UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

CENTRO TECNOLÓGICO

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

RÂNIK GUIDOLINI

A Neural-Based Model Predictive Control to Tackle Steering Delay of the IARA Autonomous Car

Vitória, ES
September 4, 2017

RÂNIK GUIDOLINI

A Neural-Based Model Predictive Control to Tackle Steering Delay of the IARA Autonomous Car

Thesis submitted to the *Programa de Pós-Graduação em Informática* of *Centro Tecnológ*ico of *Universidade Federal do Espírito Santo*, in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Advisor: Prof. Dr. Claudine Badue
Co-advisor: Prof. Dr. Alberto Ferreira De Souza

Vitória, ES
September 4, 2017

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Setorial Tecnológica,
Universidade Federal do Espírito Santo, ES, Brasil)

Guidolini, Rânik, 1991-
G948n A neural-based model predictive control to tackle steering delay of the IARA autonomous car / Rânik Guidolini. – 2017.
64 f. : il.

Orientador: Claudine Badue.
Coorientador: Alberto Ferreira de Souza.
Dissertação (Mestrado em Informática) – Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Controle preditivo. 2. Veículos autônomos. 3. Redes neurais (Computação). 4. Controladores PID. I. Badue, Claudine. II. Souza, Alberto Ferreira de. III. Universidade Federal do Espírito Santo. Centro Tecnológico. IV. Título

CDU: 004

A Neural-Based Model Predictive Control to Tackle Steering Delay of the IARA Autonomous Car

Rânik Guidolini

Dissertação submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Mestre em Informática.

Aprovada em 04 de setembro de 2017:

Profª. Drª. Claudine Santos Badue Gonçalves
Orientador

Prof. Dr. Thiago Oliveira dos Santos
Membro Interno

Prof. Dr. Denis Fernando Wolf
Membro Externo

O julgamento dessa dissertação foi realizado com a participação por meio de videoconferência do membro externo o Prof. Dr. Denis Fernando Wolf seguindo as normas prescritas na portaria normativa no. 1/2016. Desse modo, a assinatura do membro externo é representada neste documento pela respectiva assinatura do presidente da comissão julgadora, Profª. Drª. Claudine Santos Badue Gonçalves.

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
Vitória-ES, 04 de setembro de 2017

ACKNOLEDGEMENTS

First, I would like to thank God, for the wisdom, strength, protection and comfort in all moments of my life.

Secondly, I would like to thank my family, my parents Domingas and Fernandes, for the support and all sacrifices that helped me to reach this achievement; my brothers Rodrigo and Rickson, for the friendship and support; and my fiancée Karinnie, for the support, motivation and comprehension in all times I had to work over 3A.M. and on weekends.

I would like to thank my advisor Prof. Dr. Claudine Badue and co-advisor Prof. Dr. Alberto Ferreira De Souza, for the advices, support, guidance and help to perform this research.

I would like to thank all my friends of LCAD, for the friendship, support, good ideas and good discussions, mainly on the walk to have coffee in the canteen.

I would like to thank Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq, Fundação de Amparo à Pesquisa do Espírito Santo – FAPES and Coordenadoria de Aperfeiçoamento de Pessoal de Nível Superior – CAPES, for their financial support to this research work.

RESUMO

Neste trabalho, propomos uma abordagem de Controle Preditivo Baseado em Modelo Neural (*Neural Based Model Predictive Control* - N-MPC) para lidar com atrasos na planta de direção de carros autônomos. Examinamos a abordagem N-MPC como uma alternativa para a implementação do subsistema de controle de direção da *Intelligent and Autonomous Robotic Automobile* (IARA). Para isso, comparamos a solução padrão, baseada na abordagem de controle Proporcional Integral Derivativo (PID), com a abordagem N-MPC. O subsistema de controle de direção PID funciona bem na IARA para velocidades de até 25 km/h. No entanto, acima desta velocidade, atrasos na Planta de Direção da IARA são muito elevados para permitir uma operação adequada usando uma abordagem PID. Modelamos a Planta de Direção da IARA usando uma rede neural e empregamos esse modelo neural na abordagem N-MPC. A abordagem N-MPC superou a abordagem PID reduzindo o impacto de atrasos na Planta de Direção de IARA e permitindo a operação autônoma da IARA em velocidades de até 37 km/h – um aumento de 48% na velocidade máxima estável.

ABSTRACT

In this work, we propose a Neural Based Model Predictive Control (N-MPC) approach to tackle delays in the steering plant of autonomous cars. We examined the N-MPC approach as an alternative for the implementation of the Intelligent and Autonomous Robotic Automobile (IARA) steering control subsystem. For that, we compared the standard solution, based on the Proportional Integral Derivative (PID) control approach, with the N-MPC approach. The PID steering control subsystem works well in IARA for speeds of up to 25 km/h. However, above this speed, IARA's Steering Plant delays are too high to allow proper operation with a PID approach. We tried and modeled the IARA's Steering Plant using a neural network and employed this neural model in the N-MPC approach. The N-MPC approach outperformed the PID approach by reducing the impact of IARA's Steering Plant delays and allowing the autonomous operation of IARA at speeds of up to 37 km/h – an increase of 48% in the maximum stable speed.

Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABREVATIONS AND ACRONYMS

N-MPC	Neural Based Model Predictive Control
IARA	Intelligent and Autonomous Robotic Automobile
PID	Proportional Integral Derivative
ADAS	Advanced Driver Assistance Systems
DARPA	Defense Advanced Research Projects Agency
LCAD	*Laboratório De Computação de Alto Desempenho* (High Performance Computing Laboratory)
DI	*Departamento de Informática* (Department of Informatics)
UFES	*Universidade Federal do Espírito Santo* (Federal University of Espírito Santo)
MIMO	Multiple Input Multiple Output Systems
FIR	Finite Impulse Response
AOC	Arctangent of The Curvature
N-SPM	Neural-Based Steering Plant Model
GA	Genetic Algorithms
RMSE	Root Mean Square Error
LIDAR	Light Detection and Ranging
CARMEN	Carnegie Mellon Robot Navigation Toolkit
IPC	Inter Process Communication

1 INTRODUCTION

Advanced driver assistance systems (ADAS) and autonomous cars have been a major source of research since the two editions of the competitions promoted by the Defense Advanced Research Projects Agency (DARPA) – the Urban Challenge in 2005 and the Grand Challenge in 2008.

To allow autonomous navigation of a car around a structured environment, such as paved roads of cities, one has to provide a trajectory that takes into consideration the desired goal, obstacles in the way to the goal and guidelines (such as "keep inside the lane") that would allow the selection of an optimal or suboptimal path to the goal [KAT15]. A trajectory produced this way can then be send down throughout the autonomous car control pipeline for execution.

Several alternatives exist on how to design the autonomous car control pipeline, which range from the full integration of the motion planning subsystem with the control subsystem [GOT16] to the division of the pipeline in several stages [CAR16]. We have developed an autonomous car, called Intelligent and Autonomous Robotic Automobile (IARA), which is illustrated in Fig. 1. IARA has a four-level control pipeline composed of: (i) a path planner subsystem; (ii) a motion planning subsystem [CAR16], (iii) an obstacle avoidance subsystem [GUI16] and (iv) a control subsystem.

The path planning subsystem computes a path from the current car's pose (position and orientation) to a desired goal pose, which obeys restrictions imposed by the road traffic regulation and avoids obstacles. The path is a sequence of consecutive states, each one comprised of pose and velocity. The motion planning subsystem computes a trajectory from the current car's state to the goal state, which follows the path and avoids obstacles. The trajectory is a sequence of control commands, each one comprised of steering wheel angle, linear velocity and execution time. The obstacle avoidance subsystem simulates the trajectory and decreases the linear velocity commands of the trajectory, in case it is necessary to avoid an accident. The control subsystem computes the control commands that will actuate to make the car achieve each one of the trajectory's steering wheel angle and linear velocity within the execution time.

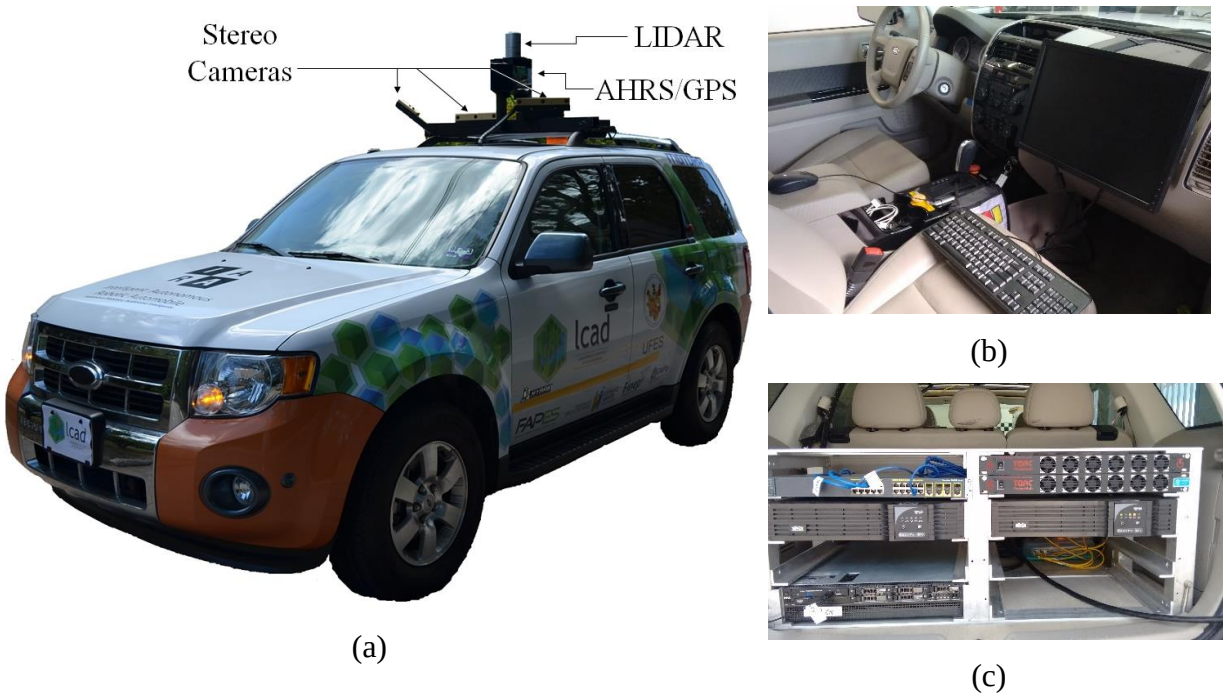

Fig. 1 – (a) Intelligent and Autonomous Robotic Automobile (IARA). (b) An inside view and of IARA. (c) The trunk with IARA's computers, no-breaks and switch. A video that shows IARA operating autonomously is available at https://youtu.be/iyKZV0ICysc.

In this work, we examine two alternatives for the implementation of the IARA's steering control subsystem: the standard Proportional Integral Derivative (PID) control approach, developed prior to this work, and a Neural-Based Model Predictive Control (N-MPC) approach.

The PID control approach involves specifying a desired control command and an error measure that accounts for how far the plant output is from the desired control command [AST08]. With this information, a PID control system computes the plant input, which is K_p times the error measure (proportional to the error according to K_p), plus K_i times the integral of the error, plus K_d times the derivative of the error, where K_p, K_i and K_d are parameters of the PID control system. Although very simple, the PID control approach is very effective and it is used in the control of many types of systems.

We have implemented a PID steering control (or lateral control) subsystem for IARA, as well as a PID velocity control (or longitudinal control) subsystem. IARA's PID control subsystems work well for speeds of up to 25 km/h. However, above this speed, delays and non-linearities of the IARA's Steering Plant (the software/hardware that goes from the point where IARA receives a desired steering angle command to the point that measures the actual steering angle, i.e., the odometry sensor) are too high to allow proper operation with a PID control approach. The delay problem occurs when a significant amount of time elapses between the

instant in which the control command is applied and the instant in which a variation in the plant is observed. It is caused by the drive by wire system comprised of the programmable logic controller, electric steering motor and motor driver installed by the company that adapted IARA for being controlled by computers. The non-linearity problem occurs when the change of the output is not proportional to the change of the input. For example, the more you turn the steering wheel, the harder it gets to turn.

To tackle IARA's Steering Plant delays, in this work, we propose the use of a N-MPC approach. The standard MPC approach tries and controls a plant using a model of it and an algorithm that, using the model, simulates the plant output some time steps ahead in order to decide what is the best plant input (i.e., the effort command that will minimize the difference between the plant output and the desired control command) to send to the plant in the current time step [ROS04]. Since the IARA's motion planning subsystem produces a trajectory about 5 seconds ahead in time [CAR16] and the IARA's Steering Plant delay is about 0.6 seconds [DES16], we can use the MPC approach to reduce the impact of steering plant delays by anticipating control commands that would timely move IARA according to the trajectory. However, standard techniques for predicting IARA's Steering Plant motion [LIU14] did not work well due to its non-linearities and delays. We then tried and modeled the IARA's Steering Plant using a neural network [DES16], in conjunction with the conjugate gradient numerical method to update the weights of the neural network, and employed this neural-based steering model in IARA's N-MPC steering control subsystem. Experimental results showed that the N-MPC approach outperforms the PID control approach by reducing the impact of IARA's Steering Plant delays and allows IARA's autonomous operation at speeds of up to 37 km/h – an increase of 48% in maximum stable speed previously achieved with IARA's PID steering control subsystem.

1.1 Motivation

This work is associated to a research project of the *Laboratório de Computação de Alto Desempenho* (High Performance Computing Laboratory - LCAD) of the *Departamento de Informática* (Department of Informatics - DI) at the *Universidade Federal do Espírito Santo* (Federal University of Espírito Santo - UFES). One of the objectives of this project is the autonomous navigation of IARA along a course named "Ida a Guarapari". This course has ap-

proximately 74 km, starts in the main campus of UFES, in Vitória, and ends in the Meaípe beach, in Guarapari.

The main motivation of this work is to contribute with the objective mentioned above by implementing a steering control subsystem that allows IARA to navigate autonomously along 74 km long “Ida a Guarapari” course.

1.2 Objective

The objective of this work is to propose a steering control subsystem that is able to tackle delays in the steering plant of the IARA autonomous car.

In a previous work, we implemented a steering control subsystem for IARA based on the standard PID control approach. In low speeds, the steering delay is not a problem for the standard PID control approach. However, in higher speeds, it causes large oscillations in the steering angle that prevents proper operation with a PID control approach.

An alternative for the implementation of the IARA’s steering control subsystem is the MPC approach. Standard MPC simulates the plant output some time steps ahead to compute the best plant input for the current time step. We can use the MPC approach to reduce the impact of steering plant delays by anticipating steering angle commands that would timely move IARA according to the trajectory. However, standard steering plant models did not work well due to its non-linearities and delays. An alternative for the implementation of the IARA’s Steering Plant model is a neural network based approach.

1.3 Contribution

The main contribution of this work is the use of a neural-based steering plant model in the MPC approach, in order to tackle delays in the steering plant of the IARA autonomous car. The neural network was necessary to proper model the complexities and nonlinearities of the IARA’s Steering Plant delays.

1.4 Organization

This work is organized as follows. After this introduction, in Chapter 2, we present previous related work. In Chapter 3, we describe the theoretical background related to this work. In Chapter 4, we describe IARA's PID steering control subsystem and, in Chapter 5, IARA's N-MPC steering control subsystem. In Chapter 6, we present experimental methodology used to evaluate IARA's N-MPC steering control subsystem and, in Chapter 7, experimental results. In Chapter 7, we present a discussion on experimental results and a critical analysis of this work. Finally, in Chapter 8, we present conclusions and directions for future work.

2 RELATED WORK

There are various approaches in the literature for the implementation of the control subsystem of autonomous cars. Funke et al. [FUN12] adopted a combined feedforward and feedback control approach for implementing the (lateral and longitudinal) control subsystems of an Audi TTS. The hardware structures of the Audi TTS plant were designed to achieve hard real-time control at 200 Hz. This high-speed hardware enabled their control subsystem to drive the car at up to 160 km/h (44.4 m/s). It allows their control subsystem to actuate at every 0.22 m when at this speed (44.4 m/s / 200 Hz = 0.22 m). The Audi TTS's plant is five times faster than IARA's plant, which operates at 40 Hz due to hardware limitations. Therefore, in theory, with a 5 times faster hardware our N-MPC steering control subsystem could drive IARA at up to 185 km/h, since the N-MPC can currently drive IARA at 37 km/h (with a 5 times faster hardware, we would have 37 km/h × 5 = 185 km/h). N-MPC actuates at every 0.26 m when IARA is at 37 km/h; so, a larger distance than the 0.22 m of the Audi TTS. Li et al. [LI15] employed the same combined feedforward and feedback control approach of Funke et al. [FUN12]. They evaluated their control subsystem on a Toyota Land Cruiser and it was able to drive the car at up to 25 km/h only.

Zhao et al. [ZHA12] employed an adaptive PID approach, based on the generalized minimum variance method, for implementing the control subsystem of an autonomous car named "Intelligent Pionner". Intelligent Pionner's control subsystem was able to drive the car at up to 60 km/h. They did not explain how they tackled delays in their car plant, perhaps because they had a fast plant and its delays where negligible for the operation of the car at up to 60 km/h.

Ziegler et al. [ZIE14] adopted a MPC approach for implementing the longitudinal control subsystem, and feedforward and feedback control approaches for the lateral control subsystem of "Bertha" – the Mercedes-Benz S-Class S500 that completed fully autonomously the historic Bertha-Benz-Memorial-Route in Germany. Bertha's control subsystem was able to drive the car at up to 100 km/h through the 100 km long Bertha-Benz-Memorial-Route. They did not mention delays in their car plant (perhaps, again, because they had a fast plant and its delays where negligible for the operation of the car at up to 100 km/h).

Koga et al. [KOG16] used a MPC approach for implementing the lateral and longitudinal control subsystem, which uses the standard bicycle model for predicting the autonomous

car motion. They evaluated their control subsystem on a small electrical car, which was able to drive the car at up to 20 km/h.

Levinson et al. [LEV11] used a mixture of a MPC approach, based upon well-known physically based vehicle models, along with a feedforward PID control approach for implementing the control subsystem of "Junior" – the (2006 Volkswagen Passat) Stanford's entry to the 2007 DARPA Urban Challenge that achieved the second place in this competition. Junior's control subsystem was able to drive the car at up to 56 km/h. Nevertheless, it is hard to be implemented because of the difficulty in deriving the control parameters for the physically-based vehicle model.

Different from all the control subsystems mentioned above, which generate longitudinal and lateral control inputs to track the trajectory generated by the motion planning subsystem, Götte et al. [GOT16] adopted a combined motion planning and lateral control subsystem. In their planning/control subsystem, a single prediction model is used to plan a trajectory and perform lateral control of the car at the same time. For implementing the planning/control subsystem, they employed a nonlinear MPC approach that incorporates environmental constraints, which leads to a model predictive planning and control approach. However, they tested their planning/control subsystem only on an autonomous car simulator.

There is a lack of details about the implementation of autonomous car control systems in the literature. Most of the systems either rely on a high precision hardware or work with low linear velocities. Besides that, we could not find any work that mention the delay problem in an autonomous car steering or velocity plant. Therefore, to deal with these issues, mainly the delay, we propose our Neural Based Model Predictive Control Approach.

3 THEORETICAL BACKGROUND

This chapter presents the theoretical background related to this work. Firstly, we present a brief description of the operation of the Proportional Integral Derivative Control approach. Then a description of how the Model Predictive Control approach works followed by a discussion on a set of System Plant Models and the Conjugate Gradient Numerical Method.

3.1 Proportional Integral Derivative (PID) Control

The PID control approach operates observing and trying to minimize the system output error e_t, the difference between a desired set-point (or desired target value) r_t and the measured system plant output value y_t. For that, PID applies corrections in the control variable u_t based on the proportional, integral, and derivative terms P, I and D respectively. Fig. 2 show the block diagram of PID controller approach, detailing how each one of the terms are obtained.

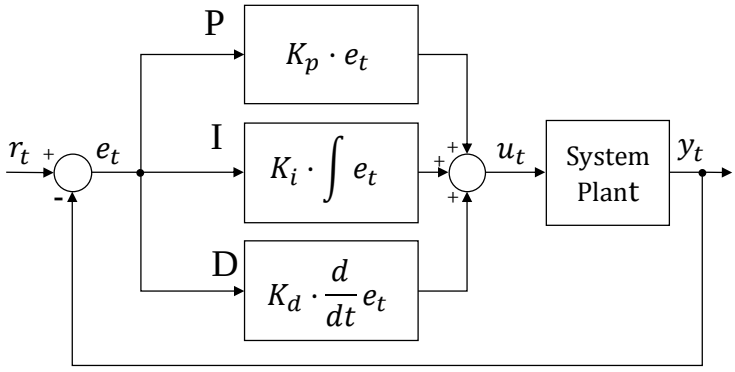

Fig. 2 – Block Diagram of PID approach.

- **(P) Proportional term:** produces an output value that is proportional to the current system output error e_t times a proportional gain K_p. A very large proportional gain can make the system become unstable and a very low gain may prevent the system to respond.
- **(I) Integral term:** produces an output value that is the sum of the system output error e_t over time, times an integrative gain K_i. Since the integral term responds to accumulated errors from the past, it can cause the system to overshoot.

- **(D) Derivative term:** produces an output value that is calculated by determining the slope of the system output error e_t over time, times a derivative gain K_d. Is used to predict the system behavior and thus improves settling time and stability of the system.

The control variable u_t is calculated based in a sum of K_p times the error e_t, K_i times the integral of the error e_t and K_d times the derivative of the error e_t, as shown in Equation (1).

$$u_t = K_p e_t + K_i \int e_t \, dt + K_d \frac{de_t}{dt}, \qquad (1)$$

The simplicity of PID controller approach and the possibility of applying it without advanced knowledge of the process that is being controlled make it widely applied in different fields. The three parameters of PID may be tuned to deal with specific process requirements and some applications may require using only one or two terms. If a mathematical system plant model can be obtained, it can be used to tune the PID parameters using one of several analytical approaches available depending on the requirements of the project. Most of the times the system plant model is so complex that a reliable system plant model cannot be obtained, so there are some heuristic tuning methods that enable the PID tuning without the system plant model. We can say that the most common heuristic PID tuning method is the Ziegler-Nichols tuning rules proposed in 1942 and will be discussed in Section 3.1.1.

The biggest disadvantage of PID approach is that it does not explicitly consider the future implication of current control actions, and so it cannot deal well with system plant delays, what may cause instability. It also has difficulties in the presence of nonlinearities, do not react to changes in the process behavior, have lag in responding to large disturbances and cannot guarantee optimal control of the system or even its stability.

3.1.1 Ziegler-Nichols Open-Loop Tuning Rules

The Ziegler-Nichols open-loop tuning rules are based in three process characteristics: process gain, dead time, and time constant. Process gain, describes how far the system plant output value y_t will travel for a given change in control variable u_t, sometimes called the sensitivity of the process. Dead time is the delay between a change in u_t and the response of y_t. Time constant describes how fast y_t moves in response to a change in u_t. These characteristics can be obtained by analyzing the open-loop system plant step response graphic [OGA10,

SMU11]. The Ziegler-Nichols method is completely graphical approximation and the analysis should be made by following these steps:

1 Apply a step change of a few percent in the control variable u_t and wait for the system plant output value y_t to settle out. The size of this step should be large enough that the system plant output value y_t moves well clear of the process noise/disturbance level. If the system plant does not have integrators or dominant complex conjugate poles, the curve of step response will show an "S" aspect as shown in Fig. 3. The Ziegler-Nichols Open-Loop Tuning Rules can only be applied if the step response shows this aspect.

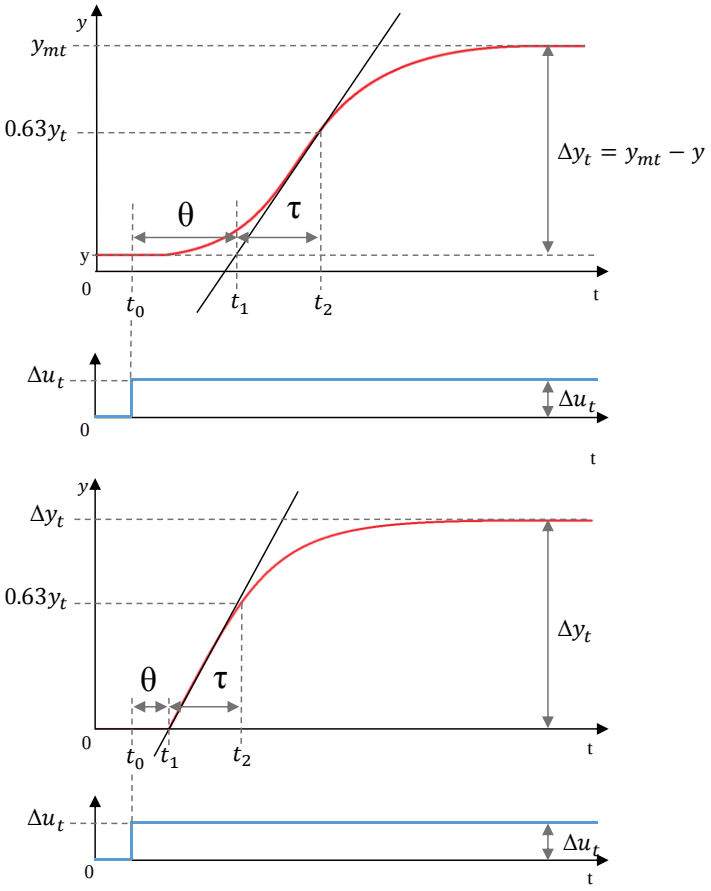

Fig. 3 – Two examples of step response curve of open-loop system, for Ziegler-Nichols PID tuning.

2 Find the point that y_t achieve 0.63% of its maximum value.

3 Find the inflection point of the curve y_t, and draw a tangential line to the curve passing through this point.

4 Using the graphic obtained from the step response, find the following parameters:

- Δy_t variation in the system plant output value

- Δu_t variation in the control variable
- t_0 time of beginning of control variable u_t variation
- t_1 time that the tangential line crosses the line that passes through the line of initial value of y_t
- t_2 time that y_t takes to achieve 0.63% of its maximum value

5 Calculate the parameters process gain K, dead time θ, and time constant τ, using equations (2), (3) e (4) as follows.

$$K = \frac{\Delta y_t}{\Delta u_t}, \quad (2)$$

$$\theta = t_1 - t_0 \quad (3)$$

$$\tau = t_2 - t_1 \quad (4)$$

6 Use the Ziegler-Nichols Open-Loop Tuning Rules shown in Table 1, to compute the proportional gain K_c, integral time τ_i and derivative time τ_d.

Table 1 – Ziegler-Nichols Open-Loop Tuning Rules.

Controller Type	$\boldsymbol{K_c}$	$\boldsymbol{\tau_i}$	$\boldsymbol{\tau_d}$
P	$\frac{\tau}{K\theta}$	–	–
PI	$\frac{0.9\tau}{K\theta}$	0.33θ	–
PID	$\frac{1.2\tau}{K\theta}$	2θ	0.5θ

The original Ziegler-Nichols tuning rules were designed for controllers using reset rate (integral gain in repeats per minute). However, all the modern texts on process control use integral time, so we need to convert reset rate parameters to integral time parameters. Equation (5) presents the reset rate PID, to convert it to integral time PID we just need to match the terms of equations (1) and (5).

$$u_t = K_c e_t + \frac{K_c}{\tau_i} \int e_t \, dt + K_c \tau_i \frac{de_t}{dt}, \quad (5)$$

Equations (6), (7) and (8) shows the calculation of K_p, K_i and K_d from K_c, τ_i and τ_d.

$$K_p = K_c, \quad (6)$$

$$K_i = \frac{K_c}{\tau_i} \quad (7)$$

$$K_d = K_c \tau_d \quad (8)$$

A combination of Table 1 and equations (6), (7) and (8) can be made to give directly the PID parameters, as shown in Table 2.

Table 2 – Ziegler-Nichols Open-Loop Tuning Rules simplification.

Controller Type	K_p	K_i	K_d
P	$\frac{\tau}{K\theta}$	–	–
PI	$\frac{0.9\tau}{K\theta}$	$\frac{3.64\tau}{K\theta^2}$	–
PID	$\frac{1.2\tau}{K\theta}$	$\frac{0.6\tau}{K\theta^2}$	$\frac{0.6\tau}{K}$

It is recommended to do two or three tests and calculate process gain K, dead time θ, and time constant τ, for each test to obtain a good average of the process characteristics. If you get vastly different numbers, do even more step tests and discard the most different ones, until you have a good average. Note that these rules produce a quarter-amplitude damping response, and the controller gain, K_c, can be reduced by a factor of up to 50%, to reduce overshoot and improve stability.

3.2 Model Predictive Control (MPC)

The Model Predictive Control (MPC) approach consists of selecting control actions which will lead to the best the system plant output y_t, using the model of the system to simulate and optimize the system plant output y_t, over a prediction horizon in the future. The main advantage of MPC is the fact that it allows the current control action to be optimized, while keeping future control actions in account. This consideration of the future set-point values r_t

in the current control action, gives MPC the ability to anticipate future events and take control actions accordingly to minimize the system error e_t. At time t, the current plant state is sampled and a cost minimizing control strategy is computed, generally using a numerical minimization algorithm, for a relatively short time horizon in the future T, the prediction horizon. Only the first step of the control is applied to the system plant. Then the plant state is sampled again and the calculations are repeated starting from the new current state.

In order to predict the future behavior of a process, we must have a model of how the process behaves. A precise model is not always required to get tight control because the decisions are updated at every control cycle, this allows MPC to deal with the model uncertainties [ROS04].

In order to choose the best current control action, we need criteria to judge which action is the best, that is made using a **cost function**, commonly is the minimization of the system error e_t. Selection of the cost function considers the predicted behavior over the prediction horizon into the future and therefore at each successive sampling instant.

The MPC controller will be as precise as the model i.e., for a **highly-tuned** controller, we need a very accurate model. If we have a very accurate model and the right cost function, the system will achieve automatically stability and tuning.

Typically, in case of a huge variation in the set-point the PID control approach will deal with this only at the instant the variation occurs, what will possible cause the system to overshoot due to excess of actuation, this problem would be solved using MPC approach. Another issue that can be easily deal with using MPC is to control Multiple Input Multiple Output systems (MIMO) since a model of the MIMO system can be build. As the PID approach designs make use of relatively small amount of information about the plant it cannot deal with the interaction of MIMO systems effectively.

3.2.1 System Plant Models

The task of obtaining an accurate model of the system plant is difficult and very dependent of the type of the system being controlled, but as the selection of the model is the most important part of an MPC design, this section will give a very brief description of some common system plant models applied with MPC approach.

The **State Space Models** represents the physical system as a set of input, output and state variables related by first-order differential equations. It has an advantage of easily extend to the multivariable case and there is a huge quantity of theoretical results which can be applied to produce controllers/observers and to analyze the models and resulting control laws.

Transfer function model is a representation in terms of spatial or temporal frequency, of the relation between the input and output of a system. Historically one advantage of this was the close relationship to popular black box identification techniques.

Finite Impulse Response (FIR) is a filter whose response to a finite length input is of finite duration, because it settles to zero in finite time. In practice, these models could be determined by a single step test, the practical requirements for identifying FIR models is that far more data is needed than to identify state-space and transfer function.

An interesting but not common way to model a system plant is to use **Neural Networks**. They can model with high precision even nonlinear systems but they are difficult to choose between all the different types of networks and for each network tune the different configurations that best fit for the kind of system to be modeled. When using a neural network as the system plant model, it is necessary to incorporate an optimization method to find the optimum control output. A well-known optimization method is the conjugate gradient method, that will be described in next section.

3.2.2 Conjugate Gradient Numerical Method

The conjugate gradient is one of the most well-known iterative methods for finding the nearest local minimum with a few steps. It uses conjugate directions instead of the local gradient for going downhill. If the vicinity of the minimum has the shape of a long, narrow valley, the minimum is reached in far fewer steps than would be the case using the method of steepest descent.

The conjugate gradient algorithm requires derivatives of the function to be minimized, with respect to the optimizing variables, to know the direction of the step to find the local minimum. Most of time the differential equations cannot or are too difficult to be calculated so they can be computed numerically. A very common numerical method for solving differential equations is the finite difference method, that consists in approximating the differential operator $f'(a)$ by replacing the derivatives in the equation $f(a)$ using a differential quotient h as shown in Equation (9).

$$f'(a) \approx \frac{f(a+h)-f(a)}{h}, \qquad (9)$$

4 IARA'S PID STEERING CONTROLLER

In this chapter, we describe the Proportional Integral Derivative (PID) controller approach implemented in the IARA autonomous car, that was developed prior to this work. Fig. 4 presents the architecture of the IARA's PID steering control subsystem, namely IARA's PID Steering Controller. In the IARA's PID Steering Controller, the desired car's steering angle is represented by φ_t^d, while the current car's steering angle is represented by φ_t. However, IARA's odometry sensor informs the car's steering angle using another metric that is called arctangent of the curvature (AOC, measured in radians). The reason is that the technology we have used to implement the electromechanical subsystem that drives IARA's steering wheel was provided by Torc Robotics [TOR10], which uses AOC instead of the steering angle in their odometry message.

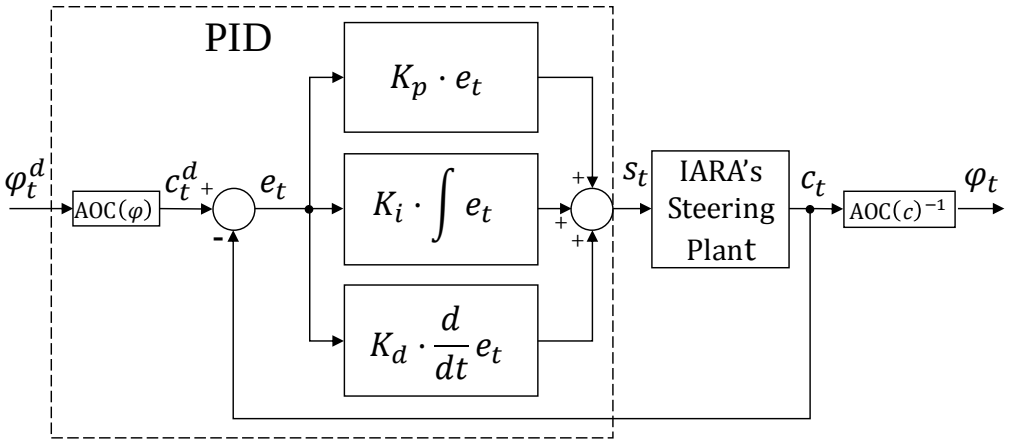

Fig. 4 – Diagram of the architecture of IARA's PID Steering Controller.

The car's curvature, C, can be defined as the inverse of the car's turning radius (measured in m^{-1}) and, in IARA, it is related to the car's steering angle, φ, by [DES16]

$$\mathrm{C} = \frac{\tan\left(\frac{\varphi}{1+v^2k}\right)}{L}, \qquad (10)$$

where v is the car's velocity, k is the car's understeer coefficient (1.5×10^{-3} in IARA) and L is the car's wheel base, i.e., the distance from the rear axle to the front axle (in meters). AOC, as a function of φ, is given by

$$\mathrm{AOC}(\varphi) = \tan^{-1}\left(\frac{\tan\left(\frac{\varphi}{1+v^2k}\right)}{L}\right). \qquad (11)$$

Torc have chosen to use AOC instead of C to provide a more evenly distributed resolution across the integer field that code C in their hardware. In IARA's PID Steering Controller, we convert φ to AOC using Equation (11) in order to make it comparable with the information provided by IARA's odometry sensor message. In Fig. 4, the block $\mathrm{AOC}(\varphi)$ converts φ to AOC.

IARA's PID Steering Controller (Fig. 4) receives as input φ_t^d, converts φ_t^d to the desired IARA's AOC, c_t^d, and computes the current steering effort, s_t, that will make IARA's Steering Plant produce as output the current AOC, c_t. It does so by computing the current error, e_t, i.e., the difference between c_t and c_t^d, and, from e_t, the s_t that will drive IARA's Steering Plant closer to the correct value. In Torc's implementation, s_t is a value in the range [-100, 100]. IARA's PID Steering Controller computes s_t by adding three values, i.e.: $s_t = K_p \times e_t + K_i \times \int e_t\, dt + K_d \times \frac{d}{dt} e_t$. We have selected the constants K_p, K_i and K_d using standard PID tuning techniques. Finally, c_t is converted to φ_t, which is sent to other subsystems of IARA. In Fig. 4, the block $\mathrm{AOC(c)}^{-1}$ converts c_t to φ_t.

IARA's PID Steering Controller only produces s_t if there is a current or previous (thanks to the integral of previous errors) e_t. Since IARA's Steering Plant has a significant delay, e_t might evolve for a significant amount of time before any change in the plant output can be perceived due to previously applied s_t related to it. Fig. 5 shows experimental results that allow estimating IARA's Steering Plant delay in the form of its response time [DES16].

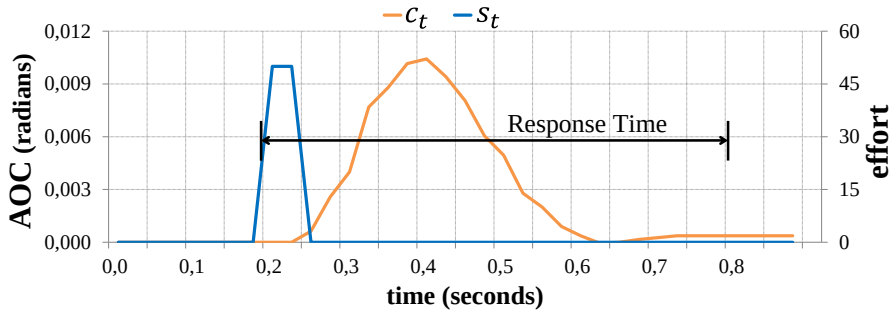

Fig. 5 – IARA's Steering Plant response time [DES16].

In Fig. 5, the x axis is time (in seconds), the right y axis is s_t(dimensionless value in the range [-100, 100]) and the left y axis is c_t (in radians). In the figure, s_t (blue curve) is an approximation of the Dirac Delta function and c_t (orange curve) captures IARA's Steering Plant response to the approximate Dirac Delta. The time that IARA takes to completely respond to the stimulus (Dirac Delta), added to the stimulus time, characterizes the time con-

stants of the dynamics that govern the IARA's Steering Plant response in the time domain, and can be used as an estimate of its maximum response time to stimuli [ANG12]. As Fig. 5 shows, the IARA's Steering Plant maximum response time, or delay, is not negligible – it is about 0.6 s. In low speeds (up to about 25 km/h), the steering delay is not a problem for IARA's PID Steering Controller; however, in higher speeds, it causes large oscillations in φ_t that prevents proper operation. That's why we have chosen to implement a new steering control subsystem based on the model predictive control approach. With our N-MPC, we take advantage of the fact that IARA's motion planning subsystem provides a time series of desired steering wheel angles, $\Phi^d = \{\varphi_t^d,\ \varphi_{t+1}^d,\ \dots,\ \varphi_{|\Phi^d|}^d\}$, and use a neural model of the IARA's Steering Plant to generate a *predicted* time series of the car's steering wheel angles, $\Phi^p = \{\varphi_t^p,\ \varphi_{t+1}^p,\ \dots,\ \varphi_{|\Phi^p|}^p\}$ using a time series of steering wheel efforts, $S = \{s_t, s_{t+1}, \dots, s_{|S|}\}$. N-MPC selects the optimal time series of steering wheel efforts that will lead φ_t according to the motion plan (i.e., the S that will minimize the difference between $\Phi^d$and Φ^p), anticipating the steering wheel efforts, s_t, that will counteract the Steering Plant delay when necessary and possible.

5 IARA'S N-MPC STEERING CONTROLLER

In this chapter, we describe the operation of the Neural Based Model Predictive Control approach proposed in this work. In Section 5.1, we describe the neural network employed to model IARA's steering plant. In Section 5.2, we describe the N-MPC operation.

5.1 Neural-Based Steering Plant Model (N-SPM)

Due to the complexity of the dynamics of the IARA's Steering Plant response to stimuli, we tried and modeled it using a neural network [DES16]. This neural network, that we call Neural-Based Steering Plant Model (N-SPM), simulates the mechanisms that govern how a time series of s_t (executed steering efforts) alters the c_t (measured AOCs) of IARA.

Fig. 6 shows a diagram of N-SPM. In the training phase, the neural network of N-SPM receives as input a set of input samples and associated outputs, $T = \{(I_1, c_1),\ (I_2, c_2), \dots, (I_t, c_t), \dots (I_{|T|}, c_{|T|})\}$. Each input sample, I_t, is a time series of executed s_t, $S_t = \{s_{t-1}, s_{t-2}, \dots, s_{t-k}\}$, and measured c_t, $C_t = \{c_{t-1}, c_{t-2}, \dots, c_{t-k}\}$, at k past time instants (i.e., $I_t = S_t \cup C_t$), which are stored into the Steering Effort Queue and AOC Queue, respectively (see Fig. 6). The output associated with each input sample is the c_t measured at the current time instant, t. For each input sample, I_t, the neural network predicts d_t and this prediction is compared with the associated output, c_t. The squared error between predictions, d_t, and measurements, c_t, calculated over different subsets of the training set, is repeatedly used by a gradient-based Learning Algorithm to update the weights of the network until a maximum number of epochs or a minimum error is achieved. This whole process is repeated for neural networks with different number of layers, neurons per layer and type of neurons, among other parameters, all selected using a genetic algorithm (see Section 5.1.1). The best individual (network) can then be used in the test phase. More details of how we have implemented N-SPM can be found in De Souza et al. [DES16].

In the test phase, the neural network of N-SPM receives as input a time series of executed s_t and measured c_t at k past time instants ($S_t$and C_t, respectively) and outputs d_t.

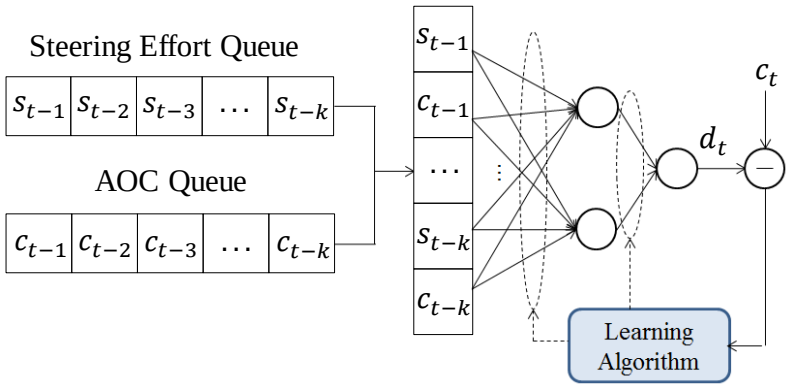

Fig. 6 – Diagram of N-SPM. The N-SPM neural network receives as input a time series of executed s_t, $S_t = \{s_{t-1}, s_{t-2}, \dots, s_{t-k}\}$, and measured c_t, $C_t = \{c_{t-1}, c_{t-2}, \dots, c_{t-k}\}$, at k past time instants, and outputs a prediction of c_t, d_t.

In previous work [DES16], we evaluated the performance of the N-SPM using real-world datasets. Experimental results showed that N-SPM was able to simulate, in real time, how a time series of s_t influences c_t and, while navigating in a map of a real-world environment, N-SPM was able to emulate the IARA's AOC with root mean squared error of 6.3×10^{-3} radians.

5.1.1 Genetic Algorithm used for Finding the N-SPM Proper Configurations

In previous work [DES16], we used Genetic Algorithms (GA) for finding proper values for the sets of parameters for the neural network of N-SPM. In this context, each individual in the GA represents a neural network configuration. The set of parameters of such a configuration and their search space are listed in Table 3.

A group of individuals defines a population in the GA. Initially, a random population is created from the parameters' search space. Subsequently, an iterative process starts and it is repeated for a maximum number of iterations, or until a user-defined minimum value of the evaluation function is achieved. At each iteration, the GA creates a new population from the previous one by following three steps: (i) selection of parents, (ii) creation of new individuals via crossover of parents, and (iii) mutation of the created individuals.

Table 3 – Parameters of the neural networks evaluated by the genetic algorithm.

Parameters	**Search Space**
Number of hidden layers	1; 2
Activation function of hidden layers	Linear; threshold; sigmoid; sigmoid with stepwise linear approximation; symmetric sigmoid (hyperbolic tangent); stepwise linear approximation to symmetric sigmoid; Gaussian; symmetric Gaussian; sigmoid proposed by David Elliott [21]; bounded linear; periodical sine; periodical cosine. (see http://goo.gl/2vJRLS)
Activation function of output layer	Same used in Activation function of hidden layers.
Number of neurons in the hidden layer	First hidden layer: 5; 8; 10; 12; 15; 18; 38; 50; 80; 100; 200; 300. When we have more than one hidden layer, the other hidden layers have half the number of neurons of the previous hidden layer.
Learning Algorithm	Online Backpropagation; Batch Backpropagation; RPROP algorithm [22]; QUICKPROP algorithm [23].
Learning Rate	0.1; 0.2; …; 0.9
Momentum	0.1; 0.2; …; 0.9
Learning Rate Decay	0.1; 0.2; …; 0.9 (relevant only for the QUICKPROP training algorithm)
Maximum epochs to train	100; 200; 300

In the selection step, two random parents are chosen for creating a new individual. In this work, the fitness of the parents is not taken in consideration for the selection process. The new individual is generated from the selected parents using a multipoint crossover strategy [SRI94]. If a new individual already exists in the population, it is discarded and a new crossover is executed. The selection and crossover steps are repeated until a new population with size equals to 90% of the size of the previous population is created. To complete the new population, the previous population is ranked according to the fitness of each individual and the top-performing 10% are transferred to the new population – this technique is called elitism [JON75]. In the mutation step, each individual of the new population suffers a mutation with 10% probability. The mutation consists of changing one of the individual's parameters (selected randomly) to a random value of its search space.

To measure the performance, or fitness, of an individual, we need to build, train, and evaluate a neural network configured according to the individual's parameters. The neural

network is trained using a subset of the training set, and evaluated with the remaining part of the training set, called the validation set. Training and evaluating the network with independent sets is important to guarantee that the network acquires generalized knowledge, instead of overfitting to the training data. The evaluation function is the root mean square error (RMSE) between the expected values of AOC and the predictions made by the network over all samples of the validation set. The evolution of the individuals, promoted by the GA, seeks to minimize this evaluation function.

5.2 N-MPC Architecture

Fig. 7 shows a diagram of the architecture of IARA's N-MPC steering control subsystem, namely IARA's N-MPC Steering Controller. The IARA's N-MPC Steering Controller has three cycles: Control Cycle, Optimization Cycle and N-SPM Prediction Cycle. A Control Cycle encompasses various Optimization Cycles, and an Optimization Cycle encompasses various N-SPM Prediction Cycles.

At each **Control Cycle**, N-MPC receives a motion plan from IARA's planner and extracts a time series of φ_t^d (desired IARA's steering wheel angles), $\Phi^d = \{\varphi_1^d, \varphi_2^d, \dots, \varphi_k^d, \dots, \varphi_{|\Phi^d|}^d\}$, from it. This time series is compared with a time series of φ_t^p (predicted IARA's steering wheel angles), $\Phi^p = \{\varphi_1^p, \varphi_2^p, \dots, \varphi_k^p, \dots, \varphi_{|\Phi^p|}^p\}$, and the result of this comparison is used by an Optimizer to produce an optimal steering effort, k_1, that is sent to the IARA's Steering Plant (see Fig. 7). This Optimizer may require several optimization cycles to achieve an optimal value of k_1. We have limited to 15 the maximum number of optimization cycles. The Control Cycle executes in the hardware maximum frequency, 40 Hz.

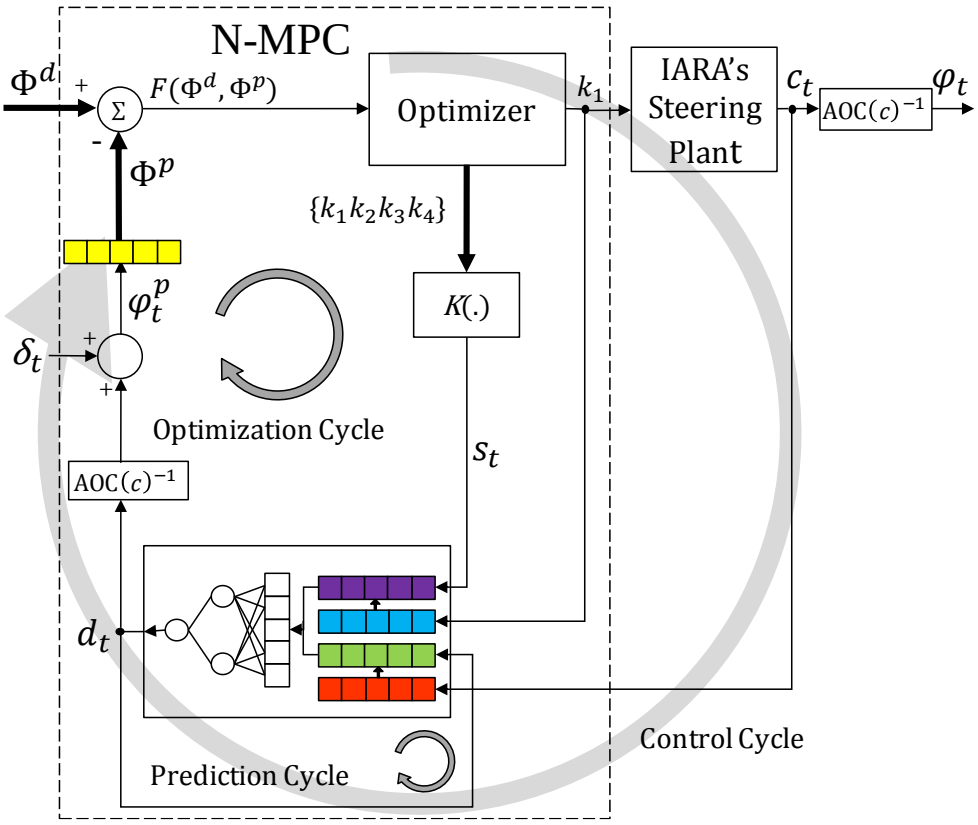

Fig. 7 – Diagram of the architecture of IARA's N-MPC Steering Controller.

At each **Optimization Cycle** (Fig. 7), Φ^d is compared with Φ^p according to the cost function

$$F(\Phi^d, \Phi^p) = \frac{1}{|\Phi^p|} \sum_{t=1}^{|\Phi^p|} \left| \varphi_t^d - \varphi_t^p \right|. \qquad (12)$$

The cost returned by $F(.)$ is used by the Optimizer to compute the optimum parameters of a spline of s_t (steering efforts), $K(k_1,\ k_2,\ k_3,\ k_4)$ (see Fig. 7). $K(.)$ is a cubic spline that specifies the evolution of s_t in time and it is defined by four steering effort knot points, k_1, k_2, k_3 and k_4. In the **first Optimization Cycle of the first Control Cycle**, the parameters of $K(.)$ are set to zero and these zeroed parameters are used as the first Optimizer seed; in the following Optimization Cycles, the parameters of $K(.)$ computed in the previous Optimization Cycle are used as the Optimizer seed. In the **first Optimization Cycle of each Control Cycle**, the optimum parameters of $K(.)$, computed in the previous Control Cycle, are used as the Optimizer seed in the current Control Cycle.

In the **beginning of each Optimization Cycle**, the Steering Effort Queue (purple queue in Fig. 7) and the AOC Queue (green queue in Fig. 7) are initialized with executed s_t and measured c_t at k past time instants that were stored in the Executed Effort Queue (blue queue

in Fig. 7) and the Measured AOC Queue (red queue in Fig. 7), respectively. After this initialization, from the current t to $t + tp$, where tp is the Prediction Horizon, N-SPM is recursively used for making predictions, d_t. These predictions are employed for building Φ^p (as described below), which is used in each Optimization Cycle (see Fig. 7).

In the **first N-SPM Prediction Cycle** of each Optimization Cycle, N-SPM predicts d_t using as input only executed s_t and measured c_t of the k past time instants, which are copied from the Executed Effort Queue to the Steering Effort Queue and from the Measured AOC Queue to the AOC Queue, respectively. At each subsequent **N-SPM Prediction Cycle**, each s_t taken from $K(.)$ and each d_t (steering effort) predicted by N-SPM are inserted into the Steering Effort Queue of N-SPM and the AOC Queue, respectively. With this information, the N-SPM predicts d_{t+1}, which is converted to φ_t^p, added to the disturbance correction variable, δ_t(explained below), and inserted into the Predicted φ Queue (yellow queue in Fig. 7). The N-SPM Prediction Cycle is recursively repeated until we have Φ^p with the same size of Φ^d, i.e., $|\Phi^p| = |\Phi^d|$. Φ^p is then compared again to Φ^d according to $F(.)$ (Equation (3)) and the cost returned by $F(.)$ is used in another Optimization Cycle. The Optimization Cycles are repeated until the Optimizer converges to the optimum set of parameters of $K(.)$. The parameter k_1 of the optimum $K(.)$ is sent to IARA's Steering Plant. The executed k_1 and the measured c_t are inserted into the Executed Effort Queue and the Measured AOC Queue, respectively, and the oldest values of each of these queues are removed. This **terminates the Control Cycle**.

The Optimizer uses a conjugate gradient optimization method to compute the optimum parameters of $K(.)$ that minimizes $F(.)$ in each Optimization Cycle. The conjugate gradient method requires derivatives of $F(.)$ with respect to the optimizing parameters (k_1, k_2, k_3, k_4). These derivatives are computed using the finite differences numerical method.

As mentioned above, Φ^d is taken from the current desired trajectory, P, which is generated by the IARA's motion planning subsystem [CAR16] at a rate of 20 Hz. Fig. 8 illustrates P, which is a set of motion commands, $P = \{m_1, m_2, \dots, m_{|P|}\}$, where each motion command is a triplet, $m_t = \{v_t, \varphi_t, \Delta t_t\}$, that specifies the car's velocity, v_t, car's steering angle, φ_t, and the time interval of the motion command, Δt_t, at the time instant t, which will lead IARA from the current state to its next goal state in the map. To compute Φ^d, for each $m_t \in P$, we take $\varphi_t \in m_t$ until the sum of $\Delta t_t \in m_t$ is greater than or equal to the Prediction Horizon, tp.

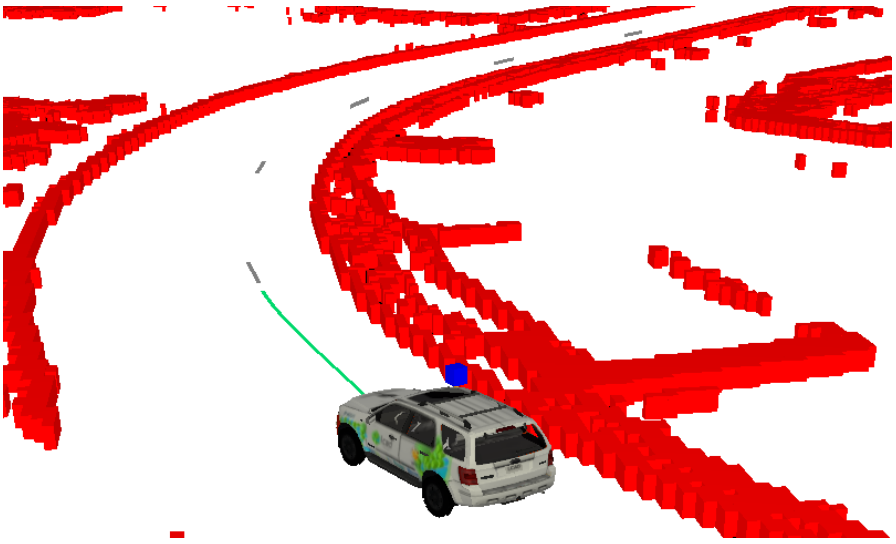

Fig. 8 – Current desired trajectory, P. The green curve denotes P, which starts at the current state of IARA. The grey traces denote the goals and the red cubes denote obstacles, which, as a whole, form the map.

The graph of Fig. 9 illustrates the operation of IARA's N-MPC Steering Controller using real-world data. A vertical line splits this graph in two parts: Past and Future. In the beginning of the Control Cycle just after the vertical line, the Steering Effort Queue and the AOC Queue are initialized with previous values of s_t (blue curve in the Past side) and c_t (the red curve in Fig. 9 shows previous values of φ_t, which are derived from measured values of c_t; see Fig. 7). A series of Optimization Cycles (and associated N-SPM Prediction Cycles) of this Control Cycle produces the values of the Prediction Horizon of Fig. 9.

At each N-SPM Prediction Cycle, a s_t taken from $K(.)$ (blue curve in the Future side of Fig. 9) is inserted into the Steering Effort Queue and a d_t, previously predicted by N-SPM, is inserted into the AOC Queue, and the N-SPM predicts a new d_t, or d_{t+1}. This process is recursively repeated until the first Prediction Horizon is complete. The Optimizer evaluates the quality of the Φ^p(yellow curve) of this first Prediction Horizon by comparing it with Φ^d (green curve in the Future side of Fig. 9), using Equation (12). If Φ^p is optimal, the Control Cycle is complete and k_1 is sent to IARA's Steering Plant; otherwise, a new Optimization Cycle (and associated N-SPM Prediction Cycles) is executed. Fig. 10 is a crop of Fig. 9 that shows more details of $K(.)$.

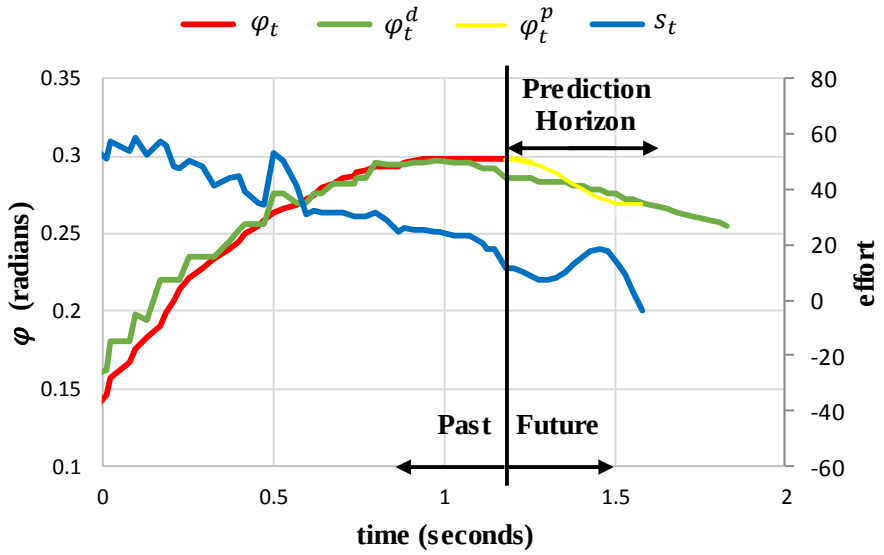

Fig. 9 – Operation of the IARA's N-MPC Steering Controller using real-world data. The vertical line splits the graph in two parts: Past and Future. In the Past side, the blue curve denotes executed s_t and the red curve denotes φ_t computed from measured c_t at k past time instants. In the Future side, the green curve denotes φ_t^d, the blue curve denotes s_t taken from $K(.)$ and the yellow curve denotes φ_t^p. The green curve in the Past side denotes φ_t^d and is shown to allow an appreciation of the performance of N-MPC (ideally, it should be equal to the red curve in the Past side).

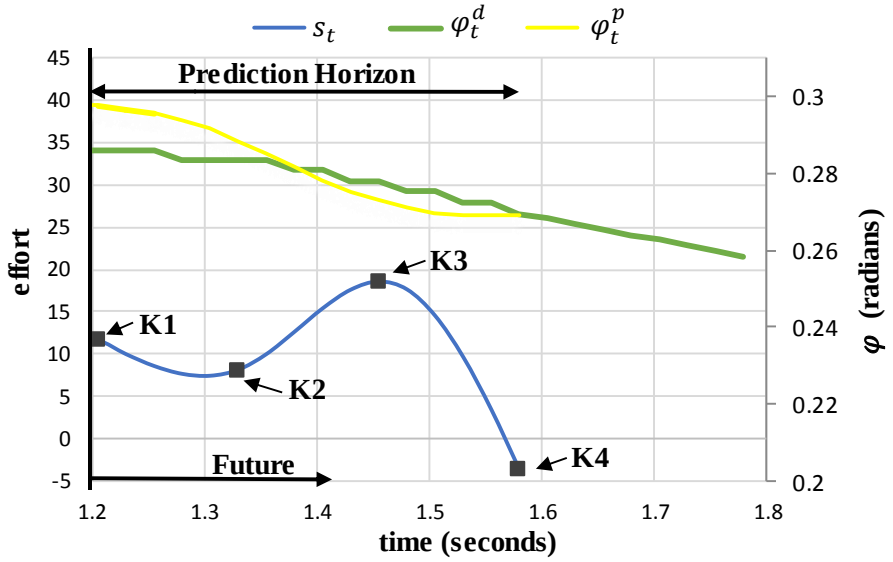

Fig. 10 – Crop of Fig. 9 that shows more details of the $K(.)$. The y axes have changed sides and origin, but the data is still the same. Only the Future Horizon is shown and the parameters of $K(.)$ are indicated.

Although the N-SPM models the IARA's Steering Plant well, there still might be modeling imprecisions, plant variations over time, or variations in terrain that may not be captured by N-SPM. To account for these, we include a disturbance correction variable, δ_t (mentioned above), which is defined as:

$$\delta_t = \varphi_t - \varphi_t^p. \qquad (13)$$

The use of δ_t allows for the compensation of the aforementioned plant modeling errors [ROS04].

In summary, the N-MPC receives as input the first part of the trajectory, about 0,4 s ahead in time, that composes Φ^d, and uses the optimizer, that employs the neural network, to modify the steering effort spline and minimize the error between the desired Φ^d and predicted Φ^p steering wheel angles sequence. Finally, when the process is complete, N-MPC selects the spline first value k_1, which is the best control command to be applied in the steering plant in actual moment, to direct the steering plant output to the desired value. N-MPC explicitly considers the implication of the actual command in the future, being able, in this way, to deal with the delay problem.

6 EXPERIMENTAL METHODOLOGY

In this chapter, we describe the methodology followed to perform the experiments and all resources employed. In Section 6.1, we describe a description of PID and N-MPC parameters tuning. In Section 6.2, we describe the experimental environment and situations of analysis. In Section 6.3, we present the metric used to compare the PID and N-MPC approaches. In Section 6.4, the hardware used in the experiments. In Section 6.5, the software. In Section 6.6, we describe the CARMEN robotics framework.

6.1 IARA's Steering Controllers Tuning

The parameters of the IARA's PID Steering Controller – K_p, K_i and K_d – were tuned using the Ziegler Nichols method described in Section 3.1.1. The parameters of N-SPM (employed by the IARA's N-MPC Steering Controller) – number of layers, neurons per layer and type of neurons, among others (see Section 5.1.1) – were tuned using a genetic algorithm described in Section 3.2.2.

6.2 IARA's Steering Controllers Evaluation

To evaluate the performance of the IARA's N-MPC Steering Controller against the IARA's PID Steering Controller, IARA was driven autonomously in two different situations along two different stretches of the ring road of the main campus of *Universidade Federal do Espírito Santo* (Federal University of Espírito Santo - UFES). Fig. 11 shows the Google Map view of the UFES main campus ring road. Fig. 11(a) shows the whole UFES main campus ring road, which has an extension of about 3.7 km. Fig. 11(b) shows the first stretch, which comprises a sharp curve, and Fig. 11(c) shows the second stretch, which comprises a series of smoother curves.

In the first situation, IARA's path planning and motion planning subsystems were switched off, in a way that IARA's steering controllers are evaluated without any interference of other subsystems and execute exactly the same control inputs. IARA was driven autonomously by simple trajectories – composed of steering angles in trapezoidal and sinusoidal

wave forms and constant linear velocities of 5 km/h, which were sent directly to IARA's steering controllers. Also, IARA was conducted along the end of the second stretch of the UFES main campus ring road (Fig. 11(c)), because it has a larger straight area.

In the second situation, IARA's path planning and the motion planning subystems were switched on, in a way that IARA's steering controllers are analyzed in standard operation mode, while interacting with other subsystems. IARA was driven autonomously with a maximum velocity of 37 km/h. Also, IARA was conducted along two different stretches of the UFES main campus ring road (Fig. 11(b) and Fig. 11(c)), which comprise a sharp curve and a series of smoother curves.

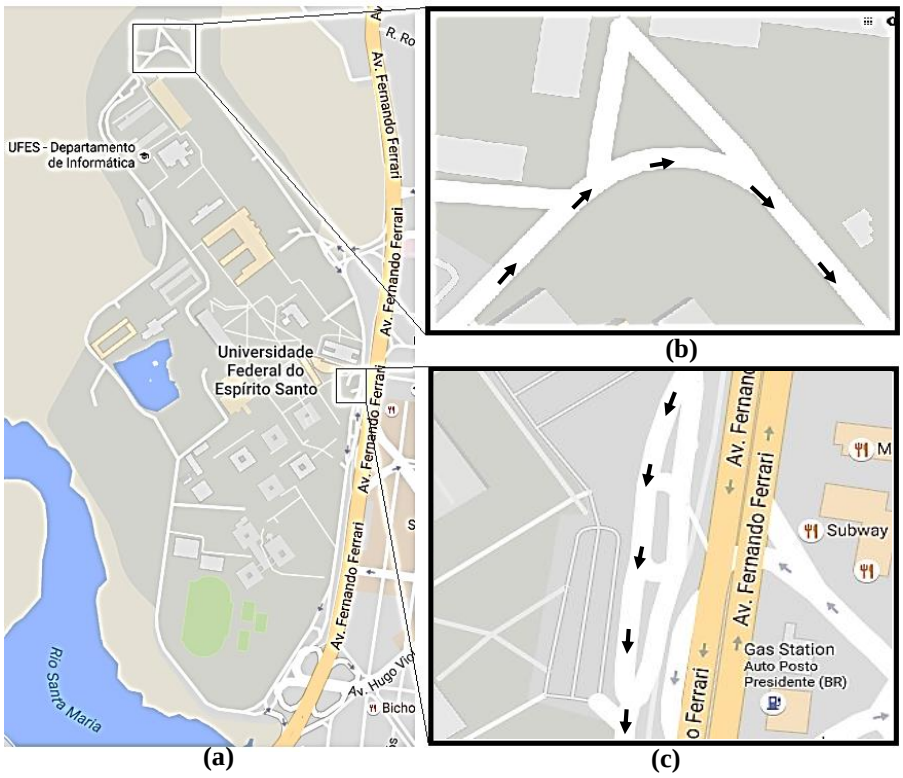

Fig. 11 – (a) Google Map view of the UFES main campus ring road. (b) First stretch of the UFES main campus ring road, which comprises a sharp curve. (c) Second stretch of the UFES main campus ring road, which comprises a series of smoother curves.

The speed limitation of 25 km/h for the PID approach and 37 km/h for the N-MPC approach are defined by the behavior of IARA, above these speeds the oscillation in the steering are too high and may cause the car to leave the lane, in any of the stretches of UFES ring road.

6.3 Metric of the Steering Control Accuracy

We evaluated the difference between φ_t^d (desired steering angle) and φ_t (measured steering angle) using the root mean squared error (RMSE) metric given by:

$$\text{RMSE} = \sqrt{\frac{\sum_{t=1}^{n} \left(\varphi_t^d - \varphi_t \right)^2}{n}} \qquad (14)$$

where n is the total number of φ_t^d being considered, and for each φ_t^d there is a correspondent φ_t. It is different from the trajectory size because a complete experiment may contain multiple trajectories.

The RMSE metric was chosen because it is a frequently used measure of the differences between predicted and observed values. RMSE aggregates the magnitudes of the errors in predictions for various times into a single measure of the prediction accuracy. Also, RMSE compounds information about average error with information about variation in the errors.

6.4 IARA's Hardware

We developed the hardware and software of the Intelligent and Autonomous Robotic Automobile (IARA, Fig. 1). The IARA's hardware is based on a Ford Escape Hybrid, which was adapted by Torc Robotics (http://www.torcrobotics.com) to enable electronic actuation of the steering, throttle and brakes; reading the car odometry; and powering several high-performance computers and sensors. IARA has one Velodyne HDL 32-E Light Detection and Ranging (LIDAR); one Trimble RTK GPS; one Xsens MTi IMU; one Point Grey Bumblebee and two Point Grey Bumblebee XB3 stereo cameras; and two Dell Precision R5500 computers.

6.5 IARA's Software

IARA's software is composed of many modules and the five main ones are: mapper [MUT16], localizer [VER15] [VER16], motion planner [CAR16], obstacle avoider [GUI16] and controller. N-MPC is part of the controller module. Fig. 12 shows a simplified block dia-

gram of IARA's software and depicts how the controller module, and hence N-MPC, connects with the other five main modules.

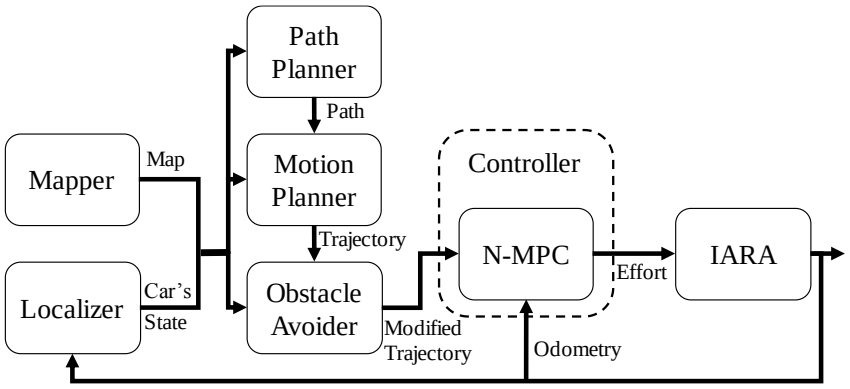

Fig. 12 – Block diagram of the five main modules of the IARA's software.

The mapper continuously computes a map of the environment around IARA using a GraphSLAM estimator algorithm to integrate the data generated by the sensors, a 3D LiDAR Velodyne HDL-32E, an Inertial Measurement Unit and a low cost Global Position System [MUT16].

The localizer continuously estimates IARA's state relative to the origin of the map. It is based on the Particle Filter localization, which corrects particles' poses by applying map-matching between 2D global occupancy grid-maps computed offline and 2D local occupancy grid-maps constructed online. The localization system converts the dense 3D point clouds of the 3D LiDAR into sparse local maps [VER15, VER16].

The path planner module extracts the path from the current car state to the goal state from a road definition data file (RDDF), a sequence of car states logged while manually driving IARA. IARA's path planner follows the 2005 DARPA Grand Challenge pattern, instead of using an approach to obtain the path online.

IARA's Model-Predictive Motion Planner continuously computes a trajectory from the current IARA's state to the next goal state, using the path, the current car's state, a goal in the path and the map. The motion planner subsystem is able to compute smooth trajectories from its current position to the goal in less than 50 μs [CAR16].

IARA's obstacle avoider firstly receives as input an updated map, the current car's state and the trajectory. Secondly, the obstacle avoider simulates the trajectory. Finally, if the trajectory crashes into an obstacle, then the obstacle avoider decreases the linear velocity commands of the trajectory to prevent the accident [GUI16].

Finally, the controller is responsible to compute the acceleration, brake and steering wheel control commands that will make IARA execute the trajectory states, the closest possible. These control commands are modeled as the effort that will be applied in the system plant. The effort values vary in the range [-100, 100]. For the break and acceleration, in case of -100, the maximum break effort is applied and, in case of 100, the maximum acceleration effort is applied. For the steering plant, in case of -100, the maximum acceleration counter-clockwise is applied and, in case of 100, the maximum acceleration clockwise is applied.

Additional modules of IARA's software include: health monitor, that checks the working condition of all modules and reinitiates those that are not operating correctly; behavior selector, that sets the mode of operation depending on current conditions; logger, that provides sensor data logs; simulator, that simulates IARA using sensor data logs [DES16]; among others.

6.6 Carmen

The modules mentioned above were implemented in the Carnegie Mellon Robot Navigation Toolkit (CARMEN). CARMEN is an open source software collection for mobile robot control. It was designed to provide basic navigation primitives, including mapping, localization, path planning, obstacle avoidance, logging, and base and sensor control. It was created by the Carnegie Mellon University (http://carmen.sourceforge.net/) and, since 2011, it has been extended and maintained by the *Laboratório de Computação de Alto Desempenho* (High Performance Computing Laboratory - LCAD) at the *Universidade Federal do Espírito Santo* (Federal University of Espírito Santo UFES) (https://github.com/LCAD-UFES/carmen_lcad).

CARMEN implements a modular software architecture, in which robot functionalities are divided in separate modules. It provides scalability and reliability, since the modules can be executed taking advantage of multi-core computers, or even in different computers, and, if a module fails, the others will not stop necessarily. The communication between modules is made by the Inter Process Communication (IPC) platform. IPC uses the publish-subscribe scheme, which allows to send and receive complex data structures even between computers using TCP/IP protocol. Fig. 8 shows an example of CARMEN's simulation environment, including the IARA's representation, the obstacle map, the global path and the trajectory.

7 EXPERIMENTAL RESULTS

This chapter presents the experimental results performed to validate the work developed in this research. Section 7.1 presents the PID parameters tuning process and the neural network system plant model, N-SPM, of N-MPC tuning process. Section 7.2 presents IARA's steering control results in different testing situations.

7.1 IARA's Steering Controllers Tuning

In Section 7.1.1, we describe the PID control subsystem tuning process using the Ziegler Nichols tuning rules described in Section 3.1.1. In Section 7.1.2, we describe the neural steering plant model, N-SPM, of MPC tuning process using the genetic algorithm described in Section 5.1.1.

7.1.1 IARA's PID Steering Controller Tuning

The parameters of the IARA's PID Steering Controller – K_p, K_i and K_d – were tuned using the Ziegler Nichols method described in Section 3.1.1. Firstly, to collect the data to build the step response of open-loop system, we sent to IARA's Steering Plant a fixed effort with value of 20 until the steering settle in a curvature. This process was repeated four times to obtain a good average of the process characteristics. For each one of the four open-loop step response data, a graph from the data was obtained and a tangent line in the inflection point was drawn, as shown in Fig. 13.

Using the Ziegler Nichols tuning rules presented in Section 3.1.1, we obtained the parameters Δy_t, Δu, t_0, t_1 and t_2, and, using Equations (2), (3) and (4), we can calculate the process characteristic parameters process gain K, dead time θ and time constant τ, from each one of the graphs presented in Fig. 13 as shown in Table 4 respectively. An average value for each parameter was calculated to obtain a better approximation.

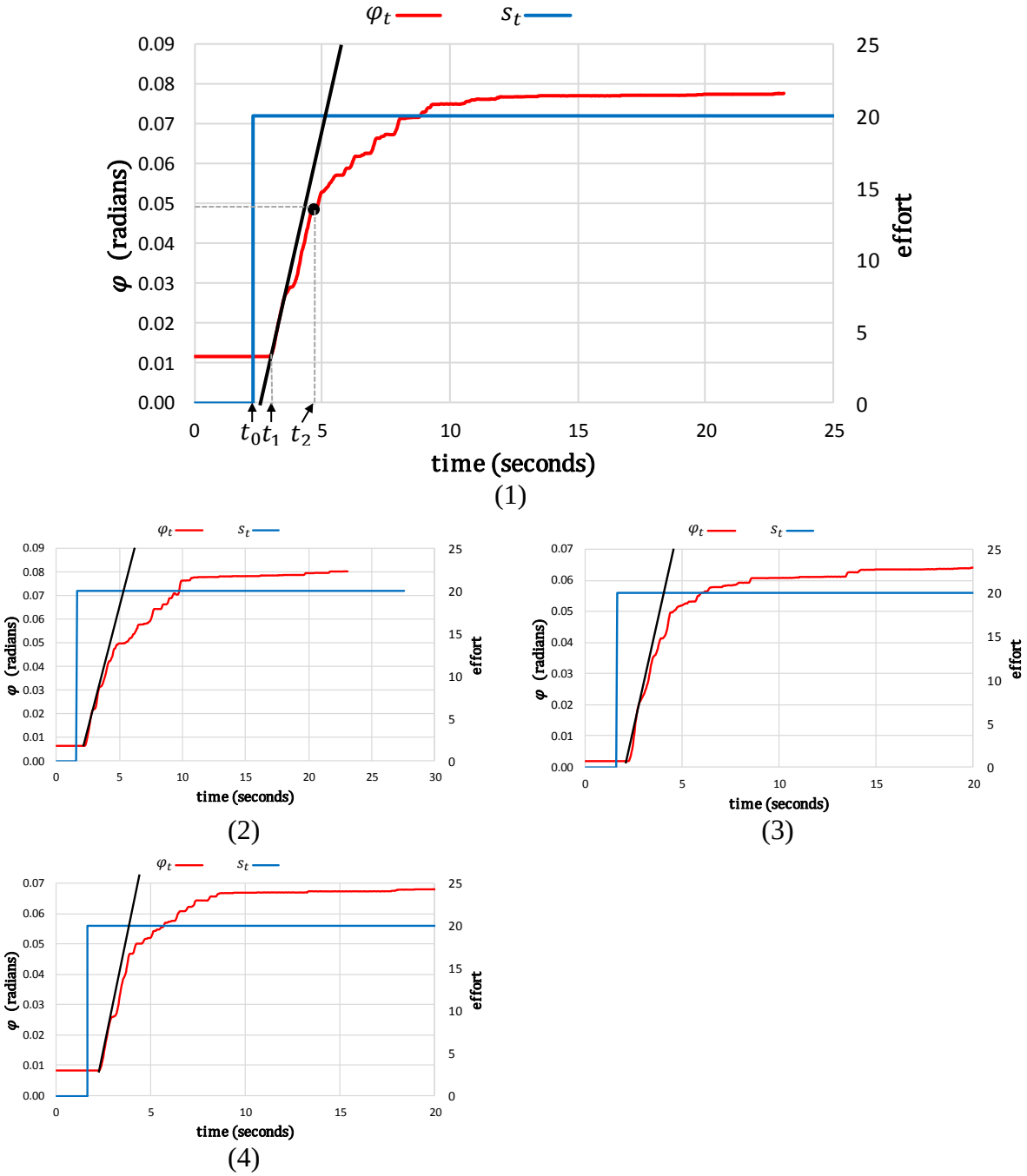

Fig. 13 – Four different IARA's open-loop step response. The red curve denotes φ_t and the blue curve denotes s_t. The black line is the derivative line in the inflection point of φ_t curve. t_0 denotes the time the step was applied, t_1 denotes the time φ_t starts responding and t_2 the time φ_t reaches 0.63% of its maximum value.

Table 4 – Four different values K, θ and τ, obtained from the graphs shown in figure Fig. 13.

	K	Θ	T
1	0.0036291	0.66551	2.616715
2	0.00297765	0.60035	1.51849
3	0.00297655	0.53341	1.54684
4	0.00330685	0.55983	1.77452
Average	**0.00314225**	**0.58009**	**1.66068**
Standard Deviation	**0.000312453**	**0.057507498**	**0.514637117**

From the process characteristics parameters presented in Table 4, the PID parameters – K_p, K_i and K_d – can be calculated using the equations presented in Table 2 for PID controller type.

Table 5 – The PID parameters – K_p, K_i and K_d – obtained from the average values of K, θ and τ, using the Ziegler Nichols tuning method.

K_p	K_i	K_d
1093.279205	942.3358489	317.1001671

We compared the PID parameters found using the Ziegler Nichols tuning method against the old parameters that we have found manually, by varying K_p, K_i and K_d values, and evaluating graphically the difference between φ_t^d and φ_t. The old PID parameters are shown in Table 6.

Table 6 – Old PID parameters – K_p, K_i and K_d – manually obtained by varying their values and evaluating graphically the difference between φ_t^d and φ_t.

K_p	K_i	K_d
1250	600	25

Fig. 14 shows graphically the performance of Ziegler Nichols PID parameters while driving IARA's Steering Plant autonomously with maximun velocity of 20 km/h. The RMSE between φ_t^d and φ_t in this graph is 5.51×10^{-3} radians.

Fig. 15 shows graphically the performance of old manually tuned PID parameters while driving IARA's Steering Plant autonomously with maximun velocity of 20 km/h. The RMSE between φ_t^d and φ_t in this graph is $7{,}68\times10^{-3}$ radians.

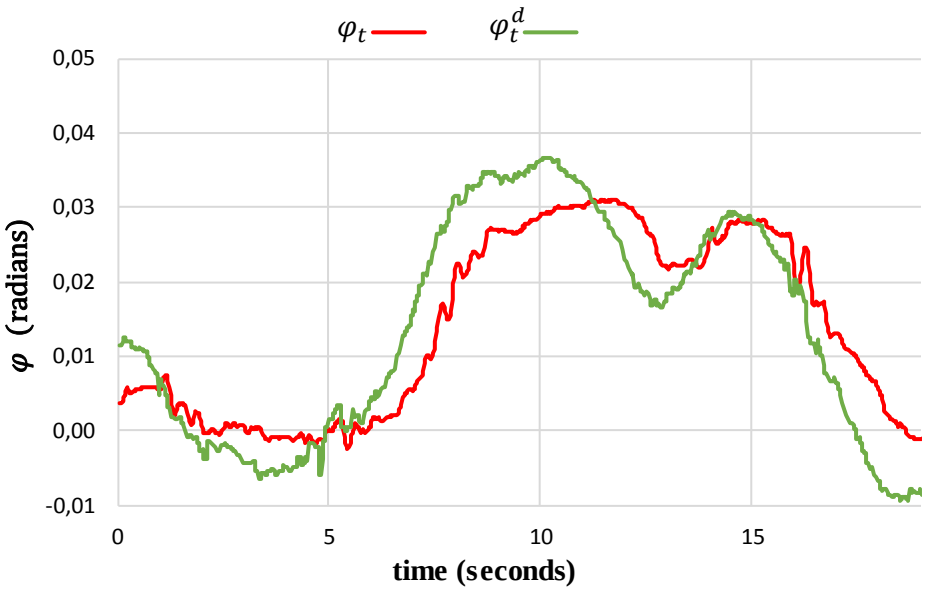

Fig. 14 – Results of the performance evaluation of the Ziegler Nichols PID parameters while driving IARA autonomously. The green curve denotes φ_t^d and the red curve denotes φ_t.

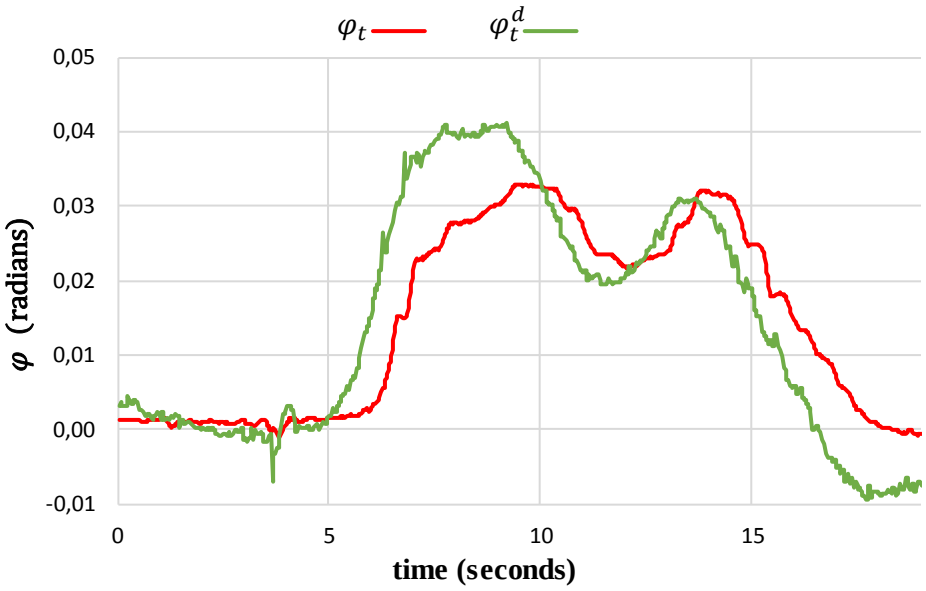

Fig. 15– Results of the performance evaluation of the old PID parameters while driving IARA autonomously. The green curve denotes φ_t^d and the red curve denotes φ_t.

The results presented in this section shows a reduction of more than 28,3% in PID's RMSE by comparing Ziegler Nichols parameters with those adjusted manually.

7.1.2 IARA's N-MPC Steering Controller Tuning

The parameters of N-SPM (employed by the IARA's N-MPC Steering Controller) – number of layers, neurons per layer and type of neurons, among others (see Section 5.1.1) – were tuned using the genetic algorithm (GA) described in Section 3.2.2. The GA found a solution in the first generation, with nine of the 150 randomly created individuals achieving a RMSE over validation below the desired error, $1x10^{-2}$ radians. These nine individuals were subsequently evaluated with the testing set, and the best individual was chosen to be used as IARA's neural based steering plant model N-SPM. The configuration of the best individual is listed in Table 7. The RMSE over the validation and testing dataset are $6.4x10^{-3}$ and $6.3x10^{-3}$ respectively.

Table 7 – Best configurations of neural networks found by the genetic algorithm.

Parameters	**AOC Network**
Number of inputs	80
Number of hidden layers	1
Number of neurons in the first hidden layer	50
Number of neurons in the second hidden layer	-
Activation function of hidden layers	Sigmoid Symmetric
Activation function of output layer	Linear
Learning Algorithm	Online Backpropagation
Learning Rate	0.5
Momentum	0.5
Learning Rate Decay	-
Maximum epochs trained	200

7.2 IARA's Steering Controllers Results

This section presents the comparison between the PID approach and N-MPC approach graphically and by the RMSE metric.

7.2.1 Results Derived from Trapezoidal and Sinusoidal Steering Control Inputs

The results presented in this subsection shows a comparison between PID and N-MPC approaches with a simple trajectory, composed of steering angles in a trapezoidal or sinusoidal wave form, which was computed empirically. Moreover, all planning modules were switched off. These experiments aim to evaluate the control approaches executing the same trajectory without interference of planning modules.

Fig. 16 shows the results of the performance evaluation of the PID Steering Controller while driving IARA's Steering Plant with a simple trajectory – composed of steering angles in a trapezoidal wave form and constant linear velocities of 5 km/h – along the straight area in the end of the second stretch of the UFES main campus ring road (Fig. 11(b)). The RMSE between φ_t^d and φ_t in this graph is 5.74×10^{-3} radians.

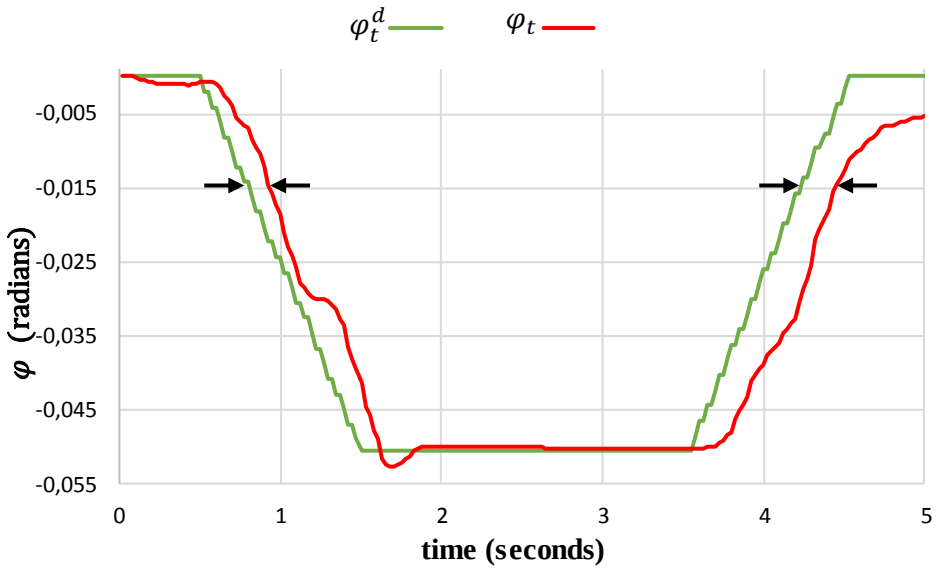

Fig. 16 – Results of the performance evaluation of the PID Steering Controller while driving IARA's Steering Plant with steering angles in a trapezoidal wave form along the straight area in the end of the second stretch of the UFES main campus ring road. The green curve denotes φ_t^d and the red curve denotes φ_t.

Fig. 17 shows the results of the performance evaluation of the N-MPC Steering Controller while driving IARA's Steering Plant with steering angles in a trapezoidal wave form along the straight area in the end of the second stretch of the UFES main campus ring road (Fig. 11(b)). The RMSE between φ_t^d and φ_t is 2.4×10^{-3} radians. A visual comparison of Fig. 16 and Fig. 17 clearly shows, that N-MPC can drastically reduce the impact of the plant delay, as indicated by black arrows in the graphs. In this experiment N-MPC approach achieved a RMSE reduction compared with PID of 58,2%.

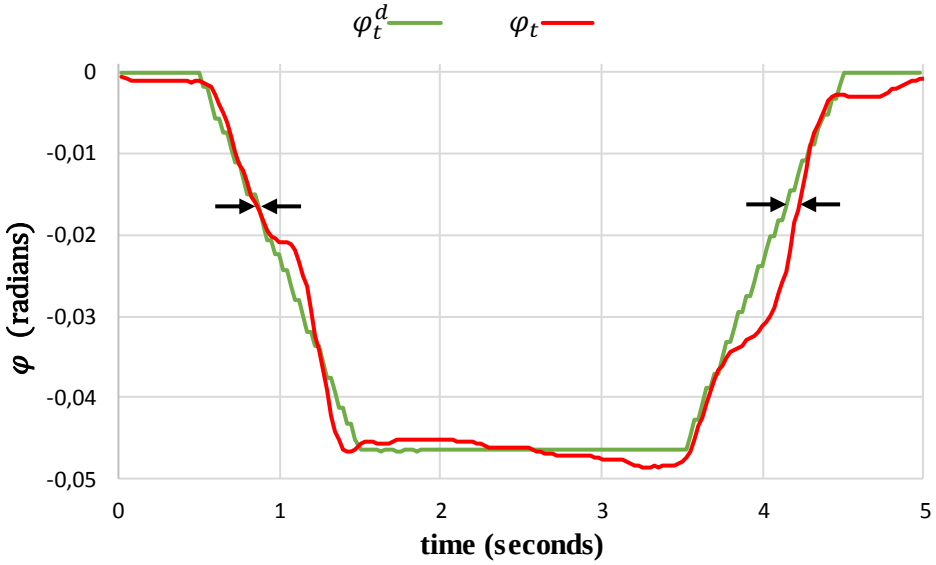

Fig. 17 – Results of the performance evaluation of the N-MPC Steering Controller while driving IARA's Steering Plant with steering angles in a trapezoidal wave form along the straight area in the end of the second stretch of the UFES main campus ring road. The green curve denotes φ_t^d and the red curve denotes φ_t.

Fig. 18 shows the results of the performance evaluation of the PID Steering Controller while driving IARA's Steering Plant with a simple trajectory – composed of steering angles in a sinusoidal wave form and constant linear velocities of 5 km/h – along the straight area in the end of the second stretch of the UFES main campus ring road (Fig. 11(b)). The RMSE between φ_t^d and φ_t in this graph is 8.28×10^{-3} radians.

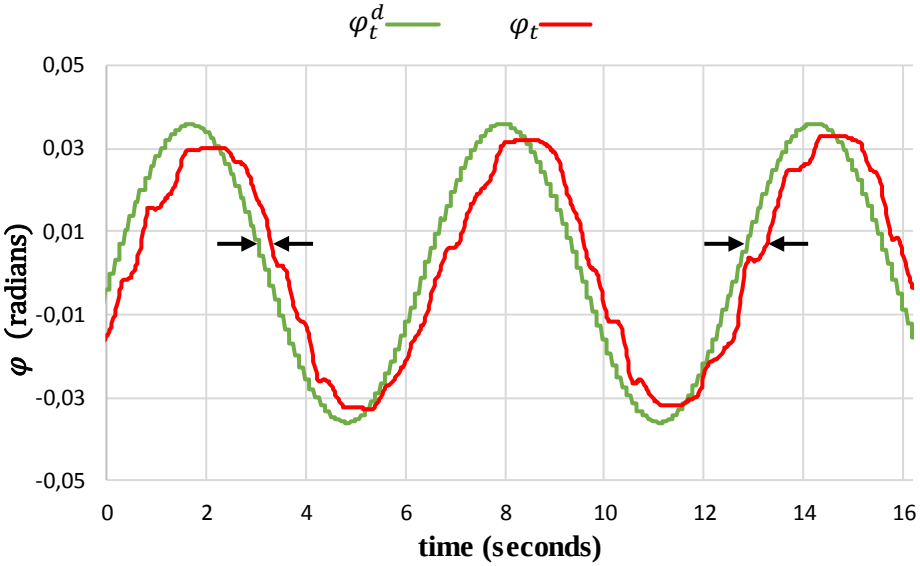

Fig. 18 – Results of the performance evaluation of the PID Steering Controller while driving IARA's Steering Plant with steering angles in a sinusoidal wave form along the straight area in the end of the second stretch of the UFES main campus ring road. The green curve denotes φ_t^d and the red curve denotes φ_t.

In Fig. 19 shows the results of the performance evaluation of the N-MPC Steering Controller while driving IARA's Steering Plant with steering angles in a sinusoidal wave form along the straight area in the end of the second stretch of the UFES main campus ring road (Fig. 11(b)). The RMSE between φ_t^d and φ_t is 3.55×10^{-3} radians. In this experiment N-MPC approach achieved a RMSE reduction compared with PID of 57,1%.

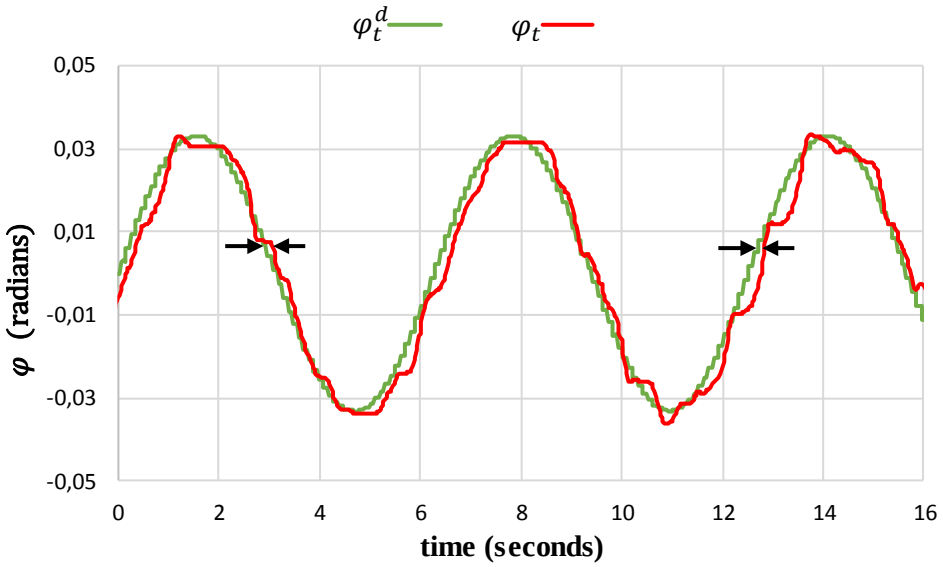

Fig. 19 – Results of the performance evaluation of the N-MPC Steering Controller while driving IARA's Steering Plant with steering angles in a sinusoidal wave form along the straight area in the end of the second stretch of the UFES main campus ring road. The green curve denotes φ_t^d and the red curve denotes φ_t.

7.2.2 Results Derived from IARA's Standard Operation Mode

The video available at https://youtu.be/iyKZV0ICysc shows IARA being driven autonomously along the testing sites, and the graphical results of the performance evaluation of the PID and N-MPC Steering Controllers while driving IARA's Steering Plant along the first (Fig. 11(b)) and second (Fig. 11(c)) stretches of UFES main campus ring road. For the experiments presented in this section IARA was driven autonomously along the two stretches of the UFES ring road with a maximum velocity of 37 km/h.

In Fig. 20, we show graphically the performance of the PID Steering Controller in first stretch of the UFES main campus ring road (Fig. 11(b)). The root mean squared error (RMSE) between φ_t^d (desired steering angle) and φ_t (measured steering angle) is 2.62×10^{-2} radians (This result corresponds to the period between 4 and 13 seconds of the video mentioned above; please note that the scales of the graphs that appear in the video are not the same as those in this work).

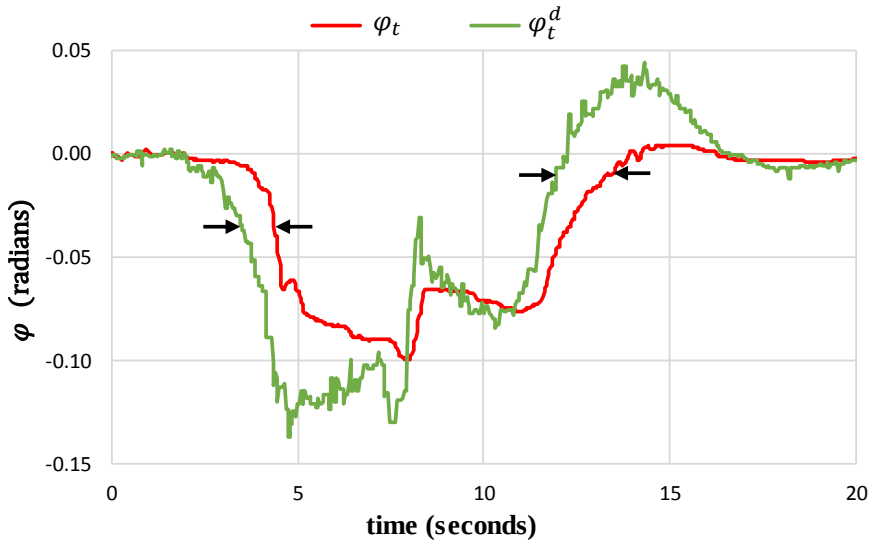

Fig. 20 – Results of the performance evaluation of the PID Steering Controller while driving IARA's Steering Plant along the first stretch of the UFES main campus ring road. The green curve denotes φ_t^d and the red curve denotes φ_t.

In Fig. 21, we show graphically the performance of the N-MPC Steering Controller in first stretch of UFES main campus ring road. The RMSE between φ_t^d and φ_t in this case is $0{,}73\times10^{-2}$ radians, which represents an error reduction of 72% if compared with the PID error (This result corresponds to the period between 23 and 32 seconds of the video mentioned above). The main cause of the errors of both controllers is the steering plant delay, as can be seen in Fig. 20 and Fig. 21. A visual comparison of these figures clearly shows, however, that N-MPC can drastically reduce the impact of the plant delay, as indicated by black arrows in the graphs. A visual comparison of Fig. 20 and Fig. 21 clearly shows, that N-MPC can drastically reduce the impact of the plant delay, as indicated by black arrows in the graphs. In this experiment N-MPC approach achieved a RMSE reduction compared with PID of 72,1%.

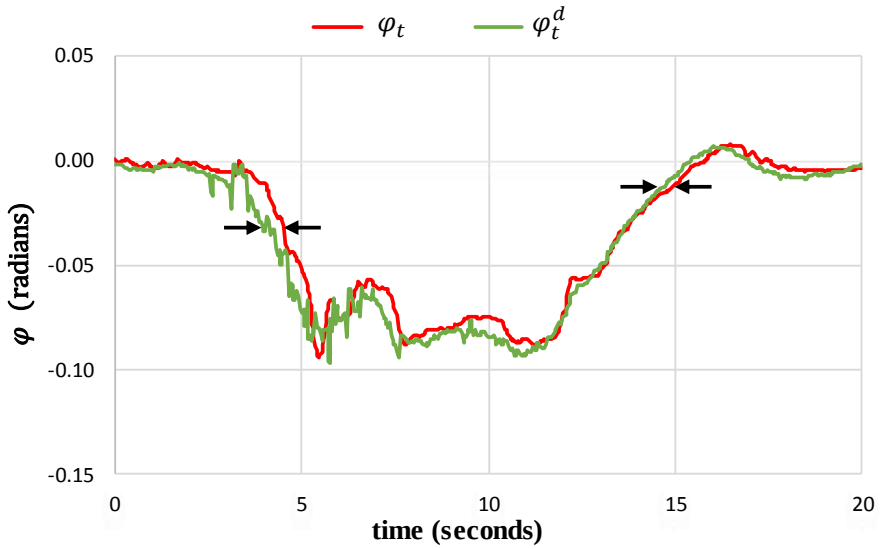

Fig. 21 – Results of the performance evaluation of the N-MPC Steering Controller while driving the IARA's Steering Plant along the first stretch of the UFES main campus ring road. The green curve denotes φ_t^d and the red curve denotes φ_t.

Fig. 22 shows the results of the performance evaluation of the PID Steering Controller while driving IARA's Steering Plant along the second stretch of UFES main campus ring road (Fig. 11(c)). As in the previous graphs, in the graph of Fig. 22 the green curve denotes φ_t^d and the red curve denotes φ_t. The RMSE between φ_t^d and φ_t in this graph is 2.66×10^{-2} radians (This result corresponds to the period between 43 and 1:01 in the video).

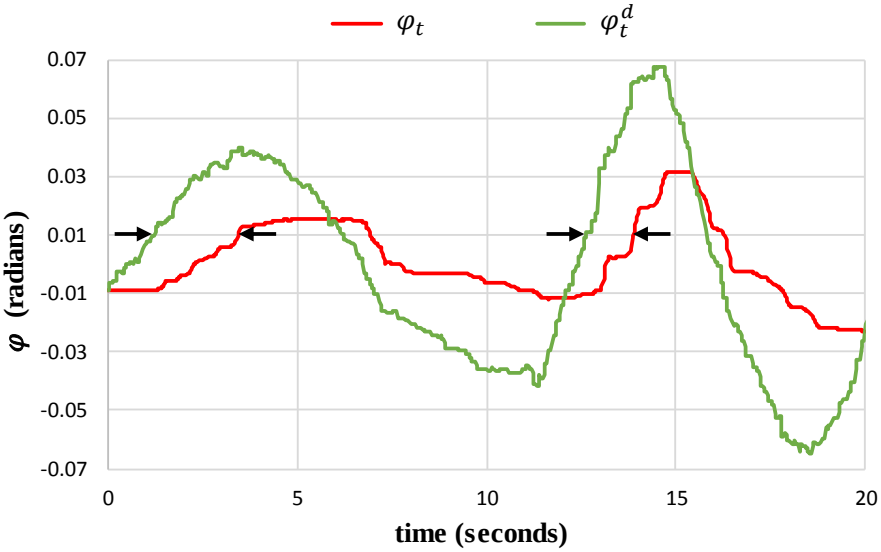

Fig. 22 – Results of the performance evaluation of the PID Steering Controller while driving IARA's Steering Plant along the second stretch of the UFES main campus ring road. The green curve denotes φ_t^d and the red curve denotes φ_t.

Fig. 23 shows the results of the performance evaluation of the N-MPC Steering Controller while driving IARA's Steering Plant along the second stretch of the UFES main campus ring

road (Fig. 11(c)). The RMSE between φ_t^d and φ_t is 0.9×10^{-2} radians, which is 66% below the errors of the PID Steering Controller in the same stretch of road (This result corresponds to the period between 1:20 and 1:38 seconds of the video). Again, the main cause of the errors of both controllers is plant delay, but N-MPC is much less impacted by it. A visual comparison of Fig. 22 and Fig. 23 clearly shows, that N-MPC can drastically reduce the impact of the plant delay, as indicated by black arrows in the graphs. In this experiment N-MPC approach achieved a RMSE reduction compared with PID of 66,2%.

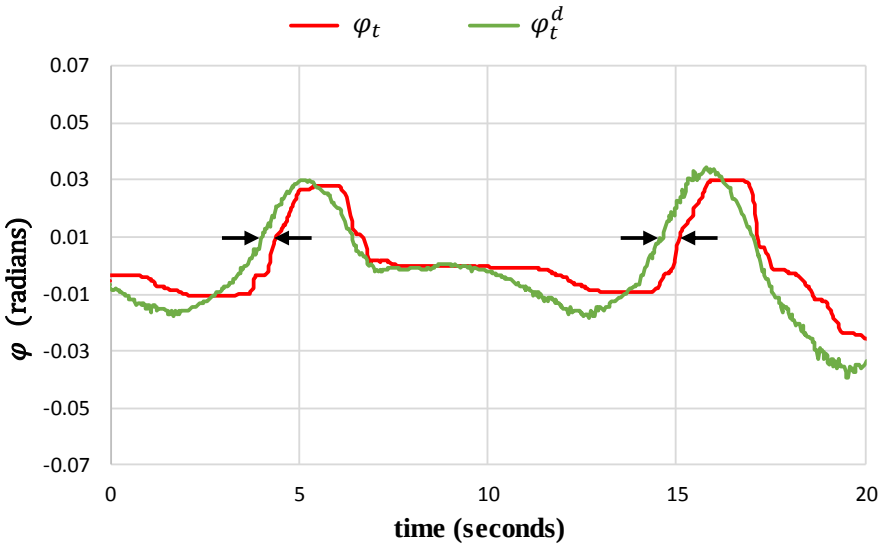

Fig. 23 – Results of the performance evaluation of the N-MPC Steering Controller while driving IARA's Steering Plant along the second stretch of the UFES main campus ring road. The green curve denotes φ_t^d and the red curve denotes φ_t.

7.3 Discussion

The results presented in Section 7.1.1, shows a reduction of more than 24% in PID's RMSE comparing Ziegler Nichols parameters with manual parameters. Comparing Ziegler Nichols tuning method with manually tuning, there is also a huge reduction in working hours. All these efforts on PID tuning demonstrate we did our best to achieve the better PID configuration we could, and demonstrates a fair comparison between PID and the proposed N-MPC.

Based on the results shown in Chapter 7 (see Section 7.2), the proposed IARA's N-MPC Steering Controller can drastically reduce the error between the desired and measured steering

angle – the error reduction was in the range of 57% to 72%, if compared with that of the IARA's PID Steering Controller.

The main cause of the errors of both controllers is the IARA's Steering Plant delay, which is caused by the drive by wire system comprised of the programmable logic controller, electric steering motor and motor driver installed by the Torc company. A visual comparison of the graphs presented in Section 7.2 clearly shows that N-MPC can drastically reduce the impact of the plant delay, as indicated by black arrows in the graphs. This was made possible by the Neural-Based Steering Plant Model (N-SPM), which is able to precisely simulate the steering plant. This allows N-MPC to anticipate the steering effort commands that need to be sent to the steering plant to reduce the controller errors.

8 CONCLUSIONS AND FUTURE WORK

This chapter presents the conclusions and future work. In Section 8.1, we preset the conclusions based on the experimental results of this research work. In Section 8.2, we present a critical analysis of this research work with direction for future work.

8.1 Conclusions

We proposed a Neural-Based Model Predictive Control (N-MPC) approach to tackle delays in the steering plant of the Intelligent and Autonomous Robotic Automobile (IARA). The previous PID steering control subsystem of IARA works well for speeds of up to 25 km/h. However, above this speed, IARA's Steering Plant delays are too high to allow proper operation with a PID control approach.

To tackle IARA's Steering Plant delays, we presented here the N-MPC approach. Due to the complexity of the dynamics of the IARA's Steering Plant response to stimuli, we tried and modeled it using a neural network and employed this neural model in the N-MPC approach. Experimental results showed that the N-MPC approach outperformed the PID control approach by reducing the impact of IARA's Steering Plant delays and allowing the autonomous operation of IARA at speeds of up to 37 km/h – an increase of 48% in maximum stable speed.

8.2 Future Work

Similar to the PID steering control subsystem, the IARA's PID velocity control subsystem does not work well for speeds higher than 25 km/h. Again, the reason is that delays of IARA's Velocity Plant are too high to allow proper operation with a PID control approach. Fig. 24 shows experimental results that allow estimating IARA's Velocity Plant delay, approximately 2 seconds, in the form of its response time. A possible direction for future work is the use of the N-MPC approach for implementing the IARA's velocity control subsystem.

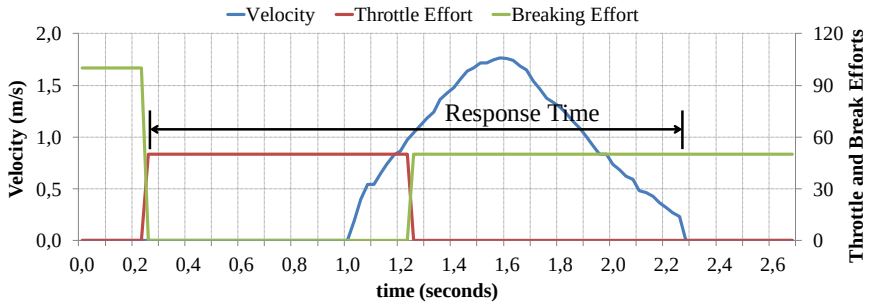

Fig. 24 – IARA's Velocity Plant response time.

IARA has a three-level control pipeline composed of the motion planning subsystem, the obstacle avoidance subsystem and the control subsystem. An alternative on how to design IARA's control pipeline would be the full integration of the motion planning, obstacle avoidance and control subsystems. The integrated control subsystem would produce directly the steering and throttle efforts, which could reduce the subsystem complexity and response time. Another direction for future research is the integration of IARA's motion planning subsystem with IARA's N-MPC steering control subsystem.

For the results shown in this work, N-SPM (employed by the IARA's N-MPC Steering Controller) was trained with data acquired while using IARA's PID Controller. This might have caused some noise in the IARA's Steering Plant model. Another direction for further research is to retrain the N-SPM with data acquired by N-MPC to try and improve its performance further. (For the results shown in this work, N-SPM was trained with data acquired while using IARA's PID steering control subsystem).

9 PUBLICATIONS

During the Master of Science in Computer Science period, five papers were published. Three of them related to the research developed in the Computer Engineering Undergraduate:

- R. Guidolini, C. Badue, M. Berger and A. F. De Souza, “A Simple Yet Effective Obstacle Avoider for the IARA Autonomous Car”, 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC 2016), Rio de Janeiro, Brazil, 2016.
- R. Guidolini, J. A. Guimarães, M. B. Andrade, T. O. dos Santos and E. Aguiar. Sistema visual para a reconstrução 3d de cenas em um ambiente submarino usando câmeras GoPro. In XXI Congresso Brasileiro de Automática (CBA 2016), Espirito Santo, Brasil, 2016.
- J. A. Guimarães, R. Guidolini, T. O. dos Santos, and E. Aguiar. A GoPro Underwater Environment Reconstruction System Based on Point Cloud Registration. In XII Workshop de Visão Computacional (WVC 2016), Mato Grosso do Sul, Brazil, 2016.

Two of the papers are related to the research developed in the Master of Science in Computer Science, which is described in this work:

- R. Guidolini, C. Badue, F. Mutz and A. F. De Souza, “Neural-Based Model Predictive Control for Tackling Steering Delays of Autonomous Cars”, 2017 IEEE 30th International Joint Conference on Neural Networks (IJCNN 2017), Anchorage, Alaska, USA, 2017.
- F. Mutz, V. Cardoso, T. Teixeira, L. F. R. de Jesus, M. A. Golçalves, R. Guidolini, J. Oliveira, C. Badue, A. F. De Souza, “Following the Leader using a Tracking System based on Pre-trained Deep Neural Networks”, 2017 IEEE 30th International Joint Conference on Neural Networks (IJCNN 2017), Anchorage, Alaska, USA, 2017.

10 REFERENCES

[ANG12] J. Angeles, “Dynamic Response of Linear Mechanical Systems: Modeling, Analysis and Simulation”, Springer, 2012.

[AST08] K. J. Aström and R. M. Murray, “Feedback Systems: An Introduction for Scientists and Engineers”, Princeton University Press, 2008.

[CAR16] V. Cardoso, J. Oliveira, T. Teixeira, C. Badue, F. Mutz, T. Oliveira-Santos, L. Veronese and A. F. De Souza, “A Model-Predictive Motion Planner for the IARA Autonomous Car”, arXiv preprint arXiv:1611.04552, 2016.

[DES16] A. F. De Souza, J. R. C. Silva, F. Mutz, C. Badue and T. Oliveira-Santos, “Simulating Robotic Cars Using Time-Delay Neural Networks”, 2016 International Joint Conference on Neural Networks (IJCNN 2016), Vancouver, Canada, 2016.

[FUN12] J. Funke, P. Theodosis, R. Hindiyeh, G. Stanek, K. Kritatakirana, C. Gerdes, D. Langer, M. Hernandez, B. Müller-Bessler and B. Huhnke, “Up to the Limits: Autonomous Audi TTS”, 2012 IEEE Intelligent Vehicles Symposium (IV 2012), Alcalá de Henares, Spain, pp. 541-547, 2012.

[GOT16] C. Götte, M. Keller, C. Rösmann, T. Nattermann, C. Haß, K. Glander, A. Seewald and T. Bertram, “A Real-Time Capable Model Predictive Approach to Lateral Vehicle Guidance”, 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC 2016), Rio de Janeiro, Brazil, 2016.

[GUI16] R. Guidolini, C. Badue, M. Berger and A. F. De Souza, “A Simple Yet Effective Obstacle Avoider for the IARA Autonomous Car”, 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC 2016), Rio de Janeiro, Brazil, 2016.

[JON75] K. A. De Jong, “An Analysis of Behavior of a Class of Genetic Adaptive Systems.”, Ph.D. Thesis, University of Michigan, Ann Arbor, Michigan, USA, 1975.

[KAT15] C. Katrakazas, M. Quddus, W.-H. Chen and L. Deka, “Real-Time Motion Planning Methods for Autonomous On-Road Driving: State-Of-The-Art and Future Research Directions”, Transportation Research Part C: Emerging Technologies, vol. 60, pp. 416-442, 2015.

[KOG16] A. Koga, H. Okuda, Y. Tazaki, T. Suzuki, K. Haraguchi and Z. Kang, “Realization of Different Driving Characteristics for Autonomous Vehicle by Using Model Predictive Control”, 2016 IEEE Intelligent Vehicles Symposium (IV 2016), Gothenburg, Sweden, pp. 722-728, 2016.

[LEV11] J. Levinson, J. Askeland, J. Becker, …, and S. Thrun, “Towards Fully Autonomous Driving: Systems and Algorithms”, 2011 IEEE Intelligent Vehicles Symposium (IV 2011), Baden-Baden, Germany, pp. 163-168, 2016.

[LI15] X. Li, Z. Sun, D. Cao, D. Liu and H. He, “Development of a New Integrated Local Trajectory Planning and Tracking Control Framework for Autonomous Ground Vehicles”, Mechanical Systems and Signal Processing, vol. 87, part

B, pp. 118–137, 2017.

[LIU14] L. Ljung, “System Identification Toolbox: User's Guide”, MathWorks Incorporated, 2014.

[MUT16] F. Mutz, L. P. Veronese, T. Oliveira-Santos, E. Aguiar, F. A. Auat-Cheeín and A. F. De Souza, “Large-Scale Mapping in Complex Field Scenarios Using an Autonomous Car”, Expert Systems with Applications, vol. 46, pp. 439-462, 2016.

[OGA10] K. Ogata, "Modern Control Engineering", Chapter 8.2, Pearson, 2010.

[ROS04] J. A. Rossiter, “Model-Based Predictive Control”, CRC Press, 2004.

[SMU11] J. F. Smuts, "Process Control for Practitioners", Chapter 6.3, pp. 137, OptiControls, 2011.

[SRI94] M. Srinivas and L. M. Patnaik, "Genetic algorithms: a survey", Computer, vol.27, no.6, pp. 17-26, 1994.

[TOR10] TORC Technologies, “ByWire XGV™ User Manual - Hybrid Escape Drive-by-Wire Platform”, version 1.5, TORC Robotics, Blacksburg, VA, 2010.

[VER15] L. P. Veronese, E. de Aguiar, R. C. Nascimento, J. Guivant, F. Auat Cheein, A. F. De Souza and T. Oliveira-Santos, “Re-Emission and Satellite Aerial Maps Applied to Vehicle Localization on Urban Environments”, 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2015), pp. 4285-4290, 2015.

[VER16] L. P. Veronese, F. A. Cheeín, T. O. Santos, F. W. Mutz, C. Badue and A. F. De Souza, “A Light-Weight Yet Accurate Localization System for Autonomous Cars in Large-Scale and Complex Environments”, 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC 2016), Rio de Janeiro, Brazil, 2016.

[ZHA12] P. Zhao, J. Chen, Y. Song, X. Tao, T. Xu and T. Mei, “Design of a Control System for an Autonomous Vehicle Based on Adaptive-PID”, International Journal of Advanced Robotic Systems, vol. 9, n. 244, 2012.

[ZIE14] J. Ziegler, P. Bender, M. Schreiber, …, and E. Zeeb, “Making Bertha Drive—an Autonomous Journey on a Historic Route”, IEEE Intelligent Transportation Systems Magazine, v. 6, n. 2, pp. 8-20, 2014.