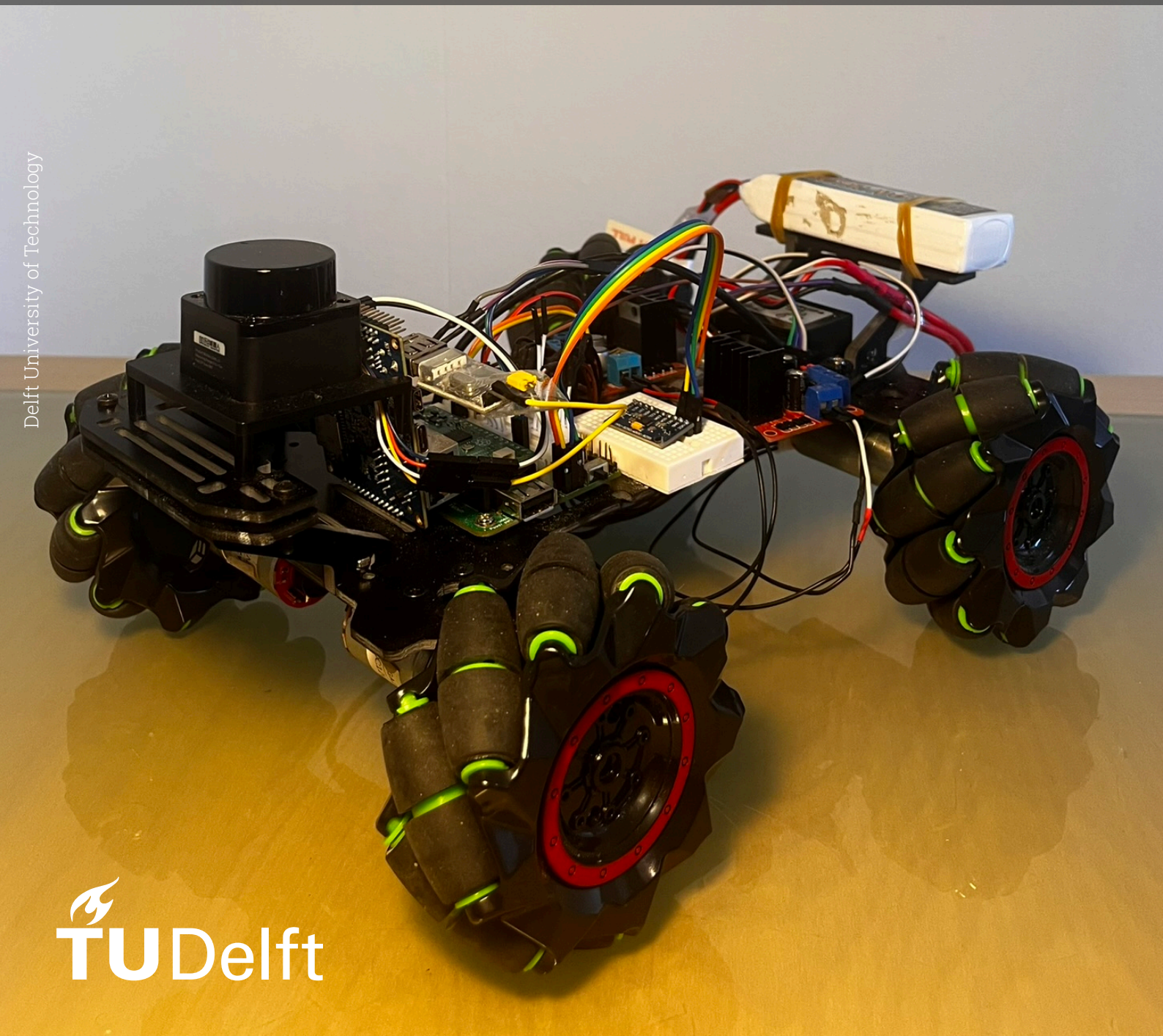
Model-Based-Control for Trajectory Tracking with a Mecanum Wheeled Vehicle

A performance comparison between kinematic and dynamic model-based control

Master of Science Embedded Systems
T.N. van der Spijk



Delft University of Technology

TU Delft

Model-Based-Control for Trajectory Tracking with a Mecanum Wheeled Vehicle

A performance comparison between kinematic and dynamic model-based control

by

T.N. van der Spijk

to obtain the degree of Master of Science
in Embedded Systems
at the Delft University of Technology

Student Number: 4693817

Thesis committee:

Chair:	B. Shyrokau
Core Member:	M. Wisse
Member:	A. Bertipaglia

Project Duration:	November 2023 - July 2024
Faculties:	Mechanical Engineering (ME) Electrical Engineering, Math & Computer Science (EEMCS)

TU Delft

Preface

I would like to thank all individuals who contributed to this project. On academic level, as thesis supervisor, Barys Shyrokau has guided and supported me through this project. He has been a great source of knowledge, inspiration, and motivation throughout the entire trajectory of the project.

I would also like to thank Alberto Bertipaglia. Available at all times, sometimes even during weekends, Alberto, as daily supervisor has helped me greatly in tackling mathematical, software, and writing challenges.

Furthermore, I would like to thank Arend-Jan van Hilten for his tireless support. While helping me find bugs and solving software-related issues, he has taught me the ins and outs of the Robot Operating System, applied to the Mirte Master platform.

Lastly, I would like to thank my family for their support and understanding, my grandfather in particular. Through his deep interest in the project, he has helped me solve several problems on different levels.

T.N. van der Spijk
Delft, July 2024

Summary

Trajectory tracking with four-Mecanum-wheeled-vehicles (FMWVs) is a critical aspect of autonomous navigation, applied in industry, healthcare, and education. Increasing trajectory tracking accuracy increases efficiency, productivity, and safety. This research applies model-based control to achieve trajectory tracking of an FMWV to answer the research question: *What are the benefits of using a dynamic-model-based trajectory tracking controller, for a four-Mecanum-wheeled-vehicle, over a kinematic-based one?*

To answer this question, an FMWV is designed and built. This consists of designing and implementing the vehicle's hardware and software, the localization, the kinematic-model-based trajectory tracking control, and the dynamic-model-based trajectory tracking control.

The vehicle's hardware is designed according to a set of requirements. The PC and microcontroller, however, were adopted from the Mirte project, as the software from this project is also used as a basis for this research to save time in software development. The consequence of this design decision was that the software had to be designed such that the computational power matched that of the PC. This is achieved in all software parts, apart from the localization.

Then, the dynamical plant model is formulated. The plant model is constructed of a simplified tire model, a DC motor model, a translational dynamic model, and a yaw moment model. The result is a nonlinear dynamical model that incorporates friction estimation for an FMWV. To design and implement two trajectory-tracking Linear Quadratic Regulators (LQRs), two models are used. First, a kinematic model of an FMWV from literature is adapted by incorporating a DC motor model to create the baseline controller. The velocity state is expanded by including the pose in the world frame to formulate the new state space. In this way, a linear state space is formulated, suitable for an LQR. Second, the dynamical plant model is linearized offline algebraically for use in an LQR. As the pose of the vehicle influences the linearization, at each controller time step, the linearization and consequent LQR gain are updated using the current reference in the trajectory.

The simulation validation shows the benefits of using a dynamic-model-based controller in the low-friction scenario. In this scenario, the vehicle overshoots its target by 67 mm when using the kinematic model-based controller. When using the dynamic model-based controller, the overshoot does not occur. This is explained by the lack of slip modeling in the kinematic model, which assumes a no-slip infinite friction scenario. The dynamic model incorporates a simplified tire model to estimate the available friction. Because of this, the vehicle tracks the trajectory without overshooting in the low-friction scenario. In the high-friction scenario, the behavior of the controllers is comparable, which is to be expected as the tire operates in the linear region of the friction coefficient.

In experimental validation, it is again concluded that in the high-friction scenario, the performance of both controllers is comparable. The results from the experiments in the low-friction scenario differed from the expectation. The trajectory of neither controller overshot the reference. Nonetheless, the benefits of using the dynamic model-based controller were shown. The dynamic model-based controller outperformed the kinematic model-based control in the low-friction scenario, in consistency, and in a reduction in translational longitudinal and lateral RMSE of 32.2 % and 41.6 % respectively.

From this research, it can be concluded that the main benefits of using a dynamic-model-based trajectory tracking controller, for a four-Mecanum-wheeled-vehicle, over a kinematic-based one are apparent in low-friction conditions. In low-friction conditions, the simulation and experimental validation show a reduction in translational RMSE. The simulation shows a reduction in maximum deviation from the trajectory, the experiment, however, shows a similar maximum deviation from the trajectory. This difference is attributed to localization and hardware challenges.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
AGV	Automated Guided Vehicles
AMR	Autonomous Mobile Robot
DOF	Degrees Of Freedom
EKF	Extended Kalman Filter
FMWV	Four-Mecanum-Wheeled-Vehicle
FPID	Fuzzy Proportional Integral Derivative
GPIO	General Purpose Input/Output
IMU	Inertial Measurement Unit
KF	Kalman Filter
LiDAR	Light Detection And Ranging
LQR	Linear Quadratic Actuator
LTI	Linear Time Invariant
MaxE	Maximum Error
MPC	Model Predictive Controller
NMPC	Nonlinear Model Predictive Controller
NTSMC	Non-singular Terminal Sliding Mode Controller
OOSM	Out-Of-Sequence-Measurements
PID	Proportional Integral Derivative
R&D	Research and Development
RMSE	Root Mean Square Error
ROS	Robot Operating System
SLAM	Simultaneous Localization And Mapping
SMC	Sliding Mode Controller
TDOA	Time Difference of Arrival
TSMC	Terminal Sliding Mode Controller
VLC-IPS	Visible Light Communication based Indoor Positioning Systems

Symbols

Symbol	Definition	Unit
v_x, v_y	Velocity in vehicle reference frame	[m/s]
Ω_z	Yaw rate in vehicle reference frame	[rad/s]
x_{world}, y_{world}	Position in world reference frame	[m]
ψ	Angular position in world reference frame	[rad]
ω_i	Angular velocity wheel i	[rad/s]
F_i	Force in direction i	[N]
μ_i	Calculated friction coefficient wheel i	[-]
$v_{slip,i}$	Slip velocity wheel i	[m/s]
I	Inertia	[kgm^2]
E_i	Voltage applied across motor i	[V]

List of Figures

List of Tables

1

Introduction

Vehicles play a big role in the transportation of goods and personnel efficiently across various terrains and applications. To meet the ever-growing demand for advanced transportation solutions, continuous research and development (R&D) is essential. To save cost and time, scaled vehicles, small versions of their full-sized counterparts, are employed to test new technologies and design principles before applying them to full-sized vehicles. These scaled models provide valuable insights into vehicle dynamics, control strategies, automation, and other critical areas, thereby streamlining the transition of innovations from concept to full-scale implementation.

Traditional vehicles rely on steering mechanisms or differential drive to change their heading angle. Equipped with conventional wheels that rotate around their axles, these vehicles can only move along their longitudinal axis. When a steering mechanism is applied, the wheels can also rotate around the contact point with the ground to reach states outside this axis. They must rotate around a vertical axis to change their heading angle, to be able to drive in a different direction. This results in a system with only two Degrees Of Freedom (DOF) in a non-slip scenario.

In contrast, vehicles fitted with omnidirectional wheels exhibit significantly enhanced maneuverability. Such a vehicle has the unique capability of omnidirectional movement. In 2D translational movement, the omnidirectional vehicle can not only move along the longitudinal axis but also the lateral axis and any combination of these two. In rotational movement, the vehicle can rotate around its axis, comparable to the differential drive vehicle. The omnidirectional vehicle therefore has three DOF in a non-slip scenario [1]. In practice, this means that the omnidirectional vehicle can reach any state in the planar environment without requiring extra orientation maneuvers. This feature is particularly advantageous for industrial applications involving precise trajectory tracking.

This research goes into the behavior of a scaled omnidirectional vehicle. It involves developing, constructing, and programming a scaled omnidirectional robot. This robot is built to compare two trajectory-tracking controllers designed specifically for an omnidirectional vehicle. The baseline trajectory controller is implemented following the design from previous research. An advanced controller is proposed to improve the trajectory tracking performance over the baseline. Both the baseline and proposed controllers are validated through simulation and experimental trials to assess their effectiveness.

The findings are expected to contribute to the field of vehicle dynamics and control. By demonstrating the benefits of the proposed controller in a scaled vehicle the research supports the adoption of this technology in full-sized omnidirectional vehicles. This can lead to more efficient and versatile transportation solutions across various industries, enhancing the overall performance and efficiency of the transportation of goods and personnel.

1.1. Historical Highlights

This section elaborates on historical highlights in the field of mobile robotics in general, followed by highlights from the field of omnidirectional vehicles.

1.1.1. Mobile Robotics

During World War 2 the first mobile robots emerged in the form of flying bombs. Smart features were included that enable detonation within a certain range of the target, guiding systems, and radar control.

After the war, in 1948, the Machina Speculatrix was presented [2]. It was a wheeled robot fitted with a light sensor. The robot would move toward a light source, and, if present, avoid obstacles on the way. In 1962, a robot called The John Hopkins Beast was presented [3]. It was controlled by boolean logic, implemented with logic gates controlling analog voltages. The cybernetic robot was able to wander around a room and look for black wall outlets using a Sonar sensor.

The first intelligent robot was created in 1972 called Shakey. It could perform tasks that required planning, route-finding, and rearranging of simple objects [4]. In 1988, the first Autonomous Mobile Robot (AMR) was presented. It could travel from one location to another, turn the light switches on and off, open and close doors, and push movable objects around. The planner could devise a plan to perform all the required actions in the right order. Other examples from space exploration include the Viking program in 1976 and Interstellar Exploration Rover in 2003 both designed and constructed by NASA. Both robots were designed to do unmanned exploration on Mars. Industrial examples include the warehouse robot made by Kiva Systems in 2005 and the Quadruped robot made by Boston Dynamics in 2005 as well, intended to carry heavy loads over terrain too rough for wheeled vehicles. [5].
The examples of mobile robots above show the advancement of the capabilities of mobile robots through decades of research and development. They provide the base for the future development of mobile robots used for research, industrial, and personal applications. From this base, with current technologies, the possibilities of robots to create are close to endless.

1.1.2. Omnidirectional Robots

A specific type of mobile robot is the omnidirectional robot. The base for this type of robot was already laid on August 6, 1919, when J. Grabowiecki filed a patent for an invention about vehicle propulsion and steering mechanisms. Part of this patent was the design of an omnidirectional wheel as seen in Fig. 1.1. The patent was granted on May 14, 1921 [6]. This early innovation laid the groundwork for the future of omnidirectional movement in robotics.

In the following decades, a significant amount of research has been done in the field of omnidirectional movement. Examples are the Omni wheel, as shown in Fig. 1.1, the Mecanum wheel, as shown in Fig. 1.2a, spherical wheels, and caster wheels. Currently, the most used type in industry is the Mecanum wheel. It was developed and patented in 1975 by Bengt Ilon while employed at Mecanum AB in Sweden [7]. The wheel Ilon developed consists of a wheel-shaped base: a cylindrical chassis with a mounting surface in the middle of the length of the cylinder. To the surface of this cylinder, rollers are mounted at a 45° angle to the wheel plane as seen in Fig. 1.2. When the wheel is spun, the force exerted by the wheel is partially dispersed by the roller. The remaining force is not in the plane of rotation, enabling omnidirectional movement by combining the forces of the other 3 wheels.
The Mecanum wheel is widely used in the industry due to its capabilities. The wheels provide enhanced maneuverability in areas where space is critical, such as warehouses. Due to the improved maneuverability, using the Mecanum wheels can reduce the required total floor space. Also, the time needed to complete a trajectory requiring extra maneuvers with traditional wheels is reduced. Another advantage is the scalability of the Mecanum wheel. It can be used from small picking robots to large transportation units. This makes them applicable in warehouses of all sizes. Due to its precise control capabilities, a vehicle equipped with Mecanum wheels is ideal for use with automated driving. Due to these advantages, the wheels have been implemented in several industries including manufacturing, logistics, healthcare, agriculture, and research.

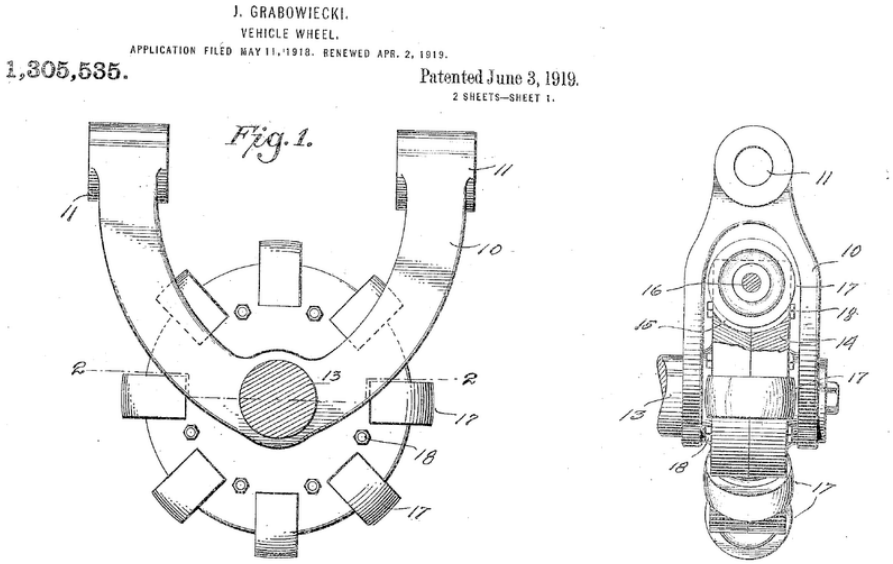
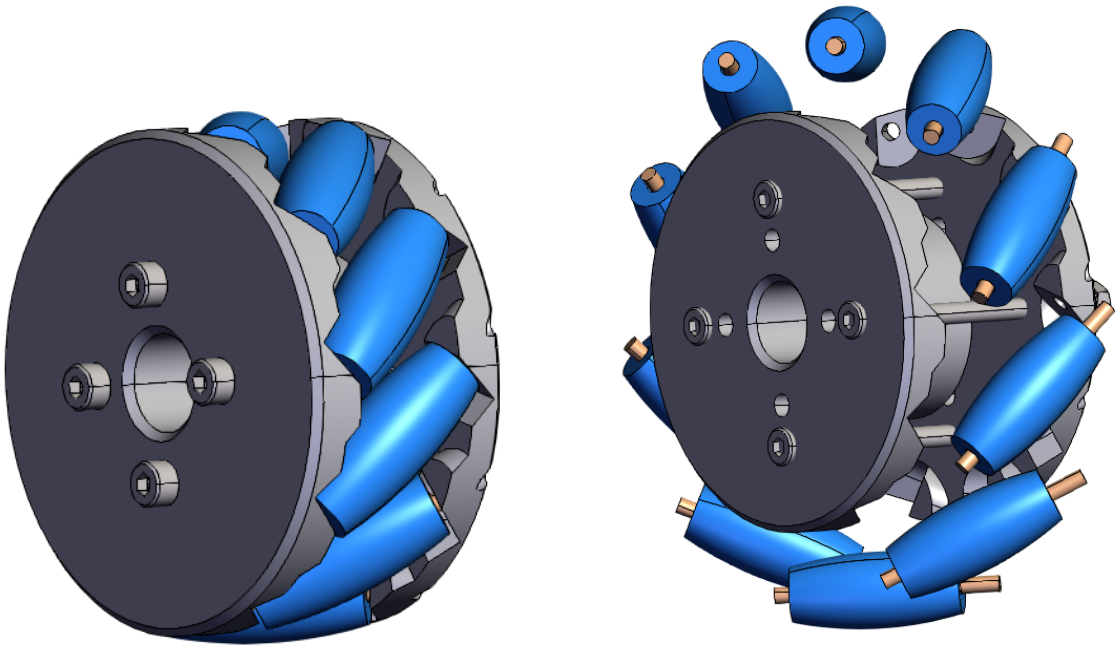


Figure 1.1: Omnidirectional wheel patented by J. Grabowiecki in 1919 [6]



(a) 3D Model of a Mecanum wheel

(b) Exploded 3D Model of a Mecanum wheel

Figure 1.2: Mecanum wheel model [8].

For high-speed, long-range transportation efficiency and durability of a wheel are key. The Mecanum wheel has more complexity and therefore more failure points. The Mecanum wheels are less durable than Mecanum wheels at high speed. Also, less traction is available in forward movement compared to a traditional wheel. The benefits of the Mecanum wheel therefore do not outweigh the downsides for high-speed, long-range transportation.

In manufacturing, however, Mecanum wheeled robots are used to streamline transportation between production lines. Also, small picking robots can be fitted with Mecanum wheels to improve their ef-

ficiency. Precise autonomous movement and navigation remove the need for drivers, which in turn enhances safety by removing human error [9]. The same applies in logistics where Mecanum wheeled vehicles are used to pick and deliver items in tight aisles of warehouses. In healthcare, Mecanum wheeled vehicles are used to provide autonomous movement in tight areas such as hospitals, nursing homes, and elderly houses for patient assistance devices.

1.2. Motivation

A Four-Mecanum-Wheeled-Vehicle (FMWV) can move omnidirectional. This ability enables complex movements applied in applications where maneuvering space is limited and holonomic movement can increase time- and space efficiency.
FMWVs are primarily used autonomously. In such autonomous applications, trajectory tracking capability is required. When the FMWV is autonomously applied in an industrial application, improvement of the trajectory tracking performance of a vehicle decreases the probability of accidents caused by path deviation [10]. An added benefit of decreasing the path deviation is the psychological effect on workers as they feel safer working with and around the vehicles, as the path is predictable [9]. Economically, a decrease in path deviation of the FMWV increases the efficiency of the vehicle in both time and energy consumption [11].

Apart from an increase in trajectory tracking performance, an FMWV is also less complex as no mechanism is needed for steering the wheels. Therefore, the steering rack, ball joints, and steering knuckles are no longer maintenance items or points of failure. Moreover, a higher number of combinations to control the four electric motors is possible, increasing the maneuverability of the vehicle.
Adding to the economic and psychological motivations of improving the trajectory tracking performance of an FMWV, other motivations exist. For example, in education, it can be used to teach students about vehicle modeling, simulation, kinematics, dynamics, programming, robotics, and several other topics. Improving on the state of the art could motivate students as they see how their contribution could make an economic and psychological impact.

1.3. Objective

In this project, the main goal is to design a vehicle equipped with four Mecanum wheels with trajectory tracking capabilities. To achieve this goal, several sub-goals are defined as follows:

1. To design the vehicle's hardware such that the specified hardware requirements are met. The hardware requirements involve the number of wheels, minimum velocity, battery capacity, current capacity, computational power, and localization capabilities.
2. To design the vehicle's software such that a ROS network runs on the vehicle's embedded PC. This network consists of actuator-, sensor-, and processing nodes that, together, form the trajectory-tracking control system.
3. To design the vehicle's localization such that the pose and velocity are updated at a minimum frequency of 25Hz. The localization must be robust against wheel slip to facilitate localization in a low-friction scenario.
4. To implement a baseline trajectory tracking controller and propose a new trajectory tracking controller incorporating dynamic modeling.
5. To validate the design in simulation such that the differences between the baseline and proposed controller become apparent.
6. To validate the findings of the simulation in experiments and explain the differences where applicable according to the key performance indicators.

The designed vehicle could, after the conclusion of this project, serve as a test-bed for further research and education.

Consequently, the research question that will be answered in this thesis is:
What are the benefits of using a dynamic-model-based trajectory tracking controller, for a four-Mecanum-wheeled-vehicle, over a kinematic-based one?

2

Related Work

This chapter elaborates on related work regarding the control of FMWVs. Firstly, several sources describing the modeling and control of an FMWV are discussed in order of the complexity of the proposed control method. Secondly, different onboard-, and offboard localization methods are discussed.

2.1. Modeling and Control of a Four-Mecanum-Wheeled-Vehicle

The kinematics of the Mecanum wheel were already described by Muir and Newman in 1987 [12]. According to their work, a Mecanum wheeled robot can be modeled by its body speed equations as listed in Eq. 2.1. A no-slip frictionless scenario is assumed. Here, v_x, v_y and Ω_z are the vehicle velocities in the vehicle frame of reference as shown in Fig. 2.1, ω_i represent the angular wheel velocities, l_1 is the distance between the two axles of the vehicle, known as the wheelbase, and l_2 is the distance between the mounting surfaces of the left and right wheels, known as the track width. r_frame is the distance from the vehicle's geometric center to the wheel's mounting point, which could be considered as the radius of the frame.

$$\begin{bmatrix} v_x \\ v_y \\ \Omega_z \end{bmatrix} = \frac{r_{frame}}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ -\frac{1}{l_1+l_2} & \frac{1}{l_1+l_2} & -\frac{1}{l_1+l_2} & \frac{1}{l_1+l_2} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} \tag{2.1}$$

2.1.1. Feed-forward Control

The inverse of this relationship is used for feed-forward control of the vehicle [13]. The relationship consists of a matrix that is not square, therefore, the inverse cannot be taken of this matrix. Instead, the left inverse matrix is determined to find the relationship that can be used for feed-forward control as stated in Eq. 2.2. Using feed-forward control, different goals such as line-following, remote control, and object avoidance can be achieved, however with low accuracy. The main advantages of applying the Mecanum wheel are the ability for omnidirectional movement, the high load-carrying capacity, and the compact design. The main cons of applying the Mecanum wheel are its discontinuous wheel contact, its high sensitivity to floor irregularities, and its complex wheel design. Especially the high sensitivity to floor irregularities is a source of path deviation when applying feed-forward control.

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \frac{4}{r_{frame}} \begin{bmatrix} 1 & 1 & -(l_1+l_2) \\ 1 & -1 & (l_1+l_2) \\ 1 & -1 & -(l_1+l_2) \\ 1 & 1 & (l_1+l_2) \end{bmatrix} * \begin{bmatrix} v_x \\ v_y \\ \Omega_z \end{bmatrix} \tag{2.2}$$

2.1.2. Motion Planning

A dynamical model of a mechanical system is created by describing the forces present in the system. The main forces exerted on the FMWV are shown in Fig. 2.1. This figure also shows the vehicle-, and world frame of reference. This model only uses the vehicle frame of reference.

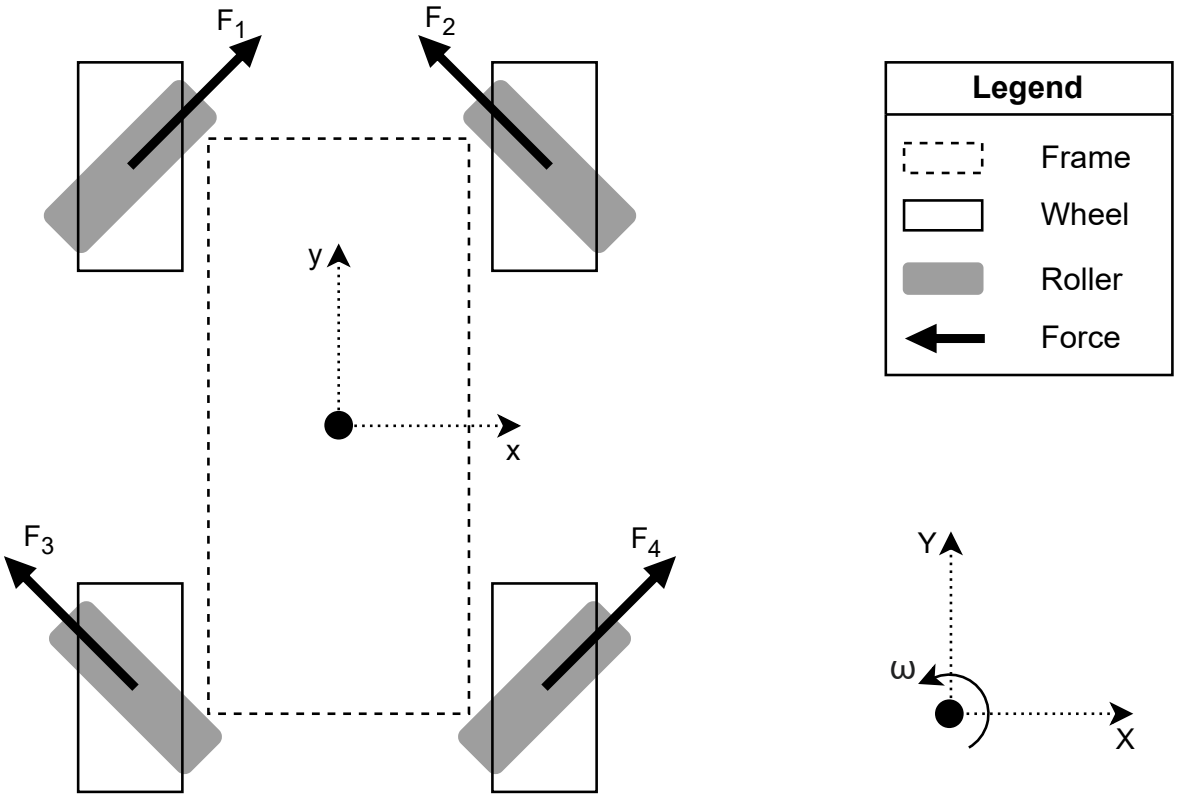


Figure 2.1: Vehicle- (x,y), and world (X,Y) reference frames with wheel forces.

The dynamic model can be used to predict the system behavior based on the current state and inputs of the system. To determine each wheel's exerted force, the interaction between the wheel and the ground must be modeled. This can be implemented using a hyper-tangent-based simplified slip model [14]. The accuracy of this model is shown by implementing motion planning without feedback. The variables are defined in Tab. B.1 in App. B. The model is defined in Eq. 2.3a to Eq. 2.3e and will be elaborated on below.

The model is constructed as follows. Firstly, in Eq. 2.3a, the slip velocity is determined using the angular wheel velocity inputs and the current velocities of the vehicle. This follows from the fact that wheel slip with a conventional wheel is defined as the difference in velocity between the wheel surface and the vehicle velocity. Following from tire modeling, the friction coefficient in the current vehicle state is calculated with the previously calculated slip velocity in Eq. 2.3b.

It is assumed that the center of gravity coincides with the geometric center of the vehicle. In that case, the weight is distributed equally on all four wheels on a flat surface. The normal force can then be calculated using a quarter of the mass of the vehicle. The total maximum force exerted by each wheel is obtained by multiplying the normal force with the previously calculated friction coefficient in Eq. 2.3c. This force is separated into a longitudinal component, in the y direction, and a lateral component, in the x direction. The sum of these separated forces from each wheel constitutes the total force acting on the center of gravity of the vehicle as calculated in Eq. 2.3d. From these forces, the vehicle accelerations can be calculated using 2.3e

$$v_{slip,i} = \begin{bmatrix} \cos(\gamma_i) & \sin(\gamma_i) & d_i \sin(\gamma_i - \alpha_i) & r_i \sin(\gamma_i) \end{bmatrix} \times \begin{bmatrix} \dot{\vec{p}} \\ \dot{\varphi}_i \end{bmatrix}, \tag{2.3a}$$

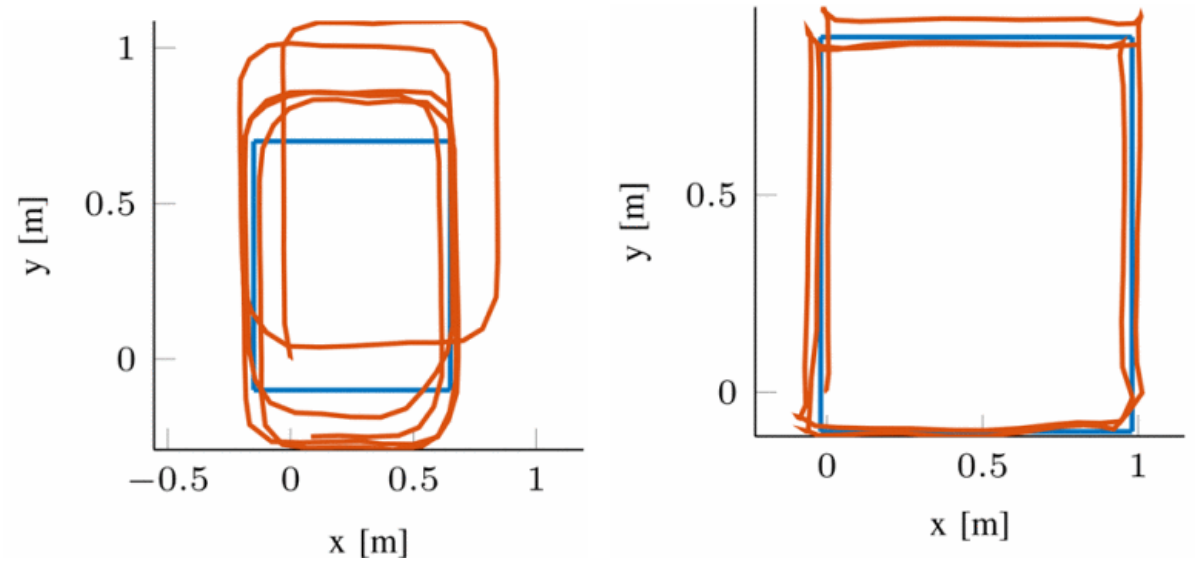
$$\mu_i = c_1 \tanh(c_2 v_{slip,i}) \text{ for } i = (1 \dots 4), \tag{2.3b}$$

$$F_i = \frac{mg}{4} \mu_i \text{ for } i = (1 \dots 4), \tag{2.3c}$$

$$\begin{bmatrix} F_x \\ F_y \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{\pi}{4}\right)(F_1 - F_2 - F_3 + F_4) \\ \sin\left(\frac{\pi}{4}\right)(F_1 + F_2 + F_3 + F_4) \end{bmatrix}, \tag{2.3d}$$

$$\dot{v} = \begin{bmatrix} \dot{v}_x \\ \dot{v}_y \end{bmatrix} = \frac{1}{m} \begin{bmatrix} F_x \\ F_y \end{bmatrix} \tag{2.3e}$$

When comparing motion planning using a kinematic model and motion planning with a dynamic model, including slip modeling, it is clear that the dynamical model is closer to reality and therefore shows less path deviation when used in motion planning. The results are shown in Fig. 2.2a and 2.2b. It must be noted that in this model, the vehicle's yaw rate is assumed to be zero due to the lack of yaw moment modeling.



(a) trajectory tracking without slip modeled

(b) trajectory tracking with slip modeled

Figure 2.2: Results of incorporating slip in a kinematic model [14].

The yaw moment can however be described by studying the forces exerted on the vehicle's body [15]. As seen in Fig. 2.3b, the relevant forces $F_{i,p}$ can be decomposed into $F_{i,x}$ & $F_{i,y}$. The magnitude of these forces can be calculated using Eq. 2.4a, Eq. 2.4b, and Eq. 2.4c.

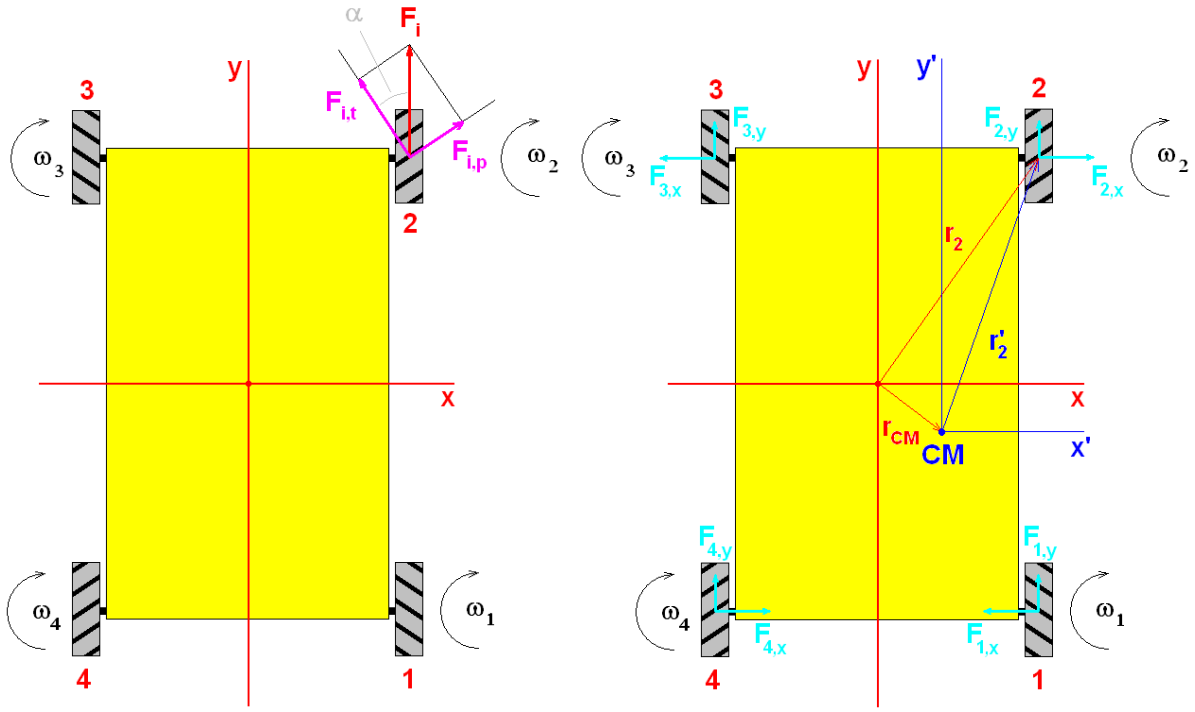
$$F_{i,p} = F_i sin(\alpha) \tag{2.4a}$$

$$F_{i,x} = F_{i,p} cos(\alpha) = F_i sin(\alpha) cos(\alpha) \tag{2.4b}$$

$$F_{i,y} = F_{i,p} sin(\alpha) = F_i sin^2(\alpha) \tag{2.4c}$$

To describe the robot's motion, a second frame of reference is defined where x' and y' are centered around the vehicle's center of mass. In this reference frame, the forces are defined as in Eq. 2.5 and 2.6 and graphically shown in Fig. 2.3b.

$$\begin{bmatrix} F_{x'} \\ F_{y'} \end{bmatrix} = \begin{bmatrix} cos(\varphi) & -sin(\varphi) \\ sin(\varphi) & cos(\varphi) \end{bmatrix} \begin{bmatrix} F_x \\ F_y \end{bmatrix} \tag{2.5}$$



(a) Driving force decomposition

(b) Attached coordinate system & Center of mass coordinate system

Figure 2.3: Force decomposition of a Mecanum wheeled robot taken from [15].

$$\vec{F_L} = F_{x'}\hat{u}_{x'} + F_{y'}\hat{u}_{y'} \tag{2.6}$$

The robot's translational motion can be obtained using Newton's second law as stated in Eq. 2.7, where m is defined as the robot's mass and $\vec{A}_{CM}$ the acceleration.

$$\vec{A}_{CM} = \vec{F_L}/m \tag{2.7}$$

The robot's rotational motion involves the yaw moment exerted on the body. This yaw moment is defined by Eq. 2.8. In this equation, $\vec{r}_{CM}$ defines the location of the center of mass in the attached reference frame (x,y), $\vec{r_i}$ the position vector of wheel i in the attached reference frame (x,y), and $\vec{r_i}'$ the position vector of wheel i in the reference frame of the center of mass. The yaw moment $\vec{T}$ will generate a rotation around the center of mass, indicating a rotation around z'.

$$\vec{T} = \sum_{i=4}^{4} \vec{r_i} \times \vec{F_{i,p}} - \vec{r}_{CM} \times \sum_{i=4}^{4} \vec{F_{i,p}} \tag{2.8}$$

The yaw rate following from the yaw moment can then be described by Eq. 2.9, where I is the vehicle's moment of inertia around z' axis, $\dot{\omega}_z$ the yaw acceleration, and $\hat{u}_{z'}$ the unit vector along the z' axis.

$$\Omega_z = \frac{\vec{T}}{I}\hat{u}_{z'} \tag{2.9}$$

2.1.3. Proportional Integral Derivative Control

A Proportional Integral Derivative (PID) controller is the least complex feedback controller. The controller output is defined in Eq. 2.10 below and consists of a proportional part, an integral part, and a derivative part[16]. With these three components, the controller takes into account the past, the current, and the future vehicle-states.

$$U(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{2.10}$$

To calculate the actual input to the system and construct a trajectory controller with the PID controller the kinematic model from Eq. 2.1 is used. In this way, a control signal in the form of four angular wheel velocities is calculated. The performance of the PID controller is outperformed by a higher-order controller, especially in the presence of disturbances. The controller however has low complexity and requires little computational power. The tuning of the controller is done heuristically. No experiments were done to prove the real-life performance of the controller.

A more advanced PID controller is the Fuzzy Proportional Derivative (FPID) Controller. An FPID Controller can be used for trajectory tracking [17]. Fuzzy gain scheduling is applied to determine the gain of the PID controller for the Mecanum wheeled robot. The research goal was to ensure that the system can handle changing conditions and reduce localization errors induced by the slipping of the wheels. It was shown that the fuzzy PID controller has increased performance over a standard PID controller. Again, the linear kinematic model from Eq. 2.1 was used. It must be noted that the gain scheduling proposed here is only possible in areas that have been mapped beforehand.

2.1.4. Linear Quadratic Regulator

The kinematic model from Eq. 2.1 can be expanded by considering the energy loss due to the wheels' viscous friction [18]. This expanded model can then be used for position rectification control of an FMWV. The control is constructed in two layers. The first is velocity control, and the second is the position control. The velocity control is implemented with a Linear Quadratic Regulator (LQR). The position control is implemented using feedback from a vision sensor. In between the accurate position measurements, the position is dead reckoned using the kinematic model and feedback from the wheel encoders.

2.1.5. Pure Pursuit Algorithm

A Pure Pursuit path-tracking algorithm can be used for trajectory tracking with an FMWV [19]. The localization is done using beacons in the robot's proximity at known locations. By determining the distance to each beacon the location of the vehicle is determined. The localization is used as input to a nonuniform dual-rate EKF to determine the immeasurable states in the system. A derived version from the kinematic model, shown in Eq. 2.1, is used as a model for the EKF. It is shown that using a nonuniform dual-rate EKF gives higher accuracy results in comparison to using a uniform dual-rate EKF. For control, a pure pursuit path-tracking algorithm is used. This algorithm uses a variable gain that depends on the proximity of the next target point. When the vehicle is in close enough proximity to the target, the next target point on the trajectory is set as a new reference.

2.1.6. Sliding Mode Control

Second-order nonlinear equations can be applied with a Sliding Mode Controller (SMC) to create a trajectory tracker that can operate in the presence of uncertainties and external force disturbances, [20]. The model equations do not need to be linearized. The kinematic model from Eq. 2.1 is used, combined with a dynamical model that considers DC motor dynamics, linearized friction coefficients, and an external force disturbance. The novel part of this paper is the description of the external forces and the matched and unmatched uncertainties. The controller design ensures chattering-free trajectory tracking of the trajectory in simulation. The resulting trajectory in simulation with corresponding input is shown in Fig. 2.4a & 2.4b respectively.

2.1.7. Nonsingular Terminal Sliding Mode Control

Trajectory tracking control of a Mecanum wheeled vehicle can be implemented with Nonsingular Terminal Sliding Mode Controller (NTSMC) [21]. The goal is to improve the trajectory tracking accuracy by focusing on the nonlinearities and uncertainties in the dynamics of the Mecanum wheeled robot. The novel part of this paper is the nonsingular terminal slide mode algorithm proposed for controlling the motion on a predefined path.

The NTSMC is a variation of the conventional SMC. The mean difference from the SMC is the finite convergence time on the sliding manifold. Nonsingular operation is achieved by avoiding the singularity at $x = 0$ for negative powers of x in the calculation of the sliding mode variable s [22].

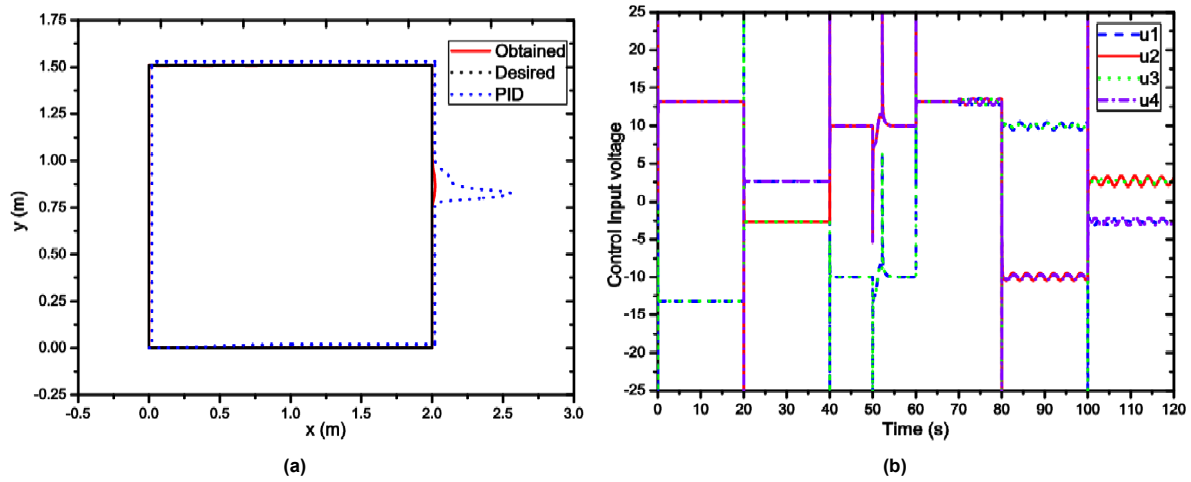


Figure 2.4: Resulting trajectory and corresponding control input from simulation of trajectory tracking with Sliding Mode Control in the presence of uncertainties and disturbances [20].

NTSMC was compared to a conventional SMC. Experiments show the superiority of NTSMC over SMC. In lateral movement, NTSMC shows a higher tracking precision in trajectory tracking and heading angle error. The control signal from the NTSMC does however show more chattering. This chattering was however alleviated in further research [23] using a fuzzy adaptive recursive terminal sliding mode control algorithm. What is noteworthy in this research is the ability of trajectory tracking without relying on sensors. However, in longer trajectories, the accumulative drifting error becomes too large. Therefore, future research must include sensors.

2.1.8. Model Predictive Control

A Model Predictive Controller can be implemented to control an omnidirectional robot with the goal of path tracking in an energy-efficient way [24]. The kinematic model from Eq. 2.1 is used, combined with a dynamical model of the FMWV. The dynamical model from Eq.2.11a is linearized around the current trajectory point in every time step. Conversions from the input voltage and current to wheel speed and torque are given in Eq. 2.11b and 2.11c respectively. The downside of this approach the MPC is the high computational load on the hardware. The upside of the MPC is the robustness to disturbances while maintaining minimal power consumption as the control method uses both model prediction as well as feedback. All variables are defined in Tab. B.2 in App. B.

$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J \end{bmatrix} \begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} - \begin{bmatrix} F_x^c \\ F_y^c \\ F_z^c \end{bmatrix} - \begin{bmatrix} F_x^v \\ F_y^v \\ F_Z^v \end{bmatrix} + \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \end{bmatrix} \tag{2.11a}$$

$$u_a(k) = L_a \frac{di_a(k)}{dt} + R_a i_a(k) + K_1 \omega_h(k) \tag{2.11b}$$

$$T(k) = l K_2 i_a(k) \tag{2.11c}$$

2.1.9. Nonlinear Model Predictive Control

Another implementation of an FMWV path follower is presented in [25]. A Nonlinear Model Predictive Controller (NMPC) is used for trajectory tracking with an FMWV. A Laser Imaging, Detection, And Ranging (LiDAR), four-wheel encoders, and an IMU are used to localize the FMWV, using an Extended Kalman Filter (EKF) to fuse the sensor values into a location estimate. The kinematic model from Eq. 2.1, is used as a model in the NMPC controller for the FMWV. No comparison to other control methods is shown, therefore the performance is difficult to interpret.

2.2. Onboard Localization

Onboard localization consists of the localization of a vehicle relative to its initial position using onboard measurements.

2.2.1. Open-loop Localization

It is possible to calculate the pose, which consists of the position and the orientation, of a moving object by having an accurate initial position, velocity information from wheel motor encoders, and the kinematic model as proposed in Eq. 2.1 [21]. Using the Runge-Kutta method, shown in Eq. 2.12 & 2.13, these variables can be combined to determine the pose of the next time-step. T is the sampling period, it should be sufficiently small to ensure constant acceleration. ω is the heading angle of the vehicle in the world frame, $\dot{\theta}$ is a vector containing the angular velocity of each wheel, and $h(\omega)$ the kinematics of the FMWV as in Eq. 2.1. This localization method is vulnerable to drift in the pose estimate due to modeling errors. The advantage however is the low computational load and the consequent ability of a relatively high refresh rate.

$$\begin{bmatrix} x_q(k) \\ y_q(k) \\ \omega_q(k) \end{bmatrix} = \begin{bmatrix} x_q(k-1) \\ y_q(k-1) \\ \omega_q(k-1) \end{bmatrix} + \frac{\tau}{4} \frac{T\left[h(\omega(k-1))\dot{\theta}(k-1) + h(\omega(k))\dot{\theta}(k)\right]}{2} \tag{2.12}$$

$$\text{where: } h(\omega) = \begin{bmatrix} -\sqrt{2}\sin(\omega) & \sqrt{2}\cos(\omega) & -\sqrt{2}\sin(\omega) & \sqrt{2}\cos(\omega) \\ \sqrt{2}\cos(\omega) & \sqrt{2}\sin(\omega) & \sqrt{2}\cos(\omega) & \sqrt{2}\sin(\omega) \\ \frac{1}{a+b} & \frac{-1}{a+b} & \frac{-1}{a+b} & \frac{1}{a+b} \end{bmatrix} \tag{2.13}$$

2.2.2. LiDAR based Localization

A more robust onboard localization method is the use of a LiDAR sensor. Combined with a Simultaneous Localization And Mapping (SLAM) algorithm, this technology can be used to determine the FMWV's pose relative to an initial location [26]. Several variations of the classic SLAM algorithm have been developed. Map-based LiDAR localization removes the mapping from the algorithm to reduce computational load [27]. The downside is that a map of the area has to be available beforehand, and the accuracy depends on the quality of the map. Visual-SLAM combines a camera with the SLAM algorithm to achieve localization. It has to be noted that when using visual information, lighting of the area is important as the quality of the information depends greatly on light intensity [28]. Another algorithm, Visual-LiDAR SLAM, is another solution that combines odometry from both a mono camera, and a LiDAR [28]. A cheaper implementation is the LiDAR-Inertial localization system [29], which is based on LIO-SAM that adopts a fusion framework of a factor graph.

2.2.3. Combining Localization Methods using Extended Kalman Filter

Accurate state estimation can be implemented with a Kalman Filter. Such a filter will fuse data from data from multiple different sensors to obtain an overall pose estimate whose error is less than using a single sensor in isolation [30]. An Extended Kalman Filter is required when the state transition model is nonlinear. Such filters are implemented on mobile robots using the ROS environment.

2.3. Off-board Localization

Off-board localization requires components external to the FMWV to localize itself within a frame. This type of localization is implemented by Visible Light Communication based Indoor Positioning Systems (VLC-IPS) that use visible LEDs for indoor localization. Time Difference of Arrival (TDOA) is used to determine the FMWV's pose indoors. Other implementations involve wireless connectivity such as Bluetooth and Wi-Fi [31]. In the case of Wi-Fi, the locations of different access points are known. The location can then be calculated by determining the signal strength for each access point at a specific point in space.

3

Vehicle Design

The development of a four-Mecanum-wheeled vehicle (FMWV) with advanced trajectory tracking capabilities involves a comprehensive process that includes design, construction, programming, and modeling. This chapter provides an in-depth examination of these stages. Section 3.1 begins with a detailed description of the FMWV's physical design, highlighting the key components and structural considerations. Section 3.2 then delves into the details of the vehicle's software architecture, including the algorithms and control systems implementation. Finally, Section 3.3 discusses the dynamic modeling of the vehicle, where the principles of dynamics are applied to develop a robust plant model. This model serves as a crucial tool for simulating and analyzing the vehicle's performance under various conditions, ensuring that the FMWV can accurately track designated trajectories.

3.1. Hardware Design

This section starts with a detailed description of the physical requirements set for the vehicle design. The considerations made for each component are elaborated on to come to the selection of the right components for each part of the system. In general, the components are chosen so that some surplus is available from the requirement.

3.1.1. Requirements

The following requirements must be taken into account to ensure the trajectory-tracking capability of the vehicle.

1. The vehicle must be fitted with 4 Mecanum wheels.
2. The frame of the vehicle must be large and strong enough to carry all sensors and actuators.
3. The vehicle must be able to move at $2\ m/s$ in the longitudinal direction.
4. The wheel speed must be measurable.
5. The vehicle must be able to operate at 50% motor power for 15 minutes on one battery charge.
6. The vehicle must provide power to all its components at the specified voltage and with sufficient current capacity.
7. The vehicle must contain a microcontroller able to process all signals from and to all components.
8. The vehicle must contain a portable computer powerful enough to process all parts of a trajectory tracking system.

Figure 3.1: DC motor as used on the FMWV as listed in App. A

9. The vehicle must be able to locate itself relative to its initial position in a $4\ m$ by $4\ m$ room.

3.1.2. Wheels

The wheels of the vehicle must be chosen such that they meet the requirements of the Mecanum wheel as described by the patent of Mecanum [7]. A spectrum of wheels is available that meet this requirement. The wheels must also be able to be mounted to the motors in such a way that the torque of the motors can be transferred to the wheels. The chosen wheels meet all requirements. The wheels are 100 mm in diameter and feature a screw-on type mounting mechanism.

3.1.3. Motors and Wheel Encoders

The vehicle must be able to move at $2\,\mathrm{m/s}$. The minimal angular velocity of the wheels can be calculated using Eq. 3.1, where ω is the angular velocity, v the translational velocity, and d the diameter of the wheel. Considering a translational velocity $2\,\mathrm{m/s}$ and a diameter of the wheel $0.1\,\mathrm{m}$, the minimal angular velocity of the wheels must be $40\,\mathrm{rad/s}$, equal to $382\,\mathrm{RPM}$.

$$\omega = \frac{2\pi v}{\pi d} = \frac{2v}{d} \tag{3.1}$$

Apart from a minimum angular velocity of the wheels, the motors must be chosen such that the angular velocity of the wheels can be controlled with feedback. The motors must therefore feature wheel encoders suitable for angular velocities of more than $382\,\mathrm{RPM}$.

The chosen motors, as seen in Fig. 3.1, are geared so that they can rotate at $400\,\mathrm{RPM}$ and feature wheel encoders mounted on the motor shaft. Therefore, to measure the angular velocity of the wheels, the gear ratio must be taken into account. According to the data sheet in Appendix A, the gear ratio is 21:1.

Therefore, the chosen motors with wheel encoders meet the requirements.

3.1.4. Motor Drivers

The motor drivers must be chosen according to the maximum current I_{max} drawn by the motors. For a DC motor, the maximum current is drawn when the motor shaft is kept stationary i.e. blocked. The datasheet of the motors in App. A provides the blocking current of $1.2\,\mathrm{A}$.

The motor drivers were chosen to fit the requirements. The L298n Motor Driver is a dual H bridge able to deliver a continuous current $1.5\,\mathrm{A}$ and a peak current $2.5\,\mathrm{A}$ per channel. This means that it can

continuously supply the blocking current of the motors.
Other benefits of this motor driver include the less than 0.1 µA standby current and the built-in overheat protection with hysteresis effect. The input signal voltage must be between 1.8 V and 7 V. The supply voltage must be between 3 V and 30 V. The output voltage will be between 0 V and the supply voltage according to the duty cycle of the input PWM signal.

3.1.5. Light Detection and Ranging Unit

To localize the robot in a space, a Light Detection And Ranging unit or LiDAR unit is added to the vehicle. The performance of the LiDAR unit is largely determined by its range, resolution, and update frequency. Other performance measures include the measurement's accuracy, and precision, meaning how repeatable the measurements are. Other factors include the available drivers for the unit and cost. The drivers must provide support for the inclusion of the sensor in a ROS network.

The chosen LiDAR unit, the LD06 LiDAR, boasts a measurement range 12 m, a scanning frequency of 13 Hz, and an average range accuracy of 45 mm. The measurement range of 12 m ensures that the LiDAR can detect all walls of the 4 m room as specified by the requirements. Using a Simultaneous Localization and Mapping or SLAM algorithm, this sensor can be used to localize the vehicle in reference to its initial position. The main downside however is the update rate of the scans and consequently the update rate of the position estimation. This can be improved by using a sensor with a superior update rate, however, the cost of such a sensor is out of the scope of this low-cost-oriented project.

3.1.6. Integrated Motion Unit

To increase the update rate of the position estimation, an Integrated Motion Unit or IMU can be integrated into the system to provide intermediate movement information. For this, an IMU must be selected that can measure the translational as well as the angular accelerations of the vehicle. Apart from that, a driver must exist that can publish the data to a ROS network.

Firstly an IMU was chosen for which no driver could be found that enabled ROS connectivity. Therefore, an IMU was chosen that specifically had a ROS driver available. The MPU9150 can provide acceleration measurements up to $16\ G$ and send this measurement over an I2C bus to the receiver. The IMU provides lateral acceleration as well as angular acceleration. The measurement can be provided at a maximum frequency of 8 kHz. This is well over the update frequency of the pose from the LiDAR SLAM.

3.1.7. Microcontroller and Portable Computer

The Mirte Educational Robot was built previously around a Raspberry Pi Pico H microcontroller combined with an Orange Pi Zero2 using a custom PCB. As the electronics of this project were used as the basis for this project, the microcontroller and PC were kept unchanged unless required. The software must be designed with a maximum computational load suitable for the Orange Pi Zero2.

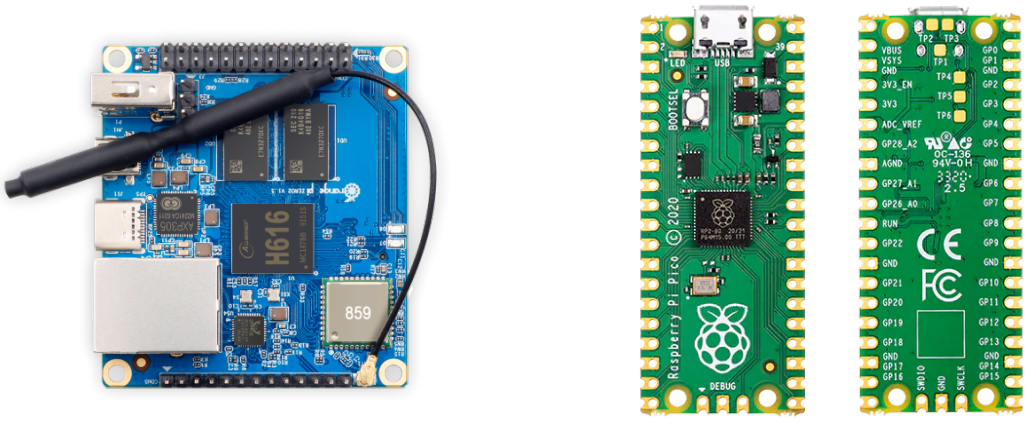
3.1.8. DC-DC Converter

As the vehicle must provide power to all its components at the specified voltage and with sufficient current capacity a power supply had to be chosen. The supply voltage is chosen at 12 V as this is the voltage required by the DC motors. However, the microcontroller, the PC, the IMU, and the LiDAR require a supply voltage of 5 V. Therefore, a DC-DC buck converter must be introduced. This converter will convert the input voltage to an output voltage of 5 V suitable for the aforementioned components. As can be calculated from the current draw of the 5 V components listed in Tab. 3.1, the total current draw on the 5 V bus is maximum 0.9 A.

The chosen DC-DC buck converter can supply 3 A at 5 V. According to the summation above, this meets the requirements.

3.1.9. Battery

The battery must be chosen according to three main parameters. Nominal supply voltage in V, capacity in mAh, and the current capability in C rating. As can be calculated from Tab. 3.1, the 5 V bus will at



(a) (b)

Figure 3.2: **(a)** OrangePi Zero2 [32]. **(b)** Raspberry Pi Pico [33].

Table 3.1: List of power consumption per device

Component	Description	Voltage(V)	Current(A)
Motors	JGA25-370 DC encoder geared motors	12	4.8
Motor driver	L298n Motor Driver Controller	12	-
Microcontroller	Raspberry Pi Pico	5	0.1
PC	Orange Pi Zero2 running Ubuntu and ROS	5	0.5
LiDAR	Okdo LiDAR Hat LD06	5	0.29
IMU	MPU9250 IMU	5	0.004

maximum draw $0.894\,\text{A}$, not taking into account losses. Considering 11% losses, this will result in a maximum current draw of $1.0\,\text{A}$ at $5\,\text{V}$. At the $12\,\text{V}$ bus, the resultant maximum current draw will be $0.41\,\text{A}$. The total maximum current at the $12\,\text{V}$ will be $5.21\,\text{A}$. At half motor power, the current draw of the $5\,\text{V}$ bus will not change. The resultant motor current draw will be $2.4\,\text{A}$. In 15 minutes, at half of its capabilities, the system will consequently consume $600\,\text{mAh}$. This defines the minimum requirement of the capacity of the battery.

The C rating of a battery defines its capability to deliver current. It is defined by how many times the battery could theoretically drain its own capacity in one hour.

For flexibility during experiments, a battery is chosen with a significantly higher capacity than required. The chosen battery has a capacity of $2200\,\text{mAh}$. The maximum current draw of the system is $5.21\,\text{A}$. Therefore, the minimum C rating of the battery is $2.36\,\text{C}$. The chosen battery has a C rating of 30C which is also significantly higher than required. However, this ensures optimal performance of the vehicle, without restriction from the battery.

3.1.10. Frame

According to the requirements, the frame of the vehicle must be able to carry all sensors and actuators as listed in Tab. 3.2. With proper space management, all sensors fit on the frame. This could not be determined beforehand, as it is not clear yet what sensors and actuators would be used throughout the project.

3.1.11. Assembly

The assembly of all separate components into a functional FMWV is performed in stages. Firstly the frame, the motors, and the wheels were assembled. Later, when the electronics were ready with basic functionality, the PC and the microcontroller were added which resulted in the first test drive of the vehicle. The partially built vehicle is shown in Fig. 3.3. All connections to the motors and encoders

Table 3.2: List of hardware components of the four-Mecanum-wheeled-vehicle

Component	Description	Amount	Dimensions (mm)
Frame	Composite frame supporting all components	1	330x142x35
Mecanum Wheel	Mecanum wheels made of plastic and metal.	4	100x47x100
Motor	JGA25-370 DC encoder geared motor	4	60x32x25
Motor driver	L298n Motor Driver Controller	2	45x45x30
Battery	Soaring 2200MAH 3S 30C LiPo	1	115x35x22
Microcontroller	Raspberry Pi Pico	1	51x25x15
PC	Orange Pi Zero2 running Ubuntu and ROS	1	55x20x64
LiDAR	Okdo LiDAR Hat LD06	1	64x55x53

were soldered to connectors that can be easily and reliably fitted. Subsequently, the IMU, as seen in Fig. 3.4a, and the LiDAR, as seen in Fig. 3.4b, were connected securely. Unlike all other peripherals that are connected to the microcontroller is the LiDAR connected to the PC directly. An I2C communication bus is set up to enable communication between the PC and the LiDAR. The result from the assembly is shown in Fig. 3.5.

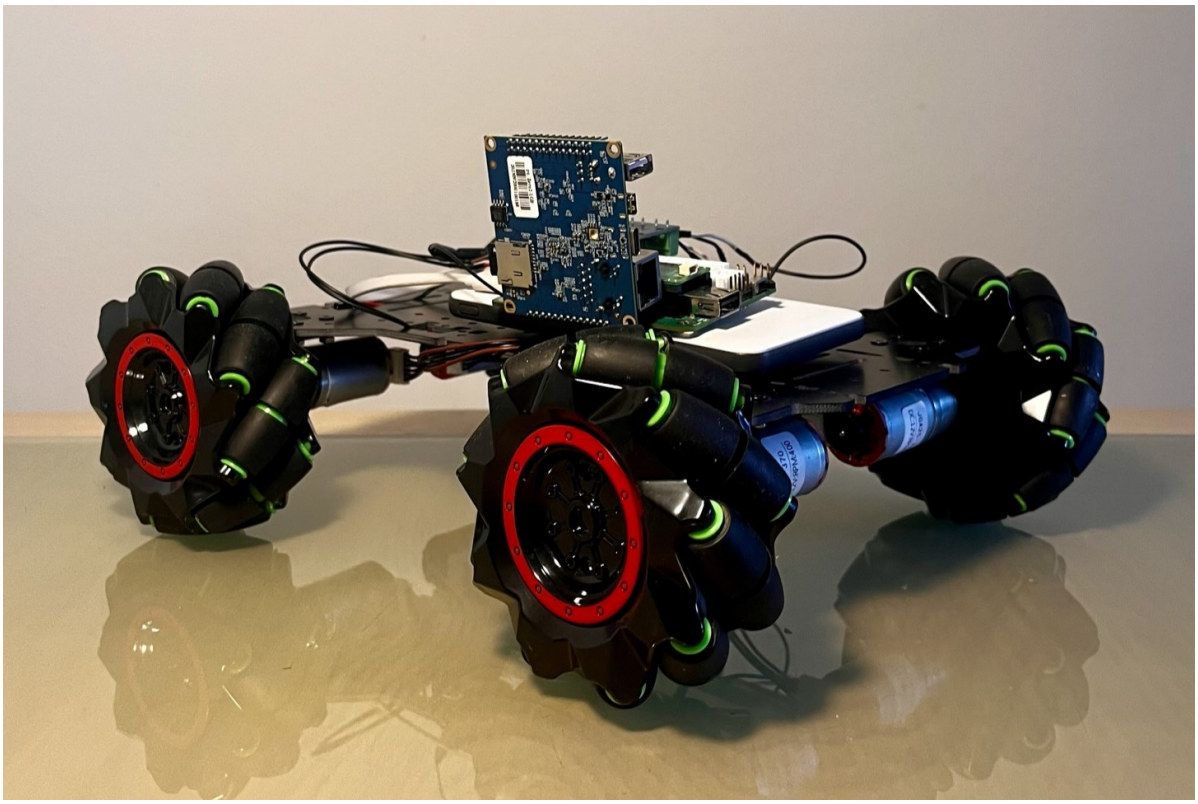
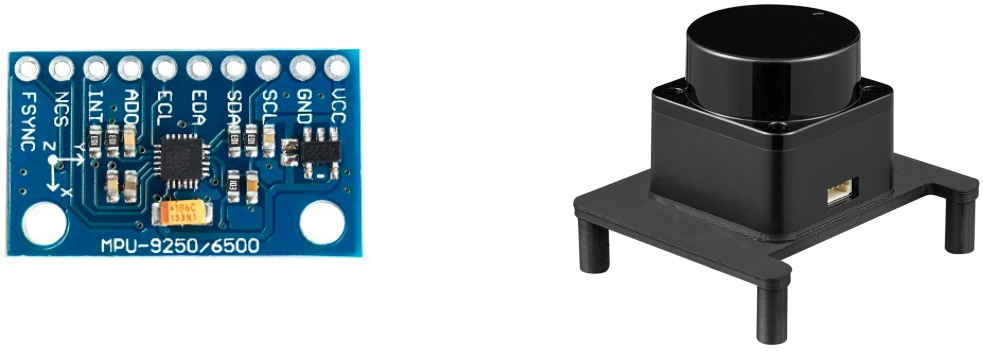


Figure 3.3: Partially built four-Mecanum-wheeled vehicle.

3.2. Software Design

Following the design and assembly of the FMWV, the vehicle software is designed. For this, the software framework of the Mirte Educational project is used together with the PC and Microcontroller from this project. This section will define the framework used and the adaptations that have been made to adapt the Mirte framework to the FMWV.



(a) (b)

Figure 3.4: (a) Inertial measurement unit of type MPU9250 [34]. **(b)** Light detection and ranging unit of type LD06 [35].

3.2.1. ROS

The Mirte software framework is built using the Robot Operating System (ROS). ROS is a set of software libraries and tools that enable the user to build robot applications.[36] A ROS network consists of Nodes and Topics. Nodes can publish and subscribe to topics. Nodes communicate using predefined message types. Because of these message types, the coding language of the node does not matter as long as the messages are published in the predefined message types. In this way, a structured way of communication is created, where different modules can be inserted as nodes.

An example of a simple ROS structure is shown in Fig. 3.6. The sensor node is the hardware interface with a sensor. The sensor node reads the sensor data and publishes this data to a predefined sensor topic using the sensor message type. The controller node is subscribed to the sensor topic and uses the sensor data to determine the required actuator input. The required actuator input is published to the actuator topic, where the subscribed actuator node takes the actuator input message and handles the hardware interaction.

3.2.2. Mirte Framework

The complete Mirte software framework is shown in Fig. 3.7. This framework is used as the basis for the FMWV. However, only part of the framework is kept. Referring to Fig. 3.7, the *browser* part is removed. In the *computer* part, that runs on the OrangePi Zero2, the *rosparam* and *mirte-ros-package* is used. In the *hardware* section, the mirte-telemetrix is used.

The *rosparam* file is a configuration file where the parameters of the ROS network are defined. It defines what components are present in the network and the mappings of these components to the correct General Purpose Input/Output (GPIO) pins.

This configuration file is used in the *mirte-ros-package*, specifically in the *mirte_bringup* file. In *mirte_bringup* the launch file for the ROS network is specified. It makes sure to start all required nodes and services. The *mirte_control* and *mirte_teleop* are replaced by the trajectory controller designed in 5. *mirte_telemetrix* defines the communication with the hardware, in this case, the microcontroller. This microcontroller is running the *mirte_telemetrix_aio* programming that is designed to control and process all actuators and sensor data respectively.

3.2.3. Adaptation of the Mirte Framework

The Mirte robot is a differential-drive robot with two traditional wheels. The robot is fitted with several accessories that enable line following and object avoidance. For this project, these sensors are not applicable and therefore not installed.

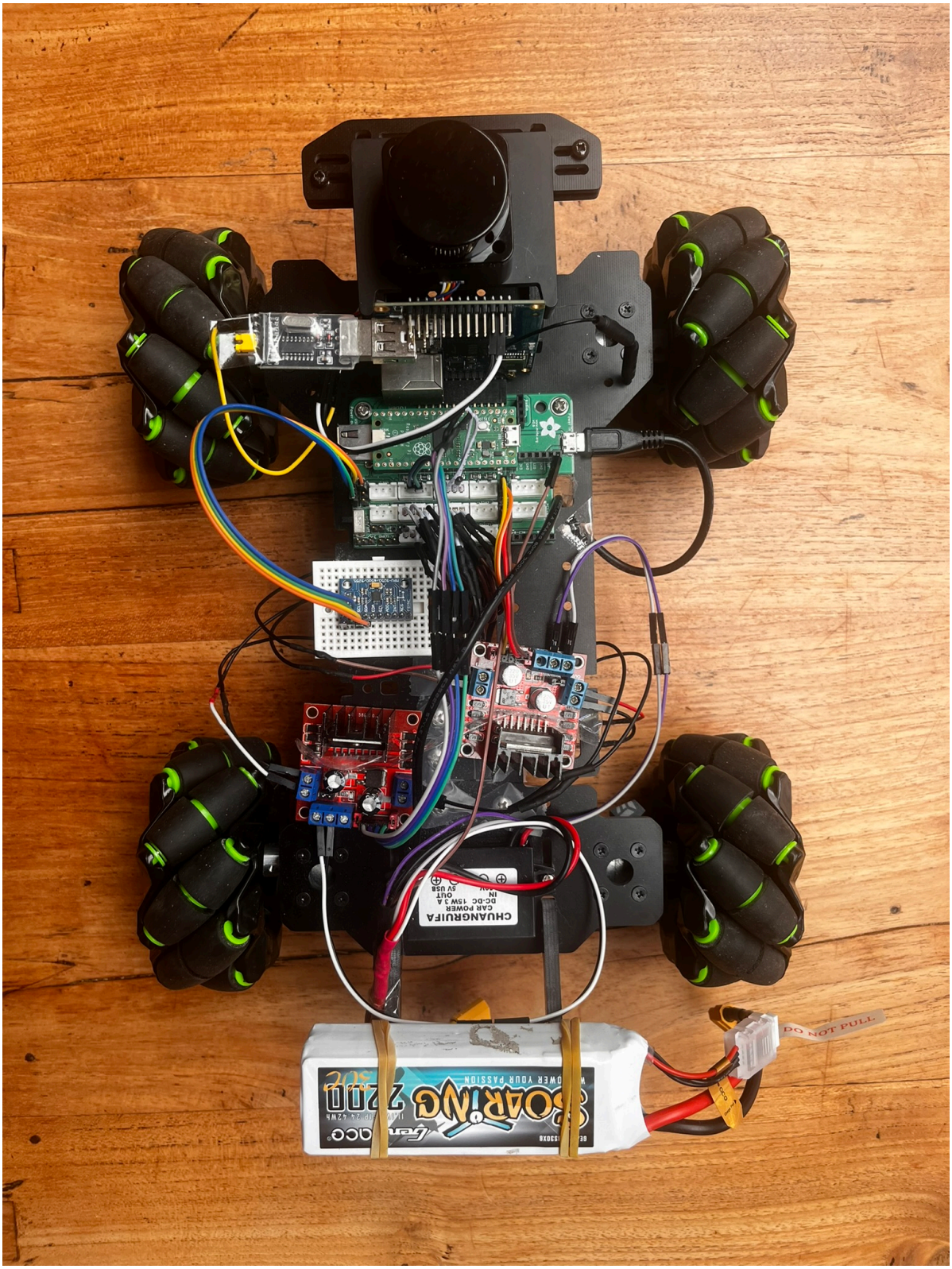


Figure 3.5: The assembled four-Mecanum-wheeled- vehicle.

The Mirte framework is used for motor control, encoder feedback, and IMU data transfer. The settings in the *rosparam* file were adapted for the specific motors and encoders. The same IMU as the Mirte

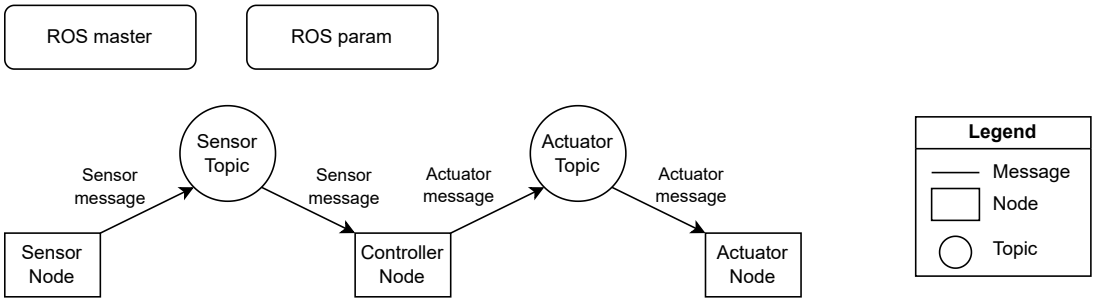


Figure 3.6: Schematic view of an example ROS structure

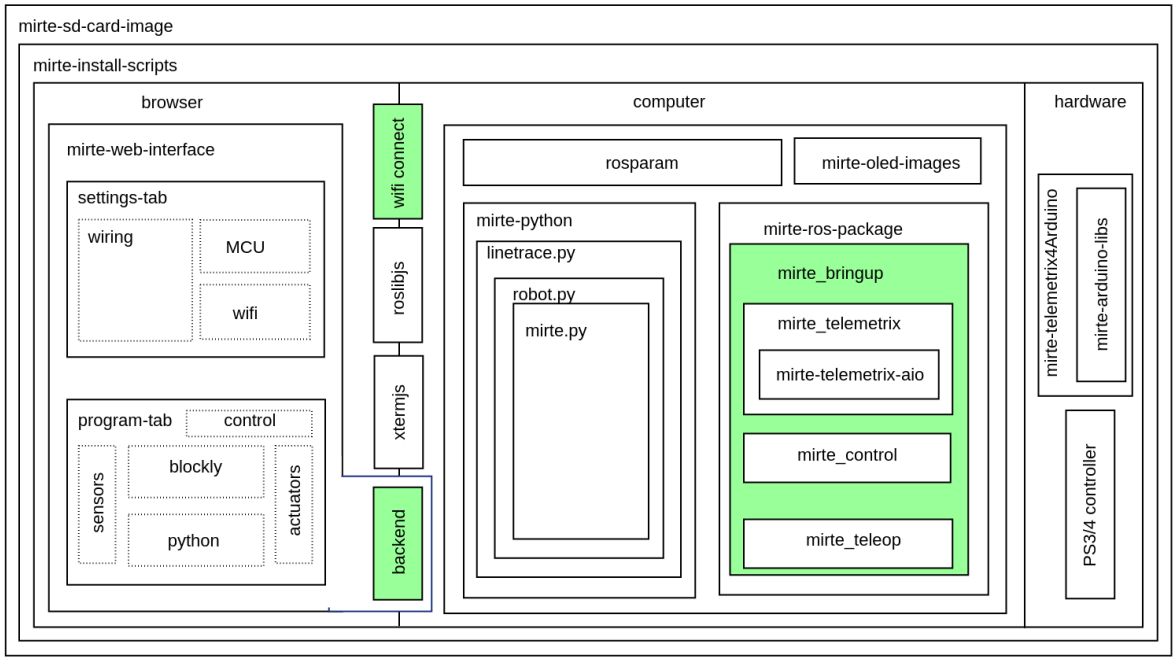


Figure 3.7: System overview of the Mirte Framework [37].

is used and requires no additional work. The LiDAR module is included by publishing the scan data to the *scan* topic. The LiDAR and its implementation were added.

3.2.4. Controller Implementation

The control and its ROS implementation were constructed using Matlab and Simulink software packages. Matlab scripting is applied to implement the initialization and logic of the control. A ROS Simulink interface is available that enables the controller to be implemented as a node as described in the example in Sec. 3.2.1. The controller design is discussed in Ch. 5.

Important for this implementation is the inclusion of safety features. Two safety features were included. Firstly, a voltage limit is introduced to limit the requested voltage sent to the motor topic. Secondly, an angular wheel velocity limit is introduced to protect the hardware from over-revving. For this, angular wheel velocity feedback using wheel encoders is implemented.

3.3. Plant Model

A plant model is created to simulate the FMWV and design the model-based trajectory tracking controllers. The plant model is a combination of three models. As a base, the model described in Sec. 2.1.2 taken from [14] is used. Here, a dynamical model is formulated. The model, as listed in Equations 2.3a

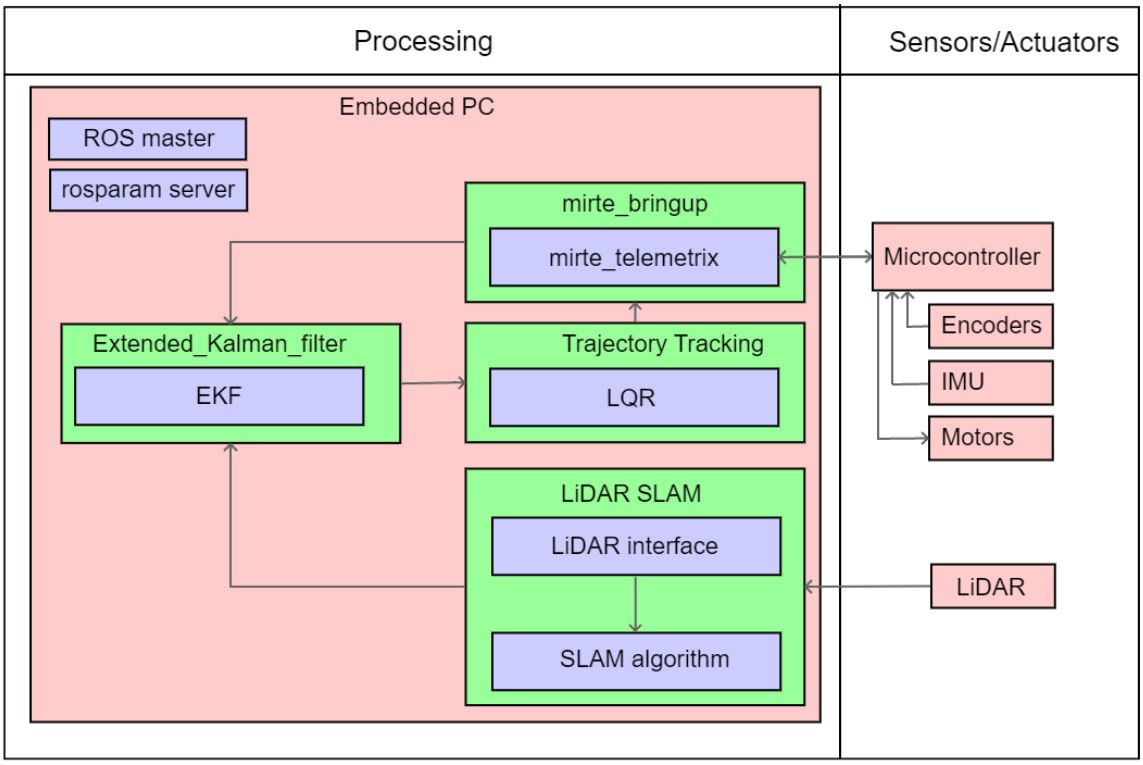


Figure 3.8: System overview of the framework adapted from the Mirte framework [37]

through 2.3e, describes the translational motion of an FMWV, lacking the model of yaw moment.

Therefore, the description of the yaw moment is added by adapting the description of the torque around the vehicle's center of mass from [15], as described in Sec. 2.1.2. As seen in Fig. 2.3, this research uses a different notation of wheel numbering. This will be adapted to use the same as in the base model. Therefore, if an anticlockwise rotation is taken to be the positive direction of rotation, this translates to F_1 and F_3 to be in the negative rotation direction, where F_2 and F_4 are directed in the positive rotation direction.

Using Newton's second law for rotation listed in Eq. 3.2a, the yaw moment T can be related to the yaw acceleration $\dot{\Omega}$ around the vertical axis, through the center of mass by the inertia I of the rotating body. As in the base model, the assumption is made that the center of mass is equal to the geometric center of the vehicle, implying a uniform weight distribution. The combined model is listed in Eq. 3.2b to 3.2e. The resultant acceleration vector can be integrated once to obtain the vehicle velocities. The pose is obtained by integrating the vehicle velocities over time.

$$\dot{\Omega}_z = \frac{T}{I_{vehicle}} \tag{3.2a}$$

$$\begin{bmatrix} v_{slip,1} \\ v_{slip,2} \\ v_{slip,3} \\ v_{slip,4} \end{bmatrix} = diag\left(\begin{bmatrix} cos(\gamma_1) & sin(\gamma_1) & d_1 sin(\gamma_1 - \alpha_1) & r_{wheel} sin(\gamma_{wheel,1}) \\ cos(\gamma_2) & sin(\gamma_2) & d_1 sin(\gamma_2 - \alpha_2) & r_{wheel} sin(\gamma_{wheel,2}) \\ cos(\gamma_3) & sin(\gamma_3) & d_1 sin(\gamma_3 - \alpha_3) & r_{wheel} sin(\gamma_{wheel,3}) \\ cos(\gamma_4) & sin(\gamma_4) & d_1 sin(\gamma_4 - \alpha_4) & r_{wheel} sin(\gamma_{wheel,4}) \end{bmatrix} \begin{bmatrix} v_x & v_x & v_x & v_x \\ v_y & v_y & v_y & v_y \\ \Omega_z & \Omega_z & \Omega_z & \Omega_z \\ \omega_1 & \omega_2 & \omega_3 & \omega_4 \end{bmatrix} \right) \tag{3.2b}$$

$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \end{bmatrix} = c_1 tanh\left(c_2 \begin{bmatrix} v_{slip,1} \\ v_{slip,2} \\ v_{slip,3} \\ v_{slip,4} \end{bmatrix} \right) \tag{3.2c}$$

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = \frac{mg}{4} \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \end{bmatrix} \tag{3.2d}$$

$$\begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{\Omega}_z \end{bmatrix} = \begin{bmatrix} \frac{cos(\frac{\pi}{4})}{m} & \frac{sin(\frac{\pi}{4})}{m} & \frac{tw+wbh}{2I_{vehicle}} \end{bmatrix} \begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{3.2e}$$

This model however lacks the DC motor dynamics. This must be included as the motor dynamics introduce an actuator delay. Especially in highly dynamic maneuvers, this will be an evident deviation from reality. Therefore, the DC motor model is implemented as listed in Eq. 3.3b, where r_{wheel} is the wheel radius, ω_i the angular wheel velocity, I_{wheel} the inertia of a Mecanum wheel, k the torque constant, R the ohmic resistance of the DC motor's armature, and E_i the voltage applied across the armature [38]. The ohmic resistance is needed as the torque generated by a DC motor is proportional to the armature current, which can be calculated using Ohm's law. The angular wheel velocity is negatively proportional to the generated motor torque.

$$I_{wheel} = \frac{1}{2} m r_{wheel}^2 \tag{3.3a}$$

$$\dot{\omega}_{wheel,i} = \frac{1}{I_{wheel}} \left(\frac{-k^2}{R} \omega_{wheel,i} + \frac{k}{R} E_i \right) \tag{3.3b}$$

4

Observer Design

This section provides a detailed elaborating of the observer design essential for accurate state estimation of the robot, including both pose and velocity measurements. The observer uses three types of sensors to gather pose and velocity information. This redundancy is included to ensure accurate localization, regardless of the available friction. Firstly, the vehicle's pose is accurately determined through a LiDAR sensor operating in conjunction with a Simultaneous Localization and Mapping (SLAM) algorithm which is, in theory, able to localize the vehicle without influence from the available friction. Secondly, the vehicle's velocity is measured using four encoders, which, when integrated with the wheel configuration matrix, offer reliable velocity estimations in the case of high friction and minimal slip. This type of localization is not robust to low-friction, however can be used for high frequency localization between the SLAM measurements. Thirdly, an Inertial Measurement Unit (IMU) is employed to capture both linear and angular accelerations of the vehicle, providing dynamic insights into its movement. To fuse the data from these sensors into one precise high-frequency state estimate, an Extended Kalman Filter (EKF) is implemented. This sophisticated fusion technique ensures that the observer delivers an accurate measurement of the robot's state by optimally integrating the sensor data together with a system model.

4.1. SLAM

A LiDAR sensor is a laser range measurement device that is mounted perpendicularly on the shaft of a motor. The output of this sensor at each measurement is a list of range measurements combined with the corresponding angle at which this measurement was taken. Combining this list of measurements, a range cloud is obtained.

The data contained in the range cloud is just data without the SLAM algorithm. The range cloud is used to construct a map of the surroundings by feature-matching scans with each other. From the matched features, the pose can be estimated. Loop closure is added to make sure that the map is closed when the vehicle reaches a location for the second time. In this sense, loop closure is a feedback mechanism to the pose estimation. This results in an accurate location estimation, however at the cost of a relatively high computational load.

4.2. Encoder with Kinematic Model

A more computationally efficient implementation of pose and velocity estimation is by the use of the encoders mounted on the motors of the wheels. The motor shaft is charged with a positive and a corresponding negative magnetic charge. Two hall effect sensors are mounted close to the motor shaft. Both hall effect sensors measure the change in magnetic field. In this way, the angular velocity of the shaft can be measured. By comparing the signal of both hall effect sensors, the direction of rotation can be determined by checking which of the two signals is lagging. Once the angular displacement and velocity of the motor shaft are known, the angular displacement and velocity of the wheel can be calculated through the gear ratio. Combining the angular wheel velocity of all four wheels with the

kinematic model from Eq. 2.1 the velocity and corresponding change in location can be calculated.

The main benefit of this method is the reduced computational load compared to the SLAM algorithm. The downside, however, is that the estimation of the change in position is based on a kinematic model, which introduces a model error. This error is enhanced even further when the wheels slip over the surface.

4.3. Inertial Measurement Unit

An Inertial Measurement Unit (IMU) is a sensor that measures linear acceleration, angular velocity, and angular position. An IMU is comprised of 3 accelerometers, 3 magnetometers, and 3 gyroscopes. The accelerometers are used to measure the linear acceleration of the device in the x, y, and z directions. The magnetometers are used to determine the angular position of the device. The gyroscopes measure the angular acceleration of the device around the x, y, and z-axis. The resulting measurements are the roll, pitch, and yaw rates.

4.4. Extended Kalman Filter

An EKF can fuse the pose and velocity measurements from the LiDAR, encoders, and IMU, together with a model of the vehicle, into one accurate state estimate. The EKF is the nonlinear extension of a Kalman Filter (KF). A nonlinear state space as in Equations 4.1 and 4.2 is assumed. It can be proved that the EKF is unbiased and has minimum variance but this is out of the scope of this project.

$$x_{k+1} = f(x_k) + w_k \tag{4.1}$$
$$y_k = h(x_k) + v_k \tag{4.2}$$
$$w_k \sim \mathcal{N}(0, Q) \tag{4.3}$$
$$v_k \sim \mathcal{N}(0, R) \tag{4.4}$$

Estimates of x_k are recursively approximated using a predicted state, as in Eq. 4.5, and a filtered state, as in Eq. 4.6

$$x_k \sim \mathcal{N}(\hat{x}_{k|k-1},\, P_{k|k-1}) \tag{4.5}$$
$$x_k \sim \mathcal{N}(\hat{x}_{k|k},\, P_{k|k}) \tag{4.6}$$

During the prediction stage, the state of the system is estimated using the dynamics of the system combined with the previous state. This can be written as in Eq. 4.7 where A_k is a linearization around x_k of the nonlinear function of the dynamics of the system, and w_k represents the assumed to be Gaussian, additive process noise as defined in Eq. 4.3 where Q is the covariance matrix of the additive process noise.

$$\hat{x}_{k+1|k} = A_k \hat{x}_{k|k} \tag{4.7}$$
$$P_{k+1|k} = A P_{k|k} A^\intercal + Q \tag{4.8}$$

The measurement update stage consists of estimating the state of the system using the newest filtered measurement of the state. The measurement function is defined by Eq. 4.2. Using the measurement and the Kalman gain as defined by Eq. 4.9, where C_k is the linearization of the measurement function around x_k, the state is estimated at time k as in Eq. 4.10 with covariance as defined in Eq. 4.11.

$$K_k = P_{k|k-1} C_k^\intercal (C_k P_{k|k-1} C^\intercal + R)^{-1} \tag{4.9}$$
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (y_k - C\hat{x}_{k|k}) \tag{4.10}$$
$$P_{k|k} = P_{k|k-1} - K_k C_k P_{k|k-1} \tag{4.11}$$

In this project, the EKF is implemented using a ROS package specifically designed for use in mobile robots using the ROS environment. The package, named *robot_localization*, developed by Charles River Analytics, Inc., contains not only the implementation of the EKF but also deals with the intricacies of using the EKF in a ROS environment.

In configuring the package, a careful decision has to be made on what variable is used from what source of information. This is specified by defining the configuration matrix for each sensor. The full configuration file is found in App. C An example is shown below for a sensor that delivers a twist-type message that contains translational and angular velocities. Of this message, v_x, v_y, and $yaw\ rate$ are used. The order of the boolean values in the configuration matrix are:
$(x, y, z, roll, pitch, yaw, v_x, v_y, v_z, roll\ rate, pitch\ rate, yaw\ rate, \dot{v}_x, \dot{v}_y, \dot{v}_z)$

```
<rosparam param="twist0_config">[false, false, false,
                                 false, false, false,
                                 true,  true, false,
                                 false, false, true,
                                 false, false, false]</rosparam>
```

Apart from the configuration matrix, for each sensor, the covariance matrix must be specified. The packages used for encoder processing and the IMU provide a covariance estimation. The SLAM package does not, so the covariance is estimated.

The SLAM covariance is calculated by having the sensor stationary, measuring the output for four minutes, and calculating the covariances of the measured signals. The measurement result is shown in Fig. 4.1a to 4.1c. The calculated covariances are listed in Tab. 4.1.

Table 4.1: Calculated covariances of the LiDAR SLAM pose measurement.

Variable	Covariance
x (m)	7.5346e-06
y (m)	8.9605e-05
ψ (rad)	5.5360e-05

After configuring each sensor separately, the process noise must be tuned. This is done in the experimental phase by running the experiment and checking the convergence of the estimate to the ground truth. By measuring the pose of the vehicle physically at the start and end point of the trajectory, and comparing this to the estimated begin and end point, the best combination is found heuristically.

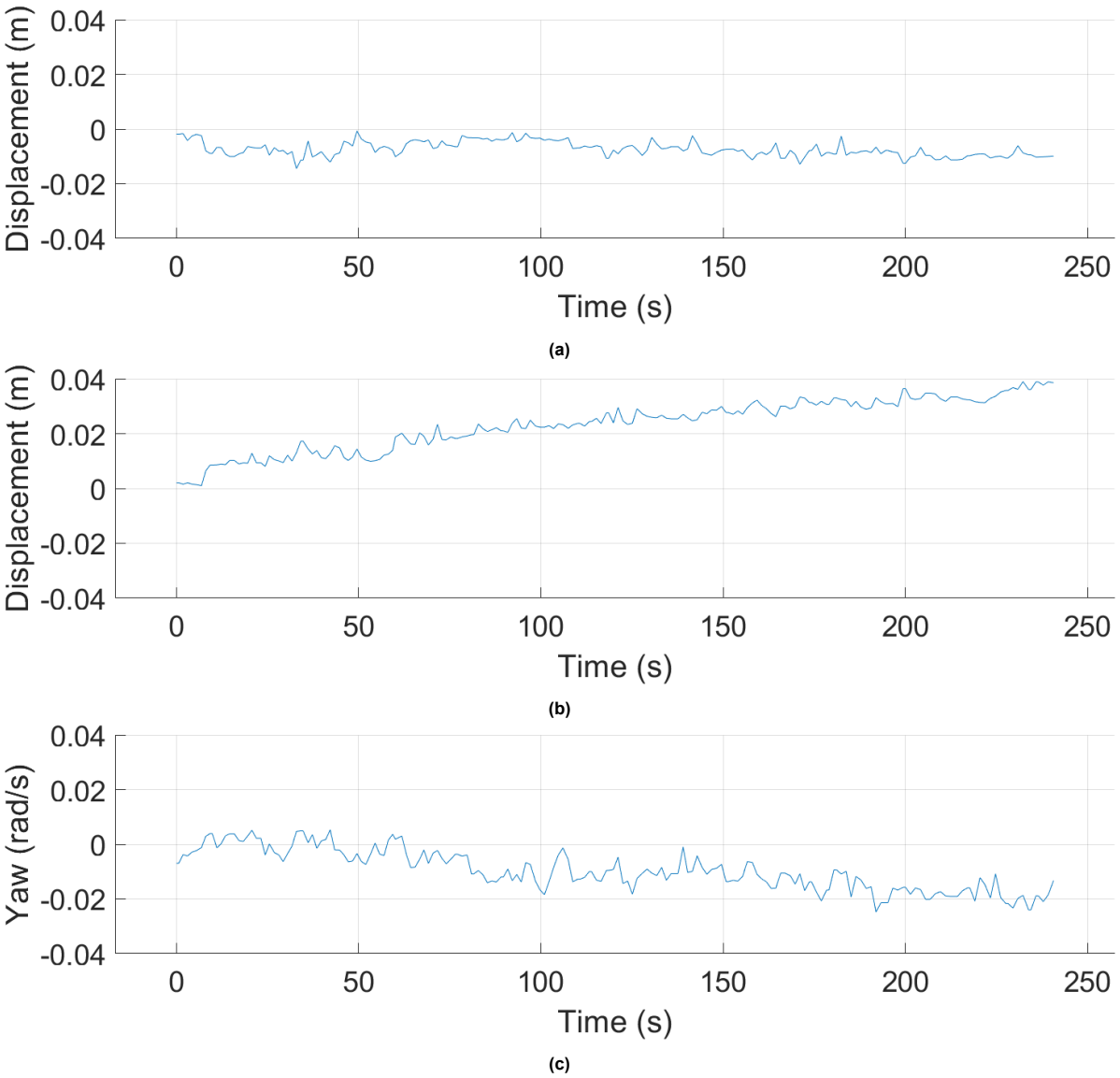


Figure 4.1: Measurement from stationary LiDAR sensor combined with the SLAM algorithm to determine the covariance of the pose estimation.

5

Control System Design

This chapter will elaborate on the control system design used to implement a trajectory tracking FMWV as found in App. D and E. Firstly, the control algorithm will be selected. Secondly, the plant model will be adapted to be compatible with the chosen control algorithm. Lastly, a tuning theory will be presented that provides a guideline for tuning the proposed controller.

5.1. Selection of Control Algorithm

This section will elaborate on the selection of the appropriate control algorithm. To do so, the advantages and disadvantages of the three control algorithms will be considered. Firstly, the linear quadratic regulator will be discussed, followed by the model predictive controller and the sliding mode controller. A controller will be chosen and tuned.

5.1.1. Linear Quadratic Regulator

An LQR is an optimal control strategy designed to control linear systems. A quadratic cost function, as shown in Eq. 5.1c, is minimized to find the optimal feedback gain matrix K. In this equation, Q and R represent the weighting matrices for the state deviation and the control efforts respectively. The minimization is subject to the linear time-invariant (LTI) first-order dynamic constraint as listed in Eq. 5.1a as well as the initial condition listed in Eq. 5.1b. There, x is the state, A the state matrix, B the input matrix and u the input. The gain K is found as in Eq. 5.1d

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{5.1a}$$

$$x(t_0) = x_0 \tag{5.1b}$$

$$J = \int_0^\infty (x^\intercal(t)Qx(t) + u^\intercal(t)Ru(t))dt \tag{5.1c}$$

$$K = R^{-1}B^\intercal P \tag{5.1d}$$

where P is obtained by solving the Riccati equation:

$$PA + A^\intercal P - PBR^{-1}B^\intercal P + Q = 0 \tag{5.1e}$$

The advantages of the LQR are listed below:

Stability: When the system is controllable and observable, the LQR guarantees closed-loop stability.

Computational efficiency: Since the LQR only deals with LTI systems, the calculation of the gain and application of the gain as the controller is not computationally heavy. This makes the LQR suitable for real-time applications with limited computational resources.

The disadvantages are listed below:

Linearity assumption: The LQR assumes the system to be LTI. However, many physical systems are not LTI. If this is the case, the model has to be adapted, for example using linearization.

Fixed gain: In an LQR, the gain is calculated offline. The gain is then constant for the system like in a proportional controller. In the case of an LTI system, this is not a problem. For most physical systems, however, operational conditions can change over time, reducing the optimality of the control.

Lack of constraint handling: The only constraints the optimal gain calculation takes into account are the model matrices and the initial condition. Other constraints can however not be taken into account in this control method.

5.1.2. Model Predictive Control

Model Predictive Control (MPC) is an optimal control technique that minimizes a cost function over a finite receding horizon. At a given iteration of its control loop, x_c is the current state, x_g the reference state. The control input u_c must be calculated such that the current state is brought closest to the reference state. The MPC implements this by solving a discrete-time optimal control problem [39]. The discretized state transition equation is listed in Eq. 5.2c. The trajectory is also discretized into states x_1, ..., x_n, where n is the length of the prediction horizon. The cost function to be minimized is listed in Eq. 5.2a subject to the constraints as listed in Eq. 5.2b, 5.2c, and 5.2d. Here, $T(x_n, x_g)$ is the terminal cost used to penalize the mismatch between the reference and the current state. $R_i(x_i, u_i)$ is the running cost that accounts for the control effort along the trajectory. $I(x_i, u_i)$ is a system of inequality constraints to characterize the set of admissible states and actions.

$$J = T(x_n, x_g) + \sum_{i=1}^{n} R_i(x_i, u_i) \tag{5.2a}$$

$$x_i = x_c \tag{5.2b}$$

$$x_{i+1} = f(x_i, u_i) \tag{5.2c}$$

$$I(x_i, u_i) \geq 0 \tag{5.2d}$$

Summarizing, at each time step, the current state is either measured or estimated using the internal plant model of the system to be controlled. Using the internal plant model, N optimal control actions are calculated. The first control action of the array of control actions over the prediction horizon is then applied to the actual plant, the other control actions are discarded. This makes the controller robust as the optimal control problem takes into account disturbances in the calculation of the control input.

The advantages of an MPC are listed below:

Adaptive control: At each time step, the control action is calculated to be optimal for that instant with the prediction horizon in mind. In this way, the control is adapted to external disturbances.

Constraint handling: In the optimization, constraints on the admissible state space can be taken into account.

The disadvantages of an MPC are listed below:

High computational load: At each time step, the current state has to be measured or estimated, the next states have to be predicted, and a cost function has to be minimized. The online optimization poses a significant computational load on the processor. This load increases with the complexity of the model. This can cause issues in mobile systems where computational power might be limited.

Model Dependence: The control actions are calculated using the predictions made with the internal plant model. Therefore, the accuracy of the plant model has a great influence on the performance of the controller.

5.1.3. Sliding Mode Control

SMC is a control technique that applies discontinuous switching control law to drive the state toward the sliding surface. Once reached, high control effort is used to keep the state on the sliding surface. The sliding surface is predefined in the state space. Pure SMC is used less often as the controller is asymptotically stable, meaning the closer the state is to the equilibrium, the slower the convergence rate. Therefore, a specific class of SMC, the more modern Terminal Sliding Mode Controller (TSMC),

will be considered. Here, the steady-state convergence is achieved by ramping up the control input to speed up the convergence to ensure that the state reaches the equilibrium in finite time[40]. For TSMC a system of the form of Eq. 5.3a, where $f(x,t)$ is the state transition function, $b(x,t)$ the internal uncertainty function, x the state vector, u the input vector, and $\xi(x,t)$ represent the external disturbances. Here, u_{eq} is defined by Eq. 5.3b. The sliding vector is defined as in Eq. 5.3c. The time to reach equilibrium is defined by Eq. 5.3d. Here, x is the state variable, β, and γ are tuning parameters where $\gamma = \frac{q}{p}$ with q and p positive odd integers.

$$\dot{x} = f(x,t) + b(x,t)t + \xi(x,t) \tag{5.3a}$$

$$u_{eq} = -\left(\frac{\partial s}{\partial x}b(x,t)\right)^{-1}\left(\frac{\partial s}{\partial x}f(x,t) + \xi(x,t)\right) \tag{5.3b}$$

$$s = \dot{x} + \beta|x|^{\gamma}sgn(x) \tag{5.3c}$$

$$t_s = \beta^{-1}(1-\gamma)^{-1}|x(0)|^{1-\gamma} \tag{5.3d}$$

The advantages of the TSMC are listed below:

Robustness: The TSMC is robust to system uncertainties and external disturbances.
Suitability for nonlinear systems: TSMC handles nonlinear system dynamics without linearization.

The disadvantages of the TSMC are listed below:

Chattering: Due to the discontinuous switching of control laws close to the sliding surface, a TSMC is prone to high-frequency oscillations in the control signals, potentially causing increased wear on actuators.

5.1.4. Selection

The goal of the project is to reduce path deviation in an FMWV trajectory tracking vehicle. Therefore, in Ch.. 3, an FMWV is designed. This vehicle is fitted with an OrangePi Zero2 microcomputer. As discussed in Ch.. 4, the microcomputer is tasked with the processing of a SLAM algorithm, as well as an extended Kalman filter. Apart from that, a wheel angular velocity controller is implemented. Considering this system, it was determined that the computational burden on the microcomputer imposed by the selected control algorithm should be minimized. This rules out the MPC, as it imposes the highest computational load. The LQR was selected to be the most suitable, as the TSCM imposes an increased wear on the physical system. TSMC provides high disturbance rejection with the trade-off being high frequency oscillations in the control signal imposing increased wear on the actuator. This is not desirable, as disturbance rejection is not a requirement. Therefore, the LQR is selected to be the most fitting control strategy for this project due to its stability guarantee, computational efficiency, and ease of implementation. A problem however arises when analyzing the plant model formulated in Sec. 3.3. The plant model is nonlinear and therefore not directly suitable for use with the LQR. Therefore in Sec. 5.2 two methods of linearizing the plant dynamics are discussed.

5.2. Control-oriented System Modeling

In Sec. 3.3, the FMWV was modeled using a dynamical description. In this section, the same FMWV will be modeled, however with the purpose of using the model with an LQR in mind. This means that a linear state space equation of the model should be determined. Firstly, the system will therefore be modeled using kinematics, which will serve as the baseline controller. Secondly, the dynamical plant model will be linearized to be suitable for application in a controller which will be the proposed controller.

5.2.1. Kinematic Modeling

The kinematic equations describing the steady state velocities of the FMWV are presented in Equations 2.1 and 2.2 [12]. These equations however do not model the DC motor dynamics. Therefore, introducing the dynamical DC motor torque model from [41], as listed in Eq. 5.4a is applied. There, $T_{L,i}$ is the load torque on the motor i, ω_i the angular velocity of wheel i, k_t the torque constant of the DC motor, R the ohmic armature resistance, and E_i the voltage applied to DC motor i. The torque is related to the angular acceleration of the wheel by applying Newton's second law of rotational motion as listed in Eq. 5.4b [38].

$$T_{L,i} = \frac{-k_t^2}{R}\omega_i + \frac{k_t}{R}E_i \tag{5.4a}$$

$$\dot{\omega}_i = \frac{T_{L,i}}{I_r}(\omega_i, E_i) \tag{5.4b}$$

[38] also proposes a kinematic model of an FMWV with different mounting orientations of the Mecanum wheels to the vehicle frame. Combining this model with the kinematic model of [12], the state space equation in Eq. 5.5 is obtained. In this state space representation of the FMWV, W is the wheel configuration matrix presented as in Eq. 2.1. The vehicle velocities, v_x, v_y, and Ω are defined in the local vehicle frame.

$$\begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{\Omega} \end{bmatrix} = -\frac{k_t^2}{I_r R}\begin{bmatrix} v_x \\ v_y \\ \Omega \end{bmatrix} + \frac{k_t}{I_r R}W\begin{bmatrix} E_1 \\ E_2 \\ E_3 \\ E_4 \end{bmatrix} \tag{5.5}$$

The goal of this project is to design a trajectory-tracking FMWV. This requires the state to be expanded, as currently, only the velocities in the reference frame of the vehicle are represented. The pose of the vehicle in the world frame is required to create adequate control over the pose of the vehicle. Therefore, the velocities in the vehicle frame are translated to the world frame using Eq. 5.6 [42]. This is used to calculate the position in the world frame by translating and integrating the vehicle velocities measured by the onboard sensors. The inverse of this matrix can be used to translate the coordinates from the vehicle reference frame to the world reference frame.

$$\begin{bmatrix} \dot{x}_{world} \\ \dot{y}_{world} \\ \dot{\psi}_{world} \end{bmatrix} = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} v_{x,vehicle} \\ v_{y,vehicle} \\ \Omega_z \end{bmatrix} \tag{5.6}$$

The combined kinematic model is represented below.

$$\begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{\Omega}_z \\ \dot{x}_{world} \\ \dot{y}_{world} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} -c1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -c1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -c1 & 0 & 0 & 0 \\ cos(\psi) & -sin(\psi) & 0 & 0 & 0 & 0 \\ sin(\psi) & cos(\psi) & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} v_x \\ v_y \\ \Omega \\ x_{world} \\ y_{world} \\ \psi_{world} \end{bmatrix} +$$

$$\frac{k_t r_{frame}}{4 I_r R}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ -\frac{1}{l_1+l_2} & \frac{1}{l_1+l_2} & -\frac{1}{l_1+l_2} & \frac{1}{l_1+l_2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} E_1 \\ E_2 \\ E_3 \\ E_4 \end{bmatrix} \tag{5.7}$$

The problem with this model is that the state space model is dependent on the state. This requires linearization around an equilibrium point and consequent continuous updating of the controller gain. To alleviate this problem, the approach of [43] is applied. Here, the world-to-vehicle frame conversion is done before the controller ensuring that the input to the controller is fully in the vehicle frame. This way, the state dependency is removed from the state space equation. The obtained state space equation is presented in Eq. 5.8. The controller performance is not affected by this translation, the computational load is however greatly reduced.

$$\begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{\Omega}_z \\ \dot{x}_{vehicle} \\ \dot{y}_{vehicle} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} -c1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -c1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -c1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} v_x \\ v_y \\ \Omega \\ x_{vehicle} \\ y_{vehicle} \\ \psi \end{bmatrix} + \frac{k_t r_{frame}}{4 I_r R} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ -\frac{1}{l_1+l_2} & \frac{1}{l_1+l_2} & -\frac{1}{l_1+l_2} & \frac{1}{l_1+l_2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} E_1 \\ E_2 \\ E_3 \\ E_4 \end{bmatrix} \tag{5.8}$$

5.2.2. Dynamic Modeling

In Sec. 3.3 a dynamical plant model of an FMWV is described. This model describes the translational as well as the rotational movement of the vehicle. This model is made nonlinear by the friction coefficient estimation, using the hyper tangent function, and the DC motor dynamics. This model must be linearized to be used for LQR control.

Linearization is implemented by calculating the Jacobian around an equilibrium point following the methodology used in [44]. Therefore, firstly, the Jacobian of both the state matrix A as well as the input matrix B must be computed as seen in Eq. 5.9d. The linearization will be around the reference state. In this reference state, the input required for equilibrium can be calculated by setting the state space equation equal to zero and filling in the reference state values. The four inputs, the input voltages to the motors E_i, are the remaining unknowns, where 6 equations are available. Therefore the reference input u_{ref} can be calculated. In this way, a linearized model is obtained in the form of Eq. 5.9c that can be used for the calculation of the LQR gain. The resultant control input is implemented as in Eq. 5.9e.

$$\bar{x} = x - x_{ref},\ \bar{u} = u - u_{ref} \tag{5.9a}$$

$$\dot{\bar{x}} = \dot{x} = f(x, u) \tag{5.9b}$$

$$\dot{\bar{x}} \approx f(x_{ref}, u_{ref}) + \frac{\partial f(x_{ref}, u_{ref})}{\partial x} + \frac{\partial f(x_{ref}, u_{ref})}{\partial u} = A_{lin}\bar{x} + B_{lin}\bar{u} \tag{5.9c}$$

$$A_{lin} = \begin{bmatrix} \frac{\partial \dot{v}_x}{\partial v_x} & \frac{\partial \dot{v}_x}{\partial v_y} & \frac{\partial \dot{v}_x}{\partial \Omega_z} & \frac{\partial \dot{v}_x}{\partial x} & \frac{\partial \dot{v}_x}{\partial y} & \frac{\partial \dot{v}_x}{\partial \psi} \\ \frac{\partial \dot{v}_y}{\partial v_x} & \frac{\partial \dot{v}_y}{\partial v_y} & \frac{\partial \dot{v}_y}{\partial \Omega_z} & \frac{\partial \dot{v}_y}{\partial x} & \frac{\partial \dot{v}_y}{\partial y} & \frac{\partial \dot{v}_y}{\partial \psi} \\ \frac{\partial \dot{\Omega}_z}{\partial v_x} & \frac{\partial \dot{\Omega}_z}{\partial v_y} & \frac{\partial \dot{\Omega}_z}{\partial \Omega_z} & \frac{\partial \dot{\Omega}_z}{\partial x} & \frac{\partial \dot{\Omega}_z}{\partial y} & \frac{\partial \dot{\Omega}_z}{\partial \psi} \\ \frac{\partial \dot{x}}{\partial v_x} & \frac{\partial \dot{x}}{\partial v_y} & \frac{\partial \dot{x}}{\partial \Omega_z} & \frac{\partial \dot{x}}{\partial x} & \frac{\partial \dot{x}}{\partial y} & \frac{\partial \dot{x}}{\partial \psi} \\ \frac{\partial \dot{y}}{\partial v_x} & \frac{\partial \dot{y}}{\partial v_y} & \frac{\partial \dot{y}}{\partial \Omega_z} & \frac{\partial \dot{y}}{\partial x} & \frac{\partial \dot{y}}{\partial y} & \frac{\partial \dot{y}}{\partial \psi} \\ \frac{\partial \dot{\psi}_z}{\partial v_x} & \frac{\partial \dot{\psi}_z}{\partial v_y} & \frac{\partial \dot{\psi}_z}{\partial \Omega_z} & \frac{\partial \dot{\psi}_z}{\partial x} & \frac{\partial \dot{\psi}_z}{\partial y} & \frac{\partial \dot{\psi}_z}{\partial \psi} \end{bmatrix}\Bigg|_{x=x_{ref}}$$

$$B_{lin} = \begin{bmatrix} \frac{\partial \dot{v}_x}{\partial E_1} & \frac{\partial \dot{v}_x}{\partial E_2} & \frac{\partial \dot{v}_x}{\partial E_3} & \frac{\partial \dot{v}_x}{\partial E_4} \\ \frac{\partial \dot{v}_y}{\partial E_1} & \frac{\partial \dot{v}_y}{\partial E_2} & \frac{\partial \dot{v}_y}{\partial E_3} & \frac{\partial \dot{v}_y}{\partial E_4} \\ \frac{\partial \dot{\Omega}_z}{\partial E_1} & \frac{\partial \dot{\Omega}_z}{\partial E_2} & \frac{\partial \dot{\Omega}_z}{\partial E_3} & \frac{\partial \dot{\Omega}_z}{\partial E_4} \\ \frac{\partial \dot{x}}{\partial E_1} & \frac{\partial \dot{x}}{\partial E_2} & \frac{\partial \dot{x}}{\partial E_3} & \frac{\partial \dot{x}}{\partial E_4} \\ \frac{\partial \dot{y}}{\partial E_1} & \frac{\partial \dot{y}}{\partial E_2} & \frac{\partial \dot{y}}{\partial E_3} & \frac{\partial \dot{y}}{\partial E_4} \\ \frac{\partial \dot{\psi}}{\partial E_1} & \frac{\partial \dot{\psi}}{\partial E_2} & \frac{\partial \dot{\psi}}{\partial E_3} & \frac{\partial \dot{\psi}}{\partial E_4} \end{bmatrix}\Bigg|_{u=u_{ref}} \tag{5.9d}$$

$$u = u_{ref} - K(x - x_{ref}) \tag{5.9e}$$

5.3. Controller Implementation, Tuning, and Reference Generation

The controller is implemented in Matlab Simulink. From Simulink, using the Simulink Coder toolbox, the model is converted and compiled to C code to make it suitable for real-time implementation.

Bryson's rule states that to determine the weights for an LQR, Eq. 5.10a must be used. This rule serves as a base for tuning the weights but a heuristic tuning process must be applied to find the best performance.

In this research, tuning of the state-deviation-weight matrix Q is done in simulation. This is done by increasing the weights until oscillations occur, indicating that the gain on state correction became too aggressive. At this point, the weights are slightly reduced to eliminate the oscillations and achieve a more stable performance. The simulation is then run with both controllers to ensure no faulty behavior.

The control-weight matrix R, on the other hand, was kept constant during this tuning process. Adjustments to R were found to reduce the system's tracking performance. Specifically, increasing the control weights led to low-magnitude control inputs, reducing the tracking accuracy. On the other hand, decreasing the control weights resulted in control inputs that exceeded the physical limitations of the system, as the maximum allowable voltage of $12\,\mathrm{V}$ was exceeded. After tuning, the controllers show a balance between control input magnitude and tracking performance, without oscillations.

$$Q_{ii} = \frac{1}{maximum\ acceptable\ value\ of\ x_i^2},\ i = 1, 2, ..., l \tag{5.10a}$$

$$Rw_{ii} = \frac{1}{maximum\ acceptable\ value\ of\ u_i^2},\ i = 1, 2, ..., k \tag{5.10b}$$

Since the FMWV is controlled not only on pose but also on velocity, a suiting reference must be generated. The main reference is the pose over time, also called the trajectory. The vehicle's actual velocity is used in the reference velocity generation to improve the robustness of the control. Starting with the trajectory, a reference velocity is calculated by differentiating the trajectory to time. It should be noted, that the reference trajectory is defined in the world frame, where the reference velocity must be defined in the vehicle reference frame. Therefore, the differentiated trajectory, $\begin{bmatrix}\dot{x}_{ref}(t) & \dot{y}_{ref}(t) & \dot{\psi}_{ref}(t)\end{bmatrix}^\intercal$ is translated into the vehicle frame using Eq. 5.11, to obtain $\begin{bmatrix}v_{x,ref}(t) & v_{y,ref}(t) & \Omega_{ref}(t)\end{bmatrix}^\intercal$ [45].

$$\begin{bmatrix} v_{x,ref}(t) \\ v_{y,ref}(t) \\ \Omega_{ref}(t) \end{bmatrix} = \begin{bmatrix} cos(\psi_{ref}(t)) & sin(\psi_{ref}(t)) \\ -sin(\psi_{ref}(t)) & cos(\psi_{ref}(t)) \end{bmatrix} \begin{bmatrix} \dot{x}_{ref}(t) \\ \dot{y}_{ref}(t) \\ \dot{\psi}_{ref}(t) \end{bmatrix} \tag{5.11}$$

This velocity vector in the vehicle frame of reference could be combined with the trajectory to obtain the complete reference state vector. However, this relation would only hold in an ideal world. In the real world, however, disturbances are expected. Consequently, it will happen that the trajectory tracking is either leading or lagging the reference trajectory. An example that could introduce a lag between the reference and the actual trajectory, could be actuator lag on the physical FMWV as described in the plant modeling in Sec. 3.3. If in this case the reference velocity would be kept as in the ideal case, the reference would not correspond to the actual required velocity. Therefore, a P controller is designed that counteracts the delta velocity. This delta velocity is calculated below, where G is a gain to be tuned in the experimental phase.

$$\begin{bmatrix} \Delta v_x(t) \\ \Delta v_y(t) \\ \Delta \Omega(t) \end{bmatrix} = G \left(\begin{bmatrix} v_{x,ref} \\ v_{y,ref} \\ \Omega_{ref} \end{bmatrix} - \begin{bmatrix} v_{x,actual} \\ v_{y,actual} \\ \Omega_{actual} \end{bmatrix} \right) \tag{5.12}$$

The combined, compensated reference signal is formulated below.

$$\begin{bmatrix} v_{x,ref,comp}(t) \\ v_{y,ref,comp}(t) \\ \Omega_{ref,comp}(t) \end{bmatrix} = \begin{bmatrix} cos(\psi_{ref}(t)) & sin(\psi_{ref}(t)) \\ -sin(\psi_{ref}(t)) & cos(\psi_{ref}(t)) \end{bmatrix} \begin{bmatrix} \dot{x}_{ref}(t) \\ \dot{y}_{ref}(t) \\ \dot{\psi}_{ref}(t) \end{bmatrix} + G \left(\begin{bmatrix} v_{x,ref} \\ v_{y,ref} \\ \Omega_{ref} \end{bmatrix} - \begin{bmatrix} v_{x,actual} \\ v_{y,actual} \\ \Omega_{actual} \end{bmatrix} \right) \tag{5.13}$$

6

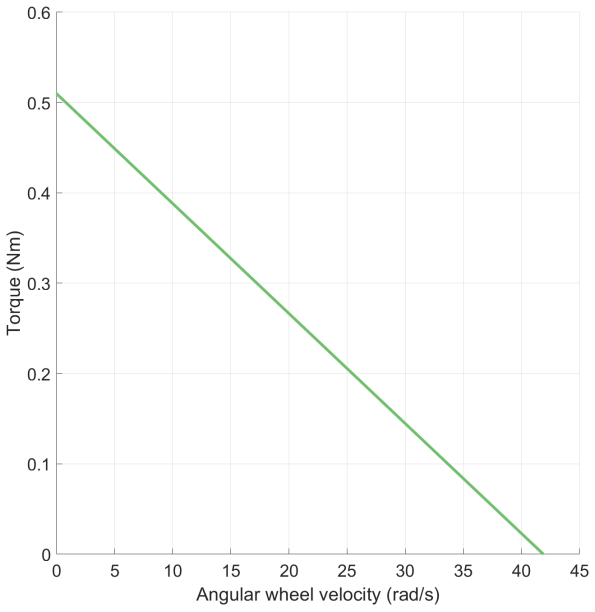
Simulation Validation

In this chapter, the designed vehicle and control system are validated. This is done in several steps. First, the DC motor model is simulated to verify the model and its parameters. Second, the tire model is simulated. Third, the control performance of the controller designed with the kinematic-based linear model will be compared to the controller designed with the linearized dynamic model.

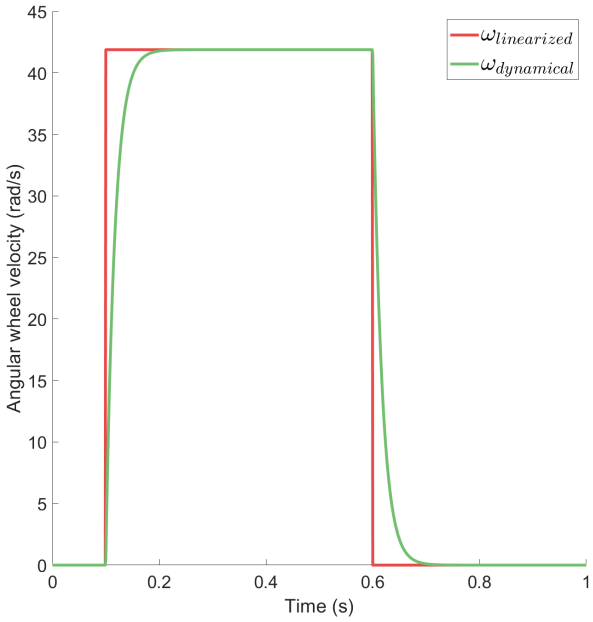
6.1. DC Motor Model

In Sec. 3.3, a dynamical model of a DC motor is presented. The parameters of this model are calculated using the datasheet of the used DC motor. The wheel inertia is calculated assuming uniform weight distribution in the wheel. The simulated DC motor torque curve is shown in Fig. 6.1a. From the figure, it can be noted that the stall torque of the motor is 0.51 Nm and the no-load angular velocity is 41.89 rad/s which translates to approximately 400 RPM, which are the rated stall torque and angular velocity of this motor respectively. The data was captured by applying a virtual step of 12 V to the simulated DC motor with the parameters from the datasheet of the JGA25-370 DC motor using the Matlab Simulink software.

Fig. 6.1b shows a comparison of the step response of the dynamical DC motor model and the linearized model. Since the linearized model does not take into account the inertia of the wheel, it can be noted that the step response is instantaneous for the linearized model. The curve of the dynamical model clearly shows the expected rise time caused by the wheel's inertia, combined with the finite torque.



(a) Simulated torque curve of the JGA25-370 DC motor



(b) Comparison of the dynamical DC motor model and its linearization using a 12V input step.

6.2. Tire Model

The tire model is presented in Ch. 3.3 as part of the plant model in Eq. 3.2c. In this model, two parameters c_1 and c_2 are to be defined. c_1 represents the friction coefficient between the wheel contact point and the floor surface. c_2 represents a unitless parameter similar to tire stiffness. Two values of the friction coefficient are chosen for simulation as listed in Tab. 6.1. c_2 is set to be $0.5-$. The results are shown in Fig. 6.2. A clear distinction can be seen in the level of friction that is generated. The slope in the linear region is increased, however as c_2 is kept constant, the width of the linear region is not changed.

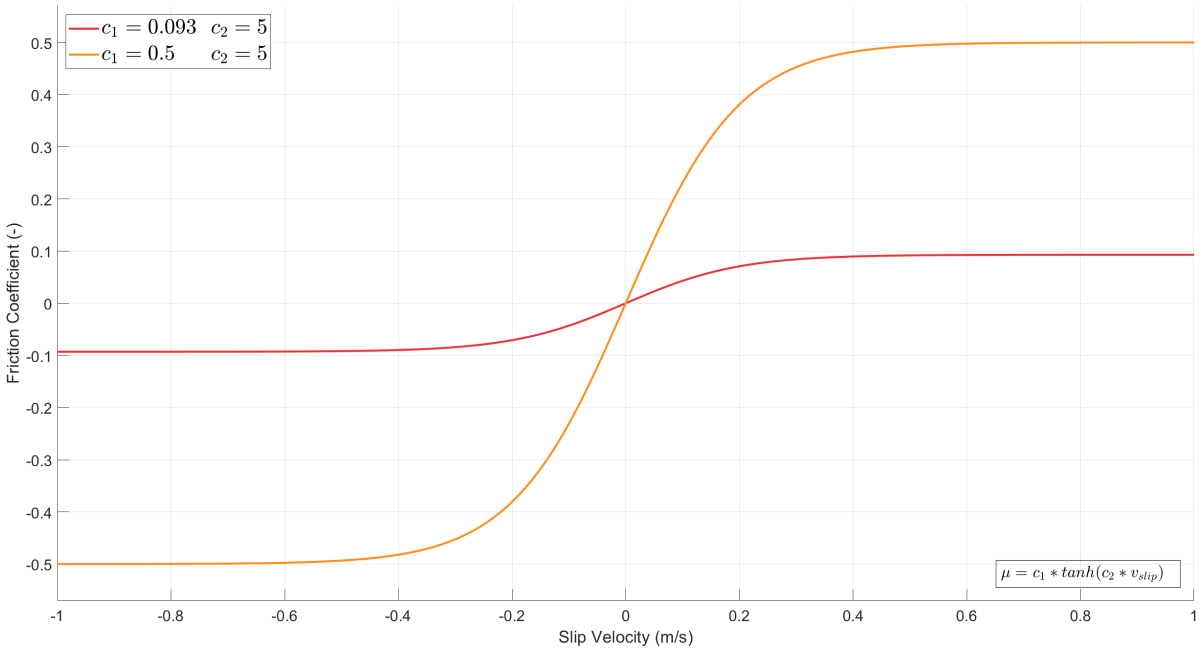


Figure 6.2: Comparison of friction coefficient with constant tire stiffness as used in the simulation.

6.3. Controller Performance

This section describes the differences between the two controllers, designed in Ch. 5, by applying the baseline and the proposed controller to the simulated FMWV in two scenarios. After defining the test scenario, a set of performance metrics is defined. These metrics are consequently checked to evaluate the performance of the controllers in all scenarios.

Table 6.1: List of different tests comparing the performance of the kinematic model- and dynamic model-based LQR.

Test #	c_1	Controller model
1	0.5	Kinematic
2	0.5	Dynamic
3	0.093	Kinematic
4	0.093	Dynamic

6.3.1. Scenario

The simulated scenario consists of a square trajectory with sides of $1\,$m in length that will be traversed in a clockwise direction as seen in Fig. 6.3. The reference is generated throughout $50\,$s where every $10\,$s the next reference point is published. The vehicle must reach the reference point as quickly as possible without overshooting. The scenario is simulated on two surfaces, one with a high and one with a low friction coefficient. The high friction coefficient, $c_1 = 0.5$, referring to the notation of Eq. 3.2c, is the static friction coefficient between dry concrete and polyurethane rubber. The low friction coefficient, $c_1 = 0.093$, is the coefficient between a wooden floor with a layer of dust and polyurethane rubber. In both cases, the polyurethane rubber is referred to as the rubber on the contact surface of the FMWV.

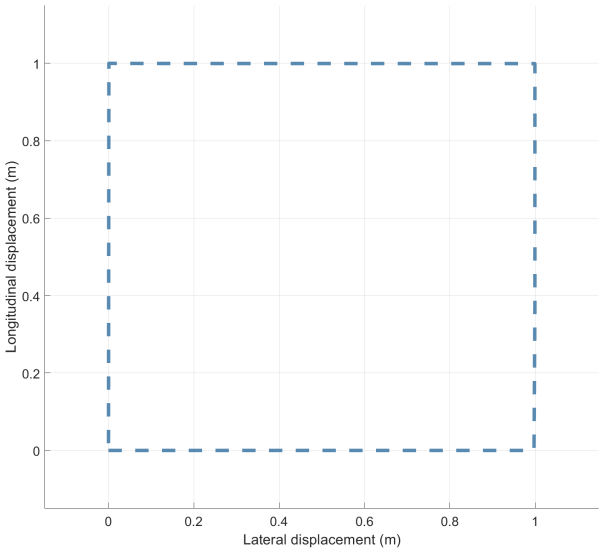


Figure 6.3: Reference path

6.3.2. Performance Metrics

A set of performance metrics must be determined to evaluate the performance of both controllers. The main metrics are the tracking error metrics. The Root Mean Square Error $(RMSE)$ is a measure of the path deviation over the whole trajectory. The RMSE is calculated using Eq. 6.1a The $RMSE$ does however not show the outliers of the error. Therefore the Maximum Error $(MaxE)$ must also be taken into account. The MaxE is calculated using Eq. 6.1b.

As the controllers are implemented on a real-time embedded system, the computational load must be limited. The computational load must not exceed the capabilities of the onboard PC. This metric is

checked in the experimental phase.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - x_{i,ref})^2} \tag{6.1a}$$

$$MaxE = max\left(\sqrt{(x_i - x_{i,ref})^2 + (y_i - y_{i,ref})^2}\right) \tag{6.1b}$$

6.3.3. Performance Evaluation: High Friction

Fig. 6.4a shows the traversed path of the vehicle in test 1 & 2, with high friction between the wheels and surface. The difference between the two controllers is not notable from this diagram. As seen in Tab. 6.2, the *RMSE* in longitudinal as well as lateral direction of the vehicle, in tests 1 and 2 is only 2 mm apart over a trajectory of 4 m distance. Also, the *MaxE* in longitudinal as well as lateral direction is zero, rounded to the nearest thousandth.

A difference can however be noted in the angular wheel velocities and the jerk imposed on the wheels in Fig. 6.5a, and 6.5c respectively. As seen in Fig. 6.5c, the jerk is increased on the initial startup of the motors. This difference shows the increase in the absolute value of the gain. The same state error invokes a higher control input. This is consequent to the dynamic model-based control accounting for the slipping of the wheels, whereas the kinematic model-based controller does not. Therefore, to achieve the same acceleration, a higher input is required according to the dynamic model-based controller. The difference can be seen visually in Fig. 6.5a. Due to the increase in jerk and consequent acceleration, the vehicle can start decelerating earlier resulting in a reduced total effort. To calculate whether the amount of energy used has been reduced as well, the current would have to be calculated, but this is out of the scope of this project.

As seen in Fig. 6.5e, the slip velocity in the high-friction scenario is limited to peaks of $0.09\,\mathrm{m/s}$. Combining this information with the friction coefficient, as shown in Fig. 6.5g, it can be seen that the wheels operate in the linear region of the friction coefficient curve. This confirms again the previous observation that the amount of wheel slip is limited. This is also the main reason that the difference between the two controllers is limited in these scenarios. It is expected that when the vehicle operates in the nonlinear part of the friction coefficient curve, the difference between the two controllers becomes more apparent. This will be researched in the next section.

6.3.4. Performance Evaluation: Low Friction

As predicted in the previous subsection, the differences between the controllers become more visible when the friction coefficient is low, implying a more slippery surface such as a dusty wooden floor. Fig. 6.4 shows the traversed path of the simulated vehicle when again controlled by the kinematic model-based LQR, as well as the dynamic model-based LQR, in the low-friction scenario respectively. In this scenario, the vehicle overshoots its target by 67 mm as reflected by the *MaxE* in Tab. 6.2 when controlled by the kinematic model-based LQR. When controlled by the dynamic model-based LQR, the vehicle shows similar performance as in the high-friction scenario. As seen in Tab. 6.2, the *RMSE* in longitudinal as well as lateral direction of the vehicle in the low-friction scenario are increased by 48% and 30% compared to the high-friction scenario respectively. As expected from Fig. 6.4b, the increase in *RMSE* is less when the dynamic model-based LQR is applied. Similarly, as seen in Tab. 6.2, the *MaxE* is 67 mm and less than 0.1 mm when using the kinematic model-based and dynamic model-based LQR respectively.

Figures 6.6a & 6.6b show the lateral velocity curves over time of the simulated FMWV in high and low-friction scenarios respectively. From these figures, notably, the kinematic model-based controlled FMWV reaches a higher velocity magnitude of $0.65\,\mathrm{m/s}$ compared to $0.58\,\mathrm{m/s}$ of the dynamic model-based controller FMWV in the low-friction scenario and $0.75\,\mathrm{m/s}$ compared to $0.65\,\mathrm{m/s}$ in the low-friction scenario.

Also, it is notable that the maximum velocity magnitude in the high-friction scenario is lower than in

the low-friction scenario. This seems counter-intuitive. However, this is caused by the lagging of the position compared to the current reference position as seen in Fig. 6.6b. This causes the velocity to be increased further than the reference velocity to try to counteract the delta position in time. The kinematic model-based controller however does not take into account the friction limit of the wheels, which can be seen in Fig. 6.5h, and therefore is not able to control the velocity in a way where the overshoot in position is avoided.

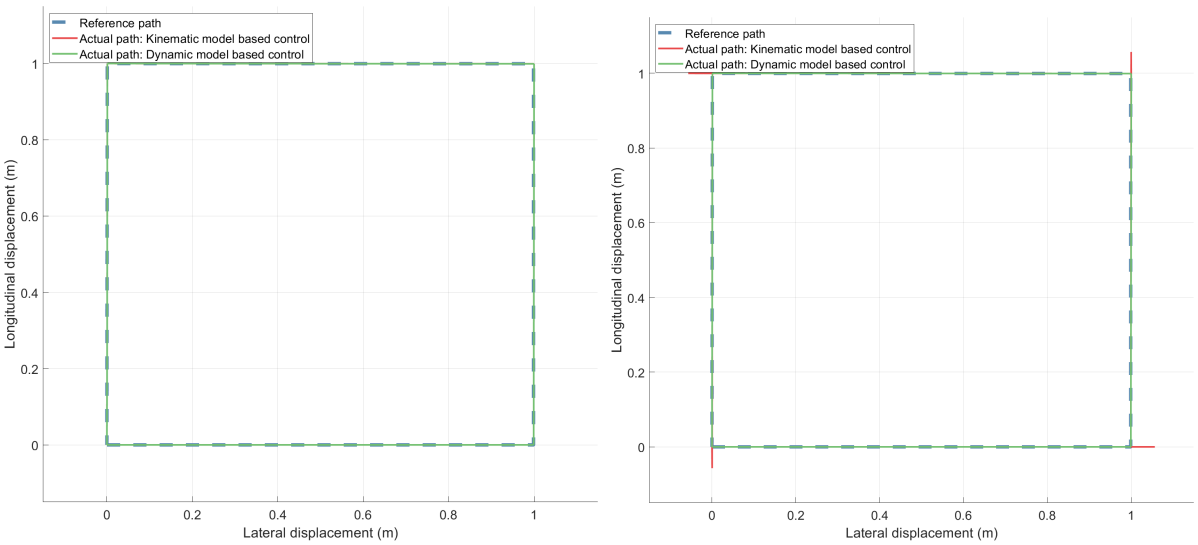


Figure 6.4: Cartesian diagrams of the traversed path in simulation using both the kinematic model-based LQR and the dynamic model-based LQR. The scenario is repeated with high friction, test 1 & 2, and with low friction, test 3 & 4.

Table 6.2: Key Performance Indicators from simulation of a kinematic model-based LQR, and a dynamic model-based LQR

Metric	Scenario							
	High friction				Low friction			
	Kinematic based (1)	Dynamic based (2)	Δ (absolute)	Δ (%)	Kinematic based (3)	Dynamic based (4)	Δ (absolute)	Δ (%)
RMSE longitudinal (mm)	58	60	2	3.4	86	78	-8	-8.8
RMSE lateral (mm)	58	60	2	3.4	86	78	-8	-8.8
RMSE angular (rad)	<0.1	<0.1	<0.1	-	<0.1	<0.1	<0.1	-
MaxE (mm)	<0.1	<0.1	<0.1	-	67	<0.1	-67	-100

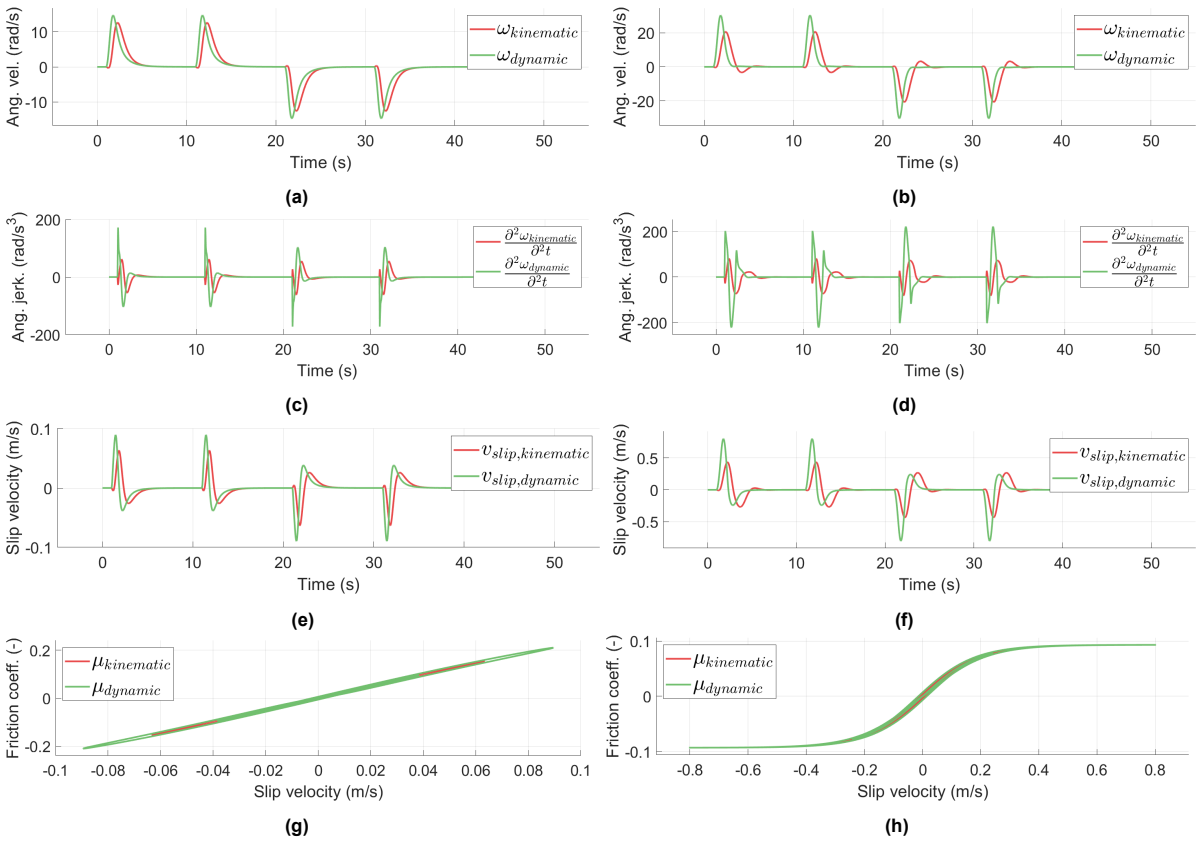


Figure 6.5: Comparison of the angular wheel velocity, angular wheel jerk, friction coefficient, and slip velocity of evaluated controllers in a square reference path with high- and low friction conditions respectively. Only data from wheel 1 is shown.

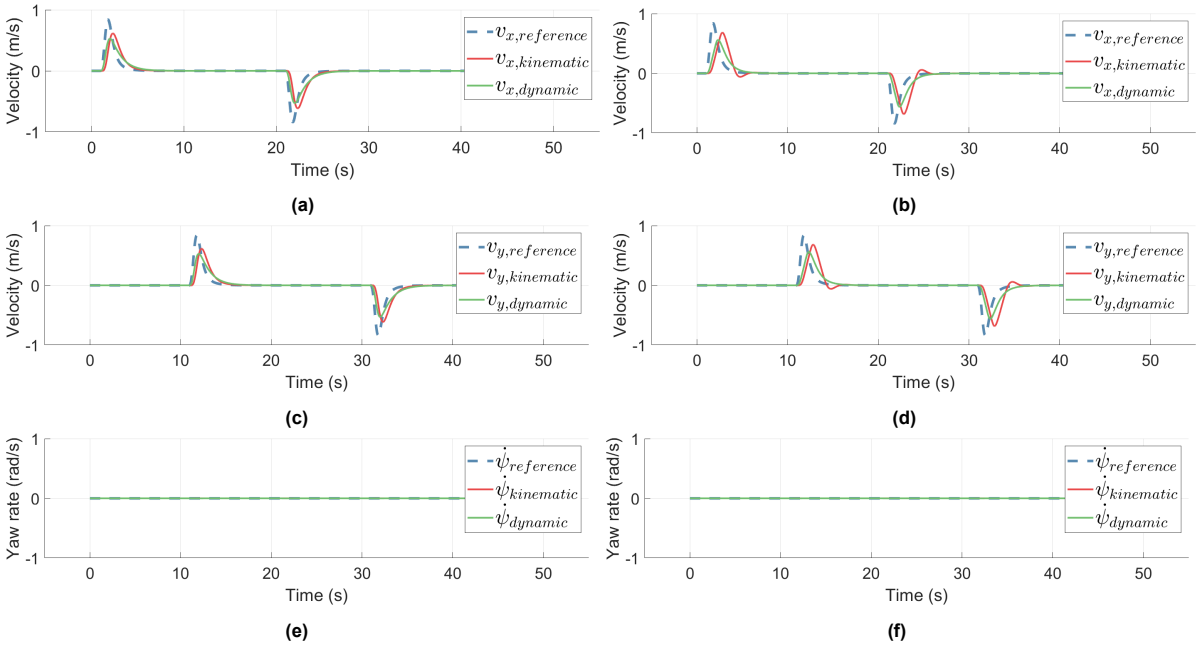


Figure 6.6: Comparison of velocity states of evaluated controllers in a square reference path with high- and low-friction scenarios respectively.

7

Experimental Validation

Following the methodology established in the simulation validation in Ch. 6, the controller's performance is assessed through a series of four tests, distributed across two distinct scenarios. This section begins with a comprehensive description of the experimental setup, detailing the environment, and procedures used to conduct the tests. Subsequently, the adjustments made to the control system in preparation for the experiments are explained. Finally, the vehicle's performance is evaluated, providing insights into its operational effectiveness under the tested conditions.

7.1. Experimental Setup

The vehicle must track a trajectory in a high-friction, and a low-friction scenario. The high-friction scenario is easily achieved as this is the nominal condition for most households. However, the challenge is to create a low friction surface to test the potential of the proposed dynamic model-based controller found in the simulation validation in Ch.6. As found by [46], the friction coefficient of a floor surface can be reduced by up to 92 % by introducing a sand-like material spread evenly on the floor surface. In a practical sense, sand was not an option due to space contamination. Therefore, flour was chosen as it is easy to clean using a vacuum cleaner but still similar to fine sand in a way that greatly reduces the available friction. It must be noted that the exact friction coefficient was not determined due to time constraints. The experimental low-friction scenario is shown in Fig. 7.1.

7.2. Control System Adjustments

In preparation for the experiments, the controllers designed in Ch.5 are adapted to the embedded platform. First, the control frequency was adapted to 25 Hz. This actuator frequency is chosen by considering the operating frequency of the Kalman filter which is 30 Hz. By choosing an actuator frequency just below the sensor frequency, it is made sure that for each new actuator command, a new measurement is used. At this set of operating frequencies, the four cores of the embedded PC do not surpass 60% of their capacity and operate at an average of 50% of their capacity. Operating around 50% of the embedded PC increases the robustness of the system, as it allows for unforeseen computations as they might occur.

One sacrifice that had to be made was the implementation of the SLAM algorithm. Without the control system running, this algorithm already saturated all four cores of the embedded PC. Using this implementation, it is measured that the update frequency of the vehicle's pose published by the SLAM algorithm did not surpass 0.5 Hz. This update frequency makes the SLAM system's contribution to the system's performance negligible. Therefore, it is decided that the full capabilities of the ROS network must be used. A separate PC was configured to operate in the same LAN network and corresponding ROS network as the robot. The localization using SLAM is then structured as follows. On the robot, the scan data taken from the LiDAR sensor is published on the scan topic at 10 Hz. As seen in Fig. 7.2, the SLAM software on the separate PC reads the scan data and publishes the calculated pose on the pose topic of the ROS network. The pose is set to update at 2 Hz. This is however only achieved when the grid of the map is constructed of 100 mm by 100 mm cells. The trade-off between accuracy

Figure 7.1: Experimental low-friction scenario created with a layer of flour on a wooden floor surface.

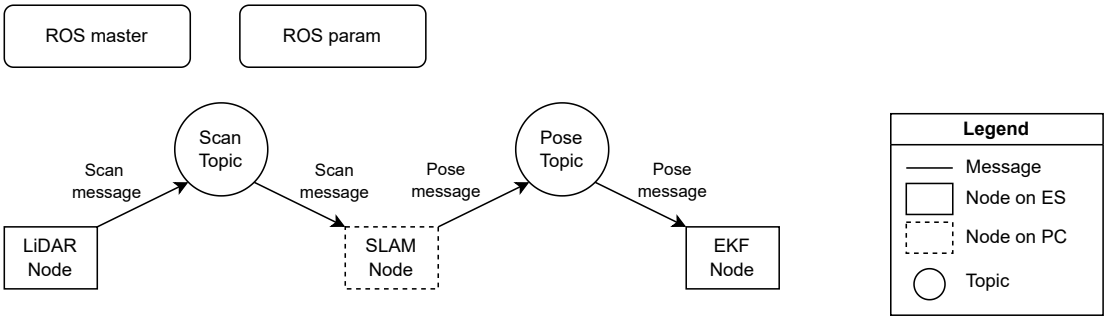


Figure 7.2: Overview of a ROS network where a SLAM node is run on a PC which communicates with the remaining ROS network on the embedded system.

and update frequency makes it so that the SLAM localization does not benefit to the trajectory tracking performance of the FMWV. During heuristic testing of the settings, it was even found that the performance was worsened because of the time lag the pose estimate from the SLAM algorithm contained. When the grid size is reduced, the lagging is no longer problematic. The location estimate however becomes precise to $100\,\text{mm}$, which is $10\,\%$ of one side of the square trajectory.
To complete the localization on the vehicle, the EKF is set up. To ensure the correct operation of the EKF, the variance of the pose and velocity measurements need to be determined. The variance is determined by keeping the vehicle stationary and recording the pose and velocity for each sensor for 1 minute. Of the obtained signals, the variance is calculated. The process noise of the EKF is determined heuristically according to the performance of the localization. Apart from that, the controller weights are tuned in an attempt to take into account the modeling errors that cause the differences between the simulations and the experiments.

7.3. Performance Evaluation: High Friction

Fig. 7.3a shows the traversed path of the vehicle in the high-friction scenario, controlled by the baseline controller. Fig. 7.3b shows the traversed path of the vehicle in the high-friction scenario, controlled by the proposed dynamic-model-based controller. This high-friction scenario experiment corresponds to the simulation result in Fig. 6.4a. This figure and analysis show that tracking with only minor refer-

ence deviations is expected from the simulation. However, the maximum absolute deviations of 77 mm and 68 mm from the reference path are noted for the baseline and proposed controller respectively in specific experiments. The mean maximum absolute deviations from the reference path are 47 mm and 54 mm, as listed in Tab. 7.1. Since the maximum deviations are 63 % and 26 % larger than the mean maximum deviations, these are considered outliers caused by localization error. A deviation in localization is further enhanced by actuator error and delay. Actuator error is when the expected output torque for a defined input voltage is not as expected from the datasheet. Actuator delay, however, is something that is expected in the system. This delay is the difference in time between the command given and the command performed. Command line saturation, motor dynamics, or computational delay cause this type of delay.

A consistent deviation in lateral displacement can be observed from Fig. 7.3 and Fig. 7.5. This deviation is consistent over both low and high-friction surface scenarios and for the two considered controllers. This deviation is explained by observing the experiment visually combined with the data. Visually it can be noted that when the vehicle is within 40 mm to the target position when moving in the lateral direction, the motor torque becomes lower than required to overcome the friction of the wheels to maintain a non-zero angular velocity. This type of friction is not modeled. It is the friction felt when rotating the wheels on one side of the vehicle to achieve lateral movement. The error is corrected as expected once the vehicle moves to the next reference point in a longitudinal direction.

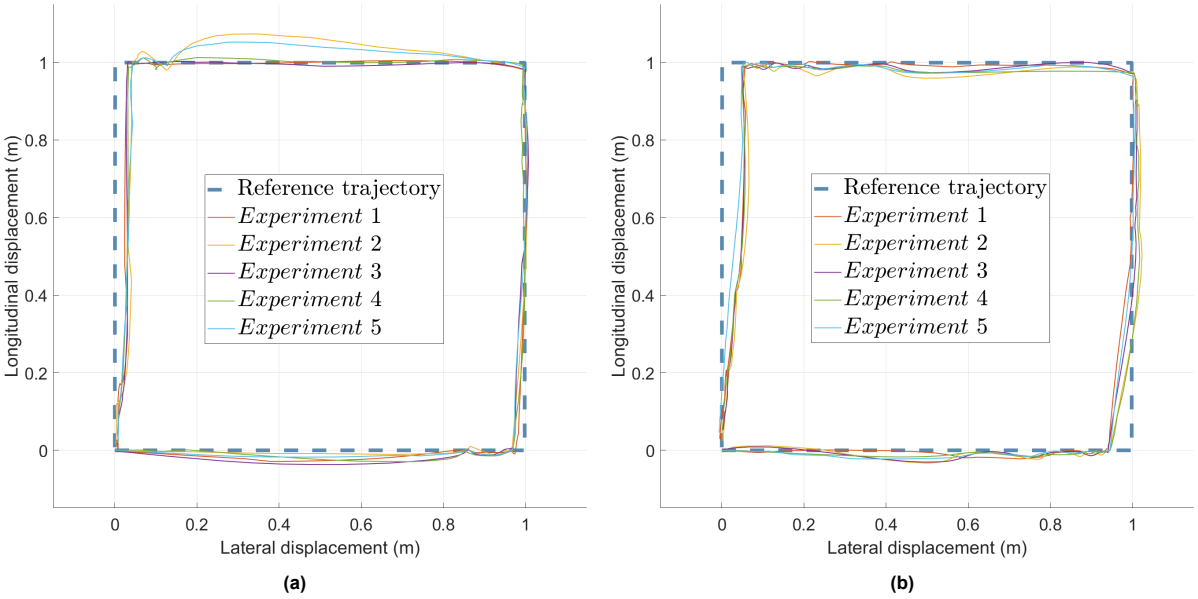


Figure 7.3: Cartesian diagrams of the traversed path in the experiment using both the baseline kinematic model-based LQR, **(a)**, and the dynamic model-based LQR **(b)**. The experiment is performed on a surface with high friction.

Analyzing Fig. 7.4, it can be noted that the vehicle's velocity in longitudinal and lateral directions is lagging for both the baseline and the proposed controller which is caused by a combination of actuator delay and wheel slip. It must also be noted that both controllers experience an oscillatory behavior in deceleration, however of a larger magnitude for the proposed controller. This oscillation is caused by the lack of resistive friction in the models used for the design of the controllers. According to the model, acceleration and deceleration of the vehicle are achieved with the same magnitude. However, in the experiment, it can be seen that in deceleration the magnitude of deceleration is higher due to the unmodeled resistive friction acting on the vehicle's and its wheels. The controller consequently receives an undershoot of the vehicle velocity as input and increases the control input. Combined with the actuator and sensor delay present in the system an oscillation of the vehicle velocity results.

As in Ch.6, KPIs are used to numerically compare the baseline and proposed controller's trajectory tracking performance. From the simulation analysis, both controllers are expected to perform similarly

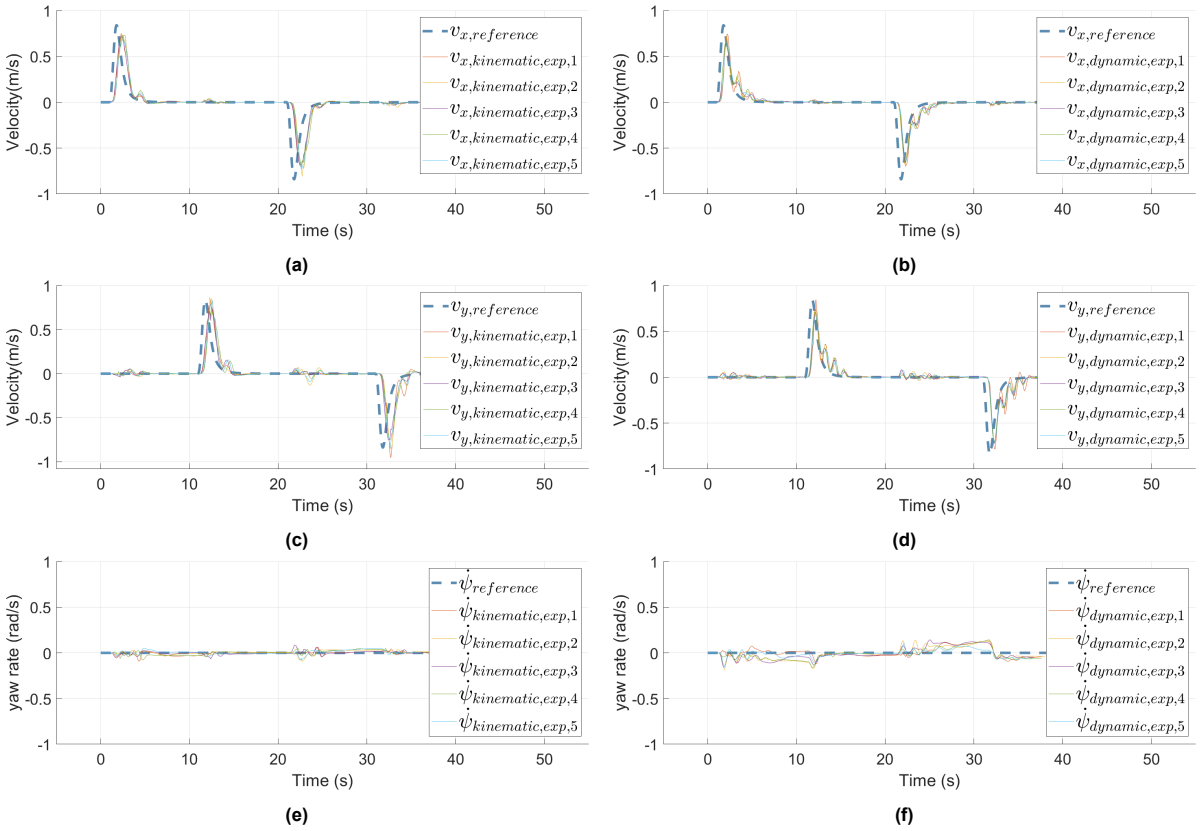


Figure 7.4: Comparison of velocity states in a square reference path using both the kinematic model-based LQR and the dynamic model-based LQR.

in the high-friction scenario. The KPIs from the experiments are listed in Tab. 7.1. It must be noted, that however the percentile differences are large, the absolute differences are small. The difference in mean RMSE is 8 mm and 18 mm for the longitudinal and lateral directions respectively. The difference in mean maximum error is 7 mm, which is negligible over a trajectory of 1 m by 1 m. These deviations can be attributed to localization errors. Therefore, it is concluded that the controllers behave similarly in the high-friction scenario, as expected from the simulation, however with deviation caused by localization error, actuator lag, and sensor lag.

7.4. Performance Evaluation: Low Friction

The simulation in Ch.6 shows and elaborates on the differences between the tracking performance of the baseline and the proposed controller in a low-friction scenario. In the simulation, the vehicle controlled by the baseline controller overshoots the reference trajectory. Fig. 7.5 shows the trajectory of the FMWV on a low friction surface. Fig. 7.5a shows the trajectory of five representative experiments where the FMWV is controlled by a kinematic model-based trajectory tracking LQR. Fig. 7.5b shows the trajectory of five representative experiments where the FMWV is controlled by a dynamic model-based trajectory tracking LQR. A total of 55 experiments were done to come to the results presented.

From these figures, it can be concluded that the result deviates from the simulation as the vehicle does not overshoot the trajectory for either controller in the low-friction scenario. Other differences are however apparent. From the trajectories, it can be noted that when the vehicle is controlled by the baseline controller in some experiments the path deviation has outliers of up to 74 mm where the proposed controller does not show such outliers. On the other hand, when the vehicle is controlled by the baseline controller it can be noted that the lateral error when moving towards a reference point laterally is reduced from 63 mm to 33 mm consistently. This behavior is discussed before in Sec. 7.3.

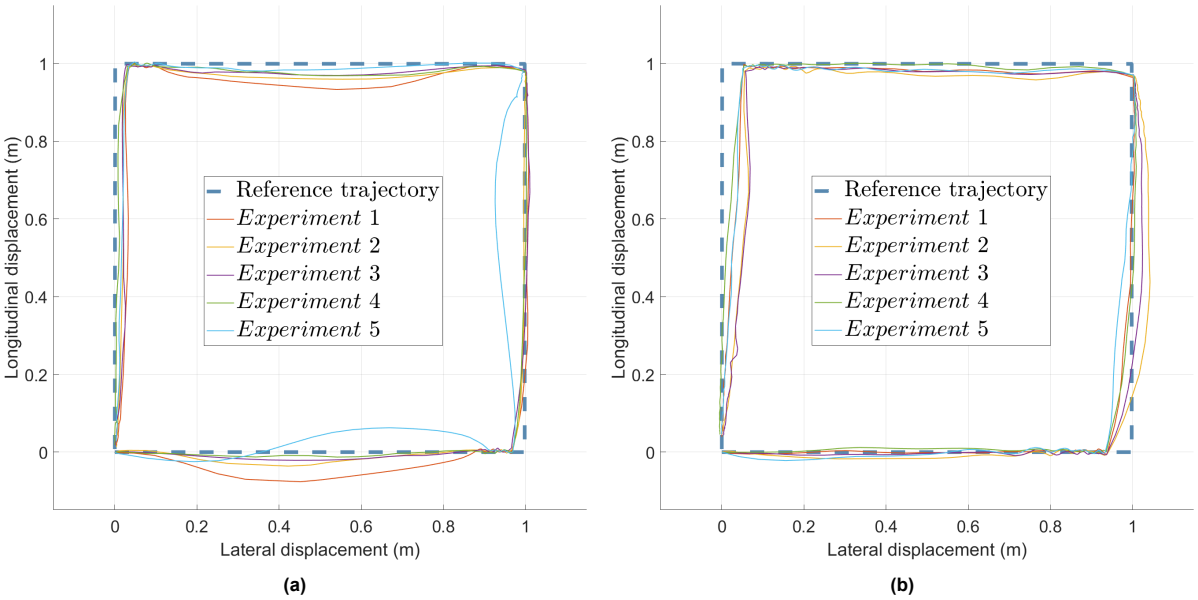


Figure 7.5: Cartesian diagrams of the traversed path in the experiment using both the baseline kinematic model-based LQR, **(a)**, and the dynamic model-based LQR **(b)**. The experiment is performed on a surface with low friction.

From Fig. 7.6 it can be noted that the maximum velocity both longitudinally and laterally is lower when the vehicle is controlled by the proposed controller compared to the baseline. The mean of the maximum lateral velocities v_x is reduced from $0.72\,\mathrm{m/s}$ to $0.65\,\mathrm{m/s}$. Also, the magnitude of the yaw rate is of greater magnitude when the vehicle is controlled by the proposed controller compared to the baseline. The mean of the maximum absolute yaw rate is increased from $0.13\,\mathrm{rad/s}$ to $0.18\,\mathrm{rad/s}$.

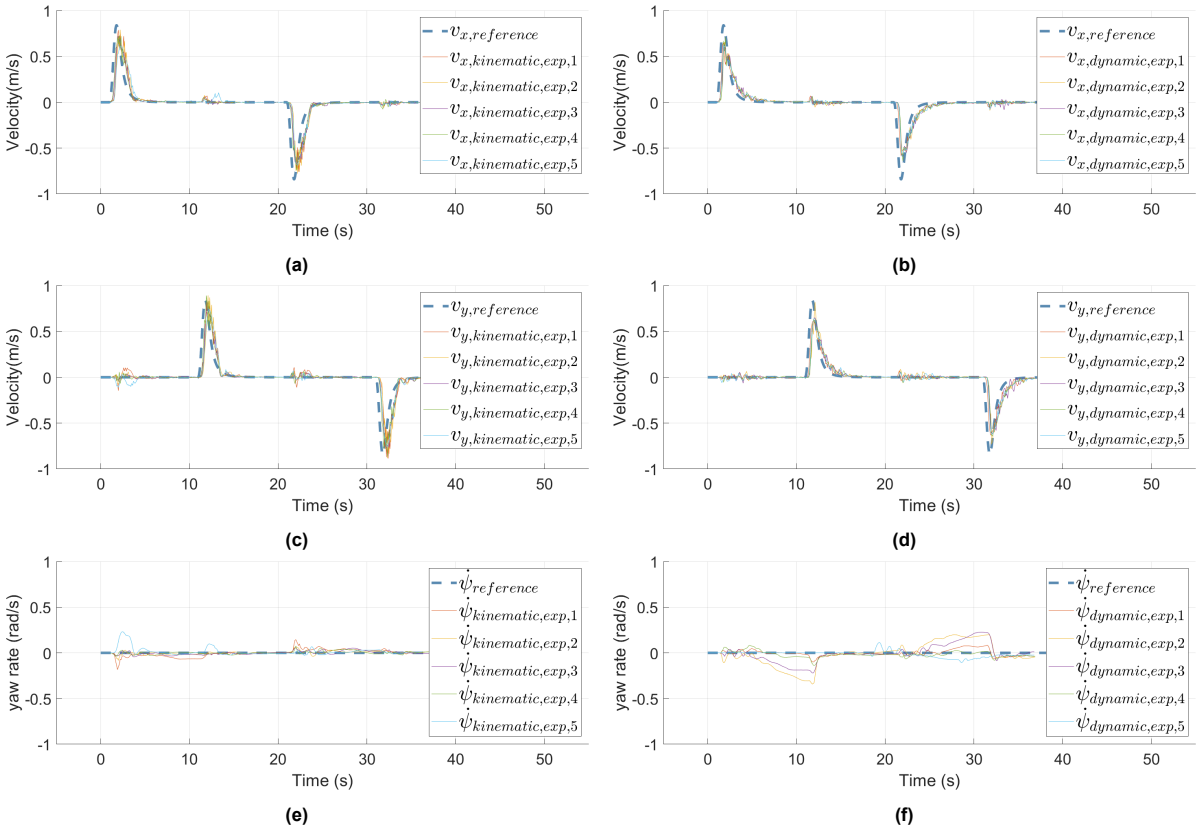


Figure 7.6: Comparison of velocity states in a square reference path using both the kinematic model-based LQR and the dynamic model-based LQR.

The KPIs for the low-friction scenario are listed in Tab. 7.1 below. In both longitudinal and lateral directions, the dynamic model-based controller shows a lower mean RMSE than the kinematic model-based controller. The mean angular RMSE is however increased when using the dynamic model-based control. The mean MaxE is similar for both controllers with only 3 mm difference, which could be attributed to localization error.

From these observations, it can be concluded that in the current experimental setup, the dynamic model-based controller outperforms the kinematic model-based controller in the low-friction scenario. More importantly, in both the high- and low-friction scenarios the dynamic model-based controller shows a reduction in means RMSE in translational movement.

Table 7.1: Key Performance Indicators from the experiments of a kinematic model-based LQR, and a dynamic model-based LQR trajectory tracker.

Metric	**Scenario**							
	High friction				**Low friction**			
	Kinematic based (1)	**Dynamic based (2)**	**Δ (absolute)**	**Δ (%)**	**Kinematic based (3)**	**Dynamic based (4)**	**Δ (absolute)**	**Δ (%)**
mean RMSE longitudinal (mm)	71	63	-8	-11.3	80	54	-26	-32.2
mean RMSE lateral (mm)	84	66	-18	-21.5	95	55	-39	-41.6
mean RMSE angular (rad)	0.022	0.060	0.038	142.6	0.028	0.068	0.040	175.1
mean MaxE (mm)	47	54	7	14.9	49	52	3	6.1

8

Conclusion

Trajectory tracking is defined as the ability of a vehicle to track a reference position and velocity over time. In this research, model-based-control is applied to achieve trajectory tracking of an FMWV to answer the research question: *What are the benefits of using a dynamic-model-based trajectory tracking controller, for a four-Mecanum-wheeled-vehicle, over a kinematic-based one?*

To answer this question, an FMWV is designed and built. This consists of designing and implementing the vehicle's hardware and software, the localization, the kinematic-model-based trajectory tracking control, and the dynamic-model-based trajectory tracking control.

The vehicle's hardware is designed according to a set of requirements. Each component is chosen according to these requirements. The PC and microcontroller, however, were adopted from the Mirte project, as the software from this project was also used as a basis for this research to save time in software development. The consequence of this design decision was that the software had to be designed such that the computational power matched that of the PC. This is achieved in all software parts, apart from the localization. For the localization, a combination of three types of sensors is used, combining to a total of 6 sensors. Four encoders, one IMU and one LiDAR were installed on the vehicle. The four encoders together provide a pose and velocity measurement. The IMU provides a linear and an angular acceleration measurement. The LiDAR provides laser scan measurements, which are processed by a SLAM algorithm to calculate the vehicle's pose and velocity. However, the SLAM algorithm is computationally too heavy for the OrangePi PC. An attempt is made to transfer the algorithm to a separate PC, however, the localization performance is insufficient.

Following the design of the vehicle's hardware, software, and localization, the plant model is defined. The plant model is constructed of a simplified tire model, a DC motor model, a translational dynamic model, and a yaw moment model. The result is a nonlinear dynamical model that incorporates friction estimation for an FMWV.

To design and implement two trajectory-tracking LQRs, two models are used. First, a kinematic model of an FMWV from the literature is adapted to this research by incorporating a DC motor model. The velocity state is expanded by including the pose in the world frame to formulate the new state space. This expansion makes the state dependent on the pose of the vehicle. Therefore, the pose in the world frame is transformed into the vehicle's reference frame. The same is done for the input to the controller to ensure correct operation. In this way, a linear state space is formulated, suitable for an LQR.

Second, the dynamical plant model is linearized for use in an LQR. The pose of the vehicle influences the linearization. Therefore, the model is linearized offline algebraically. At each controller time step, the linearization and consequent LQR gain are calculated numerically during runtime.

The simulation validation shows the benefits of using a dynamic model-based trajectory tracking controller in a low-friction scenario. In this low-friction scenario, the vehicle overshoots its target in longi-

tudinal and lateral directions using the kinematic model-based controller by 67 mm but does not with the dynamic model-based controller. This difference is explained by the lack of slip modeling in the kinematic model, which assumes a no-slip infinite friction scenario. The dynamic model incorporates a simplified tire model to estimate the available friction. Because of this, the vehicle tracks the trajectory without overshooting in the low-friction scenario. In the high-friction scenario, the behavior of the controllers is comparable, which is to be expected as the magnitude of the slip velocity is low enough that the tire operates in the linear region of the friction coefficient.

During experimental validation, it was again concluded that in the high-friction scenario, the performance of both controllers was comparable. The results from the experiments in the low-friction scenario differed from the expectation. The trajectory of neither controller overshot the reference as predicted in the simulation. Nonetheless, the benefits of using the dynamic model-based controller were shown. The dynamic model-based controller outperformed the kinematic model-based control in the low-friction scenario, both in consistency, and a reduction in translational longitudinal and lateral RMSE of 32.2 % and 41.6 % respectively.

From this research, it can be concluded that the benefits of using a dynamic-model-based trajectory tracking controller, for a four-Mecanum-wheeled-vehicle, over a kinematic-based one are apparent in low-friction conditions. In high-friction conditions, the performance is comparable. In the low-friction scenario, the simulation showed a reduction of $RMSE$ of 8.8 % in both longitudinal and lateral directions from the baseline to the proposed controller. The experimental validation showed a reduction of $RMSE$ of 32.2 % and 41.6 % respectively. The experimental validation differed from the simulation validation in MaxE as the simulation validation showed a decrease in $MaxE$ of 67 mm where the experimental validation showed an increase of $MaxE$ of 3 mm from the baseline to the proposed controller in the low-friction scenario. To determine the cause of this deviation, several limitations of the implemented FMWV are discussed below.

The results differed from the simulation, therefore, several possible sources of deviation are identified. The torque delivery could be improved, the localization performance could be improved by solving the LiDAR SLAM challenges, the quality of the Mecanum wheels could be improved, and the exact friction coefficients could be determined to reduce the difference between the simulation and experiment. In this way, the performance of the trajectory control could be improved upon further and resemble the simulation more closely.

One potential source of deviation is the rated torque. The experiment showed a difference with the simulation in lateral tracking performance, particularly when the reference point is within 50 mm distance from the vehicle. The control input, the motor voltages, is below the level needed to generate a torque large enough to maintain the rotation of the wheels. This could be solved by expanding the controller model such that the output is a set of wheel angular velocities. This way, a wheel angular velocity controller can be implemented to control the motor voltages to ensure the correct wheel angular velocity. Another solution could be to model the resistive friction in the plant model. This way, the friction can be taken into account in the design and tuning of the control system. Adding resistive friction to the controller model would be possible in the dynamic model-based controller as the model is force-based. For the kinematic model-based controller, integrating the nonlinear resistive friction model is more challenging as the friction would have to be modeled as a source of energy loss. A brute force method would be to change to motors with higher torque at lower RPM to reduce the stalling voltage. This will reduce the stalling distance to the reference point. This solution is seen as a last resort option.

Another factor to consider is the LiDAR SLAM trade-off. During the experiments, the trade-off between computational load and accuracy of the localization made it so that the LiDAR SLAM localization did not improve the pose estimate of the EKF. Either the frequency was adequate at 2 Hz, but with an accuracy of 100 mm or the accuracy was adequate with an accuracy of 25 mm but with a frequency of 0.5 Hz and consequent lag of up to 2 s. The result of the trade-off can be improved in two ways. First, the SLAM algorithm could be optimized. MathWorks, the producer of the Matlab Simulink software package, created the algorithm used in the implementation and was therefore expected to be a high-level implementation. It could be possible that this implementation is not optimal for the use in

ROS on an OrangePi Zero2. Therefore, a ROS-optimized SLAM algorithm might be the solution to reducing the computational load of the SLAM algorithm. Second, the computational hardware could be changed to a more powerful processor such that the update frequency will be optimized by increasing the number of parallel computations or speeding up the single computations. Another way to handle the sensor lag of the SLAM pose estimate could be to incorporate a strategy in the EKF that deals with Out-Of-Sequence-Measurements (OOSM) [47].

The quality of the Mecanum wheels is also a critical aspect. During testing, it was found that the quality of the Mecanum wheels could influence the performance of the controller. It was found that some of the rollers had a higher resistance to rotation on the axis. The consequence of this resistance is that when this particular roller with more resistance is active, instead of free rolling as modeled, the roller exerts a force in the direction orthogonal to the axis of the roller therefore disturbing the vehicle's motion. However labor-intensive, this could be improved by lubricating the axis of rotation of the rollers after a set of experiments. Another solution would be to use higher-quality Mecanum wheels that incorporate roller bearings pressed into the rollers, as seen in Fig.8.1, instead of a metal axle with a plastic roller without roller bearing.

Figure 8.1: Roller of a Mecanum wheel with integrated bearing[48].

Another structural issue is the rigidity of the frame. The frame used for mounting all sensors and actuators is not equipped with independent suspension for each wheel. To ensure all wheels make equal contact with the surface, the frame came equipped with a hinge at the center of the frame such that both the vehicle's axles could rotate freely around the vehicle's longitudinal center line. The downside of this hinge however is that over time the low-quality hinge developed slack. Therefore the two halves of the frame could rotate and move longitudinally slightly in respect to each other. This distorts the location and orientation of the wheels. This is a source of error for trajectory tracking. This could be improved by removing the hinge to make a solid frame. The vehicle is expected to operate on a flat surface, so a hinge is not required.

Lastly, determining the exact friction coefficients could greatly enhance the accuracy of the plant model. The experimental friction coefficient could be measured such that the controller can be tuned in the simulation. This could be a contributing factor explaining the difference between the simulation and the experiment.

References

[1] J. Palacín et al. “Phasor-Like Interpretation of the Angular Velocity of the Wheels of Omnidirectional Mobile Robots”. In: *Machines* 11 (2023).

[2] *William Grey Walter: Machina Speculatrix [Online]*. (2024, Jul, 16). URL: https://www.ros.org/blog/2021-06-29-machina-speculatrix/.

[3] *Hopkins Beast autonomous robot mod 2 with vision [Online]*. (2024, Jul, 16). URL: https://cyberneticzoo.com/cyberneticanimals/1962-5-hopkins-beast-autonomous-robot-mod-ii-sonarvision-jhu-apl-american/.

[4] *Shakey the Robot [Online]*. (2024, Jul, 16). URL: https://www.sri.com/hoi/shakey-the-robot/.

[5] L. Yang et al. “Path Planning Technique for Mobile Robots: A Review”. In: *Machines* 11.10 (2023), p. 980.

[6] J. Grabowiecki. “Vehicle Wheel”. US Patent: 1,303,535. 1919.

[7] B. E. Ilon. “Wheels for a course stable self-propelling vehicle movable in any desired direction on the ground or some other base”. US Patent: 3,876,255. 1975.

[8] D. Nedelkovski. *Arduino Mecanum Wheels Robot*. (2024, Jun, 6). URL: https://howtomechatronics.com/projects/arduino-mecanum-wheels-robot/.

[9] F. Jacob et al. “Picking with a robot colleague: A systematic literature review and evaluation of technology acceptance in human–robot collaborative warehouses”. In: *Comput. Ind. Eng.* 180 (2023), p. 109262.

[10] I. Kubasakova et al. “Implementation of Automated Guided Vehicles for the Automation of Selected Processes and Elimination of Collisions between Handling Equipment and Humans in the Warehouse”. In: *Sensors* 24.3 (2024).

[11] L. Xie, K. Stol, and W. Xu. “Energy-Optimal Motion Trajectory of an Omni-Directional Mecanum-Wheeled Robot via Polynomial Functions”. In: *Robotica* 38.8 (2020), pp. 1400–1414.

[12] P. Muir and C. Neuman. “Kinematic modeling for feedback control of an omnidirectional wheeled mobile robot”. In: *IEEE Int. Conf. Robot. Autom.* (1987), pp. 1772–1778.

[13] I. Doroftei, V. Grosu, and V. Spinu. “Omnidirectional Mobile Robot - Design and implementation”. In: *Bioinspiration and Robotics Walking and Climbing Robots* (2007), pp. 511–528.

[14] B. Van de wal et al. “Simplified Wheel Slip Modeling and Estimation for Omnidirectional Vehicles”. In: *IEEE Int. Conf. on Advanced Motion Control (AMC)*. 17th. 2022, pp. 389–395.

[15] E. Matsinos. “Modelling of the motion of a Mecanum-wheeled vehicle”. In: *arXiv* (2012).

[16] X. Chen, L. Cheng, and T. Li. “Adaptive Motion Tracking Control for Omnidirectional Mobile Robots Based on Characteristic Model”. In: *41st Chinese Control Conf. (CCC)*. 2022, pp. 2876–2881.

[17] E. Malayjerdi, H. Kalani, and M. Malayjerdi. “Self-Tuning Fuzzy PID Control of a Four-Mecanum Wheel Omni-directional Mobile Platform”. In: *Electrical Engineering (ICEE), Iranian Conf. on*. 2018, pp. 816–820.

[18] P. Viboonchaicheep, A. Shimada, and Y. Kosaka. “Position rectification control for Mecanum wheeled omni-directional vehicles”. In: 1 (2003), 854–859 vol.1.

[19] R. Pizá et al. “Nonuniform Dual-Rate Extended Kalman-Filter-Based Sensor Fusion for Path-Following Control of a Holonomic Mobile Robot with Four Mecanum Wheels”. In: *Appl. Sci.* 12.7 (2022).

[20] V. Alakshendra and S. Chiddarwar. “A robust adaptive control of mecanum wheel mobile robot: simulation and experimental validation”. In: *Int. Conf. on Intelligent Robots and Systems (IROS)*. 2016, pp. 5606–5611.

[21] Z. Sun et al. “Path-following control of Mecanum-wheels omnidirectional mobile robots using nonsingular terminal sliding mode”. In: *Mech. Syst. Signal Process* 147 (2021), p. 107128.

[22] X. Yu, Y. Feng, and Z. Man. “Terminal Sliding Mode Control – An Overview”. In: *IEEE Open J. Ind. Electron. Soc.* 2 (2021).

[23] Z. Sun et al. “Fuzzy adaptive recursive terminal sliding mode control for an agricultural omnidirectional mobile robot”. In: *Comput. Electr. Eng.* 105 (2023), p. 108529.

[24] Y. Han and Q. Zhu. “Robust Optimal Control of Omni-directional Mobile Robot using Model Predictive Control Method”. In: *2019 Chinese Control Conf. (CCC)*. 2019, pp. 4679–4684.

[25] A. A. Umar and J.S. Kim. “Nonlinear model predictive path-following for Mecanum-wheeled omnidirectional mobile robot”. In: *Trans. Korean Inst. Electr. Eng.* 70.12 (2021), pp. 1946–1952.

[26] L. Huang. “Review on Lidar-based Slam Techniques”. In: *Int. Conf. Signal Process. Mach. Learn. (CONF-SPML)* (2021).

[27] C. Hungar. “Map-based Localization for Automated Vehicles using LiDAR Features”. In: *IFAC-PapersOnLine* 50 (2021), pp. 276–281.

[28] J. Zhang and S. Singh. “Visual-lidar odometry and mapping: low-drift, robust, and fast”. In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2015, pp. 2174–2181.

[29] X. Wang, D. Ding, and W. Fu. “A Robust Lidar-Inertial Localization System Based on Outlier Removal”. In: *China Automation Congress (CAC)*. 2021, pp. 2420–2425.

[30] T. Moore and D. Stouch. “A Generalized Extended Kalman Filter Implementation for the Robot Operating System”. In: *Proc. 13th Int. Conf. Intell. Auton. Syst. (IAS-13)*. Springer, 2014.

[31] V. Varshney, R. K. Goel, and M. A. Qadeer. “Indoor positioning system using Wi-Fi & Bluetooth Low Energy technology”. In: *13th Int. Conf. Wireless Opt. Commun. Networks (WOCN)*. 2016, pp. 1–6.

[32] *Orange Pi Zero2 [Online]*. (2024, Jul, 9). URL: `http://www.orangepi.org/html/hardWare/computerAndMicrocontrollers/details/Orange-Pi-Zero-2.html`.

[33] *Raspberry Pi Pico/Pico H/Pico W RP2040-Based MCU Boards [Online]*. (2024, Jul, 9). URL: `https://nl.mouser.com/new/raspberry-pi/raspberry-pi-pico-boards/`.

[34] *IMU Breakout - MPU-9250 [Online]*. (2024, Jun, 6). URL: `http://wiki.sunfounder.cc/index.php?title=IMU_Breakout_-_MPU-9250`.

[35] *OKdo Lidar Module with Bracket [Online]*. (2024, Jun, 6). URL: `https://www.okdo.com/nl/p/okdo-lidar-module-with-bracket/`.

[36] ROS. *Why ROS? [Online]*. (2024, Jun, 6). URL: `https://www.ros.org/blog/why-ros/`.

[37] Robohouse. *Mirte Documentation v0.1 [Online]*. (2024, Jun, 6). URL: `https://docs.mirte.org/`.

[38] M. Hijikata, R. Miyagusuku, and K. Ozaki. “Wheel Arrangement of Four Omni Wheel Mobile Robot for Compactness”. In: *Appl. Sci.* 12.12 (2022), p. 5798.

[39] I. Moreno-Caireta, E. Celaya, and L. Ros. “Model Predictive Control for a Mecanum-wheeled Robot Navigating among Obstacles”. In: *IFAC-PapersOnLine* 54 (2021), pp. 119–125.

[40] X. Yu, Y. Feng, and Z. Man. “Terminal Sliding Mode Control – An Overview”. In: *IEEE Open J. Ind. Electron. Soc.* 2 (2021), pp. 36–52.

[41] A. Hughes and B. Drury. “Chapter 3 - D.C. motors”. In: *Electric Motors and Drives*. Fifth Edition. Newnes, 2019, pp. 89–129.

[42] M. Bersani et al. “Vehicle state estimation based on Kalman filters”. In: *Int. Conf. Electr. Electron. Technol. Automot.* 2019, pp. 1–6.

[43] M. A. Khan et al. “Nonlinear Control Design of a Half-Car Model Using Feedback Linearization and an LQR Controller”. In: *Appl. Sci.* 10 (2020).

[44] A. Altalbe A. Shahzad. “Computing of LQR Technique for Nonlinear System Using Local Approximation”. In: *Comput. Syst. Sci. Eng.* 46 (2023), pp. 853–871.

[45] M. Bersani et al. “Vehicle state estimation based on Kalman filters”. In: *Int. Conf. Electr. Electron. Technol. Automot.(AEIT)* (2019), pp. 1–6.

[46] K. Li et al. “Friction Measurements on Three Commonly Used Floors on a College Campus under Dry, Wet, and Sand-covered Conditions”. In: *Saf. Sci.* 45 (2007), pp. 980–992.

[47] S.R. Maskell et al. “Multi-target out-of-sequence data association: Tracking using graphical models”. In: *Information Fusion* 7 (2006), pp. 434–447.

[48] *Mecanum Aluminium Wheel Roller Set [Online]*. (2024, Jul, 9). URL: `https://robu.in/product/152mm-mecanum-wheel-roller-set-bearing-type-1pcs/`.

A

Datasheet DC motor

JGA25-370 Geared Motor

SKU 114090046

Please use this motor as an alternative to JGB37-371 and Encoder Geared Motor JGA25-371

What is the Geared motor?

The geared motor uses a gear set to convert the original high speed and low torque of the motor to a low speed and high torque state. So what are the benefits of geared motors? Under the same voltage conditions, you can manually clamp the motor to stop it, but once it is a gear motor, it is more difficult to stop the motor with an external force because the "force" of the motor becomes larger. Therefore, when you use a geared motor, you will find it is slower than a motor that does not slow down, but it can provide a larger load. Geared motors are typically used where high torque is required, such as an elevator, which will carry more than a dozen people upstairs, which will require a lot of torque. Of course, there will be some energy loss during deceleration, but it will still bring a lot of convenience to our lives.

Introduction:

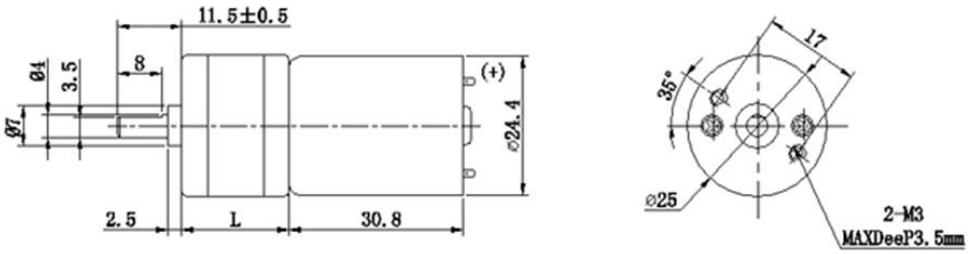
This Motor not encoder, Mainly used in robot platform and car provides power, Good quality and long lifetime, high torque and low noise.

If you need an encoder, you can choose JGA25-371 Geared Motor with Encoder.

Specification:

Voltage V	No-load		Maximum efficiency pointed				Blockage	
	speed r/min	electric current A	speed r/min	electric current A	Torque Kg.cm	Power W	Torque Kg.cm	electric current A
6	190	0.2	133	0.5	0.75	1.1	4.0	2.1
12	350	0.1	245	0.65	1.4	2.4	5.2	2.2

size:



L=21

Part List

1 x JGA25-370 Geared Motor

ECCN/HTS

ECCN	ERA99
HSCODE	8501101000
UPC	

B

Definition of Variables

Table B.1: Definitions of variables [14].

Variable	Definition
$\vec{v}_{\text{slip},i}$	slip velocity of wheel i
α_i	angular coordinate of wheel i in vehicle frame
γ_i	orientation of rollers of wheel i
d	distance between wheel and vehicle center
r	wheel radius
$\dot{\varphi}_i$	angular velocity of wheel i
$\dot{\vec{p}}$	vehicle velocity in world frame $\begin{bmatrix} v_x & v_y & \omega_z \end{bmatrix}^T$
m	vehicle mass
c_1, c_2	model parameters to be estimated

Table B.2: Definitions of variables [24].

Variable	Definition
m	mass of vehicle
J	moment of inertia
F^c	coulomb friction
F^v	viscous friction
δ	disturbance caused by slipping
u_a	armature voltage
i_a	armature current
L_a	armature inductance
R_o	resistance
l	motor reduction
K_1, K_2	constants

C

EKF Configuration File

```
#Configuation for robot odometry EKF
frequency: 30
sensor_timeout: 2
two_d_mode: true
transform_time_offset: 0.0
transform_timeout: 0.0
print_diagnostics: true
debug: false

map_frame: map
odom_frame: odom
base_link_frame: base_link
world_frame: odom
publish_tf: true
# -------------------------------------
# LiDAR SLAM:

pose0: /pose
pose0_config: [false,  false,  false,
               false, false, false,
               false, false, false,
               false, false, false,
               false, false, false]
pose0_differential: true
pose0_relative: false
pose0_queue_size: 5
pose0_rejection_threshold: 2  # Note the difference in parameter name
pose0_nodelay: false
# -------------------------------------
# Wheel odometry:

odom0: /mobile_base_controller/odom
odom0_config: [true, true, false,
               false, false, true,
               true,  true,  false,
               false, false, true,
               false, false, false]
odom0_queue_size: 10
odom0_nodelay: false
```

```
odom0_differential: false
odom0_relative: false

# Complete the imu0 configuration
# imu configure:

imu0: /mirte/movement/imu
imu0_config: [false, false, false,
              false, false, false,
              false, false, false,
              true, true, false,
              true,  true,  false]
imu0_nodelay: false
imu0_differential: false
imu0_relative: false
imu0_queue_size: 10
imu0_remove_gravitational_acceleration: true

use_control: false

process_noise_covariance: [1e-2, 0,    0,    0,    0,    0,    0,     0,
       0,    0,    0,    0,    0,    0,    0,
                             0,    1e-2, 0,    0,    0,    0,
                                0,    0,    0,    0,    0,    0,    0,
                                   0,
                             0,    0,    1e-2, 0,    0,    0,
                                0,    0,    0,    0,    0,    0,    0,
                                   0,
                             0,    0,    0,    1e-2, 0,    0,    0,
                                0,    0,    0,    0,    0,    0,    0,
                                   0,
                             0,    0,    0,    0,    1e-2, 0,    0,
                                0,    0,    0,    0,    0,    0,    0,
                                   0,
                             0,    0,    0,    0,    0,    1e-2, 0,
                                0,    0,    0,    0,    0,    0,    0,
                                   0,
                             0,    0,    0,    0,    0,    0,    1e-2,
                                0,    0,    0,    0,    0,    0,    0,
                                   0,
                             0,    0,    0,    0,    0,    0,    0,     1
                                e-2,  0,    0,    0,    0,    0,    0,
                                   0,
                             0,    0,    0,    0,    0,    0,    0,
                                0,    1e-2, 0,    0,    0,    0,    0,
                                   0,
                             0,    0,    0,    0,    0,    0,    0,
                                0,    0,    1e-2, 0,    0,    0,    0,
                                   0,
                             0,    0,    0,    0,    0,    0,    0,
                                0,    0,    0,    1e-2, 0,    0,    0,
                                   0,
                             0,    0,    0,    0,    0,    0,    0,
                                0,    0,    0,    0,    1e-2, 0,    0,
                                   0,
                             0,    0,    0,    0,    0,    0,    0,
```

```
                                    0,     0,    0,    0,    0,    1e-2, 0,
                                       0,
                                 0,    0,    0,    0,    0,    0,    0,
                                    0,     0,    0,    0,    0,    0,    1e
                                    -2, 0,
                                 0,    0,    0,    0,    0,    0,    0,
                                    0,     0,    0,    0,    0,    0,    0,
                                       1e-2]


initial_estimate_covariance: [1e-9,    0,    0,    0,    0,    0,    0,
       0,    0,    0,    0,    0,     0,    0,    0,
                                    0,    1e-9, 0,    0,    0,    0,    0,
                                       0,    0,    0,     0,     0,     0,
                                       0,    0,
                                    0,    0,    1e-9, 0,    0,    0,    0,
                                       0,    0,    0,     0,     0,     0,
                                       0,    0,
                                    0,    0,    0,    1e-9, 0,    0,    0,
                                       0,    0,    0,     0,     0,     0,
                                       0,    0,
                                    0,    0,    0,    0,    1e-9, 0,    0,
                                       0,    0,    0,     0,     0,     0,
                                       0,    0,
                                    0,    0,    0,    0,    0,    1e-9, 0,
                                       0,    0,    0,     0,     0,     0,
                                       0,    0,
                                    0,    0,    0,    0,    0,    0,    1e-9,
                                     0,    0,    0,     0,     0,     0,
                                       0,    0,
                                    0,    0,    0,    0,    0,    0,    0,
                                       1e-9, 0,    0,     0,     0,     0,
                                       0,    0,
                                    0,    0,    0,    0,    0,    0,    0,
                                       0,    1e-9, 0,     0,     0,     0,
                                       0,    0,
                                    0,    0,    0,    0,    0,    0,    0,
                                       0,    0,    1e-9,  0,     0,     0,
                                       0,    0,
                                    0,    0,    0,    0,    0,    0,    0,
                                       0,    0,    0,     1e-9,  0,     0,
                                       0,    0,
                                    0,    0,    0,    0,    0,    0,    0,
                                       0,    0,    0,     0,     1e-9,  0,
                                       0,    0,
                                    0,    0,    0,    0,    0,    0,    0,
                                       0,    0,    0,     0,     0,     1e
                                    -9, 0,    0,
                                    0,    0,    0,    0,    0,    0,    0,
                                       0,    0,    0,     0,     0,     0,
                                       1e-9, 0,
                                    0,    0,    0,    0,    0,    0,    0,
                                       0,    0,    0,     0,     0,     0,
                                       0,    1e-9]
```

D

Matlab Code

D.1. Simulink Initialization

```
%% simulation parameters
dt = 1e-3;              % step size:  0.1ms
ts = 50;                % simulation time
step_time = 10;         % input step time
stat1_dyn0 = 1;         % 1: fixed kinematic gain, 0: dynamic dynamical
    gain
input_idx  = 2;         % index to choose between input signals
denominator_transient_base = [1e-2 1];
step_vec = [0.5,1,0];   % input after step time
input_amplitude = 2;
input_frequency = pi/8;
update_freq_gain = 10;
coder.extrinsic("lqr");

%% vehicle parameters

g = 9.81;           % gravitational constant
m = 2;              % mass of vehicle
m_w = 0.163;        % mass of wheel: 163g
r_w = 0.05;         % wheel radius    5cm
tw = 0.125;         % half the trackwidth
wbh = 0.15;         % half wheelbase
k_torque = 0.2865;  % torque constant motor from datasheet
resistance = k_torque/0.5099458*12;   % resistance motor: calculated using
    blocking torque of motor
I = m*(tw^2+wbh^2); % inertia vehicle
I_wheel = (m_w*r_w^2)/2;              % inertia wheel
c1 = k_torque^2/I_wheel/resistance;   % state space constant 1
c2 = k_torque  /I_wheel/resistance;   % state space constant 2
wheel_speed_limit = 400*2*pi/60;      % wheel speed limit in rad/s
motor_voltage_limit = 12;             % motor input voltage limit
gamma = pi*[0.25; 0.75; 0.75; 0.25];  % angle of roller to wheels

% omega1 = 0.093;   % low friction
omega1 = 0.5;       % high friction
omega2 = 5;         % constant tire stiffness equivalent parameter

d = sqrt(wbh^2+tw^2);                 % distance from geo centre to wheel
```

```
alpha = [atan(wbh/-tw)+pi; atan(wbh/tw); atan(-wbh/-tw)+pi; atan(-wbh/tw)
    ];   % angle of wheel to geo centre xline
R = [-cos(gamma) -sin(gamma) -d*sin(gamma-alpha) r_w*sin(gamma)];
                % matrix for dynamic model

W = r_w/4*[         1,          -1,       -1,         1;
                            % wheel configuration matrix for kinematic model
                   1,           1,        1,         1;
          -1/(tw+wbh),1/(tw+wbh),-1/(tw+wbh),1/(tw+wbh)];


W_inv = -1/r_w*[1,  1, -(tw+wbh);
                                                % left inverse of wheel
    configuration matrix
              -1,  1,  (tw+wbh);
              -1,  1, -(tw+wbh);
               1,  1,  (tw+wbh)];

%% LQR

Rw = 1/144*diag([2,2,2,2]);                  % control weights
Qsim = diag([1e0,1e0,1e-9,1e1,1e1,1e-3]);    % output weights used in
    simulation
Qexp = diag([1e1,1e1,1e-3,1e3,1e3,1e-3]);    % output weights used in
    experiment
A = [                                        % system state matrix
    -c1   0   0 0 0 0;
    0   -c1   0 0 0 0;
    0     0 -c1 0 0 0;
    1     0   0 0 0 0;
    0     1   0 0 0 0;
    0     0   1 0 0 0
    ];
B = [c2*W;                                   % system input matrix
    zeros(3,4)];

K = lqr(A,B,Qsim,Rw);                        % kinematic model based gain
    calculation


%% lidar

covariance_lidar = [7.5e-07,       0,       0,        0,       0,    0, ...
                          0, 9.2e-07,       0,        0,       0,    0, ...
                          0,       0, 5.8e-07,        0,       0,    0, ...
                          0,       0,       0,     1e-3,       0,    0, ...
                          0,       0,       0,        0,    1e-3,    0, ...
                          0,       0,       0,        0,       0, 1e-3];
```

D.2. Algebraïc Linearization of Dynamic Model

```
clearvars -except K
%% parameters
gam = pi*[0.25; 0.75; 0.75; 0.25];  % angle(rad) of roller to wheel
g = 9.81;                           % gravitational constant (m/s^2)
m = 2;                              % mass (kg)
r = 0.1;                            % wheel radius (m)
```

```
7  tw = 0.125;                          % half the trackwidth (m)
8  wbh = 0.15;                          % half the wheelbase (m)
9  w1 = 0.8;                            % friction constant 1
10 w2 = 30;                             % friction constant 2
11 d = sqrt(wbh^2+tw^2);                % distance from geo center to wheel
12 k_torque = 0.25;                     % torque constant motor
13 alpha = [atan(wbh/-tw)+pi; atan(wbh/tw); atan(-wbh/-tw)+pi; atan(-wbh/tw)
       ];
14 I = m*(tw^2+wbh^2);                  % inertia of the vehicle
15 vx = sym('vx');                      % error state vx
16 vy = sym('vy');                      % error state vy
17 omega = sym('omega');                % error state Omega
18 x = sym('x');                        % error state x
19 y = sym('y');                        % error state y
20 psi = sym('psi');                    % error state psi
21 E = sym('E',[4 1]);                  % error input voltages of motors
22 vslip = sym('vslip',[4 1]);          % slip velocity of wheels (m/s)
23
24
25 %% model definition
26 for i = 1:4
27     vslip(i) = cos(gam(i))*vx + sin(gam(i))*vy + d*sin(gam(i)-alpha(i))*
           omega + r*sin(gam(i))/k_torque*E(i);
28 end
29 u = w1*tanh(w2*vslip);
30 F = m*g*u/4;                                % magnitude of force per wheel
31 Fx = cos(pi/4)*(F(1)-F(2)-F(3)+F(4));      % force in direction x on vehicle
32 Fy = sin(pi/4)*(F(1)+F(2)+F(3)+F(4));      % force in direction y on vehicle
33 T = (tw+wbh)/2*(-F(1)+F(2)-F(3)+F(4));
34
35 vx_dot = Fx/m;                             % acceleration in x direction
36 vy_dot = Fy/m;                             % acceleration in y direction
37 omega_dot = T/I;                           % angular acceleration around the
       z-axis
38 x_dot   = cos(psi) * vx - sin(psi) * vy;
39 y_dot   = sin(psi) * vx + cos(psi) * vy;
40 psi_dot = omega;
41
42 %% Jacobian calculation
43 A11 = diff(vx_dot, vx);
44 A12 = diff(vx_dot, vy);
45 A13 = diff(vx_dot, omega);
46 A14 = diff(vx_dot, x);
47 A15 = diff(vx_dot, y);
48 A16 = diff(vx_dot, psi);
49
50 A21 = diff(vy_dot, vx);
51 A22 = diff(vy_dot, vy);
52 A23 = diff(vy_dot, omega);
53 A24 = diff(vy_dot, x);
54 A25 = diff(vy_dot, y);
55 A26 = diff(vy_dot, psi);
56
57 A31 = diff(omega_dot, vx);
58 A32 = diff(omega_dot, vy);
59 A33 = diff(omega_dot, omega);
```

```
A34 = diff(omega_dot, x);
A35 = diff(omega_dot, y);
A36 = diff(omega_dot, psi);

A41 = diff(x_dot, vx);
A42 = diff(x_dot, vy);
A43 = diff(x_dot, omega);
A44 = diff(x_dot, x);
A45 = diff(x_dot, y);
A46 = diff(x_dot, psi);

A51 = diff(y_dot, vx);
A52 = diff(y_dot, vy);
A53 = diff(y_dot, omega);
A54 = diff(y_dot, x);
A55 = diff(y_dot, y);
A56 = diff(y_dot, psi);

A61 = diff(psi_dot, vx);
A62 = diff(psi_dot, vy);
A63 = diff(psi_dot, omega);
A64 = diff(psi_dot, x);
A65 = diff(psi_dot, y);
A66 = diff(psi_dot, psi);


B11 = diff(vx_dot,E(1));
B12 = diff(vx_dot,E(2));
B13 = diff(vx_dot,E(3));
B14 = diff(vx_dot,E(4));

B21 = diff(vy_dot,E(1));
B22 = diff(vy_dot,E(2));
B23 = diff(vy_dot,E(3));
B24 = diff(vy_dot,E(4));

B31 = diff(omega_dot, E(1));
B32 = diff(omega_dot, E(2));
B33 = diff(omega_dot, E(3));
B34 = diff(omega_dot, E(4));

A = [A11 A12 A13 A14 A15 A16; A21 A22 A23 A24 A25 A26; A31 A32 A33 A34 A35
    A36; A41 A42 A43 A44 A45 A46; A51 A52 A53 A54 A55 A56; A61 A62 A63 A64
    A65 A66;];
B = [B11 B12 B13 B14; B21 B22 B23 B24; B31 B32 B33 B34; zeros(3,4)];

Afunc = matlabFunction(A);
Bfunc = matlabFunction(B);

%% equilibrium calculation
Enew = solve([vx_dot; vy_dot; omega_dot] == [0;0;0], E); % finding wheel
   speeds at equalibrium as a function of vx, vy, & omega
Efunc = matlabFunction([Enew.E1;Enew.E2;Enew.E3;Enew.E4]);
```

E

Simulink Models

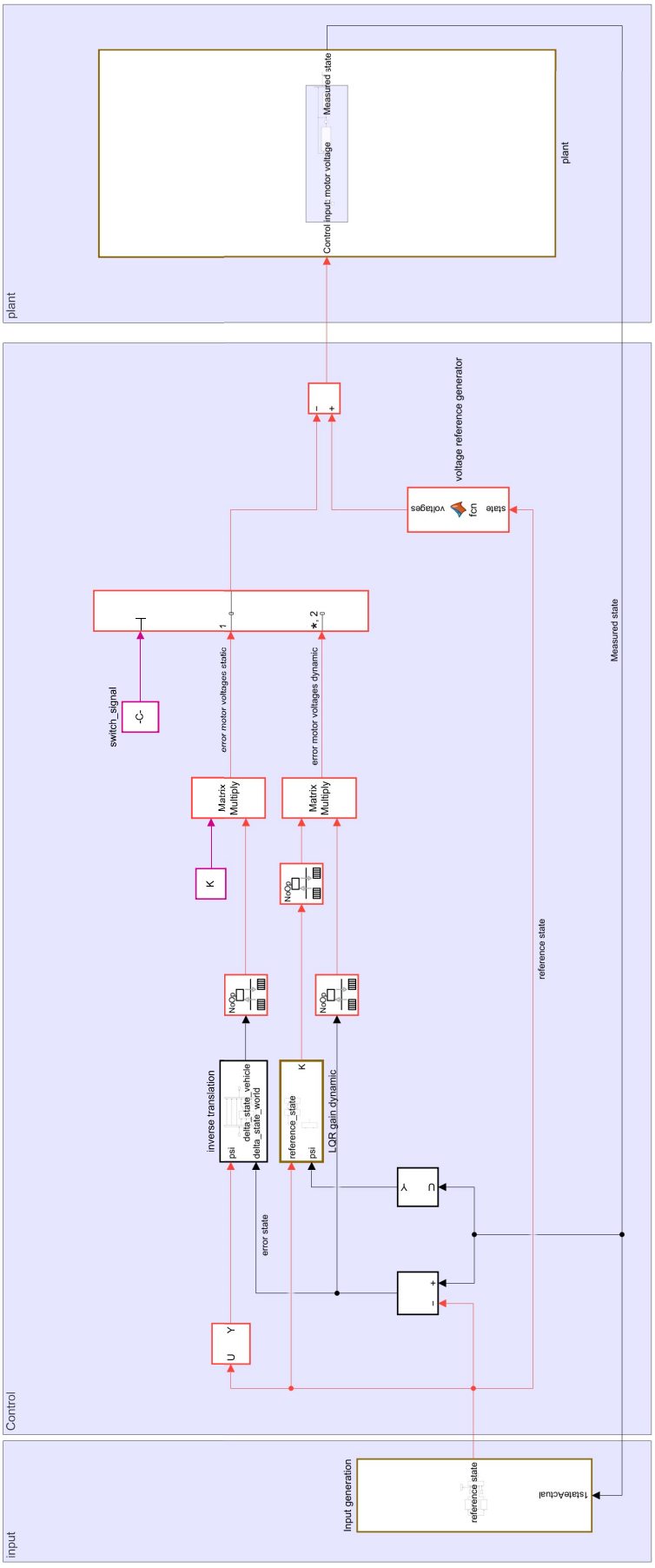


Figure E.1: Simulink implementation of the kinematic- and dynamic model-based trajectory tracking LQRs for simulation.

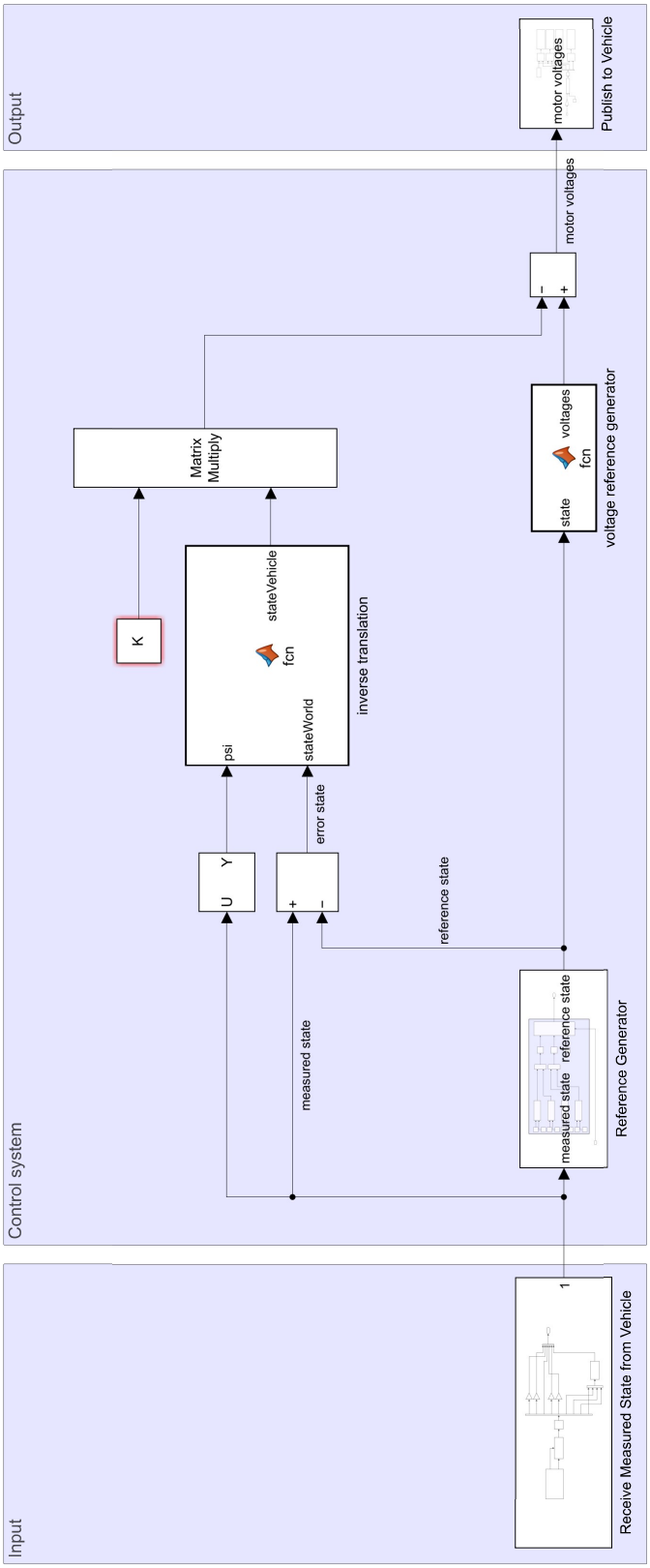


Figure E.2: Simulink implementation of the kinematic model-based trajectory tracking LQR

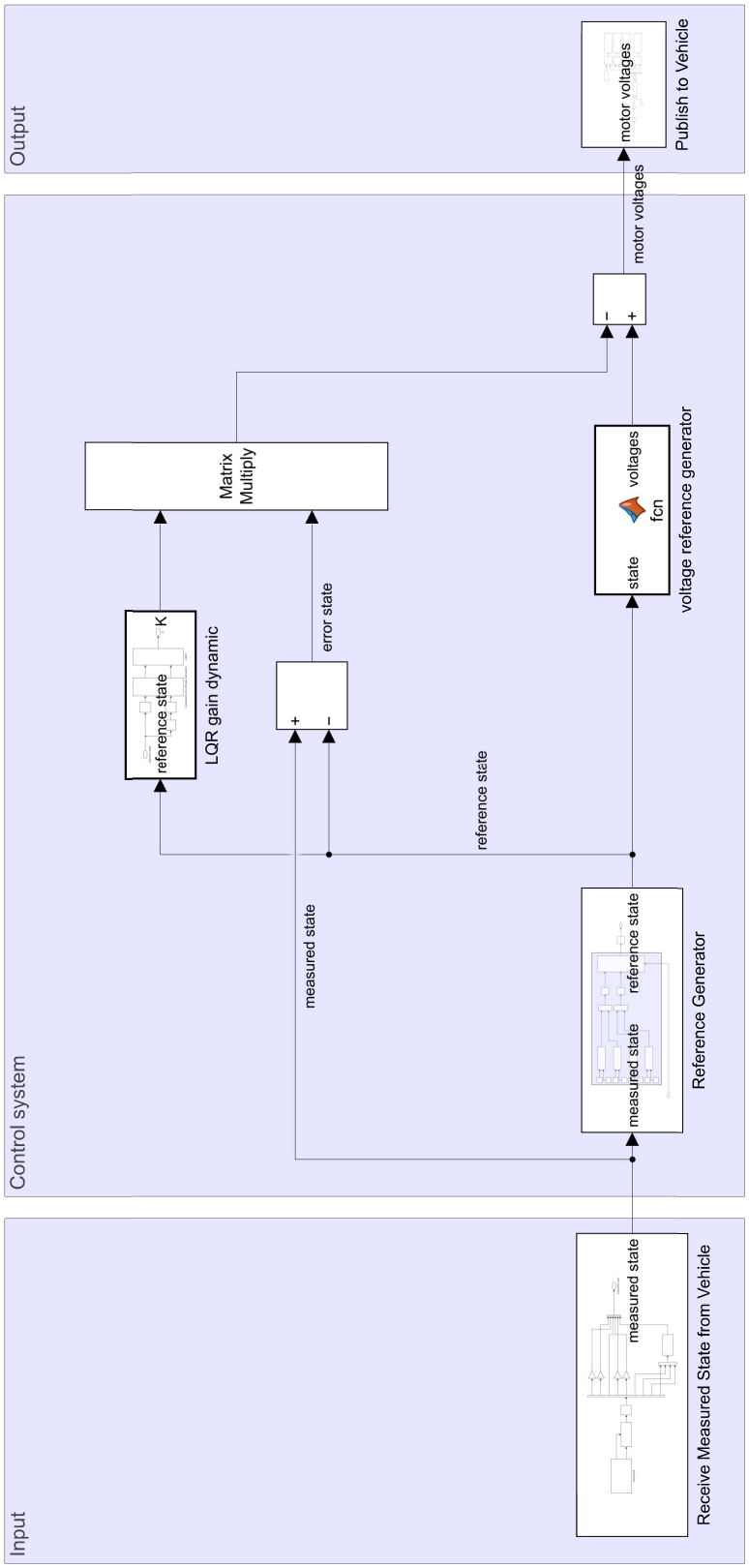


Figure E.3: Simulink implementation of the dynamic model-based trajectory tracking LQR