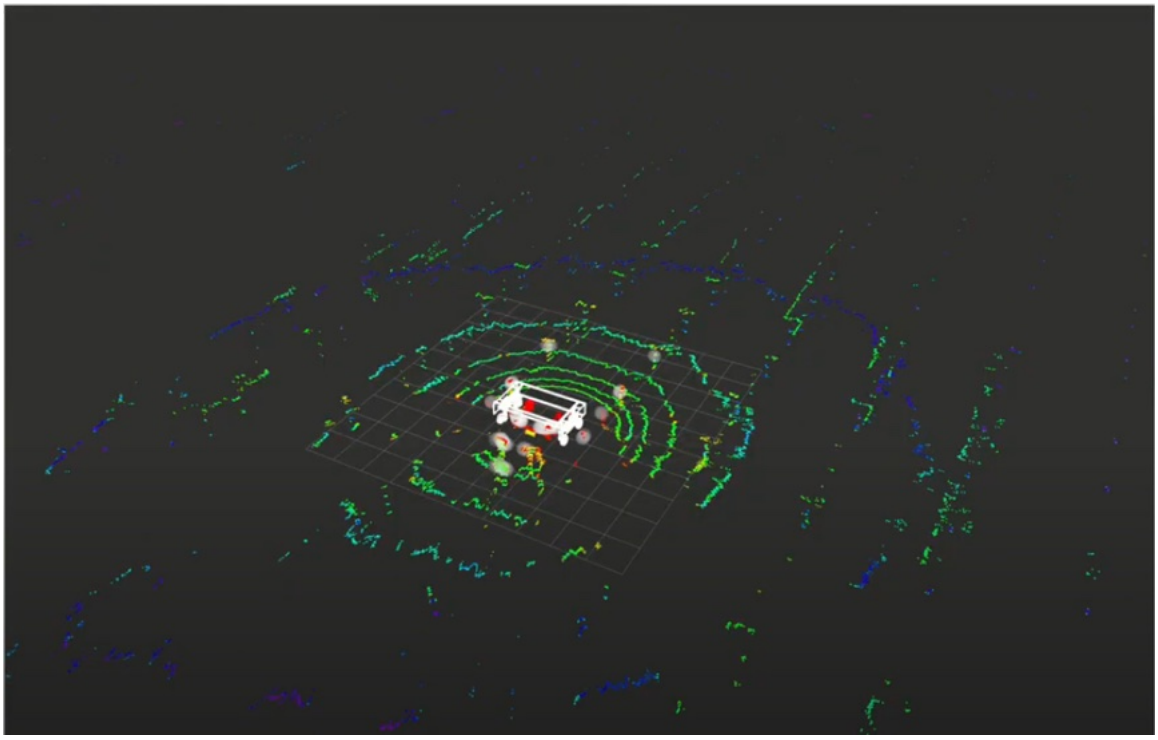

Figure 4.3 – Trunk Detection: A Pointcloud of the Vineyard.

of centimeters.

In the diagram, we see the LiDAR range of interest, indicated by the blue circle. The range of interest of the LiDAR, can be adjusted. Within the circle, the system conducts scene comprehension of the environment by employing the k-means clustering algorithm, partitioning the data into two discernible clusters. These clusters are indicated by the blue interior ovals in the diagram. Building upon this comprehensive segmentation, the system leverages the scipy library's curve-fitting method to conduct least squares on the two corresponding clusters at every time step. The resulting two lines, computed in the rover frame, provide an estimated boundary for the rover to follow along.

4.1.3 Rover Position and Orientation Derivation

To determine the rover position relative to the vineyard row, λ, the system transforms the two fitted lines from slope-intercept form to standard line form $Ax + By = C$. Utilizing the standard form of the lines, our navigation system calculates the shortest distances from the

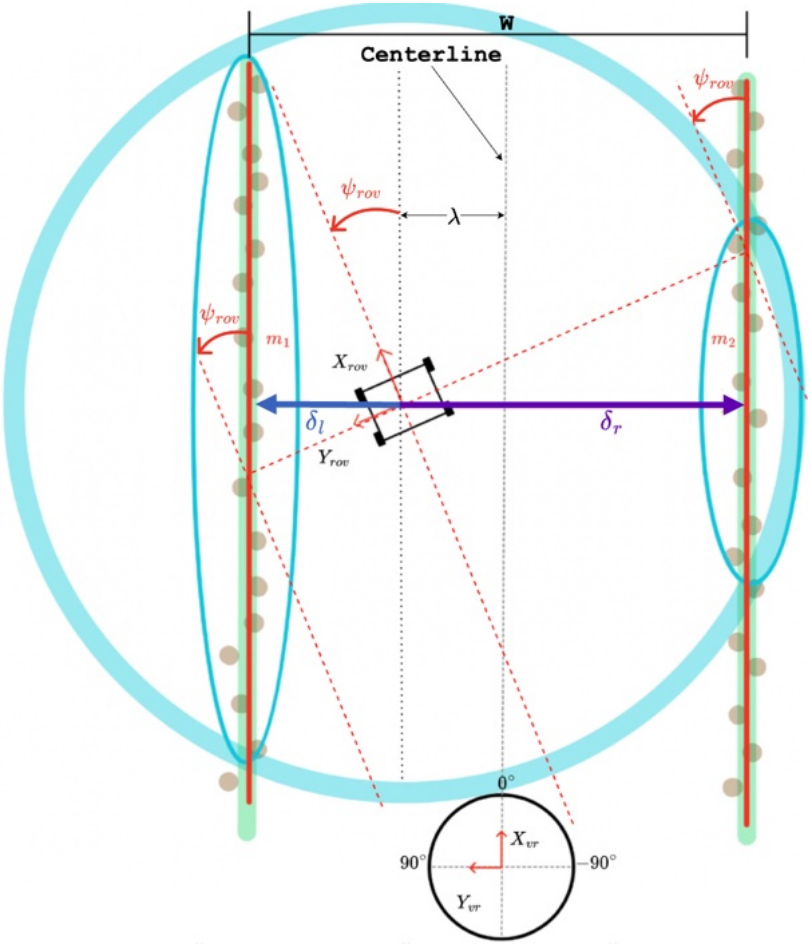

Figure 4.4 – Trunk Detection: Rover Diagram

rover to the two sides, denoted as δ_l and δ_r.

$$\delta_l = \frac{|A_1x + B_1y + C_1|}{\sqrt{(A_1)^2 + (B_1)^2}} \tag{4.4}$$

$$\delta_r = \frac{|A_2x + B_2y + C_2|}{\sqrt{(A_2)^2 + (B_2)^2}} \tag{4.5}$$

From Figure 4.4, the lateral distance of the rover, λ, is defined in the vineyard row frame. This is the distance of the rover frame's origin from the centerline measured along the $+Y_{vr}$ direction.

$$\lambda = \frac{\delta_r - \delta_l}{2} \tag{4.6}$$

In the rover frame, the system computes the slope of the left and right line, m_1 and m_2.

Using the inverse tangent of the two slopes, we can extrapolate the rover's heading. The two values are averaged, and converted to degrees, to determine the rover heading, ψ_{rov}.

$$\psi_{rov} = \frac{180}{\pi} \cdot \frac{-(\arctan(m_1) + \arctan(m_2))}{2} \tag{4.7}$$

4.1.4 Error Computation

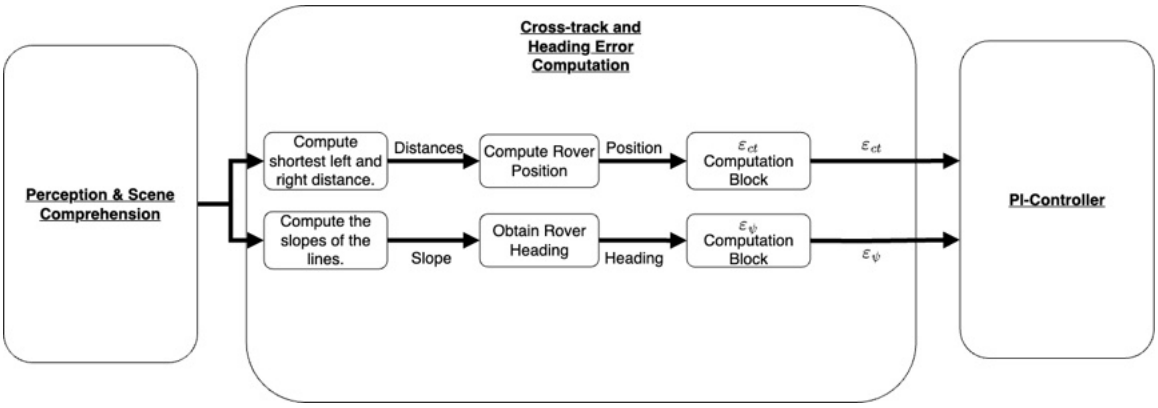

Figure 4.5 – Trunk Detection: Cross-track and Heading Error Computation Block

To enable closed-loop control of a system, it is essential to quantify an error. Consequently, as observed in Figure 4.5, we utilize the cross-track and heading errors of our system to accurately guide the rover in its intended direction. To compute the cross-track error, we assume the rover is laterally centered in the vineyard when the left and right distances are equal. Under this condition, our ideal rover position should be 0 m from the center of the vineyard row. Hence, we calculate the cross-track error using equation 4.8. Due to hardware limitations of the LiDAR deadzone, assume a maximum ε_{ct} of 0.5 m.

$$\varepsilon_{ct} = 0 - \lambda \tag{4.8}$$

Similarly, in the ideal scenario where the rover is pointed directly forward, our computed heading should align with $\psi_{rov} = 0°$. Consequently, our heading error is computed using the following equation.

$$\varepsilon_{\psi} = 0 - \psi_{rov} \tag{4.9}$$

4.1.5 Rover Control

As shown in equation 4.10, the control law for linear velocity is a discrete decision, either stopped when not in a row or moving forward at a constant value when the rover is within the vineyard row. For the experiments performed in this thesis, a linear velocity of 0.3 m/s was used. This decision process can be seen in 4.6.

$$\upsilon_x = \begin{cases} \upsilon_{constant} & \text{rover in-row} \\ 0 & \text{rover out-of-row} \end{cases} \tag{4.10}$$

Based on prior work on mobile robots by Kitts et al. [19], the control scheme is defined in this section and visualized in Figure 4.7. In the diagram, we have chosen to define the vineyard using the world frame, which is place at the beginning and center of the row. Using our trunk detection method, we define the reference path as the centerline of the vineyard row. To the right of the centerline, we see the rover is laterally offset by our cross-track error denoted as ε_{ct}. The rover heading is also offset by ψ_{rov} *degrees*. To minimize the cross-track error, our rover must steer ψ_{Des} *degrees*, which is the product of our cross-track error, ε_{ct}, measured in *meters*, and a gain, k_{ct}, expressed in deg/m.

$$\psi_{Des} = k_{ct} \cdot \varepsilon_{ct} \tag{4.11}$$

We add ε_{ψ} *degrees* to account for the orientation offset of our rover. This provides us with the error term, ε, expressed in degrees. We define our error term, ε, below.

$$\varepsilon = (\varepsilon_{\psi} + \psi_{Des}) \cdot (\frac{\pi}{180}) \tag{4.12}$$

Using a PI-controller, we control the angular velocity of our rover, ω_z, expressed in rad/s. This allows the rover to steer towards the defined path if it is not already on it. We

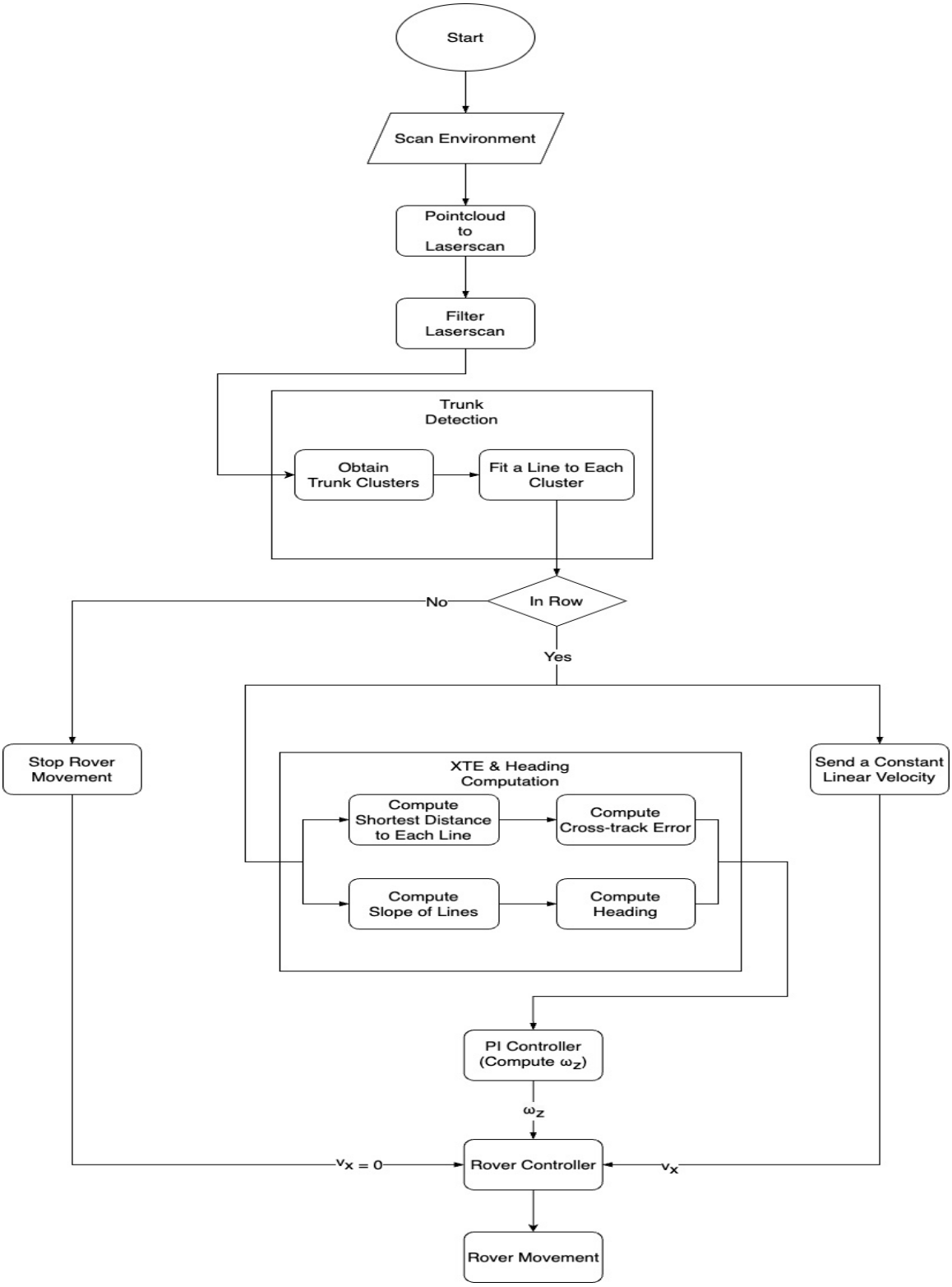

Figure 4.6 – Trunk Detection: Process Flow Chart

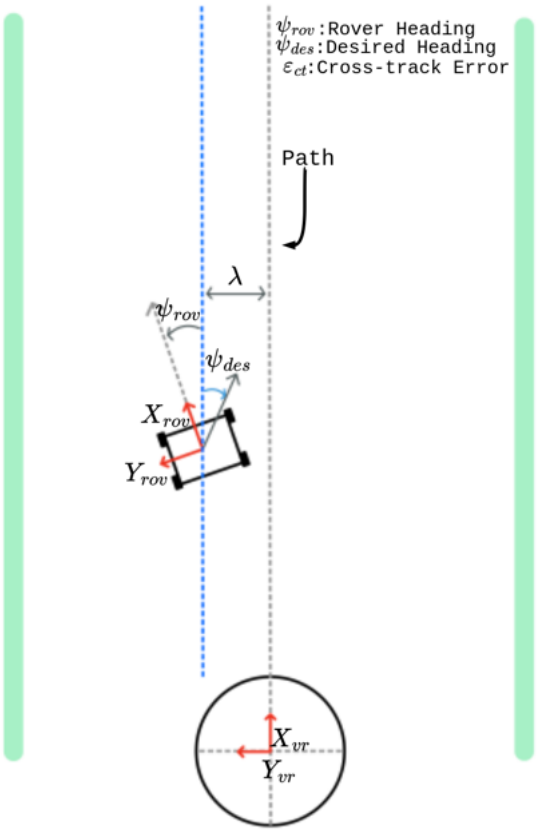

Figure 4.7 – Trunk Detection: Control Example Diagram

define our controller in equation 4.13.

$$\omega_z = k_p(\varepsilon) + k_i \int (\varepsilon) dt \tag{4.13}$$

Given our operating range of a maximum heading error of $\pm 45^{\circ}$ and a maximum cross-track error $\pm 0.5\ m$, the gains are selected such that the control input, ω_z, is within the range of $\pm 0.3\ rad/s$. This is done with the intent that the controller is typically not saturated during operation. Based on experimentation, we have chosen a proportional gain of $k_p = 0.229/s$, an integral gain of $k_i = 0.029/s^2$, and a $k_{ct} = 100\ deg/m$.

The steering control law in equation 4.13 results in being able to eliminate the cross-track error, while pointing the rover in the proper direction down the row.

4.2 Performance Verification System

To prepare our experimental environment, we positioned a 0.625 *cm* wide red strap approximately centered between the two vineyard rows. Although we positioned the strap to the best of our ability, because the position of the strap is approximate, the accuracy may vary by $\pm$ a few centimeters due to the unevenness of the terrain. This red strap is strictly used as an independent truth measure of the errors for evaluating the perception and control system. Based on averaging 10 measurements taken at about 10 *m* intervals, the row width was estimated to be 3.4 *m*.

By leveraging this setup, we can establish a ground truth for the cross-track error of our vineyard. This involves positioning the depth sensing camera at the center of the rover and measuring the distance between it and the red nylon strap. This distance serves as a reliable indicator of how accurately the rover follows the desired path between the vineyard rows.

Our performance verification consists of a NVIDIA Jetson Orin and a Zed stereo camera mounted precisely at the center of the rover. With the Zed stereo camera angled to focus on the ground directly beneath the center of the rover, as depicted in Figure 4.8, we capture depth details of the terrain as the rover moves. This configuration grants us access to highly accurate real-time depth information of both the red strap and the location directly under the rover. Retrieving the depth information, in *meters*, of the two areas on the field, allows us to map the lateral pixel offset to a measurement in *meters*. Furthermore, we assume the position of the red strap and the point directly beneath the center of the rover are in a perpendicular plane to the camera direction. Although this may not always be the case, we found the approximation to yield consistent results.

Our ROS 2-based verification system's software stack begins by detecting all red pixels within the image, creating a bit mask where red-hued pixels are white and all others are black. Using tools from the OpenCV library, the mask undergoes a Gaussian blur to reduce image noise, thereby enhancing accuracy and simplifying computations along the verification pipeline.

After frame preprocessing, the system pinpoints the white pixel nearest to the frame's center, determining the red strap's location. By then querying the depth of the rover's

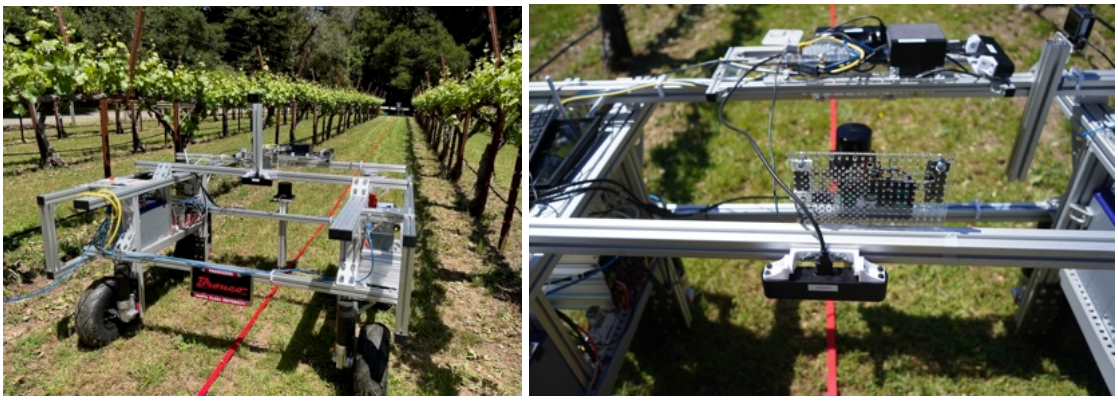

Figure 4.8 – Trunk Detection: Performance Verification System

central point and the detected white pixel, we calculate the distance between the two points in *meters*, establishing the rover's position relative to the strap. This information allows us to assess the rover's deviation from the vineyard row's center, providing us with a ground truth cross-track error based on the red strap's location. The reported cross-track error is published on the ROS 2 network and recorded in a rosbag during an experimental run.

4.2.1 Experimental Process

For our experimentation, we set up our system to operate autonomously within the vineyard row, while we closely monitored its performance in case of any anomalous behavior. The rover navigated along the length of a single row, which typically extended approximately 100 *meters*.

Throughout each run, we recorded data on the rover's reported cross-track error, heading error, and corresponding velocities, alongside the ground truth cross-track error. This comprehensive dataset allowed us to analyze the rover's performance in real-time and assess its alignment with the desired trajectory.

Following each run, we plotted the results to ensure the fidelity of our data. This visual and quantitative analysis provided valuable insights into the rover's behavior and performance within the vineyard environment.

In an attempt to optimize rover's navigation capabilities, we iteratively tuned the gains of the PI-Controller after each run.

4.3 Results

The key performance metrics of our system include mean cross-track error, root mean square error (RMSE), and standard deviation. These metrics allow us to quantify the accuracy and reliability of the navigation system by providing valuable insights into the system's ability to maintain the desired trajectory within the vineyard rows. We have computed the following metrics reported by both our navigation system and our performance verification system.

Our metrics for performance evaluation, measured in *meters*, are described below.

- **Mean Cross-track Error**: Mean cross-track error gives us the average measured lateral displacement of the rover with respect to the centerline. A nonzero mean error suggests a consistent bias in one direction.

$$\mu_{ct} = \frac{1}{N}\sum_{i=1}^{N} \varepsilon ct_i \mid N = \#\text{ of error points} \tag{4.14}$$

- **Root Mean Square Error (RMSE)**: RMSE provides an overall measure of accuracy by calculating the average size of errors, regardless of their direction. A lower RMSE indicates better accuracy and precision in following the desired path within the vineyard rows.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (\varepsilon ct_i)^2} \mid N = \#\text{ of error points} \tag{4.15}$$

- **Standard Deviation**: Standard deviation shows us how much the rover's position varies around the mean error. A higher standard deviation means more variability in performance, while a lower standard deviation indicates more consistency in maintaining the desired trajectory.

$$Std = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N} (\varepsilon ct_i - \mu_{ct})^2} \mid N = \#\text{ of error points} \tag{4.16}$$

4.3.1 System Performance

Table 4.1 displayed the performance metrics, mean cross-track error, RMSE, and standard deviation, as measured by both the navigation system and the performance verification system. Although, we conducted various experiments to allow us to tune and ensure consistent performance, the table presented below showcases the results from our most notable run.

Table 4.1 – Trunk Nav: System Performance Values at Steady State

Trunk Navigation: System Performance Values at Steady State			
	Mean Error (m)	**RMSE (m)**	**Standard Deviation (m)**
Sys Perceived	-0.0077	0.1203	0.1201
Sys Truth	-0.0179	0.0822	0.0803

The error and mean error of both the navigation system and the performance verification system are visualized in Figures 4.9 and 4.10, respectively. As indicated by the mean cross-track errors, the truth system reports the rover displays a slightly higher bias to the right side of the vineyard, than what is perceived by the navigation system. The performance verification system also reports a lower RMSE by 0.0381 m, indicating the navigation system slightly overestimates its overall cross-track accuracy. Lastly, the verification system reports lower variability in performance, with a measured standard deviation of 0.0803 m, compared to the navigation system's perceived standard deviation of 0.1201 m. After reviewing the performance metrics of the branch detection navigation system, we found the overall performance of our system to reactively navigate along the centerline of the vineyard surpasses current research standard. Furthermore, although there is an indication of bias towards one side, the measured amount is only within 0.01 m for both performance metrics.

Although the results of our novel navigation system are incredible when compared to the performance of other reactive navigation systems, there are some shortcomings that if addressed, may improve the performance of our navigation system. One example is an improvement on the data cleaning system prior to assigning clusters. Because the points within these clusters are often noisy, reducing the noise may improve the accuracy of the fitted lines. This enables a better estimation of the computed lateral distances from the rover to each of the sides of the vineyard row. Also additional tuning of the PI-controller

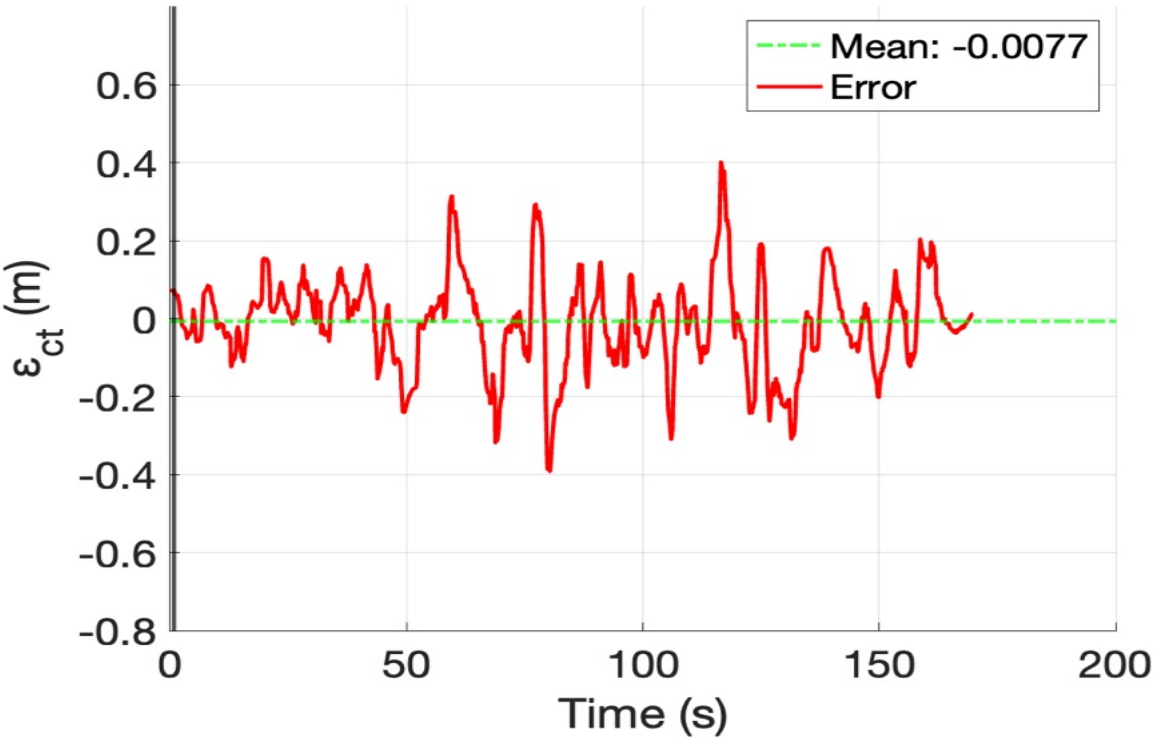

Figure 4.9 – Trunk Detection: System Performance

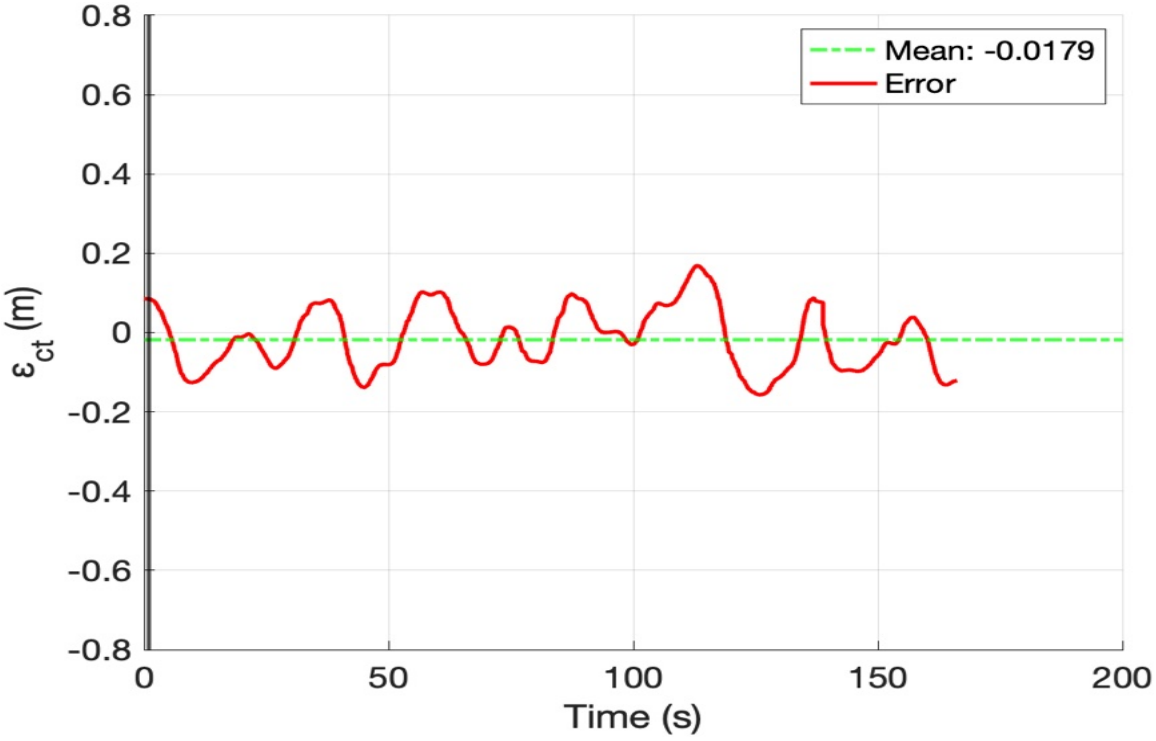

Figure 4.10 – Trunk Detection: Ground Truth Performance

may possibly result in better performance metrics.

4.4 Summary

In chapter 4, we introduce a novel method of reactive in-row navigation that leverages the vineyard branches to steer the rover along the centerline. We also provide insight to our experimentation process for validating the navigation algorithm. In the final section, section 4.3.1, we present the results from our most notable run. The results presented in the section greatly surpass the performance currently presented by research standard. We also describe some areas of modification that may improve our navigation performance.

Chapter 5

Comparison and Conclusion

Section 5.1 first provides a comparative analysis on the performance of the two novel navigation systems presented in this thesis. We then compare the remarkable performance of the trunk-detection based system against the current industry standard performance for in-row vineyard navigation. Section 5.2 summarizes the thesis material and presents possible paths for future work.

5.1 Comparative Analysis

Table 5.1 provides an comparison of the reported truth data calculations for the two algorithms. It vividly illustrates that the trunk detection method outperforms the branch detection algorithm in a quantitative manner. Although it has yet to be tested, we believe the trunk detection method to be season-invariant, which is another advantage over the branch detection method. Moreover, as the trunk method harnesses a clustering algorithm to detect the vineyard rows, it offers greater flexibility for the mechanical positioning of the LiDAR.

Table 5.1 – Comparative Analysis: Branch vs. Trunk

Branch Detection VS Trunk Detection		
	Branch	**Trunk**
Mean Err	-0.1731	-0.0179
RMSE	0.1877	0.0801
Std Dev	0.0728	0.0780

Remarkably, not only does the trunk method surpass the branch detection method in navigation, but it also exceeds much of the current industry standard in terms of accuracy performance. As seen in table 5.2, the trunk detection method yields a significantly lower RMSE than that of Bertoglio et al.[12] and Mengoli et al.[14]. Furthermore, while the algorithm by Saramento et al. [15] achieved a significantly lower RMSE, their experimentation is limited to simulation.

Table 5.2 – Comparative Analysis: Trunk compared to other algorithms.

Performance Against Other Algorithms				
Algorithms	Trunk Based	Bertoglio et al.[12]	Mengoli et al.[14]	Saramento et al.[15]
RMSE(m)	0.080	0.372	0.342	0.011

5.2 Conclusion

The purpose of this research is to enable vineyard navigation in environments where nearby woods or foliage causes GPS-signal degradation. To accomplish this, we proposed two novel methods of perception based vineyard navigation, constructed a system to validate the performance of the two navigation methods, and conducted experiments by deploying the navigation systems at Kings Mountain Vineyard. The comparative analysis shows that outstanding performance was achieved given that the system yielded a significantly lower RMSE than many other perception-based navigation techniques.

In conclusion, the findings presented in this thesis underscore the promising potential of both algorithms while also shedding light on the tradeoffs associated with each. This research contributes to the ongoing efforts aimed at advancing precision agriculture practices and underscores the significance of algorithmic innovation in addressing the evolving challenges of modern agriculture.

Additional tuning of the controller could facilitate better performance for each of the methods, however, the obtained results are beyond sufficient compared to current industry standards. As observed during testing, the trunk based algorithm is typically susceptible to environmental noise, such as unmanaged weeds. A possible solution to this would be improving the method of outlier removal to place greater emphasis on the vineyard trunk

rows themselves. Other future work should encompass the two areas of either out-of-row navigation and turning or applicable functionality that leverages the in-row navigation. The latter topic includes but is not limited to gathering viticulture telemetry, picking and collecting grapes, and multi-robot in-row collaboration.

As previously mentioned, the trunk-detection method of navigation season-invariant nature has yet to be tested. Because this attribute is particularly significant in dynamic agricultural environments where conditions may vary considerably over time, additional experimentation demonstrating this capability is encouraged.

List of References

[1] U.S. Department of Agriculture, "Farm Labor." [Online]. Available: https://www.ers.usda.gov/topics/farm-economy/farm-labor/

[2] Population Reference Bureau, "United Nations Raises Projected World Population," 06 2013. [Online]. Available: https://www.prb.org/resources/united-nations-raises-projected-world-population/

[3] Estelle Midler, "Environmental degradation: impacts on agricultural production," *IEEP*, 04 2022.

[4] A. Papadimitriou, I. Kleitsiotis, I. Kostavelis, I. Mariolis, D. Giakoumis, S. Likothanassis, and D. Tzovaras, "Loop closure detection and slam in vineyards with deep semantic cues," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 2251–2258.

[5] D. Tiozzo Fasiolo, L. Scalera, E. Maset, and A. Gasparetto, "Towards autonomous mapping in agriculture: A review of supportive technologies for ground robotics," *Robotics and Autonomous Systems*, vol. 169, p. 104514, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889023001537

[6] N. Abdelaziz and A. El-Rabbany, "Lidar/visual slam-aided vehicular inertial navigation system for gnss-denied environments," in *2022 5th International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*, 2022, pp. 1–5.

[7] H. Gim, M. Jeong, and S. Han, "Autonomous navigation system with obstacle avoidance using 2.5d map generated by point cloud," in *2021 21st International Conference on Control, Automation and Systems (ICCAS)*, 2021, pp. 749–752.

[8] Z. Li, X. Liu, H. Wang, J. Song, F. Xie, and K. Wang, "Research on robot path planning based on point cloud map in orchard environment," *IEEE Access*, vol. 12, pp. 54 853–54 865, 2024.

[9] L. Milburn, J. Gamba, M. Fernandes, and C. Semini, "Computer-vision based real time waypoint generation for autonomous vineyard navigation with quadruped robots," in *2023 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2023, pp. 239–244.

[10] D. Aghi, S. Cerrato, V. Mazzia, and M. Chiaberge, "Deep semantic segmentation at the edge for autonomous navigation in vineyard rows," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 3421–3428.

[11] D. Aghi, V. Mazzia, and M. Chiaberge, "Local motion planner for autonomous navigation in vineyards with a rgb-d camera-based algorithm and deep learning synergy," *Machines*, vol. 8, no. 2, p. 27, May 2020. [Online]. Available: http://dx.doi.org/10.3390/machines8020027

[12] R. Bertoglio, V. Carini, S. Arrigoni, and M. Matteucci, "A map-free lidar-based system for autonomous navigation in vineyards," in *2023 European Conference on Mobile Robots (ECMR)*, 2023, pp. 1–6.

[13] F. Rovira-Más, V. Saiz-Rubio, and A. Cuenca-Cuenca, "Augmented perception for agricultural robots navigation," *IEEE Sensors Journal*, vol. 21, no. 10, pp. 11 712–11 727, 2021.

[14] D. Mengoli, A. Eusebi, S. Rossi, R. Tazzari, and L. Marconi, "Robust autonomous row-change maneuvers for agricultural robotic platform," in *2021 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*, 2021, pp. 390–395.

[15] J. Sarmento, A. S. Aguiar, F. N. d. Santos, and A. J. Sousa, "Robot navigation in vineyards based on the visual vanish point concept," in *2021 International Symposium of Asian Control Association on Intelligent Robotics and Industrial Automation (IRIA)*, 2021, pp. 406–413.

[16] D. Mengoli, R. Tazzari, and L. Marconi, "Autonomous robotic platform for precision orchard management: Architecture and software perspective," in *2020 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*, 2020, pp. 303–308.

[17] Sharma, Manoj and Christopher A. Kitts, "Modular and Reconfigurable Multiple Drive-Unit Based Rover - Design and Control." *ASME International Mechanical Engineering Congress and Exposition*, vol. 87592. American Society of Mechanical Engineers, 2023.

[18] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074

[19] C. Kitts, P. Mahacek, T. Adamek, K. Rasal, V. Howard, S. Li, A. Badaoui, W. Kirkwood, G. Wheat, and S. Hulme, "Field operation of a robotic small waterplane area twin hull boat for shallow-water bathymetric characterization," *Journal of Field Robotics*, vol. 29, 2012.