B. TECH. PROJECT REPORT

On

Design of a 4-Wheel Steering and Driving Mobile Robotic Platform

by

Kshitij M Bhat

DEPARTMENT OF MECHANICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE

November 2023

Design of a 4-Wheel Steering and Driving Mobile Robotic Platform

A PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degrees

of
BACHELOR OF TECHNOLOGY
in
MECHANICAL ENGINEERING

Submitted by
Kshitij M Bhat

Guided by
Dr. Devendra Deshmukh, Department of Mechanical Engineering, Indian Institute of Technology Indore

DEPARTMENT OF MECHANICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE

November 2023

CANDIDATE'S DECLARATION

I hereby declare that the project entitled **"Design of a 4-Wheel Steering and Driving Mobile Robotic Platform"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in Mechanical Engineering completed under the supervision of **Dr. Devendra Deshmukh**, IIT Indore is an authentic work. Further, I declare that I have not submitted this work for the award of any other degree elsewhere.

Kshitij M Bhat 28/11/2023

.....

Kshitij M Bhat
Roll No.: 200003042

CERTIFICATE by BTP Guide

It is certified that the above statement made by the student is correct to the best of my knowledge

.....

Dr. Devendra Deshmukh
Professor, Department of Mechanical Engineering
Indian Institute of Technology Indore

Preface

This report on *Design of a 4-Wheel Steering and Driving Mobile Robotic Platform* is prepared under the guidance of Dr. Devendra Deshmukh. This reports presents the development, modeling, and evaluation of a four-wheel steering driving mobile robot with a comprehensive study involving initial CAD design, mathematical modeling, autonomous navigation, and electronics design. The methodology of robot's design is detailed, emphasizing efficiency and ease of assembly. Moreover, the report delves into the implementation of the controller using ROS (`ros_control`). Modeling of the robot's odometry facilitates the autonomous navigation capabilities. Additionally, localization algorithms have been implemented to assess their efficacy within Gazebo simulation scenarios. The results obtained from experiments and simulations are analyzed and presented graphically, underlining a thorough investigation into the robot's functionality and performance.

This work aims to contribute to the field of agricultural robotics by offering a study of design methodologies, control systems, simulation techniques, and algorithmic implementations.

Kshitij M Bhat
Roll No.: 200003042
B.Tech. IV Year
Department of Mechanical Engineering
Indian Institute of Technology Indore

Acknowledgements

I extend my deepest gratitude to **Dr. Thiago E. Alves de Oliveira**, Assistant Professor in the Department of Computer Science at Lakehead University - Thunder Bay, Canada, for his invaluable guidance and mentorship during my summer internship on this project. I am immensely grateful to MITACS for providing me with the opportunity to undertake this project through the MITACS Globalink Research Internship program, with support from the Government of Ontario and Canada. I express my heartfelt appreciation to Mr. Toby Godfrey for his assistance with the PCB design and electronics. I am indebted to my supervisor, **Dr. Devendra Deshmukh**, for his thoughtful and efficient guidance throughout this project. Lastly, I extend my sincere thanks to the Department of Mechanical Engineering at IIT Indore for allowing me to pursue and expand upon the research initiated during the summer internship.

Kshitij M Bhat
Roll No.: 200003042
B.Tech. IV Year
Department of Mechanical Engineering
Indian Institute of Technology Indore

Abstract

Scaling agricultural inputs is not a viable solution due to resource constraints in doubling production for a growing global population. Robotic agricultural systems offer relief from labor-intensive tasks and improve efficiency, leading to cost reduction and minimized environmental impact through precise operations. This work contributes to the area of agricultural robotics with the design of a mobile robotic platform for agricultural purposes such as crop analytics and weed detection. The unique four-wheel steering and driving configuration of the robotic platform offers advantages over other steering configurations, such as skid-steer and Ackermann steering. The software design of the robotic platform is based on the widely used `ros_control` paradigm, aiding seamless interaction between the hardware and the software. We also provide the simulation of the robotic platform using Gazebo to test sensors such as GPS and IMU. Furthermore, we implement the EKF algorithm to quantify the efficacy of the odometry model in simulation.

Contents

List of Figures

Introduction

1.1 Robotics in Agriculture

Throughout history, advancements in mechanization and automation revolutionized agriculture, exponentially boosting crop production and enhancing global quality of life. However, the rapid growth in population and income, especially in developing nations, demands a doubling of agricultural output by 2050. Merely increasing traditional inputs like land, water, and labor is not feasible due to limited resources and environmental concerns. To meet this challenge sustainably, the efficiency of the agricultural system needs to grow significantly.

Doubling the agricultural output without overtaxing resources, the total factor productivity (TFP) of global agriculture must elevate from its current ratio of 1:4 to 1:75. This necessitates substantial advancements in various aspects that impact the TFP: seeds, soil management, water usage, fertilizers, pest control, crop structure, automation, labor, cultural practices, and public policies. Automation emerges as a important factor among these elements, possessing the potential to influence agriculture comprehensively and significantly contribute to meeting future demands. The imperative to double agricultural production to accommodate a global population of nine billion is hindered by resource constraints, rendering a simple scaling of inputs ineffective. Consequently, the emphasis shifts toward enhancing agricultural system efficiency sustainably. In response, academic and commercial researchers are directing their attention to advancing sensing, mobility, and manipulation technologies as a promising avenue to boost agricultural output and productivity (Bergerman et al. [6]).

The integration of sensing technology in agriculture involves measuring diverse physical attributes like crop temperature, humidity, and imagery, enabling accurate decision-making

based on analyzed data. For instance, camera-based systems capture orchard images pre-harvest, providing precise crop yield estimates to facilitate harvest planning—an ability beyond human sensing capabilities due to inherent inaccuracies or slowness.

Mobility in agriculture refers to vehicle automation facilitating driverless or driver-assisted field coverage, exemplified by GPS-guided combines that optimize coverage and fuel consumption during crop harvesting. Advancements in auto-guidance are extending to orchard vehicles, requiring additional navigation sensors due to poor satellite reception under dense canopies. Autonomous vehicles equipped with tools enable semi-autonomous operations like seeding, weed removal, pesticide dispersal, and harvesting, relying on sensor-based perception from GPS/GNSS, cameras, radars, and more.

Robotic mobility technology presently lags behind sensing advancements. Manipulation involves direct crop operations such as pruning, harvesting, and weed removal. This technology demands more sophisticated sensor-based perception than mobility but is currently less developed in field deployment compared to sensing or mobility (Bergerman et al. [6]).

Robotic agricultural systems not only alleviate labor-intensive tasks but also enhance the efficiency of agricultural production by engaging in precise operations, resulting in decreased costs and minimized environmental impacts. Over the recent years, numerous mobile robots have been designed for diverse applications, including weed control (Slaughter et al. [20]), high-precision seeding (Katupitiya et al. [13]), fruit harvesting (Bac et al. [3]), crop yield estimation (Nuske et al. [18], Xiong et al. [23]) and phenotyping (Atefi et al. [2]).

1.2 Literature Review

Mobile robotics plays a pivotal role in precision agriculture by automating tasks like harvesting, weeding, and planting, enhancing production while minimizing resource wastage. They are significant optimizing yields while conserving water, nutrients, and pesticides. Sensor-equipped mobile robots are increasingly employed for mapping and monitoring agricultural landscapes, offering high-precision detection of spatial variations in terrains and crops (Tiozzo Fasiolo et al. [22]).

Mobile robot platforms, also known as unmanned ground vehicles (UGVs), are engineered with different steering configurations to better navigate challenging agricultural fields. Most four-wheeled robots employ various types of steering configurations such as tracked skid-steered, wheeled skid-steered, four-wheel steering, and Ackermann steering. Tracked skid-steered vehicles are primarily used to prevent sinking in soft ground and to enhance traction by reducing sinkage compared to wheeled vehicles. However, they aren't used in crop fields to avoid compacting plants. Both tracked and wheeled skid-steered vehicles employ differential steering.

The widely adopted Ackermann steering geometry eliminates the need for wheels to slide sideways during turns, making it a prevalent choice. Skid-steered and four-wheel steering solutions offer advantages in small or cluttered environments. Consequently, these kinematic solutions, particularly skid-steered and four-wheel steering, find frequent application in navigating under-canopy areas and greenhouses due to their adaptability in such spaces.

1.2.1 Ackermann Steering

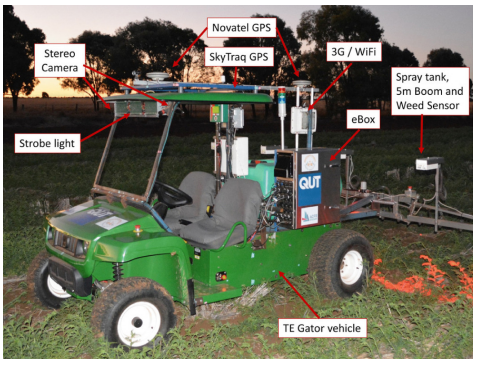
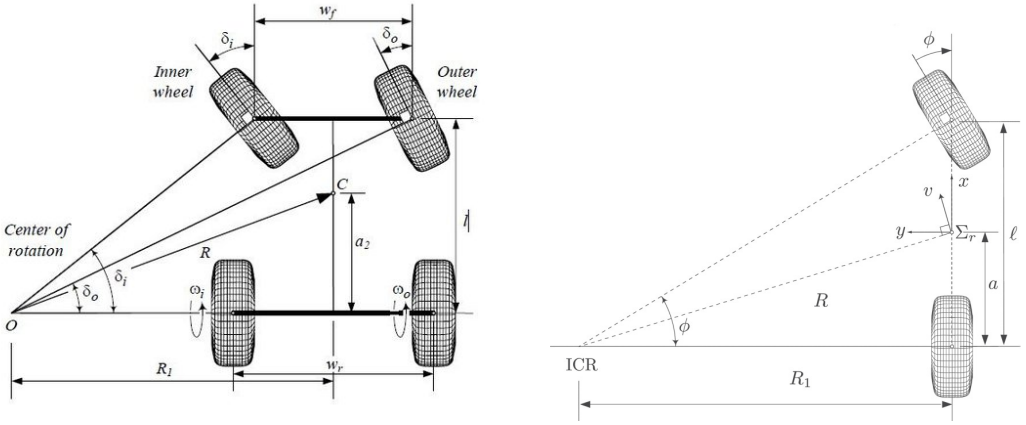


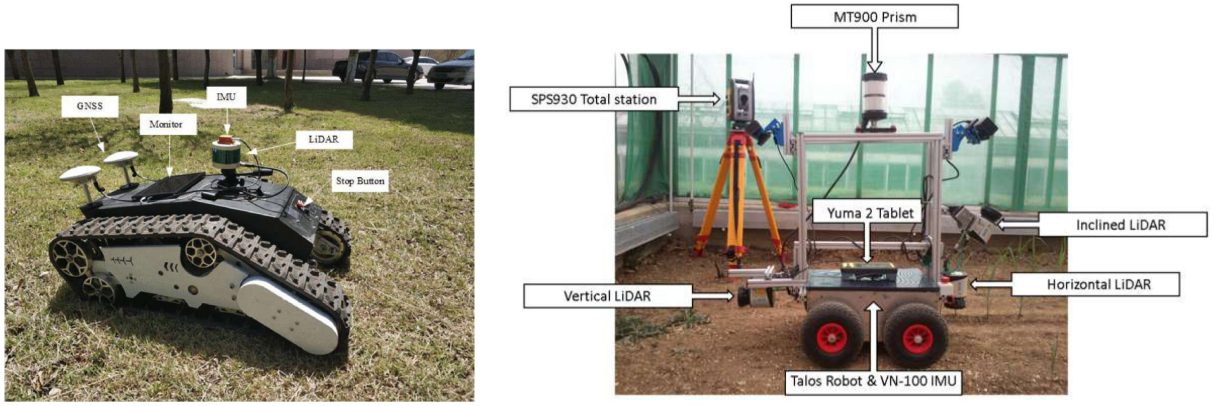
FIGURE 1.1: Ackermann Steering Agricultural Robot (Ball et al. [5])

One of the most widely used types of steering configurations for four-wheeled vehicles is the Ackermann steering. Only two wheels of the vehicle rotate in order to steer the vehicle. Multiple mechanisms, such as four-bar linkages are widely used to turn the front wheels of the vehicle. The mathematical model of Ackermann Steering conveniently expressed as a Bicycle model (Carpio et al. [8]).



Ackermann geometry Bicycle model

FIGURE 1.2: Overview of Ackermann Steering (Carpio et al. [8])



Track skid steering Wheel skid steering

FIGURE 1.3: Skid steering agricultural robots (Gao et al. [11],Garrido et al. [12])

1.2.2 Skid Steering

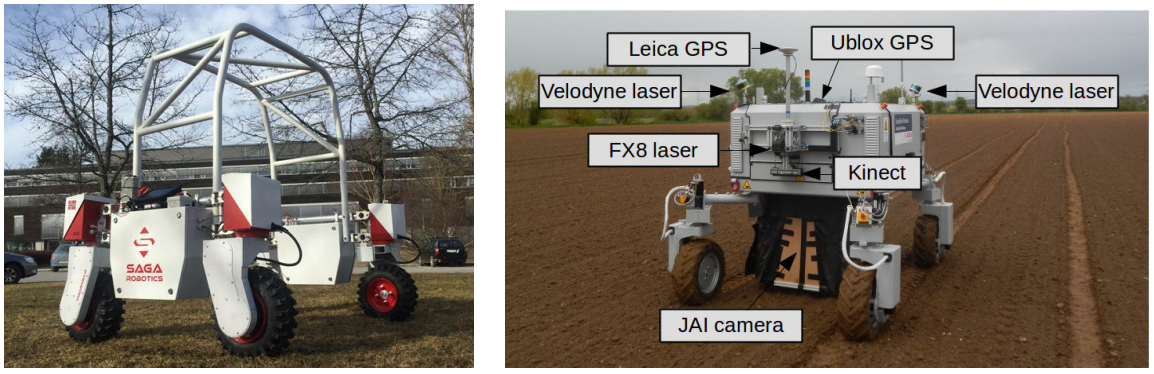
Skid-steering with both wheeled robots as well as tracked robots have been widely used for many off-road conditions in agriculture, mining,and military. The steering system involves controlling the speeds of the two side drives (wheels or tracks), much like differential drive wheeled vehicles. However, as all the wheels align with the vehicle's longitudinal axis, turning necessitates wheel slippage. This steering system resembles that of a tracked vehicle, such as a tank. While tracked locomotion generally offers superior traction, it also comes with increased mechanical complexity (Mandow et al. [15]).

1.2.3 Overview of Four-wheel Steering Platforms in Agriculture

One of the earliest work on a four-wheel steering system for agriculture is (Bak and Jakobsen [4]) They introduced a robot for surveying weed populations in agricultural areas, showcasing autonomous vehicle concepts in agriculture. This vehicle operates in 0.25 and 0.5m row crops, utilizing cameras for weed detection and visual row-guided navigation. Using four identical wheel modules, the platform achieves four-wheel steering, significantly enhancing mobility. This design allows the vehicle to move parallelly during turns by separating

position adjustments from orientation adjustments. Control is managed via an embedded system and common communication protocols. The software employs a hybrid deliberate architecture, enabling a hierarchical breakdown of operations. At the lowest level, a reactive feedback control mechanism, derived from simple control methods for car-like vehicles, is extended to the four-wheel scenario. This controller ensures the vehicle's front and rear adhere to a predetermined path, maintaining a fixed orientation relative to that path.

Another early work (Cariou et al. [7]) introduce a a four-wheel-steering (4WS) mobile robot capable of adjusting both lateral and angular deviations from a desired trajectory. They devise a path tracking control system utilizing an extended kinematic model, which considers real-time wheel skidding. Additionally, a backstepping approach is employed to manage the 4WS configuration effectively. This approach leverages both rear and front steering actions to effectively counteract sliding effects during path tracking. Furthermore, they develop a predictive algorithm to address steering actuator delays, compensating for temporary overshooting in curves.



Thorvald Bonirob (Chebrolu et al. [9])

FIGURE 1.4: Commercial 4-WSD Robots

Some commercial robots for agriculture employ the 4WS configuration . Examples include Bonirob and Saga Robotics Thorvald. Bonirob, developed by Bosch, Amazone, and the Osnabrück University of Applied Sciences, integrates a comprehensive array of sensors and effectively generates an extensive agricultural dataset. Bonirob's agility is notably attributed

to its four-wheel steering kinematics. It leverages visual servoing and a penetrometer to identify soil compaction, enhancing its functionality. On the other hand, Thorvald is a customizable robot suited for operations within a particular environment, such as a greenhouse, or a vineyard.

1.3 Motivation

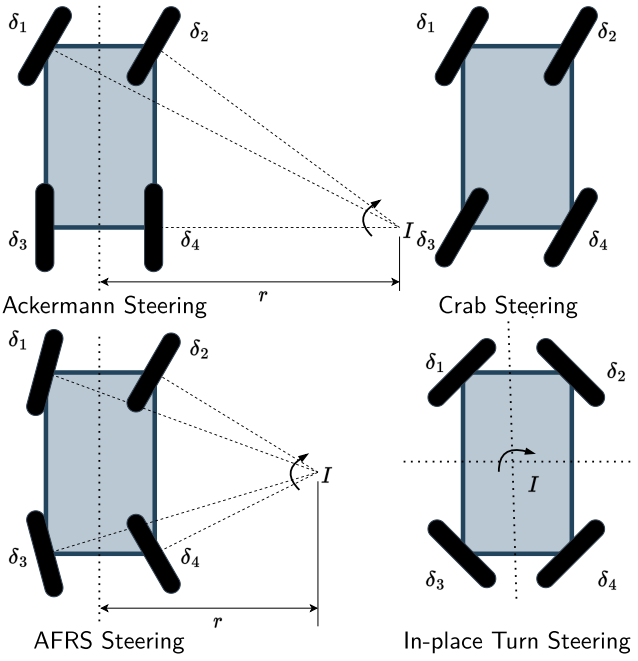


FIGURE 1.5: Steering modes of the 4-WSD robot

Wheeled skid-steering offers two primary advantages compared to other wheel configurations like Ackerman or axle-articulated systems. Firstly, it boasts mechanical simplicity and robustness. Secondly, it enables enhanced maneuverability, including zero-radius turns, utilizing only the components necessary for linear motion.

However, this locomotion method presents specific challenges concerning motion control and odometry. Skid-steering kinematics prove complex since predicting the precise motion of the vehicle solely from its control inputs is not feasible. Consequently, the assumptions of

pure rolling and no-slip conditions, typical in kinematic models for non-holonomic wheeled vehicles, do not hold in this steering configuration (Mandow et al. [15]).

Four-wheel steering configuration provides a solution to these problems incorporating the features of skid-steering such as zero-radius turn and improved maneuverability, while retaining the pure-rolling and no-slip assumptions. At the same time, it provides the capability to precisely predict motion of the vehicles from factors such as control inputs and wheel encoder readings.

A 4-WSD system facilitates independent turning of each wheel, significantly enhancing maneuverability to achieve omnidirectional movement through coordinated wheel rotations. This capability grants a robot the flexibility to pivot at any Instantaneous Center of Rotation (ICR), enabling seamless navigation within confined crop environments. The steering control system's pivotal role lies in ensuring precise path tracking and maneuvering performance, influenced by variables such as driving speed, steering modes, control gains, and the path planning algorithm.

This setup offers four distinct steering modes: Ackermann steering, active front and rear steering (AFRS), crab steering, and In-place turn steering, as depicted in Figure 1.5. Implementation of Ackermann and AFRS steering prompts changes in the robotic platform's orientation and driving direction while maintaining movement along longitudinal directions. Spinning alters the robot's orientation without adjusting its position, whereas crab steering achieves the opposite effect. Consequently, in-place turning and crab steering prove effective in scenarios requiring either orientation or position correction.

Despite substantial efforts from the scientific community to develop agricultural four-wheel steering platforms, there is a notable absence of an open-source system that combines flexibility and affordability. While some existing open-source, low-cost platforms are available, they are often constrained in functionality and features, lacking commercial availability or

adaptability to specific crop fields. Many commercially accessible agricultural robots are expensive, closed-source, and platform-specific, incorporating rigid hardware and firmware that pose challenges for modification. This limitation hampers users from exploring diverse customization or reconfiguration options, hindering the rapid development of intelligent systems. To satisfy this need, we present a four-wheel steering and four-wheel driving mobile robotic platform for diverse agricultural applications, including crop yield estimation and weed detection. Our robot is a low cost, modular and reconfigurable robot, designed to be easily modified for several types of crops. The software is compatible with ROS, enabling a great level of customization. The four-wheel steering configuration enables the vehicle to undergo parallel displacement during turns by separating adjustments in position from adjustments in orientation. This provides exceptional maneuverability, offering an advantage not only in precision operations within the crop field but also the capability for highly compact turns

Robot Design

2.1 Mechanical Design

FIGURE 2.1: CAD model of the robot

2.1.1 Motor Selection

We use a simple analysis to select the best motors for the vehicle prototype. It is assumed that the same motor will be used for both steering and driving modules.

Assumptions

$m = 20$kg (payload + vehicle mass)

$\theta = 20\,\mathrm{deg}$ (inclination)

$r_w = 0.1$m (robot wheel radius)

$$F - mg\sin\theta = ma$$

$$F_{\text{min}} = mg \sin\theta \text{ (for zero acceleration)}$$

$$\Rightarrow F_{\text{min}} = 67.04N$$

Further, this force must be supplied by four motors with unknown torque (τ). Thus we have $4\tau_{\text{min}} = Fr_{\text{w}}$, giving us $\tau_{\text{min}} = 1.675Nm$.

We choose motors with torques higher than 1.675 Nm, by making a price vs. torque analysis. Among the motors available, the JGY-370 series offers the best torque for our application.

FIGURE 2.2: JGY-370 motor

2.1.2 Material Selection

Different materials were selected for chassis and parts of the robot. The chassis comprises of pipes fitted with Tee-joints. The rest of the parts are made of 3-D printed plastic.

- **ABS:** The main chassis frame is composed of Acrylonitrile Butadiene Styrene (ABS) pipes. ABS offers lower density than PVC, thus reducing the weight of the robot. Moreover, it offers better strength and shock-resistance.

- **PETG:** The steering and driving modules are 3D printed with Polyethylene Terephthalate Glycol (PETG). It is a widely used filament material for 3D printing. It offers higher strength and resilience than Polylactic Acid (PLA), which is also a widely used material for 3D printing.

2.1.3 Assembly

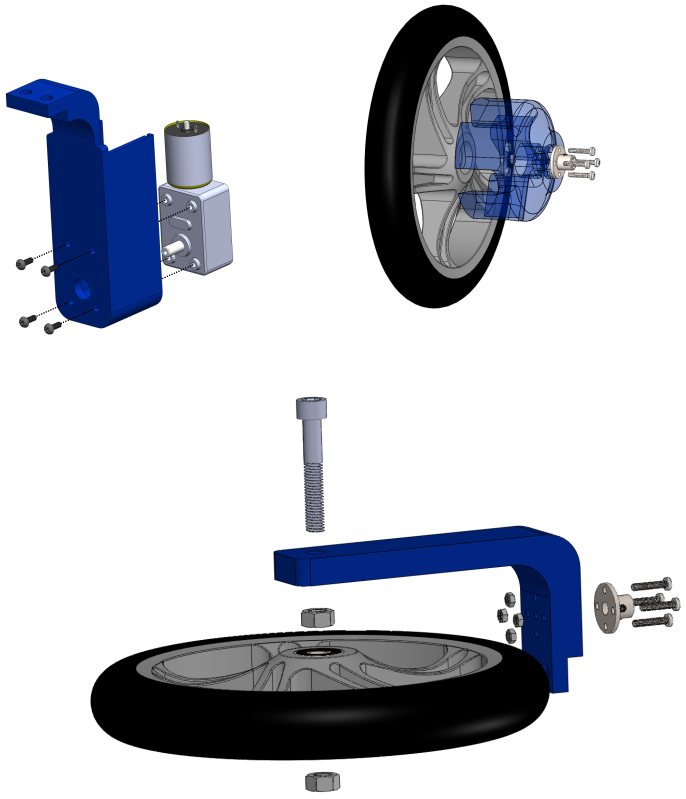


FIGURE 2.3: Assembly of the driving module

The robot is designed with parts which are designed with CAD software in order to be 3D printed. Design for Assembly (DFA) principles were used while designing the parts for the Steering-Driving modules.

The goal of Design for Assembly (DFA) involves streamlining product complexity to minimize assembly costs. As a result, integrating DFA principles into product design typically leads to enhanced quality and reliability, along with reductions in inventory and equipment costs. It's commonly observed that these additional advantages are more important than the initial cost reductions during the assembly process.

DFA fundamentally involves redesigning the parts to aid the assembly process and making improvements to the overall product design during the initial design process to drive cost efficiencies over the entire product lifecycle. It embodies a process aimed at enhancing product design to facilitate easy and cost-effective assembly, achieved by simultaneously prioritizing functionality and assembly simplicity (Mital et al. [16]).

2.2 Overview of Electronics

The main computing unit is the Raspberry Pi (Single Board Computer), which runs a distribution of the Linux operating system, Ubuntu 20.04. Four Raspberry Pico micro-controllers are connected with the Raspberry Pi via USB cables. The communication is established via serial communication. It uses one or two transmission lines to send and receive data and facilitates continuous data transmission, one bit at a time. Each micro-controller runs the one Steering-Driving module, with two PID loops controlling controlling the position and velocity of the steering motor and driving motor respectively.

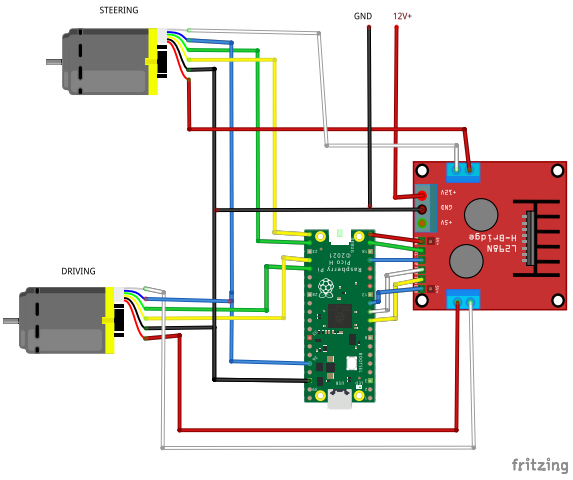


FIGURE 2.4: Wiring diagram for the steering-driving module

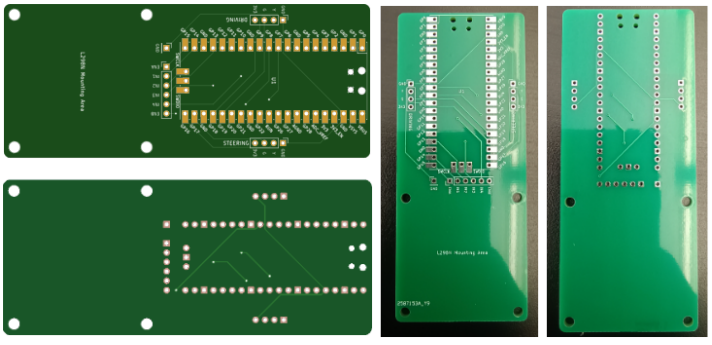


FIGURE 2.5: PCB for the Steering-Driving Module

2.2.1 PID Control

The fundamental concept behind a PID controller involves reading a sensor's input, computing the desired actuator output by evaluating proportional, integral, and derivative responses, and summing these components to generate the output. Before delving into PID controller parameters, it's crucial to understand the notion of a closed-loop system and its associated terminologies. In a standard control system, the process variable represents the parameter under control, such as the angular position of the steering fork or the wheel's angular velocity. The sensor associated with the motor, typically an encoder, provides feedback to the controller. The set point, defined as the desired or commanded value for the process variable, is established by the `four_wheel_steering_controller` ROS node via the `hardware_interface`. The PID algorithm utilizes the discrepancy between the process variable and the set point to compute the desired actuator output for motor operation. Tuning the PID controller involves optimizing the P, I, and D gains to achieve an ideal response in the control system, a process often conducted through trial-and-error.

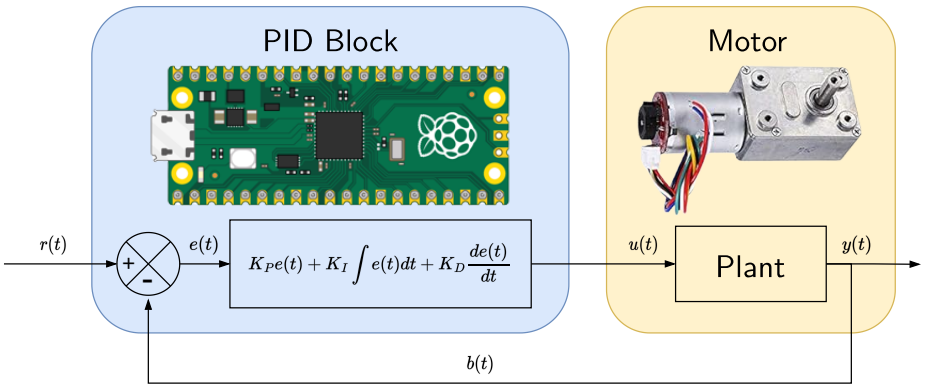


FIGURE 2.6: PID Controller block diagram

2.3 Software Design

2.3.1 ROS

The Robot Operating System, introduced by Quigley et al. [19], functions as a software framework that bridges the gap between robot hardware and the necessary software files for robot control. Over time, it has evolved into the *de facto* standard for developing robotics software, serving as a structured communication layer atop the host computer. It offers a range of utilities for monitoring processes, examining communications, handling time-series transformations, and more. Additionally, ROS has an extensive ecosystem comprising sensor, control, and algorithmic packages contributed by the community. This breadth of resources enables even small teams to construct intricate robotics applications.

2.3.2 `ros_control`

The `ros_control` framework, proposed by Chitta et al. [10], facilitates the implementation and supervision of robot controllers, emphasizing real-time performance and the ability to share controllers across different robots seamlessly. The main rationale behind introducing a distinct robot-control framework is the absence of a real-time secure communication layer within ROS. Additionally, this framework addresses controller-lifecycle and hardware resource management issues, providing abstractions for hardware interfaces while minimizing dependencies on specific hardware or operating systems. Its evident modular and lucid design renders `ros_control` well-suited for both research and industrial applications, leading to its widespread adoption in various domains.

The core of the framework relies on the Hardware Abstraction Layer, facilitated by the `hardware_interface::RobotHW` class. This abstraction acts as a link to various simulated and real robots, requiring specific robot implementations to inherit from this class. These instances model the hardware resources offered by the robot, encompassing actuators, sensors like encoders, force/torque sensors, and more. This abstraction seamlessly integrates diverse hardware components, enabling smooth transitions between real and simulated robots.

This framework supports the assembly of `RobotHW` instances, beneficial for constructing control systems involving parts from different suppliers, each providing their specific RobotHW instance. Additionally, the `hardware_interface` package defines interfaces for joint and actuator functionalities (e.g., state, position, velocity, and effort), abstracting away hardware intricacies. These typed interfaces enhance introspection, maintainability, and ensure controllers are adaptable across various hardware setups.

The `controller_manager` oversees controller and hardware resource lifecycles while managing conflicts between controllers. Controller lifecycles aren't static; they can be modified at runtime through standard ROS services. These services enable starting, stopping, and configuring controllers dynamically, offering flexibility in managing robot control systems.

The Universal Robot Description Format (URDF) file directly supports a declarative definition of transmissions alongside the robot's kinematics and dynamics description. Within the `joint_limits_interface` package, there exist data structures designed to represent joint limits, methods to fill these limits through URDF or yaml files, and functionalities to enforce these set limits. Additionally, the `control_toolbox` provides various components beneficial for controller development, including a PID controller class, smoothers, sine-wave generators, and noise generators.

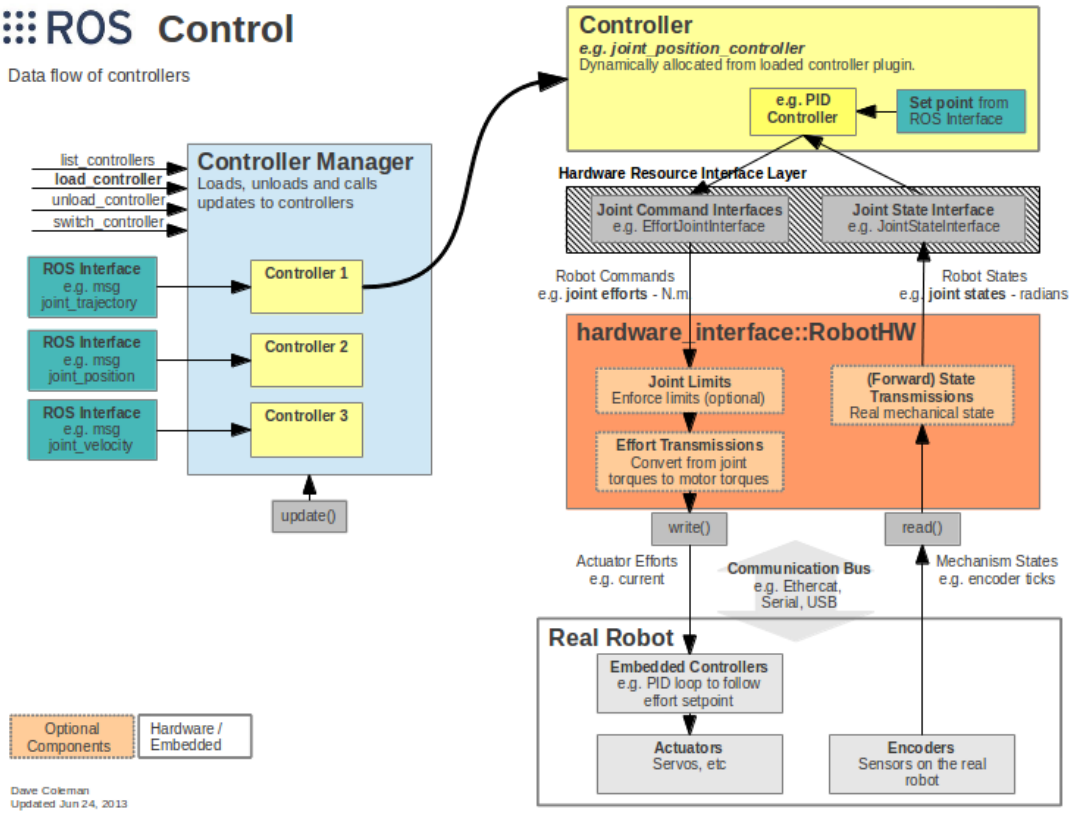


FIGURE 2.7: Overview of `ros_control` (Chitta et al. [10])

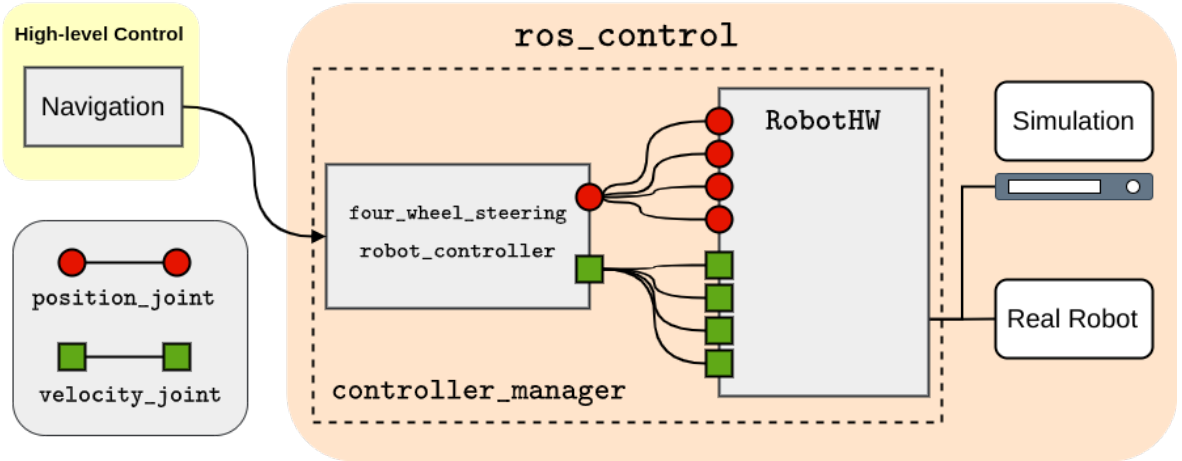


FIGURE 2.8: Overview of `four_wheel_steering_controller`

Vehicle Modelling and Simulation

3.1 Introduction

The robot illustrated in 3.1 is equipped with four independent modules, each with two degress of freedom (1DOF from steering motor and 1DOF from the driving motor). This configuration enables the robot to achieve highly maneuverable motion on the terrain, necessitating robust control over the speed and orientation of each wheel. Maintaining the alignment of each wheel's direction with the instantaneous wheel direction maximizes efficiency and minimizes lateral friction forces.

Due to the robot's overactuation, there exist numerous actuation solutions for a specified motion vector, leading to an actuation vector with dimensions exceeding those required for the motion. Actuation is executed by supplying voltage to the four drive motors while setting the desired angle for the steering motors. A closed-loop angle controller (such as servo control) governs the steering motor angle, ensuring accurate control over the robot's directional adjustments.

Symbols

Robot pose: $\boldsymbol{q} = [x\ y\ \psi\ ]^T$

Wheels angular velocities: $\boldsymbol{\omega} = [\omega_1\ \omega_2\ \omega_3\ \omega_4]^T$

Steering angles: $\boldsymbol{\delta} = [\delta_1, \delta_2, \delta_3, \delta_3]^T$

Robot velocity vector: $\boldsymbol{v} = \left[v_x\ v_y\ \dot{\phi}\right]^T$, where v_x, v_y and $\dot{\phi}$ are the longitudinal, lateral, and angular velocities.

Longitudinal forces on tires: $\boldsymbol{F}_x$, Lateral forces on tires: $\boldsymbol{F}_x$, and Torque applied to the wheels: $\boldsymbol{\tau}$

Longitudinal slip ratios: $\boldsymbol{\lambda}$ and Sideslip angles: $\boldsymbol{\alpha}$

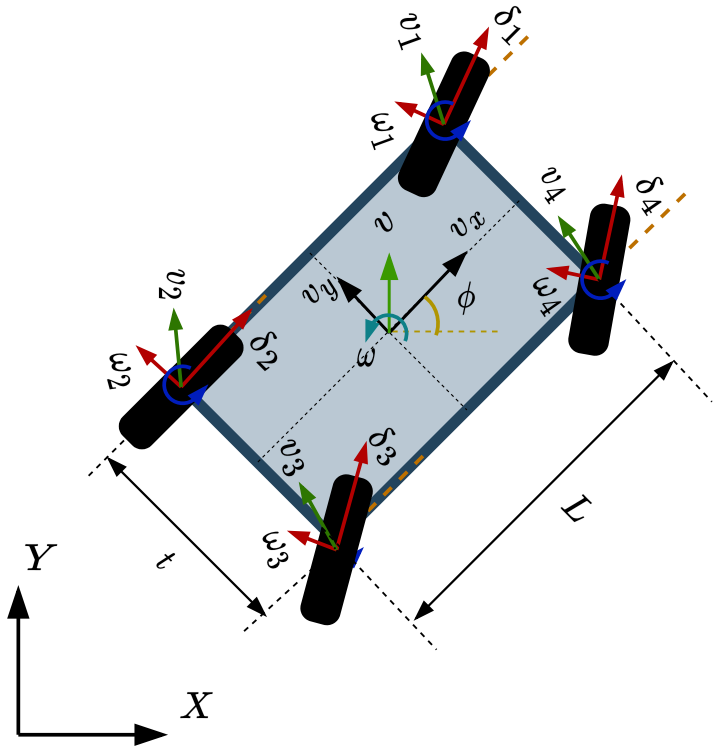


FIGURE 3.1: Top view of the robot

3.2 Kinematic Model

The drivetrain kinematics governs the relationship between the robot velocity vector $\boldsymbol{v}$, wheel steering angles $\boldsymbol{\delta}$ and the wheel velocities ($\boldsymbol{v}_{\text{wx}}$ and $\boldsymbol{v}_{\text{wy}}$). In an ideal case (assuming no slipping and no ksidding conditions), the relationship is as follows (Arab et al. [1]).

$$\boldsymbol{v}_{\text{wx}} = \mathbf{T}_{\text{wx}}(\boldsymbol{\delta})\boldsymbol{v} = r_{\text{w}}\boldsymbol{\omega},$$

$$\boldsymbol{v}_{\text{wy}} = \mathbf{T}_{\text{wy}}(\boldsymbol{\delta})\boldsymbol{v}, \text{ and under ideal conditions, } \boldsymbol{v}_{wy} = 0$$

where

$$\mathbf{T}_{\text{wy}}(\boldsymbol{\delta}) = \begin{bmatrix} \cos\delta_1 & \sin\delta_1 & -\frac{t}{2}\cos\delta_1 + \frac{L}{2}\sin\delta_1 \\ \cos\delta_2 & \sin\delta_2 & -\frac{t}{2}\cos\delta_2 - \frac{L}{2}\sin\delta_2 \\ \cos\delta_3 & \sin\delta_3 & \frac{t}{2}\cos\delta_3 - \frac{L}{2}\sin\delta_3 \\ \cos\delta_4 & \sin\delta_4 & \frac{t}{2}\cos\delta_4 + \frac{L}{2}\sin\delta_4 \end{bmatrix}$$

and

$$\mathbf{T}_{\text{wy}}(\boldsymbol{\delta}) = \begin{bmatrix} -\sin\delta_1 & \cos\delta_1 & \frac{L}{2}\cos\delta_1 + \frac{t}{2}\sin\delta_1 \\ -\sin\delta_2 & \cos\delta_2 & -\frac{L}{2}\cos\delta_2 + \frac{t}{2}\sin\delta_2 \\ -\sin\delta_3 & \cos\delta_3 & -\frac{L}{2}\cos\delta_3 - \frac{t}{2}\sin\delta_3 \\ -\sin\delta_4 & \cos\delta_4 & \frac{L}{2}\cos\delta_4 - \frac{t}{2}\sin\delta_4 \end{bmatrix}$$

Further, the inverse kinematic solution can be obtained by solving equations 3.2, solving for $\boldsymbol{\delta}$ and $\boldsymbol{\omega}$. Low-level feedback controllers (PID) on each wheel module maintain wheel angles and velocities. This information can be used to obtain the complete kinematic solution in the absence of slipping. On the other hand, when accounting for non-ideal conditions, the slipping effect is characterised by $\|\boldsymbol{v}_{wx} - r_{\text{w}}\boldsymbol{w}\| \neq 0$ and similarly, the skidding effect by $\|\boldsymbol{v}_{wy}\| \neq 0$.

3.3 Dynamic Model

Dynamic modeling is essential to control the robot reliably and considering the robot as a multi-rigid body. We assume the only external forces are tire friction forces and gravity. When all the wheels are in contact with the ground, the longitudinal and lateral forces of the wheels $\boldsymbol{F}_{\text{x}}$ and $\boldsymbol{F}_{\text{y}}$ act on the robot frame. We obtain dynamic model is as follows (Arab et al. [1]).

$$\mathbf{M}\dot{\boldsymbol{v}} + \mathbf{C}(\boldsymbol{v})\boldsymbol{v} + \boldsymbol{g}(\boldsymbol{q}) = \mathbf{T}_{\text{wx}}(\boldsymbol{\delta})^T\boldsymbol{F}_{\text{x}} + \mathbf{T}_{\text{wy}}(\boldsymbol{\delta})^T\boldsymbol{F}_{\text{y}}$$

$\mathbf{M} \in \mathbb{R}^{3\times3}$ is a symmetric, positive definitive inertia matrix and the vector representing the Coriolis and centrifugal torques is denoted as $\mathrm{C}(\boldsymbol{v})\boldsymbol{v} \in \mathbb{R}^3$ and are given by the following equations. We denote m as the robot mass and I_{zz} as the moment of inertia about the z-axis.

$$\mathbf{M} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I_{\mathrm{zz}} \end{bmatrix} \quad \mathbf{C}(\boldsymbol{v}) = \begin{bmatrix} 0 & m\dot{\phi} & 0 \\ m\dot{\phi} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$\boldsymbol{g}$ represents the gravitational force. When the robot is on a flat plane, the force $\boldsymbol{g}(\boldsymbol{q})$ acts vertically downwards and does not affect the dynamic model. However, we consider a general case where the robot may be used in an inclined plane — where the force $\boldsymbol{g}(\boldsymbol{q})$ acts on the xy-plane — and consider the term as an bounded arbitrary disturbance.

3.4 Odometry

To achieve independent navigation, efficient and safe task execution, a robot must possess the capability to autonomously navigate, path-plan, and localize itself within its environment. Accurate localization stands as a foundational challenge in mobile robot applications. Maintaining continuous knowledge of its position is crucial for a robot's autonomous navigation. Consequently, significant attention has been devoted to studying the localization problem, leading to the proposal of various techniques to address this challenge.

Among these techniques, one of the simplest forms of localization involves utilizing wheel odometry methods, which rely on wheel encoders to gauge the rotation of the robot's wheels. These methods incrementally leverage wheel rotation measurements in conjunction with the robot's motion model to determine its present location relative to a global reference coordinate system.

Based on the kinematic model explained in section 3.2 the odometry is computed with the following relations using the steering angles $\boldsymbol{\delta} = [\delta_1, \delta_2, \delta_3, \delta_4]^T$ and wheel velocities $\boldsymbol{v} = [v_1, v_2, v_3, v_4]^T$

$$\delta_f = \tan^{-1}\left(\frac{2\tan\delta_1\tan\delta_4}{\tan\delta_1 + \tan\delta_4}\right) \quad \delta_r = \tan^{-1}\left(\frac{2\tan\delta_2\tan\delta_3}{\tan\delta_2 + \tan\delta_3}\right)$$

$$a_f = \frac{\cos\delta_f(\tan\delta_f - \tan\delta_r)}{L} \quad a_r = \frac{\cos\delta_r(\tan\delta_f - \tan\delta_r)}{L}$$

$$v_f = r_w\sqrt{\frac{v_1^2 + v_4^2}{2 + (ta_f)^2/2}} \quad v_r = r_w\sqrt{\frac{v_2^2 + v_3^2}{2 + (ta_r)^2/2}}$$

$$v_x = \frac{v_f\cos\delta_f + v_r\cos\delta_r}{2} \quad v_y = \frac{v_f\sin\delta_f + v_r\sin\delta_r}{2}$$

$$v = \sqrt{v_x^2 + v_y^2}$$

$$\omega = \frac{v_f a_f + v_r a_r}{2} \quad \Rightarrow \phi = \sum \omega\Delta t$$

$$x = \sum(v_x\cos\phi - v_y\sin\phi)\Delta t \;\text{ and }\; y = \sum(v_x\sin\phi + v_y\cos\phi)\Delta t$$

We finally obtain relations for determining the pose $\boldsymbol{q} = [x, y, \phi]^T$ of the robot along with its linear and angular velocities.

3.5 Simulation

For simulating the robot, we use the Gazebo (Koenig and Howard [14]) simulation environment. Gazebo facilitates the simulation of robotic and sensor applications within 3D indoor and outdoor environments. Operating on a Client-Server architecture, it employs the same topic-based Publish and Subscribe model of ROS for communication between processes. This is similar to the design of ROS and thus, it offers a smooth interface with ROS. It's Player interface allows Gazebo clients to access data via shared memory. Within Gazebo, each simulation object can link to one or more controllers responsible for interpreting commands to control the object's behavior and providing its state. The data produced by these controllers are disseminated into shared memory through Ifaces. Other processes' Ifaces can access this data from shared memory, conducting communication between the robot control software processes and Gazebo, irrespective of programming language or the hardware architecture used in the computer. (Takaya et al. [21]).

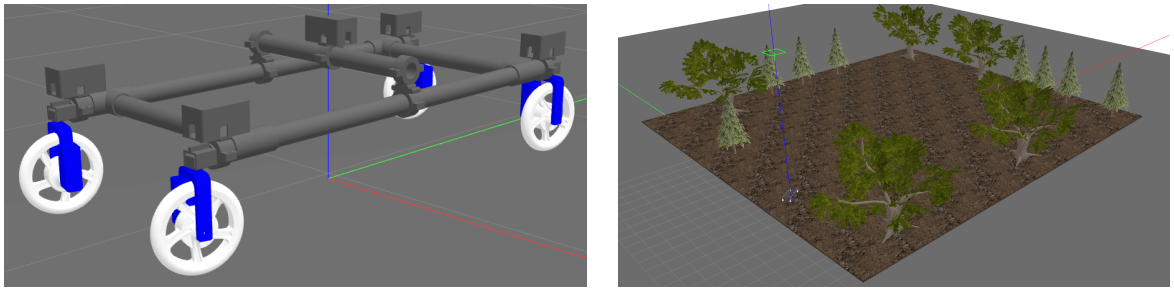


FIGURE 3.2: Gazebo simulation

We can simulate multiple sensors in the Gazebo simulation environment using open-source plugins. Some of these sensors incorporated into the robot is simuation are GPS, IMU, LiDAR, and cameras. The sensors are provided to test on simulation before incorporating real sensors on the robot. We also implement some autonomous navigation features, such as outdoor localization in addition to the control software.

3.6 Localization Algorithms

3.6.1 Introduction

Robot localization refers to the fundamental process by which a robot determines its precise position and orientation within its environment in real time. This critical capability allows robots to navigate autonomously, execute tasks accurately, and interact effectively with their surroundings. The process involves integrating sensor data, such as measurements from cameras, laser scanners, wheel encoders, or other environmental sensors, with a map of the surroundings to estimate the robot's location. Numerous techniques are employed for localization, ranging from simple methods like wheel odometry to more complex approaches such as simultaneous localization and mapping (SLAM). The goal is to continually update the robot's estimated position relative to a known reference frame, enabling it to make informed decisions and navigate effectively in changing or unknown environments. Efficient and accurate localization is pivotal for the successful operation of robots in diverse applications, including but not limited to industrial automation, autonomous vehicles, and robotic exploration.

3.6.2 Extended Kalman Filter (EKF) Localization

One of the most common algorithms used for robot localization is the EKF algorithm. The Kalman filter, conceptualized in by Rudolph Emil Kalman, is a widely used method employed in filtering and prediction, particularly in linear systems. This algorithm is designed for computing beliefs or estimating the state of continuous systems, aiming to refine and predict their future states by incorporating sensor measurements and system dynamics. However, its application is limited to systems with continuous states; it doesn't effectively handle

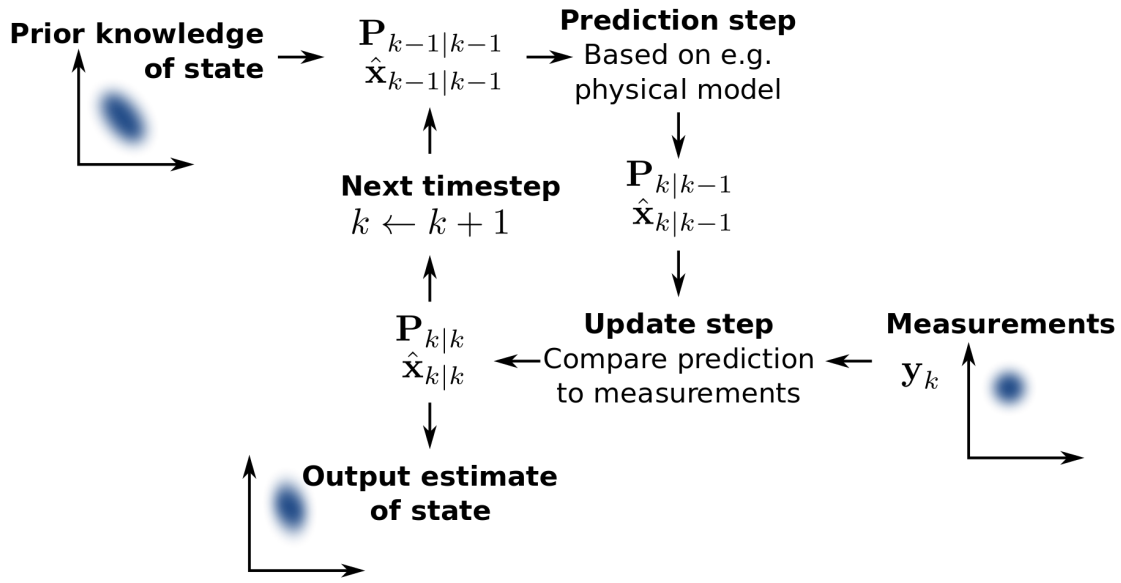


FIGURE 3.3: Kalman Filtering

discrete or hybrid state spaces. For the Kalman filter to function optimally, the probability distribution of the next state must follow a linear function concerning its parameters, accompanied by Gaussian noise, ensuring an adequate representation of uncertainty in the system's measurements and predictions. This reliance on linearity and Gaussian noise characteristics restricts its use in systems that do not conform to these particular probabilistic assumptions.

The EKF algorithm solves this problem by linearizing a non-linear probability distribution and applying the Kalman Filter on it. Our goal is to estimate the six degree of freedom (6-DOF) pose $\mathbf{q}$ and velocity of a robot over time. The EKF algorithm is briefly discussed as follows. The process can be described as a non-linear dynamic system with $\mathbf{q}_k = f(\mathbf{q}_{k-1}) + \mathbf{w}_{k-1}$, where $\mathbf{q}_k$ is the pose of the robot at time k, f is a non-linear state transition function, and $\mathbf{w}_{k-1}$ is the process noise. Further, we recieve measurements from the on-board sensors like GPS, IMU and Odometry, which is generally represented as $\mathbf{z}_k = h(\mathbf{q}_k) + \mathbf{v}_k$. Here, $\mathbf{z}_k$ is the measurement at time k, h is a nonlinear sensor model that maps the state into measurement space, and $\mathbf{v}_k$ is the normally distributed measurement noise.

The prediction step of the EKF algorithm is as follows.

$$\hat{\mathbf{q}_k} = f(\mathbf{q}_{k-i})$$

$$\hat{\mathbf{P}} = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{Q}$$

where f is our kinematic model elaborated in 3.2. The estitmate error covariance $\mathbf{P}$, is projected via $\mathbf{F}$, the Jacobian of f, and then perturbed by $\mathbf{Q}$, which is the process noise covariance.

The update steps of the EKF are as follows.

$$\mathbf{K} = \hat{\mathbf{P}}_k\mathbf{H}^T\left(\mathbf{H}\hat{\mathbf{P}}_k\mathbf{H}^T + \mathbf{R}\right)^{-1}$$

$$\mathbf{q}_k = \hat{\mathbf{q}}_k + \mathbf{K}\left(\mathbf{z} - \mathbf{H}\hat{\mathbf{q}}_k\right)$$

$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}\mathbf{H}\right)\hat{\mathbf{P}}_k\left(\mathbf{I} - \mathbf{K}\mathbf{H}\right)^T + \mathbf{K}\mathbf{R}\mathbf{K}^T$$

The Kalman gain is calculated using the observation matrix, $\mathbf{H}$, the measurement covariance, $\mathbf{H}$, and $\hat{\mathbf{P}}_k$. We use the gain to update the state vector and covariance matrix. The mentioned Extended Kalman Filter (EKF) formulation is utilized via the ROS package `robot_localization` (Moore and Stouch [17]), incorporated as the `ekf_localization_node`. This module is designed to accommodate various sensors, assuming that each sensor provides measurements for the estimated state variables. A key feature of the `ekf_localization_node` is its ability to execute partial updates of the state vector. This functionality is essential for any future state estimation nodes integrated into robot_localization. It proves crucial in handling sensor data that doesn't capture every variable in the state vector, which is often the norm. In practical terms, achieving partial updates can be accomplished through $\mathbf{H}$.

Integrating GPS data: The `navsat_transform_node` adjusts GPS data to align with the initial pose (position and orientation) of the robot within its world frame. This streamlined transformation significantly eases the integration of GPS data. To operate effectively, the node relies on three essential pieces of information: the robot's current pose estimate within its world frame, a heading referenced to the Earth's surface, and geographic coordinates represented as latitude/longitude pairs.

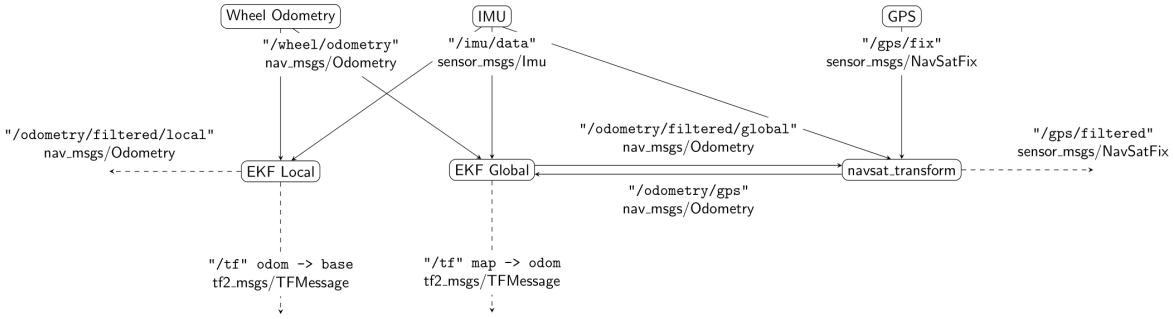


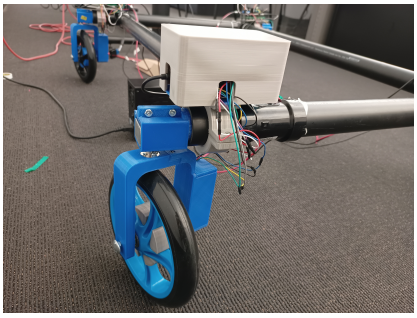
FIGURE 3.4: Overview of `robot_localization` workflow with GPS data

3.6.3 Unscented Kalman Filter (UKF) Localization

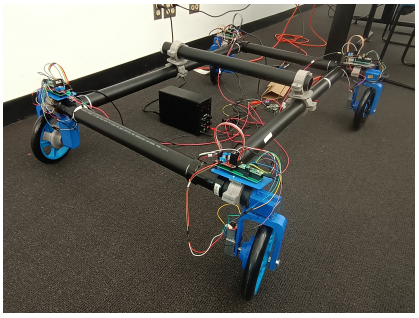
An alternative approach to the EKF is the Unscented Kalman Filter (UKF), introduced to enhance performance compared to the EKF. Rather than approximating nonlinear functions through linearization, the UKF employs a set of points that traverse the actual nonlinear function directly. These points are selected ensuring their statistical characteristics, such as mean, covariance, and higher-order moments, align with those of a Gaussian random variable. By propagating these points through the system, the UKF recalculates their mean and covariance, yielding more precise estimations compared to the conventional linearization approach using Taylor's series for ordinary functions. This methodology offers improved accuracy when dealing with complex nonlinear systems, providing a robust alternative to the EKF.

Results and Conclusion

4.1 Design



Steering-Driving module



Isometric view

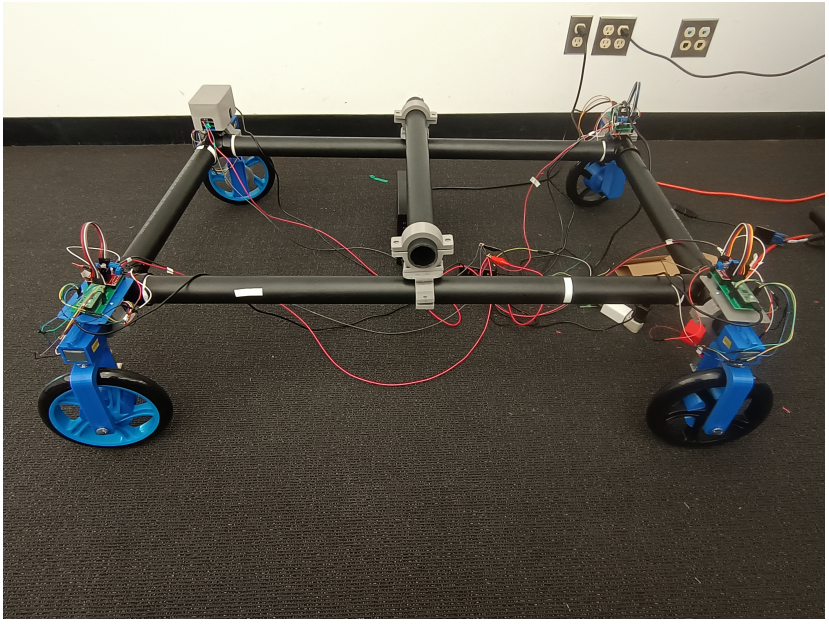


FIGURE 4.1: Robot model prototype

4.2 EKF Localization

Figure 4.2 shows the result of the fusion of IMU data with the odometry data collected by the wheel encoders. The modeling details of the odometry can be found in Section 3.4. The implementation of the EKF Sensor fusion is based on the details provided in Section 3.6. We compare the efficacy of the localization algorithms against the ground truth poses of

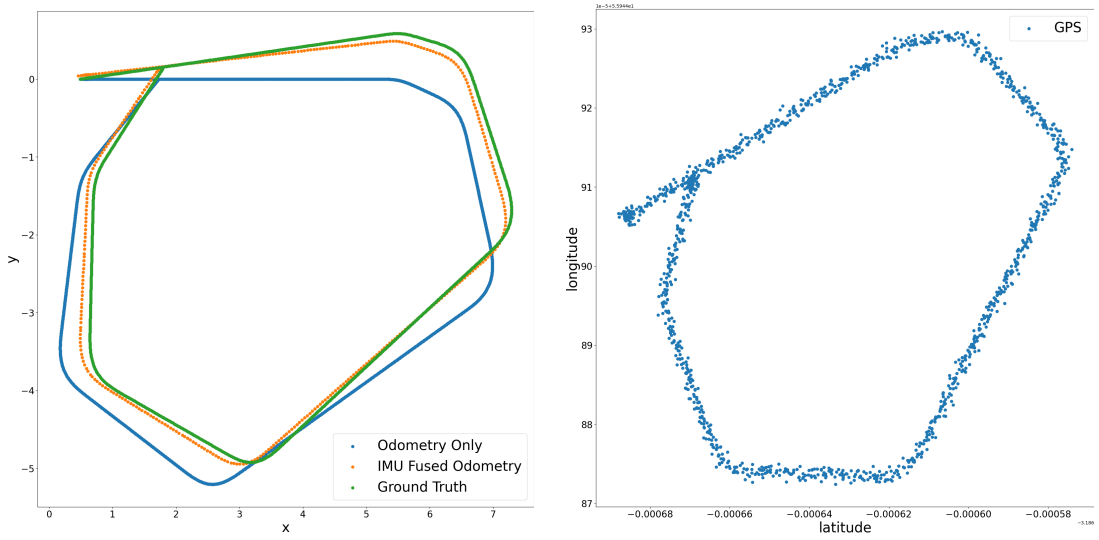


FIGURE 4.2: EKF Localization results with IMU

the robot. It is evident from the Figure 4.2 that the odometry-only localization suffers from drift of the sensor values. However, the IMU-fused odometry offers a better estimate of the localization and matches closely with the ground truth.

In Figure 4.2 the GPS data is represented as a scatter plot. The GPS in real-life is prone to noise. Thus, it is modelled with a Gaussian noise profile so as to simulate the real-world conditions as far as possible.

The top row of the Figure 4.5 shows the plot of the combined results of GPS-IMU-Odometry EKF Sensor fusion. The GPS data is incorporated for the estimation using the `navsat_transform` mentioned in Section 3.6. We can see that the fused trajectory closely matches with the

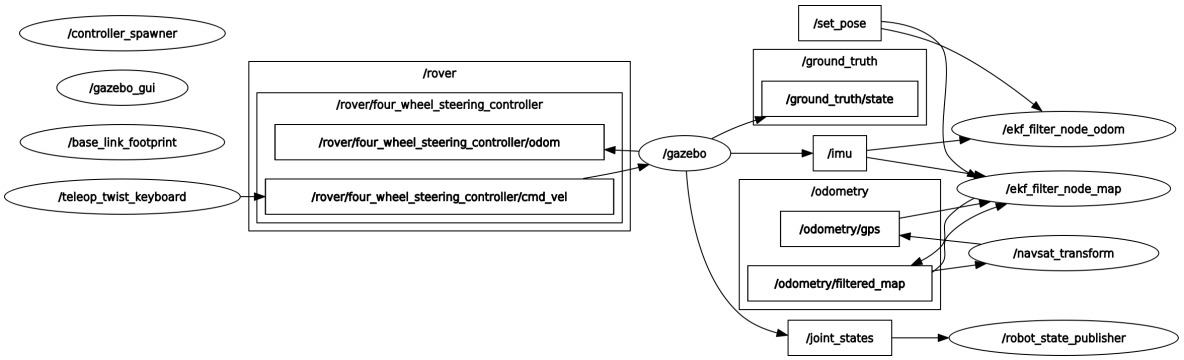


FIGURE 4.3: ROS Graph depicting all the EKF nodes

ground truth. The results of both GPS-Odometry fusion and GPS-IMU-Odometry fusion are better than with only odometry data. The GPS-IMU-Odometry fusion process uses the EKF fused result of IMU and odometry to further refine the GPS fusion. The bottom row of the

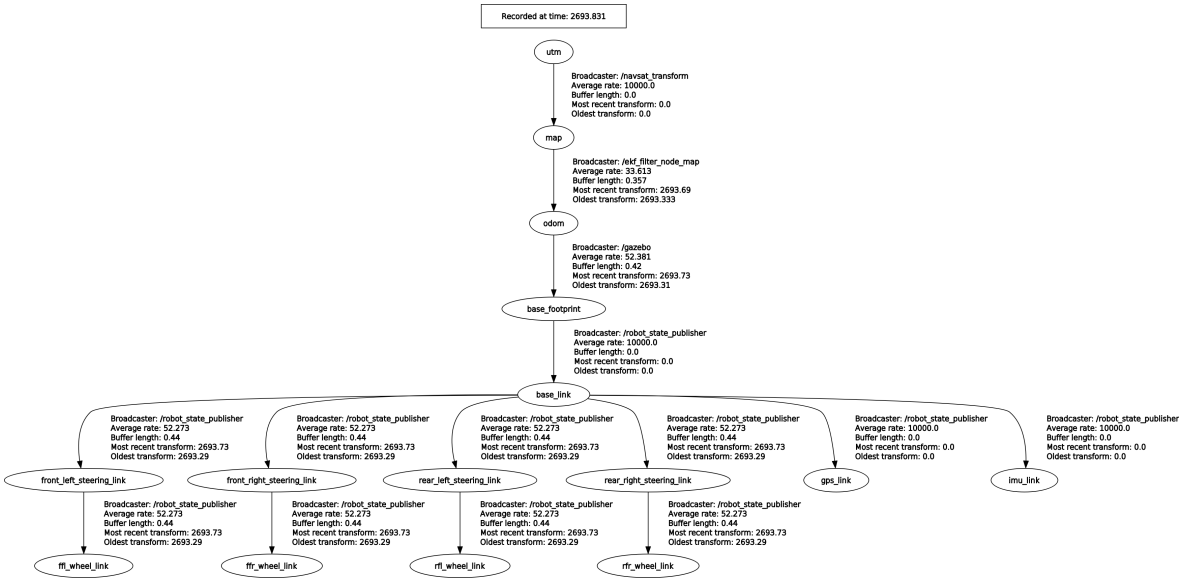


FIGURE 4.4: ROS Transform Tree depicting all the frames of the robot

Figure 4.5 shows the Filtered GPS outputs against the raw GPS sensor readings. It can be observed that the variance of the the Filtered GPS data is lower than than the variance of the raw GPS readings. This is the result of the EKF sensor fusion algorithm, which improves the GPS readings as well.

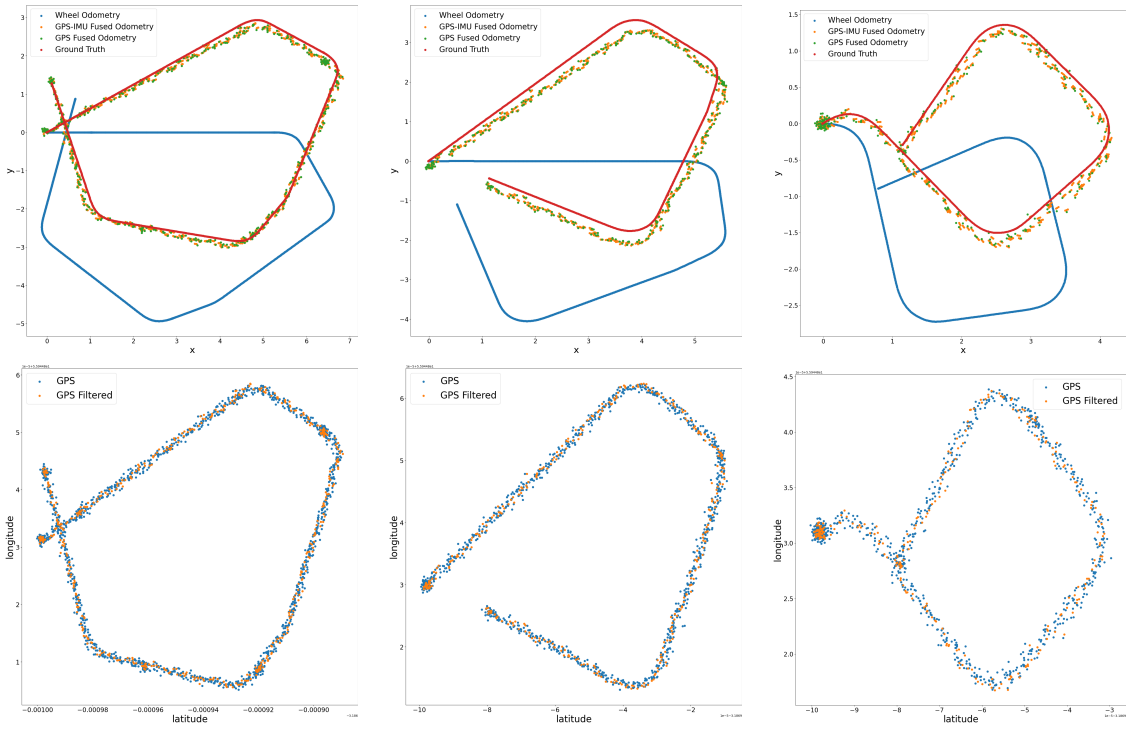


FIGURE 4.5: EKF Localization results with GPS

4.3 Conclusion and Future Scope

In this project we study the four-wheel steering and driving configuration of the robotic platform and compare it with other steering configurations in literature. We design the robotic platform with Design for Assembly principles. The software design of the robotic platform is based on the widely used `ros_control` paradigm, aiding seamless interaction between the hardware and the software. We also provide the simulation of the robotic platform using Gazebo to test sensors such as GPS and IMU. Furthermore, we implement the EKF algorithm to quantify the efficacy of the odometry model in simulation.

Future research would focus on the path planning and path tracking control algorithms that are particularly suited for the 4-WSD configuration and its motion model. Accuracy of the motion and odometry models can be improved by considering the dynamic model of the robot, involving tire-ground interaction models in the formulation.

References

[1] Aliasghar Arab, Ilija Hadžić, and Jingang Yi. Safe predictive control of four-wheel mobile robot with independent steering and drive. pages 2962–2967, 2021.

[2] Abbas Atefi, Yufeng Ge, Santosh Pitla, and James Schnable. Robotic technologies for high-throughput plant phenotyping: Contemporary reviews and future perspectives. *Frontiers in Plant Science*, 12, 2021.

[3] C. Wouter Bac, Eldert J. van Henten, Jochen Hemming, and Yael Edan. Harvesting robots for high-value crops: State-of-the-art review and challenges ahead. *Journal of Field Robotics*, 31(6):888–911, 2014.

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

[5] David Ball, Ben Upcroft, Gordon Wyeth, Peter Corke, Andrew English, Patrick Ross, Tim Patten, Robert Fitch, Salah Sukkarieh, and Andrew Bate. Vision-based obstacle detection and navigation for an agricultural robot. *Journal of Field Robotics*, 33(8): 1107–1130, 2016.

[6] Marcel Bergerman, John Billingsley, John Reid, and Eldert van Henten. Robotics in agriculture and forestry. pages 1463–1492, 2016.

[7] Christophe Cariou, Roland Lenain, Benoit Thuilot, and Michel Berducat. Automatic guidance of a four-wheel-steering mobile robot for accurate field operations. *Journal of Field Robotics*, 26(6-7):504–518, 2009.

[8] Renzo Carpio, Ciro Potena, Jacopo Maiolini, Giovanni Ulivi, Nicolás Bono Rosselló, Emanuele Garone, and Andrea Gasparri. A navigation architecture for ackermann vehicles in precision farming. *IEEE Robotics and Automation Letters*, PP:1–1, 2020.

[9] Nived Chebrolu, Philipp Lottes, Alexander Schaefer, Wera Winterhalter, Wolfram Burgard, and Cyrill Stachniss. Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields. *The International Journal of Robotics Research*, 36:027836491772051, 2017.

[10] Sachin Chitta, Eitan Marder-Eppstein, Wim Meeussen, Vijay Pradeep, Adolfo Rodríguez Tsouroukdissian, Jonathan Bohren, David Coleman, Bence Magyar, Gennaro Raiola, Mathias Lüdtke, and Enrique Fernández Perdomo. ros_control: A generic and simple control framework for ros. *The Journal of Open Source Software*, 2017.

[11] Peng Gao, Junsheng Jiang, Jian Song, Fuxiang Xie, Yang Bai, Yuesheng Fu, Zhengtao Wang, Xiang Zheng, Shengqiao Xie, and Baocheng Li. Canopy volume measurement of fruit trees using robotic platform loaded lidar data, 2021.

[12] Miguel Garrido, Dimitris S. Paraforos, David Reiser, Manuel Vázquez Arellano, Hans W. Griepentrog, and Constantino Valero. 3d maize plant reconstruction based on georeferenced overlapping lidar point clouds. *Remote Sensing*, 7(12):17077–17096, 2015.

[13] Jayantha Katupitiya, Ray Eaton, and Tahir Yaqub. Systems engineering approach to agricultural automation: New developments. pages 1–7, 2007.

[14] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. 3:2149–2154 vol.3, 2004.

[15] Anthony Mandow, Jorge L. Martínez, Jesús Morales, José-Luis Blanco, Alfonso J. García-Cerezo, and Javier González. Experimental kinematics for wheeled skid-steer mobile robots. *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1222–1227, 2007.

[16] Anil Mital, Anoop Desai, Anand Subramanian, and Aashi Mital. 7 - designing for assembly and disassembly. In *Product Development (Second Edition)*, pages 159–202. Elsevier, Oxford, second edition edition, 2014.

[17] T. Moore and D. Stouch. A generalized extended kalman filter implementation for the robot operating system. 2014.

[18] Stephen Nuske, Supreeth Achar, Terry Bates, Srinivasa Narasimhan, and Sanjiv Singh. Yield estimation in vineyards by visual grape detection. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2352–2358, 2011.

[19] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Ng. Ros: an open-source robot operating system. *ICRA Workshop on Open Source Software*, 3, 2009.

[20] D.C. Slaughter, D.K. Giles, and D. Downey. Autonomous robotic weed control systems: A review. *Computers and Electronics in Agriculture*, 61(1):63–78, 2008. Emerging Technologies For Real-time and Integrated Agriculture Decisions.

[21] Kenta Takaya, Toshinori Asai, Valeri Kroumov, and Florentin Smarandache. Simulation environment for mobile robots testing using ros and gazebo. pages 96–101, 2016.

[22] Diego Tiozzo Fasiolo, Lorenzo Scalera, Eleonora Maset, and Alessandro Gasparetto. Towards autonomous mapping in agriculture: A review of supportive technologies for ground robotics. *Robotics and Autonomous Systems*, 169:104514, 2023.

[23] Ya Xiong, Cheng Peng, Lars Grimstad, Pål Johan From, and Volkan Isler. Development and field evaluation of a strawberry harvesting robot with a cable-driven gripper. *Computers and Electronics in Agriculture*, 157:392–402, 2019.