POLITECNICO DI TORINO

Master Degree course in Automotive Engineering

Master Degree Thesis

Development of a terrain-aware omnidirectional robotic platform for enhanced mobility in agricultural fields.

Supervisors

Dott. Andrea Botta

Prof. Giuseppe Quaglia

Dott. Francesco Amodio

Candidate

Davide Danzi

Academic Year 2025-2026

Abstract

The introduction of robotics into the precision agriculture sector requires platforms capable of operating in unstructured environments, where the interaction between soil and vehicle becomes a critical factor for motion. This thesis describes the development of AgriMaRo, an omnidirectional robotic platform designed for complex agricultural contexts. The proposed solution adopts a three-wheeled kinematic configuration, featuring one translating wheel to allow the robot to adapt to various crop geometries and variations in row width.

The work begins with an analysis of the state of the art and the platform architecture, describing its functionalities and the original sensor suite. Throughout the discussion, it is highlighted that the initial equipment was insufficient to accurately describe the vehicle's dynamics; consequently, a targeted analysis was conducted to identify new transducers, leading to their physical integration into the AgriMaRo system. Through the implementation of an Extended Kalman Filter (EKF), a sensor fusion algorithm was developed, aimed at the robust estimation of the robot's state and variables that are not directly measurable by the onboard sensors. The use of the filter allows for a detailed analysis of the robot's actual dynamics, reducing the impact of noise and providing a reliable dataset for subsequent analyses.

Particular emphasis is placed on the evaluation of longitudinal slip, a fundamental quantity for understanding traction efficiency. The investigation addresses the methodological challenges related to the instantaneous calculation of slip, highlighting how the precision and veracity of the acquired data are decisive in correctly identifying the discrepancy between linear velocity and wheel angular velocity.

The data obtained form the basis for a terrain classification system: by correlating the vehicle's dynamic response with different operating conditions, the system can identify the soil type through comparison with a reference database. Finally, the thesis outlines how accurate knowledge of slip and terrain nature enables the development of advanced traction control algorithms. Such systems are essential for improving driving precision and robot stability, ensuring safe and efficient operation even on soft or uneven ground. The work performed thus lays the foundations for a terrain-aware platform, capable of adapting its dynamic behavior to real-world challenges.

Contents

List of Figures

Chapter 1

Introduction

The need to ensure sustainable food production in the face of constant global population growth (estimated to increase food demand by 50 - 54% by 2050, according to the FAO [7]) is driving the agricultural sector towards rapid and profound modernization [10]. The response to this systemic challenge is **Precision Agriculture**, a methodological approach that leverages advanced technologies, such as sensor networks, data analytics, and GPS systems, to monitor the spatial and temporal variability of crop fields. This allows for the optimization of water resources, fertilizers, and pesticides, thereby reducing environmental impact while simultaneously maximizing yields [29]. Within this scenario, **greenhouse** cultivation takes on an increasingly strategic role, ensuring higher net productivity and rigorous climate control compared to open-field farming. However, operating within such environments imposes severe engineering constraints: restricted maneuvering spaces, the presence of fixed infrastructure, and the extreme proximity of the crops require the use of compact machinery that is highly maneuverable and capable of navigating safely without damaging the foliage or fruits [23].

To address the progressive shortage of seasonal labor and increase the efficiency of routine agricultural processes, academic research and industry have recently proposed various cutting-edge robotic solutions. There has been a rapid transition from the use of autonomous tractors for soil tillage and drones for aerial monitoring, to complex systems equipped with computer vision for automated crop harvesting (Robotic Harvesters). Although these technologies offer undeniable advantages, their large-scale adoption still encounters obstacles related to high initial investment costs, technological resistance from operators, and the need for specialized technical personnel for maintenance. Regarding mobility specifically within greenhouses, the most effective architectures are shifting towards intelligent and reconfigurable mobile platforms [23]. It is to meet this particular operational need that **AgriMaRo (Agricultural Mate Robot)** was originally developed. AgriMaRo is a three-wheel-drive omnidirectional rover, structurally designed to adapt to row widths and navigate with extreme agility in confined spaces.

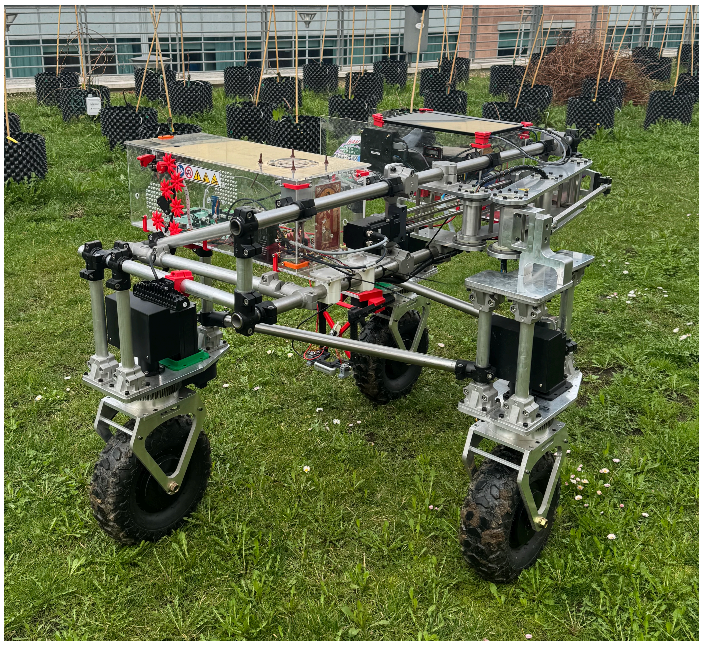

Figure 1.1. AgriMaRo.

Despite the high positioning capabilities offered by its omnidirectional kinematics, recent testing campaigns conducted on the AgriMaRo prototype have revealed critical limitations related to the perception and management of its actual driving dynamics. The low-level control system, based exclusively on ideal theoretical models and lacking mechanical differentials, imposed angular velocities on the wheels that frequently diverged from the vehicle's actual forward motion on the ground. This discrepancy resulted in a high and uncompensated level of **longitudinal slip**, a phenomenon that leads to unnecessary energy dissipation, potential soil damage, and a marked loss of odometric precision. Furthermore, the absence of high-fidelity sensory data prevented the design of an advanced traction control system capable of actively adapting to the changing conditions of the agricultural terrain.

The main objective of this thesis work is to resolve these critical issues through a radical optimization of sensory acquisition and the accurate estimation of the vehicle's dynamics. The proposed approach involves the integration of new transducers, including a camera for visual-inertial odometry, and the implementation of an **Extended Kalman Filter (EKF)** to fuse the signals in real-time and reduce measurement noise. This newly filtered and robust dataset will pursue a dual and synergistic objective: on the one hand, it will provide the essential input for a **Convolutional Neural Network** (CNN), developed in parallel, for the instantaneous recognition of terrain difficulty and compliance; on the other hand, it will allow for the reliable and continuous calculation of the longitudinal slip of each wheel. This parameter will become the fundamental process variable to hypothesize and structure a high-level **traction control** architecture aimed at maximizing motor efficiency, preventing sinkage, and ensuring the robot's stability on various and treacherous agricultural grounds.

To systematically illustrate the methodological rigor and the development of the work performed, this thesis is structured according to a sequential logical flow. Following this initial introduction, the second chapter analyzes the **State of the Art** of the AgriMaRo platform, defining its original kinematics, hardware constraints, and the architectural limits that motivated this study. The third chapter describes the initial **Sensor Suite**, illustrating the physical operating principles of the onboard rotary and inertial transducers, as well as the necessary calibration procedures. Moving forward, the fourth chapter outlines the fundamental mathematical theory and the initial design of the **Extended Kalman Filter** (EKF), evaluating its tuning criteria and the operational limits that emerged. In light of these limitations, the fifth chapter illustrates the enhancement of the perception system (**Sensor Augmentation**) through the analysis, selection, and software integration of the Intel RealSense T265 Tracking Camera. The sixth chapter then collects the final and **recalibrated implementation** of the EKF, presenting the experimental validation of the algorithm using data acquired in the field. The seventh chapter delves into the theory and evaluation of **longitudinal slip**, analyzing the robustness of the developed algorithm for real-time calculation. Finally, the eighth chapter illustrates how the optimized sensory data obtained from the EKF were used to test a **terrain recognition neural network**, developed in a parallel study, and introduces a preliminary, theoretical architecture for the **adaptive traction control**. The manuscript concludes by summarizing the main results of this work and suggesting potential future developments.

Chapter 2

State of the Art of the AgriMaRo Platform

This chapter aims to illustrate the state of the art of the **AgriMaRo** robotic platform at the beginning of this thesis work. The structural characteristics, the design choices related to the locomotion system, and the kinematic model governing its motion will be analyzed in detail. This technical overview is essential to understand the technological starting point and the mechanical constraints upon which the subsequent implementations and optimizations, subject of this study, were built, providing a comprehensive picture of the vehicle's operation in its original configuration.

2.1 General Characteristics and System Architecture

The AgriMaRo project stems from the need to develop a robot capable of operating inside greenhouses with variable soil-based crops [2]. Unlike hydroponic or tabletop greenhouses, which offer structured and paved environments, or open-field farming with ample maneuvering space, this operational scenario presents unique challenges: narrow spaces and unstructured, irregular surfaces. Furthermore, the wide variety of crops possible in these facilities requires a platform characterized by high design flexibility.

A fundamental requirement that emerged during the preliminary analysis, precisely due to these restricted maneuvering spaces, is **omnidirectionality**: the ability to translate in any direction and rotate around an arbitrary instantaneous center of rotation is essential for moving nimbly in the narrow spaces between crop rows, thereby mitigating the issues related to spatial constraints.

Considering the heterogeneity of the crops, the founding idea of the project was to create a machine capable of adapting its morphology to the working environment. Although initially a robot capable of varying its dimensions along all three main axes (height, length, and width) was hypothesized, a careful analysis of the cost-benefit ratio and mechanical complexity led to a simplified design. It was therefore decided to limit the **reconfigurability** solely to the widening of the track, considered by far the most critical parameter for operating independently of the crop row widths.

To define the dimensional limits of the vehicle, an in-depth survey of typical sowing configurations was conducted, analyzing the distances between rows and along the rows of plants. The results highlighted the need for a platform capable of varying its operational width between 40 cm and 80 cm, dimensions assumed as the lower and upper limits for the extension of the third leg. Regarding height, it was established that the chassis must guarantee a ground clearance of at least 70 cm. This value is not arbitrary; it takes into account the average height of the crops, the possible presence of obstacles or steps on the ground, and the vertical footprint of the operational tools installed on board.

From an architectural perspective, the robot is based on a **three-wheel drive** configuration. The choice to adopt exactly three contact points derives from a precise kinematic need: in space, three points always define a plane. This isostatic characteristic ensures that all wheels maintain constant contact with the ground, even in the presence of the strong irregularities typical of agricultural terrain, ensuring continuous traction without the need for complex and heavy suspension systems. The design of AgriMaRo was developed starting from a careful analysis of classic three-wheeled vehicle typologies, aiming to opt for the solution that best suited the specific and complex required characteristics. In particular, during the conceptual phase, the following reference configurations were studied:

- **Delta:** equipped with a single steering front wheel and two fixed rear wheels;

- **Tadpole:** equipped with two front wheels and a single rear wheel, declinable in two variants depending on whether the steering is applied to the front axle or exclusively to the rear wheel.
- **Sidecar:** two wheels arranged on one side and a single wheel on the other, with the front wheel steering. This became the standard configuration adopted for the manual control of the vehicle via joystick;

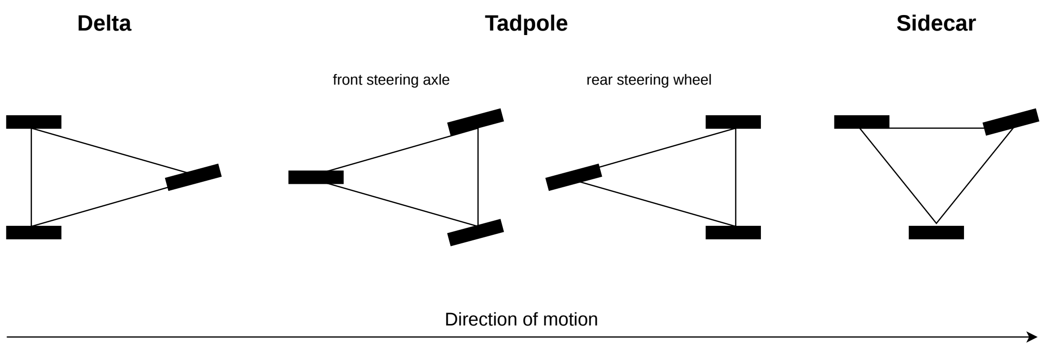

Figure 2.1. Main layouts of road three-wheelers.

The final choice fell on the "Sidecar" scheme as the default configuration. This decision stems directly from the need to implement an asymmetrical gantry structure: this solution allows the wheelbase to be varied in an extremely simple and functional manner, as it only requires acting on the extension of the single mobile wheel, without risking compromising the overall stability or operational symmetry during motion. The ultimate goal of the kinematic model is therefore to exploit this versatility to determine, instant by instant, the most efficient configuration in relation to the path to be followed or any obstacles to be avoided.

To achieve this variation reliably, the structure integrates an actively actuated **Track Adjustment Mechanism** driven by a stepper motor (NEMA 23) coupled with a **ball screw mechanism**. The core of this kinematic architecture is a **Scott-Russell linkage**, a mechanism specifically designed to convert linear motion along one axis into a perpendicular linear translation. The third leg of the robot is not mounted directly onto the linear guides; rather, it is rigidly attached to the final link of the Scott-Russell mechanism. This design allows the robot to adapt its track width in real time to the geometry of the crop row.

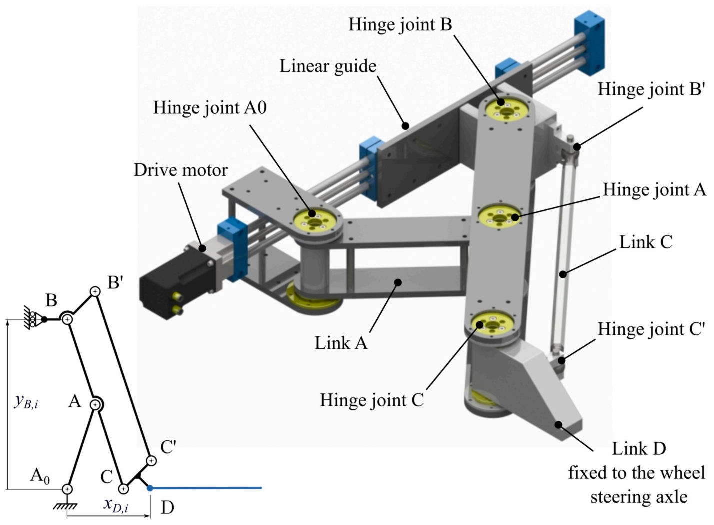

Figure 2.2. Track adjustment mechanism.

This particular **gantry structure**, supported by **three independent and steerable hub-motors** positioned at the vertices of the frame, renders the platform omnidirectional and allows the end-effector to be positioned directly above the row. This greatly facilitates direct operations on the plants without damaging them, while simultaneously optimizing the useful workspace.

The robot's footprint can dynamically vary from 1000 $\times$ 800,mm to 1000 $\times$ 1200,mm. Its lightweight design minimizes both soil compaction and power consumption, while the gantry configuration provides a large, unobstructed central volume between the wheels, leaving ample space for the integration of agricultural end-effectors, terrain monitoring sensors, and state estimation equipment.

Consequently, AgriMaRo was developed strictly following these dimensional and functional specifications, enabling it to operate in the vast majority of the studied greenhouse environments. The choice of this specific architecture allowed maintaining a structure that is simple to assemble, which facilitates the rapid replacement of components in case of damage or wear. The most important aspect of the project, however, remains its modular and multi-functional nature: the robot is indeed capable of performing various agricultural tasks depending on the specific needs of the moment, simply by modifying the track width and replacing the working tool on the end-effector.

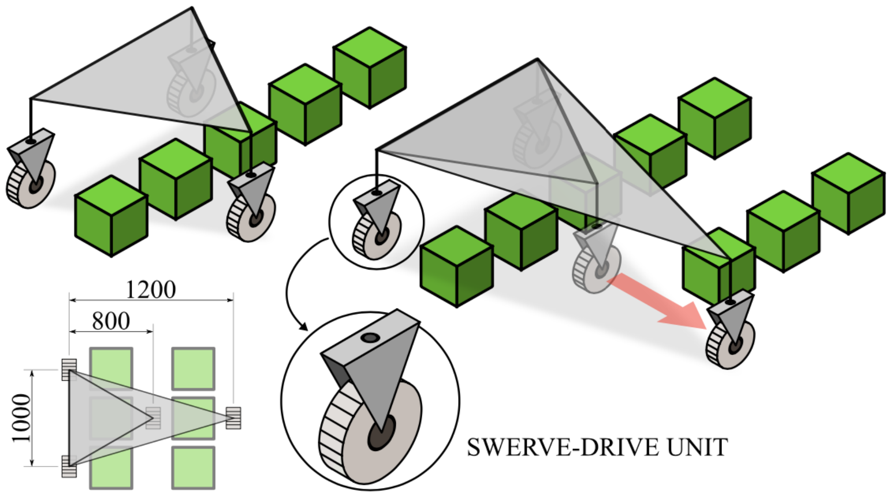

Figure 2.3. Concept of the AgriMaRo robot.

2.2 Locomotion System Design

The locomotion system of AgriMaRo was developed to physically support the robot's structure and, at the same time, transmit the energy necessary for movement to the ground. It must perform several crucial functions for greenhouse operations:

- **Omnidirectionality:** ensuring free movement devoid of directional constraints;
- **Stability:** maintaining the dynamic equilibrium of the vehicle during maneuvers and while overcoming slight obstacles;
- **Robustness:** safely supporting the weight of the entire gantry structure and the operational payload;
- **Adaptability:** allowing the vehicle to perform its work in different ground conditions (compacted soil, mud, obstacles);
- **Control:** providing precise and continuous sensory feedback for odometry and navigation monitoring.

The robot is equipped with three independent locomotion units 2.4, each mainly composed of two subsystems: a drive module (hub-motor) that guarantees translation and a steering module responsible for orientation. The combined action of these groups must ensure the types of movement, speeds, accelerations, and torques necessary for the correct functioning of the entire vehicle.

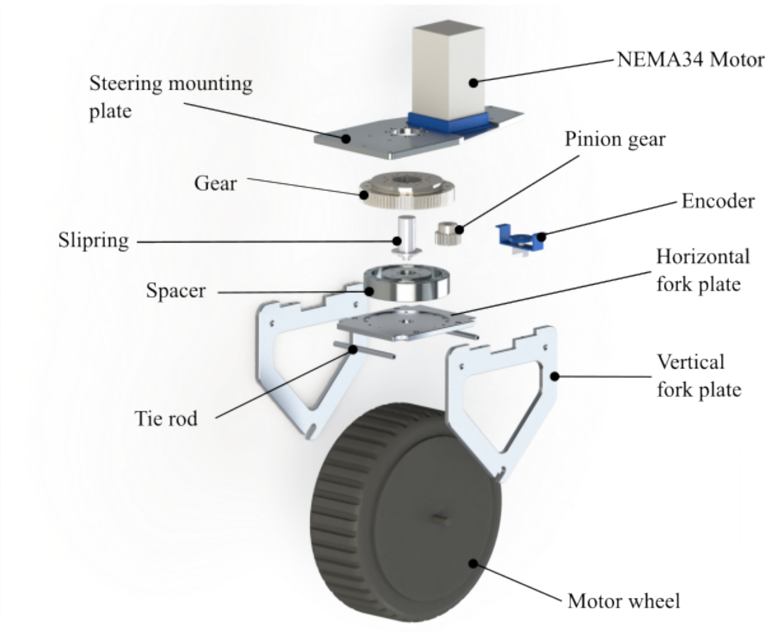

Figure 2.4. Exploded render model of the locomotion unit.

2.2.1 Traction Subsystem Design (Hub-motor)

The design of the traction subsystem based on a hub-motor was chosen to eliminate the mechanical losses typical of traditional transmission systems, simultaneously improving control flexibility (each wheel is completely independent) and the compactness of the wheel assembly. The sizing was carried out considering the acting forces derived from the vehicle's dynamic equations. In particular, the required traction force, and consequently the driving torque, was calculated to reach the desired operational speed in different conditions, such as flat surfaces or inclined planes. This study was conducted by assuming a series of simplifications (justified by the robot's low maximum speed) and adopting standard friction coefficients for agricultural surfaces. Subsequently, to compensate for these analytical approximations and account for electromechanical torque losses, an appropriate safety factor was applied. The calculations revealed a **maximum required torque** per wheel of **22.44 Nm**.

At the end of the design and sizing phase, the choice of the commercial wheel was evaluated with the goal of overestimating the calculated theoretical maximum torque and having a tire suitable for harsh operating conditions. The choice fell on a tire with deep treads, ideal for obtaining high traction on uneven terrain, combined with a large contact patch capable of guaranteeing excellent load distribution on the ground and greater stability. The final selection was a hub-motor produced by **UuMotor** (a model typically used for earth-moving applications such as motorized electric wheelbarrows), powered at 48V, which integrates a BLDC motor and an electromagnetic parking brake, capable of delivering a maximum torque of 60 Nm.

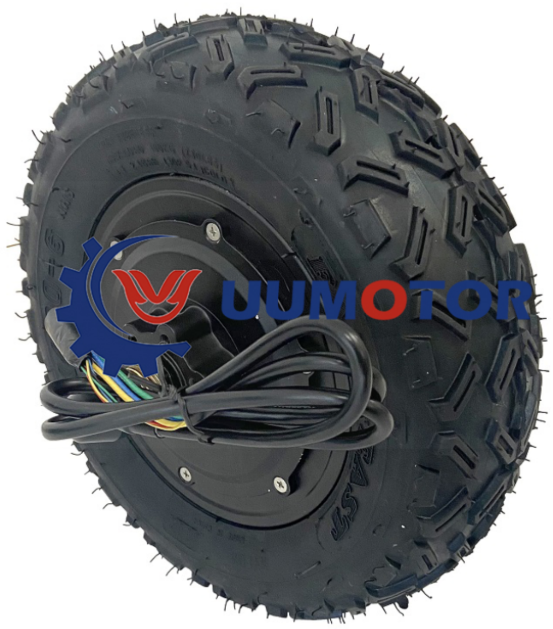

Figure 2.5. Hub-motor UuMotor.

2.2.2 Steering Subsystem Design

Regarding the design of the steering system, the high amount of assumptions necessary for a purely analytical calculation of the resisting torque (due to the complex tire-agricultural soil interaction) required an empirical design approach via experimental tests. A critical aspect analyzed in the preliminary phase was the offset between the vertical steering axis and the contact point of the wheel on the ground. Although a zero offset theoretically reduces the torsional moment necessary for rotation (minimizing sliding friction during stationary steering), a small mechanical trail is often desirable to ensure greater directional stability. However, considering AgriMaRo's low operational speed, the design priority was placed on reducing the resisting torque during static steering maneuvers (scrubbing torque).

A test bench was therefore built to test a prototype of the module in various environments that replicated the real working conditions of the robot (grass surfaces, stone paving, loose soil) and under different load conditions. This testing campaign allowed measuring a required **steering torque**, in both static and dynamic conditions, equal to **18.126 Nm**. Taking into account the steering speed imposed by the kinematic requirements (20 rpm) and applying a conservative overestimation to the experimentally derived torque (setting the design target at 25 Nm), it was decided to implement a torque multiplication mechanism. This system was achieved using a straight-cut spur gear transmission. Specifically, the ring gear consists of a **slewing ring (Igus PRT-04)**, a structural component that allows the two sections of the leg to be decoupled while maintaining adequate resistance to vertical loads and overturning moments, coupled with a pinion driven by a **NEMA 34 stepper motor**.

To ensure infinite steering rotation without angular limits, a slip ring produced by MOLFON was adopted. This component allows the continuous transmission of power and signals from the wheel motor (located on the rotating fork) to the system's control unit on the chassis, preventing the progressive twisting of the cables. Finally, to achieve closed-loop control of the steering angle, an absolute encoder is used. This guarantees that the orientation of each individual wheel is immediately available at startup and perfectly consistent with the kinematic model.

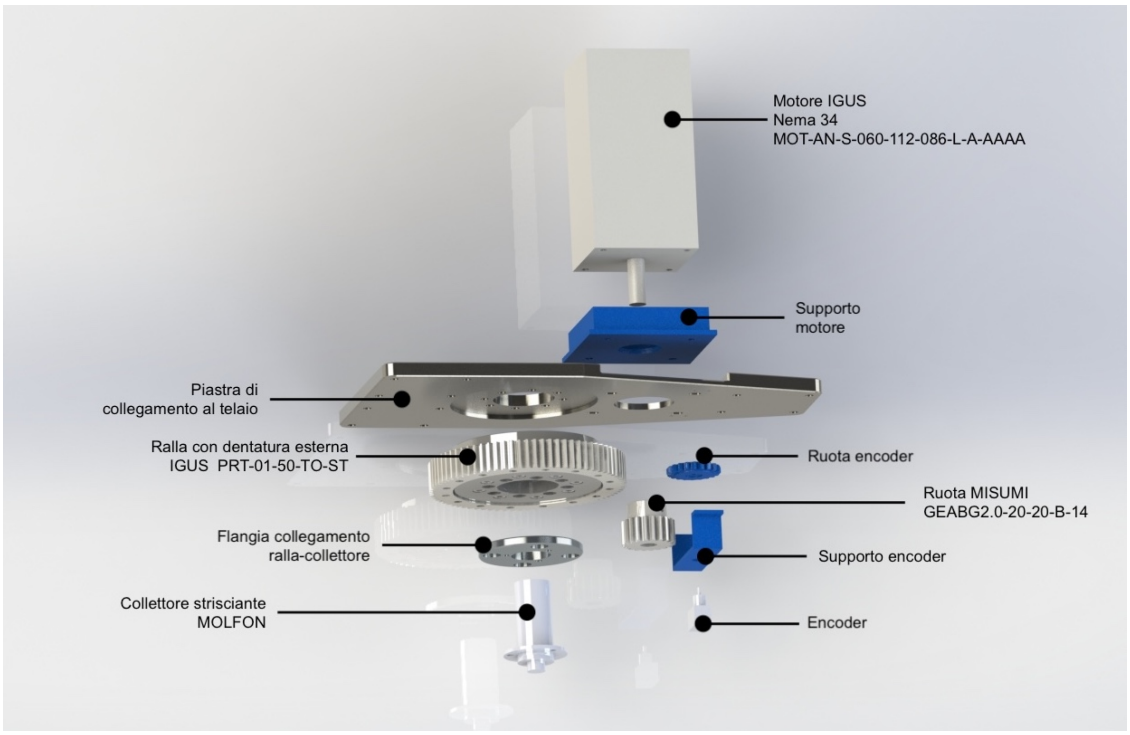

Figure 2.6. Exploded view of the traction system.

2.3 Kinematics and Driving Configurations

This subsection analyzes the design choices underlying AgriMaRo's mobility and introduces the kinematic model of the base platform, originally developed and formalized in the literature by the research group [5]. In both teleoperated and autonomous modes, the kinematic model is essential for controlling the robot's movements. In particular, **inverse kinematics** is necessary to calculate the reference values for the degrees of actuation once the desired velocities are known, while **forward kinematics** is fundamental for computing odometry starting from the current state of the actuators.

From a functional point of view, the base of the robot can be modeled as a rigid body suspended on three independent driving units, known in the literature as **Swerve Drive Units**, which allow the orientation of the wheel to be decoupled from its rotation speed. More precisely, from a kinematic standpoint, AgriMaRo is classified as a **pseudo-omnidirectional** mobile robot. Unlike systems with pure omnidirectional or mecanum wheels, the standard steerable wheel configuration requires physical time to orient the thrust modules before executing an instantaneous change of direction, while maintaining the ability to reach any position and orientation configuration in the plane. This approach guarantees optimal traction, greater payload capacity, and constant contact with the ground, significantly improving maneuverability in tight spaces.

2.3.1 Kinematic Model

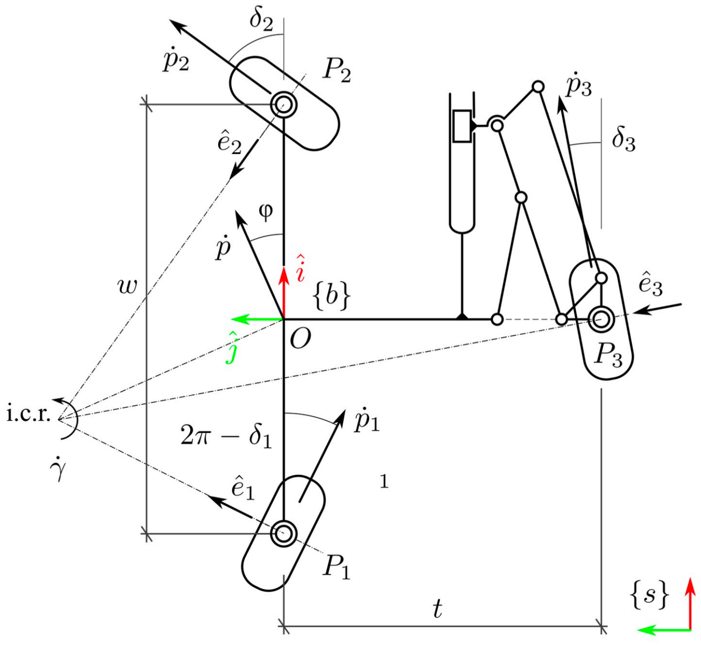

Figure 2.7. AgriMaRo kinematic model.

To describe the system mathematically, let us define the state vector for each swerve-drive unit as $\mathbf{q}_i = [\delta_i, \omega_i]^T$, where δ_i represents the steering angle and ω_i is the angular velocity of the $i-th$ wheel. Consequently, the overall command vector of the robot is $\mathbf{q} = [\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3]^T$. Assuming a strictly planar motion, i.e., hypothesizing zero translation along the vertical axis $\hat{k}$ and blocked rotations around the axes $\hat{i}$ and $\hat{j}$, the movement of the robot chassis can be defined via the spatial velocity Twist vector $\mathbf{V}_b = [\dot{\gamma}_b, \dot{x}_b, \dot{y}_b]^T$, expressed within the reference system attached to the vehicle body.

By imposing a strict pure rolling condition between the $i-th$ wheel and the ground, the velocity of the contact point P_i is expressed vectorially as follows:

$$\dot{\mathbf{p}}_i = \boldsymbol{\omega}_i \times \mathbf{d_i} = r \begin{bmatrix} \cos\delta_i \\ \sin\delta_i \end{bmatrix} \omega_i = \mathbf{s}_i(\delta_i)\omega_i \tag{2.1}$$

Where:

- $\mathbf{d_i}$ is the directional vector from the wheel-ground contact point to the wheel center $r\hat{\mathbf{w}}$.
- r indicates the wheel radius, assumed identical for all driving units;

Simultaneously, by virtue of rigid body kinematics, the velocity of the same point P_i can be calculated starting from the spatial Twist $\mathbf{V}_b$ of the chassis:

$$\dot{\mathbf{p}}_i = \dot{\mathbf{p}}_i + \dot{\gamma}\hat{\mathbf{w}} \times \mathbf{p}_{P_i} = \begin{bmatrix} \dot{x}_b - y_i\dot{\gamma}_b \\ \dot{y}_b + x_i\dot{\gamma}_b \end{bmatrix} = \begin{bmatrix} -y_i & 1 & 0 \\ x_i & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\gamma}_b \\ \dot{x}_b \\ \dot{y}_b \end{bmatrix} = \mathbf{h}_i\mathbf{V}_b \tag{2.2}$$

where $\mathbf{p}_{P_i} = [x_i, y_i]^T$ identifies the position vector of the attachment point of the $i-th$ unit with respect to the mobile reference frame.

Adapting these equations to the specific layout of AgriMaRo, the global matrix relationship linking the state of the actuators to the Twist of the robot takes this form:

$$\begin{bmatrix} \mathbf{s}_1(\delta_1) \\ \mathbf{s}_2(\delta_2) \\ \mathbf{s}_3(\delta_3) \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ w/2 & 0 & 1 \\ 0 & 1 & 0 \\ -w/2 & 0 & 1 \\ t & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\gamma}_b \\ \dot{x}_b \\ \dot{y}_b \end{bmatrix} \iff \mathbf{S}(\delta)\boldsymbol{\omega} = \mathbf{H}\mathbf{V}_b \tag{2.3}$$

Analyzing the equation, it emerges that the matrix $\mathbf{H}$ depends only on the fixed geometry of the vehicle with respect to the mobile reference system (where w is the distance between the wheels on the same side and t is the track width). Conversely, the matrix $\mathbf{S}(\delta)$ is a function of the current steering state. Given that the robot moves in a plane (3 degrees of freedom) but has six actuation variables, kinematic constraints are present: in order not to violate the rigid body and pure rolling hypotheses, the actuation variables are forced to reside on a specific three-dimensional surface within their six-dimensional space.

- **Forward Kinematics:** The matrix mapping just derived allows solving the forward kinematics directly, which is fundamental for odometric purposes, by resorting to the Moore-Penrose pseudo-inverse matrix $\mathbf{H}^{\dagger}$:

$$\mathbf{V}_b = \mathbf{H}^{\dagger}\mathbf{S}(\delta)\boldsymbol{\omega} \tag{2.4}$$

  Therefore, given the actuation variables δ and ω,chosen according to the kinematic constraint, it is possible to compute the velocity twist of the system.

- **Inverse Kinematics:** The solution to the inverse problem does not enjoy the same mathematical immediacy, since the matrix $\mathbf{S}$ is tied to the current configuration. A practical resolution strategy consists of uncoupling the problem by evaluating the individual locomotion units independently, first calculating the norm of the wheel velocity vector $||\dot{\mathbf{p}}_i|| = \sqrt{\dot{\mathbf{p}}_i^T \dot{\mathbf{p}}_i}$. Consequently, the actuation variables for the $i-th$ unit are derived as:

$$\begin{cases} \delta_i = \operatorname{atan2}(\dot{y}_i/||\dot{\mathbf{p}}_i||, \dot{x}_i/||\dot{\mathbf{p}}_i||) + k\pi, \quad k \in \mathbb{Z} \\ \dot{\theta}_i = (-1)^k ||\dot{\mathbf{p}}_i||/r \end{cases} \tag{2.5}$$

  As evidenced by the presence of the integer constant k, two mathematical solutions are generated for each module, a result perfectly in accordance with the physics of the system: the velocity imposed at the center of the wheel does not change if the module undergoes a 180° rotation accompanied by a reversal of the rotation direction.

2.3.2 Operational Space and Hardware Constraints

The operation of a real robotic system is inevitably subject to **physical actuation limits**. Having equipped AgriMaRo with slip rings that guarantee infinite steering rotation, the main constraint limiting the vehicle's performance is dictated by the maximum rotation speed that the hub-motors can deliver (ω_{max}).

Recalling the kinematic equations, the norm of the linear velocity for the $i-th$ wheel $||\dot{\mathbf{p}}_i||$ always admits two possible solutions (related to the possibility of rotating the module by 180° and reversing the traction direction):

$$\frac{\dot{\mathbf{p}}_i}{\omega_{max}r} = \pm\sqrt{\left(\frac{\dot{x}_b}{\omega_{max}r} - y_i\frac{\dot{\gamma}_b}{\omega_{max}r}\right)^2 + \left(\frac{\dot{y}_b}{\omega_{max}r} - x_i\frac{\dot{\gamma}_b}{\omega_{max}r}\right)^2} \tag{2.6}$$

Applying this formulation to the specific geometry of the rover, the velocity system expands into:

$$\Omega_1 = \frac{\dot{\mathbf{p}}_1}{\omega_{max}r} = \pm\sqrt{\left(\frac{\dot{x}_b}{\omega_{max}r}\right)^2 + \left(\frac{\dot{y}_b}{\omega_{max}r} - \frac{w}{2}\frac{\dot{\gamma}_b}{\omega_{max}r}\right)^2} \tag{2.7}$$

$$\Omega_2 = \frac{\dot{\mathbf{p}}_2}{\omega_{max}r} = \pm\sqrt{\left(\frac{\dot{x}_b}{\omega_{max}r}\right)^2 + \left(\frac{\dot{y}_b}{\omega_{max}r} - \frac{w}{2}\frac{\dot{\gamma}_b}{\omega_{max}r}\right)^2} \tag{2.8}$$

$$\Omega_3 = \frac{\dot{\mathbf{p}}_3}{\omega_{max}r} = \pm\sqrt{\left(\frac{\dot{x}_b}{\omega_{max}r} - t\frac{\dot{\gamma}_b}{\omega_{max}r}\right)^2 + \left(\frac{\dot{y}_b}{\omega_{max}r}\right)^2} \tag{2.9}$$

To analytically introduce the upper limit imposed by the motors, these quantities are normalized with respect to $\omega_{max}r$, imposing the saturation condition:

$$||\Omega_i|| < 1 \tag{2.10}$$

In order to evaluate the impact of these limits on the overall behavior, it is useful to define a dimensionless Twist space. For this purpose, the chassis velocities are normalized and the dimensionless geometric parameter ξ is introduced, defined as the ratio between the track width and the lateral half-distance ($a = w/2$):

$$\dot{\Gamma}_b = a\frac{\dot{\gamma}_b}{\omega_{max}r}, \quad \dot{X} = \frac{\dot{x}_b}{\omega_{max}r}, \quad \dot{Y} = \frac{\dot{y}_b}{\omega_{max}r} \tag{2.11}$$

The intersection of these saturation constraints generates a 3D convex polyhedron that represents the envelope of all admissible Twists for the AgriMaRo platform. From the theoretical analysis of this spatial domain, the influence of the variable track width t clearly emerges:

- Under conditions of **pure translation** (absence of rotations), the vehicle is capable of expressing 100% of the speed allowed by the motors.

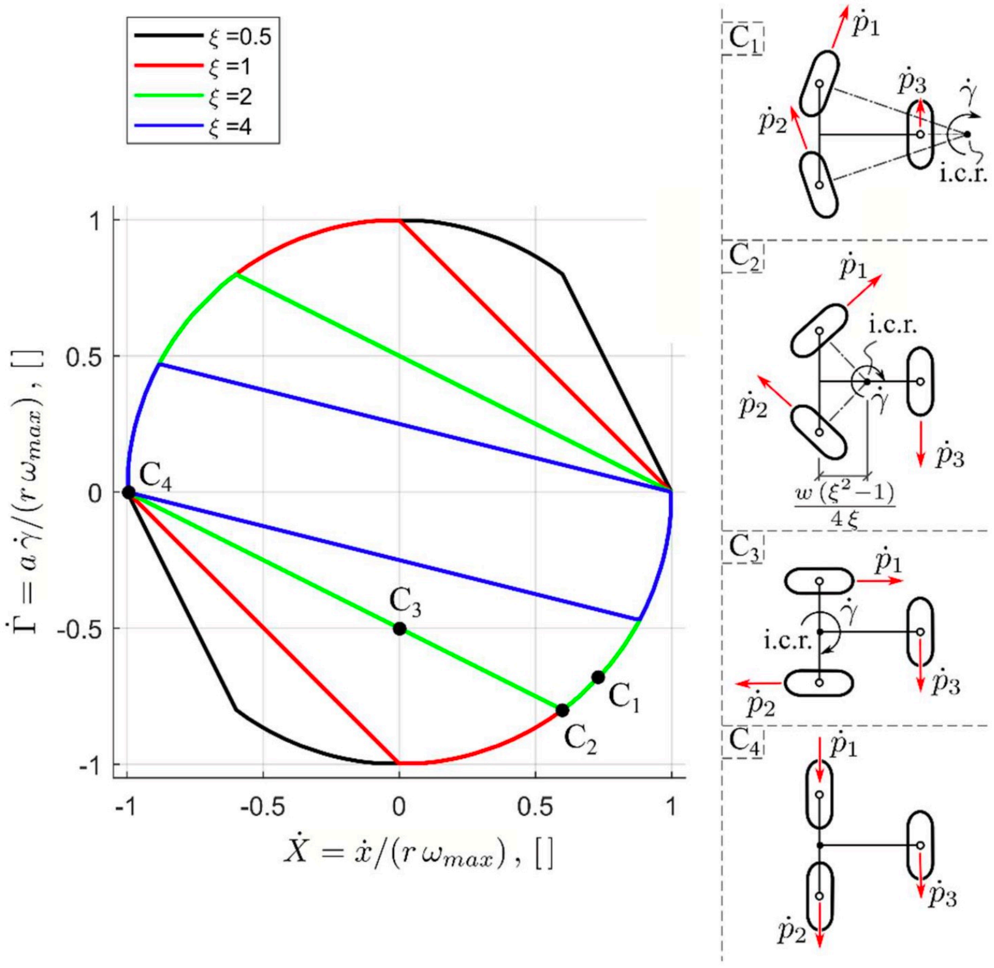

Figure 2.8. Admissible body twist regions with $\dot{Y} = 0$ and different values of ξ.

- In the presence of **yaw rates** (e.g., tight turns or spot turns), the overall operational limit is drastically reduced. In these maneuvers, the wheel located at the greatest geometric distance from the Instantaneous Center of Rotation (frequently the third outer wheel when the robot is in the extended configuration) saturates its maximum speed early. Consequently, it acts as an insurmountable kinematic limit for the entire chassis: for example, in a pure spot turn motion, the dimensionless yaw rate decreases proportionally to the increase in the track width.

Chapter 3

Sensor Suite Characterization and Data Acquisition

This chapter describes the sensory architecture of the AgriMaRo robot, analyzing the solutions adopted for data acquisition necessary to study the vehicle's dynamics. At the beginning of this thesis work, the platform featured a **limited sensor suite**, used exclusively for low-level control functions. The purpose of this section is therefore to define the baseline sensory setup, highlighting the connection between the acquired electrical signals and the kinematic variables required for the robot's modeling.

The accuracy of the dynamic analyses conducted in the following chapters depends directly on the fidelity of these measurements. Therefore, the subsequent sections will detail the installation criteria, the physical signal conversion models, and the calibration procedures implemented to correct systematic biases in the transducers.

3.1 Sensor Positioning

The arrangement of transducers on the AgriMaRo platform was defined seeking an optimal compromise between protecting electronic components and the need to obtain measurements faithful to the vehicle's dynamics.The sensors analyzed and integrated for the purposes of this thesis include:

- **Rotary encoders** for monitoring the steering angle;

- **RPM sensors** for measuring the angular velocity of the wheels;
- **Inertial Measurement Unit (IMU)** for detecting Euler angles, accelerations and angular velocities along the three axes.

3.1.1 Rotary Sensors (Encoders and RPM Sensors)

The steering encoders and the traction RPM sensors are both located within the wheel assemblies, but their integration with the mechanical transmission follows two distinct approaches.

In the steering system, the encoder is installed at the end of the mechanical transmission line. It connects to the **slewing ring** through a dedicated transmission stage designed to guarantee a strict 1:1 gear ratio. Maintaining this unitary ratio is essential to prevent measurement drift and multi-turn offset issues. This requirement is directly linked to the platform's omnidirectionality, which demands continuous and unlimited rotations of the wheel assemblies. To avoid mechanical stress and cable entanglement during these infinite rotations, the system incorporates a slip ring (or rotary electrical joint), ensuring the uninterrupted transmission of signals and power between the fixed chassis and the rotating module. Thanks to the 1:1 mechanical coupling, the steering encoder directly and accurately reads the true steering angle of the wheel, eliminating the need for mathematical reductions.

In contrast, the traction RPM sensors are integrated directly into the motor drivers and measure the rotational speed of the motor shaft before the mechanical gear reduction. Being located upstream of the gearbox, the acquired raw data does not directly represent the final ground speed. As a result, the traction measurements require an analytical conversion that factors in the reduction ratio to accurately estimate the actual kinematic variables of the robot.

3.1.2 Inertial Measurement Unit (IMU)

The positioning of the IMU has undergone a significant evolution compared to the platform's original design. Initially, the sensor was integrated directly on the main electronic board (PCB) of the robot for cabling simplicity.

However, to optimize the quality of the state estimation, the IMU was subsequently moved to a dedicated position, aligned with the geometric center of the

robot's reference system. This repositioning was dictated by specific technical requirements which will be detailed and analyzed in the remainder of the chapter.

3.2 Sensor Operating Principles and Kinematic Modeling

In order to evaluate the technological limitations and clearly define how transducers convert physical quantities into digital signals, the following sections will illustrate the operating principle of the components used and the mathematical model for interpreting the acquired data.

3.2.1 Rotary Sensors (Encoder and RPM)

To monitor the kinematic state of the wheels, the project employs two distinct data acquisition solutions. The requirements for controlling steering and traction are indeed structurally different: therefore, absolute encoders are used to obtain the steering angle, while Hall-effect sensors are used to measure the motor RPM.

The orientation of the steering modules is detected using absolute encoders. The primary advantage over classic incremental systems lies in the absence of mechanical zeroing (homing) procedures. The true position of the module is known the exact moment the machine is powered on, without the need to integrate pulses over time.

From a construction standpoint, this relies on an optoelectronic architecture. A coded disk is rigidly coupled to the steering shaft, featuring several concentric circular tracks. Each of these tracks alternates between transparent and opaque sectors. Physical reading occurs by placing the disk between a light emitter (typically an LED array) and a bank of light detector (phototransistor). By reading the state of all tracks in parallel, where light passes and where it is blocked, the sensor outputs a multi-bit digital string.

In practice, however, the disk mask does not use standard pure binary code, but implements Gray Code [21]. This is a mandatory engineering choice due to mechanical tolerances. If standard binary were used, moving from one angular sector to the next would often mean switching multiple bits simultaneously (for instance, from 011 to 100). Unfortunately, optical sensors are never perfectly aligned down to the micrometer. Consequently, the phototransistor would register state changes with slight timing offsets, generating a completely false angular value for a fraction of a second. This produces what are technically called "glitches" in the trajectory control.

Gray code bypasses this obstacle structurally. The mathematics behind this code ensures that between two adjacent positions, always and only one single bit changes at a time. If the light beam falls exactly on the boundary between two sectors, the sensor's uncertainty will only affect a single bit. The maximum error is therefore reduced to the instrument's basic resolution step, providing the control system with an extremely clean and reliable reading of δ.

Moving on to the reading of the traction angular velocity (ω), the hardware changes completely. There are no external toothed wheels or optical encoders. Instead, the system directly exploits the electromechanical nature of the BLDC (Brushless DC) motors mounted in AgriMaRo's hub-motors.

In these machines, the rotor (the rotating outer casing) is internally lined with a series of permanent magnets, arranged by alternating North and South poles. The RPM calculation is managed by the power driver. The latter receives signals from

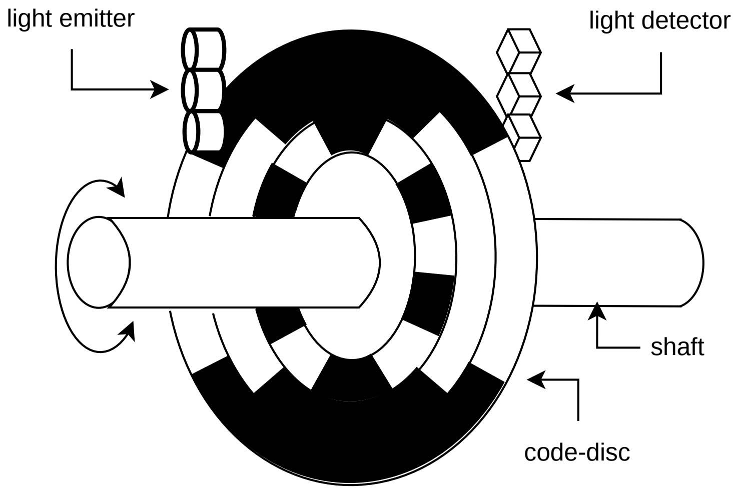

Figure 3.1. Operating principle of an optical absolute rotary encoder with a Gray-coded disk.

dedicated sensors fixed on the static part of the motor (the stator), which detect the continuous variations of the magnetic field during the wheel's rotation.

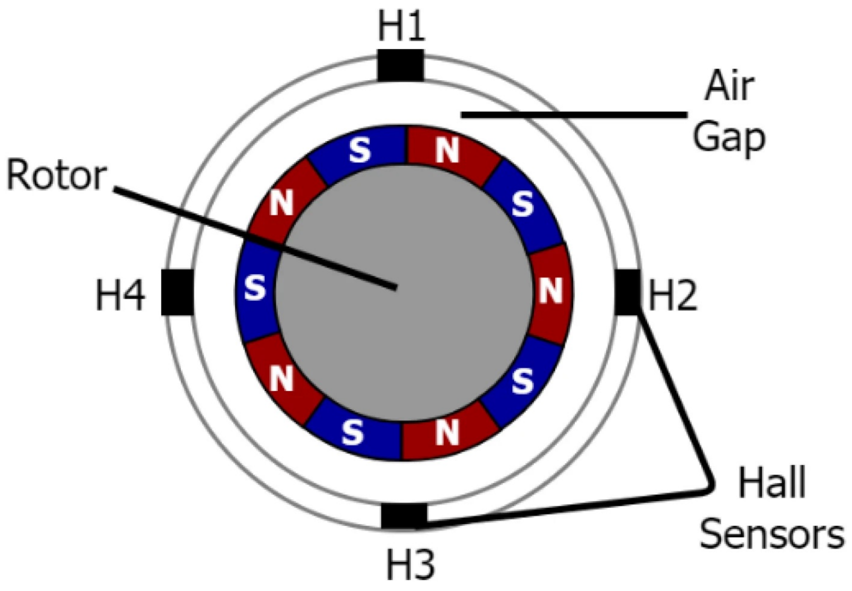

Figure 3.2. Schematic of Hall-effect rotary encoder [1].

The wheel RPM sensors are based on the **Hall effect**, which describes the interaction between a magnetic induction field and electrically charged particles in motion. A particle carrying an electrical charge q and moving in a magnetic induction field $\mathbf{B}$ with a speed vector $\mathbf{v}$ is subjected to the Lorentz force:

$$\mathbf{F}_{Lorentz} = q(\mathbf{v} \times \mathbf{B}) \tag{3.1}$$

In a conductor, this interaction can be expressed as:

$$\mathbf{F}_{Lorentz} = i(\mathbf{l} \times \mathbf{B}) \tag{3.2}$$

where:

- l is the length of the conductor,
- i is the electric current flow.

By positioning the magnetic field orthogonally to the current flow, the resulting force deflects the flow of charges in the z direction (across the thickness of the chip).

$$\mathbf{F}_{Lorentz} = Bli\hat{z} \tag{3.3}$$

As these charges accumulate on one side of the chip, an electrostatic repulsion force $\vec{F}_{el}$ is generated, leading to an equilibrium state:

$$\mathbf{F}_{el} = Q\mathbf{E} = Q\frac{V}{w}\hat{z} \tag{3.4}$$

where:

- Q is the flowing charge,
- $\mathbf{E}$ is the electrostatic field,
- w is the chip width.

At equilibrium, it is possible to calculate the **induced Hall Voltage** (V):

$$V = \frac{Bwli}{Q} = \frac{BSi}{Q} \tag{3.5}$$

where $S = w \cdot l$ represents the surface area of the chip.This relationship highlights that the Hall voltage is inversely proportional to the thickness d of the sensing element, a critical parameter for sensor sensitivity."

The sensor utilizes this principle by placing a slice of conductive material within a U-shaped support, oriented orthogonally to the field lines generated by a permanent magnet. A constant current i is forced through this material. The transit of a ferromagnetic tooth (or rod) between the ribs of the U-shaped element alters the magnetic induction field $\mathbf{B}$ orthogonal to the chip. This change in the magnetic field causes the Hall voltage to vary depending on whether a tooth is present or not. The received signal allows the system to accurately measure the wheel's position and angular velocity.

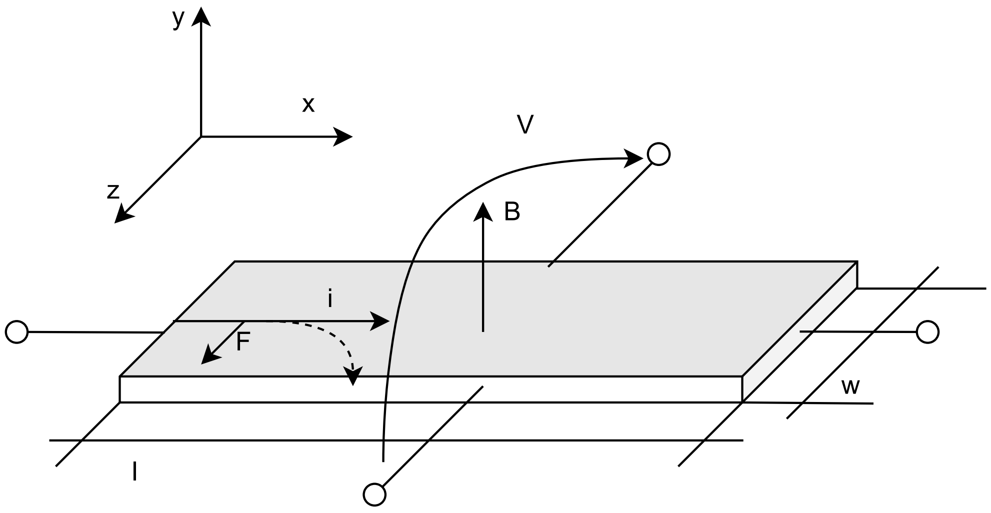

Figure 3.3. Hall Effecte.

3.2.2 Inertial Measurement Unit (IMU)

Another essential component of the AgriMaRo sensory system is the **BNO055 Inertial Measurement Unit**. This is a 9-axis (9-DOF) sensor, meaning it integrates three accelerometers, three gyroscopes, and three magnetometers positioned along the X, Y, and Z axes. This configuration allows for the evaluation of linear accelerations, angular velocities, and magnetic fields relative to the three axes. By processing these data, it is possible to determine the IMU's **orientation** in space, the vehicle's dynamic **accelerations**, and its **rotations**.

The **accelerometer** utilizes **Micro-Electro-Mechanical Systems (MEMS)** technology [6]. The sensor is composed of two miniaturized metal plates: one fixed to the reference frame and the other suspended by an elastic support with a rigidity constant k_{el}. The electrical capacitance C between these two plates is defined by:

$$C = \frac{\varepsilon S}{x} \tag{3.6}$$

where:

- ε is the permittivity of the dielectric,

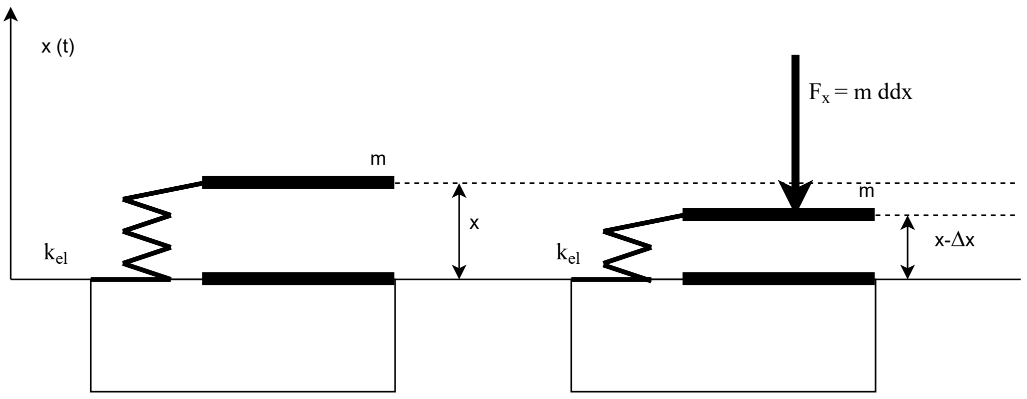

Figure 3.4. MEMS accelerometer operating principle.

- S is the surface area of the plates,
- x is their separation distance.

When the system undergoes an acceleration $\ddot{x}$ in the x direction, the moving plate (with mass m) is subjected to an inertial force $F_x = m \cdot \ddot{x}$, causing a displacement Δx:

$$\Delta x = \frac{F_x}{k_{el}} = \frac{m \cdot \ddot{x}}{k_{el}} \tag{3.7}$$

This displacement generates a change in electrical capacitance ΔC:

$$\Delta C = \varepsilon \frac{S}{x - \Delta x} - \varepsilon \frac{S}{x} = \varepsilon \frac{S \Delta x}{x(x - \Delta x)} \cong \frac{\Delta x}{x} C = \frac{m \cdot \ddot{x}}{x \cdot k_{el}} C \tag{3.8}$$

This variation is proportional to the acceleration. Using a capacitive bridge structure, this change is converted into a voltage variation $v_d(t)$, enabling the digital controller to measure the acceleration.

The **gyroscope** exploits the **Coriolis effect** [6]. This system consists of a mass oscillating at a specific frequency in a direction perpendicular to the axis of the angular velocity to be measured. In the presence of an angular velocity $\mathbf{\Omega}$, the mass is subjected to a Coriolis force in the direction perpendicular to both the velocity vector of the mass and the angular velocity vector:

$$\mathbf{F}_c = -2m(\mathbf{v} \times \mathbf{\Omega}) \tag{3.9}$$

where:

- m is the mass,
- $\mathbf{v}$ is its oscillating velocity.

By connecting this mass to a MEMS sensing element with an evaluation direction parallel to the Coriolis force, it is possible to measure the force as a function of the capacitance variation:

$$\Delta C \cong \frac{\Delta y}{y} C \propto \Omega \tag{3.10}$$

This allows the system to determine the angular velocity in the direction perpendicular to the mass oscillation.

The **magnetometer** utilizes the **Hall effect**, as previously described for the RPM sensors subsection 3.2.1. It detects the potential difference generated in a conductor carrying a current when subjected to a magnetic field perpendicular to the current flow. This allows the sensor to recognize the IMU's orientation relative to the Magnetic North, acting as a digital compass to provide an absolute heading reference.

3.3 Sensor Calibration

This section details the calibration procedures implemented to ensure the reliability and accuracy of the acquired data. These steps became essential as the platform transitioned from simple low-level control, where high precision was not a primary concern, to a more complex analysis of vehicle dynamics. Furthermore, recent hardware modifications and the integration of new components, such as the IMU (which was previously unutilized as it provided no support for low-level functions), necessitated a rigorous re-evaluation of all measurement scales.

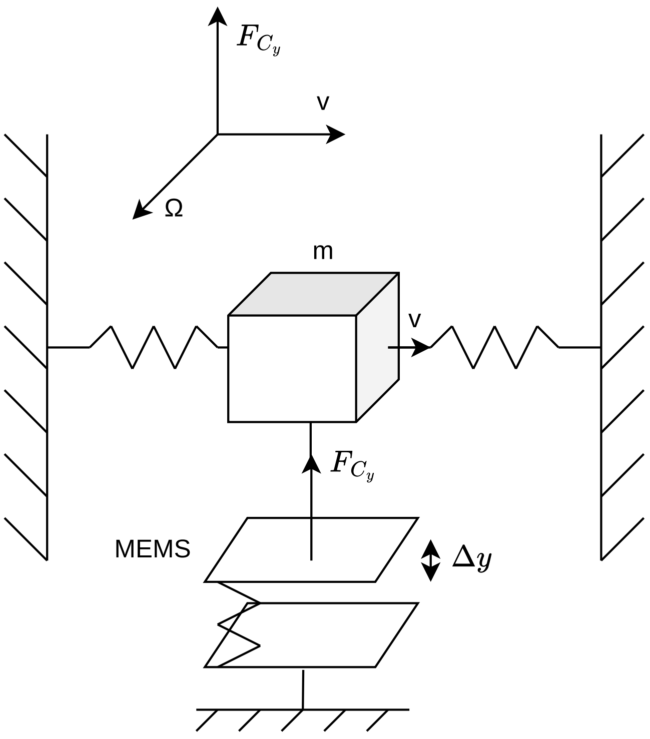

Figure 3.5. Angular velocity sensor operation principle.

3.3.1 Steering Encoders

Although the steering encoders were already part of the existing hardware, their measurements were found to be insufficiently accurate for high-level dynamic analysis. The primary source of error was an incorrect **initial offset** within the AgriMaRo control code, which caused the steering to initialize at a non-zero position even when the wheels were physically straight. This resulted in a constant bias in the steering angle evaluation.

This parameter must be recalibrated whenever there is a loss of mechanical contact between the slewing ring and the encoder's toothed wheel, such as during maintenance or disassembly. The calibration procedure involved manually positioning the wheels to the true zero configuration ($\delta_{i_0} = 0$), recording the raw value measured by the encoder, and storing this data as an initialization constant. This value is then subtracted from the raw reading in the software to obtain the actual steering angle.

3.3.2 RPM Sensors

Regarding the RPM sensors, the main discrepancy arose from an incorrect evaluation of the **transmission ratio (τ)**. Since the motorized wheel's datasheet lacked clear technical specifications, an experimental validation was required.

To correct this, tests were conducted to establish the relationship between the angular distance traveled by the wheel over a specific time and the velocity calculated by the RPM sensor. This empirical approach allowed for the determination of the actual reduction ratio, ensuring that the angular velocity data used for subsequent slip analysis is both accurate and physically consistent.

3.3.3 IMU Calibration and Integration (BNO055)

The integration phase of the inertial unit required a thorough analysis, beginning with a series of functional tests aimed at verifying the consistency of the reported data with the real dynamics of the vehicle. Initially, the robot was subjected to maneuvers with known accelerations and rotations, comparing the physical inputs with the output provided by the sensor. From this analysis, it immediately emerged that the BNO055, configured in **Sensor Fusion mode**, reported data significantly different from the input values, making the sensor unusable for a reliable state estimation.

The incorrect operation of the system was attributed to the intrinsic limitations of the automatic self-calibration procedure provided by the original firmware. This procedure activates at every startup of the device; however, it was observed that:

- **Accelerometer and Gyroscope:** they were calibrated while the robot was already subject to mechanical vibrations, affecting the calculation of static biases.

- **Magnetometer:**the static self-calibration prevented the sensor from mapping the entire necessary spatial range. Without performing full rotations, the sensor could not correctly distinguish the Earth's magnetic field from local interference.

To overcome these limits, it was decided to proceed with a "one-time" **manual calibration** to generate configuration profiles to be permanently saved in the sensor's memory. For this purpose, an external test circuit was assembled consisting of the IMU and an Arduino Nano connected to a PC, allowing the calibration status to be monitored in real-time through the sensor's internal registers (values from 0 to 3 for each axis). The procedure followed three distinct phases:

- **Gyroscope:** the sensor was kept immobile for several seconds until calibration level 3 was reached.
- **Accelerometer:**the device was placed in at least six different static orientations for a couple of seconds each.
- **Magnetometer:** slow and wide "figure-8" movements were performed, ensuring that the sensor perceived the Earth's magnetic field in all spatial directions.

At the end of the procedure, with all parameters (System, Gyroscope, Accelerometer, Magnetometer) at level 3, an additional verification was carried out by comparing the measured gravitational acceleration with the theoretical value to confirm the accuracy of the data. These calibration values were then saved within the IMU, ensuring that at startup the sensor loads the pre-set parameters without being affected by the vibrations or initial static state of the robot. It should be noted, however, that this solution assumes a relatively constant magnetic and thermal environment, as variations in temperature, humidity, electrical noise, and other factors can still introduce small drifts.

After the reintegration of the calibrated sensor into the vehicle, new problems emerged regarding the accuracy of the orientation (handling). The Sensor Fusion mode uses magnetometer data to obtain absolute orientation relative to Magnetic North. In a complex environment like that of the AgriMaRo robot, the presence of power cables carrying high currents to the motors generates **time-varying magnetic fields**. Initially, the IMU was positioned on the **Printed Circuit Board (PCB)**, in extreme proximity to these high-current cables. The resulting sudden changes in the local magnetic field made the orientation data unstable and unusable. To mitigate this issue, the IMU was relocated to a position as far as possible from the motors and their wiring. However, even in this new location, the magnetometer continued to experience interference. This was caused by fluctuating magnetic fields induced by the variable return currents flowing through the metal tubes of the robot frame, which serves as the system's electrical ground. To permanently resolve the data instability, it was decided to configure the BNO055 in **IMUPLUS mode**. As indicated in the datasheet, this mode excludes the magnetometer, basing data acquisition exclusively on the accelerometer and gyroscope. Although this choice resolves the problem of sudden jumps in orientation due to electrical interference, it introduces a technical limitation: the angular acceleration calculation system is subject to a slight temporal **drift** that cannot be corrected, leading to a degradation over time of the Euler angles and angular velocities.

In conclusion, it was decided to maintain the IMU in this strategic position because it represents the origin of the **robot's reference frame**. This positioning is of fundamental importance: it allows the robot's accelerations to be acquired directly from the IMU without using the spatial transposition that was necessary in the previous positioning on the PCB, greatly simplifying the subsequent calculations for the dynamic analysis.

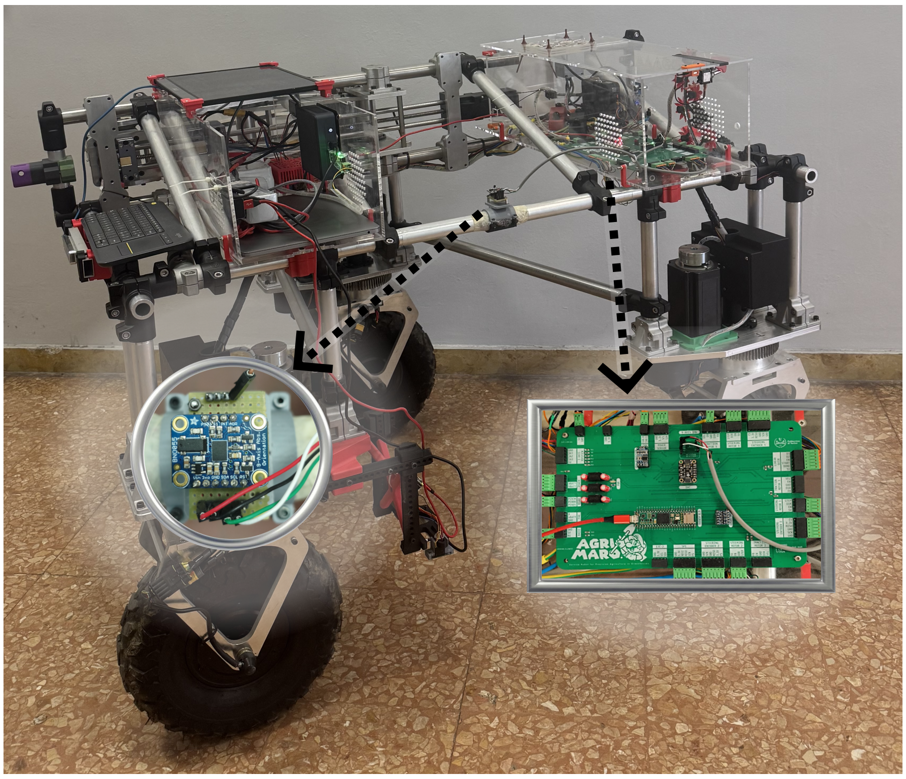

Figure 3.6. IMU Shift: From PCB to Robot Reference Frame.

Chapter 4

Extended Kalman Filter

This chapter analyzes the development of a system aimed at optimizing the acquisition and integration of data from the various sensors installed on AgriMaRo. The objective is twofold: to improve the precision of existing measurements through sensor fusion and, simultaneously, to estimate kinematic variables that are not directly measurable, in order to obtain a representation of the robot's state that is as faithful as possible to its actual dynamic behavior during operation.

4.1 Introduction to the Extended Kalman Filter

To implement this estimation strategy, an Extended Kalman Filter (EKF) was chosen. The selection of this variant over the standard Kalman Filter (KF) is dictated by the inherently non-linear nature of the system governing AgriMaRo. While the classical KF operates correctly only on linear models, the equations describing the kinematics and dynamics of an omnidirectional robot (which include trigonometric functions and wheel-ground contact models) exhibit **significant non-linearities**. The EKF addresses this issue through a local linearization of the model [18], calculated at each time step via Jacobian matrices, thus allowing for a reliable estimation of the states even though it does not guarantee the absolute optimality characteristic of purely linear systems. The Extended Kalman Filter operates according to a recursive cycle based on a logical framework divided into two main phases: **prediction** and **correction** [22].

- **Prediction Phase**: In this first stage, the filter uses a mathematical model (known as the **state-space model**) to estimate the future state of the robot based on its previous condition. Essentially, the filter "forecasts" where the robot should be or at what speed it should be moving. Simultaneously, an **observation model** predicts which sensor measurements should be detected if the robot were actually in that predicted state.

- **Correction Phase**: In this stage, the filter compares the actual measurements from the sensors with those mathematically predicted. This difference is used to calculate the so-called **Kalman Gain.**

The Kalman Gain is the key element of the system: it represents the degree of trust that the filter places in the mathematical model versus the actual measurements. If the sensors are noisy or imprecise, the filter will give more weight to the model's prediction; if, instead, the measurements are reliable, the filter will use them to "correct" the initial estimate. This mechanism allows for the optimal combination of theoretical information and real-world data, resulting in a state estimation that is much more accurate than what would be obtained using a single sensor or the physical model alone.

4.1.1 Prediction Phase

Moving into the analytical details, the first phase of the filter concerns prediction. The starting point is the definition of the **discrete-time non-linear system** through the state and observation equations:

$$X_{k+1} = f\left(X_k\right) + \omega_k \tag{4.1}$$

$$Z_k = h\left(X_k\right) + \upsilon_k \tag{4.2}$$

Where:

- $X_k \in \mathbb{R}^n$ is the **state vector** describing the system at time step k.
- $Z_k \in \mathbb{R}^m$ represents the **measurement vector** at time step k.
- f and h are non-linear functions defining the state transition model and the observation model, respectively.

- ω_k and υ_K are random vectors representing the **process noise** and **measurement noise**. They are assumed to be zero-mean additive white Gaussian noise (**AWGN**) processes:
  - $\omega_k \sim \mathcal{N}(0, Q_k)$
  - $\upsilon_k \sim \mathcal{N}(0, R_k)$

The matrices Q_k and R_k are fundamental parameters that define the stochastic behavior of the filter, acting as statistical weights:

- **Process Noise Covariance Matrix (Q_k):** This matrix models the **uncertainty related to the mathematical model** of the system. As highlighted by Welch and Bishop [28], it is extremely rare for a deterministic model to perfectly capture the dynamics of a real-world system; Q_k quantifies discrepancies caused by unmodeled phenomena, time discretization errors, or unpredictable environmental disturbances

- **Measurement Noise Covariance Matrix (R_k):** This represents the intrinsic **uncertainty of the on-board instrumentation**. Each sensor is characterized by white Gaussian noise, whose variance is typically derived from the hardware data sheets. A high value in R_k indicates significant noise, leading the filter to rely more on the mathematical prediction rather than the sensor data.

The **state-space model**, a system of functions that estimates the state evolution through physical models, allows for the computation of the a priori state estimate at time $k+1$ based solely on the knowledge of the estimated state at the previous time step k:

$$\text{state-space model: } \hat{X}_{k|k-1} = f(\hat{X}_{k-1}) \tag{4.3}$$

Where:

- $\hat{X}_{k-1}$ is the previous state
- $\hat{X}_{k|k-1}$ is the state predicted at time k given the state at the previous instant $k-1$.
- $f(\hat{X}_{k-1})$ represents the mapping functions relating the two states.

In this phase, the fundamental operation is the **local linearization** of the functions f and h. Since the EKF works on a first-order Taylor series approximation, it is necessary to compute the Jacobian matrices to correctly propagate the error covariance, aiming for an optimal estimate of the robot's real state.

Once the state estimate is computed, the filter proceeds to update the **State Covariance Matrix (P_k)**. Formally, this matrix represents the error covariance estimate and is defined as the expected value of the product between the estimation error and its transpose:

$$P_k = E[e_{x,k} e_{x,k}^T] \tag{4.4}$$

where $e_{x,k}$ is the state estimation error, i.e., the error between the real state and the estimated one: $e_{x,k} = X_k - \hat{X}_{k|k}$. The matrix P_k thus provides a measure of the **precision of the current estimate**: small values on the main diagonal indicate a very accurate estimate. In the prediction phase, this uncertainty is projected into the future (a priori estimate) through the following equation:

$$\text{state covariance matrix: } P_{k|k-1} = F_k \cdot P_{k-1|k-1} \cdot F_k^T + Q_k \tag{4.5}$$

Where:

- $P_{k|k-1}$ is the state covariance matrix (a priori).
- $P_{k-1|k-1}$ is the state covariance matrix at the previous step.
- **F_k is the Jacobian matrix of the state transition model**.

Specifically, F_k is defined as the matrix of partial derivatives of the state transition functions f with respect to the state vector X:

$$F_k = \frac{\partial f}{\partial X} = \begin{bmatrix} \frac{\partial f_1}{\partial X_1} & \cdots & \frac{\partial f_1}{\partial X_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial X_1} & \cdots & \frac{\partial f_n}{\partial X_n} \end{bmatrix} \tag{4.6}$$

This matrix allows the filter to track how the state vector has changed relative to the previous time step through a local linearization. In this operation, the previous uncertainty is amplified by the system dynamics via the Jacobian F_k and further increased by the process noise Q_k. This reflects the loss of information (and thus the increase in statistical doubt) that occurs over time as the system evolves without the corrective support of new external observations.

4.1.2 Correction Phase

Following the prediction phase, the filter performs the correction of the a priori estimate using the real-time measurements from the on-board sensors. The first step involves calculating the **innovation** (or measurement residual y_k), which represents the difference between the actual measurement Z_k and the measurement predicted by the observation model $h(\hat{X}_{k|k-1})$:

$$y_k = Z_k - h(\hat{X}_{k|k-1}) \tag{4.7}$$

To proceed with the update, the **Jacobian matrix of the observation model (H_k)** must be computed. Similarly to the prediction phase, the non-linear function h is locally linearized via partial derivatives with respect to the state.

$$H_k = \frac{\partial h}{\partial X} = \begin{pmatrix} \frac{\partial h_1}{\partial X_1} & \cdots & \frac{\partial h_1}{\partial X_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial X_1} & \cdots & \frac{\partial h_m}{\partial X_n} \end{pmatrix} \tag{4.8}$$

This matrix is essential as it describes the relationship between state variations and the corresponding sensor measurements.

Subsequently, it is possible to calculate the **innovation covariance** (S_k). Its formulation is analogous to that of the state covariance, as it represents the uncertainty associated with the innovation measurement, which is directly related to the difference between predicted and actual measurements.

This error $y_k = Z_k - h(\hat{X}_{k|k-1})$ is the result of the mathematical inaccuracy of the model, sensor measurements, process noise, and environmental factors, combining the prediction uncertainty P with the intrinsic uncertainty of the sensors R.

This uncertainty is projected into the future (a priori estimate) through the following equation:

$$S_k = H_k P_{k|k-1} H_k^T + R_k \tag{4.9}$$

Where:

- S_k is the **innovation covariance**.
- $P_{k|k-1}$ is the predicted state covariance matrix (a priori).
- H_k is the Jacobian matrix of the observation model.
- R_k is the covariance matrix of the measurement noise.

The core of the correction phase is the computation of the **Kalman Gain** (K_k), which determines the weight to be assigned to the actual measurement relative to the theoretical prediction:

$$K_k = P_{k|k-1} H_k^T S_k^{-1} \tag{4.10}$$

Where:

- K_k is the Kalman gain.
- H_k^T is the transpose of the measurement Jacobian matrix.
- S_k^{-1} is the inverse of the innovation covariance.

As highlighted by Welch and Bishop [28], the gain K acts as a statistical mediator: if the measurement uncertainty (R) approaches zero, the gain increases, leading the filter to rely almost exclusively on the sensor data. Conversely, if the prediction uncertainty is low compared to the sensor noise, the filter will maintain an estimate closer to the physical model.

Once the gain K_k is determined, it is possible to calculate the **state update** to obtain the a posteriori estimate. The state is updated by combining the predicted estimate with the innovation y_k, weighted by the Kalman gain:

$$\hat{X}_{k|k} = \hat{X}_{k|k-1} + K_k y_k \tag{4.11}$$

Simultaneously, the error covariance matrix is updated, reflecting the reduction in uncertainty thanks to the acquisition of the new measurement:

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \tag{4.12}$$

Where:

- $\hat{X}_{k|k}$ is the updated state estimate (a posteriori).

- I represents the identity matrix.
- $P_{k|k}$ is the updated state covariance matrix.

In conclusion, through this iterative process, we have obtained the optimal estimate of the system state at time k, minimizing the error covariance relative to the estimate made at the previous time step.

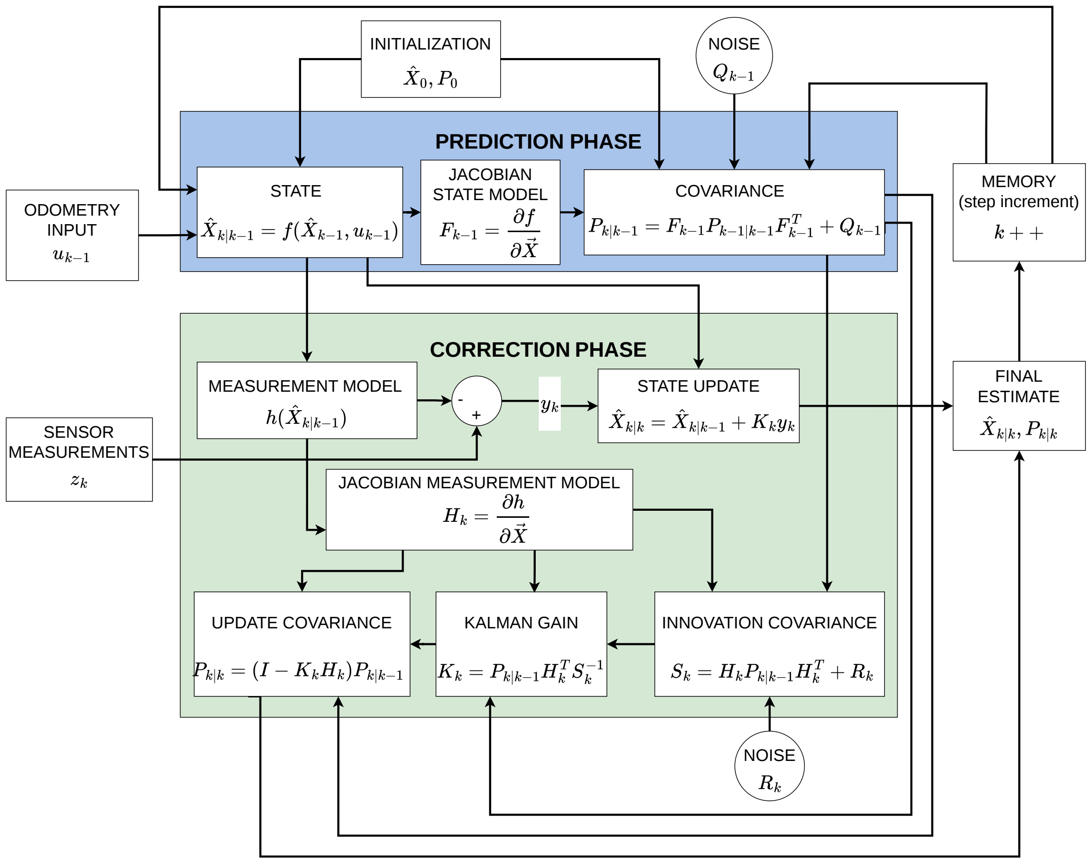
INITIALIZATION
$\hat{X}_0, P_0$
NOISE
Q_{k-1}
PREDICTION PHASE
STATE
$\hat{X}_{k|k-1} = f(\hat{X}_{k-1}, u_{k-1})$
JACOBIAN STATE MODEL
$F_{k-1} = \frac{\partial f}{\partial \vec{X}}$
COVARIANCE
$P_{k|k-1} = F_{k-1}P_{k-1|k-1}F_{k-1}^T + Q_{k-1}$
MEMORY (step increment)
$k++$
ODOMETRY INPUT
u_{k-1}
CORRECTION PHASE
MEASUREMENT MODEL
$h(\hat{X}_{k|k-1})$
-
+
y_k
STATE UPDATE
$\hat{X}_{k|k} = \hat{X}_{k|k-1} + K_k y_k$
FINAL ESTIMATE
$\hat{X}_{k|k}, P_{k|k}$
SENSOR MEASUREMENTS
z_k
JACOBIAN MEASUREMENT MODEL
$H_k = \frac{\partial h}{\partial \vec{X}}$
UPDATE COVARIANCE
$P_{k|k} = (I - K_k H_k)P_{k|k-1}$
KALMAN GAIN
$K_k = P_{k|k-1}H_k^T S_k^{-1}$
INNOVATION COVARIANCE
$S_k = H_k P_{k|k-1} H_k^T + R_k$
NOISE
R_k

Figure 4.1. EKF block diagram.

4.2 Principles for Correct Tuning

The effectiveness of the Extended Kalman Filter in state estimation depends drastically on the correct configuration of the process noise covariance matrix Q and the measurement noise covariance matrix R, as well as on a proper definition of the initial state and its associated uncertainty. Although theory provides the mathematical foundations, the practical implementation on a real system like AgriMaRo presents significant challenges related to the stochastic nature of these parameters.

The starting point is the definition of the **Initial State vector** ($X_{0|0}$), containing the initial values of the state variables. This is the first coefficient to be set with care, as it influences the filter's behavior during the initial transient phase.

In parallel, it is necessary to define the **Initial State Covariance** ($P_{0|0}$): this matrix determines the confidence the filter places in the initial estimate. A high value of $P_{0|0}$ indicates high uncertainty, allowing the filter to quickly correct the initial estimate by relying more heavily on the first measurements.

Subsequently, the critical parameters to be tuned are the **measurement noise covariance matrix** R and the **process noise covariance matrix** Q. As highlighted in recent literature on EKF auto-tuning [3], the manual "tuning" process is often burdensome and requires deep knowledge of the system. An incorrect choice of these parameters not only worsens estimation accuracy but can lead to two critical issues:

- **Filter Divergence:**When the values of Q and R are unbalanced, the filter can accumulate increasing errors, leading it to diverge irreversibly from the real state.

- **Inconsistency:** The filter provides an estimate of its own uncertainty ($P_{k|k}$) that does not reflect the actual error, rendering the data unreliable.

The balance between these matrices defines the filter's "trade-off". On one hand, the matrix R is linked to the physical characteristics of the sensors (often derivable from sensor variance as Gaussian white noise); on the other hand, the matrix Q must capture everything the chosen mathematical model does not see. The latter does not have a direct correlation with specific test results, as not all state variables are observable by sensors, making this parameter more problematic. The difficulty lies in the fact that, in variable operating contexts, a set of parameters optimal for a certain driving condition might not be so for another, necessitating an in-depth analysis of the filter's sensitivity to such variations.

To optimize this process, it is fundamental to identify performance metrics. The main parameters to minimize are the state covariance $P_{k|k}$ (and consequently the state estimation error $e_{x,k}$), the innovation covariance S_k, and the measurement residual from which it derives (y_k, previously referred to as innovation). In conclusion, a fundamental parameter to consider for validating the tuning is **Filter Consistency**. A filter is defined as consistent if it satisfies the following statistical conditions:

- The estimation errors are **zero-mean**.
- The estimation is **efficient**, meaning the covariance calculated by the filter corresponds to the actual mean squared error: $E[e_{x,k}e_{x,k}^T] = P_{k|k}$.
- The innovation follows a **white Gaussian noise** sequence: $y_k \sim \mathcal{N}(0, S_k)$.

These conditions can be analytically verified through two specific statistical tests:

- **Normalized Estimation Error Squared (NEES):** Evaluates the consistency of the state error.

$$\epsilon_{NEES,k} = e_{x,k}^T P_{k|k}^{-1} e_{x,k} \tag{4.13}$$

- **Normalized Innovation Squared (NIS):**Evaluates the consistency of the innovation with respect to its covariance.

$$\epsilon_{NIS,k} = y_k^T S_k^{-1} y_k \tag{4.14}$$

Based on the **Chi-squared (χ^2) hypothesis**, consistency is verified if the mean values of these tests approximate the system's degrees of freedom:

$$E[\epsilon_{NEES,k}] \approx n \quad \text{(where } n \text{ is the number of states)} \tag{4.15}$$

$$E[\epsilon_{NIS,k}] \approx p \quad \text{(where } p \text{ is the number of measurements)} \tag{4.16}$$

Analyzing these parameters allows us to establish whether the filter is operating correctly or if a retuning of the Q and R matrices is required.

However, the practical application of these metrics presents a significant operational limitation. While the NIS can be calculated in real-time based exclusively on available sensor measurements, the calculation of the NEES requires knowledge of the true estimation error ($e_{x,k}$), which implies the availability of the so-called **Ground Truth** (the exact real state of the system). In the absence of a high-precision external reference system (such as RTK or Motion Capture), the NEES cannot be directly computed during field tests. For this reason, for the online validation of the filter on AgriMaRo, the analysis will focus primarily on the NIS test and the physical coherence of the estimated trajectories.

4.3 EKF Design

This section illustrates the design of the first Extended Kalman Filter prototype designed for the AgriMaRo platform. The initial filter design was conceived to enable the filtering and estimation of the fundamental variables required to study the robot's dynamics. The underlying mathematical models for this implementation will be examined in detail, along with the tuning principles adopted to evaluate the performance, consistency, and effectiveness of the resulting EKF structure.

4.3.1 State and Measurement Vectors

The first step in developing the Extended Kalman Filter (EKF) for the AgriMaRo platform involves a detailed analysis of the states required for the specific application and the measurement outputs provided by the onboard sensors.

Based on the sensor suite integrated into the robot, the measurement vector z is composed of the following inputs:

- **Angular velocities** ($\omega_1, \omega_2, \omega_3$) obtained from the RPM sensors on each wheel.

- Rigid body **accelerations** in the longitudinal and lateral directions (a_{Bx}, a_{By}), the **yaw** and the **yaw rate** ($\varphi, \dot{\varphi}$) provided by the Inertial Measurement Unit (IMU).

- **Steering angles** ($\delta_1, \delta_2, \delta_3$) measured by the absolute encoders.

The measurement vector is thus defined as:

$$Z = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ a_{Bx} \\ a_{By} \\ \dot{\varphi} \\ \varphi \\ \delta_1 \\ \delta_2 \\ \delta_3 \end{bmatrix} \tag{4.17}$$

Regarding the **state vector** X, an initial configuration was chosen to include 14 variables to allow for a comprehensive optimization of all parameters essential for the robot's control:

- $X_{pos}, Y_{pos}, \varphi$: representing the global **pose** of the robot.

- $v_{Bx}, v_{By}, \dot{\varphi}$: representing the longitudinal **velocity**, lateral velocity, and yaw rate in the body-fixed frame.

- a_{Bx}, a_{By}: representing the body **accelerations**.

- s_1, s_2, s_3: representing the **longitudinal slip** for each of the three wheels.

- $\delta_1, \delta_2, \delta_3$: representing the actual **steering angles** of the wheels.

The resulting state vector is defined as:

$$X = \begin{bmatrix} X_{pos} \\ Y_{pos} \\ \varphi \\ v_{Bx} \\ v_{By} \\ \dot{\varphi} \\ a_{Bx} \\ a_{By} \\ s_1 \\ s_2 \\ s_3 \\ \delta_1 \\ \delta_2 \\ \delta_3 \end{bmatrix} \tag{4.18}$$

Having defined the measurement and state variables, the next phase consists of deriving the theoretical model that describes the evolution of each variable. This requires an in-depth study of the robot's kinematics and dynamics to determine the state transition functions.

4.3.2 Measurement Model

The next step consists in defining the equations that constitute the measurement model, starting from the formulation of the wheel angular velocities (ω_i). To correctly determine these values, it is necessary to transpose the linear velocities and the yaw rate from the origin of the **robot's body-fixed frame to** the center of each individual **wheel**. This operation is based on the fundamental formula of rigid body kinematics:

$$\mathbf{V}_P = \mathbf{V}_O + \boldsymbol{\Omega} \times \mathbf{r}_{OP} \tag{4.19}$$

Where:

- V_P represents the velocity vector at the point of interest (wheel center);
- V_O is the velocity calculated at the origin of the robot's reference frame;

- Ω is the angular velocity vector of the rigid body;
- r_{OP} is the position vector connecting the origin of the reference frame to the center of the $i-th$ wheel.

In this analysis, the kinematic formula is simplified by considering the yaw rate ($\dot{\varphi}$) as the only relevant component of the angular velocity vector. As described by Genta in *Motor Vehicle Dynamics* [8], for the study of the dynamics of vehicles operating on planar surfaces (**2D models**), the contributions of roll and pitch can be considered negligible. This simplification allows the kinematic model to focus on the rotation around the vertical axis, which represents the primary factor for estimating the robot's trajectory and orientation.

To derive the wheel angular velocities (ω_i), the linear velocity component at the wheel center must be projected along the **wheel's longitudinal direction**. This projection accounts for the steering angle δ_i, resulting in the longitudinal velocity V_{wi}:

$$V_{wi} = (V_{Bx} - \dot{\varphi} y_i)\cos\delta_i + (V_{By} + \dot{\varphi} x_i)\sin\delta_i \tag{4.20}$$

Where x_i and y_i represent the wheel coordinates relative to the body frame origin. To relate this linear velocity to the RPM sensor measurement, the **longitudinal slip** (s_i) is introduced. Based on the definition for a vehicle under traction:

$$s_i = \frac{\omega_i r - V_{wi}}{\omega_i r} \tag{4.21}$$

Where r represents the effective wheel radius. The theoretical expression for the angular velocity is derived as:

$$\omega_i = \frac{V_{wi}}{(1 - s_i) r} \tag{4.22}$$

Integrating measurements from the Inertial Measurement Unit (IMU) requires accounting for the sensor being fixed to the robot chassis. In the moving body frame, the **accelerations** a_{Bx} and a_{By} do not correspond to the simple time derivative of the linear velocities ($\dot{v}_{Bx}, \dot{v}_{By}$), but are affected by the frame rotation. According to the mechanics of rotating systems, extensively covered by Genta for automotive applications [8], measurements in a non-inertial frame must include transport terms, also known as **Coriolis acceleration**. This term comes from the cross product

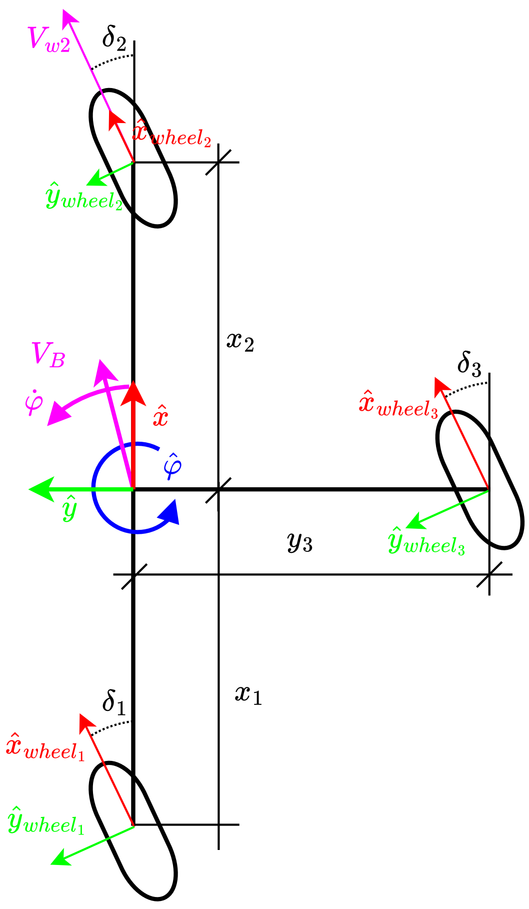

Figure 4.2. Wheel linear velocity evaluation.

between angular and linear velocity:

$$\mathbf{a}_{absolute} = \left[\frac{d\mathbf{v}}{dt}\right]_{body} + \mathbf{\Omega} \times \mathbf{v} \tag{4.23}$$

For the AgriMaRo platform, as previously discussed regarding wheel velocities, the angular velocity vector can be approximated by the yaw rate ($\dot{\varphi}$) alone. The equations for the measured accelerations are:

$$a_{Bx} = \dot{v}_{Bx} - \dot{\varphi} v_{By} \tag{4.24}$$

$$a_{By} = \dot{v}_{By} + \dot{\varphi} v_{Bx} \tag{4.25}$$

The terms $-\dot{\varphi} v_{By}$ and $+\dot{\varphi} v_{Bx}$ represent the acceleration components detected by the IMU detects when the velocity vector changes direction during cornering. This means that even at constant speed, the sensor records a value because the frame is rotating. Including these terms is necessary to prevent the EKF from interpreting centripetal acceleration as a change in linear velocity.

However, given the nature of the robot's operations,characterized by low speeds and extremely slow dynamics, a design simplification can be introduced. Assuming that the time derivatives of the **linear velocity** ($\dot{v}_{Bx}, \dot{v}_{By}$) are **negligible** compared to the rotational terms, the model can be approximated using only the Coriolis contribution:

$$a_{Bx} \approx -\dot{\varphi} v_{By} \tag{4.26}$$

$$a_{By} \approx \dot{\varphi} v_{Bx} \tag{4.27}$$

This approximation helps stabilize the filter, preventing the high-frequency noise typically associated with acceleration derivatives from degrading the estimation of the primary states.

The other variables in the measurement vector, specifically the yaw angle (φ), the yaw rate ($\dot{\varphi}$), and the steering angles (δ_i), are already included in the state vector X. This means there is a **direct correspondence** between the sensor readings and the estimated states. For these components, the measurement equations are simple equalities, where each measurement is mapped directly to its state without any additional transformations. Finally, we obtain the following model for the measurement variables, combining the kinematic formulas for the wheels, the inertial terms for the IMU, and the direct identities for the other states:

$$h(X) = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ a_{Bx} \\ a_{By} \\ \dot{\varphi} \\ \varphi \\ \delta_1 \\ \delta_2 \\ \delta_3 \end{bmatrix} = \begin{bmatrix} \frac{(V_{Bx}-\dot{\varphi}y_1)\cos\delta_1+(V_{By}+\dot{\varphi}x_1)\sin\delta_1}{(1-s_1)r} \\ \frac{(V_{Bx}-\dot{\varphi}y_2)\cos\delta_2+(V_{By}+\dot{\varphi}x_2)\sin\delta_2}{(1-s_2)r} \\ \frac{(V_{Bx}-\dot{\varphi}y_3)\cos\delta_3+(V_{By}+\dot{\varphi}x_3)\sin\delta_3}{(1-s_3)r} \\ -\dot{\varphi}V_{By} \\ \dot{\varphi}V_{Bx} \\ \dot{\varphi} \\ \varphi \\ \delta_1 \\ \delta_2 \\ \delta_3 \end{bmatrix} \tag{4.28}$$

This completes the definition of the measurement model $h(x)$. These equations allow the EKF to compare the sensor data with the predicted states during the correction phase, ensuring a reliable estimate of the robot's dynamic behavior.

4.3.3 State Model

As explained in subsection 4.1.1, the state model describes the time evolution of the system, defining how the current state (X_k) transforms into the next one (X_{k+1}). Since AgriMaRo operates in a real environment, the mathematical model, though accurate, is still an approximation of reality. For this reason, the model is not perfectly deterministic but includes a **process noise** term (ω_k), which represents modeling uncertainties and external disturbances:

$$X_{k+1} = f(X_k) + \omega_k \tag{4.29}$$

First, we analyze the pose equations in the global frame. The position (X_{pos}, Y_{pos}) and the orientation (φ) of the robot are updated by integrating the velocities over time and projecting them from the vehicle's reference frame (body frame) to the fixed global frame (**world frame**):

$$X_{pos_{k+1}} = X_{pos_k} + T(v_{Bx}\cos\varphi_k - v_{By}\sin\varphi_k) \tag{4.30}$$

$$Y_{pos_{k+1}} = Y_{pos_k} + T(v_{Bx}\sin\varphi_k + v_{By}\cos\varphi_k) \tag{4.31}$$

$$\varphi_{k+1} = \varphi_k + T\dot{\varphi}_k \tag{4.32}$$

Where T represents the **sampling time**. These equations describe the basic kinematics of the robot by assuming that the velocity remains constant within the interval T. Mathematically, this is a simple numerical integration where the displacement is calculated as the product of time and instantaneous velocity.

Regarding the velocities, these are evaluated in the vehicle-fixed reference frame (**body frame**). Similarly to the pose equations, we use an integration method where the sampling time is multiplied by the linear acceleration of the body. However, since the reference frame is rotating, we must include correction terms to relate the accelerations measured by the IMU to the actual change in velocities v_{Bx} and v_{By}:

$$v_{Bx_{k+1}} = v_{Bx_k} + T(a_{x_k} + \dot{\varphi}v_{By_k}) \tag{4.33}$$

$$v_{By_{k+1}} = v_{By_k} + T(a_{y_k} - \dot{\varphi}v_{Bx_k}) \tag{4.34}$$

As previously discussed in the measurement model section (subsection 4.3.2), these terms are necessary to compensate for rotational effects (Coriolis acceleration). This ensures that the state model is consistent with the sensor physics and correctly isolates the pure linear velocity variation.

For the remaining states in vector X, specifically the accelerations (a_{Bx}, a_{By}), the yaw rate $(\dot{\varphi})$, the slip coefficients (s_i), and the steering angles (δ_i), we chose to use the **Random Walk** approach.This decision is based on the fact that modeling the time evolution of these variables analytically would require extremely complex equations. For instance, to accurately describe the dynamics of steering angles or slip coefficients, we would need to account for friction forces, tire deformation, and actuator dynamics. This would introduce parameters that are very difficult to identify precisely.

With a Random Walk, we assume that the state at the next step remains the same as the current one, leaving it to the process noise (ω_k) to represent how the variable changes over time:

$$X_{k+1} = X_k + \omega_k \tag{4.35}$$

From the Kalman filter perspective, this means the model does not provide an active prediction of how the variable will change. Instead, it relies almost entirely on the correction phase based on sensor measurements. To make this strategy work, we must assign low confidence to the model for these states by setting a high process noise variance. This allows the EKF to quickly update the estimated value as soon as the sensors detect a change, enabling the filter to track the real robot dynamics even without a complex deterministic model.

In conclusion, the complete state transition model $f(x)$ for the 14 states of the AgriMaRo platform is defined as follows:

$$f(X) = \begin{bmatrix} X_{pos_k} + T(v_{Bx}\cos\varphi_k - v_{By}\sin\varphi_k) \\ Y_{pos_k} + T(v_{Bx}\sin\varphi_k + v_{By}\cos\varphi_k) \\ \varphi_k + T\dot{\varphi}_k \\ v_{Bx_k} + T(a_{x_k} + \dot{\varphi} v_{By_k}) \\ v_{By_k} + T(a_{y_k} - \dot{\varphi} v_{Bx_k}) \\ a_{Bx_k} \\ a_{By_k} \\ \dot{\varphi}_k \\ s_{1_k} \\ s_{2_k} \\ s_{3_k} \\ \delta_{1_k} \\ \delta_{2_k} \\ \delta_{3_k} \end{bmatrix} \tag{4.36}$$

4.4 EKF Tuning and Implementation

Once the mathematical state and measurement models have been defined, the next and most critical phase is the tuning of the filter parameters. As discussed in the general principles of section 4.2, the performance of an EKF is strictly dependent on the correct definition of the covariance matrices. Incorrect tuning can lead to divergent estimates or an excessive lag in the system response. Following the approach suggested in the literature to improve state estimation [3], a systematic calibration was carried out to adapt the algorithm to the specific dynamics of AgriMaRo.

4.4.1 Post-Processing Approach

Before the final implementation on the robot's real-time control system, a post-processing tuning and validation strategy was adopted. This choice was driven by several technical requirements.

First, regarding the **testing efficiency**, working with recorded data allows for the instantaneous testing of many different combinations for the parameters Q, R, and P_0 on the same dynamic scenario. This avoids the need for repeated physical field tests for every single modification, drastically speeding up the optimization process.

Secondly, for the **operational safety**, an uncalibrated filter can generate unstable estimates. If these were fed directly to the AgriMaRo controller, they could cause abrupt movements or exceed mechanical safety limits. Offline analysis allows for a "safe failure," ensuring that only validated parameters are eventually deployed on the robot.

Finally, the comparative analysis is a key factor because **offline visualization** makes it easier to compare filter estimates with raw sensor data. This allows for the precise identification of the causes behind any drift or lag that would be difficult to isolate in real-time.

4.4.2 Initial State Definition (X_0 and P_0)

To initialize the EKF algorithm, it is necessary to define the initial state of the system (X_0) and the corresponding initial error covariance matrix (P_0). For the **state vector** X_0, all 14 values were set to **zero**, based on the specific operational conditions of AgriMaRo at power-on:

- **Pose ($X_{pos}, Y_{pos}, \varphi$):** The **initial position** is assumed to be the **origin** of the reference frame, with the yaw angle (φ) considered relative to the starting orientation. As explained in subsection 3.3.3, the IMU was calibrated without using the magnetometer, making the **yaw a relative measurement**.

- **Velocities and Accelerations:** At startup, the robot is **stationary**; therefore, linear velocities, yaw rate, and accelerations are set to zero.

- **Slip coefficients (s_i):** Since the robot's **velocities** are **null** at the start, the slip coefficients are also assumed to be zero because there is no relative motion between the wheels and the ground.

- **Steering angles (δ_i):** The wheels are assumed to be straight thanks to an **automatic homing** procedure managed by the robot's low-level control system. At every power-on, this controller sends a specific command to the steering motors, forcing them to move until they reach their calibrated mechanical zero position.

Consequently, the initial state vector is defined as follows:

$$X_0 = \begin{bmatrix} X_{pos_0} \\ Y_{pos_0} \\ \varphi_0 \\ v_{Bx_0} \\ v_{By_0} \\ \dot{\varphi}_0 \\ a_{Bx_0} \\ a_{By_0} \\ s_{1_0} \\ s_{2_0} \\ s_{3_0} \\ \delta_{1_0} \\ \delta_{2_0} \\ \delta_{3_0} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.37}$$

Regarding the **matrix** P_0, which is set as a **diagonal matrix** to represent the uncertainty of these initial values, a differentiated strategy was adopted. For the states related to position, velocities, and slip, very small values were assigned to the diagonal because the starting conditions of the stationary robot are well known and highly certain.

An intermediate value was chosen for the accelerations; this is because, even though the robot is stationary, a slight acceleration signal from the IMU can always be measured due to sensor noise or small offsets.

On the contrary, a higher covariance value was set for the **steering angles**. This choice is necessary because the resolution of the encoders is much finer than the actual mechanical precision of the steering system. Due to the presence of a mechanical **dead band** and backlash, the wheels never return to a perfect mathematical zero after the homing procedure. Since the encoders are sensitive enough to detect even these tiny physical discrepancies, the initial reading is rarely zero. Assigning a **high initial covariance** (low confidence) allows the EKF to quickly "ignore" the assumed zero state and converge to the actual sensor measurements as soon as the robot begins to move. The resulting initial covariance matrix P_0 follows a diagonal structure where each p_i represents the initial uncertainty of the corresponding state:

$$P_0 = \text{diag}(10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-3}, 10^{-3}, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-1}, 10^{-1}, 10^{-1}) \tag{4.38}$$

4.4.3 Calculation of Covariance Matrices R and Q

As previously mentioned in the general principles of section 4.2, the performance of the EKF depends drastically on the values assigned to the noise covariance matrices. In statistical terms, the diagonal elements of these matrices represent the variance of the noise associated with the sensor measurements (matrix R) and the uncertainty of the state model (matrix Q), respectively.

Variance (σ^2) is a measure of data dispersion relative to its mean value. In practice, it tells us how much a sensor signal "dances" around the true value due to electronic or environmental noise. Following the classical estimation theory [11], the variance for a set of n samples is calculated using the following formula:

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{4.39}$$

Where:

- x_i represents the individual acquired data point;
- $\bar{x}$ is the arithmetic mean of the series.

Regarding the **R matrix**, it was decided to perform **static tests** directly on the AgriMaRo hardware instead of relying solely on manufacturer datasheets. The robot was kept powered on in a static condition for approximately 10 minutes, and the real variance for each channel was then calculated from these recordings. During the analysis, it was noted that some sensors exhibited an extremely low variance. In these cases, a minimum limit of 10^{-5} was set. This threshold, also defined as the **"noise floor"** [11], is essential for the numerical robustness of the filter: it prevents the EKF from considering a sensor as "perfect," which would lead it to completely ignore the state model and cause instability in the presence of unforeseen disturbances.

$$R = \text{diag}(10^{-5}, 10^{-5}, 10^{-5}, 10^{-5}, 10^{-5}, 10^{-5}, 10^{-5}, 7.2 \cdot 10^{-5}, 7.2 \cdot 10^{-5}, 7.2 \cdot 10^{-5}) \tag{4.40}$$

Regarding the **Q matrix**, the calculation was differentiated based on whether the states were directly observable and how their respective measurements were handled in the R matrix.

For the steering angles (δ_i), which are **directly observable** and characterized by a clear mechanical uncertainty, the variance obtained from the static tests ($7.2 \cdot 10^{-5}$) was used for both R and Q. This ensures a balanced trust between the encoder readings and the Random Walk model. Conversely, for other measured states such as accelerations and yaw rate, a different approach was taken. Since these sensors were so precise that their R values had to be artificially limited by the 10^{-5} noise floor, they were treated similarly to non-observable states. By prioritizing the tuning of their process noise in Q, we ensure that the filter's behavior is driven by the desired system dynamics rather than being overly constrained by an arbitrary sensor floor. Specifically, the yaw rate process noise was set to 10^{-5} to maintain consistency with the minimum measurement uncertainty.

For the **non-directly observable** states, the values were assigned based on their physical role in the system. The global pose (X, Y, φ) was set to 10^{-8}; as these are purely kinematic states derived from velocity integration, their uncertainty must be "guided" by the velocity estimates rather than random model jumps. The linear velocities (v_{Bx}, v_{By}) were set to 10^{-6} to account for minor inaccuracies in the dynamic model. The accelerations (a_{Bx}, a_{By}), being the "engine" of the Random Walk for dynamic transitions, were set to 10^{-3} to allow the EKF to promptly track real IMU transients. Finally, the slip coefficients (s_i) were set to 10^{-5} to provide a smooth estimation, avoiding oscillations while still allowing the filter to adapt to different ground conditions The resulting initial Q matrix is therefore defined as follows:

$$
\begin{aligned}
Q = \operatorname{diag}(&10^{-8}, 10^{-8}, 10^{-8}, 10^{-6}, 10^{-6}, 10^{-5}, 10^{-3}, 10^{-3}, 10^{-5}, 10^{-5}, 10^{-5}, \\
&7.2 \cdot 10^{-5}, 7.2 \cdot 10^{-5}, 7.2 \cdot 10^{-5})
\end{aligned}
\tag{4.41}
$$

Starting from this initial matrix, a series of **post-processing tuning sessions** was conducted [24], refining the parameters to achieve the most accurate and stable evaluation for the AgriMaRo system's specific dynamics.

4.5 Limitations of the Initial EKF and Redesign Strategy

Despite the systematic tuning of the covariance matrices described in the previous section, the experimental results revealed that the initial EKF configuration was not sufficient to provide an accurate state estimation for the AgriMaRo platform. The **high degree of non-linearity** in the measurement model, particularly regarding the wheel angular velocity equations, led to significant estimation errors and filter instability.

4.5.1 Theoretical Limitations of the EKF Approach

The difficulties encountered are consistent with the known limitations of the Extended Kalman Filter when applied to highly non-linear systems. As discussed by Wan and van der Merwe in their work on the Unscented Kalman Filter (UKF) [27], the EKF relies on a first-order Taylor series expansion to linearize the system dynamics. This approach introduces three primary technical constraints:

- **First-Order Approximation and Mean Bias:** By approximating non-linear functions with a Jacobian matrix, the EKF only considers the tangent line at the current state. For a system like AgriMaRo, which exhibits strong non-linearities and sharp variations in the state equations, this **first-order approximation** is insufficient to capture the true curvature of the system's evolution. By ignoring higher-order terms (which describe how the curve "bends"), the filter fails to accurately propagate the mean of the probability distribution. This leads to a **systematic bias**, causing a constant drift of the estimated state away from the ground truth.

- **Covariance Propagation Error:** The EKF assumes that the uncertainty distribution remains **Gaussian** even after passing through a non-linear transformation. In reality, non-linearities **distort** the distribution, often making it asymmetric or multi-modal, leading the filter to systematically underestimate the actual covariance. The result is a "self-confident" filter that perceives its own estimate as far more accurate than it truly is. This phenomenon drastically reduces the Kalman Gain (K), making the filter unable to correct the state through measurements. Mechanically, the filter stops weighting external corrections and relies blindly on an internal model that becomes increasingly erroneous.

- **Instability and Divergence:** If the **linearization** occurs at a **point far** from the actual state, due to poor initialization, excessive noise, or abrupt dynamic changes, the calculated Jacobian becomes highly inaccurate, as the tangent no longer reflects the function's behavior at that point. This triggers an exponential growth in the state estimation error: every correction based on an incorrect Jacobian moves the state even further from the truth, initiating a positive feedback loop that leads to numerical instability and complete filter divergence.

4.5.2 Model Redesign: Simplification and Linearization

Having concluded that the EKF can be effectively utilized as an optimal filter primarily for systems characterized by low non-linearity, a redesign of the dynamic model was undertaken to drastically reduce its analytical complexity. The primary objective of this strategy was to mitigate the sources of filter instability by directly intervening in the structure of the state vector and the measurement equations.The most substantial modification involved the **exclusion** of the **longitudinal slip** coefficients (s_i) from the filter's state vector. Consequently, the calculation of the wheel angular velocities was simplified according to the following kinematic formulation:

$$\omega_i = \frac{(V_{Bx} - \dot{\varphi} y_1)\cos\delta_1 + (V_{By} + \dot{\varphi} x_1)\sin\delta_1}{r} \tag{4.42}$$

This solution significantly decreases the system's non-linearity. From an analytical perspective, in the previous model, the presence of slip introduced the term $f(s) = \frac{1}{1-s}$, a highly non-linear function. The EKF algorithm was thus forced to continuously recalculate the partial derivative of this term, namely $f'(s) = \frac{1}{(1-s)^2}$, which changes significantly at each time step and whose linear approximation is valid only within an extremely narrow range around the operating point.

By adopting the new formulation, it is implicitly hypothesized that the slip is zero ($s = 0$). Under this condition, the function $f(s)$ assumes a unit value and, more importantly, its derivative $f'(s)$ becomes zero (i.e., constant). This approach fundamentally eliminates the need for continuous linearization for this specific model component, as the term becomes **inherently linear**.

However, the practical implications of such simplification must be considered. With this approach, the slip is no longer directly estimated by the EKF, necessitating a subsequent evaluation through post-processing analysis of the obtained data. A critical aspect of this modeling is that, if developed following a classical setup, it generates a strong **correlation** between the measured **angular velocities** and the estimated **linear velocities**. This can lead to results where the slip appears to be zero even in dynamic conditions where physical slippage is present, as the filter tends to enforce the ideality condition set to guarantee the numerical stability of the system.

4.5.3 Integration of the Yaw Rate Bias

The second fundamental modification introduced in the filter redesign involves the integration of a yaw rate bias term as an additional state variable.

As described in the previous chapter regarding sensors chapter 3, the hardware configuration of the AgriMaRo platform features an IMU without a magnetometer. This choice implies the impossibility of referencing the Earth's magnetic field to correct the robot's absolute orientation. Consequently, the **yaw angle** calculation is subject to a **temporal drift** phenomenon: small constant offsets and noise present in the angular velocity measurement are integrated over time, leading to a growing error in the estimated orientation (heading). Introducing the **yaw rate bias** (b_g) directly into the state vector allows the EKF to actively estimate and "absorb" this systematic error during operation. Instead of interpreting every signal from the gyroscope as an actual rotation of the vehicle, the filter is now able to distinguish between real movement and the sensor's intrinsic offset. This solution drastically stabilizes the pose estimation, preventing accumulated errors from compromising the robot's overall precision during operational missions.

4.5.4 Revised Filter Formulation

Following the redesign choices described above, the structure of the Extended Kalman Filter has been updated. The new state vector excludes the slip coefficients $[s_1, s_2, s_3]^T$ and integrates the yaw rate bias b_g, while maintaining the kinematic variables and steering angles necessary for the platform's control.The state vector now consists of 12 variables.

$$X = \begin{bmatrix} X_{pos} \\ Y_{pos} \\ \varphi \\ V_{Bx} \\ V_{By} \\ \dot{\varphi} \\ a_{Bx} \\ a_{By} \\ \delta_1 \\ \delta_2 \\ \delta_3 \\ b_g \end{bmatrix} \tag{4.43}$$

The state equations follow a discrete-time model (with sampling time (T). The dynamics for pose, velocities, and accelerations remain consistent with the previous model, while the bias is assumed to have zero variation between steps:

$$f(X) = \begin{bmatrix} X_{pos_k} + T(v_{Bx}\cos\varphi_k - v_{By}\sin\varphi_k) \\ Y_{pos_k} + T(v_{Bx}\sin\varphi_k + v_{By}\cos\varphi_k) \\ \varphi_k + T\dot{\varphi}_k \\ v_{Bx_k} + T(a_{x_k} + \dot{\varphi}v_{By_k}) \\ v_{By_k} + T(a_{y_k} - \dot{\varphi}v_{Bx_k}) \\ a_{Bx_k} \\ a_{By_k} \\ \dot{\varphi}_k \\ \delta_{1_k} \\ \delta_{2_k} \\ \delta_{3_k} \\ b_{g_k} \end{bmatrix} \tag{4.44}$$

The measurement vector z remains unchanged from the initial design.

$$Z = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ a_{Bx} \\ a_{By} \\ \dot{\varphi} \\ \varphi \\ \delta_1 \\ \delta_2 \\ \delta_3 \end{bmatrix} \tag{4.45}$$

The equations of the measurement model $h(x)$ have been modified only for the components relating to wheel velocities and the robot's angular velocity:

- **Wheel angular velocities (ω_i):** As discussed in subsection 4.5.2 , the zero-slip hypothesis ($s = 0$) drastically simplifies the non-linear term. The measurement now depends only on linear velocities, angular velocity, and the steering angle:

$$h_{1,2,3}(x) = \frac{(V_{Bx} - \dot{\varphi} y_1)\cos\delta_1 + (V_{By} + \dot{\varphi} x_1)\sin\delta_1}{r} \tag{4.46}$$

- **Measured yaw rate ($\dot{\varphi}$):** This equation now integrates the estimated bias to correct the gyroscope measurement:

$$h_6(x) = \dot{\varphi} + b_g \tag{4.47}$$

The remaining equations for accelerations (h_4, h_5) and steering angles (h_7, h_8, h_9) remain direct mappings of their respective states, as defined in the original model.

By considering the simplifications and integrations discussed, all components of the measurement function are re-aggregated, thus obtaining the measurement model:

$$h(X) = \begin{bmatrix} \frac{(V_{Bx}-\dot{\varphi} y_1)\cos\delta_1+(V_{By}+\dot{\varphi} x_1)\sin\delta_1}{r} \\ \frac{(V_{Bx}-\dot{\varphi} y_2)\cos\delta_2+(V_{By}+\dot{\varphi} x_2)\sin\delta_2}{r} \\ \frac{(V_{Bx}-\dot{\varphi} y_3)\cos\delta_3+(V_{By}+\dot{\varphi} x_3)\sin\delta_3}{r} \\ -\dot{\varphi} V_{By} \\ \dot{\varphi} V_{Bx} \\ \dot{\varphi} + b_g \\ \varphi \\ \delta_1 \\ \delta_2 \\ \delta_3 \end{bmatrix} \tag{4.48}$$

4.6 Experimental Analysis and Tuning Limitations

The search for optimal EKF parameters (P, Q, R) was initially conducted through offline analysis, starting from the linearized configuration. As a starting point, the covariance matrices were set following rigorous physical criteria: Q values were chosen in consistency with the dynamic modeling, while the uncertainties in R were determined based on the nominal variance of the onboard sensors. This initial calibration served as a basis for a specific tuning process, leading to a final configuration capable of ensuring filter stability and an accurate estimation of the robot's linear velocity, key requirements for optimizing data acquisition.

Once a filter with satisfactory convergence was obtained, the system was configured for **real-time longitudinal slip** calculation. The architecture ensures that the velocity values estimated instantaneously by the EKF are used to derive the slip for each individual wheel. To achieve this, calculation based on the standard SAE formulation (chapter 7) was implemented, which first requires the kinematic transformation of velocities from the center of mass to the individual hubs Equation 4.19. This step is essential to project the quantities estimated by the EKF onto the longitudinal direction of each wheel, thereby obtaining a slip measurement for each wheel. However, during this live integration phase, the most critical structural limitation emerged: the strong correlation (or kinematic coupling) between the angular velocity measured by the encoders (ω) and the estimated linear velocity (v_{Bx} and v_{By}).

Due to the zero-slip hypothesis introduced to stabilize the model, the EKF finds itself in a state of "**modeling blindness**": it lacks equations that allow it to distinguish whether an increase in angular velocity is due to an actual acceleration of the robot or to a physical slippage phenomenon. Since these two velocity values are **linearly dependent by construction**, the filter intrinsically tends to interpret every encoder variation as actual vehicle movement. Consequently, the **real-time calculated slip** value is systematically **zero** in all analyzed tests, as any potential slippage is "absorbed" by the linear velocity estimation to satisfy the ideality constraint imposed on the model.

4.6.1 Adaptive Tuning Strategy and Hardware Limitations

To overcome the linear dependency between angular and linear velocities, a compromise was sought between a plausible velocity estimation and a slip value reflecting physical reality. Consequently, an adaptive tuning strategy was implemented, inspired by dynamic parameter calibration approaches [3,27]. Specifically, the tuning focused on two areas:

- **Process Noise Matrix (Q):** The parameters related to the modeling of angular velocities were increased to signal to the filter that the kinematics based solely on encoders were not entirely accurate during slippage.

- **Measurement Noise Matrix (R):** The uncertainty associated with the wheel RPM sensors (encoders) was increased, while simultaneously decreasing the uncertainty related to the IMU.

The theoretical goal of this configuration was to force the EKF to rely more heavily on the IMU, from which accelerations are derived, and less on the wheel angular velocities. This approach was intended to create the necessary discrepancy between the wheel peripheral speed and the vehicle speed required for slip calculation, while still maintaining a physically meaningful linear velocity.

However, this solution introduced intrinsic problems from the very beginning of implementation. Despite the use of the EKF, the estimated linear velocity became almost exclusively dependent on the acceleration provided by the IMU. Since velocity is calculated through integration in this setup, systematic evaluation errors (**integration bias**) inevitably arose, which were difficult to minimize when other sensors were assigned high uncertainty (R). A second critical issue is related to the tolerance and variance of the accelerations extracted from the IMU. Even minimal noise in the accelerometer, once integrated, generates a velocity value that tends to progressively deviate from the actual value (drift), compromising the reliability of the estimation over time. The combined effect of integration bias and **noise-induced drift** frequently led to filter divergence.

In conclusion, despite numerous adaptive tuning attempts and repeated IMU calibrations to minimize noise, these issues nullified the possibility of obtaining both a reliable velocity estimate and a real slip value. It was therefore concluded that the current sensor suite and the level of hardware accuracy were insufficient to ensure optimal data acquisition and a reliable slip evaluation,key requirements for a traction control system.

Chapter 5

Sensor Augmentation and Perception System

As highlighted in the previous chapter, AgriMaRo's original sensory architecture, while effective for low-level movement and control functions, proved insufficient for an accurate characterization of the vehicle's dynamics. In particular, the lack of a direct measurement of absolute velocity relative to the ground (**ground truth**) represents the main obstacle to estimating longitudinal wheel slip.

Consequently, this chapter focuses on the process of selecting, analyzing, and integrating new transducers. First, the various candidate technologies will be discussed, highlighting their advantages and operational limits in both structured and unstructured agricultural contexts. Subsequently, the actual hardware and software implementation of the chosen solution will be described, addressing issues related to data communication, system synchronization, and the kinematic transformations necessary to align the measurements with the robot's reference frame.

5.1 Analysis for the Integration of New Transducers

AgriMaRo's construction philosophy has always been based on **sensory minimalism**, with the goal of obtaining the maximum amount of dynamic information from the fewest possible components. However, to study the soil-wheel interaction, it is essential to have a velocity reference independent of the rotation of the wheels themselves.

In selecting the new sensor, two guiding criteria were followed:

- **System Autonomy:** Any solution requiring the sensorization of the surrounding environment was rejected a priori, as it would be difficult to implement in both greenhouses and open fields, in addition to involving higher costs.
- **Cost-Performance Ratio:** High-end sensors such as LiDAR (Light Detection and Ranging) were excluded, as their capabilities exceed the needs of this study relative to their high cost.

The primary systems considered were **GPS RTK** and the **Tracking Camera**.

5.1.1 Evaluation of the Global Positioning System (GPS) RTK

Traditional GPS is a positioning system based on the travel time of signals emitted by satellites. A standard receiver calculates its position by triangulating signals from at least four satellites, identifying the distance via the so-called "code" (C/A code). However, this technology has critical limits for precision agricultural robotics: accuracy is limited to approximately **10 meters**, and the update frequency is rarely higher than **10 Hz**. To overcome these limitations, **RTK (Real-Time Kinematic)** technology was evaluated.

Unlike standard GPS, the RTK system does not rely solely on the signal code but on the measurement of the carrier phase of the radio wave [25]. Since the wavelength of the carrier is much smaller than that of the code, spatial resolution increases drastically. The system requires two receivers:

- **Base Station:**A fixed receiver placed at a geographically known position with millimeter precision.
- **Rover:**The receiver mounted on the robot.

The base instantaneously calculates the error introduced by atmospheric disturbances (ionosphere and troposphere) and sends it to the rover in real time. The rover subtracts this error from its own measurement, achieving a position with a

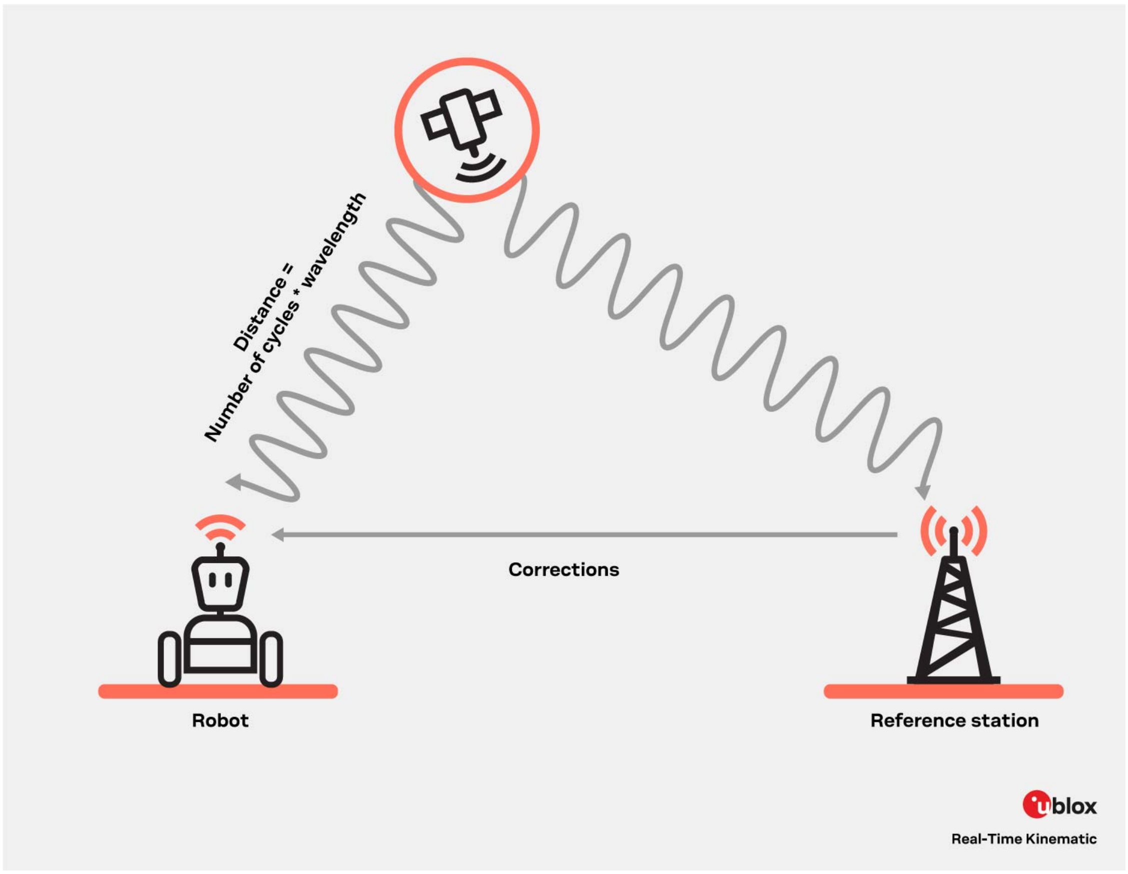

Figure 5.1. Infographic real time kinematic RTK [25].

kinematic precision of **1-2 cm** [25]. Despite its high precision, the use of GPS RTK on AgriMaRo was deemed unsuitable for the following reasons [30]:

- **Shielding and Sky Obstruction:**The RTK system requires a perfect line of sight with the satellites. In agricultural contexts (under tree canopies) or inside greenhouses, the signal is physically blocked. This prevents the sensor from resolving phase ambiguity, causing it to drop from a "Fixed" solution (centimeter-level) to a "Float" solution (meter or decimeter-level), which is useless for slip calculation.

- **Multipath Effect:**In the presence of metal structures (such as greenhouse poles or warehouses), the radio signal can bounce before reaching the sensor. These "echoes" generate unpredictable position errors that the onboard filter struggles to discard.

- **Infrastructure Dependency and "Baseline":**To function correctly, RTK requires proximity to a base station. Millimeter precision decays proportionally to the increase in distance between the robot (Rover) and the Base (a distance known as the baseline). In many remote agricultural areas, the lack of a base station within a few kilometers, or the absence of cellular network coverage to receive corrections via the NTRIP protocol, makes it impossible to activate high-precision mode.

- **Latency and Frequency:**The typical update frequency (10Hz) and the latency of the radio link between the base and rover introduce a time delay. For a detailed analysis of instantaneous velocity, this delay can generate phase shifts between the measured velocity and the rotation of the wheels.

- **Environmental Constraints:** The fact that the sensor works exclusively outdoors makes the AgriMaRo robot unusable for all indoor or covered greenhouse applications, significantly limiting the platform's versatility.

5.1.2 Evaluation of the Intel RealSense T265 Tracking Camera

As an alternative to the GPS system, the Intel RealSense T265 tracking camera was analyzed. This solution was already available in the laboratory and allows for overcoming localization limits in closed environments. It should be noted that, despite its high performance, implementation on the robot required careful management of software criticalities, as the device was discontinued by Intel in 2021; this condition has, over time, led to communication problems and a progressive lack of official driver updates.

The T265 is based on a **Simultaneous Localization and Mapping (SLAM)** solution [12], a framework composed of sensors and algorithms that allow for determining the position, movement, and orientation of an agent in 3D space. This system operates on the **"inside-out" tracking** principle: unlike "outside-in" systems that rely on external references, the camera determines its own pose based exclusively on its onboard sensors and the constant creation and updating of a map of the surrounding environment. The main purpose is to track **6 degrees of freedom (6DoF)**, providing real-time spatial coordinates (X, Y, Z) and orientation through roll, pitch, and yaw angles.

The specific technology used by the T265 is **Visual Inertial Odometry (VIO)** [12], which relates the various internal sensors to obtain a global perception of the surrounding world. This system fuses visual information from the cameras and inertial information provided by an IMU to generate an instantaneous mapping.

Visual Odometry analyzes the environment starting from captures consisting of millions of pixels. These images are processed by a **feature detection** algorithm to identify characteristic points (features). Once these nominal pixels are identified, the algorithm assigns each a "descriptor" or "feature vector," ignoring the remaining non-significant pixels. By repeating this process for every frame, the system compares the displacement of features between consecutive images; by taking into account the time elapsed between frames, it is possible to derive linear and angular velocities. To maximize the effectiveness of this process, the T265 uses optical sensors with an extremely **wide field-of-view (FOV)** through the use of fisheye lenses. These allow for the tracking of features even peripheral to the direction of motion, ensuring navigation stability even when the robot moves near objects lacking detail or in cramped spaces.

Inertial Odometry, based on the use of an IMU, is integrated into the system for two fundamental reasons. The first concerns depth perception: a single camera cannot generate an accurate 3D map of features. Although the problem is partially solved by using calibrated stereo cameras, which perceive depth through **parallax**, the IMU guarantees continuity of the calculation when the cameras fail, for example, in the presence of objects that are too distant where the image seen by the two lenses appears nearly identical. The second reason involves computational cost and latency. Vision algorithms require extremely high computing power, which often depends not on the acquisition speed of the cameras but on the complexity of pixel analysis. Using a high-speed IMU allows for tracking velocity and orientation with high reliability over short time intervals. Since the IMU suffers from drift if used for long periods (on the order of a few seconds), it measures 6DoF correctly for only a few milliseconds before being corrected by camera data; this hybrid approach significantly reduces the number of frames that must be constantly analyzed by the CPU.

The hardware design of the RealSense T265 includes a set of **stereo fisheye cameras** with a circular FOV of approximately 165 degrees. Images are captured by monochromatic global shutter sensors with a diameter of about 800 pixels, a technology essential for avoiding distortions during movement. Internally, it integrates a **Bosch BMI055** inertial unit consisting of a 200 Hz gyroscope and a 62.5 Hz accelerometer synchronized via hardware. The entire system is managed by a powerful Intel Movidius Myriad 2 **Visual Processing Unit (VPU)**, which executes the SLAM algorithm directly on board, fusing all information in real time and providing data with extremely low latency.

In conclusion, the choice of the tracking camera proved preferable for both economic and technical reasons. On one hand, the ability to use the T265 without new purchases optimized laboratory resources; on the other hand, the limitations of GPS RTK, such as limited precision in dynamic positioning, low update frequency, and the requirement to operate exclusively outdoors, made it insufficient for a detailed velocity analysis. The T265, by contrast, provides the versatility necessary to operate even inside greenhouses, delivering the high-precision data required for the study of the AgriMaRo robot's dynamics.

Figure 5.2. Intel RealSense T265 [12]

5.2 Intel RealSense T265 Implementation

The integration of the Intel RealSense T265 Tracking Camera represented a crucial step in the evolution of the AgriMaRo platform, requiring substantial updates to both the hardware architecture and the software pipeline.

5.2.1 Hardware Architecture and Computational Management

To manage the high data flow originating from the T265 and the execution of the state estimation algorithm, it was necessary to integrate a high-performance computing unit on board the vehicle, specifically a mini-PC (**Intel NUC** NUC11TNK), equipped with the latest available Ubuntu distribution (version 22.04 LTS).

This architectural choice was dictated by two critical factors: the computational limits encountered in the previous configuration, based exclusively on the Teensy 4.1 microcontroller, and the physical interface constraints required for high-bandwidth communication with the camera. Early experiments highlighted that executing the Extended Kalman Filter (EKF) directly on the microcontroller imposed an unacceptable compromise between computational capacity and data flow management. This forced a drastic limitation of the overall volume of exchanged data and their sampling frequency, resulting in a inevitable degradation of the global estimation accuracy.

Moving to an onboard PC was the only viable solution to break the **computational deadlock** encountered with the Teensy: the architecture fully supports the computational load of the EKF and, crucially, enables the installation of the software environment (SDK and proprietary drivers) essential for interfacing with the Intel camera, which is not natively supported by the microcontroller architecture. In this configuration, the Teensy is freed from high-level tasks, dedicating itself exclusively to low-level control (motor actuation and analog sensor management).

Communication between the onboard PC and the Teensy occurs via the **CAN bus** (Controller Area Network) protocol. The choice of CAN over a classic USB serial connection was motivated by the need for determinism and robustness: the high bandwidth required and the volume of data exchanged over USB tended to saturate the buffer, causing variable latencies or packet losses that would have destabilized the estimation filter.

5.2.2 Software Challenges and Drivers

The software implementation of the T265 presented significant challenges due to the device being declared EOL (**End of Life**) by Intel in 2021. The End-of-Life status created a severe integration bottleneck, as modern development environments no longer supported the device out-of-the-box [14]:

- **Linux Kernel:** Recent kernel versions do not natively recognize the device using the standard librealsense library installation. To ensure correct peripheral recognition, it was necessary to manually modify specific configuration parameters and build flags prior to the library installation process.

- **SDK Versioning:** Newer versions of the RealSense SDK have removed full support for the T265, necessitating a downgrade to previous versions for both the device firmware and the visualization software (realsense-viewer).

- **ROS2 Integration:** Official ROS2 wrappers no longer support the correct transmission of T265 odometry data. Navigating this lack of support required extensive trial-and-error, eventually forcing the adoption of antecedent wrapper libraries, the only ones still capable of interfacing with the hardware. However, these legacy libraries were incompatible with both the ROS2 communication middleware and the current Python versions installed on the system.

  To resolve this versioning conflict, a custom intermediate "bridge node" was developed using a Python version compatible with the legacy libraries. This node is responsible for acquiring raw data from the camera via the old drivers and managing their subsequent serialization and transmission over the ROS2 network, finally making the odometry data available to the algorithms developed for the EKF.

5.2.3 Data Acquisition and Reference Frame Transformation

The data acquired by the T265 camera are diverse and include both the pose (position X, Y, Z and orientation via quaternions q_x, q_y, q_z, q_w) and the corresponding velocities, both linear (v_x, v_y, v_z) and angular $(\dot{\psi}, \dot{\theta}, \dot{\varphi})$.

Unfortunately, the direct utilization of this raw data within the robot control system is not possible due to two fundamental issues related to the **reference systems**: the nature of the velocity frame (fixed vs. mobile) and the axis orientation.

Before proceeding with the mathematical formulation of the transformations, it was necessary to experimentally validate the actual orientation of the axes. The official Intel documentation and the T265 datasheet [13] presented ambiguities regarding the spatial relationship between the internal IMU frame and the optical frame (fisheye). Relying exclusively on theoretical documentation initially led to interpretation errors, resulting in inverted signs for kinematic variables.

To resolve these uncertainties, specific **empirical tests** were conducted: the camera was manually rotated along its principal axes to isolate and understand the IMU orientation and angular velocity vectors; concurrently, linear translation tests were performed to verify the directional consistency of linear velocities. This practical approach allowed for a definitive mapping of the geometric correspondence between the physical device and the digital data, confirming the necessity of the composite rotations described in the following sections.

Specifically, the experimental tests identified three distinct coordinate conventions that must be reconciled:

The T265 Tracking Frame (Pose output):

- X axis: Right
- Y axis: Up
- Z axis: Backward

The T265 Internal IMU Frame (Raw inertial data):

- X axis: Left
- Y axis: Up

- Z axis: Forward

The Standard Robot Reference Frame (Target):

- X axis: Forward
- Y axis: Left
- Z axis: Up

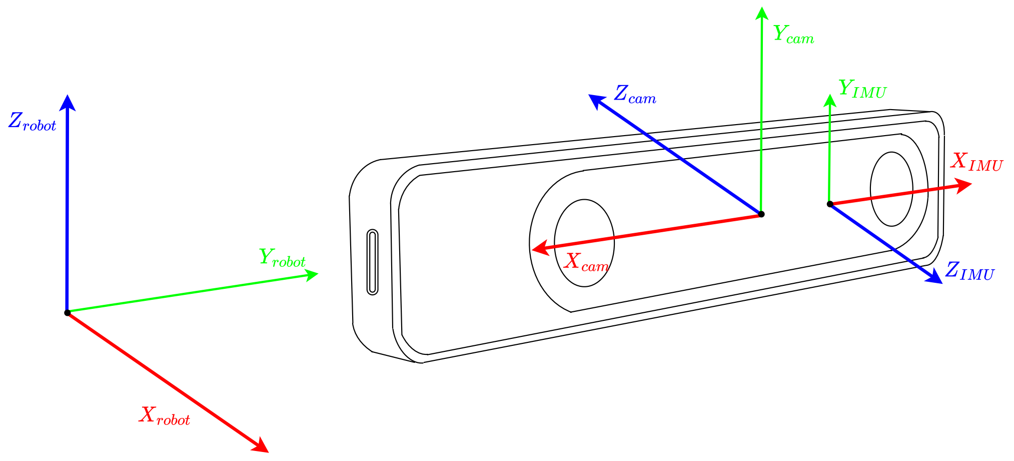

Figure 5.3. Comparison between the Robot reference frame, the T265 Tracking frame (Pose), and the T265 internal IMU frame.

Velocity Data Transformation and Axis Alignment

The first issue concerns the velocity reference system. The T265 calculates and returns linear velocities with respect to its initial fixed reference system (**World frame**), which originates at the physical power-on point of the camera.

For robot control, however, it is essential to operate in a mobile reference system integral to the vehicle (**Body frame**). Consider, for example, the longitudinal velocity v_{Bx}: in the robot's system, this must always represent the velocity in the forward direction of motion. If velocities referred to the fixed frame were used, a simple 90° rotation of the robot (yaw) would cause an axis swap, resulting in the sensor interpreting a longitudinal displacement as a transverse velocity v_{By}, and vice versa.

To resolve this issue and express velocities in the system integral to the camera, it is necessary to apply an inverse rotation using the current quaternions. Given the orientation quaternion $q_{cam} = [q_x, q_y, q_z, q_w]$, we construct the **rotation matrix** R_{cam}^{world} representing the rotation from the camera frame to the world frame. Since the opposite transformation (from world to local camera) is required, the inverse of this rotation must be calculated.

Mathematically, given the velocity in the world frame $\mathbf{v}_{world}$, the local velocity $\mathbf{v}_{local}$ is obtained by applying the inverse rotation:

$$R_{world}^{cam} = (R_{cam}^{world})^{-1} \tag{5.1}$$

$$\mathbf{v}_{local} = R_{world}^{cam} \cdot \mathbf{v}_{world} \tag{5.2}$$

Where the rotation matrix R_{cam}^{world} is derived from the quaternions according to the Rodrigues formula (implemented with SciPy libraries):

$$R_{cam}^{world} = \begin{bmatrix} 1 - 2(q_y^2 + q_z^2) & 2(q_x q_y - q_z q_w) & 2(q_x q_z + q_y q_w) \\ 2(q_x q_y + q_z q_w) & 1 - 2(q_x^2 + q_z^2) & 2(q_y q_z - q_x q_w) \\ 2(q_x q_z - q_y q_w) & 2(q_y q_z + q_x q_w) & 1 - 2(q_x^2 + q_y^2) \end{bmatrix} \tag{5.3}$$

By applying this transformation, the velocities are now correctly referred to a mobile reference system integral to the camera at each instant.

To reconcile these discrepancies, the kinematic data must be re-projected through specific rotation matrices. Based on the axis mappings identified in the previous section, to make the data consistent, a static rotation must therefore be applied to the axes. This **transformation** must be applied uniformly to linear and angular velocities, mapping the camera axes to the robot axes according to the necessary transformation.

Once the data is aligned with the robot's orientation, the physical displacement of the sensor must be addressed. The camera is mounted frontally, offset from the robot's center of rotation by a distance vector $\mathbf{d_{cam}} = [0.6685, -0.26875, 0.0]^T$ m.

According to rigid body kinematics, angular quantities are invariant across the body; thus, the angular velocity and Euler angles correctly rotate from the camera are identical to those of the robot ($\boldsymbol{\omega}_{robot} = \boldsymbol{\omega}_{camera}$ and $\text{Euler}_{robot} = \text{Euler}_{camera}$). However, linear velocities differ due to the lever arm effect. The relationship between the velocity at the robot's center and the camera is governed by the **Poisson formula**:

$$\mathbf{v}_{robot} = \mathbf{v}_{camera} - (\boldsymbol{\omega} \times \mathbf{d_{cam}}) \tag{5.4}$$

Where:

- $\mathbf{v}_{robot}$ is the velocity vector at the robot's center of rotation.
- $\mathbf{v}_{camera}$ is the velocity vector measured by the sensor (in the aligned frame).
- $\boldsymbol{\omega}$ is the angular velocity vector.
- $\mathbf{r}$ is the lever arm vector (distance from the robot center to the camera).

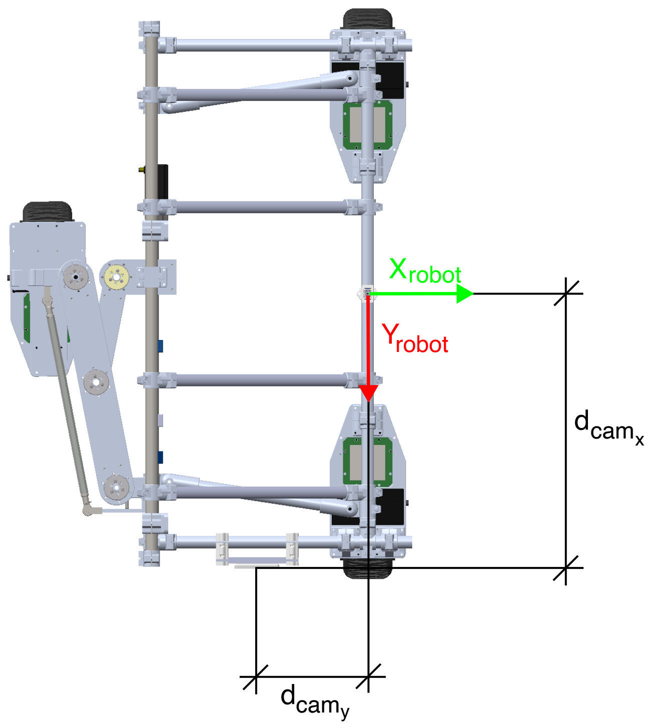

Figure 5.4. Top view of the AgriMaRo platform illustrating the geometric offset between the robot's reference frame and the camera mounting point.

Crucially, this transformation is **highly sensitive to noise** in the angular velocity measurement. Even small fluctuations in $\boldsymbol{\omega}$, when multiplied by the lever arm $\mathbf{r}$ (cross product), can introduce significant errors in the calculated linear velocity.

To mitigate this, it was decided not to perform this algebraic correction as a preprocessing step on the raw data. Instead, the **transformation** is modeled directly within the Extended Kalman Filter (**EKF**) equations. By including the lever arm effect in the measurement model, the EKF can utilize its covariance matrices to weigh the uncertainty of the angular velocity, effectively filtering the noise before it propagates to the linear velocity estimate. This ensures a more robust and accurate estimation of the robot's true state compared to a deterministic calculation.

Orientation and Frame Alignment via Quaternions

Regarding orientation estimation, the T265 provides raw data in the form of **quaternions** (q_x, q_y, q_z, q_w). The native use of quaternions was preferred over Euler angles in this processing phase to ensure the numerical stability of the system. Representations based on three sequential angles suffer from the phenomenon known as Gimbal Lock, which entails the loss of a degree of freedom when two rotation axes align parallel to each other. As extensively analyzed in the literature [16], the use of quaternions offers a singularity-free parameterization, allowing the robot's orientation to be managed in any configuration without risks of mathematical indeterminacy.

Furthermore, orientation management must compensate for geometric discrepancies between the various reference systems. In addition to the misalignment between the optical frame of the camera and the robot body, it is necessary to consider that the T265 integrates an internal IMU (Inertial Measurement Unit). The **native reference system** of this IMU is not integral with that of the lenses but possesses its own orientation defined during manufacturing. To reconcile these frames into a coherent system integral to the robot, a composite static transformation is required. Given the mounting specifications, alignment is not achievable with a single elementary rotation but requires a sequence of two rotations:

- A first rotation around the X_{IMU} axis of -90 degrees;
- A subsequent rotation around the Z_{IMU_2} axis of -90 degrees.

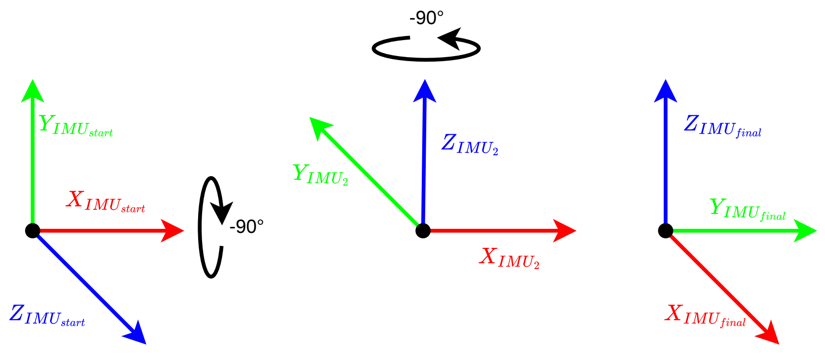

Figure 5.5. RealSense IMU rotation.

Since the second rotation occurs with respect to the new reference system generated by the first rotation (rotation on mobile or **intrinsic axes**), the mathematical composition of the quaternions follows the direct order. The total alignment quaternion q_{align} is calculated as the Hamilton product:

$$q_{align} = q_1 \otimes q_2 \tag{5.5}$$

Where:

- q_1 is the quaternion representing the first rotation;
- q_2 is the quaternion representing the second rotation.

The correct robot orientation in the global frame (q_{robot}) is therefore obtained by applying this correction to the raw sensor measurement (q_{sensor}) through the relationship:

$$q_{robot} = q_{align} \otimes q_{sensor} \tag{5.6}$$

Only at the end of this processing chain, for control and debugging purposes, is the estimated quaternion converted into the intuitive representation of **Euler**

angles (Roll ψ, Pitch θ, Yaw φ) using standard conversion formulas:

$$\begin{aligned}
\psi &= \operatorname{atan2}\left(2(q_w q_x + q_y q_z),\ 1 - 2(q_x^2 + q_y^2)\right) \\
\theta &= \operatorname{asin}\left(2(q_w q_y - q_z q_x)\right) \\
\varphi &= \operatorname{atan2}\left(2(q_w q_z + q_x q_y),\ 1 - 2(q_y^2 + q_z^2)\right)
\end{aligned} \tag{5.7}$$

Pose Transformation and Offset Compensation

Unlike the velocity transformation discussed in the previous section 5.2.3, which required a transition to the vehicle-fixed reference system, position estimation necessitates a different approach. The pose that the navigation system must estimate represents the **cumulative distance** traveled relative to a fixed reference system (**World Frame**), with its origin coinciding with the robot's geometric center (base link) at the moment of initialization. However, the T265 camera natively generates a pose referred to a fixed system but with a distinct origin, located at the camera's physical power-on point.

Since the camera is mounted with a **physical offset** relative to the vehicle's center, the raw data tracks the trajectory of the sensor rather than that of the robot. This geometric discrepancy becomes critical during rotational maneuvers: consider, for example, an in-place rotation (pure yaw). In this condition, the robot's center experiences zero linear displacement ($\Delta\mathbf{P}_{robot} = 0$), whereas the camera, located at a distance $\mathbf{r}$ from the center of rotation, traces an arc in space. As a direct result of this geometric offset, the sensor would register a fictitious position change proportional to the rotation angle, introducing a systematic error in the trajectory estimation.

To resolve this issue, a rigid body transformation must be applied to translate the tracking point from the camera to the robot's center. Defining the physical offset vector $\mathbf{r}_{body} = [0.6685, -0.26875, 0.0]^T$ meters within the robot's mobile frame, it is necessary to project this vector into the fixed reference system to subtract it from the global position. This operation is performed by multiplying the offset vector by the rotation matrix R_{cam}^{world} corresponding to the robot's instantaneous attitude:

$$\mathbf{r}_{world} = R_{cam}^{world} \cdot \mathbf{r}_{body} \tag{5.8}$$

It is crucial to note that the matrix R_{cam}^{world} used in this equation corresponds exactly to the one derived from quaternions in Equation 5.3. However, in this context, it is applied directly (without calculating its inverse). While the velocity transformation required the inverse operation to map from the global to the local frame, in this case, the objective is to project the physical offset vector (defined in the robot's Mobile frame) into the Fixed World frame; therefore, the rotation matrix is used in its direct form.

Once the rotated offset $\mathbf{r}_{world}$ is obtained, the position of the robot's center $\mathbf{P}_{robot}$ is calculated by subtracting this vector from the position detected by the camera $\mathbf{P}_{camera}$:

$$\mathbf{P}_{robot} = \mathbf{P}_{camera} - \mathbf{r}_{world} \tag{5.9}$$

Yet, the application of this formula introduces an initialization issue. At the initial instant ($t = 0$), the camera defines its own position as the origin (0,0,0). Since the robot's center is located at $-\mathbf{r}_{body}$ relative to the camera, the calculation of $\mathbf{P}_{robot}$ at start-up would result in a non-zero value (specifically, equal to the negative of the offset). To ensure that the odometry originates exactly at (0,0,0) relative to the robot's starting point, a **"zeroing"** procedure is implemented. The position calculated at the first valid frame, defined as $\mathbf{P}_{start}$, is stored and subtracted from all subsequent estimates, yielding the final published pose:

$$\mathbf{P}_{final} = \mathbf{P}_{robot} - \mathbf{P}_{start} \tag{5.10}$$

This procedure ensures that the odometry is physically consistent with the robot's center of rotation and mathematically aligned with the desired origin of the fixed reference system.

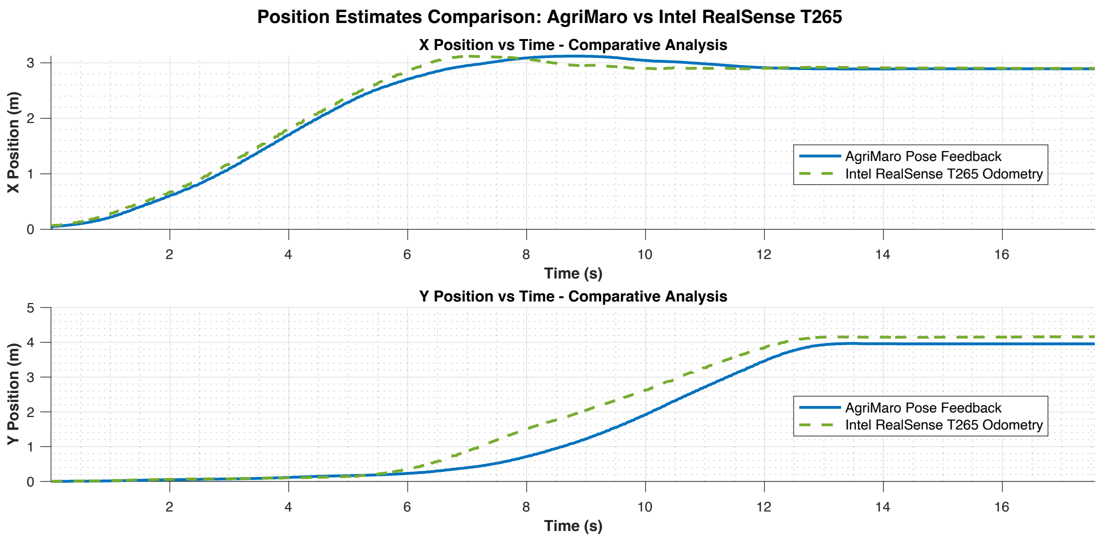

Figure 5.6. Comparison between the position estimated by AgriMaRo's kinematics and the one evaluated by the Intel RealSense T265 camera Vs Time.

Chapter 6

Final EKF Implementation and Experimental Validation

This chapter details the definitive architecture of the Extended Kalman Filter (EKF) implemented on AgriMaRo, illustrating the sensor integration required to achieve a robust estimation of the vehicle dynamics. The mathematical models underlying the filter and the final parameter tuning will be analyzed, a crucial step for optimally balancing and managing the fusion of odometric, inertial, and visual data. The objective of this processing is to provide an accurate and reliable estimation of the robot's true state during motion. The linear velocities thus estimated, and filtered from sensory noise, will serve as the essential computational baseline for the subsequent dynamic evaluations in the field, laying the theoretical and practical foundations for the slip analysis that will be addressed in the following chapter.

6.1 Final Extended Kalman Filter Design

The introduction of the Intel RealSense T265 Tracking Camera necessitated an expansion of the Extended Kalman Filter architecture to enhance estimation quality. The primary objective of this integration is to obtain a **velocity vector** that is as realistic as possible, a fundamental prerequisite for the accurate evaluation of **tire slip**.

Compared to previous iterations, the system now integrates the linear velocities ($v_{x_{cam}}, v_{y_{cam}}$), the yaw rate ($\dot{\varphi}_{cam}$), and the yaw angle (φ_{cam}) provided by the T265

Tracking Camera. These quantities were deemed essential for the consistency of the kinematic model. Although it would be theoretically possible to further **extend the measurement vector** by including the pose, angular velocities and Euler angles for roll and pitch and the linear velocity along the Z-axis this option was discarded. Including these parameters would have significantly increased computational complexity by requiring a higher number of states, and introduced severe non-linearities into the mathematical model, without providing tangible benefits for the navigation of a planar agricultural robot.

Consequently, the state vector X and the process model $f(X)$, which describes the time evolution of the system, remain defined similarly to the EKF analyzed in Chapter subsection 4.5.4, specifically:

$$X = \left[X_{pos}, Y_{pos}, \varphi, v_{Bx}, v_{By}, \dot{\varphi}, a_{Bx}, a_{By}, \delta_1, \delta_2, \delta_3, b_g\right]^T \tag{6.1}$$

$$X_{k+1} = f(X_k) = \begin{bmatrix} X_{pos_k} + T(v_{Bx_k} \cos\varphi_k - v_{By_k} \sin\varphi_k) \\ Y_{pos_k} + T(v_{Bx_k} \sin\varphi_k + v_{By_k} \cos\varphi_k) \\ \varphi_k + T\dot{\varphi}_k \\ v_{Bx_k} + T(a_{Bx_k} - \dot{\varphi}_k v_{By_k}) \\ v_{By_k} + T(a_{By_k} + \dot{\varphi}_k v_{Bx_k}) \\ \dot{\varphi}_k \\ a_{Bx_k} \\ a_{By_k} \\ \delta_{1_k} \\ \delta_{2_k} \\ \delta_{3_k} \\ b_{g_k} \end{bmatrix} \tag{6.2}$$

The measurement vector Z, however, was expanded from 10 to 14 elements to accommodate the **Visual-Inertial Odometry (VIO) data**:

$$Z = \left[\omega_1, \omega_2, \omega_3, a_{Bx}, a_{By}, \dot{\varphi}, \varphi, \delta_1, \delta_2, \delta_3, v_{x_{cam}}, v_{y_{cam}}, \dot{\varphi}_{cam}, \varphi_{cam}\right]^T \tag{6.3}$$

Regarding the observation model $h(X)$, the equations related to the pre-existing sensors (encoders, potentiometers, accelerometers) remain unchanged. However, the camera integration required specific modeling choices. The angular quantities

$(\varphi_{cam}, \dot{\varphi}_{cam})$ are treated directly, assuming their variation follows a random walk model driven by noise, as explained in Chapter subsection 4.3.3. This results in a direct equality with the state variables:

$$\begin{aligned} \varphi_{cam} &= \varphi \\ \dot{\varphi}_{cam} &= \dot{\varphi} \end{aligned} \tag{6.4}$$

Conversely, the linear velocities $(v_{x_{cam}}, v_{y_{cam}})$ require the application of the **Poisson formula** to compensate for the sensor's physical offset relative to the robot's body-fixed frame. A strategic decision was made to perform this transformation directly within the filter's observation model, rather than pre-processing the raw data during acquisition. This approach allows the EKF to correctly correlate angular velocity errors with the resulting linear velocity components, as detailed in section 5.2.3.

The applied transformation is:

$$\mathbf{v}_{cam} = \mathbf{v}_B - \boldsymbol{\omega} \times \mathbf{r}_{cam} \tag{6.5}$$

In conclusion, the non-linear measurement function $h(X)$ results in:

$$h(X) = \begin{bmatrix} \frac{(v_{Bx}-\dot{\varphi}y_1)\cos\delta_1+(v_{By}+\dot{\varphi}x_1)\sin\delta_1}{r} \\ \frac{(v_{Bx}-\dot{\varphi}y_2)\cos\delta_2+(v_{By}+\dot{\varphi}x_2)\sin\delta_2}{r} \\ \frac{(v_{Bx}-\dot{\varphi}y_3)\cos\delta_3+(v_{By}+\dot{\varphi}x_3)\sin\delta_3}{r} \\ a_{Bx} \\ a_{By} \\ \dot{\varphi}+b_g \\ \varphi \\ \delta_1 \\ \delta_2 \\ \delta_3 \\ v_{Bx}-\dot{\varphi}\cdot r_{y,cam} \\ v_{By}+\dot{\varphi}\cdot r_{x,cam} \\ \dot{\varphi} \\ \varphi \end{bmatrix} \tag{6.6}$$

Finally, the EKF algorithm linearizes these functions at each step k by computing the Jacobian matrices $H_k = \frac{\partial h}{\partial X}$ and $F_k = \frac{\partial f}{\partial X}$ to update the covariances, thereby performing the state estimation as explained in the EKF Theory chapter (section 4.1).

6.1.1 Initialization and Parameter Tuning

Like the previous models, this iteration requires adequate tuning. However, in this phase, we will delve deeper into the choices made compared to previous explanations. Having added new sensors and re-calibrated (or even replaced) some of the previous ones, it became necessary to restart the parameter optimization process from the beginning, placing greater attention on the individual dynamics.

Initial State (X_0) and Covariance (P_0)

First, the initial state of all data was reset to 0, as, explained previously (see subsection 4.4.2), the standard condition for the system start-up is with the robot stationary (**zero-start**):

$$X_0 = [0, 0, \dots, 0]^T \tag{6.7}$$

Regarding the error covariance matrix P, low parameters equal to 10^{-5} were kept for all states where the zero-start condition is valid. However, the parameters specifically calculated are those relating to the steering angles. In the microcontroller code managing AgriMaRo, the wheel start is set to zero, but given the intrinsic error generated by the mechanical steering system, there is an unmanaged "**dead band**" of ± 0.021 rad. From this value, the variance was evaluated assuming a uniform distribution:

$$\sigma^2 = \frac{(0.021 - (-0.021))^2}{12} \approx 1.35 \cdot 10^{-4} \tag{6.8}$$

Therefore, $P_{\delta_i} = 1.5 \cdot 10^{-4}$ is used.

For the yaw bias parameter (b_g), a value of 10^{-4} is used to allow slight freedom of variation from the very first instant while still keeping this parameter sufficiently stable.

Obtaining the following diagonal matrix:

$$P_0 = \text{diag}\begin{pmatrix} 10^{-5}, 10^{-5}, 10^{-5} & (X_{pos}, Y_{pos}, \varphi) \\ 10^{-5}, 10^{-5}, 10^{-5} & (v_{Bx}, v_{By}, \dot{\varphi}) \\ 10^{-5}, 10^{-5} & (a_{Bx}, a_{By}) \\ 1.5 \cdot 10^{-4}, 1.5 \cdot 10^{-4}, 1.5 \cdot 10^{-4} & (\delta_{1,2,3}) \\ 10^{-4} & (b_g) \end{pmatrix} \tag{6.9}$$

Measurement Noise Matrix (R)

Regarding the parameter R, we re-analyze each individual sensor. Consequently, for a correct evaluation of the order of magnitude, a new static test (stationary robot) was performed to calculate the **variance** generated by each single sensor of the **new suite** (as already explained in subsection 4.4.3). However, the measured values were adapted to the filter's needs, using a **specific tuning strategy** to create a correct correlation between variables:

- **Angular Velocities (ω_i):** Starting from the wheels, we set a particularly high parameter (10^{-2}) compared to the actual variance of the sensor. We deliberately increase the parameter to avoid giving too much reliability to the encoders. This is because, as explained before, even if the sensor is reliable, we do not want to impose a direct and **rigid correlation** between angular and linear velocity within the filter, in order to subsequently calculate a slip value different from zero.

- **IMU Variables:** We use 10^{-4} for accelerations (a value slightly increased compared to the measured variance). For yaw rate and yaw, $5 \cdot 10^{-3}$ and $5 \cdot 10^{-1}$ are chosen respectively. This choice is dictated by the fact that the IMU calibration, however reliable, still encounters errors; in particular, the **yaw** suffers from **drift** generated by the lack of a magnetometer.

- **Steering Encoders:** The absolute encoders for steering calculation are particularly reliable and have a very low variance; it was chosen to use 10^{-5}, a parameter that nevertheless remains slightly higher than the actual variance evaluated in the static tests.

- **Camera T265:** Parameters very close to the actual static values ($3 \cdot 10^{-6}$) were chosen. Specifically, 10^{-5} is used for linear velocities and yaw, and $5 \cdot 10^{-5}$ for the yaw rate.

We therefore obtain the following vector:

$$R = \operatorname{diag}\begin{pmatrix} 10^{-2}, 10^{-2}, 10^{-2} & (\omega_{1,2,3}) \\ 10^{-4}, 10^{-4} & (a_{Bx}, a_{By}) \\ 5 \cdot 10^{-3}, 5 \cdot 10^{-1} & (\dot{\varphi}, \varphi) \\ 10^{-5}, 10^{-5}, 10^{-5} & (\delta_{1,2,3}) \\ 10^{-5}, 10^{-5} & (v_{x_{cam}}, v_{y_{cam}}) \\ 5 \cdot 10^{-5}, 10^{-5} & (\dot{\varphi}_{cam}, \varphi_{cam}) \end{pmatrix} \tag{6.10}$$

Process Noise Matrix (Q)

We then move to the Q values. These parameters are more difficult to initialize correctly, so we started with values based on reasoning and then finely tuned them with actual field tests. The initial choice is based on how much we trust the model versus the sensors and the speed with which we want the parameters to vary. Recall that increasing Q gives more reliability to the sensors than to the model, but this leads to a "nervous" signal that follows every minimum variation (including **noise**). Decreasing Q, the model becomes predominant, but an excessive decrease leads to a result that is slowed down (**lagged**) compared to reality.

- **Pose (X, Y):** Initially 0.015 is used to have a medium value of both model reliability and speed of variation.

- **Yaw and Yaw Rate:** We use 0.08. Since we have two sensors (IMU and Camera) measuring the same quantity, the sensor result is sufficiently reliable to allow the filter to follow it quickly.

- **Linear Velocities:** We use 0.01 as we consider the velocities obtained from the camera sufficiently reliable to guide the state evolution.

- **Accelerations:** Set to 0.04 as we want to maintain a medium filter speed, giving balanced importance to both sensors and the model.

- **Steering:** We use 0.002. These values must vary slowly: steering angles physically cannot vary instantaneously, while the bias is kept low because its purpose is to converge on the mean drift over the long term, not to acquire every instantaneous variation.

- **Bias:**The bias is kept low 0.002 because its purpose is to converge on the mean drift over the long term, not to acquire every instantaneous variation.

$$Q = \text{diag}\begin{pmatrix} 0.015, 0.015, 0.08 & (X_{pos}, Y_{pos}, \varphi) \\ 0.01, 0.01, 0.08 & (v_{Bx}, v_{By}, \dot{\varphi}) \\ 0.04, 0.04 & (a_{Bx}, a_{By}) \\ 0.002, 0.002, 0.002 & (\delta_{1,2,3}) \\ 0.002 & (b_g) \end{pmatrix} \tag{6.11}$$

Signal Pre-filtering

In addition to these parameters, a **Low-Pass Filter** was used to decrease the noise generated by each sensor prior to input into the filter:

$$z_{new} = \alpha z_{prev} + (1 - \alpha) z_{raw} \tag{6.12}$$

The value of α was chosen low (0.1) to allow damping of only noise peaks, ensuring minimal delay in the acquisition of new data, since the Kalman filter already performs the main filtering task.

Final Tuning

The final calibration phase was based on the previously discussed theoretical parameters, which served as a well-established preliminary analytical reference from which the final tuning was conducted directly through physical trials and **experimental field** data acquisition.

However, although section 4.2 described a validation procedure based on Normalized Innovation Squared (**NIS**) and innovation analysis, this methodology proved not to be fully suitable for the specific characteristics of the implemented design. This discrepancy primarily stems from the design choice to exclude slip from the EKF state variables to preserve the linearity of the kinematic model. This decision necessitated the use of values for the measurement noise covariance matrix R, regarding the steering angles, that are significantly higher than the sensors' nominal variance.

Consequently, in the innovation covariance formulation $S = HPH^T + R$, the R term becomes predominant. This phenomenon leads to a saturation of the variable S, which ceases to reflect the evolution of the error covariance matrix P, making the NIS index (Equation 4.14) no longer a reliable indicator for monitoring filter consistency. In light of these considerations, the filter's performance was verified through a direct analysis of the **states** estimated in the field. Specifically, the **consistency** of the estimates was evaluated against the actual vehicle dynamics, verifying the absence of systematic biases, the stability of the obtained values, and the correlation between the state variables and the raw data acquired from the sensors.

Following a series of experimental trials based on the analysis of actual field results, the optimal configuration was achieved. While the initial covariance vector P and the low-pass filter constants remained unchanged, the Q and R matrices were defined as follows:

$$R = \operatorname{diag}\begin{pmatrix} 0.05, 0.05, 0.05 & (\omega_{1,2,3}) \\ 0.001, 0.001 & (a_{Bx}, a_{By}) \\ 0.001, 0.01 & (\dot{\varphi}, \varphi) \\ 0.00001, 0.00001, 0.00001 & (\delta_{1,2,3}) \\ 0.00005, 0.00005 & (v_{x_{cam}}, v_{y_{cam}}) \\ 0.001, 0.0005 & (\dot{\varphi}_{cam}, \varphi_{cam}) \end{pmatrix} \tag{6.13}$$

$$Q = \operatorname{diag}\begin{pmatrix} 0.5, 0.5, 0.08 & (X_{pos}, Y_{pos}, \varphi) \\ 0.08, 0.08, 0.15 & (v_{Bx}, v_{By}, \dot{\varphi}) \\ 0.12, 0.12 & (a_{Bx}, a_{By}) \\ 0.002, 0.002, 0.002 & (\delta_{1,2,3}) \\ 0.002 & (b_g) \end{pmatrix} \tag{6.14}$$

6.2 Experimental Validation

Analyzing the results obtained through the EKF tuning just described, attention will focus on the longitudinal and lateral velocities, as well as the yaw rate. Specifically, we will evaluate the trend of the filter's estimates compared to the raw values derived directly from the sensors. Furthermore, the innovations related to these states will be analyzed to fully understand and justify the chosen tuning strategy. To achieve a comprehensive validation, it was decided to examine a **complex test** involving curves and forward/backward driving in both directions.

First, the longitudinal velocity trend is evaluated.

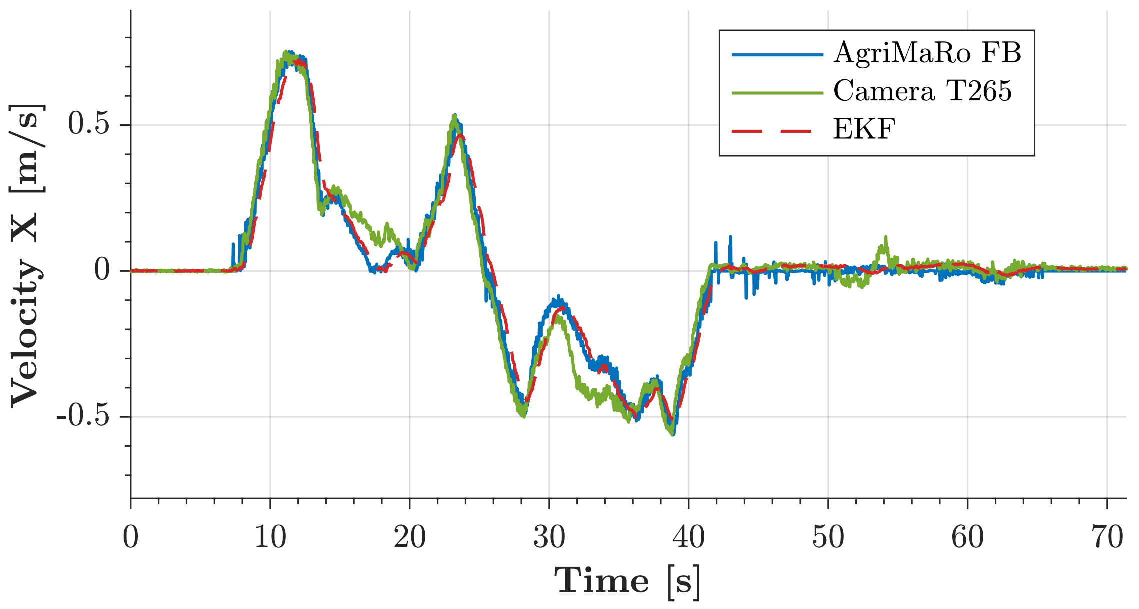

Figure 6.1. Comparison between the longitudinal velocity estimated by the EKF, the one derived from forward kinematics (AgriMaRo FB) and the measurements recorded by the T265 tracking camera at its mounting position.

Figure 6.1 shows that the velocity estimated by the EKF closely follows the real dynamics: the filter primarily relies on the velocity detected by the camera, but appropriately detaches from it when the latter provides erroneous readings. Moreover, especially during rotations, it can be noted how the reference frame transformation performed inside the EKF works significantly better than the first tested implementation, where this transformation was applied directly to the raw data (accumulating errors due to the use of non-optimized velocity and yaw rate values). Since this test was conducted on tiles, it is also notable that the linear velocity calculated by the EKF does not deviate significantly from the one estimated via encoder odometry. This behavior is exactly what is expected on rigid surfaces where slip remains low; a dynamic that, as will be shown later, changes radically on yielding terrains. Figure 6.2 allows for the analysis of the **innovations** related to

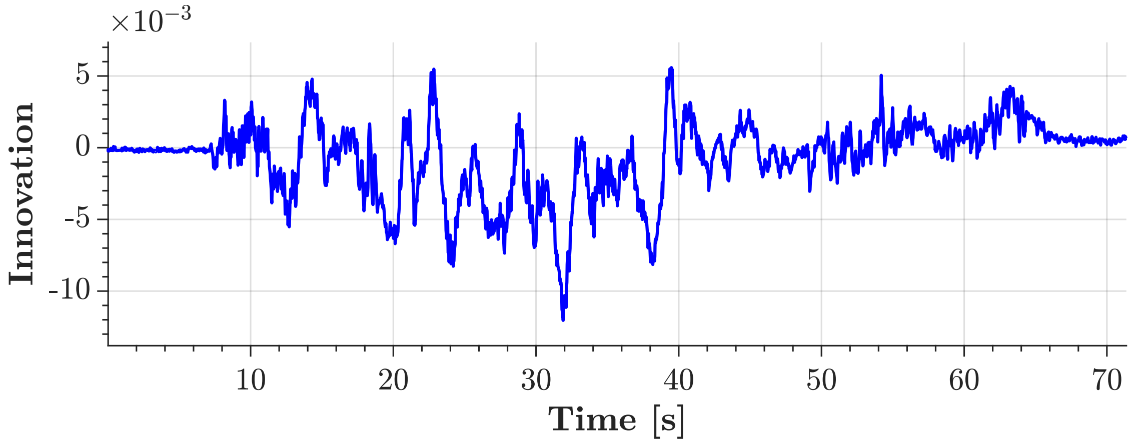

Figure 6.2. Innovation of longitudinal velocity.

longitudinal velocity. From these data, an innovation variance of $7{\cdot}10^{-6}$ is obtained, a value lower than the set measurement noise covariance parameter ($R = 5 \cdot 10^{-5}$). This means that the filter is estimating the measurement noise conservatively: the innovations (i.e., the difference between the actual measurement and the filter's prediction) are smaller than the expected noise. This is a positive result, as it indicates that the filter is not underestimating the sensor noise and does not risk being too "optimistic" toward potentially flawed measurements. In practice, the filter trusts the measurements slightly less than it theoretically could, thus avoiding instability and oscillations due to unmodeled dynamics. Conversely, if the variance were higher than R, it would mean the filter is underestimating the noise, becoming too sensitive to raw readings and risking divergence. Finally, it is crucial to note that the innovations maintain a stable value around zero, without presenting any **drift**; this guarantees that the EKF does not introduce bias and does not force the estimates toward values unrepresentative of physical reality.

Moving on to the lateral velocity analysis, the previous considerations remain valid. Figure 6.3 also clearly highlights the correct functioning of the transformation of the velocities estimated by the camera. The variance of the innovations Figure 6.4 is $4 \cdot 10^{-6}$, again confirming to be lower than the parameter $R = 5 \cdot 10^{-5}$.

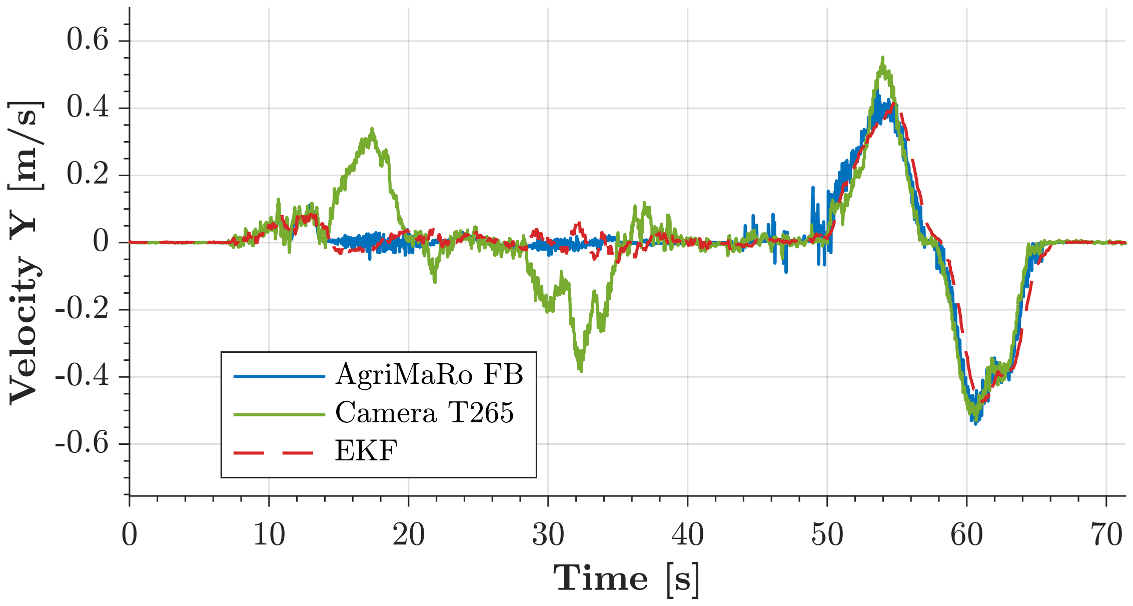

Figure 6.3. Comparison between the lateral velocity estimated by the EKF, the one derived from forward kinematics (AgriMaRo FB) and the measurements recorded by the T265 tracking camera at its mounting position.

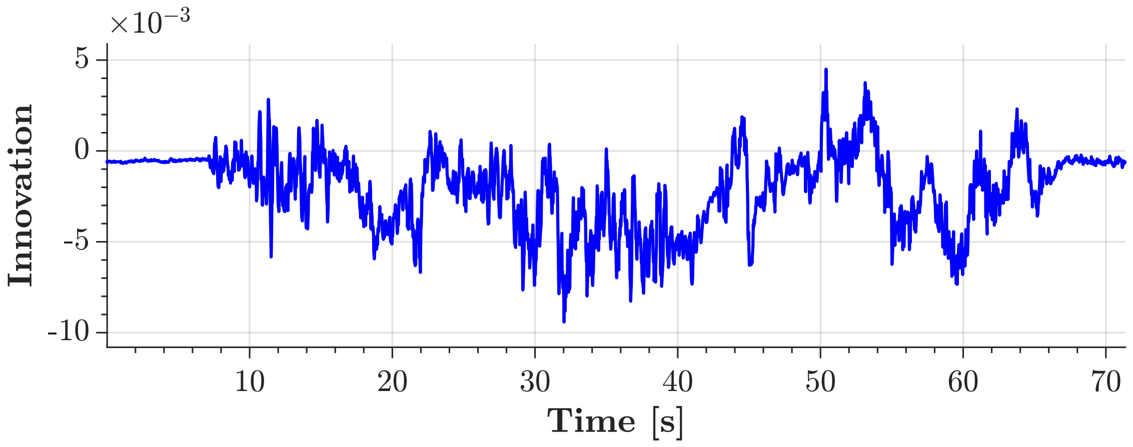

Figure 6.4. Innovation of lateral velocity.

Finally, the yaw rate was analyzed. The same premises apply to this state

Figure 6.5: the EKF performs excellent filtering, allowing the acquisition of much cleaner and more representative values compared to the raw measurements. Figure 6.6 confirms a zero mean, with no drift, and their variance is $3.81 \cdot 10^{-4}$, well below the parameter set for the yaw rate ($R = 1 \cdot 10^{-3}$).

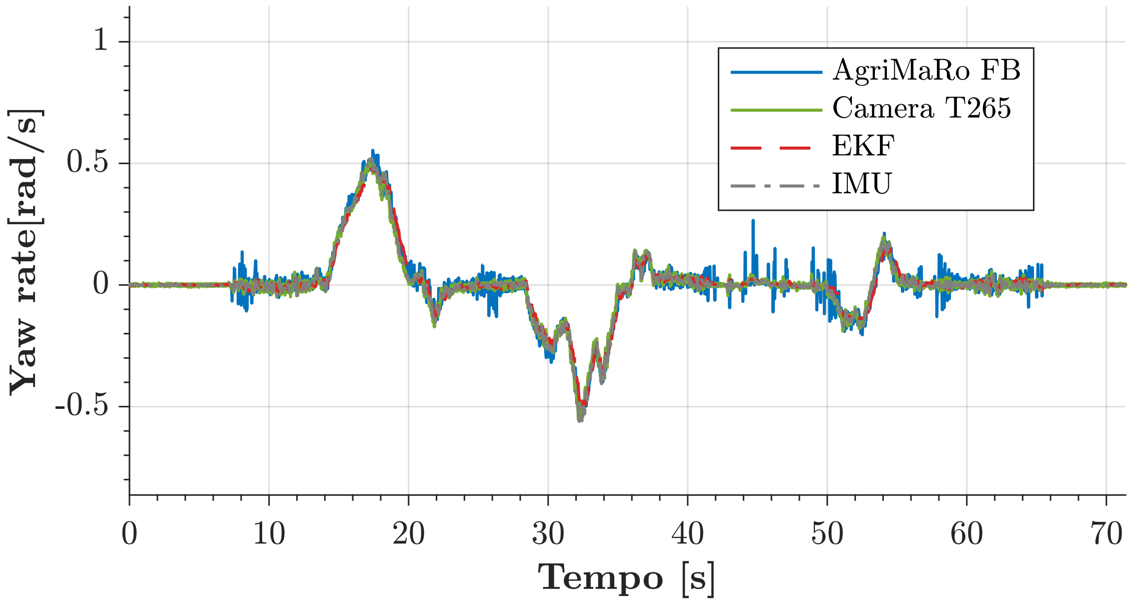

Figure 6.5. Comparison between the yaw rate estimated by the EKF, the one derived from forward kinematics (AgriMaRo FB), the measurements recorded by the T265 tracking camera at its mounting position and the IMU data.

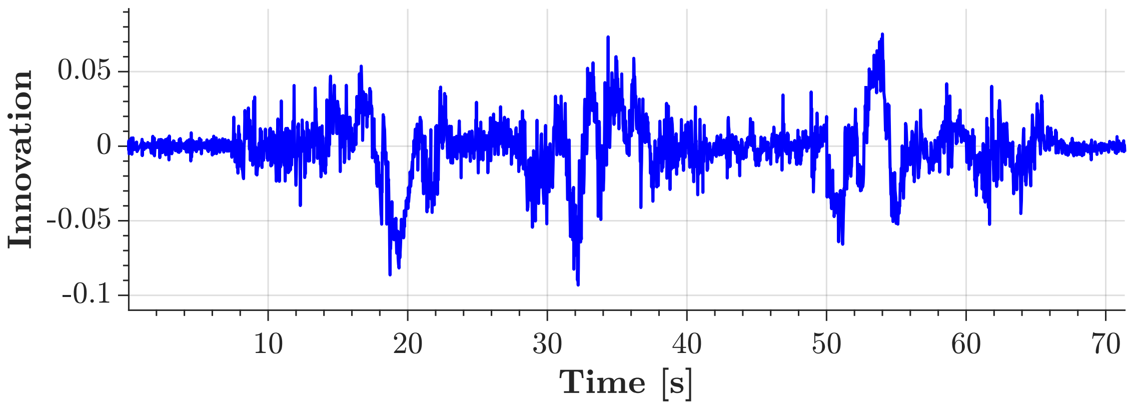

Figure 6.6. Innovation of yaw rate.

It can therefore be deduced that the current tuning is appropriate and robust for all the analyzed variables. The filter exhibits **conservative** behavior, the innovations are physically consistent with the set measurement noise, and there is no risk of underestimating sensory uncertainty. Although there is theoretically the possibility of reducing the R parameters to obtain a filter even more reactive to transients, such a choice would risk overestimating measurement precision, drastically reducing the overall robustness of the system. Instead, the adopted values provide a **stable** and reliable filter, characterized by a responsiveness adequate for our rover's requirements.

Chapter 7

Slip Evaluation and Experimental Validation

This chapter is dedicated to the analysis and estimation of the longitudinal slip of the drive wheels, which represents one of the main objectives of this thesis work. Leveraging the noise-filtered velocity estimates obtained through the previously described Extended Kalman Filter, the chapter is divided into three fundamental sections.

The first part will present the theory underlying slip kinematics and the related mathematical formulation used for its calculation. Subsequently, the practical implementation of the algorithm developed to estimate the slip of each individual wheel in real-time will be detailed. Within this section, the technical issues encountered and their respective resolutions will be extensively discussed.

Finally, the last section will present the experimental validation of the system. The trend of the slip calculated by the algorithm during various test campaigns conducted on multiple types of terrain will be analyzed, thus evaluating the effectiveness of the code and the dynamic interaction of the robot with the operating environment.

7.1 Theory and Kinematics of Longitudinal Slip

In section 2.3, during the definition of AgriMaRo's basic kinematic model, the simplifying assumption of **pure rolling** was made. In an ideal condition, the linear velocity of the wheel center perfectly matches its peripheral tangential velocity; however, in real-world operations on agricultural terrain (characterized by deformability, mud, or poor traction), this assumption no longer holds. The physical interaction between the tire and the ground generates micro-slips and deformations that cause a discrepancy between the rotational motion imparted by the motors and the actual advancement of the chassis.

To quantify this discrepancy, reference is made to international vehicle dynamics standards, specifically the definitions introduced by the **SAE** (Society of Automotive Engineers) [4]. The SAE defines longitudinal slip, denoted by the letter s, as a dimensionless parameter describing the relative difference between the peripheral **tangential velocity of the wheel** and the actual **linear velocity** along the longitudinal axis of travel.

To **normalize** the slip value within a physically interpretable range (between -1 and +1) and, simultaneously, to avoid mathematical singularities in the control software (such as divisions by zero), the standard formulation requires distinguishing two dynamic scenarios: the traction phase and the braking phase [15]. The approach consists of always placing the predominant quantity at that specific moment in the denominator.

7.1.1 Slip in Traction (Acceleration)

When the peripheral velocity of the wheel exceeds the vehicle's forward velocity ($\omega_i R_i > v_{ix}$), the robot is in the traction phase. The slip is normalized with respect to the **product** of the **wheel's rotational velocity** and the **rolling radius**:

$$s = \frac{\omega_i r_i - v_{ix}}{\omega_i r_i} \tag{7.1}$$

Under this condition, s assumes positive values ranging between 0 (pure rolling, where $\omega_i r_i = v_{ix}$) and 1 (complete slip or skidding). The value of 1 occurs in extreme situations, for example, when the robot is physically blocked ($v_{ix} = 0$) but the wheel spins freely immersed in mud.

7.1.2 Slip in Braking (Deceleration)

When the vehicle's forward velocity is greater than the wheel's peripheral velocity ($\omega_i r_i < v_{ix}$), the robot is in the braking or strong deceleration phase. In this case, the slip is normalized with respect to the wheel's linear velocity:

$$s = \frac{\omega_i r_i - v_{ix}}{v_{ix}} \tag{7.2}$$

In this scenario, s assumes negative values ranging between 0 and -1. The lower limit is reached in a locked wheel condition ($\omega_i = 0$), where the tire stops rotating but the vehicle continues to move forward due to inertia, sliding on the ground.

The physical parameters used in these equations are defined as follows:

- ω_i: angular velocity of the individual $i - th$ wheel, measured directly by the encoders aboard the swerve drive modules [rad/s];
- r_i: rolling radius, equal for all wheels [m]
- v_{ix}: actual longitudinal linear velocity of the center of the wheel [m/s]. In the context of this thesis, v_{ix} does not represent the theoretical kinematic command sent to the motors, but rather the real and filtered data estimated instant by instant.

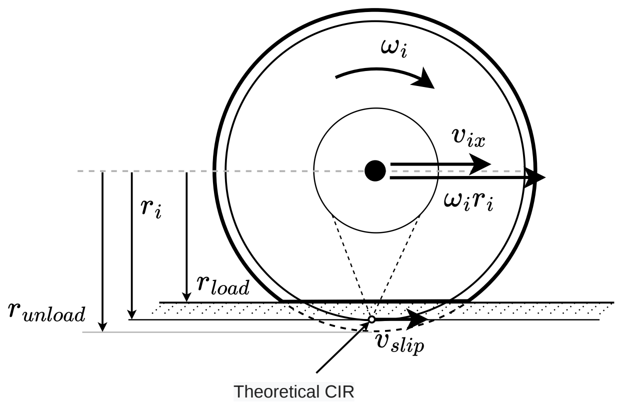

Figure 7.1. Wheel subjected to longitudinal slip.

The adoption of this standardized and computationally robust metric allows for the continuous evaluation of the AgriMaRo platform's traction efficiency, providing a mathematically safe indicator to analyze the **robot's dynamic interaction** on different types of agricultural terrain.

7.2 Software Implementation and Real-Time Calculation

As described in the theoretical discussion, the purpose of the algorithm is to evaluate the longitudinal slip in **real-time** and independently for each wheel. To achieve this goal, it is strictly necessary to derive the local longitudinal velocity of the i-th wheel starting from the global state of the vehicle.

The Extended Kalman Filter outputs the linear velocities of the chassis along the main axes (v_{Bx} and v_{By}) and the yaw rate ($\dot{\varphi}$). To transfer these velocities from the center of the robot (origin of the body frame) to the center points of the individual wheels, the fundamental law of rigid body kinematics is used. Having defined the position vectors of the wheel centers with respect to the origin of the chassis, which for AgriMaRo are $\mathbf{d}_1 = [-0.5075, 0.0000]^T$, $\mathbf{d}_2 = [0.5075, 0.0000]^T$ and $\mathbf{d}_3 = [0.0000, -0.7560]^T$ (expressed in meters), the local velocity components $v_{x\_wheel,i}$ and $v_{y\_wheel,i}$ are calculated as follows:

$$v_{x\_wheel,i} = v_{Bx} - \dot{\varphi} \cdot d_{y,i} \tag{7.3}$$

$$v_{y\_wheel,i} = v_{By} + \dot{\varphi} \cdot d_{x,i} \tag{7.4}$$

The components thus obtained are oriented parallel to the axes of the robot's reference frame. Therefore, it is necessary to project this resulting vector along the **actual longitudinal axis** of advancement of the single wheel, which depends instant by instant on its steering angle δ_i. This transposition occurs through the following equation:

$$v_{long\_wheel,i} = v_{x\_wheel,i} \cdot \cos(\delta_i) + v_{y\_wheel,i} \cdot \sin(\delta_i) \tag{7.5}$$

The term $v_{long\_wheel,i}$ thus represents the actual longitudinal linear velocity at the center of the wheel, which is fundamental for evaluating the denominator (or numerator) of the SAE formula. Concurrently, the peripheral tangential velocity resulting from the rotation of the motors is calculated:

$$v_{ang\_wheel,i} = \omega_i \cdot r_i \tag{7.6}$$

Where r_i is the dynamic rolling radius and ω_i is the angular velocity. It is fundamental to emphasize that the value of ω_i is extracted directly from the raw readings of the **encoders**. As justified in the previous chapters, the choice was made not to use the filtered angular velocity downstream of the EKF to avoid a mathematical correlation (coupling) with the estimation of the linear velocity, which would alter and invalidate the relative slip calculation.

7.2.1 Signal Conditioning and Safety Logic

Following the purely kinematic calculation, the practical implementation of the code required the introduction of specific conditioning logic to ensure data reliability under all operating conditions.

Firstly, the result of the slip equation was subjected to strict **saturation** within the range of -1 and $+1$. Although the split SAE formulation inherently prevents divisions by zero, this saturation acts as a software safety catch. Should this parameter feed into subsequent control loops (such as the Traction Control system), an anomalous peak due to a temporary numerical error would result in unrealistic torque requests to the motors, compromising the stability and physical safety of the platform.

A second limitation was introduced to manage the **stationary vehicle** condition. A deadband was implemented, forcing the slip to a value of 0 if both the calculated longitudinal velocity ($v_{long\_wheel,i}$) and the peripheral velocity from the encoders ($\omega_i \cdot r_i$) fall below a pre-established minimum **threshold** of 0.05 m/s. At near-zero velocities, the intrinsic noise of the encoders and the camera would cause the slip calculation to fluctuate erratically, losing any physical meaning. This threshold ensures a clean, zero output when the robot is considered stationary.

Finally, the implementation of the SAE formula's denominator was slightly adapted by evaluating the maximum between the **absolute values** of the two reference velocities. The denominator was therefore coded as:

$$\text{denom} = \max\left(|\omega_i \cdot r_i|, |v_{long\_wheel,i}|\right) \tag{7.7}$$

This variation does not alter the slip calculation in any way as long as the robot maintains wheel steering angles between ±90° while proceeding in the forward direction. Conversely, when these geometric conditions are not met, the use of absolute values entails a slight loss of physical veracity in the precise calculation of the slip. However, this choice was deliberately implemented to allow a high-level traction control algorithm (based on a PID controller) to correctly interpret and manage the slip parameter, considering all the complex kinematic conditions and direction reversals generated by the robot's **omnidirectionality**. The validity and detailed motivations behind this specific evaluation will be explained more thoroughly in subsection 8.3.2.

7.2.2 Resolution of Sensor Issues in Transient Phases

During the initial experimental validation campaigns to test the validity of the code, a non-negligible issue related to the dynamic nature of the sensors emerged. Although the slip was evaluated correctly in steady-state conditions (constant velocity), during **high-dynamic transient phases**, particularly during sudden accelerations from a standstill, the value returned by the algorithm proved to be incorrect.

Specifically, it was observed that the estimation of the angular velocity derived from the encoders was visibly **phase-delayed** compared to the longitudinal linear velocity detected by the visual odometry (camera) and mediated by the EKF. This discrepancy generated transients in which the wheel velocity was mathematically lower than the robot's forward velocity, resulting in negative slip peaks during full traction. Since this represents a physical impossibility (the vehicle cannot accelerate faster than the wheels pushing it), a hardware and software troubleshooting process became necessary.

The investigation focused on the behavior of the ODrive Pro drivers controlling the hub motors. It was found that, inside the driver, the signals generated by the Hall effect encoders (inherently affected by low resolution) undergo heavy integrated low-pass filtering. This operation is essential to linearize the acquisition and ensure smooth torque control at low speeds, but it inevitably introduces a time delay in the estimation of the peripheral velocity.

Initially, an attempt was made to reduce the intensity of this internal filter to minimize the delay and prioritize signal responsiveness. However, the physical limits of the installed Hall effect encoders generated such quantization noise that the ODrive quickly returned error signals, interrupting the correct functioning of the system. Consequently, it proved impossible to adopt filtering parameters that deviated significantly from the default ones provided by the manufacturer, unless a future hardware replacement with higher-performance encoders is planned.

Having established the need to operate with the lowest filtering setting tolerable by the drivers without incurring errors, the resolution involved a software workaround focused on signal **synchronization**. The strategy, while representing a slight engineering compromise regarding the system's ideal responsiveness, consisted of deliberately slowing down the data coming from the visual odometry to align it with the delay introduced by the encoders.

To achieve this phase alignment, adjustments were made to the parameters of the **digital low-pass filter** based on an Exponential Moving Average (EMA), already implemented in the software architecture and described in the section 6.1.1, governed by law:

$$z_{new} = \alpha z_{prev} + (1 - \alpha) z_{raw} \tag{7.8}$$

After several testing iterations, the quantities acquired by the camera were heavily smoothed by raising the damping coefficient to $\alpha = 0.2$. Concurrently, the opposite action was taken on the RPM sensors, removing the software-level EMA filtering and entrusting the cleaning of their signal solely to the ODrive driver.

This "re-tuning" operation sacrificed a fraction of the experimental **responsiveness** of the visual data, but it allowed for the realignment of the time constants of the two measurement subsystems. Thanks to this precaution, the algorithm is now able to compute a physically consistent slip value free from fictitious inversions, even during the most violent transients.

7.2.3 Experimental System Validation

This section presents the results derived from the longitudinal slip calculation. The primary objective of the analysis is to evaluate the slip magnitude under steady-state conditions across various terrain types. A secondary goal is to determine whether the implemented filtering techniques effectively mitigate the noise and anomalies associated with rapid transient phases, or if replacing the current RPM sensors will be necessary for future developments.

To provide a clear framework for the analysis, the testing methodology was standardized. The robot was commanded to follow a purely straight-line trajectory along its positive x_{robot} axis. The imposed velocity profile is characterized by an initial acceleration of 1 m/s^2 followed by a constant steady-state cruising speed of 1 m/s maintained until the end of the run.

To assess whether the slip estimation is realistic and consistent under varying grip conditions, the experiments were conducted on three distinct surfaces: tiled pavement, grass, and firm soil with loose gravel. Although numerous preliminary tests were performed to tune the system parameters, the data and graphs presented in this section refer to the final validation runs. Specifically, they illustrate a single representative run for each terrain, acquired using the optimally tuned filtering parameters.

Initially, a test was conducted on a **tiled** surface Figure 7.2. Besides being commonly found inside greenhouses, this type of flooring is also highly representative of the dynamic behavior observed on asphalt.

The collected data clearly show that the steady-state slip maintains a mean value deviating by a maximum of **0.01 from zero**. This outcome accurately reflects the physical phenomenon anticipated for these testing conditions: on rigid, **non-deformable surfaces** like dry asphalt or tiles, the wheel's kinematics closely approximates the **pure rolling** assumption. Forward motion is, in fact, ensured by adhesion forces and the local elastic deformation between the tire tread and the surface micro-roughness, while soil deformation is entirely negligible. Because the ground does not yield under the robot's weight or driving torque, the wheels' angular velocity translates almost entirely into longitudinal linear velocity [20].

Regarding the transient phase induced by the initial acceleration, the data display values that, while mathematically bounded, reach a maximum slip peak of

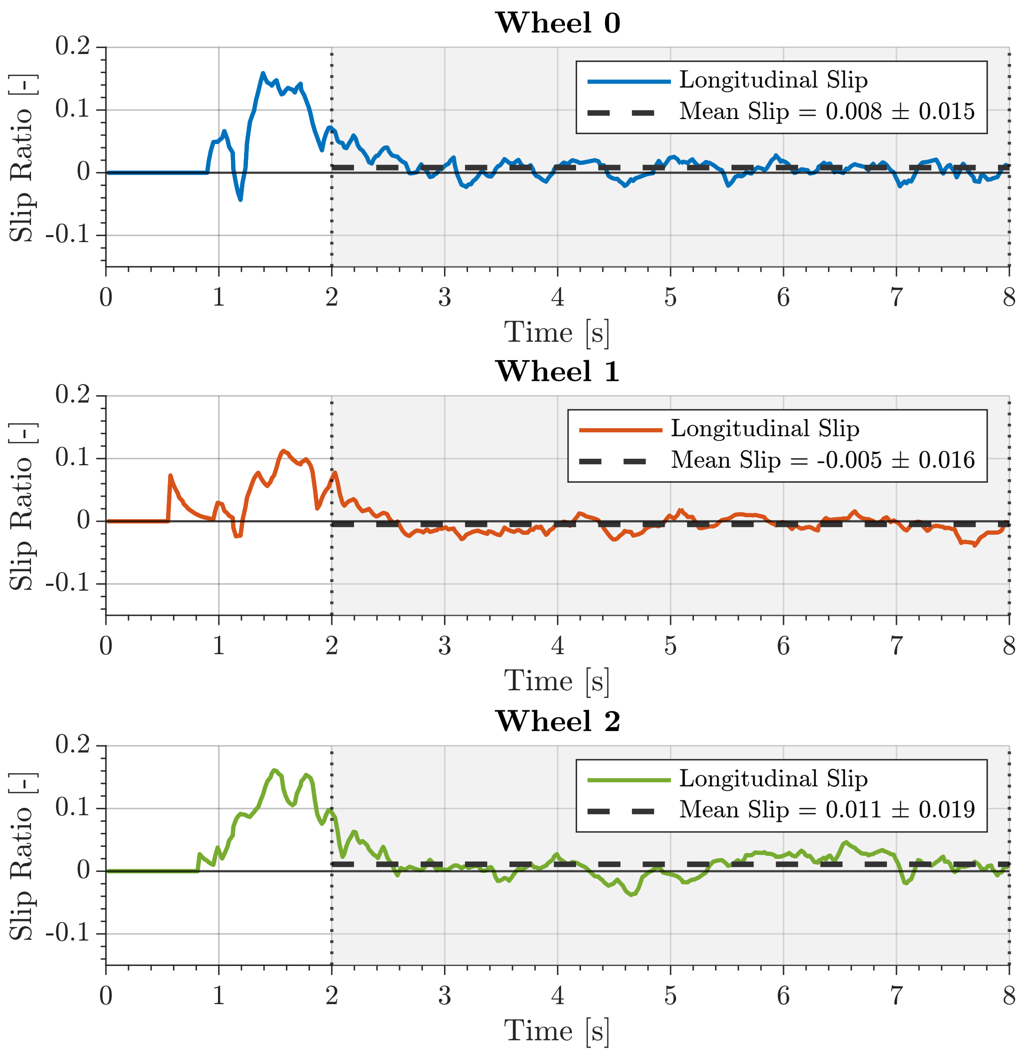

Figure 7.2. Longitudinal slip on tiled surface ($v_{cmd} = 1m/s$).

0.16. According to fundamental tire dynamics literature, on dry and rigid surfaces, slip values between 0.10 and 0.20 typically correspond to the saturation region of the longitudinal force [9]. In this region, the tire operates near its maximum adhesion limit, a condition strictly characteristic of severe acceleration or aggressive braking maneuvers. However, the experimental run performed by the rover involved an acceleration phase designed solely to reach a cruising speed of 1 m/s. A peak slip of 16% is therefore an extremely high value. While this magnitude could be partially representative of the specific dynamic conditions imposed by the test protocol, it must be noted that this value is also heavily influenced by the engineering solutions implemented to estimate velocities during highly transient phases. Due to the limited resolution of the current sensors at low speeds and high accelerations, the filtering techniques applied to the raw data prevent us from having absolute certainty regarding the exact magnitude of this peak. Consequently, the adoption of higher-performance encoders is strongly recommended to minimize estimation errors and obtain reliable ground-truth data during these specific transient maneuvers.

Subsequently, the longitudinal slip evaluation was performed on terrains with characteristics radically different from the previous one; specifically, tests were conducted on grass and on firm soil with loose gravel. The purpose of these tests was indeed to evaluate whether the system could reliably perceive the physiological increase in slip caused by these surfaces.

The dynamics change radically when the robot transits over these terrains characterized by a **high coefficient of compliance**. As described in vehicle dynamics literature for off-road locomotion [9], on these surfaces, the concept of surface friction loses its validity. For the robot to advance, the tire lugs must penetrate the top layer of the soil; the rotation of the wheel thus uses a portion of its energy to literally push a volume of soil backward. Because the ground yields plastically under this thrust, the wheel is forced to undergo a greater rotation compared to the actual linear distance traveled by the chassis. This physiological "free slip" is correctly captured by the algorithm, which records a structural increase in the mean slip ratio.

Regarding the longitudinal slip on **grass** Figure 7.3, it was found that the mean steady-state slip ranges between **11.6% and 13.5%**. These values represent a more than plausible slip for a test conducted on slightly damp grass, where the wheel needs to tear or flatten the vegetation before finding a rigid layer to gain the necessary grip for advancement. Furthermore, in this test, it can be noted that the maximum slip obtained during acceleration is 33.9%, which is a more than reasonable value for the physical conditions under which the test was performed.

As for the analysis on **firm soil with loose gravel** Figure 7.4, it can be observed that the mean steady-state slip settles between **12.2% and 14.6%**. Even under these conditions, a high slip is noted and, as previously explained, this is due

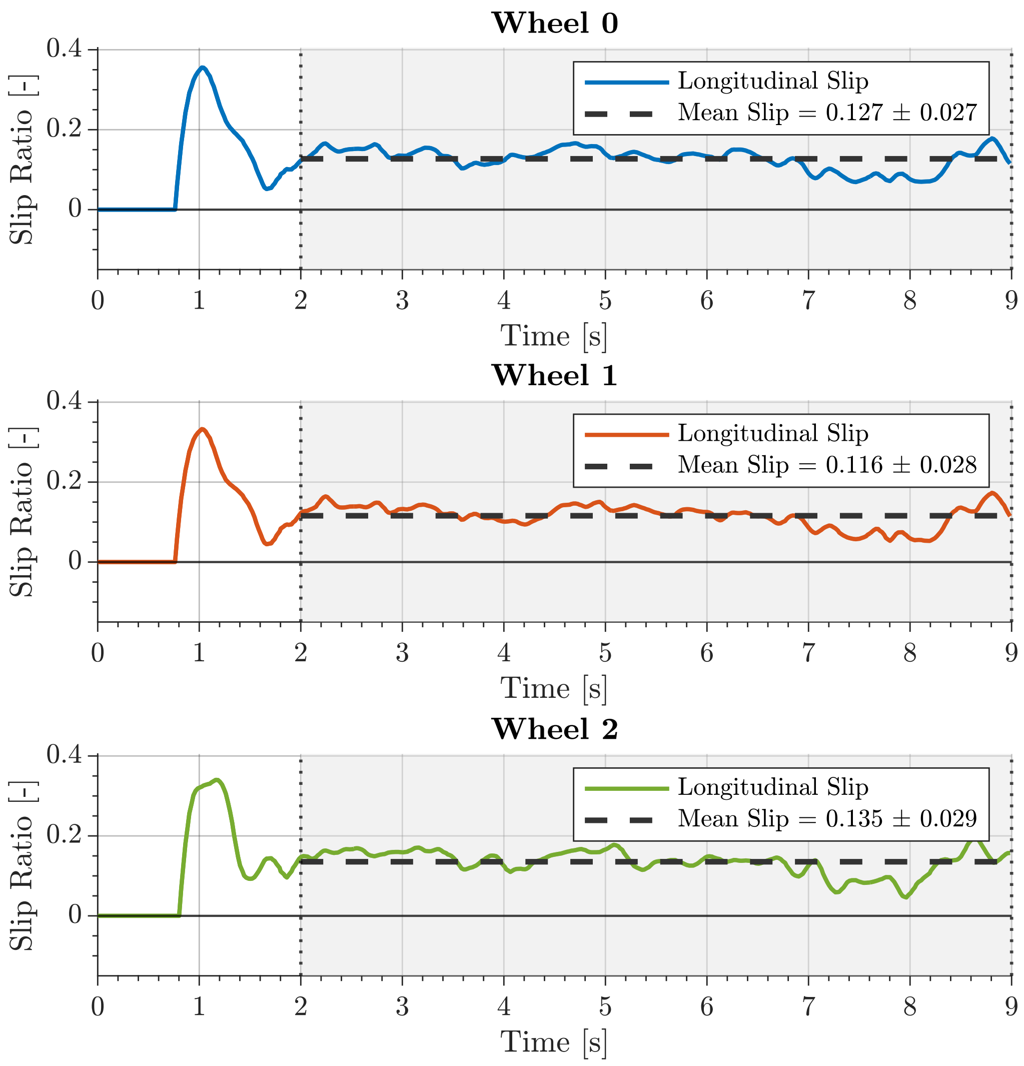

Figure 7.3. Longitudinal slip on grass surface ($v_{cmd} = 1m/s$).

to the fact that the wheels need to displace the underlying stones before reaching the non-yielding ground. Considering therefore that in the area where the test was conducted the wheels did not have the possibility to sink deeply, but only the need to displace the first incoherent layer of stones, these estimates are highly accurate. Additionally, the values reached during acceleration peak at 47.5%. According to off-road tire models [9], such macroscopic slip values are entirely valid and justified for soil conditions, as high transient slip is mechanically required to overcome the low shear strength of the unpaved surface and generate the maximum longitudinal

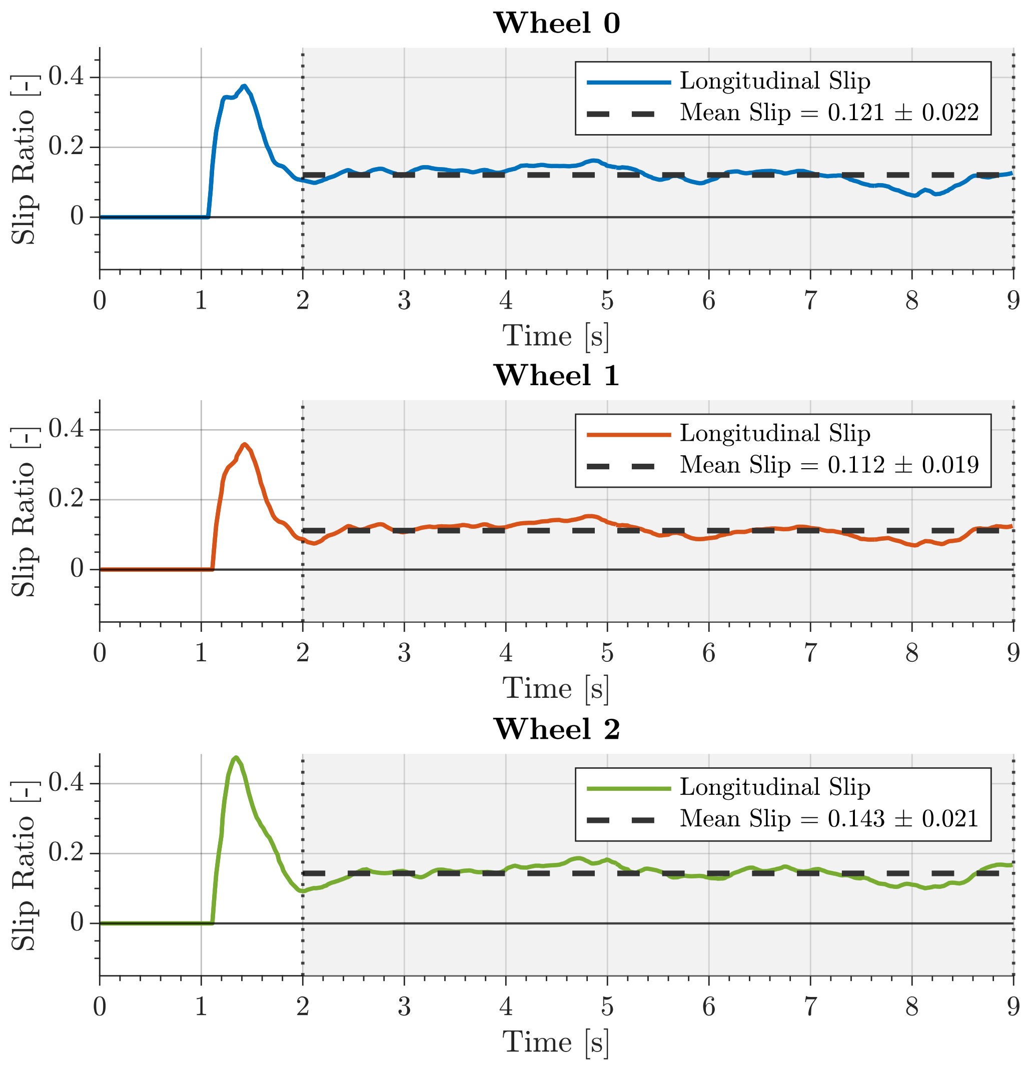

Figure 7.4. Longitudinal slip on Firm Soil with Loose Gravel surface ($v_{cmd} = 1m/s$).

thrust.

In conclusion, it can therefore be deduced that the slip value evaluated by the system represents the real values in the steady-state phases very well, but that the transition phases must be better studied and refined to reach the same conclusion. A final point of attention regarding the obtained results is that they were processed with a low-pass filter to ensure greater visual and analytical clarity. The time constant of this filter needs to be increased or decreased depending on the purpose for which the slip data must be used; in particular, the filtering can be strongly decreased if the goal is to have a highly reactive PID control, but an excessive decrease would lead to highly noisy readings and non-credible results at certain instants. Consequently, this value must necessarily be tuned concurrently with the development of the actual traction control loop.

Chapter 8

Adaptive Traction Control Architecture

In this final chapter, the effects and potential derived from the optimization of kinematic data obtained through the previously described Extended Kalman Filter (EKF) will be analyzed. Specifically, it will first be evaluated how the acquisition of this high-fidelity data constituted a highly reliable input for a terrain recognition and classification software developed by a parallel research group, thus allowing for extremely high recognition rates across the various test scenarios conducted.

Subsequently, the direct advantages brought about by such environmental awareness will be analyzed, illustrating how soil classification can be translated into an ideal kinematic parameter (Desired Slip) for the implementation of an efficient traction system.

The discussion will then move to the practical application of these variables, presenting the theoretical study and an initial logical implementation of a PID controller for the active regulation of slip. This control architecture will strictly take into account the constraints and kinematic peculiarities imposed by the omnidirectionality of the robotic platform.

The ultimate goal of this analysis is to verify and demonstrate that the developed system represents a solid foundation for the implementation of a definitive traction control system, capable of dynamically adapting the vehicle's behavior to various operating conditions while, at the same time, maximizing energy consumption efficiency.

8.1 Data Optimization and Terrain Recognition

The accurate calculation of slip and the optimization of sensory acquisition through the Extended Kalman Filter (EKF), as described in previous chapters, do not merely represent an analytical achievement in their own right; rather, they constitute the fundamental prerequisite for developing high-level control architectures. Indeed, a primary objective of this thesis work was to provide an extremely clean and reliable data foundation (or ground truth).

This volume of high-fidelity data served as the direct input for parallel research aimed at the **real-time estimation** and classification of the traversed **terrain**. As documented in the literature, increasing terrain awareness is an enabling technology for mobile robot autonomy in agricultural and off-road contexts. The classification approach adopted is based on the assumption that physical measurements derived from the direct wheel-terrain interaction are intimately linked to the mechanical properties of the surface. Specifically, the dynamic parameters estimated by the sensors and optimized through the EKF act as true mechanical descriptors of the soil. Utilizing these signals allows the robot to physically "feel" the surface moment by moment, providing a level of tactile and dynamic information that complements and enriches the vehicle's global perception.

For **Machine Learning** algorithms to extract a reliable "signature" from these physical quantities, the quality of the source signals is indispensable. The EKF-optimized data, integrated with additional physical variables such as the current absorbed by each wheel, were analyzed by a second research group [26]. The goal was to generate a classifier capable of interpreting the terrain, grouping the surfaces into **clusters** based primarily on the level of stiffness or deformability of the soil. Considering this metric, compact and paved surfaces represent the low-difficulty extreme, as they offer excellent traction and low compressibility. Conversely, yielding terrains, and particularly sandy ones, pose significantly greater propulsive challenges. Currently, the artificial intelligence system adopts a classification based on eight distinct terrain classes.

The synergy between the data acquisition optimization developed in this thesis and the robust algorithmic architecture implemented by the artificial intelligence group has led to excellent results. By processing the signals, the system extracted an extensive feature set to feed a classifier based on a **Support Vector Machine** (SVM). For training, an experimental dataset was employed, consisting of 208 labeled samples obtained from 2-second non-overlapping windows, rigorously extracted during steady-motion phases. The classifier was then trained and evaluated using **cross-validation** to ensure maximum robustness and generalization capability across different terrain realizations. The results demonstrate a solid and consistent discrimination ability among all the eight analyzed terrain classes.

In conclusion, the excellent accuracy achieved by the classification software cross-validates the effectiveness of the estimation architecture developed in this work. From a vehicle control perspective, the success of this real-time recognition paves the way for adaptive traction control. Knowing the exact terrain difficulty combined with the robot's physical parameters makes it possible to mathematically calculate the Desired Slip parameter in real time. This deduction represents the missing link for maximizing platform efficiency and autonomy, and will be formalized in the following section.

8.2 From Terrain Recognition to "Desired Slip"

Identifying the driving surface in real time gives the robot a fundamental tactical advantage, but the system must integrate this information into the traction control logic to turn it into a real performance increase. The ultimate goal of terrain awareness is not simple classification, but optimizing Tractive Efficiency.

As chapter 7 extensively discussed in the experimental results analysis, off-road vehicle dynamics follow the principle that a universally optimal slip value does not exist. The curve describing the traction coefficient as a function of slip features an absolute maximum point, which the literature defines as the **Traction Peak Point** (TPP). Operating near this point means maximizing the forward thrust while simultaneously minimizing the energy wasted on deforming the soil or spinning the tires uselessly.

Terramechanics, and particularly the **soil's shear strength** and reaction index, strictly determine the positioning of the TPP on the slip axis. On compact and rigid pavements (like asphalt), the traction peak occurs at very low slip rates because elastic friction guarantees propulsion. Conversely, on plastic and yielding terrains (such as mud, loose soil, or grass), the vehicle requires physiologically higher slip

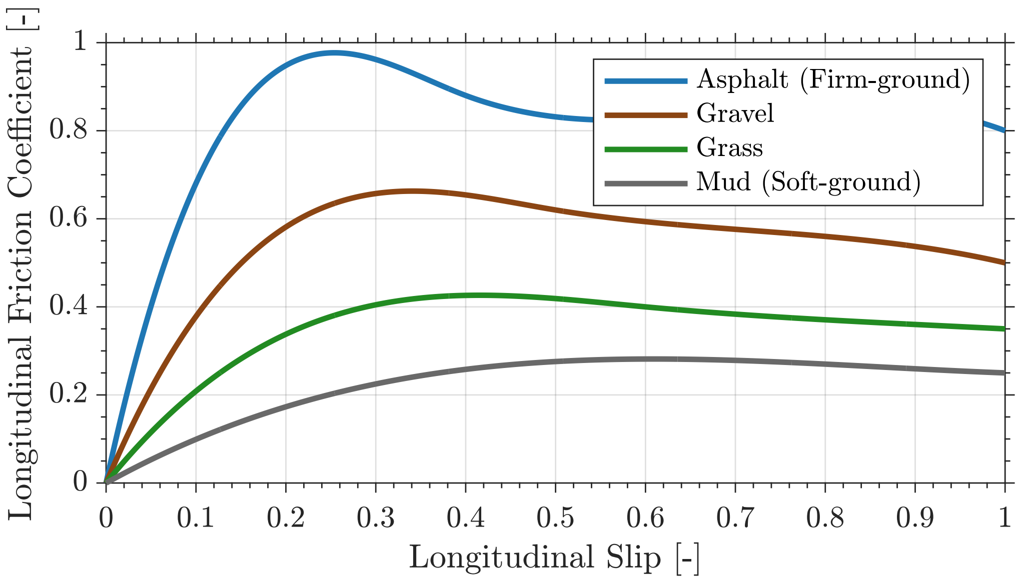

Figure 8.1. Representative trends of the longitudinal friction coefficient ($\mu_x = F_x/F_z$) as a function of longitudinal slip for different terrain conditions.

rates (often exceeding 20-30%) to allow the tire lugs to penetrate the ground, generate the necessary shear stress, and push the robot forward [9].

By exploiting the output of the artificial intelligence algorithm described in the previous section, AgriMaRo's control architecture can perform a fundamental transition: moving from the measured "Real Slip" to the "Desired Slip" (or Target Slip).

From an implementation standpoint, once the classifier returns the label regarding the traversed terrain's difficulty, the system accesses an **internal mapping** to extract the optimal slip value associated with that specific condition. This mapping relies on empirical terramechanic models, the robot's design specifications (such as mass, load distribution, and geometric parameters), and the robot's dynamic behavior (like traction, braking, or steady state).

The dynamic knowledge of the Desired Slip represents a turning point for the rover's autonomy. Maintaining the slip around this **target**, while also considering the need to remain within the robot's controllable handling region, yields two crucial benefits:

- **Sinkage Prevention:** It prevents the motors from delivering excessive torque on low-bearing terrains, stopping the robot from digging in and getting stuck.

- **Energy Optimization:** Regulating the wheel rotation speed to avoid exceeding the TPP drastically reduces power dissipation. By exploiting the maximum possible traction for the specific terrain, the system consequently maximizes the battery pack's autonomy.

Defining and making the Desired Slip calculable conceptually closes the environmental estimation and analysis phase. The system now possesses both the current slip value (which the EKF calculates) and the optimal reference value. This duality of information forms the algorithmic basis for closing the feedback loop [19], which the PID controller implementation illustrated in the following section will manage.

8.3 Architecture and Feedback Loop Closure

The final logical and engineering step to create an intelligent propulsion system consists of closing the **feedback loop**. By utilizing the vehicle's instantaneous slip (the process variable, which the EKF architecture described in chapter 7 calculates in real time) and the ability to compute the theoretical optimal slip (the terrain-derived setpoint), the system can instantly quantify a dynamic error, mathematically defined as:

$$e(t) = s_{\text{desired}} - s_{\text{real}} \tag{8.1}$$

To cancel this error and force the robot to operate constantly under maximum tractive efficiency conditions, the software architecture incorporates a **Proportional-Integral-Derivative (PID) controller**. The task of this regulator is to translate the kinematic error into an **active** modulation of the **speed commands** sent to the individual wheels' ODrive drivers.

The following paragraphs first analyze the theory behind this control structure, and then detail its initial practical implementation, defining the operational conditions required to ensure it functions correctly.

It is essential to clarify that the final prototype developed here does not represent a fully tuned and optimized traction control system for outdoor use; rather, it constitutes a **preliminary implementation** that requires a precise methodological abstraction. To test the control loop's robustness while isolating it from any temporary uncertainties the classification phase might introduce, and considering the current development stage of the terrain-based Desired Slip lookup table, the system does not yet dynamically inject this parameter based on the neural network's output. Instead, the user sets it manually as a configuration variable.

This operational choice made it possible to verify that the controller firmware successfully reads the data processed by the EKF, calculates the feedback error, and independently modulates the direct output to the motors for each swerve module. Even when operating with basic tuning and a manual input parameter for the desired slip, the resulting values and behaviors do not aim to maximize absolute field performance. Instead, they demonstrate the full validity, stability, and responsiveness of the developed software structure.

Future project developments can execute a real-world field tuning, accompanied by a quantitative analysis of the system's effectiveness, as soon as the artificial intelligence algorithm's data enables the real-time evaluation of the Desired Slip parameter.

8.3.1 PID Controller Theory

The Proportional-Integral-Derivative (PID) controller is a feedback loop control mechanism widely used in industrial control systems [17]. The operating principle of this algorithm relies on calculating an error value, defined as the difference between the measured process variable and the desired response (or setpoint). The controller then continuously attempts to minimize this error by appropriately adjusting the control input sent to the process.

The PID calculation algorithm involves three constant parameters: the proportional term (P), the integral term (I), and the derivative term (D), whose values one can interpret as a function of time. The following mathematical equation describes the controller's output in the time domain, $Y(t)$:

$$Y(t) = e(t)K_P + K_I \int_0^t e(t)dt + K_D \frac{de(t)}{dt} \tag{8.2}$$

Where the involved variables represent the following:

- e: Error signal.
- K_P: Proportional constant.

- K_I: Integral constant.
- K_D: Derivative constant.

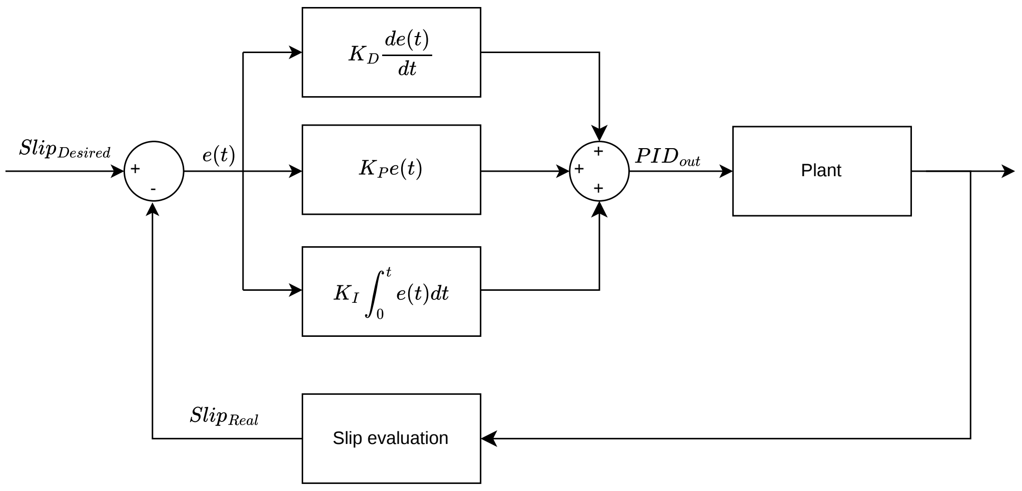

Figure 8.2. PID diagram.

Each of these three control actions evaluates the error from a different temporal perspective:

- **The Proportional Action (K_P)** depends exclusively on the **current error** present in the system.
- **The Integral Action (K_I)** relies on the accumulation of past errors over time, guaranteeing the elimination of the **steady-state error**.
- **The Derivative Action (K_D)** represents a prediction of the future error, relying on the current **rate of change** (derivative) of the **error** itself.

The system uses the weighted sum of these three actions to regulate the process through a control element, such as the power supplied to a motor.

Although the PID is the most common controller due to its simplicity and applicability, the literature highlights some inherent disadvantages in its classical form. These include the possibility of generating an unwanted velocity overshoot, a sluggish response in the event of sudden torque load variations, and a high sensitivity to the K_I and K_P gains. Furthermore, this controller's performance heavily depends on the accuracy of the system models and their parameters.

This sensitivity to parametric variations and non-linearities justifies the need for careful methodologies for **tuning** the constants. This tuning is a fundamental step to apply this theory to the actual modulation of the commands directed to the robot's motors.

8.3.2 Traction Control Implementation for Omnidirectional Vehicles

Having defined the theory behind the regulator, we can now delve into the actual implementation of this control principle. Although the theoretical structure of the PID is inherently linear, applying it practically to an omnidirectional vehicle (equipped with fully steerable wheels) requires imposing specific operational constraints. The system needs these constraints to disable the controller's intervention in situations where it would be harmful and to ensure consistent traction management across all possible driving configurations.

First, we must define the actual operational scope of the control: the system does not aim to unstick the vehicle from completely bogged-down situations (a condition where the PID action would achieve nothing, as the wheels would spin freely regardless of the speed command sent to the drivers). Instead, the objective is preventive: it limits slip to prevent the robot from reaching such a critical condition, while simultaneously utilizing a traction value that optimizes energy consumption.

Based on this premise, we introduced two velocity-based parameters. The first is a threshold parameter called $v_{low\_speed\_threshold}$. This value represents the **minimum velocity** of each individual **wheel** below which the algorithm bypasses the slip error calculation by setting the error to zero. By setting this **threshold** at 0.05 m/s, we prevent the PID from calculating and applying a control output when we consider the robot virtually stationary. This architectural choice prevents the system from injecting erroneous commands generated solely by sensor noise at low speeds. We also implemented a final verification based on the magnitude of the overall chassis velocity (and not just the longitudinal velocity of the single wheel): the system applies the PID output to the motors only if the robot's **global velocity** exceeds the predetermined **threshold**. This acts as a redundant safety measure to guarantee stationarity when required.

Once the robot meets these enabling conditions, the system calculates the slip error, from which the controller derives the PID$_{out}$ value (according to the characteristic equation defined in the previous section). The architecture then transmits this output to the Teensy board via the CAN bus and adds it to the theoretical angular velocity value, which the low-level kinematic control evaluates. The result is the **final velocity command** that the ODrive drivers actually receive as input:

$$\omega_{\text{ODrive}} = \omega_{\text{low-level}} + \text{PID}_{\text{out}} \tag{8.3}$$

To ensure this algebraic sum leads the vehicle to converge toward the desired slip, regardless of the driving direction dictated by its **omnidirectionality**, we developed an algorithm to evaluate the dynamic state, dividing it into several logical processes. The first step consists of calculating the current slip of the $i-th$ wheel:

$$s_{\text{real},i} = \frac{\omega_i \cdot r_i - v_{long\_wheel,i}}{denom_i} \tag{8.4}$$

To prevent sign inversion resulting from negative velocities, the algorithm calculates the denominator using the absolute values of the kinematic quantities as anticipated in the subsection 7.2.1:

$$denom_i = \max(|\omega_i \cdot r_i|, |v_{long\_wheel,i}|) \tag{8.5}$$

Next, the system evaluates the **dynamic condition of the single wheel** using a parameter called Power$_{\text{state}}$, which operates independently of the absolute direction of motion of the vehicle along its axes. We calculate this parameter by multiplying the individual wheel's acceleration by its longitudinal velocity:

$$\text{Power}_{\text{state},i} = a_{local_i} \cdot v_{long\_wheel,i} \tag{8.6}$$

The sign of this parameter tells us whether the wheel is in **traction**, **braking**, or **steady state**. Specifically, a Power$_{\text{state},i} > 0$ means the wheel pushes in the exact same direction it is already moving, creating a Traction condition. A Power$_{\text{state},i} < 0$ means the wheel pushes opposite to its current direction of travel, resulting in a

Braking condition. Finally, if $\text{Power}_{\text{state},i} = 0$, we are in a steady state. By comparing the $\text{Power}_{\text{state},i}$ value against a threshold parameter named accel_threshold, the algorithm identifies three distinct operational conditions. Each condition requires a specific s_{desired}.The optimal s_{desired} mainly depends on two interdependent factors:

- The terrain type (which dictates the shape of the Pacejka grip curve).
- The dynamic condition (traction, braking, or constant velocity).

As a result, we can encounter the following operational conditions:

- **Traction Condition ($\mathbf{Power}_{\mathbf{state},i} > \mathbf{State}_{\mathbf{threshold}}$):**The goal is to maximize thrust while maintaining **stability**. We choose the s_{desired} near the peak of the traction curve: we keep it far enough from the chaotic slip zone to guarantee full directional control, yet close enough to ensure **highly effective traction**. This position represents the optimal compromise between performance and stability during acceleration. Next, we calculate the sign of this desired slip by multiplying it by the sign of the wheel's longitudinal velocity:

$$s_{\text{desired},i} = \text{sgn}(v_{long\_wheel,i}) \cdot s_{\text{target\_traction}} \tag{8.7}$$

- **Braking Condition ($\mathbf{Power}_{\mathbf{state}} < -\mathbf{State}_{\mathbf{threshold}}$):** The goal here is to maximize deceleration without completely locking the wheel (thus avoiding a loss of control). We position the s_{desired} at the peak of the braking curve, which naturally features a higher absolute value compared to the traction phase. This choice allows us to brake in the shortest possible distance while still retaining sufficient grip. Just as we did for traction, we evaluate the final desired slip by applying the correct sign (inserting a negative sign to the braking target if the initial value was positive):

$$s_{\text{desired},i} = \text{sgn}(v_{long\_wheel,i}) \cdot s_{\text{target\_braking}} \tag{8.8}$$

- **Constant Velocity Condition ($-\mathbf{State}_{\mathbf{threshold}} \leq \mathbf{Power}_{\mathbf{state}} \leq \mathbf{State}_{\mathbf{threshold}}$):** In this scenario, the objective is not to generate maximum traction, but rather to maintain **cruising speed**. The s_{desired} takes on a minimal value, providing exactly what we need to balance external resistances (such as friction and aerodynamics) without triggering any acceleration or deceleration. We obtain the final value using the formula:

$$s_{\text{desired},i} = \text{sgn}(v_{long\_wheel,i}) \cdot s_{\text{target\_steady}} \tag{8.9}$$

Relying directly on the sign of the wheel velocity ($\text{sgn}(v_{\text{long}})$) guarantees that our error calculation maintains total physical consistency across any driving configuration.

Verification of the Omnidirectional Control Logic

To validate the effectiveness and stability of the described calculations under omnidirectional conditions, we analyzed specific case studies. For the purpose of this analytical demonstration, we assume a simplified proportional gain ($K_p = 1.0$) and set the derivative and integral values to 0. Additionally, we fix the desired slip (s_{desired}) at 0.1 for traction and -0.15 for braking. For this analysis, we also assume a unit rolling radius ($r = 1$ m) to establish a direct correlation between the peripheral velocity ωr (in m/s) and the angular velocity ω (in rad/s). While this simplification does not reflect the actual dimensions of the robot, it facilitates the interpretation of the **control logic's** response during this explanation. We chose the variables v_{long} and ωr_i randomly to obtain the characteristic trend, while maintaining the consistency necessary to quantify the ODrive's reaction.

Case 1A: Forward driving in traction with excessive slip

Under this condition, the robot pushes forward ($v_{\text{long}} = +0.6$ m/s), but the wheel rotates too fast ($\omega r = +0.70$ m/s).

- **Slip calculation:** We obtain $\frac{0.70-0.6}{0.70} = +0.1429$ (positive, indicating forward slip).
- **Desired target:** During the traction phase, starting from a base target of 0.1, we calculate $s_{\text{desired}} = (+1) \cdot 0.1 = +0.1$.
- **Error and PID:** We evaluate the error as $e = 0.1 - 0.1429 = -0.0429$. Consequently, we obtain a $\text{PID}_{out} = -0.0429$.

- **Physical effect:** We add the negative PID output to the theoretical wheel velocity (0.7 rad/s). This reduces the final velocity sent to the ODrive, bringing the corresponding ωr_{ODrive} to 0.6571 rad/s. The wheel slows down its excessive rotation, lowering the real slip and allowing the system to correctly converge toward the +0.1 target.

Case 2A: Backward driving in traction with excessive slip

The robot accelerates backward ($v_{\text{long}} = -0.6$ m/s), and the wheel rotates excessively in that direction ($\omega r = -0.7$ m/s).

- **Slip calculation:** We obtain $\frac{-0.7-(-0.6)}{0.7} = -0.1429$ (negative, as it represents traction congruent with backward motion).
- **Desired target:** In backward traction, we set $s_{\text{desired}} = (-1) \cdot 0.1 = -0.1$.
- **Error and PID:** We calculate the error as $e = -0.1 - (-0.1429) = +0.0429$. The output is therefore $\text{PID}_{out} = +0.0429$.
- **Physical effect:** By adding a positive quantity to a negative theoretical velocity (-0.7 rad/s), we obtain a less negative final command (-0.6571 rad/s). The motor reduces its backward rotation, the magnitude of the peripheral velocity decreases, and the slip converges to the -0.1 target.

Case 2D: Backward driving in braking with excessive slip

The robot is moving backward ($v_{\text{long}} = -0.5$ m/s) but is braking sharply, causing the wheel to rotate very slowly ($\omega r = -0.39$ m/s).

- **Slip calculation:** We obtain $\frac{-0.39-(-0.5)}{0.5} = +0.22$ (strongly positive, indicating high opposition to motion).
- **Desired target:**In backward braking, assuming a braking target of 15%, we set $s_{\text{desired}} = (-1) \cdot -0.15 = 0.15$.
- **Error and PID:** We calculate the error as $e = 0.15 - 0.22 = -0.07$. The output is $\text{PID}_{out} = -0.07$.

- **Physical effect:** By adding the negative PID_{out} to the negative theoretical command (-0.39 rad/s), we obtain a more negative final command (-0.4600 rad/s). The motor accelerates its backward rotation, reducing the excessive braking force and realigning the slip to the predicted target.

The dynamic behaviors deduced for all possible vector combinations confirm the reliability of the implemented logic and are comprehensively summarized in the Table.

Table 8.1. Summary analysis of traction control operational cases.

Case	**Dir.**	**Man.**	$\mathbf{v}_{\text{long}}$ **[m/s]**	$\boldsymbol{\omega}$**r [m/s]**	**Slip**	**Desired**	**Error**	**PID**	$\boldsymbol{\omega}_{\text{ODrive}}$ **[rad/s]**
1A	Forward	Traction	+0.6	+0.7	+0.143	+0.1	-0.043	-0.043	+0.657
1B	Forward	Traction	+0.6	+0.63	+0.048	+0.1	+0.052	+0.052	+0.682
1C	Forward	Braking	+0.5	+0.47	-0.060	-0.15	-0.090	-0.090	+0.380
1D	Forward	Braking	+0.5	+0.39	-0.220	-0.15	+0.070	+0.070	+0.460
2A	Backward	Traction	-0.6	-0.7	-0.143	-0.1	+0.043	+0.043	-0.657
2B	Backward	Traction	-0.6	-0.63	-0.048	-0.1	-0.052	-0.052	-0.682
2C	Backward	Braking	-0.5	-0.47	+0.060	+0.15	+0.090	+0.090	-0.380
2D	Backward	Braking	-0.5	-0.39	+0.220	+0.15	-0.070	-0.070	-0.460

Chapter 9

Conclusion

The work carried out in this thesis marks a fundamental evolution for the AgriMaRo omnidirectional platform, transforming it from a system based on ideal kinematic assumptions to a vehicle truly aware of the terrain it moves on. To achieve this goal, the first essential step was to optimize the acquisition of physical data. Since the initial sensor equipment was not enough to describe the complex dynamics of the wheel-terrain interaction, **new sensors** were carefully integrated, preferring visual-inertial odometry over GPS due to the severe reception limits typical of greenhouses. The fusion of all these signals required the development and rigorous tuning of an **Extended Kalman Filter** (EKF). Specifically, the filter architecture was redesigned by defining a 12-state model. This choice guaranteed the numerical stability of the system and optimized a series of variables necessary for a correct evaluation of the vehicle's dynamics. In this way, it was possible to obtain a set of optimized data, filtered from noise, and above all, synchronized to the same frequency. However, to support this level of data processing and fusion, the original hardware of the vehicle was completely inadequate. Therefore, it was necessary to install and configure a dedicated **onboard PC** from scratch, a hardware integration that made the real-time execution of the algorithm possible.

Once this extremely solid and synchronized database was obtained, two parallel and complementary paths opened up. On the one hand, the EKF output became the ideal input for a **Support Vector Classifier** (SVC), developed in synergy with another research group, which successfully verified the accuracy in recognizing eight different classes of terrain stiffness. On the other hand, this same estimation architecture made it physically possible to calculate the real **longitudinal slip** for each single wheel, effectively managing phase delays in transients and providing a fundamental parameter for traction.

Having both soil recognition and the exact measurement of instantaneous slip available represented the starting point for implementing a **traction control system**. The logic that was initially structured is based precisely on the error generated between the "desired slip", which varies according to the terrain and dynamic conditions, and the real slip just measured. In this first phase, the goal was to obtain a correct algorithmic behavior, capable of managing the specific features of AgriMaRo's omnidirectionality. However, we chose not to perform a deep tuning of the controller: to do so with scientific rigor, a dataset, or a complete mapping, that associates the exact targets of desired slip with the different environmental combinations is still missing.

Looking at **future perspectives**, the first algorithmic goal will be precisely the implementation of this dataset to close the traction control loop, allowing a definitive tuning of the PID, combined with a stricter validation of the slip evaluated during very low-speed transients. Finally, practical field experience highlighted some structural criticalities that the control software can no longer ignore. Specifically, the mechanical play related to the opening system of the third leg and the inevitable dead band of the steering system frequently lead the vehicle to accumulate an unwanted yaw that is not commanded by the joystick. Solving this directional drift by developing a high-level controller dedicated to active trajectory correction will be the decisive step to make AgriMaRo fully autonomous and reliable in the real agricultural world.

Bibliography

[1] Ritik Agarwal, Ghanishtha Bhatti, R. Raja Singh, V. Indragandhi, Vishnu Suresh, Laura Jasinska, and Zbigniew Leonowicz. Intelligent fault detection in hall-effect rotary encoders for industry 4.0 applications. *Electronics*, 11(21), 2022.

[2] Francesco Amodio. Design of the locomotion system for a new robot for precision agriculture in greenhouses. Master's thesis, Politecnico di Torino, 2024.

[3] Boulaid Boulkroune, Kurt Geebelen, Jia Wan, and Ellen van Nunen. Autotuning extended kalman filters to improve state estimation. In *2023 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–6. IEEE, 2023.

[4] Christopher R Carlson and J Christian Gerdes. Calculating longitudinal wheel slip and tire parameters using gps velocity. In *Proceedings of the 2001 American Control Conference. (Cat. No. 01CH37148)*, volume 3, pages 1800–1805. IEEE, 2001.

[5] Giovanni Colucci, Andrea Botta, Luigi Tagliavini, Lorenzo Baglieri, Simone Duretto, and Giuseppe Quaglia. AGRIMARO.Q, A Service Robot for Precision Agriculture in Greenhouses. In *The International Conference of IFToMM ITALY*, pages 292–299. Springer, 2024.

[6] Paolo S. Crovetti. *Electronic Systems for Vehicles - Antilock Braking System (ABS) and Electronic Stability Program (ESP).* Politecnico di Torino, 2024. Lecture Notes: 10_ABS and ESP.

[7] FAO. *The future of food and agriculture: Alternative pathways to 2050.* Food and Agriculture Organization of the United Nations, Rome, Italy, 2018.

[8] Giancarlo Genta. *Motor Vehicle Dynamics: Modeling and Simulation.* World Scientific, Singapore, 2003.

[9] Giancarlo Genta and Lorenzo Morello. *The Automotive Chassis: Volume 1: Components Design.* Springer Science & Business Media, New York, NY, 2009.

[10] Jeemoni Gogoi, J. P. Hzaraika, Utpal Barman, and Nivedita Deka. Comparative study of input use, productivity and profitability of hybrid and traditional rice cultivation in assam, india. *Economic Affairs*, 65(3):389–394, 2020.

[11] Mohinder S. Grewal and Angus P. Andrews. *Kalman Filtering: Theory and Practice with MATLAB*. Wiley-IEEE Press, Hoboken, NJ, USA, 4th edition, 2015.

[12] Anders Grunnet-Jepsen, Michael Harville, Brian Fulkerson, Daniel Piro, Shirit Brook, and Jim Radford. An introduction to intel realsense visual slam and the t265 tracking camera. Whitepaper, Intel Corporation, 2019. Version 1.0.

[13] Intel Corporation. *Intel RealSense Tracking Camera T265 Datasheet*. Intel Corporation, March 2019. Revision 002, Document Number: 572522-002.

[14] Intel Corporation. Intel realsense sdk 2.0 (librealsense). `https://github.com/realsenseai/librealsense`, 2024. GitHub repository.

[15] Jacek Kropiwnicki. Problem of slip definition in driving systems. *MATEC Web of Conferences*, 332:01019, 2021.

[16] Jack B Kuipers. *Quaternions and rotation sequences: a primer with applications to orbits, aerospace, and virtual reality*. Princeton University Press, 1999.

[17] Manoj Kushwah and Ashis Patra. Tuning pid controller for speed control of dc motor using soft computing techniques-a review. *Advance in Electronic and Electric Engineering*, 4(2):141–148, 2014.

[18] Darryl Morrell. Extended kalman filter lecture notes. EEE 581, Arizona State University, 1997. Spring 1997.

[19] Katsuhiko Ogata. *Modern Control Engineering*. Prentice Hall, Boston, MA, 5th edition, 2010.

[20] Hans B. Pacejka. *Tyre and Vehicle Dynamics*. Butterworth-Heinemann, Oxford, UK, 2nd edition, 2006.

[21] David Peterson. An introduction to gray code for encoders. `https://control.com/technical-articles/an-introduction-to-gray-code-for-encoders/`. Accessed: 2026-03-10.

[22] Sasha Przybylski. The math behind extended kalman filtering. Medium, 2023. Available online: https://medium.com/@sasha_przybylski/the-math-behind-extended-kalman-filtering-0df981a87453.

[23] Jorge Antonio Sánchez-Molina, Francisco Rodríguez, Juan Carlos Moreno, Juan Sánchez-Hermosilla, and Antonio Giménez. Robotics in greenhouses. scoping review. *Computers and Electronics in Agriculture*, 219:108750, 2024.

[24] Dan Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. John Wiley & Sons, Hoboken, NJ, USA, 2006.

[25] u-blox. Rtk - real time kinematic: High precision gnss for the masses. Whitepaper, u-blox Holding AG, 2019.

[26] Angelo Ugenti, Fabio Vulpi, Raul Dominguez, Florian Cordes, Annalisa Milella, and Giulio Reina. On the role of feature and signal selection for terrain learning in planetary exploration robots. *Journal of Field Robotics*, 39:355–370, 12 2021.

[27] Eric A. Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pages 153–158. IEEE, 2000.

[28] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical Report TR 95-041, University of North Carolina at Chapel Hill, 1995. Updated 2006.

[29] Naiqian Zhang, Maohua Wang, and Ning Wang. Precision agriculture—a worldwide overview. *Computers and Electronics in Agriculture*, 36(2-3):113–132, 2002.

[30] Qin Zhang. *Precision Agriculture Technology for Crop Farming*. CRC Press, 2015.