TÉCNICO LISBOA

Steering and Speed Control for Autonomous Electric Vehicles

Filipe Miguel Bronze Parrado

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors

Professor José António da Cruz Pinto Gaspar
Professor João Filipe Pereira Fernandes

Examination Committee

Chairperson: Professor João Fernando Cardoso Silva Sequeira
Supervisor: Professor José António da Cruz Pinto Gaspar
Member: Professor António Pedro Rodrigues Aguiar

December 2021

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Resumo

Nos últimos anos, os veículos autónomos têm estado na vanguarda dos estudos de engenharia como uma solução comercialmente viável. Existem muitos objetivos associados à condução autónoma, nomeadamente a minimização do erro entre o veículo e a trajetória de referência, a otimização do tempo ou da energia necessária para completar a tarefa de condução autónoma e que todos os objectivos sejam concretizados com segurança.

Uma solução que permite abordar todos estes problemas simultaneamente é a formulação da tarefa de condução autónoma como um problema de otimização com múltiplos objetivos. É então possível utilizar técnicas de controlo ótimo para resolver o problema da condução autónoma. Uma dessas técnicas é *Model Predictive Control*.

Nesta dissertação, propõe-se um controlador para resolver a tarefa de condução autónoma, formulada como um problema de *path-following*, minimizando simultaneamente a energia necessária. Este controlador consiste num anel externo que fornece uma referência de velocidade de energia mínima, e um anel interno que utiliza *Model Predictive Control* para calcular os comandos de direção e aceleração ótimos que conduzem o veículo através do caminho fornecido. Para testar o controlador foi desenvolvido um modelo robusto e complexo do VIENA, um carro elétrico desenvolvido pelo Instituto Superior Técnico.

O controlador é comparado com outros controladores bem estabelecidos, obtendo resultados melhores em vários casos. Estes resultados caracterizam-se por serem sinais de direção suave, uma condição necessária para a experiência de condução segura.

Palavras chave: Condução Autónoma, *Path-Following*, *Model Predictive Control*, Referência de Velocidade de Energia Mínima , Modelo de Veículo, Modelo de Bicicleta

Abstract

Self-driving vehicles have in recent years come to the forefront of engineering studies as a commercially viable solution. There are many objectives associated with autonomous driving, namely minimising the error between the vehicle and the reference provided, optimising either the time or the energy needed to complete the self-driving task, and most important of all, accomplishing these objectives safely.

A solution that tackles all of these problems simultaneously is the formulation of the autonomous driving task as an optimisation problem with multiple, often concurrent, objectives. By doing so, it is possible to use optimal control techniques to solve the autonomous driving problem. One such optimal control technique is Model Predictive Control.

In this dissertation, a controller is proposed to solve the autonomous driving task, formulated as a path-following problem, while minimising the energy needed. This controller consists of an outer loop that provides a minimum energy speed reference profile and an inner loop that uses Model Predictive Control to compute the optimal steering and acceleration commands that lead the vehicle through the provided path. To test the controller, a robust and complex model of VIENA, an eletric car developed by IST was created.

The controller was compared against well-known path-following controllers and it is capable of producing results that are either better or on par with the aforementioned controllers. This was accomplished while producing a smooth steering signal, a relevant contribution to provide a safer driving experience.

Keywords: Autonomous Driving, Path-Following, Optimal Control, Model Predictive Control, Minimum Energy Speed Profile, Vehicle Model, Bicycle Model.

Contents

List of Figures

List of Tables

List of Acronyms

AC	Alternating Current
AGV	Autonomous Ground Vehicle
CG	Centre of Gravity
DARPA	Defense Advanced Research Projects Agency
DC	Direct Current
FOC	Field Oriented Control
GPS	Global Positioning System
MPC	Model Predictive Control
MSTD	Moving Standard Deviation
OCP	Optimisation Control Problem
PD	Proportional Derivative (Controller)
PI	Proportional Integral (Controller)
SAE	Society of Automotive Engineers
VIENA	*Veículo Inteligente Elétrico de Navegação Autónoma*
NHTSA	National Highway Traffic Safety Administration

Chapter 1

Introduction

In 2016, around 1.4 million people were injured and 25600 lost their lives in road traffic accidents in the Member States of the European Union [8]. In 2018 approximately 25100 road fatalities were reported by the Member States [9]. In 2016, a study carried by the United States of America NHTSA found that human error accounts for 94% to 96% of all car accidents [10]. By diminishing human error, autonomous vehicles have the potential to dramatically reduce the number of traffic accidents that occur.

A vehicle's autonomy can range from fully human operated to fully autonomous. The Society of Automotive Engineers (SAE) grades an Autonomous Ground Vehicle (AGV) in five different levels of autonomy [11]. In a summary manner, in level 0 there is no automation present. All actions are performed by the driver. In level 1 there is some level of driving assistance, such as cruise control and electronic stability control. Level 2 introduces partial driving automation modules, such as automatic braking and hazard-minimising longitudinal and lateral control. In level 3 conditional automated driving is achieved, in which the AGV is capable of monitoring the surrounding environment and drive itself with full autonomy under certain conditions but the human operator is still required to take control of the vehicle in case the driving task is outside the scope of the AGV's operational settings. At level 4, high driving automation is present. In this level, the AGV is fully capable of autonomous driving and, in case the driving task is not contemplated by its operational settings, it has fallback modules that allow it to safely control itself if the operator fails to take action. Full driving automation is attained in level 5. A level 5 AGV is unconditionally autonomous, without any expectation that a user will intervene, being able to envelop all driving tasks under its scope. This work's focus is on levels 3 and above.

Driverless vehicles are, at their core, autonomous decision-making systems, who, through the use of sensors, process acquired information about the surrounding environment and use said information along with prior knowledge (about the road network, driving rules, vehicle dynamics and sensor models) to make the optimal driving decision in any given situation. This decision system is hierarchically organised into four components [1]. At its highest level, a route is planned through the road network. Next, a behavioural layer is responsible of selecting a driving task that progresses the car in the way of its objective and abides by the rules of the road. This is followed by a motion planning module, that creates a continuous path through the environment to accomplish the local navigational task. A local feedback control system reactively corrects the errors in the execution of the planned path. Figure 1.1 illustrates the decision making hierarchy of an AGV. This work's focus is on the last decision making level, the local feedback control.

The objective of this work is to develop a local feedback control technique that acts on the vehicle's steering and speed and validate it in a simulation environment. To validate the local feedback control technique, a model of an AGV must be used. The vehicle characterised in the simulation environment is VIENA[1], a car developed by IST, which is the subject of many projects, all of which culminate with the creation of a fully autonomous electric vehicle. Figure 1.2 presents some photographs of VIENA.

[1] `http://viena.tecnico.ulisboa.pt/pt/`

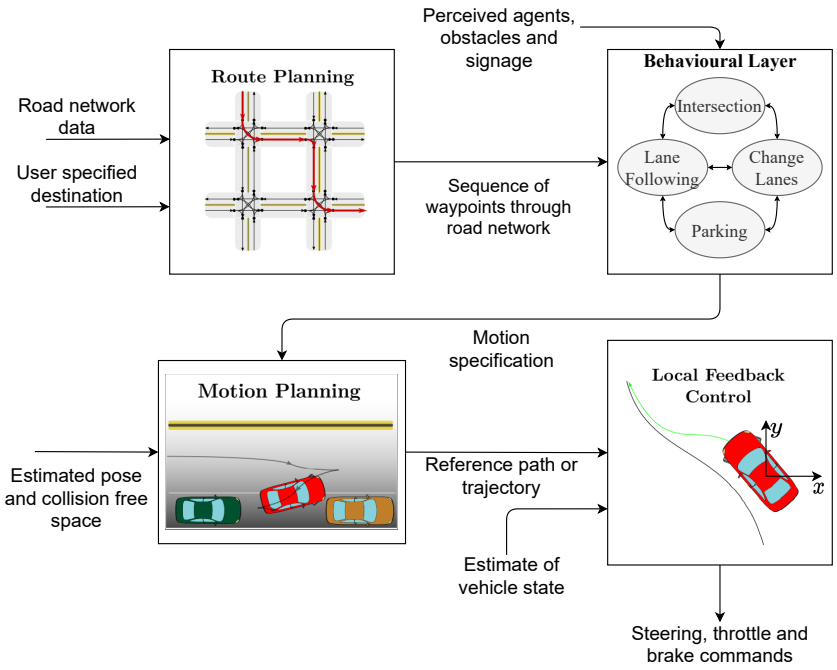


Figure 1.1: The decision making hierarchy of an AGV, taken from [1] and adapted.

(a) (b)

Figure 1.2: VIENA, showcasing its GPS (a) and batteries (b).

1.1 Problem Formulation

The objective of the local control feedback technique is to drive and maintain the considered AGV as close as possible to the reference path (a sequence of points in space) or trajectory (a time indexed sequence of points in space) provided while considering the presence of several forms of uncertainty, namely external disturbances and modelling errors.

In order to do so, it is possible to act both on the AGV's speed and heading, by actuating on its steering system and its motors. Depending on the reference provided (either a path or a trajectory) the problem to be solved is either a path-following problem or a trajectory-tracking problem. Subsections 2.4.1 and 2.4.2 further elaborate on the aforementioned problems.

In this work, a local control technique that acts simultaneously on an AGV's heading and speed is proposed to solve the path-following problem.

1.2 Thesis Structure

The present dissertation is organised in the following manner: Chapter 1 introduces the problem to approach in this thesis, emphasising its motivation. Chapter 2 presents the background as well as some State of the Art solutions that have been proposed to solve the path-following and the trajectory-tracking problems. Chapter 3 presents the steps taken to model the VIENA, the car in which the control technique would be applied. Chapter 4 presents the controller proposed to solve the path-following problem. Chapter 5 provides a detailed account of the proposed technique strengths and weaknesses, giving an overview of the controller's operation and comparing it to well established path-following solutions. Chapter 6 summarises the work performed and highlights the main achievements in this work. Moreover, it proposes further avenues of work to extend the activities described in this document.

Chapter 2

Background and State of the Art

The following sections and subsections detail some of the most pertinent information needed to accomplish the proposed solution. Section 2.1 and the subsections therein detail the various models available to describe AGVs. Section 2.2 details the most relevant control techniques already used to solve the problem. Section 2.3 introduces the problem of speed profile optimisation and presents the possible avenues of optimisation. Section 2.4 presents the formulations of the Path-Following and of the Trajectory-Tracking Problem.

2.1 Vehicle Models

The vehicle model consists of the set of assumptions and equations that reduce the actual physical system into an approximation that faithfully mimics its behaviour. Ideally, the system's model should be as robust as possible, whilst maintaining the level of simplicity that can be afforded. While a detailed model can better represent the system, the level of complexity it carries may complicate without need the control of said system. In order to describe AGVs, several models are available. The following subsections highlight some of the most commonly used models. It is important to preface the following subsections by referring that all the models are subject to the non-holonomic constraint i.e. the vehicles in study are incapable of having lateral displacement without the presence of longitudinal motion.

2.1.1 Unicycle Model

The unicycle model reduces the AGV to a wheel, moving on a plane, with a given velocity, v, and heading, θ. Preferred for its simplicity and inherent contemplation of the non-holonomic constraint, it is often used to model AGVs. Figure 2.1(a) represents the unicycle model.

Kinematic Unicycle Model

As stated in [12], the kinematic unicycle model can be characterised by the following set of equations:

$$\begin{cases} \dot{x} = v\cos(\theta) \\ \dot{y} = v\sin(\theta) \\ \dot{\theta} = \omega \end{cases} \tag{2.1}$$

where (x, y, θ) represent the pose of the AGV in the world reference frame and (v, ω) represent the AGV's linear and angular velocity, the input signals for this model. It is important to note that the kinematic model only contemplates the geometric description of the AGV, disregarding the origin of the input forces and therefore neglecting any inertial effects to which the AGV might be subjected.

Dynamic Unicycle Model

In order to tackle the limitations of the kinematic model, it is possible to derive a dynamic model using Newton's Second Law. As stated in [12] the translational and rotational dynamics of the AGV are given by

$$\begin{cases} M\dot{v} = F - B_v v \\ J\dot{\omega} = T - B_\omega \omega \end{cases} \tag{2.2}$$

where M is the AGV's mass, J its inertial moment, F the sum of the applied forces to the vehicle and B_v and B_ω the translational and rotational friction coefficients, respectively.

By coupling (2.1) and (2.2) through integration blocks it is possible contemplate the dynamics of the AGV in the model.

2.1.2 Bicycle Model

One of the most commonly used models for car-like AGVs is the bicycle model. The vehicle is modelled by two wheels connected through a rigid bar of length L. Steering is introduced by adding to the front wheel an extra degree of freedom, normal to the plane in which the vehicle moves. Figure 2.1(b) represents the kinematic bicycle model.

Kinematic Bicycle Model

By assuming that there is no slippage of the wheels at their contact point with the road it is possible to reduce the bicycle model to a geometric description of the spatial positions of the wheels along time. By considering the linear speed of the rear wheel, v_r, the bicycle model's rear wheel pose is governed (see [1]) by

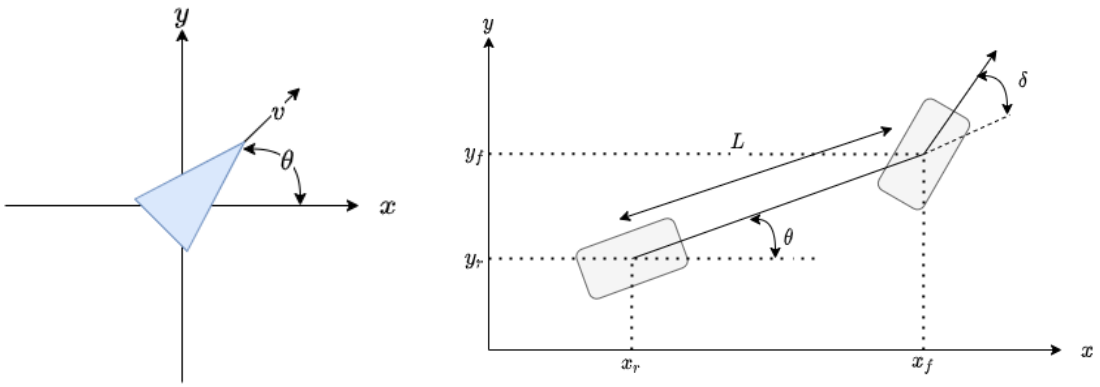
$$\begin{cases} \dot{x}_r = v_r \cos(\theta) \\ \dot{y}_r = v_r \sin(\theta) \\ \dot{\theta} = \frac{v_r}{L} \tan(\delta) \quad . \end{cases} \tag{2.3}$$

where θ is the vehicle's heading angle, δ its steering angle and L is the vehicle's wheelbase. The front, v_f, and the rear wheels speed, v_r, are related by

$$\begin{cases} \dot{x}_f = v_f \cos(\theta + \delta) \\ \dot{y}_f = v_f \sin(\theta + \delta) \\ \dot{\theta} = \frac{v_f}{L} \sin(\delta) \quad . \end{cases} \tag{2.4}$$

The front and rear wheels speed are related by

$$\frac{v_r}{v_f} = \cos(\delta) \quad . \tag{2.5}$$



(a) (b)

Figure 2.1: The Unicycle (a) and Bicycle (b) Kinematic Vehicle Models.

Dynamic Bicycle Model

Kinematic models are suitable for representing AGVs operating at low speeds. However, when the AGV's accelerations take values of higher magnitude, the no slip assumption made for the kinematic models becomes erroneous. When this is the case, a more accurate representation is obtained by modelling the vehicle as a rigid body that satisfies momentum principles. This reduces the vehicle to its centre of mass whose acceleration is proportional to the sum of the forces applied by the ground to the vehicles tires. A derivation of a dynamic bicycle model can be found in [1]. One of the approaches taken to solve the path-following problem is to convert it into a yaw stabilisation problem, where the side-slip angle compensation is adopted to reduce the steady-state errors and then the yaw rate reference is generated for the path-following purpose. In [13] a model that contemplates the presence of roll dynamics is needed in order to tackle this problem.

2.2 Control Methodologies

Several control techniques have been applied to the vehicle control problem. In [1] several vehicle control strategies are presented. The following subsections present some of the most relevant control strategies.

2.2.1 Pure Pursuit Controller

One of the earliest controllers proposed to solve the path-following problem, the pure pursuit controller, fits a semi-circle through the desired point (x_{ref}, y_{ref}) in the reference path and the current position of the car, which is reduced to a point (x_r, y_r), located frequently in the centre of the rear axle of the car. The euclidean distance between the desired path point and the current position of the car is known as the lookahead distance, l_d. Figure 2.2 presents the pure pursuit controller geometry.

The curvature of the circle, κ, is given by

$$\kappa = \frac{2\sin(\alpha)}{l_d}. \tag{2.6}$$

The α angle can be computed directly from a camera output data [1] or, considering an inertial coordinate system,

$$\alpha = \arctan\left(\frac{y_{ref} - y_r}{x_{ref} - x_r}\right) - \theta \tag{2.7}$$

where θ is the vehicle's heading angle. For a vehicle with speed v_r the commanded heading rate is

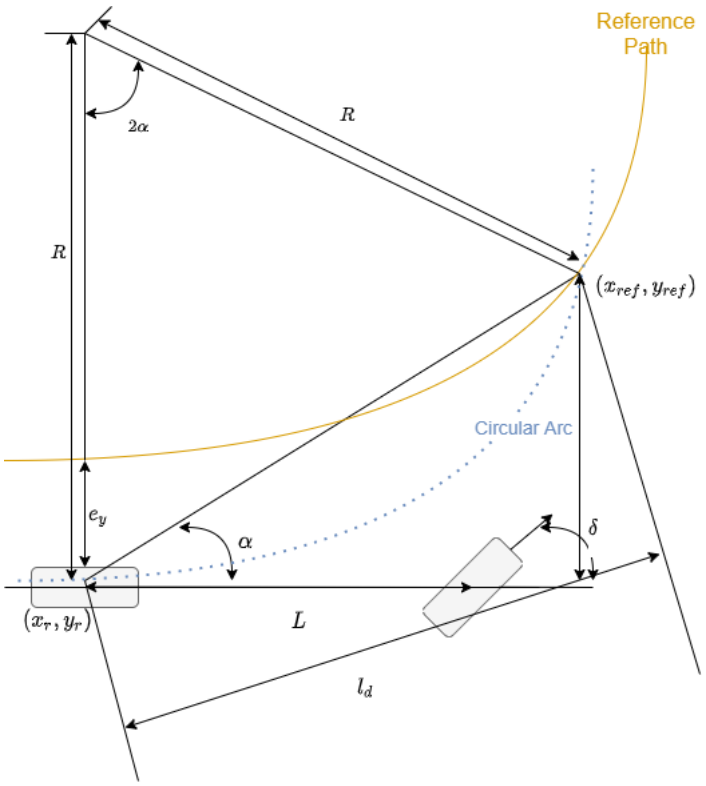


Figure 2.2: Pure Pursuit Controller, adapted from [2].

$$\omega = \frac{2v_r \sin(\alpha)}{l_d}. \tag{2.8}$$

If the desired quantity is steering angle, by considering an Ackerman model with two degrees of freedom, the steering angle is given by

$$\delta = \arctan(\kappa L), \tag{2.9}$$

where L is the vehicle's wheelbase. Combining equations (2.6) and (2.9), the steering angle can be rewritten as

$$\delta = \arctan\left(\frac{2L \sin(\alpha)}{l_d}\right). \tag{2.10}$$

While useful for its simplicity and adequate performance, the pure pursuit controller has some limitations. First, changes in the reference path curvature lead to deviations from the reference trajectory, known as the *cutting corners phenomenon*. The use of a PD controller has been shown to correct these deviations to some extent [2]. Second, the heading rate command becomes increasingly sensitive to the feedback angle α as the vehicle's speed increases. By defining the lookahead distance as a function of the vehicle's speed, this problem can be mitigated [1][2]. Third, the steering command computed through the pure pursuit controller presents a high number of unwanted fluctuations as evidenced in [2] and [14].

2.2.2 Front Wheel Position Based Feedback

Firstly proposed by Stanford University's team to complete the 2005 DARPA Grand challenge [15], this approach solves the path-following by taking the position of the front wheel of a bicycle model as the regulated variable and acts on the steering command to lead the vehicle to the desired pose. Let

$$\dot{e} = v_f \sin(\theta_e + \delta) \tag{2.11}$$

denote the evolution of the transverse error rate, where v_f, θ_e and δ denote the speed of the front wheel, the difference between the tangent at the nearest path point of the path to the front wheel and the car heading and the steering command respectively. The aforementioned error rate can be directly controlled by the steering angle for error rates with magnitude less than v_f [1]. Solving for the steering angle such that $\dot{e} = -ke$ drives $e(t)$ to zero exponentially fast:

$$\begin{aligned} -ke &= v_f \sin(\theta_e + \delta) \Leftrightarrow \\ \Leftrightarrow \delta &= \arcsin\left(\frac{-ke}{v_f}\right) - \theta_e. \end{aligned} \tag{2.12}$$

Please note that in (2.12), k is a gain parameter. This control law however, is not defined when $|ke/v_f| > 1$. To overcome this drawback, the control law becomes

$$\delta = \arctan\left(\frac{-ke}{v_f}\right) - \theta_e. \tag{2.13}$$

This controller has been proven to be locally exponentially stable, stabilising the car to paths with varying curvature with the condition that the path is continuously differentiable [1][15]. A drawback to this controller is that it is not stable for $v_f < 0$, making it unsuitable for manoeuvres that involve reverse driving, such as parking. Figure 2.3 represents the front wheel based feedback control strategy.

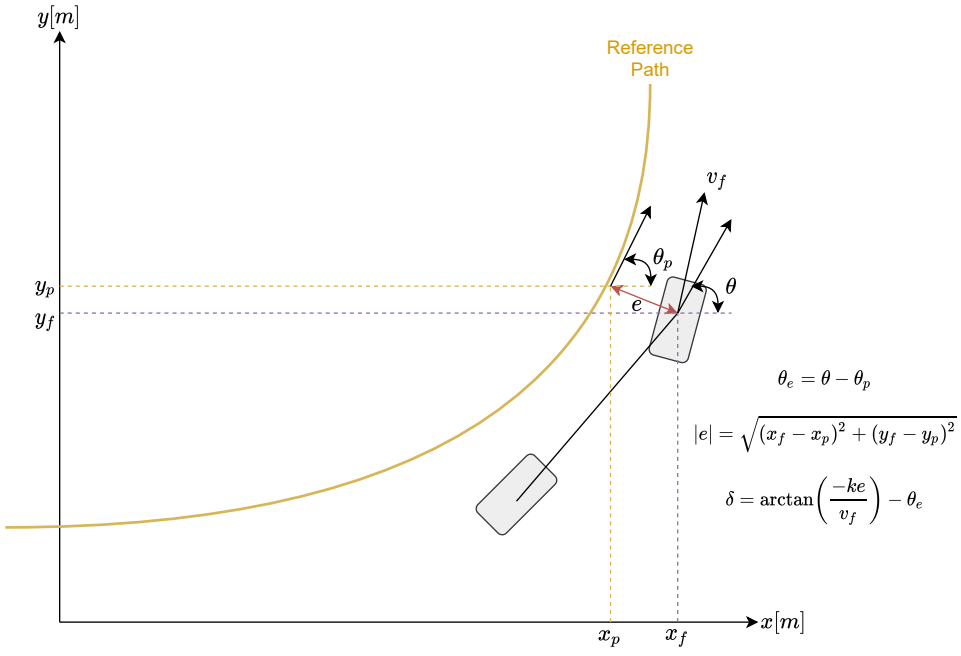


Figure 2.3: Front Wheel Position Based Feedback Control.

2.2.3 Lyapunov Based Control

It is also possible to perform both trajectory-tracking and path-following using a control design based on a control Lyapunov function.

Trajectory-Tracking

Consider an AGV moving along in a 2D plane. Let

$$p(t) = (x(t)\ ,\ y(t),\ \theta(t)) \tag{2.14}$$

denote the pose of the AGV in a inertial coordinate system at time t. Let

$$p_{ref}(t) = (x_{ref}(t),\ y_{ref}(t),\ \theta_{ref}(t)) \tag{2.15}$$

denote reference pose in which the AGV should be located at time t. The reference trajectory is therefore the set of points comprised of the reference poses along the considered time horizon. Thus, it is possible to define the pose error $p_e(t)$ between $p(t)$ and $p_{ref}(t)$ as

$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{ref} - x \\ y_{ref} - y \\ \theta_{ref} - \theta \end{bmatrix}. \tag{2.16}$$

Using both $p_e(t)$ and the reference velocities v_{ref} and ω_{ref}, the evolution of the pose error is given by

$$\begin{cases} \dot{x}_e = \omega y_e - v + v_{ref} cos(\theta_e) \\ \dot{y}_e = -\omega x_e + v_{ref} sin(\theta_e) \\ \dot{\theta}_e = \omega_{ref} - \omega \end{cases} \tag{2.17}$$

where v and ω are respectively the linear and angular speed of the AGV at time t. The proof for (2.17) can be found in Lemma 1 of [16]. With the control assignment (provided in the same paper)

$$\begin{cases} v = v_{ref} cos(\theta_e) + k_x x_e \\ \omega = \omega_{ref} + v_{ref}(k_y y_e k_\theta sin(\theta_e)) \end{cases} \tag{2.18}$$

and the result shown presented in (2.17) the closed loop error dynamics become

$$\begin{cases} \dot{x}_e = (\omega_{ref} + v_{ref})(k_y y_e + k_\theta sin(\theta_e)) y_e - k_x x_e \\ \dot{y}_e = -(\omega_{ref} + v_{ref}(k_y y_e + k_\theta sin(\theta_e)))x_e + v_{ref} sin(\theta_e) \\ \dot{\theta}_e = -v_{ref}(k_y y_e + k_\theta sin(\theta_e)) \quad . \end{cases} \tag{2.19}$$

For $k_{x,y,\theta} > 0$, $\dot{\omega}_{ref}$ = 0 and $\dot{v}_{ref} = 0$, the Lyapunov function

$$V = \frac{1}{2}(x_e^2 + y_e^2) + \frac{1 - cos(\theta_e)}{k_y}, \tag{2.20}$$

with negative semi-definite time derivative

$$\dot{V} = -k_x x_e^2 - \frac{v_{ref} k_\theta sin^2(\theta_e)}{k_y}, \tag{2.21}$$

verifies that, using the control assignment presented in (2.18), $p_e(t) = (0, 0, 0)$ is a state equilibrium point. It can be further shown that if v_{ref} and ω_{ref} are continuous, v_{ref}, ω_{ref}, k_x and k_θ are bounded, and v_{ref} and ω_{ref} are sufficiently small, then $p_e(t)$ is uniformly asymptotically stable over $[0, \infty)$. The proof for both statements can be found in [16].

Path-Following

The path-following problem consists in designing a control law that forces the considered AGV to converge and follow a geometric path. Let l define the difference between the AGV position and the nearest path point and $\tilde{\theta}$ the difference between the AGV's orientation and the tangent at the nearest path point. If l and $\tilde{\theta}$ are driven to zero then the designed control law accomplished its desired objective. Let s denote the signed curvilinear distance along the path. With these set of variables, the system's motion is described in [17] as

$$\begin{cases} \dot{s} = \frac{v\cos(\tilde{\theta})}{1-\kappa(s)l} \\ \dot{l} = v\sin(\tilde{\theta}) \\ \dot{\tilde{\theta}} = \omega - \frac{v\tilde{\theta}\kappa(s)}{1-\kappa(s)l} \end{cases} \tag{2.22}$$

where v and ω represent the AGV's linear and angular speed and $\kappa(s)$ denotes the path's curvature at any given point, assumed to be uniformly bounded and differentiable. By introducing the auxiliary control variable

$$u = \omega - \frac{v\cos(\tilde{\theta})c(s)}{1-\kappa(s)l}, \tag{2.23}$$

it is possible to rewrite (2.22) as

$$\begin{cases} \dot{s} = \frac{v\cos(\tilde{\theta})}{1-\kappa(s)l} \\ \dot{l} = v\sin(\tilde{\theta}) \\ \dot{\tilde{\theta}} = u \quad . \end{cases} \tag{2.24}$$

It is possible to design both a linear and a nonlinear feedback control strategy. Considering first the linear feedback design, the linearization in the neighbourhood of $(l = 0, \tilde{\theta} = 0)$ gives

$$\begin{cases} \dot{l} = v\tilde{\theta} \\ \dot{\tilde{\theta}} = u \quad . \end{cases} \tag{2.25}$$

Considering v as a constant different from zero, the linearized system is controllable and thus asymptotically stable by using linear state feedbacks. The stabilizing linear feedbacks are of the form

$$u = -k_2 vl - k_3|v|\tilde{\theta} \quad with \quad k_{2,3} > 0. \tag{2.26}$$

Instead of linearizing the system, it is possible to consider the nonlinear control

$$u = -k_2 vl\frac{\sin(\tilde{\theta})}{\tilde{\theta}} - k(v)\tilde{\theta}, \tag{2.27}$$

where k_2 is a positive constant and $k(v)$ is a continuous, strictly positive function when $v \neq 0$. Assuming that

$$\lim_{t\to\infty} v(t) \neq 0 \tag{2.28}$$

and that

$$l(0)^2 + \frac{1}{k_2}\tilde{\theta}(0) < \frac{1}{lim_{sup}(\kappa(s)^2)} \quad . \tag{2.29}$$

The Lyapunov function

$$V = k_2\frac{l^2}{2} + \frac{\tilde{\theta}}{2} \tag{2.30}$$

with negative semi-definite time derivative

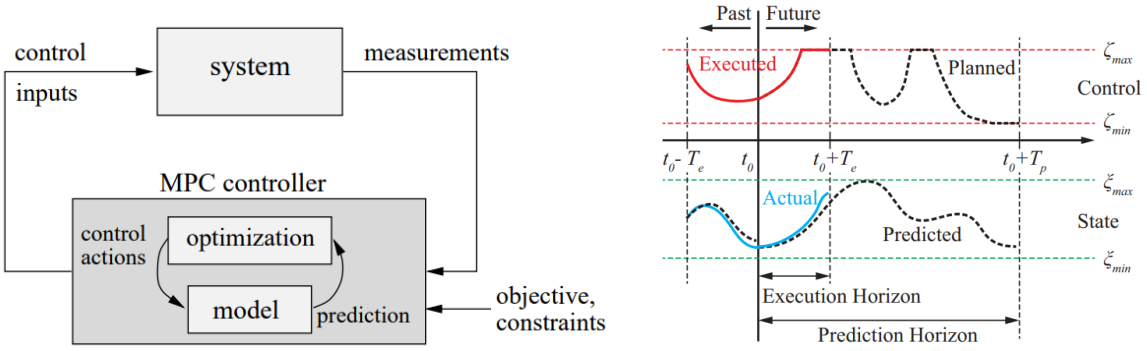
$$\dot{V} = -k(v)\tilde{\theta}^2 \tag{2.31}$$

verifies that the control strategy (2.27) stabilises ($l = 0$, $\tilde{\theta} = 0$). Condition (2.29) is needed to prove that $1 - \kappa(s)l$ remains positive and avoid singularities due to the parameterization [17].

2.2.4 Model Predictive Control

Introduction

Model Predictive Control (MPC), also kown as Receding Horizon Control (RHC), is a control technique that uses a model of the target system to predict and optimise the future system behaviour. It does so by solving an optimisation control problem (OCP) over a short time horizon, known as the prediction horizon, T_p, take a short interval of the computed control signal, known as the execution time, T_e, and apply it to the system. While executing these steps a new OCP is solved so that a suitable control signal for the next time interval is found [1][4]. Figure 2.4(a) shows the basic structure of the MPC controller structure. Figure 2.4(b) illustrates the MPC working principle.



(a) (b)

Figure 2.4: MPC's basic structure (a) and Working Principle (b), taken from [3] and [4] respectively.

As already stated, Model Predictive Control solves an optimisation problem to compute the control signal. Unless the optimisation problem at hand has a known closed-form solution, this supposes the use of a computer and a numerical optimisation algorithm. This, in turn, produces the need to have a discrete formulation of the OCP instead of a continuous one. Consequently, the prediction horizon becomes an integer, N, that translates how many steps ahead the control signal is computed, and the execution time usually becomes the first element of the computed control signal. From hereon the optimisation control problems will be formulated in discrete form.

General OCP Formulation

As in a general optimisation problem, the MPC optimisation control problem is composed of an objective function and its constraints. In the particular case of the OCP, the constraints can divided into two subsets: the state constraints and the input constraints. A generic discrete formulation of an OCP makes use of the following tem-

plate [18]:

$$\begin{aligned}
\underset{U}{\text{minimize}} \quad & p(x_N) + \sum_{k=0}^{N-1} q(x_k, u_k) && && \text{(2.32a)}\\
\text{subject to} \quad & x_{k+1} = Ax_k, Bu_k, && k = 0, \ldots, N-1, && \text{(2.32b)}\\
& x_k \in \mathbb{X}, && k = 0, \ldots, N, && \text{(2.32c)}\\
& x_N \in \mathbb{X}_f, && && \text{(2.32d)}\\
& x_0 = x_{init}, && && \text{(2.32e)}\\
& u_k \in \mathbb{U}, && k = 0, \ldots, N-1 && \text{(2.32f)}
\end{aligned}$$

where (2.32a) is the problem's cost function and equations (2.32b) through (2.32f) are the problem's constraints.

The problem's cost function can be broken down into two distinct costs, the terminal cost, $p(x_N)$, and the stage cost, $q(x_k, u_k)$. The problem's constraints can be broken down into two sets of constraints: the state constraints, denoted by equations (2.32b) through (2.32e), and the input constraints, denoted by equation (2.32f). Equation (2.32b) represents the internal model of the system. By solving the aforementioned OCP the optimal control sequence, U, is obtained.

MPC Feasibility and Stability

The following paragraphs provide necessary and sufficient conditions to guarantee feasibility and stability of an OCP. The definitions, theorems, lemmas, corollaries and remarks, that are reproduced in the next paragraphs, were taken from [18], and are here presented to make the text of the thesis more self-contained.

Feasibility of MPC MPC feasibility is a system (model) and time dependent subject. In particular is challenging the aspect that the feasibility of one (local) OCP does not imply that the following OCP will also be feasible. In other words, feasibility at the initial instant does not imply feasibility for all future instants. It is desirable to design a MPC such that feasibility for all future times is guaranteed, a property commonly known in the literature as *persistent* or *recursive* feasibility [18][19].

To establish some conditions that ensure persistent feasibility, one must first define a body of sets. Let

- $\mathbb{C}_\infty$ denote the maximal control invariant set. It is only affected by sets $\mathbb{X}$ and $\mathbb{U}$, i.e. the state and input constraints. It is the largest set in which any controller is expected to work;
- $\mathbb{X}_0$; the control problem is feasible if $x_0 \in \mathbb{X}_0$. The set $\mathbb{X}_0$ depends on $\mathbb{X}$ and $\mathbb{U}$, on the controller horizon N and on the controller terminal set $\mathbb{X}_f$. It does not depend on the objective function and it has genereally no relation with $\mathbb{C}_\infty$;
- $\mathbb{O}_\infty$ denote the maximal positive invariant set for the closed-loop system. It is dependent on the controller and therefore, on all the parameters that affect the controller i.e. $\mathbb{X}$, $\mathbb{U}$, N, $\mathbb{X}_f$ and the objective function, with all the parameters that may pertain to it. Clearly, $\mathbb{O}_\infty \subseteq \mathbb{X}_0$ since if that was not the case, there would be points in $\mathbb{O}_\infty$ for which the control problem would not be feasible. It should also be clear that $\mathbb{O}_\infty \subseteq \mathbb{C}_\infty$. Because of invariance, the closed-loop is persistently feasible for all states $x_0 \in \mathbb{O}_\infty$.

With these sets defined, it is now possible to state necessary and sufficient conditions that guarantee persistent feasibility by means of invariant set theory. Please note that all these conditions and their proofs can be found in [18].

Lemma 2.2.1. *Let $\mathbb{O}_\infty$ be the maximal invariant set for the closed-loop system $x_{k+1} = f_{cl}(x_k)$ for OCP* (2.32a)-(2.32f). *The MPC problem is persistently feasible if and only if $\mathbb{X}_0 = \mathbb{O}_\infty$*

Lemma 2.2.2. *Consider the MPC law* (2.32a)-(2.32f) *with* $N \geq 1$. *If* $\mathbb{X}_1$ *is a control invariant set, then the MPC is persistently feasible.*

Theorem 2.2.3. *Consider the MPC law* (2.32a)-(2.32f) *with* $N \geq 1$, *If* $\mathbb{X}_f$ *is a control invariant set for the system at hand, then the MPC is persistently feasible*

Corollary 2.2.3.1. *Consider the MPC law* (2.32a)-(2.32f) *with* $N \geq 1$. *If there exists* $i \in [1, N]$ *such that* $\mathbb{X}_i$ *is a control invariant set for the considered system, then the MPC is persistently feasible for all cost functions.*

Please note that persistent feasibility does not guarantee that the closed-loop trajectories converge towards the desired equilibrium points. One of the most popular approaches to guarantee persistent feasibility and stability of the MPC law makes use of a control invariant terminal set $\mathbb{X}_f$ and a terminal cost, P, which drives the closed-loop optimal trajectories towards the origin.

Stability of MPC This paragraph presents the main stability result for MPC. The objective is to find a Lyapunov function for the closed-loop system. If the terminal cost and constraint are properly chosen, then the cost function is a Lyapunov function.

Theorem 2.2.4. *Consider the given system, and the MPC law* (2.32a)-(2.32f). *If*

1. *The stage cost* $q(x_k, u_k)$ *and terminal cost* $p(x_N)$ *are continuous and positive definite functions;*
2. *The sets* $\mathbb{X}$, $\mathbb{X}_f$ *and* $\mathbb{U}$ *contain the origin in their interior and are closed;*
3. $\mathbb{X}_f$ *is a control invariant set, i.e.* $\mathbb{X}_f \in \mathbb{C}_\infty$*;*
4. $\min\limits_{v \in \mathbb{U},\ Ax+Bv \in \mathbb{X}_f} \quad -p(x) + q(x, v) + p(Ax + Bv) \leq 0, \quad \forall x \in \mathbb{X}_f$*;*

Then the origin of the closed-loop system is asymptotically stable, with domain of attraction $\mathbb{X}_0$.

There is a series of remarks that should me made concerning Theorem 2.2.4.

Remark 2.2.5. *The assumption on the positive definiteness of the stage cost in Theorem 2.2.4 can be relaxed as in standard optimal control.*

Remark 2.2.6. *The procedure outlined in Theorem 2.2.4 is, in general, conservative because it requires the introduction of an artificial terminal set* $\mathbb{X}_f$ *to guarantee persistent feasibility and a terminal cost to guarantee stability. Requiring* $x_N \in \mathbb{X}_f$ *usually decreases the size of the region of attraction* $\mathbb{X}_0 = \mathbb{O}_\infty$. *Also the performance may be negatively affected.*

Remark 2.2.7. *A function* $p(x)$ *satisfying assumption condition 4 of Theorem 2.2.4 is often called a control Lyapunov function.*

Condition 3 of Theorem 2.2.4 is required for guaranteeing persistent feasibility. Is is typically required that the horizon N is sufficiently large to ensure feasibility of MPC (2.32a)-(2.32f) at all instants.

MPC Advantages and Disadvantages

MPC is capable of both solving path-following as well as trajectory-tracking problems. MPC is of particular interest due to its extensibility since it is capable of controlling nonlinear, time-varying systems. This can be exploited by considering more complex models that can take into account such phenomena as tire behaviour [14][4] or longitudinal load transfers [4].

Several simulated results have been achieved using MPC. In [4] the authors successfully developed a steering and speed controller capable of manoeuvring a high CG AGV through a static obstacle course, with no prior

knowledge of the location or shape of said obstacles. This was further developed in [20] where moving obstacles were taking into consideration. Both publications document findings that improve upon the results that served as benchmarks. In [14] a combination of MPC and fuzzy control is utilised to ensure both tracking accuracy as well as steering smoothness.

The main challenges associated with MPC are twofold: first, it is a time consuming, computationally heavy controller. The need to continuously solve an OCP in every loop of the controller difficults a practical implementation in a autonomous vehicle, whose response has to be quick in order to ensure passenger/cargo safety. Second, the careful definition of OCP (2.32a) - (2.32f) is essential to ensure the correct computation of U as well as a computationally efficient solution. In [20] an example is given between two valid definitions of (2.32a) - (2.32f), one of them finding the OCP solution in 45 seconds, and the other finding the solution in 1 minute and 50 seconds, approximately two and a half times faster.

2.3 Speed Profile Optimisation

As well as providing a reference path, there is also the need of providing a reference speed along the path that the vehicle needs to follow. This is turn, poses the possibility of computing an optimal speed profile.

In the scope of speed profile computation, the avenues of optimisation are threefold. One can either compute the speed profile that minimises the time needed to complete a reference path, the speed profile that minimises the energy needed to complete a given reference path, or a speed profile that concurrently attempts to solve both objectives, i.e. finding the minimum energy speed profile that completes the given reference in a given time frame.

The following subsections will present various solutions proposed to optimise the speed profile with the different aforementioned optimisation objectives as well as presenting the different assumptions made by each of them and their respective limitations.

2.3.1 Minimum Time Speed Profile Optimisation

The minimum time speed profile optimisation problem consists on computing the speed profile that leads the vehicle to track through the provided reference path on the least possible time. This, in general, implies disregard for the energy needed to complete the path, save for the need to guarantee that the energy needed to accomplish task does not exceed the maximum energy available in the vehicle's energy storage system. For the minimum time speed profile optimisation problem, two different approaches were studied, one that computes the speed profile with the support of the equations of linear motion, and one that makes use of Pontryagin's Maximum Principle.

Speed Profile Optimisation using the equations of Linear Motion

This method is perhaps the most straightforward strategy that analytically solves the minimum time speed profile optimisation problem. It considers the vehicle accelerates, with the maximum acceleration that it can produce, up to its maximum speed and maintains that speed until the point where it decelerates, with the maximum deceleration that it can produce, until it stops. Therefore, this particular speed profile can be broken down into three distinct regions: a region of constant acceleration, a region of constant speed and a region of constant deceleration.

This speed profile is computed over the reference path's length and disregards its characteristics, such as inclination and curvature. Because of this, it is possible to make use of the equations of linear motion to determine the points at which the vehicle stops accelerating and starts decelerating.

Region of constant acceleration Considering first the region of constant acceleration, let

$$a_1(t) = a_{max} \tag{2.33}$$

denote the acceleration of the vehicle during the acceleration stage of the speed profile. The vehicle's speed during this stage can be expressed as

$$v_1(t) = \int_0^t a_1(t)dt \Leftrightarrow v_1(t) = a_{max}t + v_1^0 \tag{2.34}$$

where v_1^0 denotes the vehicle's initial speed in the region of maximum acceleration. As the vehicle begins from a standstill,

$$v_1^0 = 0 \tag{2.35}$$

and

$$v_1(t) = a_{max}t. \tag{2.36}$$

Let v_{max} denote the vehicle's known maximum speed. The time range, Δt_1, required for the vehicle to achieve the maximum speed is

$$\Delta t_1 = \frac{v_{max}}{a_{max}}. \tag{2.37}$$

The speed profile however, must be a function of the reference path's length and therefore there is the need to find the path point where the vehicle is after Δt_1 has passed. Let

$$x_1(t) = \int_0^t v_1(t)dt \Leftrightarrow x_1(t) = \frac{1}{2}a_{max}t^2 + v_1^0 t + x_1^0 \tag{2.38}$$

denote the vehicle's position along the reference path's length during the acceleration region. It has already been assumed that $v_1^0 = 0$ and since the vehicle is at the path's beginning,

$$x_1^0 = 0. \tag{2.39}$$

Therefore, after Δt_1, the vehicle's position is

$$x_1^\star = \frac{1}{2}a_{max}\Delta t_1^2. \tag{2.40}$$

Region of constant speed In this region, it is assumed that the car has reached its peak velocity at $x_1^\star$ and will maintain said speed up until the point where it starts to decelerate, henceforth denoted as $x_2^\star$. Since speed in this region is constant, the acceleration is of course null

$$a_2(t) = 0, \tag{2.41}$$

and the speed is given by

$$v_2(t) = \int_0^t a_2(t) \Leftrightarrow v_2(t) = v_2^0 = v_{max}. \tag{2.42}$$

The distance travelled in the region of constant speed is given by

$$x_2(t) = \int_0^t v_2(t) \Leftrightarrow x_2(t) = v_{max}t + x_2^0 \Leftrightarrow x_2(t) = v_{max}t + x_1^\star \tag{2.43}$$

and the time spent in this region is given by

$$\Delta t_2 = \frac{x_2^\star - x_1^\star}{v_{max}}. \tag{2.44}$$

In sum, for the region of constant speed

$$v_2(t) = v_{max},\ v_2(t) \in [x_1^\star,\ x_2^\star] \tag{2.45}$$

Region of constant deceleration In the region of constant deceleration, the vehicles decelerates from its maximum speed to a standstill. Let

$$a_3(t) = d_{max} < 0 \tag{2.46}$$

denote the acceleration of the vehicle during the deceleration stage. The speed, $v_3(t)$ in this region is therefore

$$v_3(t) = \int_0^t a_3(t) \Leftrightarrow v_3(t) = d_{max}t + v_3^0 \tag{2.47}$$

where v_3^0 denotes the vehicle's initial speed during the deceleration stage. From (2.45) one gathers that

$$v_3^0 = v_{max} \tag{2.48}$$

and therefore

$$v_3(t) = d_{max}t + v_{max}. \tag{2.49}$$

It is now possible to compute the time it takes to decelerate from v_{max} to 0,

$$\Delta t_3 = -\frac{v_{max}}{d_{max}}. \tag{2.50}$$

To find $x_2^\star$, one must first define the movement of the vehicle during the deceleration stage:

$$x_3(t) = \int_0^t v_3(t) \Leftrightarrow x_3(t) = \frac{1}{2}d_{max}t^2 + v_{max}t + x_3^0. \tag{2.51}$$

Please note that

$$x_2^\star = x_3^0 \tag{2.52}$$

and that to find the reference path's position where the vehicle needs to start braking, $x_2^\star$, it is important to keep in mind that at the final time. t_{final},

$$x_3(t_{final}) = L \tag{2.53}$$

where L denotes the total path's length. Thus,

$$x_2^\star = L - \frac{1}{2}d_{max}\Delta t_3 - v_{max}\Delta t_3. \tag{2.54}$$

Figure 2.5 presents the aspect of a speed profile computed using the method presented above.

Speed Profile Optimisation using Prontryagin's Maximum Principle

In [21], a semi-analytical method to generate minimum-time optimal speed profiles for a vehicle with given acceleration limits is proposed. A point mass model of the vehicle is used and the problem in formulated using an optimal control framework, namely Pontryagin's Maximum Principle. Several cases in which loss of controllability occurs were addressed. The methodology proposed has minimal computational cost when compared to common numerical optimisation approaches and is suitable for real time implementations.

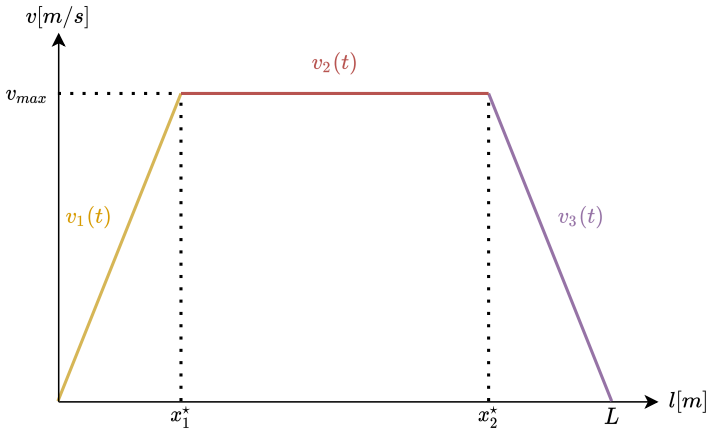


Figure 2.5: Speed profile computed using the equations of linear motion.

2.3.2 Minimum Energy Speed Profile Optimisation

Instead of generating a minimum time optimal speed profile, one could change the optimisation target to be the energy needed to track through the given path.

Various techniques to obtain globally energy efficient driving profiles have been proposed [22] [23], involving optimisation algorithms such as dynamic programming or sequential quadratic programming. These methods, however, are difficult to implement on real vehicles, due to their significant computational requirements and the need for precise a-priori knowledge of the scenario ahead.

In [24], an optimal solution is provided using Dynamic Programming, comparing the computed speed profile against simple, intuitive speed profiles.

2.4 The Path-Following and Trajectory-Tracking Problems

The following sections introduce the path-following and trajectory-tracking problems and specify what type of problem is to be tackled in this dissertation.

2.4.1 The Path-Following Problem

In path-following, the objective is to steer the vehicle along a given reference geometric path that is free of temporal parameterization. Figure 2.6(a) presents the path-following problem in a planar setting.

In Figure 2.6(a), P represents the AGV's position and M is the orthogonal projection of P on the path, (M, η_T, η_N) is a Serret-Frenet frame moving along the path, η_T is the vector tangent to the path in the closest point to the vehicle and η_N is the vector normal in said point. l represents the distance between M and P, s is the signed curvilienar distance along the path, from some initial point to point M, θ_d is the angle between the x-axis and the tangent to the path at point M, $\kappa(s)$ is the path's curvature at point M, assumed uniformly bounded and differentiable and $\tilde{\theta} = \theta - \theta_d$ is the orientation error [17].

With the aforementioned variables and considering $(s, l, \tilde{\theta})$ as the set of state coordinates, it is possible to define the path-following problem. Problem 1 presents the path-following problem, as stated in [17].

Problem 1. *Given a path in the x-y plane and the AGV's translational velocity $v(t)$, assumed to be bounded along with its time derivative $\dot{v}(t)$, find a smooth feedback control law*

$$\omega = k(s, l, \tilde{\theta}, v(t)) \tag{2.55}$$

such that

$$\begin{cases} \lim_{t\to\infty} l(t) = 0 \\ \lim_{t\to\infty} \tilde{\theta}(t) = 0 \end{cases} . \tag{2.56}$$

The path-following problem can be solved by employing any of the techniques discussed in Section 2.2: Pure Pursuit, Front Wheel Based Feedback Control, Lyapunov based control and MPC.

2.4.2 The Trajectory-Tracking Problem

The trajectory-tracking problem for an AGV is formulated with the introduction of a virtual reference vehicle to be tracked [17]. The trajectory-tracking problem consists in driving the AGV to track a series of time parameterized reference positions [12]. Figure 2.6(b) represents the trajectory-tracking problem in a planar setting. In Figure 2.6(b), $(x(t), y(t), \theta(t))$ and $(x_{ref}(t), y_{ref}(t), \theta_{ref}(t))$ represent, respectively, the current and desired AGV positions at time t. $v(t)$ and $v_{ref}(t)$ represent the current and desired linear velocity at time t, respectively. Problem 2 presents the trajectory-tracking problem, as stated in [17].

Problem 2. *Let*

$$\begin{cases} \dot{x}_{ref} = v_{ref}\cos(\theta_{ref}) \\ \dot{y}_{ref} = v_{ref}\sin(\theta_{ref}) \\ \dot{\theta}_{ref} = \omega_{ref} \end{cases} \tag{2.57}$$

denote the equations that model the movement of a reference unicycle type AGV, whose linear and angular velocities are bounded, along with their time derivatives. While assuming that

$$\begin{cases} \lim_{t\to\infty} v_{ref}(t) \neq 0 \\ \lim_{t\to\infty} \omega_{ref}(t) \neq 0 \end{cases} \tag{2.58}$$

find a feedback control law

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = k(z, z_{ref}, v_{ref}, \omega_{ref}) \tag{2.59}$$

such that

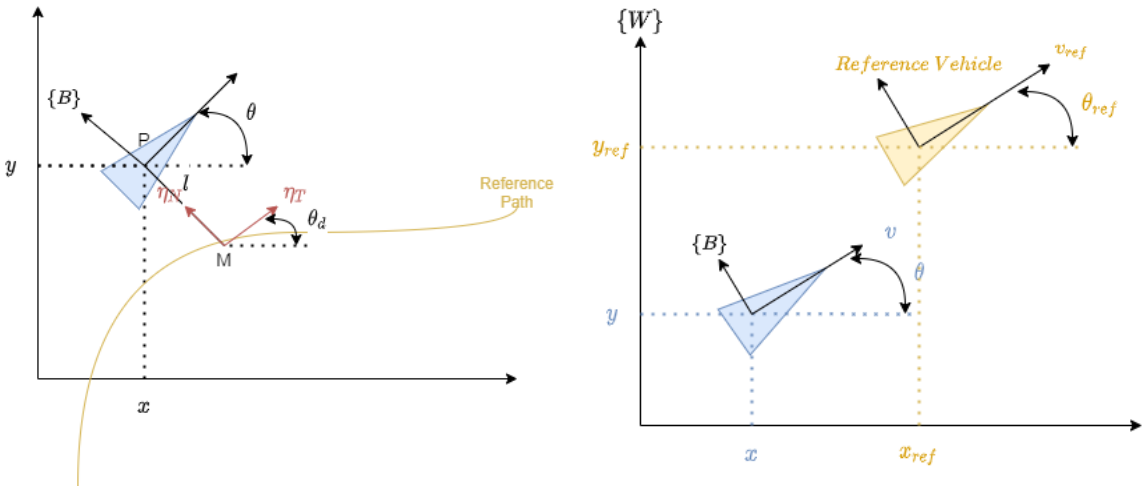
$$\lim_{t\to\infty} z(t) - z_{ref}(t) = 0 \tag{2.60}$$

where $z(t)$ and $z_{ref}(t)$ represent the current and desired poses at time t. Please note that (2.58) *implies that the reference trajectory is not at rest all the time. Hence, pose stabilization is outside the scope of the trajectory tracking problem.*

Of the control techniques shown in Section 2.2, Lyapunov based control and MPC have been used to solve the trajectory-tracking problem.

Because the path-following problem is free of time parameterization, it is less stringent than its trajectory-tracking counterpart. Because of this, the problem at hand is formulated as a Path-Following Problem. The main difficulty in path-following lies in computing a reference path that abides by the rules presented in the Path-Following Problem subsection. This is further discussed in Chapter 4.

The two following chapters present the proposed solution to solve the Path-Following Problem. Chapter 3 details the steps and assumptions made to simulate VIENA's behaviour. Chapter 4 presents the proposed control technique to solve the Path-Following Problem.



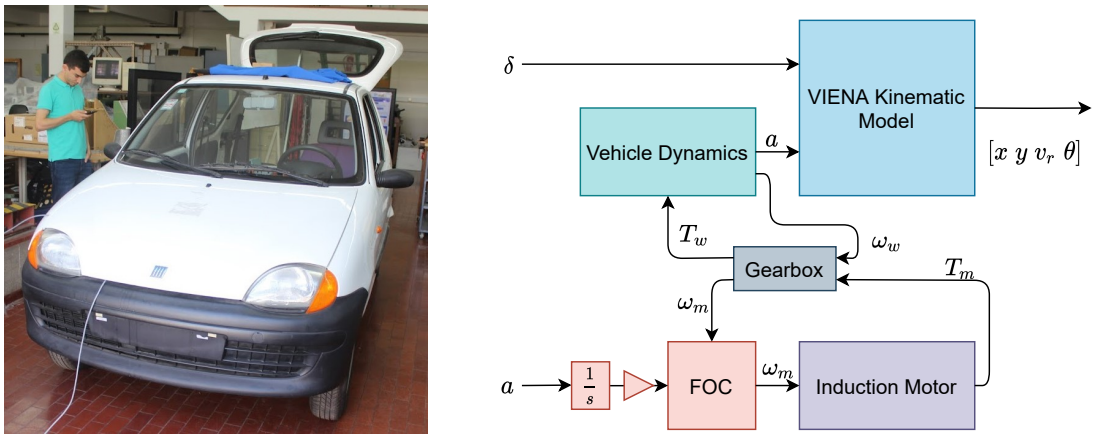
(a) (b)

Figure 2.6: The Path-Following (a) and Trajectory-Tracking (b) Problems, adapted from [5].

Chapter 3

VIENA Car Model

This chapter introduces the equations that characterise the Intelligent Electirc Vehicle with Autonomous Navigation [1]. Section 3.1 presents equations that govern VIENA's movement, also known as its kinematic model. Section 3.2 presents all the equations that model VIENA's dynamics, from the forces and frictions that produce its acceleration, in subsection 3.2.1, to the equations that model its induction motor, and the Field Oriented Control that is used to drive the motor's rotor to its desired velocity, presented in subsection 3.2.3. Figure 3.1 shows a picture of VIENA as well as a block diagram of VIENA's model.



(a) (b)

Figure 3.1: VIENA (a), and its abstraction as a Block Diagram (b).

3.1 VIENA Kinematic Model

In [6] is introduced one model for VIENA based on a simple 2D body frame dynamic model with Ackermann steering. Figure 3.2 represents the vehicle's model, along with the location of the sensors used and the dimensions of VIENA. In this model, is assumed the vehicle is subject to the non-holonomic constraint. Rotation of the vehicle occurs only around the z-axis. Other rotations are the consequence of suspension movement and road unevenness. The vehicle Euler angle θ (rotation along the y-axis) at the wheel axis level is close to zero and it will be considered zero. There is neither lateral nor longitudinal slip, the tires are capable of sustaining the lateral forces generated during dynamics and they do not loose contact to the ground.

Using the assumptions above and a simplified bicycle model, the equations that govern this model are

[1] http://viena.tecnico.ulisboa.pt

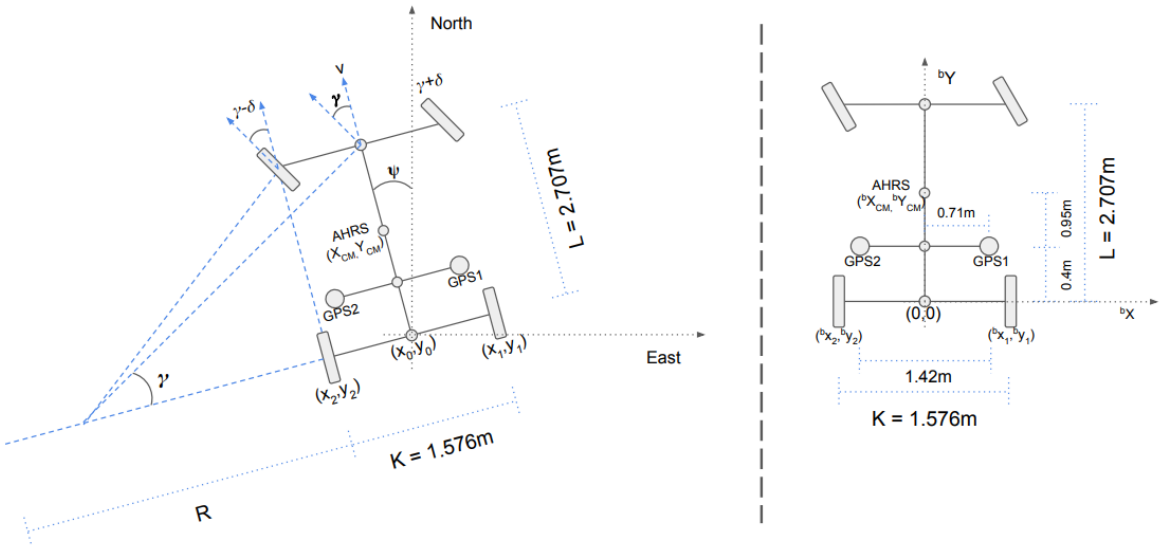


Figure 3.2: Vehicle model used to represent VIENA, taken from [6].

$$\begin{cases} {}^{w}\dot{X}_0 = -{}^{b}V_y \sin(\psi)\cos(\phi) \\ {}^{w}\dot{Y}_0 = {}^{b}V_y \cos(\psi)\cos(\phi) \\ \dot{\psi} = \frac{{}^{b}V_y}{L}\tan(\gamma)\cos(\phi) \\ {}^{b}\dot{V}_y = {}^{b}A_y \end{cases} \tag{3.1}$$

In this model, the rate of ψ must be estimated through the propagation of attitude due to angular speed. Note that ϕ represents the angle between the road and the car, i.e. the pitch.

In order to maintain consistency of the reference frames and notations of the VIENA kinematic model with the vehicle models presented in the Chapter 2, as well as the reference literature, a change of notation and reference frame in (3.1) must be performed. The state vector $[{}^{w}X_0 \ {}^{w}Y_0 \ \psi \ {}^{b}V_y]^T$ is rewritten with the notation $[x \ y \ \theta \ v_r]^T$, the steer command γ becomes δ and the last two equations are reordered. The VIENA kinematic model utilised henceforth is given by the following set of equations:

$$\begin{cases} \dot{x} = v_r \cos(\theta)\cos(\phi) \\ \dot{y} = v_r \sin(\theta)\cos(\phi) \\ \dot{v}_r = a \\ \dot{\theta} = \frac{v_r}{L}\tan(\delta)\cos(\phi) \quad . \end{cases} \tag{3.2}$$

3.2 VIENA Dynamic Model

The model of VIENA's behaviour is composed of a model of an induction motor, controlled via Field Oriented Control (FOC), a gearbox, the dynamics the vehicle is subject to and its kinematic model. The next subsections provide further insight into the remainder of these components.

3.2.1 Vehicle Dynamics

VIENA's vehicle dynamics take as input the vehicle's wheel torque and determines the vehicle's angular wheel speed as well as its linear speed. To characterise VIENA's vehicle dynamics, one makes use of Newton's second

law of motion:

$$\sum F = Ma. \tag{3.3}$$

The forces applied to the vehicle are the traction torque, T_t, applied in the rear wheels, the component of the force created by the vehicle's own weight in line with the vehicle's movement, F'_g, the friction force applied by the ground on the vehicle's wheel, F_r, and the aerodynamic drag force applied whilst the vehicle is moving, F_a. The vehicle's total mass, M, is divided into the vehicle's mass, m, and the equivalent mass of the rotating parts, m_{eq}^{rot}. Therefore, (3.3) can be rewritten as

$$\frac{T_t}{r_w} + F_g + F_r + F_a = (m + m_{eq}^{rot})a \tag{3.4}$$

where

$$F'_g = -mg\sin(\phi) \tag{3.5}$$

denotes the force created by the gravitational pull, with g denoting earth's acceleration and ϕ the path's inclination,

$$F_r = -C_{rr}mg\cos(\phi) \tag{3.6}$$

denotes the rolling resistances force, with C_{rr} denoting the rolling resistance coefficient,

$$F_a = -\frac{1}{2}\rho C_d A_f (v_r - (-v_w))^2 \tag{3.7}$$

denotes the aerodynamic frag force, with ρ denoting the air's density, C_d the drag coefficient, A_f the frontal area of the vehicle v_r the vehicle's rear wheel speed and v_w the wind's speed opposite to the vehicle movement and

$$m_{eq}^{rot} = \frac{1}{r_w^2}\left(I_w + I_m g_r^2\right) \tag{3.8}$$

denotes the equivalent mass of the rotating parts, with I_w and I_m denoting the wheel's and the motor's rotor inertia, respectively and g_r denotes the gear ratio of the gearbox between the motor and the wheels.

Please note that the forces described by equations (3.5) through (3.7) have associated with them a minus sign because they are contrary to the vehicle's movement. The vehicle's rear wheel speed is given by

$$v_r(t) = \int_{t_1}^{t_2} a(t)dt \tag{3.9}$$

and the wheel's angular speed is given by

$$\omega_w(t) = \frac{v_r(t)}{r_w}. \tag{3.10}$$

3.2.2 Gearbox Subsystem

The adopted gearbox model simply uses the gear ratio to convert the wheel's speed into the motor's speed and the motor's torque into the wheel's torque, using the following equalities:

$$\begin{cases} \omega_m = g_r \omega_w, \\ T_w = g_r T_m, \end{cases} \tag{3.11}$$

where ω_m and T_m denotes the motor's angular speed and torque respectively.

3.2.3 Induction Motor and Field Oriented Control

To further characterise VIENA's behaviour, the induction motor that receives the desired reference speed and outputs the needed torque to achieve the desired speed was modelled and integrated into the overall system. The presence of the dynamics of the induction motor implies the need to control it. Field Oriented Control (FOC) is used to drive the rotor speed to its desired value. The use of FOC supposes the characterisation of the induction motor behaviour in the direct-quadrature-zero, *dq0*, model. The following sections present the *dq0* model as well as the FOC structure, working principle and the equations that govern it.

The Induction Motor Dynamic Model

Let Figure 3.3 represent a simplified model of the induction machine, assuming that the rotor is composed of three-phases, there is symmetry between the rotor and the stator and there is no neutral connection, i.e. the homopolar currents are null.

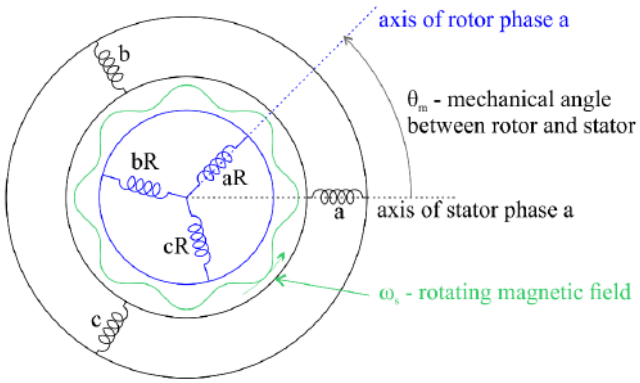


Figure 3.3: Dynamic Model of the Induction Motor, taken from [7].

The induction motor's stator and rotor voltages are governed by the induction law applied to the stator and rotor windings

$$\begin{cases} v_{abc_s} = R_s i_{abc_s} + \dot{\lambda}_{abc_s} \\ v_{abc_r} = R_r i_{abc_r} + \dot{\lambda}_{abc_r} \end{cases} \tag{3.12}$$

where v_{abs_s} and v_{abc_r} respectively denote the stator and rotor three-phase voltages, i_{abc_s} and i_{abc_r} denote the stator and rotor three-phase currents, R_s and R_r the stator and rotor coil resistances and λ_{abc_s} and λ_{abc_r} the stator and rotor linkage fluxes.

Assuming the magnetic materials considered are homogeneous and isotropic, no situation of saturation occurs, and the rotor has a cylindrical shape, the linkage fluxes can be described as a linear combination of the stator and rotor currents:

$$\begin{cases} \lambda_{abc_s} = L_s i_{abc_s} + L_m i_{abc_r} \\ \lambda_{abc_r} = L_r i_{abc_r} + L_m i_{abc_s} \end{cases} \tag{3.13}$$

where L_s and L_r respectively denote the self-inductances of the stator and the rotor and L_m denotes the mutual inductance coefficient between the stator and rotor, defined as

$$\begin{cases} L_s = L_m + l_{\sigma_s} \\ L_r = L_m + l_{\sigma_r} \end{cases} \tag{3.14}$$

where l_{σ_s} and l_{σ_r} respectively denote the stator and rotor leakage inductances, assumed to be constant.

The *dq0* Induction Motor Model

The *dq0* model allows to perform a transportation of three-phase variables present in the *abc* model of the induction motor to a two-phase variable system, *dq*, as the homopolar quantities are null. Furthermore, to simplify the complexity of the motor model, the *dq* axis can be defined to be rotating with the rotating magnetic field. Please note that, in the three-phase *abc* system, sinusoidal voltages are applied to each three-phase axis, each one lagged by 120º, to create the rotating magnetic field required to create motion in the rotor. However, the same rotating magnetic field can be originated by creating fictitious *dq* windings, physically rotating with the rotating magnetic field with DC voltages. This is only a fictitious model but helps simplifying the motor model, from 3-phase to 2-phase system and from AC to DC quantities. In Figure 3.4 it is presented the *dq* model of the motor based on phase A of the three-phase system.

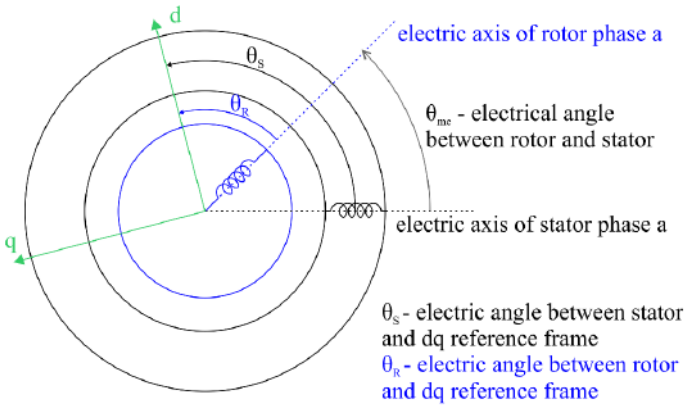


Figure 3.4: Electric angle relationships between the stator, rotor and the *dq0* reference frame, taken from [7].

From Figure 3.4, one obtains

$$\theta_s = \theta_r + \theta_{me} \tag{3.15}$$

where θ_s denotes the electrical angle between the stator and the d-axis, θ_r denotes the electrical angle between the rotor and the d-axis and θ_{me} denotes the electrical angle between the stator and the rotor. This angle is given by

$$\theta_{me} = \theta_m n_{pp} \tag{3.16}$$

where θ_m denotes the rotor mechanical angle, and n_{pp} denotes the number of pole pairs present in the considered induction machine.

The transformation between the three-phase system and the dq-system can be achieved by applying the Park transformation matrix $P(\theta)$, where $\theta = \theta_s$ for the stator quantities and $\theta = \theta_r$ for the rotor quantities:

$$P(\theta) = \frac{2}{3}\begin{bmatrix} \cos(\theta) & \cos\left(\theta - \frac{2\pi}{3}\right) & \cos\left(\theta + \frac{2\pi}{3}\right) \\ -\sin(\theta) & -\sin\left(\theta - \frac{2\pi}{3}\right) & -\sin\left(\theta + \frac{2\pi}{3}\right) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix}. \tag{3.17}$$

Using the Park transformation matrix, it is possible to write the stator and rotor voltages, currents and fluxes in the *dq0* reference frame:

$$v_{dq0_{s,r}} = P(\theta_{s,r}) v_{abc_{s,r}} \tag{3.18}$$

$$i_{dq0_{s,r}} = P(\theta_{s,r}) i_{abc_{s,r}} \tag{3.19}$$

$$\lambda_{dq0_{s,r}} = P(\theta_{s,r}) \lambda_{abc_{s,r}}. \tag{3.20}$$

Combining (3.12), (3.13) and (3.18) one gets

$$\begin{cases} P(\theta_s)^{-1} v_{dq0_s} = R_s P(\theta_s)^{-1} i_{dq0_s} + \frac{d}{dt}\left(P(\theta_s)^{-1} \lambda_{dq0_s}\right) \\ P(\theta_r)^{-1} v_{dq0_r} = R_r P(\theta_r)^{-1} i_{dq0_r} + \frac{d}{dt}\left(P(\theta_r)^{-1} \lambda_{dq0_r}\right). \end{cases} \tag{3.21}$$

Knowing that

$$\frac{d}{dt}\left(P(\theta_{s,r})^{-1} \lambda_{dq0_{s,r}}\right) = \frac{d}{dt}\left(P(\theta_{s,r})^{-1}\right) \lambda_{dq0_{s,r}} + P(\theta_{s,r})^{-1} \frac{d}{dt} \lambda_{dq0} \tag{3.22}$$

and that

$$W(\dot{\theta}) = \frac{d}{dt}\left(P(\theta)^{-1}\right) = P(\theta)^{-1} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \dot{\theta}. \tag{3.23}$$

it is possible to rewrite (3.21) as

$$\begin{cases} v_{dq0_s} = R_s i_{dq0_s} + \dot{\lambda}_{dq0_s} + W(\dot{\theta}_s) \lambda_{dq0_s} \\ v_{dq0_r} = R_r i_{dq0_r} + \dot{\lambda}_{dq0_r} + W(\dot{\theta}_r) \lambda_{dq0_r} \end{cases} \tag{3.24}$$

where

$$W(\dot{\theta}_s) = \begin{bmatrix} 0 & -\dot{\theta}_s & 0 \\ \dot{\theta}_s & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.25}$$

and

$$W(\dot{\theta}_r) = \begin{bmatrix} 0 & -\dot{\theta}_r & 0 \\ \dot{\theta}_r & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{3.26}$$

For the synchronous reference frame, i.e. with the *d*-axis moving with the rotating magnetic field,

$$\theta_s = \omega_e t \tag{3.27}$$

where ω_e is the synchronous electric frequency and

$$\begin{aligned} \theta_r = \theta_s - n_{pp} \theta_m \Leftrightarrow \\ \Leftrightarrow \theta_r = \theta_s - \theta_{me} \end{aligned} \tag{3.28}$$

Using the rationale that was used to achieve (3.21), it is possible to rewrite (3.13) in the *dq0* reference frame:

$$\begin{cases} \lambda_{dq0_s} = L_s i_{dq0_s} + L_m i_{dq0_r} \\ \lambda_{dq0_r} = L_r i_{dq0_r} + L_m i_{dq0_s} \end{cases}. \tag{3.29}$$

Using (3.24) and (3.29) one can write the equations that govern the induction motor in the *dq0* reference frame:

$$\begin{bmatrix} v_{d_s} \\ v_{q_s} \\ v_{d_r} \\ v_{q_r} \end{bmatrix} = \begin{bmatrix} R_s & 0 & 0 & 0 \\ 0 & R_s & 0 & 0 \\ R_s & 0 & R_r & 0 \\ 0 & R_s & 0 & R_r \end{bmatrix} \begin{bmatrix} i_{d_s} \\ i_{q_s} \\ i_{d_r} \\ i_{d_s} \end{bmatrix} + \begin{bmatrix} \dot{\lambda}_{d_s} \\ \dot{\lambda}_{q_s} \\ \dot{\lambda}_{d_r} \\ \dot{\lambda}_{q_r} \end{bmatrix} + \begin{bmatrix} 0 & -\omega_s & 0 & 0 \\ \omega_s & 0 & 0 & 0 \\ 0 & 0 & 0 & -\omega_s + \omega_{me} \\ 0 & 0 & \omega_s - \omega_{me} & 0 \end{bmatrix} \begin{bmatrix} \lambda_{d_s} \\ \lambda_{q_s} \\ \lambda_{d_r} \\ \lambda_{q_r} \end{bmatrix}, \tag{3.30}$$

$$\begin{bmatrix} \lambda_{d_s} \\ \lambda_{q_s} \\ \lambda_{d_r} \\ \lambda_{q_r} \end{bmatrix} = \begin{bmatrix} L_s & 0 & L_m & 0 \\ 0 & L_s & 0 & L_m \\ L_m & 0 & L_r & 0 \\ 0 & L_m & 0 & L_r \end{bmatrix} \begin{bmatrix} i_{d_s} \\ i_{q_s} \\ i_{d_r} \\ i_{d_s} \end{bmatrix}. \tag{3.31}$$

Please note that in the *dq0* reference frame, the rotor voltages, v_{d_r} and v_{d_s}, are null, as the considered induction motor is of the squirrel-cage type, i.e.

$$v_{d_r} = v_{q_r} = 0. \tag{3.32}$$

The total power of the induction motor can be obtained by multiplying the voltage with their respective components:

$$P = \frac{3}{2} \left(v_{d_s} i_{d_s} + v_{q_s} i_{q_s} \right). \tag{3.33}$$

Substituting the voltages by the definitions presented in (3.30), it is possible to rewrite (3.33) as

$$\begin{aligned} P = \frac{3}{2} (& R_s (i_{d_s}^2 + i_{q_s}^2) + \\ & + \dot{\lambda}_{d_s} i_{d_s} + \dot{\lambda}_{q_s} i_{q_s} + \dot{\lambda}_{d_r} i_{d_r} + \dot{\lambda}_{q_r} i_{q_r} \\ & + \dot{\theta}_s (\lambda_{d_s} i_{q_s} - \lambda_{q_s} i_{d_s}) - \dot{\theta}_r (\lambda_{d_r} i_{q_r} - \lambda_{q_r} i_{d_r})). \end{aligned} \tag{3.34}$$

Please note that (3.34) is purposefully divided into three lines as to denote the different sources that impact the total power of the induction motor. The first part translates the power wasted into heat due to the copper losses in the stator and rotor. The second line translates the power lost due to the changes in the magnetic field that may occur during the operation of the motor. The final term refers to the mechanical power demanded by the load attached to the motor. Focusing on this last term, it is possible to draw a conclusion as to what is the electromagnetic torque produced by the machine. Using the last line, one achieves an expression for the mechanical power of the induction motor:

$$P_m = T_e \dot{\theta}_m = \frac{3}{2} \left(\dot{\theta}_s (\lambda_{d_s} i_{q_s} - \lambda_{q_s} i_{d_s}) - \dot{\theta}_r (\lambda_{d_r} i_{q_r} - \lambda_{q_r} i_{d_r}) \right). \tag{3.35}$$

where P_m represents the mechanical power of the induction motor, T_e its electromagnetic torque and $\dot{\theta}_m$ the rotor's angular speed. Making use of (3.29) and (3.35), one can compute the electromagnetic torque:

$$T_e = \frac{3}{2} \frac{\dot{\theta}_s - \dot{\theta}_r}{\dot{\theta}_m} \frac{L_m}{L_r} \left(\lambda_{d_r} i_{q_s} - \lambda_{q_r} i_{d_s} \right). \tag{3.36}$$

Recalling that

$$\dot{\theta}_r = \dot{\theta_s} - n_{pp} \dot{\theta}_m \tag{3.37}$$

the electromagnetic torque can be further rewritten as

$$T_e = \frac{3}{2} n_{pp} \frac{L_m}{L_r} \left(\lambda_{d_r} i_{q_s} - \lambda_{q_r} i_{d_s} \right) \tag{3.38}$$

or, alternatively

$$T_e = \frac{3}{2} n_{pp} L_m \left(i_{q_s} i_{d_r} - i_{d_s} i_{q_r} \right). \tag{3.39}$$

The Field Oriented Control Algorithm

Field Oriented Control (FOC) is a control method that seeks to mimic the control of DC motors in induction machines. In DC motors, there are two DC currents, the excitation current that controls the magnitude of the magnetic flux density inside the motor and the armature current that controls the torque.

For the induction machine, FOC separates the control of the flux and torque between the two *dq* currents, the torque controlled by quadrature component of the stator's current, i_{q_s}, and flux is controlled by the direct component of the stator current, i_{d_s}. Figure 3.5 presents the FOC system structure.

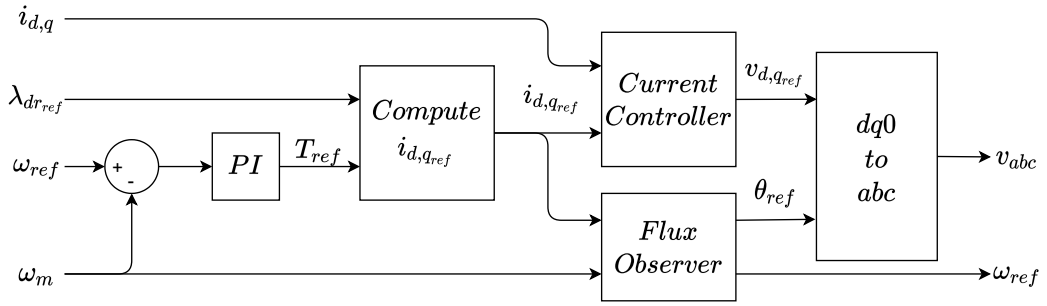


Figure 3.5: Field Oriented Control Structure.

FOC consists in aligning the direct axis of *dq0* reference frame with the rotor linkage flux, which implies that

$$\lambda_{q_r} = 0. \tag{3.40}$$

This has a series of implications, namely:

$$i_{d_r} = 0, \tag{3.41}$$

$$\lambda_{d_r} = L_m i_{d_s}, \tag{3.42}$$

$$\lambda_{d_s} = L_s i_{d_s} \tag{3.43}$$

and

$$T_e = \frac{3}{2} n_{pp} \frac{L_m}{L_r} \lambda_{d_r} i_{q_s}. \tag{3.44}$$

Please note that (3.42) indicates that the direct rotor linkage flux is only commanded by the direct stator current and that the electromagnetic torque is commanded by the stator's quadrature current. By using a field-oriented controller it is possible to have separate control of the direct flux and torque.

This poses the matter of the orientation of the rotor's quadrature linkage flux, in a manner that its value is null. For this, a flux observer must be implemented.

By making use of equations (3.42) and (3.44) it is possible to compute the direct and quadrature stator reference currents. Let $T_{e_{ref}}$ denote the reference electromagnetic torque and $\lambda_{dr_{ref}}$ the direct rotor linkage reference

flux, which corresponds to the rated peak flux. The stator direct and quadrature reference currents are therefore respectively

$$i_{ds_{ref}} = \frac{\lambda_{dr_{ref}}}{L_m} \tag{3.45}$$

and

$$i_{qs_{ref}} = \frac{T_{e_{ref}}}{\frac{3}{2} n_{pp} \frac{L_m}{L_r} \lambda_{dr_{ref}}}. \tag{3.46}$$

And the rated peak flux is given by

$$\lambda_{dr_{ref}} = \sqrt{2} I_{mb} L_m \tag{3.47}$$

where I_{mb} is the motor's no load current. The direct and quadrature stator reference voltages are computed in the "Current Controller" subsystem by computing the error between the reference stator currents and their current values and feeding the error through a pair of PI controllers. Figure 3.6 presents the block diagram of the Current Controller subsystem.

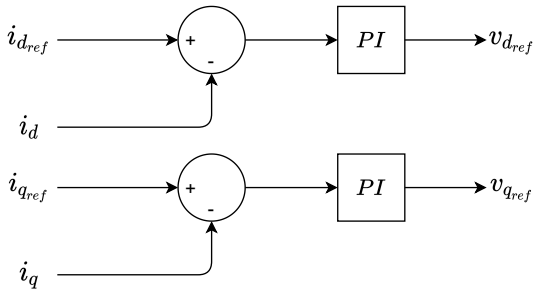


Figure 3.6: The Current Controller Subsystem.

The electrical angle and speed reference, θ_{ref} and ω_{ref} are computed in the "Flux Observer" subsystem, which is nothing more than an abstraction of the equation that arises from the consideration that the rotor's quadrature voltage is null in steady-state:

$$v_{d_r} = 0 = R_r i_{q_r} + (\omega_s - \omega_{me}) \lambda_{d_r}, \tag{3.48}$$

$$\omega_s = \omega_{me} \frac{i_{q_r}}{\lambda_{d_r}}, \tag{3.49}$$

and

$$\lambda_{d_r} = L_m i_{d_s}, \tag{3.50}$$

$$i_{q_r} = -\frac{L_r}{L_m} i_{d_s}, \tag{3.51}$$

results in

$$\omega_s = \omega_{me} + \frac{R_r}{L_r} \frac{i_{q_s}}{i_{d_s}}. \tag{3.52}$$

All that is left to do is to integrate the aforementioned angular speed to obtain the desired reference

$$\theta_{ref} = \int_0^t \omega_{me} + \frac{1}{\tau_r} \frac{i_{qs_{ref}}}{i_{ds_{ref}}} dt' + \theta_0 \quad . \tag{3.53}$$

Please note that ω_{ref} is simply the value computed before applying the integral. To output the desired phase quantities, one needs only to apply the inverse Park transform to the computed reference stator voltages, using the reference electrical angle:

$$\begin{bmatrix} v_{a_s} \\ v_{b_s} \\ v_{c_s} \end{bmatrix} = \begin{bmatrix} \cos(\theta_{ref}) & -\sin(\theta_{ref}) & 1 \\ \cos\left(\theta_{ref} - \frac{2\pi}{3}\right) & -\sin\left(\theta_{ref} - \frac{2\pi}{3}\right) & 1 \\ \cos\left(\theta_{ref} + \frac{2\pi}{3}\right) & -\sin\left(\theta_{ref} + \frac{2\pi}{3}\right) & 1 \end{bmatrix} \begin{bmatrix} v_{d_s} \\ v_{q_s} \\ v_{0_s} \end{bmatrix} \quad . \tag{3.54}$$

Chapter 4

Steering and Speed Control

In this chapter is proposed a controller to solve the path-following problem consisting of an inner MPC controller that computes the optimal acceleration and steer commands that drives the controlled variables as close as possible to their desired reference values whilst complying with the given constraints, and an outer loop speed profiler that updates the reference speed that should be tracked in case the vehicle is lagging behind the reference travelled distance.

The MPC controller uses the equations of the VIENA kinematic model presented in equation (3.2) to make predictions of the upcoming state variables given its measurements and the control signals already computed by the Linear MPC controller. Then, the VIENA kinematic model is linearized using the referred predictions, and a new set of optimal control signals is computed by the Linear MPC controller. These control signals are then applied to a more realistic version of VIENA, already presented in Chapter 3.

The outer loop speed profiler computes the optimal speed profile that drives the vehicle through the provided reference path in a given time interval while consuming the least possible energy. To accomplish this dynamics such as rolling resistances and aerodynamic drag are contemplated, as well as the reference path's inclination. Figure 4.1 presents an overview of the system's architecture.

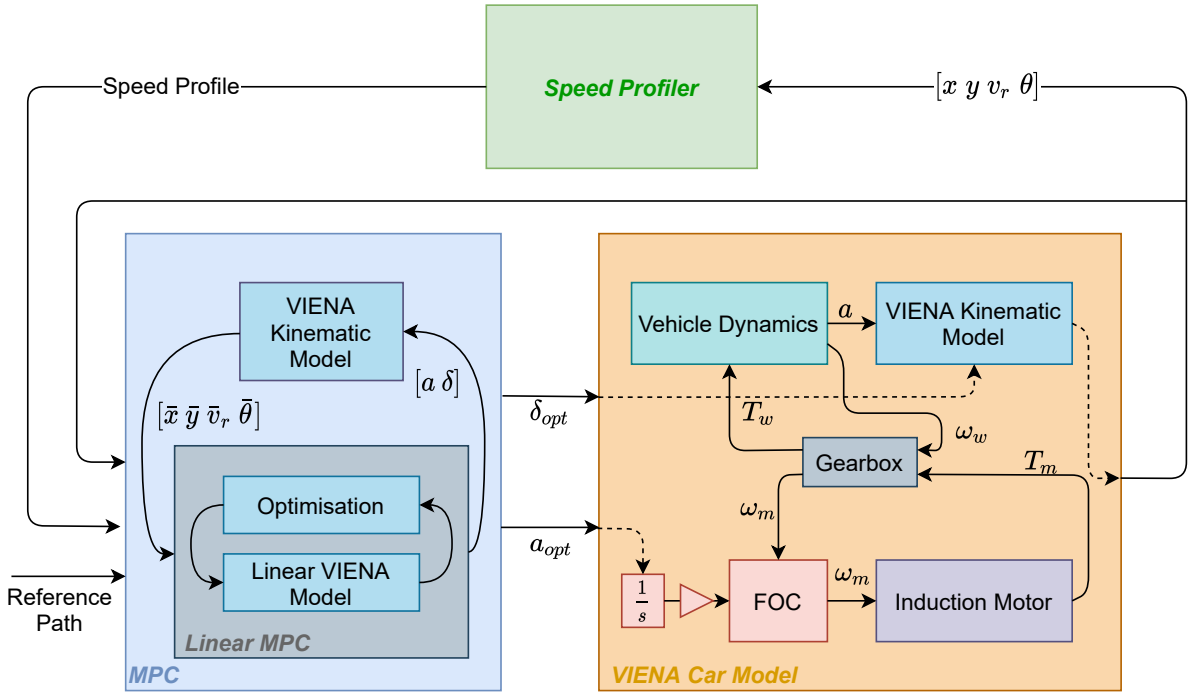


Figure 4.1: Overall System Architecture.

The following sections detail each of the major blocks. Section 4.1 details the steps taken to define the reference path. Section 4.2 explains the MPC controller utilised to generate the optimal steer and acceleration commands and section 4.3 details the computation of the optimal speed profile.

4.1 Reference Path Definition

The careful definition of the input reference path is one of the most important aspects of the path-following problem. A poorly defined reference path may imply the controller fails to drive safely the vehicle. The following sections detail how the input reference paths used to test the controller were created.

The input reference path is defined by the spatial disposition of its path points, its orientation, the distance along the path, also known as the path's length, and the path's curvature. The path's length, local orientation and curvature can all be extracted from the spatial path points. Even though these quantities are directly related to the spatial points, it is considered that it is the aggregate of these quantities that fully define a path.

Beginning with the path's spatial points, in the scope of this thesis, it is considered that the path spatial points live in $\mathbb{R}^2$, i.e. the input reference path is a two-dimensional quantity, as shown in Figure 4.2.

The vector tangent to the path can be interpreted as the path's orientation relative to the x-axis of the considered two-dimensional reference frame and can thus be computed as

$$\theta = \arctan_2 \left(\frac{\partial y}{\partial s}, \frac{\partial x}{\partial s} \right) \tag{4.1}$$

where θ is the path's orientation and s is the path's length.

The path's curvature can be seen as the variation of the path's orientation with respect to the variation of the path's length. In its most general form, it is therefore defined as

$$\kappa = \frac{d\theta}{ds} \tag{4.2}$$

where κ represents that path's curvature and s the path's length.

To have a better understanding of the aforementioned quantities, let us consider a simple example, the circular path whose curvature is well known. Figure 4.2 presents the circular path along and the quantities that will make possible the computation of its curvature.

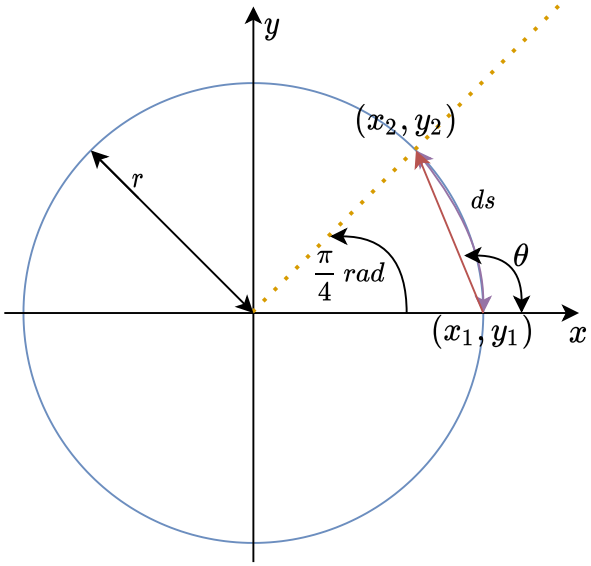


Figure 4.2: Circular path and its quantities.

First, let us recall the definition of curvature, equation (4.2). In the example of Figure 4.2, one has a 45^o circular arc, i.e. $d\theta = \theta_2 - \theta_1 = \frac{\pi}{4} - 0 = \frac{\pi}{4}\text{rad}$. From the perimeter of a circle one computes the path's length between the two path points as $ds = \frac{2\pi r}{8}$. Knowing $d\theta$ and ds, it is possible to compute the path's curvature between the two points, which is the expected result $\kappa = \frac{d\theta}{ds} = \frac{\pi}{4} \Big/ \frac{2\pi r}{8} = \frac{1}{r}$.

As it can be seen the result obtained for the circular arc path example corresponds to the well known circle's curvature. This same rationale is applied to compute the orientation and curvature of the input reference paths that will be presented henceforth.

4.2 MPC Controller

The MPC controller, receives as input the reference states, z_{ref}, and the current vehicle state, z_{meas} and outputs the optimal acceleration and steer commands, u_{opt}, that drive the vehicle as close as possible to the desired pose and speed.

It does so by first predicting the future states by using the VIENA kinematic model. These predictions, $\bar{Z}$, are then used to linearize the VIENA kinematic model so that a optimisation control problem (OCP) can be solved. This OCP outputs a sequence of vehicle control signals that are used to update the predictions made using the VIENA kinematic model. When the difference between two OCP iterative outputs is smaller than a given threshold, Δu_{min}, or the maximum number of iterations, K, has been reached, the algorithm stops and outputs the optimal control signal sequence, u_{opt}. Figure 4.3 presents a flowchart of the MPC algorithm.

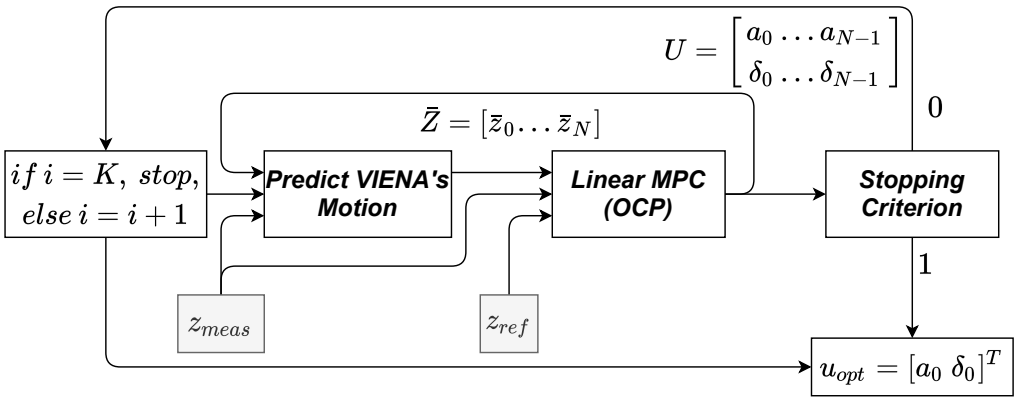


Figure 4.3: MPC Algorithm.

4.2.1 Predicting VIENA's Motion

The prediction of VIENA's future pose and speed is made with the help of its kinematic model, presented in equation (3.2), and the input signals produced by the Linear MPC Optimisation Control Problem. This results in a series of predictions made along the chosen prediction horizon.

4.2.2 Linearization and Discretization of the VIENA Kinematic Model

One of the conditions needed to guarantee convexity of the OCP is for the equality constraints to be affine functions. For that, the model of the system considered must be linear. Since VIENA's kinematic model is a non-linear system, the need to linearize it arises. Since the implementation of the OCP implies a discretization of the problem at hand, there is also the need to discretize the computed linear model.

Let us recall the VIENA kinematic model is given by (3.2) and is replicated herein for ease of reading

$$\begin{cases} \dot{x} = v_r \cos(\theta)\cos(\phi) \\ \dot{y} = v_r \sin(\theta)\cos(\phi) \\ \dot{v}_r = a \\ \dot{\theta} = \frac{v_r}{L}\tan(\delta)\cos(\phi) \end{cases} \Leftrightarrow \begin{cases} f_1(z,u) = v_r \cos(\theta)\cos(\phi) \\ f_2(z,u) = v_r \sin(\theta)\cos(\phi) \\ f_3(z,u) = a \\ f_4(z,u) = \frac{v_r}{L}\tan(\delta)\cos(\phi) \end{cases}$$

.

The linearization of the system can be foud by computing the Jacobians A' and B':

$$\dot{z} = \frac{\partial}{\partial z} z = f(z,u) \Leftrightarrow \delta\dot{z} = A'\delta z + B'\delta u \tag{4.3}$$

where

$$A' = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial v_r} & \frac{\partial f_1}{\partial \theta} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial v_r} & \frac{\partial f_2}{\partial \theta} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial v_r} & \frac{\partial f_3}{\partial \theta} \\ \frac{\partial f_4}{\partial x} & \frac{\partial f_4}{\partial y} & \frac{\partial f_4}{\partial v_r} & \frac{\partial f_4}{\partial \theta} \end{bmatrix} \Leftrightarrow A' = \begin{bmatrix} 0 & 0 & \cos(\bar{\theta})\cos(\phi) & -\bar{v}_r \sin(\bar{\theta})\cos(\phi) \\ 0 & 0 & \sin(\bar{\theta})\cos(\phi) & \bar{v}_r \cos(\theta)\cos(\phi) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\tan(\bar{\delta})\cos(\phi)}{L} & 0 \end{bmatrix} \tag{4.4}$$

and

$$B' = \begin{bmatrix} \frac{\partial f_1}{\partial a} & \frac{\partial f_1}{\partial \delta} \\ \frac{\partial f_2}{\partial a} & \frac{\partial f_2}{\partial \delta} \\ \frac{\partial f_3}{\partial a} & \frac{\partial f_3}{\partial \delta} \\ \frac{\partial f_4}{\partial a} & \frac{\partial f_4}{\partial \delta} \end{bmatrix} \Leftrightarrow B' = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & \frac{\bar{v}_r \cos(\phi)}{L\cos^2(\bar{\delta})} \end{bmatrix} \tag{4.5}$$

where $\bar{z} = [\bar{x}\ \bar{y}\ \bar{v}_r\ \bar{\theta}]^T$ and $\bar{u} = [\bar{a}\ \bar{\delta}]^T$ are equilibrium points.

It is possible to discretize the model using Forward Euler Discretization:

$$z_{k+1} = z_k + f(z_k, u_k)\Delta t. \tag{4.6}$$

Using the first degree Taylor expansion around the equilibrium point $(\bar{z}, \bar{u})$, (4.6) becomes

$$\begin{aligned} & z_{k+1} = z_k + \left(f(\bar{z},\bar{u}) + \begin{bmatrix} \frac{\partial f}{\partial z} & \frac{\partial f}{\partial u} \end{bmatrix} \left(\begin{bmatrix} z_k \\ u_k \end{bmatrix} - \begin{bmatrix} \bar{z} \\ \bar{u} \end{bmatrix} \right) \right) \Delta t \Leftrightarrow \\ & \Leftrightarrow z_{k+1} = z_k + \left(f(\bar{z},\bar{u}) + [A'\ B'] \begin{bmatrix} z_k - \bar{z} \\ u_k - \bar{u} \end{bmatrix} \right) \Delta t \Leftrightarrow \\ & \Leftrightarrow z_{k+1} = z_k + (f(\bar{z},\bar{u}) + A'(z_k - \bar{z}) + B'(u_k - \bar{u}))\,\Delta t \Leftrightarrow \\ & \Leftrightarrow z_{k+1} = z_k + (f(\bar{z},\bar{u}) + A'z_k - A'\bar{z} + B'u_k - B'\bar{u})\Delta t \Leftrightarrow \\ & \Leftrightarrow z_{k+1} = (I + A'\Delta t)z_k + B'\Delta t u_k + (f(\bar{z},\bar{u}) - A'\bar{z} - B'\bar{u})\Delta t \Leftrightarrow \\ & \Leftrightarrow z_{k+1} = Az_k + Bu_k + C \end{aligned} \tag{4.7}$$

where

$$A(\bar{z},\bar{u}) = (I + A'\Delta t) = \begin{bmatrix} 1 & 0 & \cos(\bar{\theta})\cos(\phi)\Delta t & -\bar{v}_r \sin(\bar{\theta})\cos(\phi)\Delta t \\ 0 & 1 & \sin(\bar{\theta})\cos(\phi)\Delta t & \bar{v}_r \cos(\bar{\theta})\cos(\phi)\Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{\bar{v}_r \tan(\bar{\delta})\cos(\phi)\Delta t}{L} & 1 \end{bmatrix}, \tag{4.8}$$

$$B(\bar{z},\bar{u}) = B'\Delta t = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \Delta t & 0 \\ 0 & \frac{\bar{v}_r \cos(\phi)\Delta t}{L\cos^2(\bar{\delta})} \end{bmatrix} \tag{4.9}$$

and

$$\begin{aligned} &C(\bar{z},\bar{u}) = (f(\bar{z},\bar{u}) - A'\bar{z} - B'\bar{u})\Delta t \Leftrightarrow \\ &\Leftrightarrow C = \left(\begin{bmatrix} \bar{v}_r\cos(\bar{\theta})\cos(\phi) \\ \bar{v}_r\cos(\bar{\theta})\cos(\phi) \\ \bar{a} \\ \frac{\bar{v}_r\tan(\bar{\delta})\cos(\phi)}{L} \end{bmatrix} - \begin{bmatrix} \bar{v}_r\cos(\bar{\theta})\cos(\phi) - \bar{v}_r\bar{\theta}\sin(\bar{\theta})\cos(\phi) \\ \bar{v}_r\sin(\bar{\theta})\cos(\phi) + \bar{v}_r\bar{\theta}\cos(\bar{\theta})\cos(\phi) \\ 0 \\ \frac{\bar{v}_r\tan(\bar{\delta})\cos(\phi)}{L} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \bar{a} \\ \frac{\bar{\delta}\bar{v}_r\cos(\phi)}{L\cos^2(\bar{\delta})} \end{bmatrix} \right) \Delta t \Leftrightarrow \\ &\Leftrightarrow C = \begin{bmatrix} \bar{v}_r\bar{\theta}\sin(\bar{\theta})\cos(\phi)\Delta t \\ -\bar{v}_r\bar{\theta}\cos(\bar{\theta})\cos(\phi)\Delta t \\ 0 \\ -\frac{\bar{\delta}\bar{v}_r\cos(\phi)\Delta t}{L\cos^2(\bar{\delta})} \end{bmatrix}. \end{aligned} \tag{4.10}$$

The final assumption made is that the equilibrium point $\bar{u} = [0\ 0]^T$. In practical terms, this means that the linearization is made for $\bar{\delta} = 0$. Hence, matrices $A(\bar{z},\bar{u})$, $B(\bar{z},\bar{u})$ and $C(\bar{z},\bar{u})$ degenerate into

$$A(\bar{z}) = \begin{bmatrix} 1 & 0 & \cos(\bar{\theta})\cos(\phi)\Delta t & -\bar{v}_r\sin(\bar{\theta})\cos(\phi)\Delta t \\ 0 & 1 & \sin(\bar{\theta})\cos(\phi)\Delta t & \bar{v}_r\cos(\bar{\theta})\cos(\phi)\Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.11}$$

$$B(\bar{z}) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \Delta t & 0 \\ 0 & \frac{\bar{v}_r\cos(\phi)\Delta t}{L} \end{bmatrix} \tag{4.12}$$

and

$$C(\bar{z}) = \begin{bmatrix} \bar{v}_r\bar{\theta}\sin(\bar{\theta})\cos(\phi)\Delta t \\ -\bar{v}_r\bar{\theta}\cos(\bar{\theta})\cos(\phi)\Delta t \\ 0 \\ 0 \end{bmatrix}. \tag{4.13}$$

With matrices $A(\bar{z})$, $B(\bar{z})$ and $C(\bar{z})$ computed, it is now possible to have a discrete linear representation of the VIENA model, which will be used in the formulation of the Linear MPC problem to update the state vector.

4.2.3 Linear MPC Formulation

The optimisation control problem (OCP) solved by the MPC controller optimises the control vector, u, over a certain prediction horizon, denoted T_p. Since the implementation of an OCP implies a discretization of the problem at hand, not only the state-space model of the linearized VIENA vehicle needs to be discretized, but also the prediction horizon must be converted into a discrete number, N. Let $\Delta t \in \mathbb{R}$ denote the discretization step of the considered problem. N is therefore

$$N = \frac{T_p}{\Delta t} \in \mathbb{N}. \tag{4.14}$$

It is important to take into account that the OCP produces a solution for each step of the prediction horizon. Let m denote the dimension of the control vector and p denote the dimension of the state vector. For the considered dimensions, the OCP would output a solution in which $U \in \mathbb{R}^{m \times N-1}$ and $Z \in \mathbb{R}^{p \times N}$, where U denotes the set of control inputs and Z the states produced by subjecting the internal state-space model to the computed control signals.

The proposed MPC controller makes use of the template provided in [18], replicated herein, and was based on [25] with slight changes to the notation in order for it to be consistent with the proposed linear MPC controller:

$$\begin{aligned}
\underset{U \in \mathbb{R}^{m \times N-1}}{\text{minimize}} \quad & p(z_N) + \sum_{k=0}^{N-1} q(z_k, u_k) && && \text{(4.15a)}\\
\text{subject to} \quad & z_{k+1} = Az_k + Bu_k, && k = 0, \ldots, N-1, && \text{(4.15b)}\\
& z_k \in \mathbb{Z}, && k = 0, \ldots, N, && \text{(4.15c)}\\
& z_N \in \mathbb{Z}_f, && && \text{(4.15d)}\\
& z_0 = z_{init}, && && \text{(4.15e)}\\
& u_k \in \mathbb{U}, && k = 0, \ldots, N-1. && \text{(4.15f)}
\end{aligned}$$

Equation (4.15a) denotes the problem's cost function and equations (4.15b) - (4.15f) denote the constraints. The cost function can be divided into two separate costs, the terminal cost, $p(z_N)$ and the stage cost, $q(z_k, u_k)$. The constraints can be broken down into two sets: the state constraints, denoted by equations (4.15b) through (4.15e), and the input constraints, denoted by equation (4.15f).

The proposed cost function makes use of a weighted sum of squared errors to define the terminal and stage costs. This means that the terminal cost is defined by

$$p(z_N) = (z_N^{ref} - z_N)^T W_f (z_N^{ref} - z_N) \tag{4.16}$$

where z_N^{ref} is the provided reference state at the terminal iteration N (recall Figure 4.3) and $W_f \in \mathbb{R}^{p \times p} \succ 0$ denotes the terminal cost weight matrix. The stage cost is given by

$$q(z_k, u_k) = (z_k^{ref} - z_k)^T W (z_k^{ref} - z_k) + u_k^T R u_k + (u_k - u_{k-1})^T R_d (u_k - u_{k-1}) \tag{4.17}$$

where z_k^{ref} denotes the provided reference state at iteration k, $W \in \mathbb{R}^{p \times p} \succ 0$ the stage cost weight matrix, $R \in \mathbb{R}^{m \times m} \succ 0$ the input cost weight matrix and $R_d \in \mathbb{R}^{m \times m} \succ 0$ the input difference cost weight matrix. Matrix R allows the penalisation of very large inputs and matrix R_d the penalisation of big differences between consecutive inputs, in an effort to have smoother input signals.

Moving on to the OCP's constraints, the internal linearized VIENA model is given by

$$z_{k+1} = A(\bar{z})z_k + B(\bar{z})u_k + C(\bar{z}) \tag{4.18}$$

where $A(\bar{z})$, $B(\bar{z})$ and $C(\bar{z})$ are the linearized discrete matrices that model VIENA's movement. These matrices are updated using the predictions made using the VIENA kinematic model, as seen in Figure 4.3. The linearization and discretization of VIENA's kinematic model are detailed in section 4.2.2, but are nonetheless reproduced herein:

$$A(\bar{z}) = \begin{bmatrix} 1 & 0 & \cos(\bar{\theta})\cos(\phi)\Delta t & -\bar{v}_r \sin(\bar{\theta})\cos(\phi)\Delta t \\ 0 & 1 & \sin(\bar{\theta})\cos(\phi)\Delta t & \bar{v}_r \cos(\bar{\theta})\cos(\phi)\Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.19}$$

$$B(\bar{z}) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \Delta t & 0 \\ 0 & \frac{\bar{v}_r \cos(\phi)\Delta t}{L} \end{bmatrix}, \tag{4.20}$$

$$C(\bar{z}) = \begin{bmatrix} \bar{v}_r \bar{\theta} \sin(\bar{\theta})\cos(\phi)\Delta t \\ -\bar{v}_r \bar{\theta} \cos(\bar{\theta})\cos(\phi)\Delta t \\ 0 \\ 0 \end{bmatrix}. \tag{4.21}$$

The remaining state constraints are the VIENA's maximum and minimum speed constraints:

$$v_{r_{min}} \leq v_{r_k} \leq v_{r_{max}}. \tag{4.22}$$

The input constraints encompass the maximum steering speed and the upper and lower bounds of the steer and acceleration commands:

$$|\delta_k - \delta_{k-1}| \leq \Delta\delta_{max}, \tag{4.23}$$

$$\delta_{min} \leq \delta_k \leq \delta_{max}, \tag{4.24}$$

$$a_{min} \leq a_k \leq a_{max}. \tag{4.25}$$

The initial condition constraint is simply given by the measured state vector:

$$z_0 = z_{init} = z_{meas}. \tag{4.26}$$

With the cost function and the state and input constraints fully defined, the proposed Linear MPC controller that solves the path-following problem can be completely expressed. Let there be recalled that, for the problem at hand, $z = [x\ y\ v_r\ \theta]^T \in \mathbb{R}^4$ and $u = [a\ \delta]^T \in \mathbb{R}^2$ and therefore the OCP is given by

$$\underset{U \in \mathbb{R}^{2\times N-1}}{\text{minimize}} \quad (z_N^{ref} - z_N)^T W_f (z_N^{ref} - z_N) + \sum_{k=0}^{N-1} (z_k^{ref} - z_k)^T W (z_k^{ref} - z_k) + \tag{4.27a}$$

$$u_k^T R u_k + (u_k - u_{k-1})^T R_d (u_k - u_{k-1}) \tag{4.27b}$$

$$\text{subject to} \quad z_{k+1} = A(\bar{z}_k) z_k + B(\bar{z}_k) u_k + C(\bar{z}_k), \quad k = 0, \ldots, N-1, \tag{4.27c}$$

$$v_{r_{min}} \leq v_{r_k} \leq v_{r_{max}}, \quad k = 0, \ldots, N, \tag{4.27d}$$

$$\delta_{min} \leq \delta_k \leq \delta_{max}, \quad k = 0, \ldots, N-1, \tag{4.27e}$$

$$a_{min} \leq a_k \leq a_{max}, \quad k = 0, \ldots, N-1, \tag{4.27f}$$

$$|\delta_k - \delta_{k-1}| \leq \Delta\delta_{max}, \quad k = 0, \ldots, N-1, \tag{4.27g}$$

$$z_0 = z_{meas} \tag{4.27h}$$

With the OCP fully defined there is now the need to validate its convexity, feasibility and stability. Annex A details these subjects.

4.2.4 Stopping Criterion for the MPC Controller

The MPC algorithm stops either when the final iteration has been reached or when the sum of the differences between two consecutive vehicle control signals does not exceed a certain threshold. In other hands, when two consecutive vehicle control signals computed by the Linear MPC are very similar, there is no point in adapting the model further as the Linear MPC has reached a form of consensus on the optimal input signals.

The variation on two consecutive input signals Δu is therefore computed as

$$\Delta u = \sum_{i=1}^{N-1} |\delta_i - \delta_{i-1}| + \sum_{i=1}^{N-1} |a_i - a_{i-1}|. \tag{4.28}$$

Please note that in (4.28), $N-1$ denotes the dimension of the computed input signal matrices, as seen in 4.2.3.

When that variation is lower than a given threshold Δu_{min}, the MPC algorithm stops. As a mathematical expression, the stopping criterion is

$$\Delta u \leq \Delta u_{min}. \tag{4.29}$$

4.3 Speed Profiler

Since there is the need to produce a reference for the vehicle's speed, the possibility for the computation of a optimal speed profile arises. The proposed solution consists on a minimum energy speed profile optimisation that takes into account a time constraint, by providing the optimisation problem a time interval in which the track should be performed and penalising differences between said time interval and the time the vehicle took to complete the given reference path.

The following subsections detail two different possibilities for the computation of the energy needed to achieve an optimal speed profile. The first assumes linear speed variation between two points in the path's length, while the other assumes constant acceleration between two points. Ultimately, the solution that was chosen was the latter since it more closely resembles the actions of a human driver.

4.3.1 Linear Speed Variation Between Two Points in the Path's Length

The assumption made for the considered speed profile energy computation is that the velocity variation between two consecutive spatial points is an affine function of the form

$$v(x) = kx + v_i. \tag{4.30}$$

By computing the integral of the velocity in time, an expression of the position of the vehicle can be achieved

$$\begin{aligned} x(t) = \int_{t_i}^{t} v(x)dt \Leftrightarrow x(t) = \int_{t_i}^{t} kx + v_i dt \Leftrightarrow \\ \Leftrightarrow x(t) = \int kx(t)dt\Big|_{t} - \int kx(t)dt\Big|_{t_i} + v_i(t - t_i) \Leftrightarrow \\ \Leftrightarrow x(t) = \int kx(t)dt - kx_i + v_i(t - t_i). \end{aligned} \tag{4.31}$$

Please note that in (4.31), t_i, v_i and x_i denote the initial time, speed and position, respectively. Taking its time derivative, one achieves

$$\begin{aligned} &\dot{x} = kx - kx_i + v_i \Leftrightarrow \\ &\Leftrightarrow \dot{x} - kx - v_i = 0. \end{aligned} \tag{4.32}$$

By solving (4.32) as an ordinary differential equation, one achieves the general solution

$$x(t) = -\frac{v_i}{k} + x_i + Ae^{kt} \tag{4.33}$$

where $-\frac{v_i}{k} + x_i$ corresponds to the forced regime and Ae^{kt} corresponds to the free transient regime.

Using the initial condition $x(t_i) = x_i$, one can find the value of A:

$$t = t_i : x(t_i) = x_i = -\frac{v_i}{k} + Ae^{kt_i} \rightarrow A = \frac{v_i}{ke^{kt_i}}. \tag{4.34}$$

By knowing that $x(t_{i+1}) = x_{i+1}$, it is possible to know the time, t_{i+1} at which the vehicle arrives at x_{i+1}:

$$t_{i+1} = \frac{1}{k} \ln \left(\frac{1}{A} \left(x_{i+1} - x_i + \frac{v_i}{k} \right) \right). \tag{4.35}$$

Please note that (4.35) considers that $v_{i+1} \neq v_i$, as if $k = 0$, the solution could not be determined. Would one consider otherwise, t_{i+1} can be computed in a simpler manner, similar to the region of constant speed, as in (2.44):

$$t_{i+1} = \frac{x_{i+1} - x_i}{v_i} + t_i. \tag{4.36}$$

To estimate the energy spent, one can resort to the definition of work. Between to consecutive time instants, the energy consumed, E_{T_i} is given by

$$E_{T_i} = \int_{t_i}^{t_{i+1}} P_T(t)dt \Leftrightarrow E_{T_i} = \int_{t_i}^{t_{i+1}} F_T(t)v(t)dt, \tag{4.37}$$

where $F_T(t)$ is the traction force needed to carry the vehicle forward with a given acceleration, and can be computed with the help of equations (3.3) through (3.8):

$$\begin{aligned} &F_T(t) = Ma(t) - Fg' - F_r - F_a(t) \Leftrightarrow \\ &\Leftrightarrow F_T(t) = Ma(t) + mg\sin(\phi) + C_{rr}mg\cos(\phi) + \frac{1}{2}\rho C_d A_f v(t)^2. \end{aligned} \tag{4.38}$$

When $v_{i+1} \neq v_i$, the speed and acceleration of the vehicle can be computed by taking the time derivative of (4.33):

$$v(t) = kAe^{kt}, \tag{4.39}$$

$$a(t) = k^2 Ae^{kt}. \tag{4.40}$$

With the expressions of $F_T(t)$, $v(t)$ and $a(t)$ obtained, it is possible to rewrite $P_T(t)$ as an expression of time:

$$
\begin{aligned}
&P_T(t) = F_T(t)v(t) \Leftrightarrow \\
&\Leftrightarrow P_T(t) = Mk^3A^2e^{2kt} + mg(\sin(\phi) + C_{rr}\cos(\phi))e^{kt} + \frac{1}{2}\rho C_d A_f k^3 A^3 e^{3kt} \Leftrightarrow \\
&\Leftrightarrow P_T(t) = a_1e^{2kt} + b_1e^{3kt} + c_1e^{kt}.
\end{aligned} \tag{4.41}
$$

The consumed energy between points x_i and x_{i+1}, corresponding to time instants t_i and t_{i+1} is given by

$$
E_{T_i} = \int_{t_i}^{t_{i+1}} P_T(t)dt = \left[\frac{a_1}{2k}e^{2kt} + \frac{b_1}{3k}e^{3kt} + \frac{c_1}{k}e^{kt}\right]_{t_i}^{t_{i+1}}. \tag{4.42}
$$

In the particular case in which $v_{i+1} = v_i$, $a(t) = 0$ and therefore

$$
P_T = mg\sin(\phi) + C_{rr}mg\cos(\phi) + \frac{1}{2}\rho C_d A_f v_i^2 \tag{4.43}
$$

and

$$
E_{T_i} = P_T(t_{i+1} - t_i). \tag{4.44}
$$

This method of computing the energy consumed between two consecutive path length points has the drawback of being defined via exponentials, which do not provide a solution when $v(t) = 0$. This is particularly troublesome either when the desired initial or final speed are null. If the initial speed is null, this means that the vehicle will never leave the initial position, and, therefore, the speed cannot increase. In practise, this problem can be easily solved by defining as the initial and final speed a value close to zero, for example, $v_{init,final} = 0.1\,m/s$. To further solve this problem, the constant acceleration solution is explored in the next subsection.

4.3.2 Constant Acceleration Between Two Points in the Path's Length

In comparison with the previous presented solution, the constant acceleration formulation more closely resembles the actuation of a human driver, since a driver controls not the speed but rather the motor torque via the acceleration pedal of the vehicle.

This formulation makes use of the typical equations of linear motion. If the acceleration between two points is constant then

$$
v(t) = at + v_i \tag{4.45}
$$

and

$$
x(t) = \frac{1}{2}at^2 + v_it + x_i. \tag{4.46}
$$

Let v_i denote the speed at track point x_i and v_{i+1} denote the speed at track point x_{i+1}. Applying (4.45) results in

$$
\Delta t = \frac{v_{i+1} - v_i}{a} \Leftrightarrow \Delta t = \frac{\Delta v}{a} \tag{4.47}
$$

where Δt is the time interval between two spatial points, x_i and x_{i+1}. It is also possible to rewrite (4.46) as

$$
x_{i+1} - x_i = \frac{1}{2}a\Delta t^2 + v_i\Delta t \Leftrightarrow \Delta x = \frac{1}{2}a\Delta t^2 + v_i\Delta t. \tag{4.48}
$$

Using (4.47) into (4.48) results in

$$\Delta x = \frac{1}{2}a\frac{\Delta v^2}{a^2} + v_i\frac{\Delta v}{a} \Leftrightarrow \Delta x = \frac{1}{2}\frac{\Delta v}{a} + v_i\frac{\Delta v}{a} \Leftrightarrow a = \frac{1}{\Delta x}\left(\frac{\Delta v^2}{2} + v_i\Delta v\right). \tag{4.49}$$

Please note that, in the scope of this problem, Δx is a known value, and therefore, only Δv impacts the value of a. To compute the needed energy to complete the path, two distinct cases must be considered: the one in which $\Delta v \neq 0$ and its counterpart, i.e. $\Delta v = 0$.

Beginning with the latter, if $\Delta v = 0$, then $a = 0$ and one can compute Δt with the use of (4.36):

$$\Delta t = \frac{\Delta x}{v_i}. \tag{4.50}$$

Since $a = 0$, the speed is constant, as is the traction power:

$$E_{T_i} = P_{T_i}\Delta t_i \Leftrightarrow E_{T_i} = F_{T_i}v_i\Delta t_i \Leftrightarrow E_{T_i} = \left(mg\left(\sin(\phi) + C_{rr}\cos(\phi)\right) + \frac{1}{2}\rho C_d A_f v_i^2\right)v_i\Delta t_i. \tag{4.51}$$

In the case that $\Delta v \neq 0$, Δt can be obtained by solving (4.48) as function of Δt:

$$\Delta t_i = \frac{-b_1 + \sqrt{b_1^2 - 4a_1c_1}}{2a_1} \tag{4.52}$$

with

$$\begin{aligned} a_1 &= \frac{1}{2}a \\ b_1 &= v_i \\ c_1 &= -\Delta x. \end{aligned} \tag{4.53}$$

Knowing that

$$E_{Ti} = \int_0^{\Delta t_i} P_{Ti}dt, \tag{4.54}$$

and

$$\begin{aligned} &P_{Ti} = F_{Ti}v_i \Leftrightarrow \\ &\Leftrightarrow P_{Ti} = \left(ma + \frac{1}{2}\rho C_d A_f v_i^2 + mg(C_{rr}\cos(\phi) + \sin(\phi))\right)v_i \end{aligned} \tag{4.55}$$

and recalling (4.45) it is possible to compute E_{Ti}:

$$E_{Ti} = ma\Delta t_i\left(a\frac{\Delta t_i}{2} + v_i\right) + \frac{\beta}{4a}\left((a\Delta t_i + v_i)^4 - v_i^4\right) + \gamma\Delta t_i\left(a\frac{\Delta t_i}{2} + v_i\right) \tag{4.56}$$

where β and γ are respectively

$$\begin{aligned} \beta &= \frac{1}{2}\rho C_d A_f \\ \gamma &= mg(C_{rr}\cos(\phi) + \sin(\phi)). \end{aligned} \tag{4.57}$$

In the case that there is no regenerative braking,

$$E_{T_i} = \max\{0, E_{T_i}\} \quad . \tag{4.58}$$

4.3.3 Optimisation Problem Formulation

Now that the different ways of calculating the energy necessary to complete a portion of the path have been presented, an optimisation problem can be formulated to find the optimal speed profile that minimises energy. The current speed profile is found by solving the following optimisation problem:

$$\begin{aligned}
\underset{v \in \mathbb{R}^N}{\text{minimize}} \quad & \sum_{i=0}^{N} E_{T_i} + \alpha |t_{max} - \sum_{n=0}^{N} \Delta t_i| && \text{(4.59a)} \\
\text{subject to} \quad & v_0 = v_{init}, && \text{(4.59b)} \\
& v_N = v_{final}, && \text{(4.59c)} \\
& |a_i| - a_{max} \leq 0, \quad i = 0, \ldots, N-1. && \text{(4.59d)}
\end{aligned}$$

In optimisation problem (4.59a) - (4.59d), E_{T_i} is computed using the solution proposed in section 4.3.2. Equations (4.59b) and (4.59c) provide the initial and final speeds and (4.59d) guarantees that the acceleration computed does not exceed the maximum acceleration available.

The optimisation problem's cost function minimises both the energy required to complete the path and the difference between the target time and the time elapsed, in which α is a weighting constant translating the relative importance of one term of the cost function in relation to the other. The reason for having this latter term in the cost function rather than in the constraints is to smooth the time constraint. Would it be a constraint, it could happen that the time needed to complete the course would exceed the time available, which would result in an infeasible solution. By having this term in the cost function, the optimisation problem becomes more lenient, and the importance of completing the course in a given time frame can be adjusted by increasing the value of α. Please note that t_{max} is computed with the help of the equations found in section 2.3.1

$$t_{max} = \gamma \left(\Delta t_1 + \Delta t_2 + \Delta t_3 \right), \tag{4.60}$$

where γ seeks to represent a "relaxation" constant and Δt_1, Δt_2 and Δt_3 correspond to the acceleration, constant speed and deceleration time intervals of the Minimum Time Speed Profile Optimisation section present in Chapter 2.

Chapter 5

Experiments and Results

This chapter details the experiments that were conducted to evaluate the performance of the developed controller. The first experiment presented serves to provide an overall vision of the controller, more specifically, it highlights the updating step of the reference speed profile.

The next experiment will focus on the fine-tuning of the weight matrices present in the Linear MPC cost function, and how their values impact the performance of the proposed controller. It will also present the combination of weight matrices that provides the best controller performance

The final experiment will compare the performance of the developed controller when compared against two well established path-following controllers: the Pure Pursuit and the Front Wheel Position Based Feedback controllers, presented in sections 2.2.1 and 2.2.2 respectively. Since the aforementioned controllers do not contemplate the control of the vehicle's speed, a simple proportional controller was implemented to have the vehicle follow the speed profile as close as possible. The VIENA model, presented in Chapter 3 will be used as the vehicle model. This will allow to compare and contrast the performance of the controllers in the presence of unforeseen dynamics, such has inertias, friction, motor dynamics and so forth.

In order to measure the performance of the aforementioned controllers, a set of metrics must be chosen. To evaluate how closely the controller is capable of making the vehicle follow the reference path, the euclidean distance between the current vehicle position and the closest path point will be used. To quantify the error between the reference speed and the vehicle's speed, the absolute value of the error was chosen as a metric. To further analyse the controller's performance, a small statistical characterisation of the errors is performed, namely their maximum, minimum and average values are presented.

Since the focus of the speed profile optimisation is the minimisation of the energy spent, the energy spent during the path-following process will also serve has a performance metric. It is also important that the vehicle completes the track in the allotted time and thus the elapsed time during the path-following process in comparison with the predicted time will also serve as a performance metric. One final metric that must be accounted for is the smoothness of the steer command. It is desirable that there are not abrupt changes in it as to provide a safe and comfortable driving experience. To evaluate this, the moving standard deviation of the steer command was used.

5.1 Overall Operation of the Proposed Controller

This first experiment seeks to provide an overarching vision on the proposed controller's operation. In this experiment, it is possible to observe not only the overall operation of the MPC controller concerning its path-following and velocity tracking capabilities, but are also highlighted the updating steps of the reference speed profile. The computation of the aforementioned reference speed profile is done by solving the optimisation problem (4.59a)-(4.59d), with the computation of the needed energy, E_{T_i}, being performed as shown in section 4.3.2. The reference speed profile is updated in periods of ten seconds if the position of VIENA along the path is five meters or more away from its predicted position. Figures 5.1 shows the evolution of VIENA's position and speed along the reference path, as well as the changes in the reference speed profile. The following results were obtained with the weight matrices of equation (4.27a) equal to

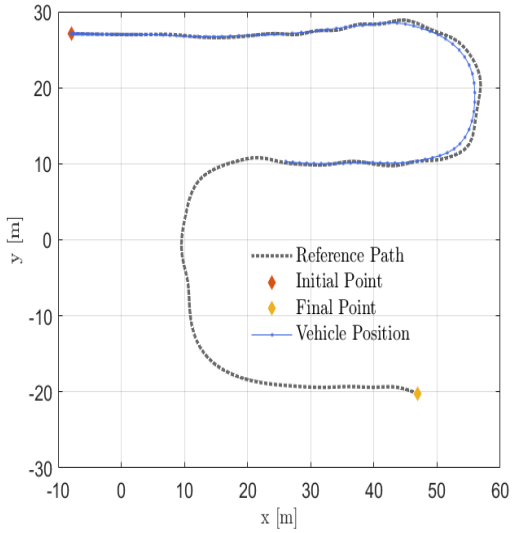
$$W_f = W = \begin{bmatrix} 0.75 & 0 & 0 & 0 \\ 0 & 0.75 & 0 & 0 \\ 0 & 0 & 0.75 & 0 \\ 0 & 0 & 0 & 0.75 \end{bmatrix}, \quad R = R_d = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}. \tag{5.1}$$

Figures 5.1(a) and 5.1(b) show the initial position of VIENA, the initial computed reference speed profile, and the evolution of the vehicle's position and speed until a travelled distance of approximately 105 meters, at which point a new reference speed will be defined due to the error between the vehicle and the reference position being too high.

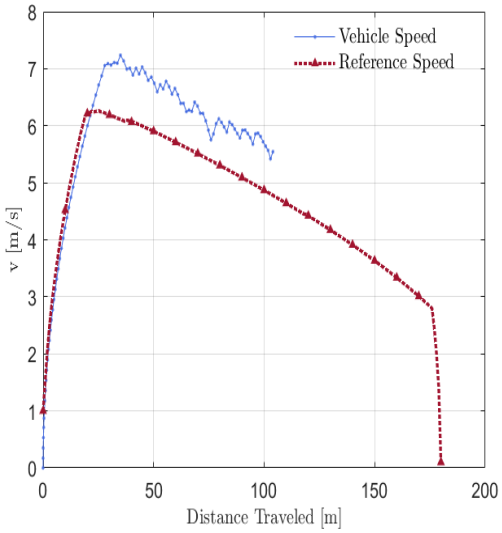
As it can be seen in Figures 5.1(a) and 5.1(b), although the proposed MPC controller is capable of making VIENA follow the provided reference path very closely and anticipate the path's turns, the speed reference tracking is not as good. At this point in the track, because the VIENA's speed has been above the reference speed, it is ahead of the predicted position and thus a new reference speed profile must be computed. This new speed profile is computed taking into account not only the current position and speed of the car but also the remainder of the time available to complete the reference path. Figure 5.2(d) shows the speed profile after its updating step.

After the speed profile has been updated, VIENA will continue its progress, trying to follow the references provided as close as possible. Should the need to update the speed profile again arise, the same process described above will take place.

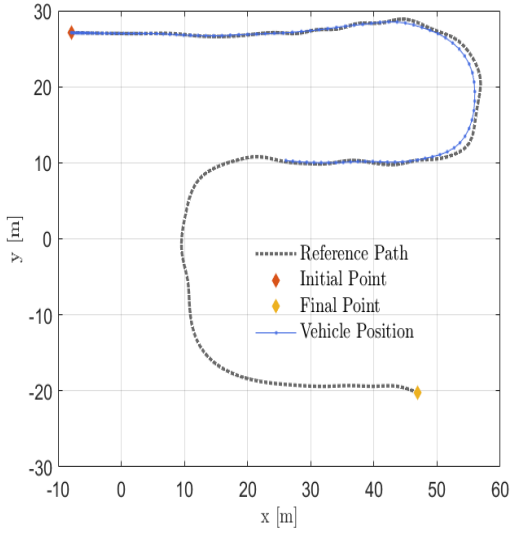
When VIENA reaches the path point seen in Figure 5.1(e), it is again time to update the reference speed profile since its speed has still been above the reference profile, VIENA is still ahead of the predicted reference position. A new speed profile must then be computed for the remainder of the path. The computed speed profile can be seen in Figure 5.1(f). This final reference speed profile will lead VIENA to the end of the reference path. Figure 5.2 shows the final results for this path-following experiment as well as the errors between the reference path and the vehicle's position and the reference speed profile and the vehicle's speed. Tables 5.1 and 5.2 show the remainder of the metrics that evaluate the controller's performance.



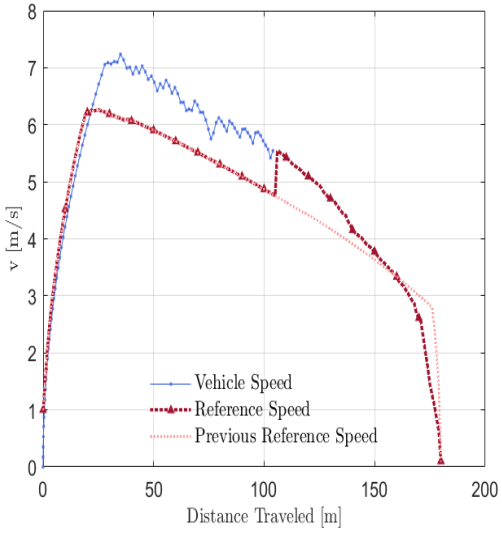
(a) Reference path and Vehicle Position



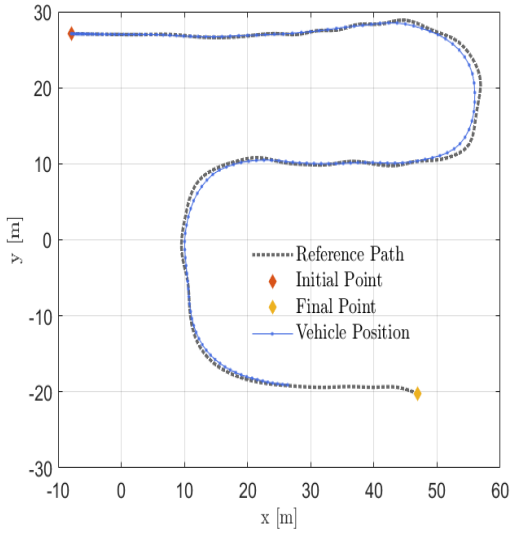
(b) Speed profile



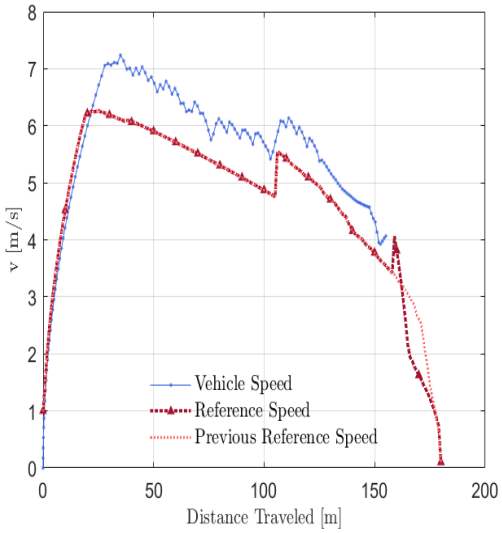
(c) Reference Path and Vehicle Position



(d) Speed profile

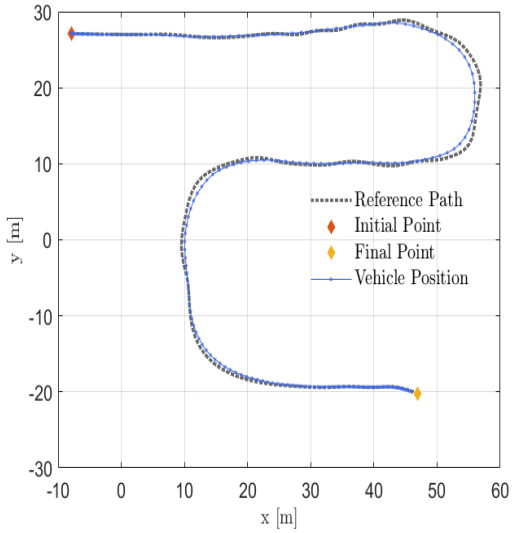


(e) Reference Path and Vehicle Position

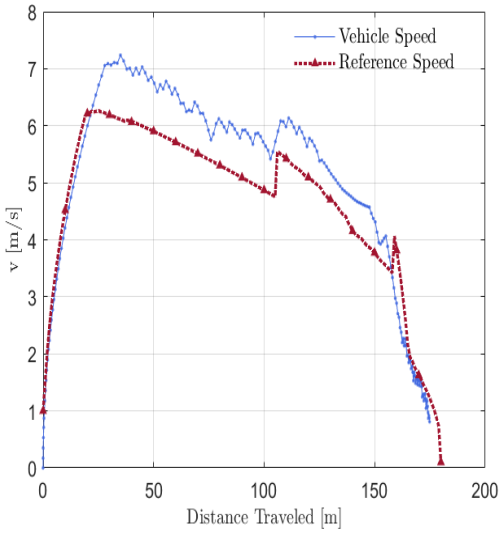


(f) Speed profile

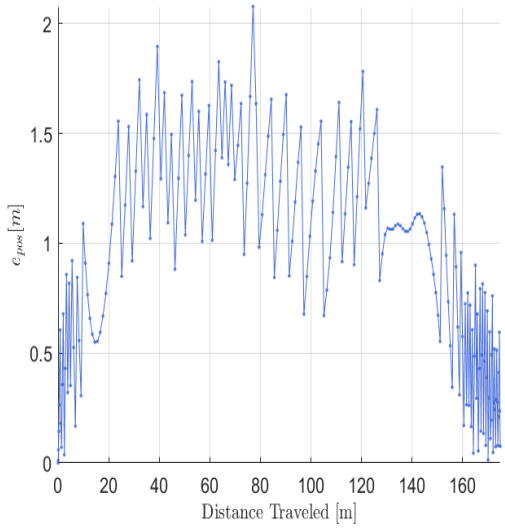
Figure 5.1: Vehicle position and speed profile evolution along the path .



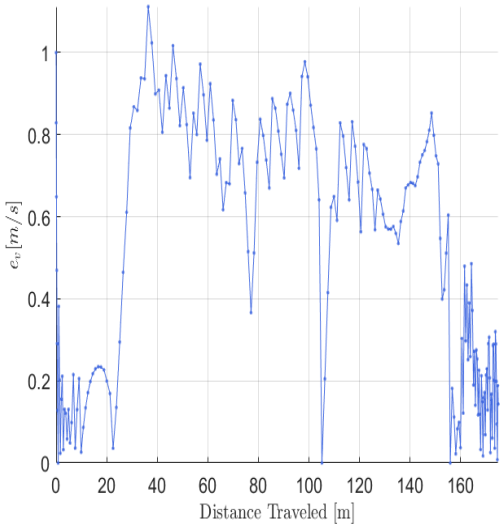
(a) Reference path and Vehicle Position



(b) Speed profile



(c) Distance to Path



(d) Speed Tracking Error

Figure 5.2: Final vehicle position, speed profile and their respective error metrics.

Table 5.1: Consumed energy and elapsed time target and real values.

	Target Values	Real Values	$\epsilon_r[\%]$
E_T [kWh]	0.4640	0.5789	24.76
t [s]	44.67	40.60	10.03

Table 5.2: Position and speed error characteristics.

e_{pos}^{max} [m]	2.07	e_v^{max}[m/s]	1.11
e_{pos}^{min}[m]	0.01	e_v^{min}[m/s]	0.008
$\bar{e}_{pos}$[m]	0.90	$\bar{e}_v$[m/s]	0.498

As it can be seen in Figure 5.2 the proposed controller is capable of solving the path-following problem, albeit with some limitations. While Figures 5.2(a) and 5.2(c) show that the vehicle is capable of closely following the reference path, Figures 5.2(b) and 5.2(d) evidence the limitations of the controller in keeping track of the reference speed profile.

This behaviour is partly due to the weights in the W_f, W, R and R_d matrices. To highlight the proposed controller's difficulty in closely following the reference speed profile, the values of the W_f and W were all made equal. In truth, this is not the combination that produces neither the best path-following solution nor the best speed reference tracking solution.

The following section presents the fine-tuning process to which these matrices were subjected in order to obtain the best possible performance out of the proposed MPC controller.

As a choice, these weights are tuned in such a way that the proposed controller privileges the path-following problem rather than the reference speed tracking as it is considered that the latter is more important than the former.

5.2 Controller Fine-tuning

As it can be seen in section 4.2.3, especially in equation (4.27a), the considered MPC controller has a rather large number of constants that have a direct impact on its performance. The values of the weights present in matrices W_f, W, R and R_d have a severe impact on the overall performance of the controller. Let us recall that all the aforementioned matrices are diagonal, that W_f and $W \in \mathbb{R}^{4\times4}$ and that R and $R_d \in \mathbb{R}^{2\times2}$. This means that there is a total of 12 constants that can be fine-tuned in order to produce the best performance out of the proposed MPC controller. This section seeks to detail the process that was used to find the combination of weights that produced the best possible performance of the MPC controller.

Controller Fine-tuning Methodology

To fine-tune the controller's weight matrices, a grid search with some considerations about the quantities that are being considered was performed.

The first consideration that will constraint the controller fine-tuning is that every path point is equally important and therefore matrices W_f and W should be equal. The second consideration is that in an effort to simplify the controller fine-tuning, in a first stage, only the values of W_f and W were altered, while the values of matrices R and R_d were kept constant. After finding the W_f and W that produced the best overall results, the same process was performed for matrix R, with matrices W_f and W obtained with the previous grid search. The same process is then used for matrix R_d, with matrices W_f, W and R kept constant.

Finally, the third consideration regards the assumptions made while performing the grid search to find the best weight combination. For that, let it be recalled the state vector that is being considered:

$$z = [x\ y\ v_r\ \theta]^T. \tag{5.2}$$

The state vector considered can be broken down into two major parts, the part which concerns the VIENA's pose and the part which concerns VIENA's speed. This is of course a rather large simplification of the problem, as VIENA's speed clearly affects its pose, but nonetheless serves as a basis for the fine-tuning process.

By following the aforementioned premise, it was considered that the components of the weight matrices that affect VIENA's pose should be changed in tandem. There is no particular reason why VIENA's longitudinal position should be more important than its lateral position or its orientation, or any other way. In that sense, all the weights of these quantities are changed simultaneously. The weight associated with the reference speed tracking is therefore changed for every combination of the pose tracking weights.

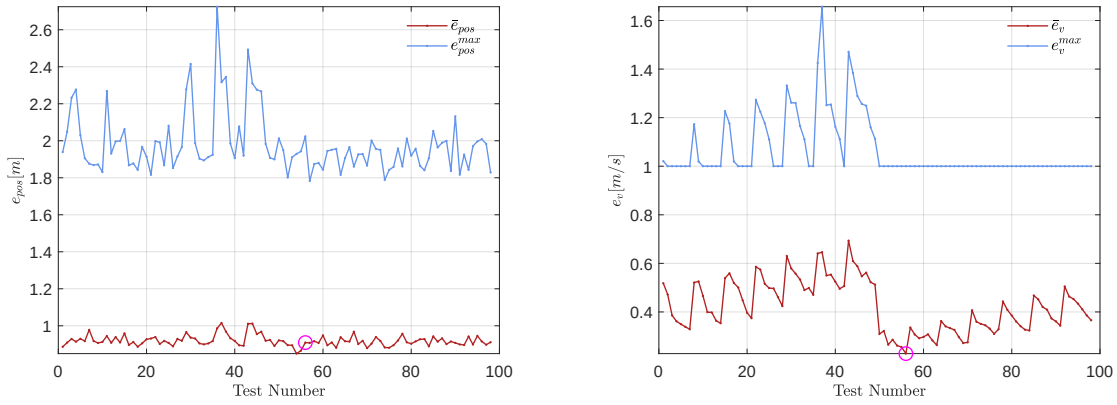
The choice to divide the fine-tuning of the weight matrices into the fine-tuning of the pose and speed tracking weights also stems from the fact that the metrics used to evaluate the controllers performance are how well the controller minimises the pose and speed errors, as seen in Table 5.2. These same metrics will be used to evaluate what are the matrices the produce the best overall results. The following sections present the results of the controller's fine-tuning.

Fine-tuning of the W and W_f Matrices

For the W and W_f matrices, the weights associated with the vehicle's pose ranged from 0.5 to 0.8, and the weights associated with the vehicle's speed ranged from 0.5 to 1.15, all in increments of 0.05. For the remainder of this fine-tuning process, the values of matrices R and R_d are

$$R = R_d = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}. \tag{5.3}$$

Figure 5.3 shows the maximum and mean errors for the pose and speed of the the vehicle, tracking through the path presented in section 5.1.



(a) (b)

Figure 5.3: Position error metrics for the different tests (a), and speed error metrics for the different tests (b).

Figure 5.3 shows that there are weight matrices that create an upper bound for the speed tracking errors. In truth, this upper bound is given by the initial difference between the vehicle's speed and the reference speed profile. The second aspect that can be observed is that even though the maximum position errors varies quite a lot, the mean position error does not, maintaining its values around 0.9 m. The final aspect which is quite important is that, for the range of values selected, a combination that produced a global minimum of the mean speed tracking error was found and is marked in magenta in Figure 5.3. As section 5.1 shows, the controller has a clear difficulty in following the reference speed profile. Figure 5.3 shows that it is possible to have a large improvement on the reference speed tracking without major loss of performance of the path-following solution. The weight matrices that produced these results are

$$W_f = W = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 1.15 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}, \quad R = R_d = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}. \tag{5.4}$$

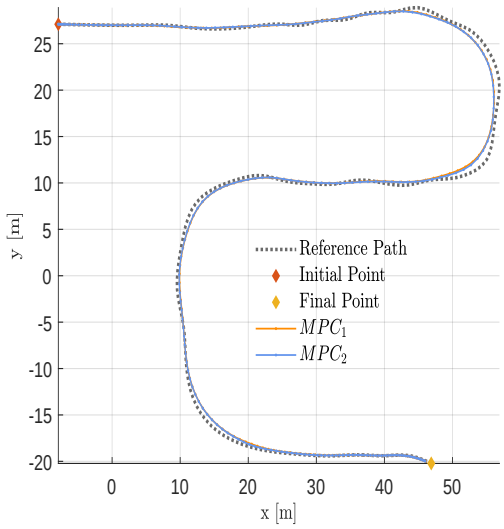
Figure 5.4, along with Tables 5.3 and 5.4 present the differences between the controller used in section 5.1, henceforth known as MPC_1, and the controller defined with the matrices present in (5.4), here termed as MPC_2.

Table 5.3: Energy and Time, Fine-tuning of the W and W_f matrices.

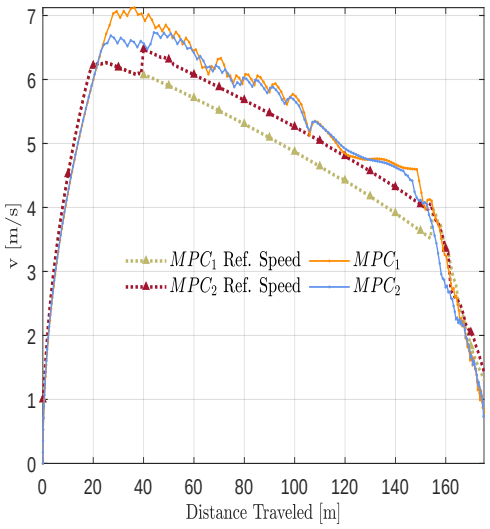
		Target Values	Real Values	$\epsilon_r[\%]$
MPC_1	E_T [kWh]	0.4658	0.5788	24.26
	t [s]	44.67	40.60	9.12
MPC_2	E_T [kWh]	0.5113	0.5580	9.131
	t [s]	44.67	41.40	7.326

Table 5.4: Error Metrics, Fine-tuning of the W and W_f matrices.

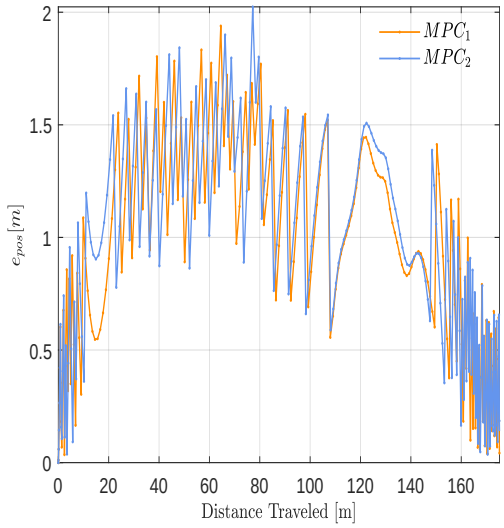
	e_{pos}^{max} [m]	e_{pos}^{min} [m]	$\bar{e}_{pos}$ [m]	e_v^{max} [m]	e_v^{min} [m/s]	$\bar{e}_v[m/s]$
MPC_1	2.077	0.011	0.8945	1.112	0.0079	0.4954
MPC_2	2.024	0.011	0.9092	1	0.0020	0.2285



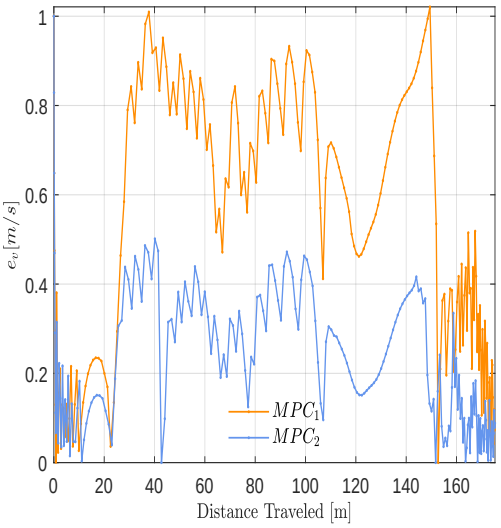
(a) Reference path and Vehicle Position



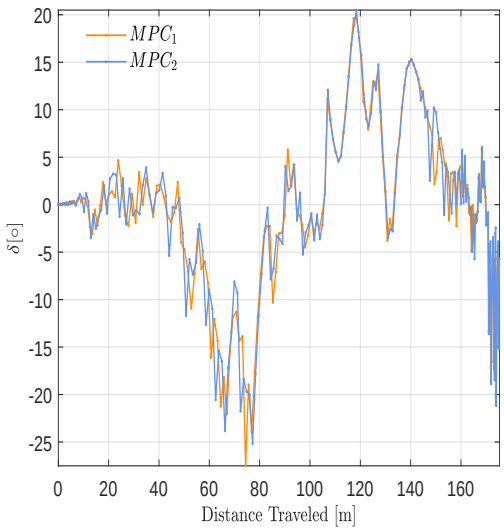
(b) Speed profile



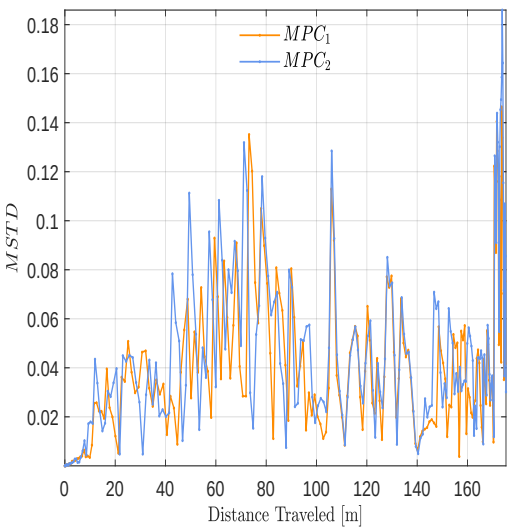
(c) Distance to Path



(d) Speed Tracking Error



(e) Steering Command



(f) Steering Moving Standard Deviation

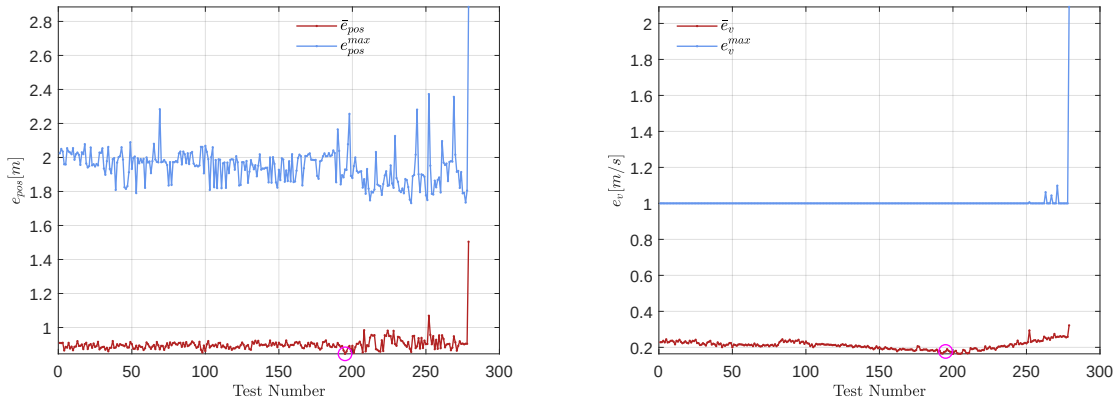
Figure 5.4: Results for the Fine-tuning of the W and W_f matrices.

As it can be seen in Figures 5.4(a) - (d) and in Tables 5.3 and 5.4, there is virtually no deterioration of the path-following capabilities of the controller but there is a noticeable increase in performance in its speed reference tracking capabilities. This in turn, results in a large improvement in the energy consumed and time elapsed, which is evidenced in Table 5.3. It is also noticeable a slight improvement in the steering signal's smoothness, as seen in Figures in 5.4(e) and 5.4(f).

With the fine-tuning of the W_f and W completed, the next step is to set their values and use the same process to fine-tune the R matrix.

Fine-tuning of the R Matrix

In contrast with matrices W and W_f, the grid search performed to fine-tune the values of the R matrix was performed in stages for three different intervals. Unlike the elements of matrices W_f and W, the intuition for the elements of matrix R was much more limited and so the fine-tuning process was performed in a much more experimental manner. The first stage of the fine-tuning process was performed between 0.01 and 0.05, in increments of 0.005. The second stage of the process was performed between 0.01 and 0.1, in increments of 0.01. The final stage of the process was performed between 0.1 and 1, in increments of 0.1. These results were then agglomerated and their maximum and mean errors for the vehicle trekking through the same path used in the section above can be seen in Figure 5.5.



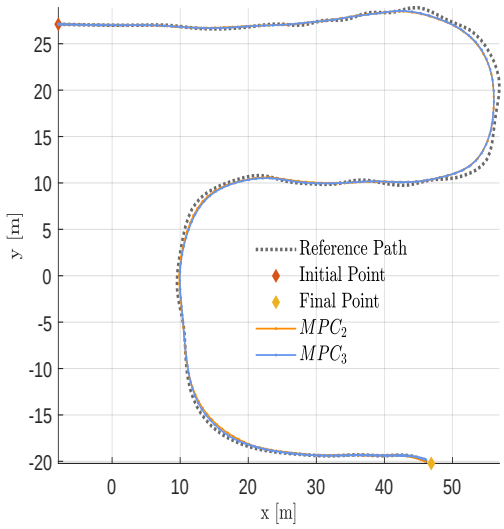
(a) (b)

Figure 5.5: Position error metrics for the different tests (a), and speed error metrics for the different tests (b).

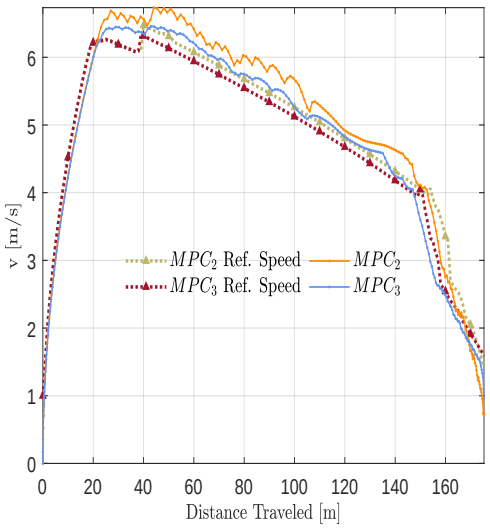
The R matrix that was considered was the one that produced the smallest mean position error. This matrix also produced a smaller mean speed error and thus it was chosen as the final R matrix. The weight matrices that produced these results are

$$W_f = W = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 1.15 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}, \quad R = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.4 \end{bmatrix}, \quad R_d = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}. \tag{5.5}$$

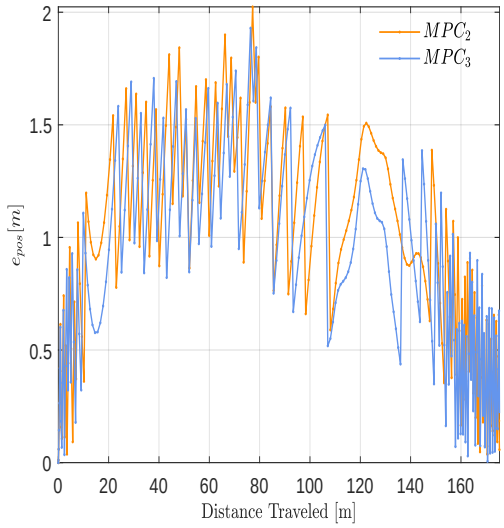
Figure 5.6 and Tables 5.5 and 5.6 compare the MPC controller defined by the matrices present in equation (5.4), known as MPC_2 with the MPC controller defined by the matrices present in equation (5.5), henceforth referred as MPC_3.



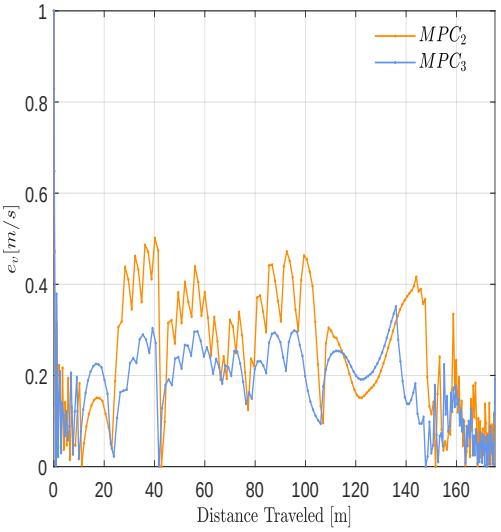
(a) Reference path and Vehicle Position



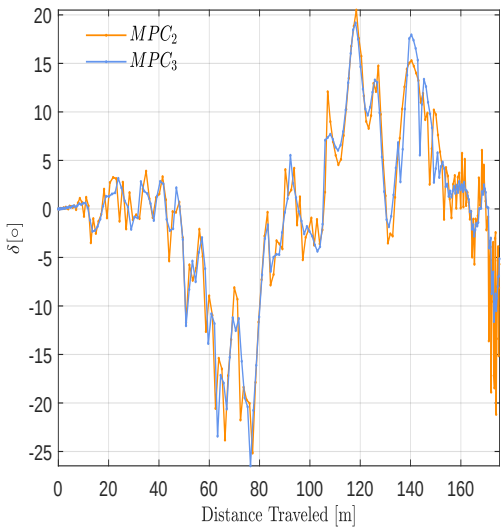
(b) Speed profile



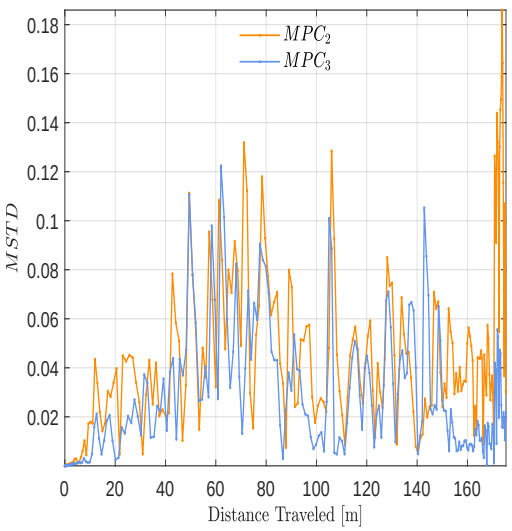
(c) Distance to Path



(d) Speed Tracking Error



(e) Steering Command



(f) Steering Moving Standard Deviation

Figure 5.6: Results for the Fine-tuning of the R matrix.

Table 5.5: Energy and Time, Fine-tuning of the R matrix.

		Target Values	Real Values	$\epsilon_r[\%]$
MPC_2	E_T [kWh]	0.5113	0.5580	9.131
	t [s]	44.67	41.40	7.326
MPC_3	E_T [kWh]	0.5049	0.5361	6.195
	t [s]	44.67	42.60	4.639

Table 5.6: Error Metrics for the fine-tuning of the R matrix.

	e_{pos}^{max} [m]	e_{pos}^{min} [m]	$\bar{e}_{pos}$ [m]	e_v^{max} [m]	e_v^{min} [m/s]	$\bar{e}_v[m/s]$
MPC_2	2.024	0.011	0.9092	1	0.0020	0.2285
MPC_3	1.930	0.054	0.8452	1	0.0022	0.1772

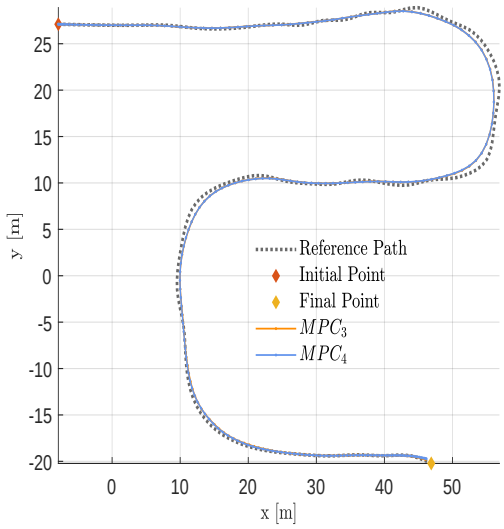
As it can be seen in Figures 5.6(a) - (d) and in Tables 5.5 and 5.6, the fine-tuning of matrix R produced a visible improvement in the MPC controller's performance. It is also noteworthy that the smoothness of the steering improved noticeably, as is evidenced in Figures 5.6(e) and 5.6(f). Continuing this process, the last matrix to be fined-tuned is the R_d matrix.

Fine-tuning of the R_d Matrix

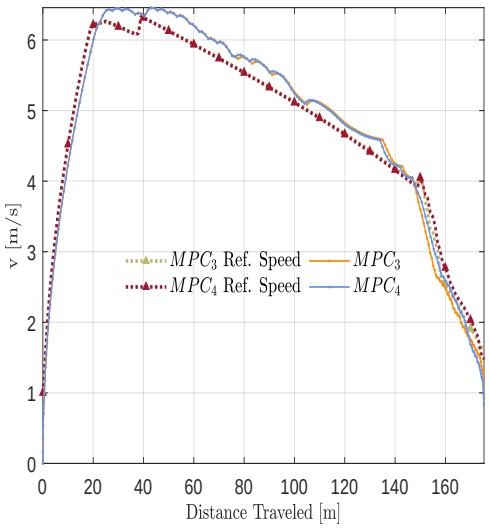
The final grid-search performed to fine-tune the weights of the R_d matrix was performed from 0.01 up to 1.5 in steps of 0.005. Like the previous fine-tuning processes, the chosen R_d matrix was the one that produced the smallest mean position error. The final set of weight matrices is therefore

$$W_f = W = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 1.15 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}, \quad R = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.4 \end{bmatrix}, \quad R_d = \begin{bmatrix} 0.01 & 0 \\ 0 & 1.11 \end{bmatrix}. \tag{5.6}$$

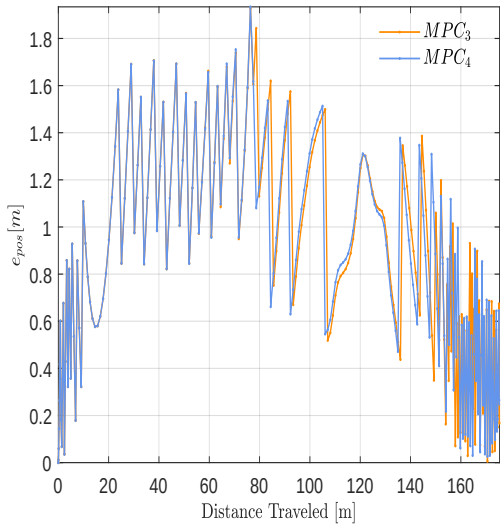
Figure 5.7 and Tables 5.7 and 5.8 present the comparison of the MPC controller defined by this final set of matrices, hence known as MPC_4 against the one defined by the matrices shown in (5.5), already termed as MPC_3.



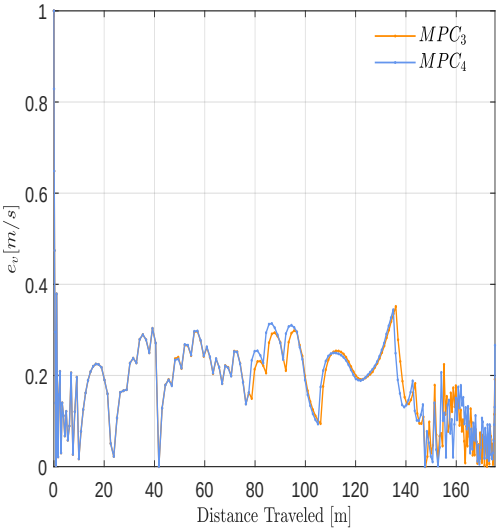
(a) Reference path and Vehicle Position



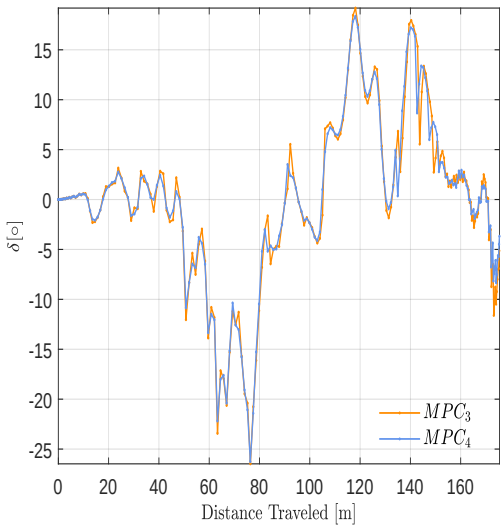
(b) Speed profile



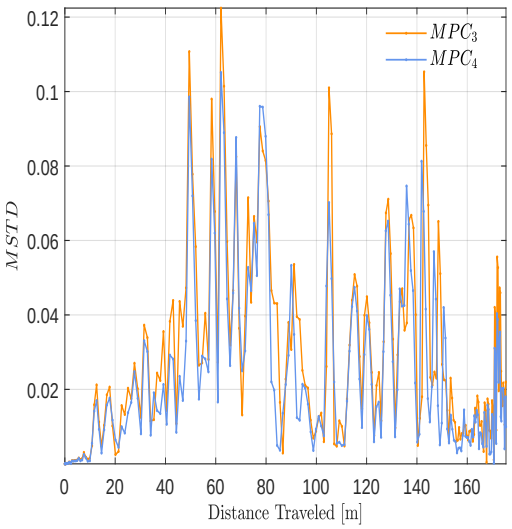
(c) Distance to Path



(d) Speed Tracking Error



(e) Steering Command



(f) Steering Moving Standard Deviation

Figure 5.7: Results for the Fine-tuning of the R_d matrix.

Table 5.7: Energy and Time, Fine-tuning of the R_d matrix.

		Target Values	Real Values	$\epsilon_r[\%]$
MPC_3	E_T [kWh]	0.5049	0.5361	6.195
	t [s]	44.67	42.60	4.639
MPC_4	E_T [kWh]	0.5028	0.5372	6.208
	t [s]	44.67	42.60	4.639

Table 5.8: Error Metrics for the fine-tuning of the R_d matrix.

	e_{pos}^{max} [m]	e_{pos}^{min} [m]	$\bar{e}_{pos}$ [m]	e_v^{max} [m]	e_v^{min} [m/s]	$\bar{e}_v[m/s]$
MPC_3	1.930	0.0054	0.8452	1	0.0022	0.1772
MPC_4	1.934	0.011	0.8401	1	0.0014	0.1794

As Figures 5.7(a) through 5.7(d) and Tables 5.7 and 5.8 suggest, the fine-tuning of the weight matrices has reached a point of minimum returns. While the mean position error improved slightly, the mean speed error deteriorated slightly, which is further reflected on the energy consumed during the path-following process. The fine-tuning of the R_d matrix did somewhat improve the steering signal's smoothness, as can be seen in Figures 5.7(e) and 5.7(f).

Considering that the path-following tasks is the priority over the reference speed tracking, the final set of matrices chosen is the one presented in equation (5.6).

All there is left to discuss is the possibility of the over-fitting of the weight matrices to solve a specific path-following problem. The matrices found are not guaranteed to produce the best global results for every input reference path and speed profile. However, one mitigates the problem by choosing typical trajectories for which the controller is tuned. This is one of the aspects that could warrant further research. To further validate the controller found in this fine-tuning process, it will be compared with the Pure Pursuit and the Front Wheel Position Based Feedback Controllers in the upcoming section.

5.3 Controller Comparison

The following sections compare the performances of three different controllers: the designed MPC controller, the Pure Pursuit Controller, and the Front Wheel Position Based Feedback Controller, henceforth known as the Stanley controller. The same metrics used in the sections above will be used to evaluate the performance of the aforementioned controllers and compare them against each other.

5.3.1 Path used for Fine-tuning the Controller

The results for the first track can be found in Figure 5.8 and on Tables 5.9 and 5.10. The analysis of Figures 5.8(a) and 5.8(c) and of Table 5.10 highlights the fact that, when it comes to the path-following problem, the proposed MPC controller is at least on par with the solutions found by the Pure Pursuit and the Stanley controllers, finding a solution in which the mean position error is the smallest. It is worth noting the interesting capability of the MPC controller of anticipating the turns along the path. This allows the MPC controller to lead VIENA to make turns on the inside whereas the Pure Pursuit and Stanley controllers are only capable of making turns on the outside. Unlike the MPC controller, the Pure Pursuit and Stanley controllers only look at one point on the path and are thus incapable of making decisions based on anything but that point.

Figure 5.8(e) shows that the steer signal computed by the proposed MPC controller is much smoother than its counterparts, highlighting the importance of the R_d weight matrix.

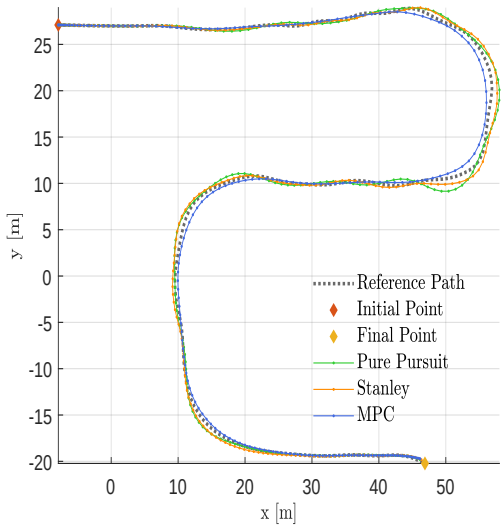
Figures 5.8(b) and 5.8(d) and Table 5.9 highlight the major limitation of the proposed MPC controller, its difficulty of following the reference speed profile, a problem which can have severe implications on the energy needed to complete the track. In this matter, the proportional controller used in the Pure Pursuit and Stanley controllers experiments is capable of computing an acceleration command that leads VIENA's speed much closer to its desired values.

Table 5.9: Energy and Time metrics for the Fine-tuning Path.

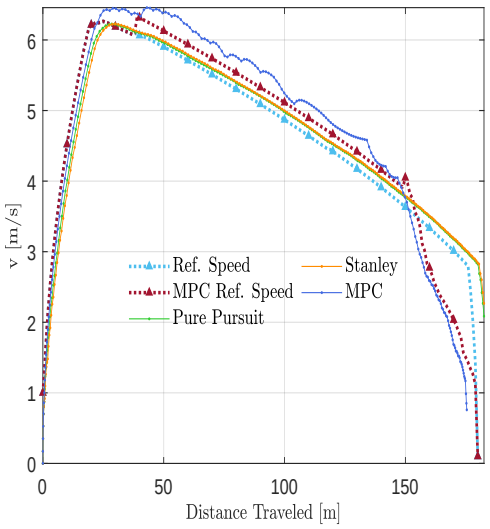
		Target Values	Real Values	$\epsilon_r[\%]$
MPC	E_T [kWh]	0.5058	0.5372	6.208
	t [s]	44.67	42.6	4.639
Pure Pursuit	E_T [kWh]	0.5416	0.5378	0.6908
	$t[s]$	43.67	42.80	1.998
Stanley	E_T [kWh]	0.5395	0.5363	0.5996
	t [s]	43.67	42.8	1.998

Table 5.10: Error Metrics for the Fine-tuning Path.

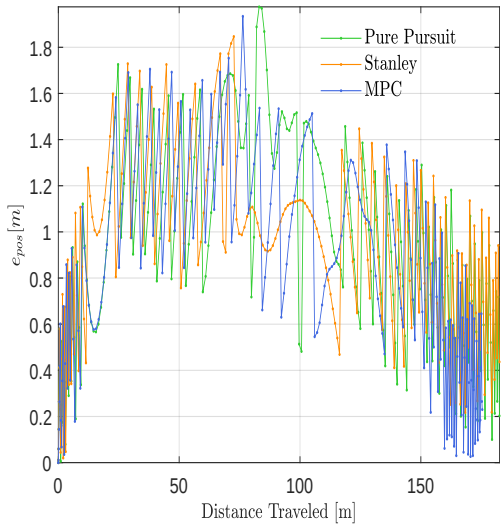
	e_{pos}^{max} [m]	e_{pos}^{min} [m]	$\bar{e}_{pos}$ [m]	e_v^{max} [m]	e_v^{min} [m/s]	$\bar{e}_v[m/s]$
MPC	1.9341	0.011	0.8401	1	0.0014	0.1794
Pure Pursuit	1.975	0.0057	0.9269	1	0.0020	0.1041
Stanley	1.774	0.011	0.9041	1.001	0.0157	0.1423



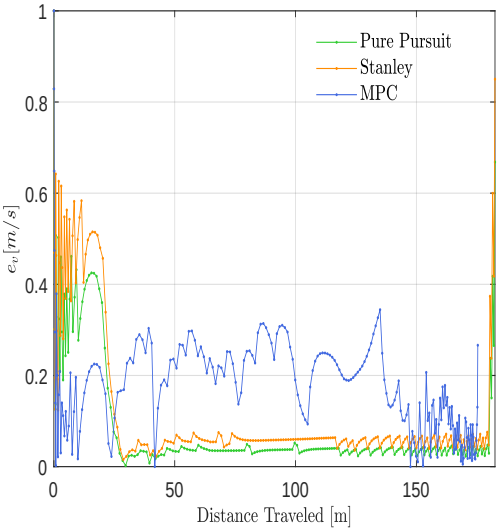
(a) Reference path and Vehicle Position



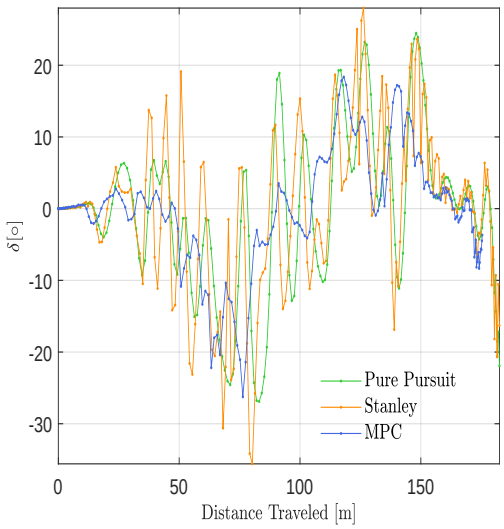
(b) Speed profile



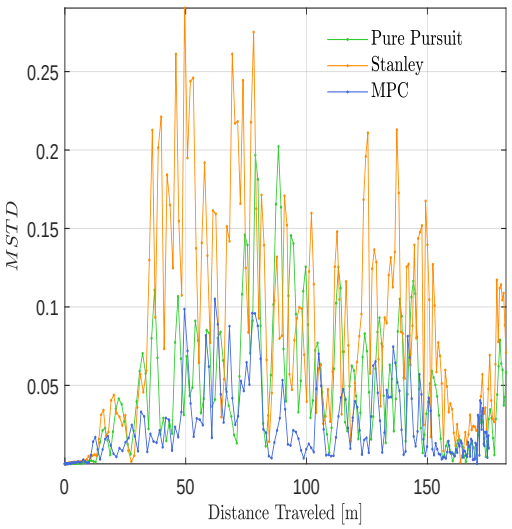
(c) Distance to Path



(d) Speed Tracking Error



(e) Steering Command



(f) Steering Moving Standard Deviation

Figure 5.8: Results for the Controller Fine-tuning Path Experiment.

5.3.2 Smooth Turns Path

The results for the second track can be seen in Figure 5.9 and in Tables 5.11 and 5.12. As it can be seen in Table 5.12, the controllers are very much on par when it comes to the path-following task. The Pure Pursuit controller has the lowest maximum position error, but the largest mean position error. The Stanley controller has the smallest mean position error. The MPC controller has both the second best maximum and mean position error, but it is worthy to point out that the differences in the mean position errors are rather small for the considered controllers. It is also relevant to point that, when compared to the Pure Pursuit and the Stanley Controllers, the proposed MPC controller produces a much smoother steering signal, as can be seen in Figure 5.9(f).

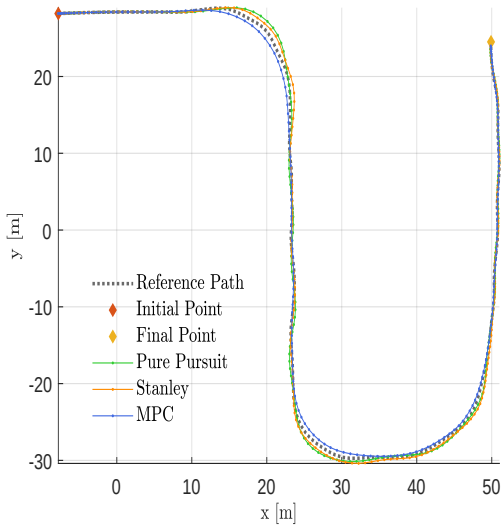
In what concerns the speed reference tracking, it is visible that while the proportional controller associated with the Pure Pursuit and Stanley controllers still outperforms the MPC controller, these differences are somewhat smaller for this path. As it can be seen in Figure 5.9(b), in this particular path, the reference speed profile for the MPC controller was not updated. This is due to the fact that the considered path is mainly composed of straight sections, with few curves, which facilitates the reference speed tracking task.

Table 5.11: Energy and Time metrics for the Smooth turns path.

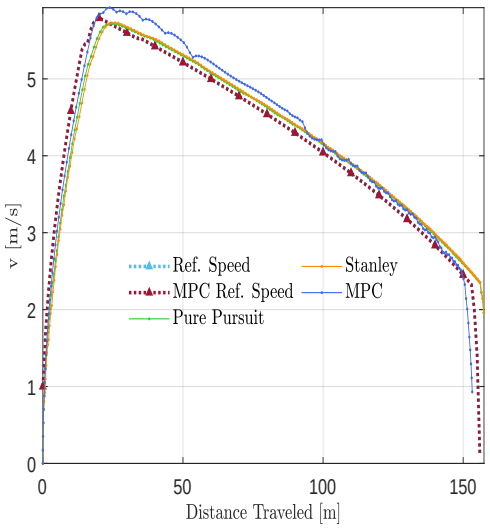
		Target Values	Real Values	$\epsilon_r[\%]$
MPC	E_T [kWh]	0.3936	0.4195	6.605
	t [s]	43.08	39.2	9.011
Pure Pursuit	E_T [kWh]	0.4216	0.4185	0.7392
	t [s]	42.08	41.00	2.572
Stanley	E_T [kWh]	0.4213	0.4185	0.6667
	t [s]	42.08	41.20	2.096

Table 5.12: Error metrics for the smooth turns path.

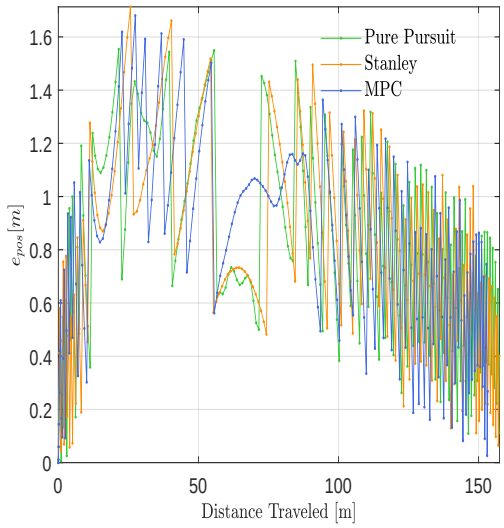
	e_{pos}^{max} [m]	e_{pos}^{min} [m]	$\bar{e}_{pos}$ [m]	e_{v}^{max} [m]	e_{v}^{min} [m/s]	$\bar{e}_v[m/s]$
MPC	1.680	0.011	0.8018	1	0.0163	0.1932
Pure Pursuit	1.555	0.0057	0.8024	1	0.0016	0.0991
Stanley	1.713	0.011	0.7813	1	0.0095	0.1402



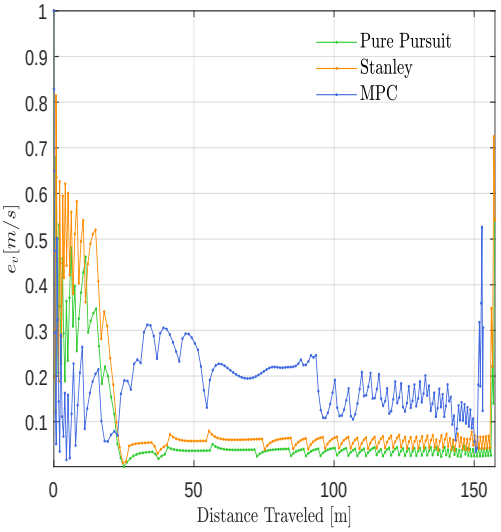
(a) Reference path and Vehicle Position



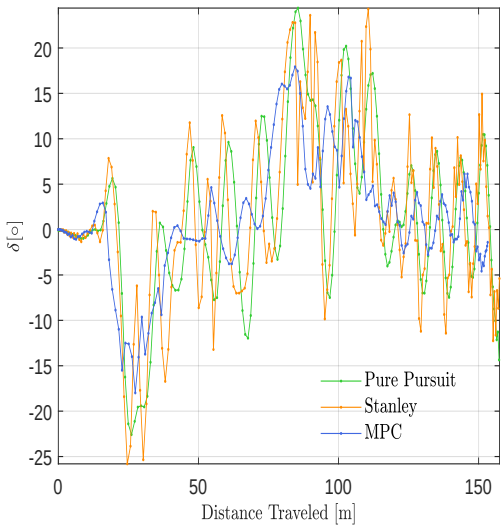
(b) Speed profile



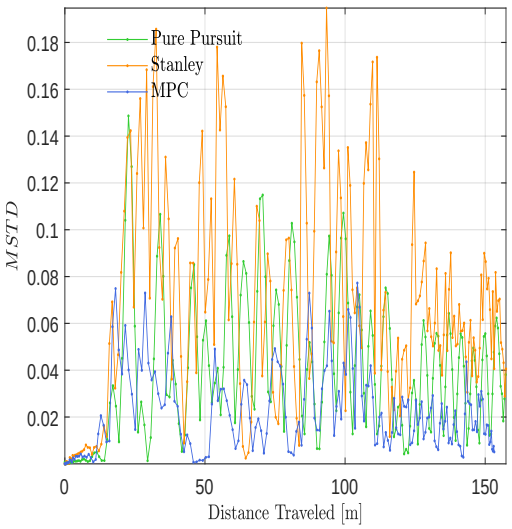
(c) Distance to Path



(d) Speed Tracking Error



(e) Steering Command



(f) Steering Moving Standard Deviation

Figure 5.9: Results for the Smooth Turns Path Experiment.

5.3.3 Sharp Turn Path

The results for the third track are presented in Figure 5.10 and in Tables 5.13 and 5.14. This path has the peculiarity of having a very sharp turn, and it is an useful case study to evaluate the performance of the controllers in the presence of a fringe case such as this one.

As it can be seen in Figure 5.10(a) and Table 5.14, while Stanley and the MPC controller are capable of dealing with the sharp turn in a satisfactory manner, the Pure Pursuit controller not only strays rather far away from the given reference path but also displays an oscillatory behaviour when trying to realign VIENA with it. These results are further reinforced in Table 5.14 in which it is visible that proposed MPC controller is capable of providing the solution with both the smallest maximum deviation from the path and the smallest average position error. It is worthy to point where the steering signal changes in a more abrupt manner is the sharp turn location, which was expected. Figure 5.10(f) validates this behaviour. While Pure Pursuit controller exhibits the smoothest steering signal, it strays to far from the path to be a considered a viable option to solve problems of this nature.

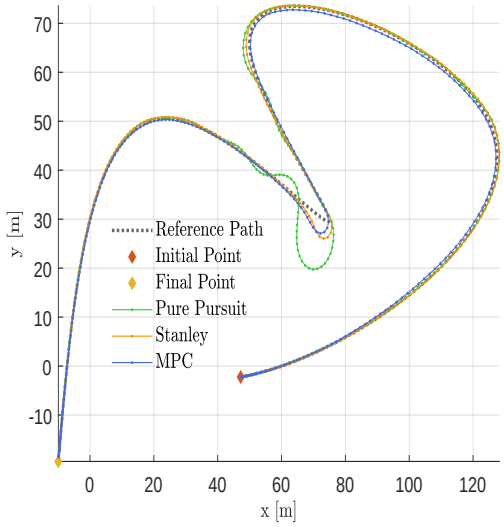
In what concerns the reference speed tracking however, the proposed MPC controller has severe difficulties in keeping VIENA's speed close to the desired values, as it can be seen in Figures 5.10(b) and 5.10(d). This has a severe implication on the energy spent while moving along this path, as it can be seen in Table 5.13.

Table 5.13: Energy and time metrics for the sharp turn path.

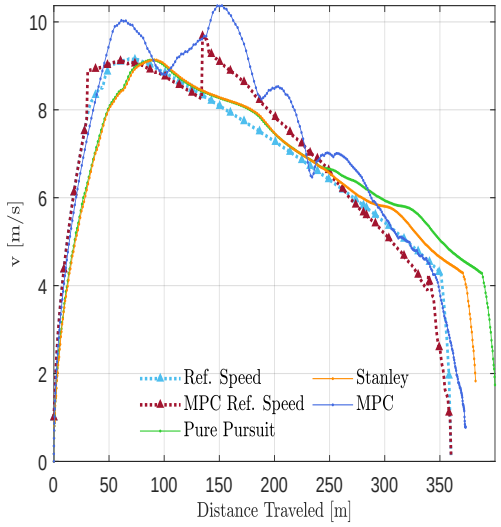
		Target Values	Real Values	$\epsilon_r[\%]$
MPC	E_T [kWh]	1.488	1.708	14.80
	t [s]	60.42	62.6	3.617
Pure Pursuit	E_T [kWh]	1.692	1.690	0.1164
	t [s]	59.42	66	11.08
Stanley	E_T [kWh]	1.620	1.618	0.1149
	t [s]	59.42	63.4	6.707

Table 5.14: Error metrics for the sharp turn path.

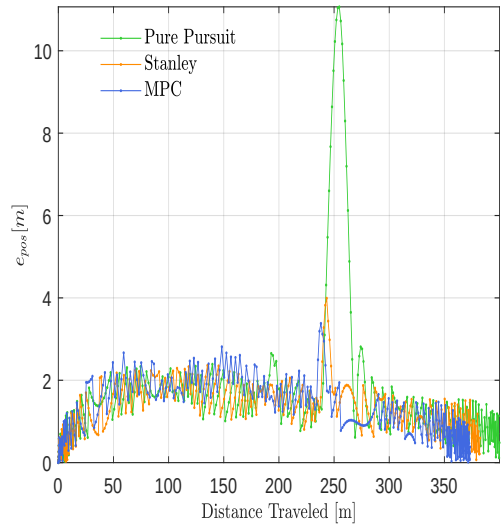
	e_{pos}^{max} [m]	e_{pos}^{min} [m]	$\bar{e}_{pos}$ [m]	e_v^{max} [m]	e_v^{min} [m/s]	$\bar{e}_v[m/s]$
MPC	3.381	0.011	1.237	1.423	0.0011	0.4444
Pure Pursuit	11.07	0.011	1.655	1	0.0041	0.0995
Stanley	3.992	0.011	1.285	1.162	0.0067	0.1542



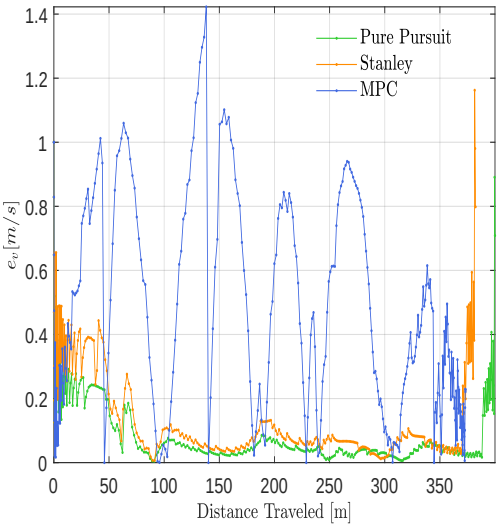
(a) Reference path and Vehicle Position



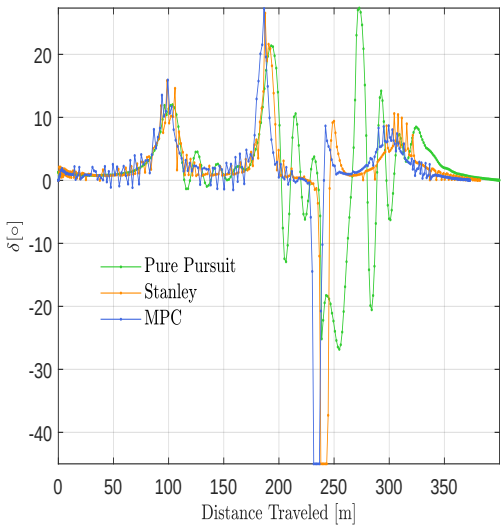
(b) Speed profile



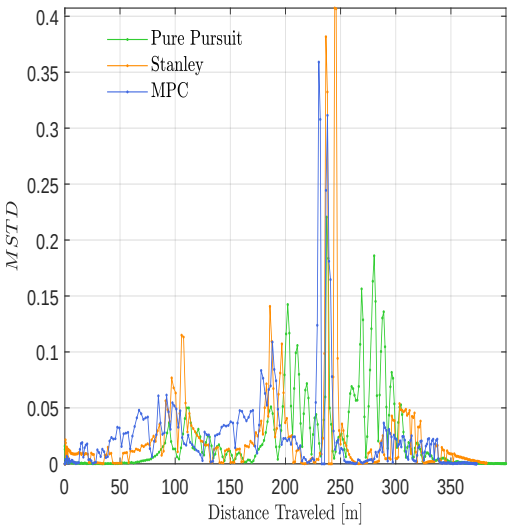
(c) Distance to Path



(d) Speed Tracking Error



(e) Steering Command



(f) Steering Moving Standard Deviation

Figure 5.10: Results for the Sharp Turn Path Experiment.

5.3.4 Start on Curved Path Point

The results of the fourth track are presented in Figure 5.11 and in Tables 5.15 and 5.16. The objective of this path is to evaluate the performance of the controllers when subjected to a narrow, spiralling path. As was the case with most of the experiments already shown, the MPC controller outperforms both the Pure Pursuit and Stanley Controllers in the path-following task, as can be seen in Table 5.16. And, as was the case with most of the previous experiments, the MPC controller's speed tracking performance is lacking when compared to the proportional controller used in the Pure Pursuit and Stanley cases, as is evidenced in Figures 5.11(b) and 5.11(d).

This experiment however, highlights a major limitation on behalf of the Pure Pursuit and Stanley controllers: the difficulty of staying close to the reference path causes them to lead VIENA to "step on" an undesired section of the path. In other words, if this path were a representation of a drivable road, the difficulty of the Pure Pursuit and Stanley controllers of making VIENA staying close to the path would cause it to be driven on the wrong side of the road.

The predictive nature of the MPC controller makes it capable of leading VIENA close to the given path so that this is not a problem. This is also in part due to the steering and acceleration commands being computed in tandem in the MPC controller, which is not the case in the Pure Pursuit and Stanley controllers. The MPC controller is capable of perceiving that, with the speed requested, it will not be possible to keep VIENA as close to the given path as desired, and thus elects to compute a steer and acceleration command that drives VIENA closer to its desired position, rather than the desired speed.

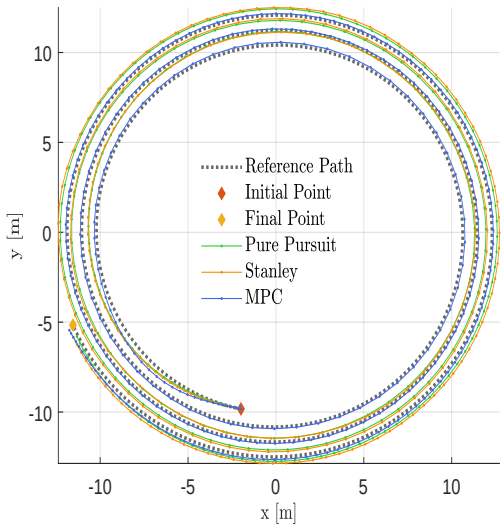
It is also possible to observe that in this case, the smoothness of the steering input of the MPC controller is sacrificed in order to achieve a better performance of the path-following task. Nonetheless, the MPC controller still manages to have a smoother signal than the Stanley controller, which is evidenced in Figures 5.11(e) and 5.11(f).

Table 5.15: Energy and time metrics for the start on a curved path.

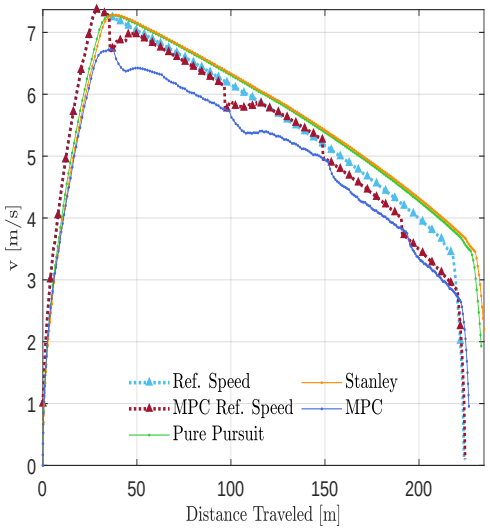
		Target Values	Real Values	$\epsilon_r[\%]$
MPC	E_T [kWh]	0.8115	0.6990	13.86
	t [s]	47.54	51.40	8.114
Pure Pursuit	E_T [kWh]	0.8027	0.8010	0.2094
	t [s]	69.74	68.6	1.63
Stanley	E_T [kWh]	0.8075	0.8056	0.2336
	t [s]	46.54	47.2	0.4875

Table 5.16: Error metrics for the start on a curved path.

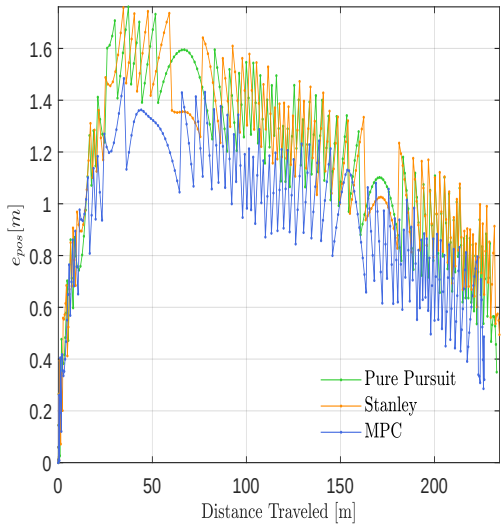
	e_{pos}^{max} [m]	e_{pos}^{min} [m]	$\bar{e}_{pos}$ [m]	e_v^{max} [m]	e_v^{min} [m/s]	$\bar{e}_v[m/s]$
MPC	1.484	0.0080	0.8914	1	0.616	0.3575
Pure Pursuit	1.761	0.011	1.060	1.165	0.0072	0.1122
Stanley	1.770	0.011	1.079	1.433	0.0017	0.1600



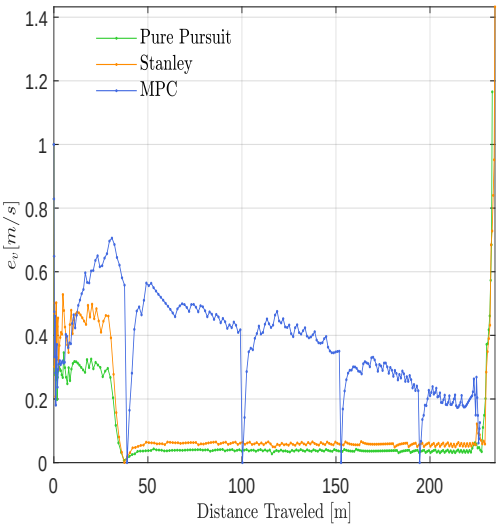
(a) Reference path and Vehicle Position



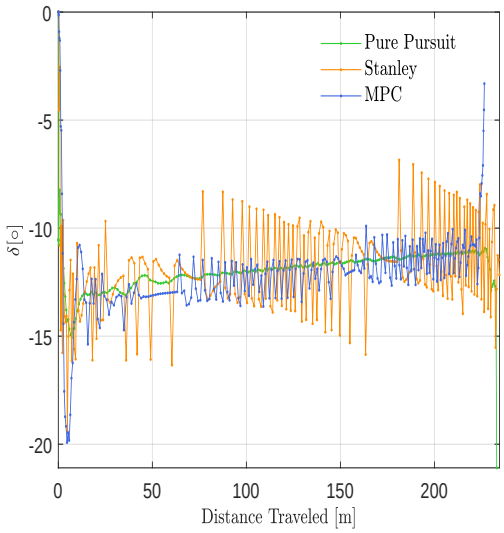
(b) Speed profile



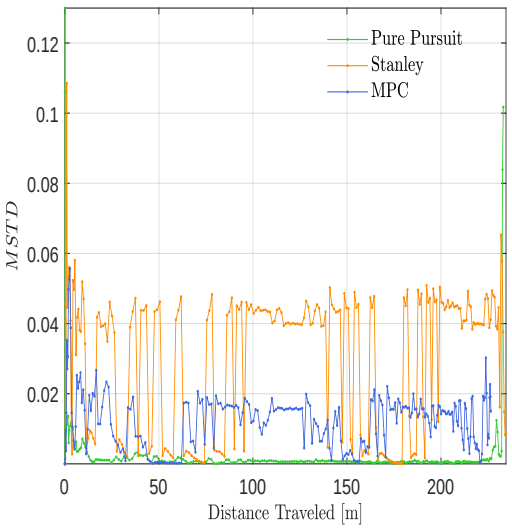
(c) Distance to Path



(d) Speed Tracking Error



(e) Steering Command



(f) Steering Moving Standard Deviation

Figure 5.11: Results for the start on a curved path point experiment.

Chapter 6

Conclusion and Future Work

In this work a Model Predictive Controller that solves the path-following problem with the least energy required was proposed. Along with the controller proposed, a model of an electric vehicle was also developed. This model characterises not only the dynamics of the vehicle's movement, but also the dynamics of its induction motor, providing an internal control technique that drives its speed to the desired reference.

As it was shown in Chapter 5, the proposed controller has several advantages over the well established Pure Pursuit and Front Wheel Based Feedback controllers. Not only is it the controller that consistently produces the smallest average distance to the path, it is also the one that is capable of doing so with the smoothest steering signal. Its capacity of computing simultaneously a steering and acceleration signal makes it capable of reducing VIENA's speed to keep it close to the provided reference path and its predictive capabilities makes it possible to have the vehicle perform turns on the inside, rather than on the outside.

In future works, the fine-tuning process of the weight matrices could be improved. A machine learning approach could provide better overall results. Moreover, focusing on computing a more realistic reference speed profile that takes into account the path's characteristics, such as its curvature, could greatly improve the speed tracking error.

Another aspect that suggests further study is the implementation of the proposed controller that decreases the computation time of the vehicle control signals. As it stands, the current MPC implementation takes around 0.3 seconds to compute the desired commands. As noted before the proposed implementation was done in Matlab. For a real-time implementation, a faster implementation needs to be considered. It is possible that implementing the proposed controller in a language such as C++ will decrease the computation time significantly.

The further development of VIENA's model is also an important step to evaluate the possibility of a real-time implementation. The development of model's that contemplate sensor dynamics, as well as lateral displacements of the vehicle is needed to evaluate the controller's robustness to uncertainty and noise.

Appendix A

Convexity, Feasibility and Stability of the OCP

A.1 Convexity of the OCP

Convexity of the OCP is of the utmost importance. On one hand, it guarantees that the OCP has an attainable solution at each time step. On the the other, a convex OCP can be solved using well known numerical methods, such a standard quadratic programming. This is of particular interest as one of the main setbacks of MPC is the computational power it requires. A convex OCP is a much faster problem to solve, making it very appealing for real time applications. For this reason, the OCP presented in equations (4.27a)-(4.27h) is by design a convex optimisation problem. The following section provides an analysis on the convexity of this OCP.

Before beginning the convexity analysis, it is important to recall the sufficient conditions that guarantee convexity of an optimisation problem. Consider the following generic optimisation problem:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) && \text{(A.1a)} \\ \text{subject to} \quad & g_i(x) \leq 0, \quad i = 0, \ldots, m, && \text{(A.1b)} \\ & h_j(x) = 0, \quad j = 0, \ldots, n && \text{(A.1c)} \end{aligned}$$

where (A.1a) denotes the problem's objective function, (A.1b) denotes the inequality constraints and (A.1c) denotes the equality constraints. The optimisation problem is convex if the cost function and the inequality constraints are convex functions and the equality constraints are affine functions.

Starting with OCP's cost function, it is quite clear that (4.27a) is a sum of functions. It is a well known property that the sum of convex functions is also a convex function. Therefore, if every function in the sum is convex, the cost function of the OCP is also convex. Let

$$\begin{aligned} & p(z_N) : \mathbb{R}^4 \longrightarrow \mathbb{R}, \\ & p(z_N) = (z_N^{ref} - z_N)^T W_f (z_N^{ref} - z_N). \end{aligned} \tag{A.2}$$

with $W_f \in \mathbb{R}^{4\times4} \succ 0$. $p(z_N)$ can be seen as a composition of two distinct functions. Let

$$p(z_N) = (f \circ g)(z_N) = f(g(z_N)) \tag{A.3}$$

with

$$\begin{aligned} g(z_N) &: \mathbb{R}^4 \longrightarrow \mathbb{R}^4, \\ g(z_N) &= z_N^{ref} - z_N \end{aligned} \tag{A.4}$$

and

$$\begin{aligned} f(Z) &: \mathbb{R}^4 \longrightarrow \mathbb{R}, \\ f(Z) &= Z^T W_f Z. \end{aligned} \tag{A.5}$$

Equation (A.4) is an affine, and therefore convex function. Equation (A.5) is a quadratic, and therefore convex function. It is a known property that the composition of a convex function with an affine map is also a convex function. Therefore $p(z_N)$ is a convex function. Next, considering the second part of the OCP's cost function, let

$$q(z_k, u_k) = \sum_{k=0}^{N-1} (z_k^{ref} - z_k)^T W (z_k^{ref} - z_k) + u_k^T R u_k + (u_k - u_{k-1})^T R_d (u_k - u_{k-1}) \tag{A.6}$$

define the function that composes the sum that is the second part of the OCP's cost function. It is clear that $q(z_k, u_k)$ is a sum of identical functions and therefore there is only the need to prove convexity of one of them. Let (A.6) be rewritten as

$$q(z_k, u_k) = \sum_{k=0}^{N-1} f_k'(z_k, u_k), \tag{A.7}$$

with

$$f_k'(z_k, u_k) = (z_k^{ref} - z_k)^T W (z_k^{ref} - z_k) + u_k^T R u_k + (u_k - u_{k-1})^T R_d (u_k - u_{k-1}). \tag{A.8}$$

If all the functions that compose its sum are convex, $f_k'(z_k, u_k)$ is also convex. Let

$$f_k'(z_k, u_k) = f_1'(z_k) + g_1'(u_k) + h_1'(u_k), \tag{A.9}$$

with

$$\begin{aligned} f_1'(z_k) &: \mathbb{R}^4 \longrightarrow \mathbb{R}, \\ f_1'(z_k) &= (z_k^{ref} - z_k)^T W (z_k^{ref} - z_k), \end{aligned} \tag{A.10}$$

$$\begin{aligned} g_1'(z_k) &: \mathbb{R}^2 \longrightarrow \mathbb{R}, \\ g_1'(z_k) &= u_k^T R u_k \end{aligned} \tag{A.11}$$

and

$$\begin{aligned} h_1'(u_k) &: \mathbb{R}^2 \longrightarrow \mathbb{R}, \\ h_1'(u_k) &= (u_k - u_{k-1})^T R_d (u_k - u_{k-1}) \end{aligned} \tag{A.12}$$

denote the functions that compose the sum that is $f_k'(z_k, u_k)$. Beginning with equation (A.10), it is quite clear that it possesses the same shape as $p(z_N)$, and is therefore also a convex function. Please note that, like W_f, $W \in$

$\mathbb{R}^{4\times4} \succ 0$. Equation (A.11) is clearly and quadratic and therefore a convex function, since $R, R_d \in \mathbb{R}^{2\times2} \succ 0$. Finally, equation (A.12) can be written as

$$h_1'(u_k) = (i_1' \circ l_1')(u_k) = i_1'(l_1'(u_k)) \tag{A.13}$$

where

$$\begin{aligned} &l_1'(u_k) : \mathbb{R}^2 \longrightarrow \mathbb{R}^2, \\ &l_1'(u_k) = u_k - u_{k-1} \end{aligned} \tag{A.14}$$

and

$$\begin{aligned} &i_1'(Z) : \mathbb{R}^2 \longrightarrow \mathbb{R}, \\ &i_1'(Z) = Z^T R_d Z. \end{aligned} \tag{A.15}$$

Using the rationale that was used in previous functions, it should be clear that $l_1'(u_k)$ is an affine map and that $i_1'(Z)$ is a quadratic function. Therefore $h_1'(u_k)$ is also a convex function. Since $f_k'(z_k, u_k)$ is a sum of convex functions, it is also a convex function and the exact same thing can be said about $q(z_k, u_k)$. This in turn, proves that the problem's cost function is a sum of convex function which makes it a convex function. Figure A.1 shows the convexity diagram of the problem's cost function.

Next, the convexity of the OCP's constraints will be analysed. Beginning with the OCP's state evolution constraints, present in (4.27c), let it be recalled that they take form

$$z_{k+1} = A(\bar{z})z_k + B(\bar{z})u_k + C(\bar{z}) \tag{A.16}$$

which can be rewritten and expanded into the following set of functions:

$$\begin{aligned} \begin{bmatrix} f_1(z_k,u_k) \\ f_2(z_k,u_k) \\ f_3(z_k,u_k) \\ f_4(z_k,u_k) \end{bmatrix} = &\begin{bmatrix} 1 & 0 & \cos(\bar{\theta})\cos(\phi)\Delta t & -\bar{v}_r \sin(\bar{\theta})\cos(\phi)\Delta t \\ 0 & 1 & \sin(\bar{\theta})\cos(\phi)\Delta t & \bar{v}_r \cos(\bar{\theta})\cos(\phi)\Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ v_{rk} \\ \theta_k \end{bmatrix} + \\ &\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \Delta t & 0 \\ 0 & \frac{\bar{v}_r \cos(\phi)\Delta t}{L} \end{bmatrix} \begin{bmatrix} a_k \\ \delta_k \end{bmatrix} + \begin{bmatrix} \bar{v}_r \bar{\theta} \sin(\bar{\theta})\cos(\phi)\Delta t \\ -\bar{v}_r \bar{\theta} \cos(\bar{\theta})\cos(\phi)\Delta t \\ 0 \\ 0 \end{bmatrix}. \end{aligned} \tag{A.17}$$

As already stated, for the optimisation problem to be convex, the equality constraints must be affine functions. Beginning with $f_1(z_k, u_k)$, this function can be written as

$$f_1(z_k, u_k) = x_k + \cos(\bar{\theta})cos(\phi)\Delta t v_{rk} - \bar{v}_r \sin(\bar{\theta})\cos(\phi)\Delta t \theta_k + \bar{v}_r \bar{\theta} \sin(\bar{\theta})\cos(\phi)\Delta t. \tag{A.18}$$

Please recall that every element inside the matrices is a constant and therefore (A.18) can be rewritten as

$$\begin{aligned} &f_1(z_k, u_k) = x_k + k_1 v_{rk} - k_2 \theta_k + k_3 \Leftrightarrow \\ &\Leftrightarrow f_1(z_k, u_k) = \begin{bmatrix} 1 & 0 & k_1 & -k_2 \end{bmatrix} z_k + k_3 \end{aligned} \tag{A.19}$$

which is clearly an affine function of the form

$$f(x) = ax + b. \tag{A.20}$$

By applying the same reasoning to the remaining functions present in (A.17) it should be clear that all are affine. Therefore all of the state evolution constraints are affine functions. Since the relation between the state and the input is linear (affine), it is also convex, which corroborates the convexity of (A.2) and (A.10). Moving on to the speed constraint present in (4.27d), it should be noted that this constraint can be separated into two separate constraints:

$$\begin{aligned} v_{r_{min}} \leq v_{rk} \leq v_{r_{max}} \Leftrightarrow \begin{cases} v_{rk} \geq v_{r_{min}} \\ v_{rk} \leq v_{r_{max}} \end{cases} \Leftrightarrow \begin{cases} v_{rk} - v_{r_{min}} \geq 0 \\ v_{rk} - v_{r_{max}} \leq 0 \end{cases} \Leftrightarrow \\ \Leftrightarrow \begin{cases} -v_{rk} + v_{r_{min}} \leq 0 \\ v_{rk} - v_{r_{max}} \leq 0. \end{cases} \Leftrightarrow \begin{cases} f_1(v_{rk}) \leq 0 \\ f_2(v_{rk}) \leq 0. \end{cases} \end{aligned} \tag{A.21}$$

It should be clear that equations $f_{1,2}(v_{rk})$ are both affine and therefore convex equations, which upholds the conditions that the equality constraints are affine and that the inequality constraints are convex. Please note that the exact same reasoning used to prove the convexity of these constraints can be used to prove the convexity of constraints (4.27e) and (4.27f). Finally, constraint (4.27g) can be rewritten as a composition of two functions:

$$\begin{aligned} &|\delta_k - \delta_{k-1}| \leq \Delta\delta_{max} \Leftrightarrow \\ &\Leftrightarrow |\delta_k - \delta_{k-1}| - \Delta\delta_{max} \leq 0 \Leftrightarrow \\ &\Leftrightarrow f_1(\delta_k) \leq 0 \end{aligned} \tag{A.22}$$

with

$$f_1(\delta_k) = h_1 \circ g_1 \tag{A.23}$$

and

$$\begin{aligned} &g_1(\delta_k) : \mathbb{R} \longrightarrow \mathbb{R}, \\ &g_1(\delta_k) = \delta_k - \delta_{k-1}, \end{aligned} \tag{A.24}$$

$$\begin{aligned} &h_1(Z) : \mathbb{R} \longrightarrow \mathbb{R}, \\ &h_1(Z) = |Z| - \Delta\delta_{max}. \end{aligned} \tag{A.25}$$

It should be clear that $g_1(\delta_k)$ is an affine function and that $h_1(Z)$ is a norm minus a constant, which has no effect on the convexity of the function. It is well known property that all norms are convex. $f_1(\delta_k)$ is therefore a composition of convex function and an affine map and is therefore convex.

This means that all constraints of the inequality constraints of the OCP are convex and that all equality constraints of the OCP are affine. Since it has already been shown that the OCP's cost function is convex, it can be stated that the OCP is convex. MPC convexity is further explored in [26].

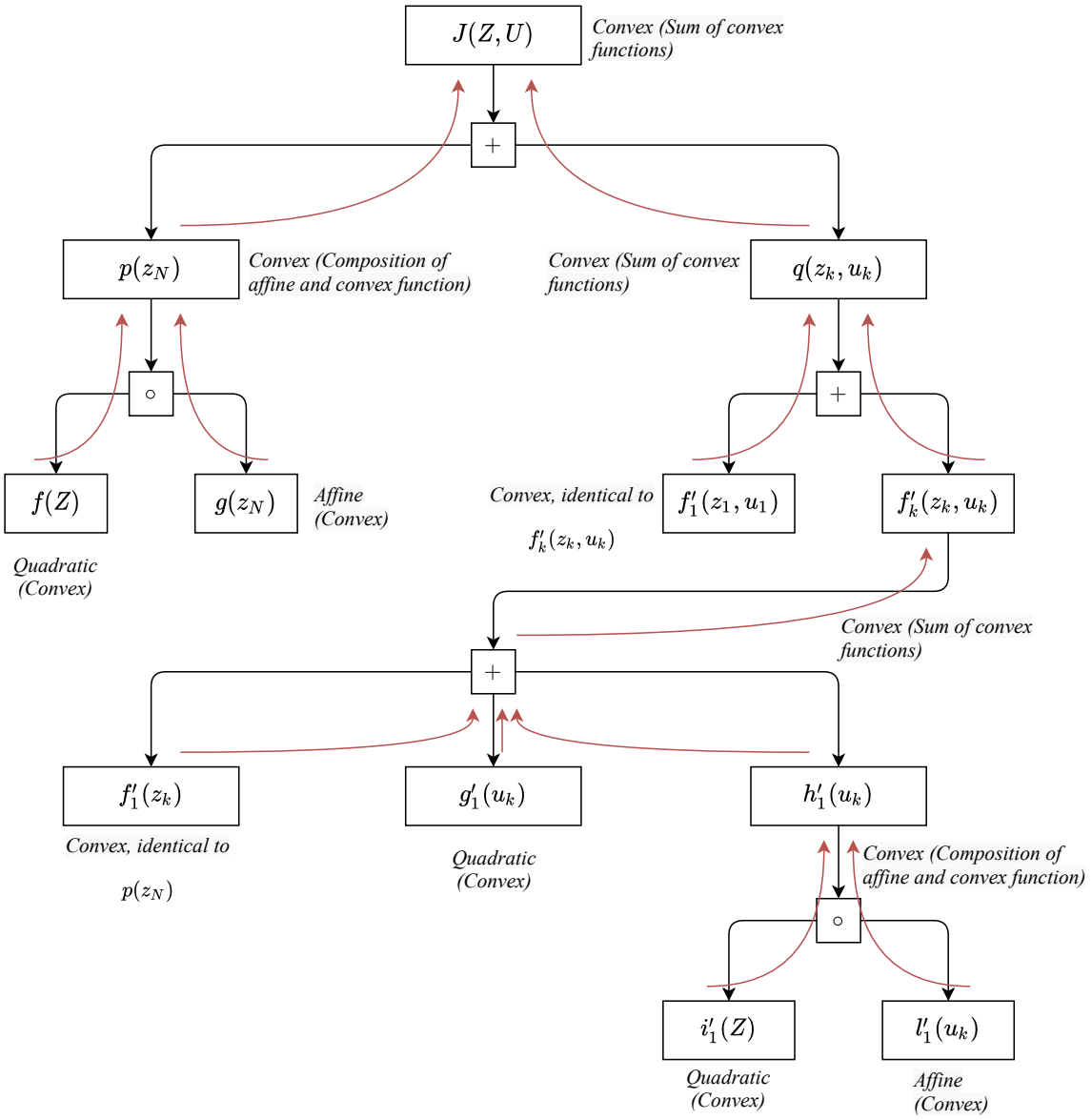


Figure A.1: Convexity Diagram of the problem's cost function.

A.2 Feasibility and Stability of the OCP

Guaranteeing the stability and feasibility of the proposed MPC is a non-trivial subject. In [27], conditions for which MPC fails to converge to the correct target are provided. In [28] and [29] Linear MPC problems are presented, with guarantees about their stability and persistent feasibility. It is worthy to point out that if the considered MPC is stable, it forcefully also is persistently feasible. In [19], a linear MPC problem alike the one presented herein is given some guarantees of stability.

It should be stated that the references above tackle problems in which the internal representation of the system remains unchanged throughout the different optimisation problems. In the presented case, since the space-state matrices that model the vehicle behaviour change along the optimisation problems that are solved. In [30] a nonlinear predictive control for set point families is presented, which considers a pseudolinearization of the system, mush alike the one presented in this work, and a parametrization of the set points. The stability is ensured thanks to the quasi-infinite nonlinear MPC strategy, but the solution of the tracking problem is not considered.

Appendix B

Tables of Constants

Table B.1: VIENA Parameters.

VIENA Parameters	**Values**
Wheelbase (L) [m]	2.7
Frontal Area (A_f) [m^2]	2.14
Gravitational Acceleration (g) [$\mathrm{m/s}^2$]	9.8
Weight (m) [kg]	900
Wheel Radius (r_w) [m]	0.165
Aerodynamic Drag Coefficient (C_d)	0.33
Rolling Resistance Coefficient (C_{rr})	0.01
Gear Ratio (g_r)	8
Wheel Inertia (I_w) [kg m^2]	0.25
Motor's rotor Inertia (I_m) [kg m^2]	0.0025
Air Density (ρ) [$\mathrm{kg/m}^{-3}$]	1.225
Wind Speed (v_w) [m/s]	0

Table B.2: Motor *dq* model Parameters

Motor *dq* Model Parameters	**Values**
Pairs of Poles (n_{pp})	2
Stator Resistance (R_s) [Ω]	8.56×10^{-3}
Rotor Resistance (R_t) [Ω]	2.5×10^{-3}
Stator Leakage Inductance (l_{σ_s}) [H]	0.06292×10^{-3}
Rotor Leakage Inductance (l_{σ_s}) [H]	0.06709×10^{-3}
Mutual Inductance Coefficient (L_m) [H]	1.0122×10^{-3}
Motors no Load Current (I_{mb}) [A]	160

Table B.3: FOC Parameters

FOC Parameters		**Values**
Speed Controller	K_P	10
	K_I	50
Current Controller	K_P	5
	K_I	50

Table B.4: MPC Parameters

MPC Parameters	**Values**
Δt [s]	0.2
K	4
Δu_{min}	0.1
N	5
$v_{r_{min}}$ [m/s]	-5.56
$v_{r_{max}}$ [m/s]	15.28
δ_{min} [rad]	$-\frac{\pi}{4}$
δ_{max} [rad]	$\frac{\pi}{4}$
a_{min} [m/s^2]	-1
a_{max} [m/s^2]	1
$\Delta\delta_{max}$ [rad/s]	$\frac{\pi}{12}$

Table B.5: Speed Profile Optimisation Problem Parameters

Speed Profile Optimisation Problem Parameters	**Values**
α	1000
γ	1.2

Bibliography

[1] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7490340

[2] M. Park, S. Lee, and W. Han, "Development of steering control system for autonomous vehicle using geometry-based path tracking algorithm," *ETRI Journal*, vol. 37, 05 2015. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.4218/etrij.15.0114.0123

[3] M. Arnold, R. Negenborn, G. Andersson, and B. De Schutter, "Multi-area predictive control for combined electricity and natural gas systems," *2009 European Control Conference, ECC 2009*, 03 2015. [Online]. Available: https://ieeexplore.ieee.org/document/7074603

[4] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "Combined speed and steering control in high-speed autonomous ground vehicles for obstacle avoidance using model predictive control," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 10, pp. 8746–8763, 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7932889

[5] M. I. Ribeiro and P. Lima, "Guidance methods for wheeled mobile robots," Lecture Notes, 2002.

[6] B. Tibério, "An online filter study for inertial properties estimation based on low-cost sensors." 2017. [Online]. Available: https://www.semanticscholar.org/paper/An-online-filter-study-for-inertial-properties-on-Almeida/4caaf8b97c040a88928a35ea721ace33a7ce0802

[7] J. Fernandes, F. Silva, and B. Tibério, "Electrical drives and electrical vehicles - *dq* model for the induction motor," Lecture Notes, 2020.

[8] E. R. S. Observatory. Annual accident report 2018. [Online]. Available: https://ec.europa.eu/transport/road_safety/sites/roadsafety/files/pdf/statistics/dacota/asr2018.pdf

[9] E. Brevio and S. Meder. 2018 road safety statistics: what is behind the figures? [Online]. Available: https://ec.europa.eu/commission/presscorner/detail/en/MEMO_19_1990

[10] NHTSA. Traffic safety facts. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812456

[11] O.-R. A. D. committee. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. [Online]. Available: https://saemobilus.sae.org/content/J3016_201806

[12] R. Carona, A. Aguiar, and J. Gaspar, "Control of unicycle type robots - tracking, path following and point stabilization," *IV Jornadas de Engenharia Electrotécnica e Telecomunicações e de Computadores*, pp. 180–185, 11 2008.

[13] C. Hu, Z. Wang, H. Taghavifar, J. Na, Y. Qin, J. Guo, and C. Wei, "MME-EKF-based path-tracking control of autonomous vehicles considering input saturation," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 6, pp. 5246–5259, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8675462

[14] H. Wang, B. Liu, X. Ping, and Q. An, "Path tracking control for autonomous vehicles based on an improved MPC," *IEEE Access*, vol. 7, pp. 161 064–161 073, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8854189

[15] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, *Stanley: The Robot That Won the DARPA Grand Challenge*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1–43. [Online]. Available: https://doi.org/10.1007/978-3-540-73429-1_1

[16] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, "A stable tracking control method for an autonomous mobile robot," in *Proceedings., IEEE International Conference on Robotics and Automation*, 1990, pp. 384–389 vol.1. [Online]. Available: https://ieeexplore.ieee.org/document/126006

[17] C. C. de Wit, H. Khennouf, C. Samson, and O. J. Sordalen, *Nonlinear Control Design For Mobile Robots*. World Scientific, 1994, pp. 121–156. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/9789814354301_0005

[18] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017. [Online]. Available: https://dl.acm.org/doi/book/10.5555/3164811

[19] C. Jones and M. Zeilinger, "Theory in model predictive control: Constraint satisfaction and stability," Lecture Notes. [Online]. Available: https://www.uiam.sk/pc11/data/workshops/mpc/MPC_PC11_Lecture1.pdf

[20] H. Febbo, J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "Moving obstacle avoidance for large, high-speed autonomous ground vehicles," in *2017 American Control Conference (ACC)*, 2017, pp. 5568–5573. [Online]. Available: https://ieeexplore.ieee.org/document/7963821

[21] E. Velenis and P. Tsiotras, "Optimal velocity profile generation for given acceleration limits: theoretical analysis," in *Proceedings of the 2005, American Control Conference, 2005.*, 2005, pp. 1478–1483 vol. 2. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1470174

[22] A. Sciarretta, G. De Nunzio, and L. L. Ojeda, "Optimal ecodriving control: Energy-efficient driving of road vehicles as an optimal control problem," *IEEE Control Systems Magazine*, vol. 35, no. 5, pp. 71–90, 2015. [Online]. Available: https://ieeexplore.ieee.org/document/7265166

[23] F. Ding and H. Jin, "On the optimal speed profile for eco-driving on curved roads," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 12, pp. 4000–4010, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8307449

[24] K. M. So, P. Gruber, D. Tavernini, A. E. H. Karci, A. Sorniotti, and T. Motaln, "On the optimal speed profile for electric vehicles," *IEEE Access*, vol. 8, pp. 78 504–78 518, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9045962

[25] A. Sakai. Path tracking. [Online]. Available: https://pythonrobotics.readthedocs.io/en/latest/modules/path_tracking.html

[26] B. Lautenschlager, K. Kruppa, and G. Lichtenberg, “Convexity properties of the model predictive control problem for subclasses of multilinear time-invariant systems,” *IFAC-PapersOnLine*, vol. 48, no. 23, pp. 148–153, 2015, 5th IFAC Conference on Nonlinear Model Predictive Control NMPC 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896315025598

[27] L. Shead, K. Muske, and J. Rossiter, “Conditions for which MPC fails to converge to the correct target,” *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 6968–6973, 2008, 17th IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667016400650

[28] A. Ferramosca, D. Limon, I. Alvarado, T. Alamo, and E. Camacho, “MPC for tracking of constrained nonlinear systems,” *Proceedings of the IEEE Conference on Decision and Control*, pp. 7978–7983, 12 2009. [Online]. Available: https://www.researchgate.net/publication/221041978_MPC_for_tracking_of_constrained_nonlinear_systems

[29] D. Limon, M. Pereira, D. Muñoz de la Peña, T. Alamo, C. N. Jones, and M. N. Zeilinger, “MPC for tracking periodic references,” *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 1123–1128, 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7172461

[30] R. Findeisen, H. Chen, and F. Allgöwer, “Nonlinear predictive control for setpoint families,” *American Control Conference, 2000. Proceedings of the 2000*, vol. 1, pp. 260 – 264 vol.1, 10 2000. [Online]. Available: https://ieeexplore.ieee.org/document/878860