Neural-Based Model Predictive Control for Tackling Steering Delays of Autonomous Cars*

Rânik Guidolini[1], Alberto F. De Souza[1], Filipe Mutz[2] and Claudine Badue[1]
[1]Departamento de Informática
Universidade Federal do Espírito Santo
Vitória, Brasil
{ranik, alberto, claudine}@lcad.inf.ufes.br

[2]Coordenadoria de Informática
Instituto Federal do Espírito Santo
Vitória, Brasil
filipe.mutz@ifes.edu.br

***Abstract*—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 a N-MPC approach. For speeds of up to 25 km/h, the IARA's steering plant delay is not a problem for the PID control approach. However, in higher speeds, it causes large steering oscillations, which prevent proper operation. For this, we modeled the IARA's steering plant using a neural network and employed the neural model in the N-MPC. Our experimental results showed N-MPC can drastically reduce the impact of IARA's steering plant delays, which allowed its autonomous operation at speeds of up to 37 km/h.**

Keywords—model predictive control; neural networks; autonomous cars.

I. INTRODUCTION

In order 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 [1]. A motion plan 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 [2] to the division of the pipeline in several stages [3]. We have developed an autonomous car, called Intelligent and Autonomous Robotic Automobile (IARA), which is illustrated in Fig. 1. IARA has a three-level control pipeline composed of: (i) a motion planning subsystem [3], (ii) an obstacle avoidance subsystem [4], and (iii) a control subsystem. In this paper, we examine two alternatives for the implementation of the IARA's steering control subsystem: the standard Proportional Integral Derivative (PID) control approach, 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 [5]. 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. These control subsystems work well for speeds of up to 25 km/h. However, above this speed, delays 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 measure the actual steering angle, i.e., the odometry sensor) are too high to allow proper operation.

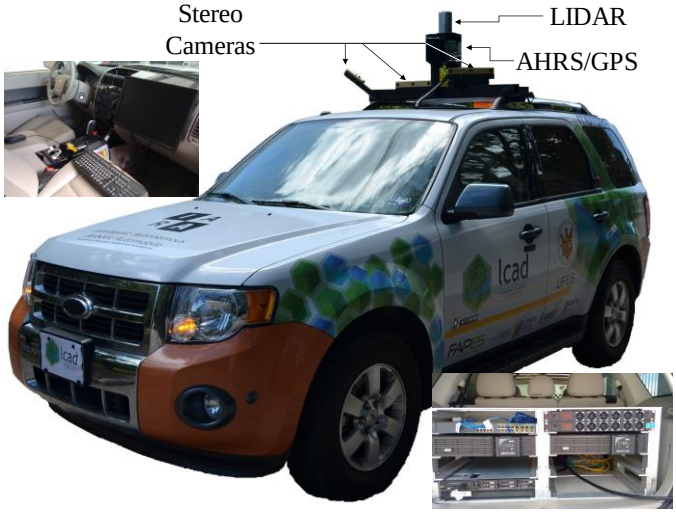

Fig. 1. Intelligent and Autonomous Robotic Automobile (IARA). At the top left corner, an inside view and, at the bottom right corner, the trunk, with IARA's computers, nobreaks and switch. A video that shows IARA operating autonomously is available at https://youtu.be/iyKZV0ICysc.

To tackle IARA's steering plant delays, in this paper we propose the use of a N-MPC approach. Standard MPC tries and

* Research supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Brasil (grants 12786/2013-1, 552630/2011-0) and Fundação de Amparo à Pesquisa do Espírito Santo (FAPES), Brazil (grant 48511579/2009).

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 [6]. Since the IARA's motion planning subsystem produces a motion plan about 5 seconds ahead in time [3] and the IARA's steering plant delay is about 0.6 seconds [7], 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 motion plan. However, standard techniques for predicting IARA's steering plant motion [8] 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 [7] and employed this neural-based steering model in our N-MPC steering controller. Our experimental results showed that N-MPC outperforms PID 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 a PID steering controller.

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 autonomous cars. The neural network was necessary to proper model the complexities of the steering plant, especially the IARA's Steering Plant delays.

This paper is divided as follows. After this introduction, we briefly present previous related work. In Section III, we describe our PID controller and, in Section IV, our N-MPC controller. In Section V, we present our experimental methodology and results. Finally, in Section VI, we conclude and present directions of future work.

II. Related Work

There are various approaches in the literature for the implementation of the control subsystem of autonomous cars. Funke et al. [9] 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 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. [10] employed the same combined feedforward and feedback control approach of Funke et al. [9]. 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. [11] 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. [12] 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. [13] 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. [14] 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. [2] 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 on an autonomous car simulator that does not take in consideration robot features (weight, sensorial and actuation noise, and balancing of the wheels, for example), environment features (road friction, for example), and interaction between them (how the friction between the wheels and the road influences the robot's acceleration, for example).

III. PID Control

Fig. 2 presents the architecture of the IARA's PID steering control subsystem, namely IARA's PID Controller. In the IARA's PID 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 [15], which uses AOC instead of the steering angle in their odometry message.

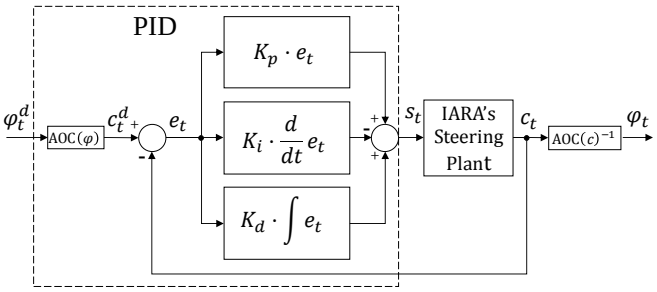

Fig. 2. Diagram of the architecture of IARA's PID Controller.

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 [7]

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

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

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

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 Controller, we convert φ to AOC using Equation (2) in order to make it comparable with the information provided by IARA's odometry sensor message. In Fig. 2, the block AOC(φ) converts φ to AOC.

IARA's PID Controller (Fig. 2) 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 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 tuned the constants K_p, K_i and K_d manually by varying their values and evaluating graphically the difference between φ_t^d and φ_t. Finally, c_t is converted to φ_t, which is sent to other subsystems of IARA. In Fig. 2, the block AOC(c)$^{-1}$ converts c_t to φ_t.

IARA's PID 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. 3 shows experimental results that allow estimating IARA's Steering Plant delay in the form of its response time [7].

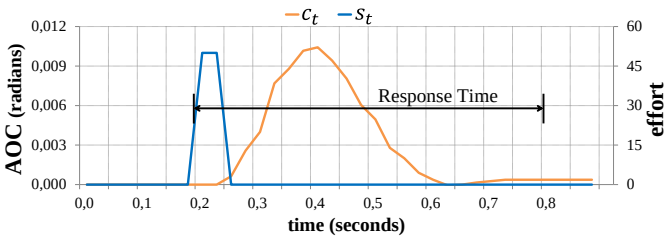

Fig. 3. IARA's Steering Plant response time [7].

In Fig. 3, 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 constants 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 [16]. As Fig. 3 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 our PID 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 Φ^d and Φ^p), anticipating the steering wheel efforts, s_t, that will counteract the Steering Plant delay when necessary and possible.

IV. Neural-Based Model Predictive Control (N-MPC)

A. 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 [7]. 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. 4 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. 4). 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 hidden layers, activation function of hidden layers, activation function of output layer, number of neurons in the hidden layer, learning algorithm, learning rate, momentum, learning rate decay and maximum epochs to train. All these parameters are selected using a genetic algorithm. For more details on the search space of all these parameters and their best configurations, please refer to [7]. The best individual (network) can then be used in the test phase (please see details of how we have implemented N-SPM in [7]). We note that the training phase is computer demanding, but it is executed offline.

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.

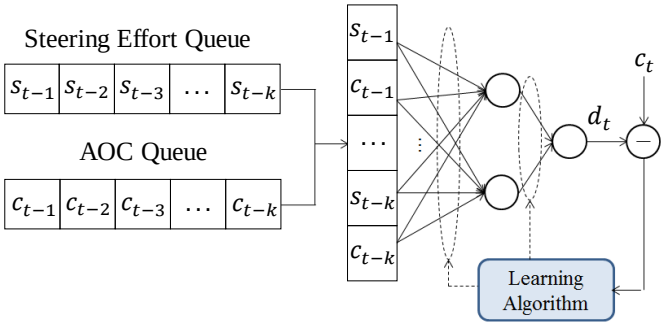

Fig. 4. Diagram of N-SPM. The N-SPM neural network receives as input a time series of executed s_t, $S_t = \{s_{t-1}, s_{t-2}, \dots, s_{t-k}\}$, and measured c_t, $C_t = \{c_{t-1}, c_{t-2}, \dots, c_{t-k}\}$, at k past time instants, and outputs a prediction of c_t, d_t.

In previous work [7], 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 mean squared error of 4.0×10^{-5} radians^2. Even with a large input, the neural network is fast enough to run in real time with an average response time of 5.0×10^{-6} seconds.

B. N-MPC Architecture

Fig. 5 shows a diagram of the architecture of IARA's N-MPC steering control subsystem, namely IARA's N-MPC Controller. The IARA's N-MPC Controller has three cycles: Control Cycle, Optimization Cycle and N-SMP Prediction Cycle. A Control Cycle encompasses various Optimization Cycles, and an Optimization Cycle encompasses various N-SMP 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. 5). This Optimizer may require several optimization cycles to achieve an optimal value of k_1.

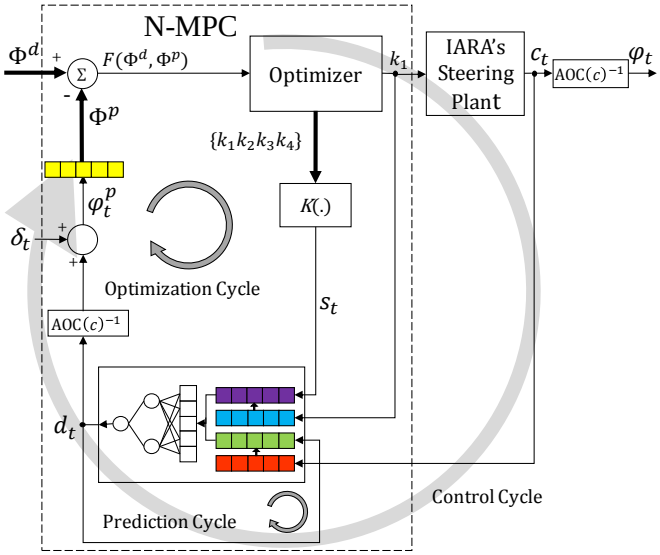

Fig. 5. Diagram of the architecture of IARA's N-MPC Controller.

At each **Optimization Cycle** (Fig. 5), Φ^d is compared with Φ^p according to the cost function

$$F(\Phi^d, \Phi^p) = \frac{1}{|\Phi^p|}\sum_{t=1}^{|\Phi^p|} \left| \varphi_t^d - \varphi_t^p \right|. \quad (3)$$

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. 5). $K(.)$ 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. 5) and the AOC Queue (green queue in Fig. 5) 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. 5) and the Measured AOC Queue (red queue in Fig. 5), 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. 5).

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-SMP 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. 5). The N-SMP 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 [3] at a rate of 20 Hz. Fig. 6 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.

The graph of Fig. 7 illustrates the operation of IARA's N-MPC 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. 7 shows previous values of φ_t, which are derived from measured values of c_t; see Fig. 5). A series of Optimization Cycles (and associated N-SMP Prediction Cycles) of this Control Cycle produces the values of the Prediction Horizon of Fig. 7.

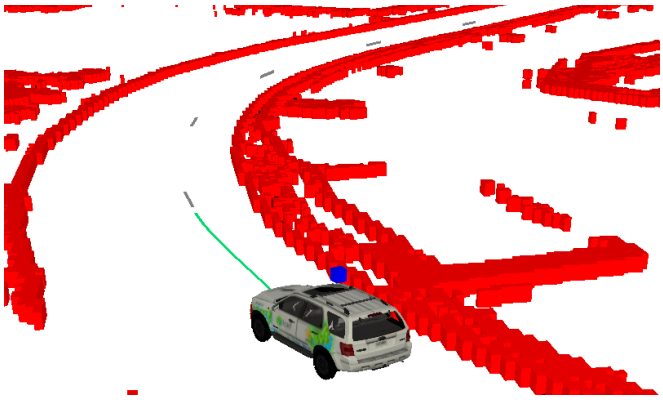

Fig. 6. Current desired trajectory, P. The green curve denotes P, which starts at the current position of IARA. The grey traces denote the goals and the red cubes denote obstacles, which, as a whole, form the map.

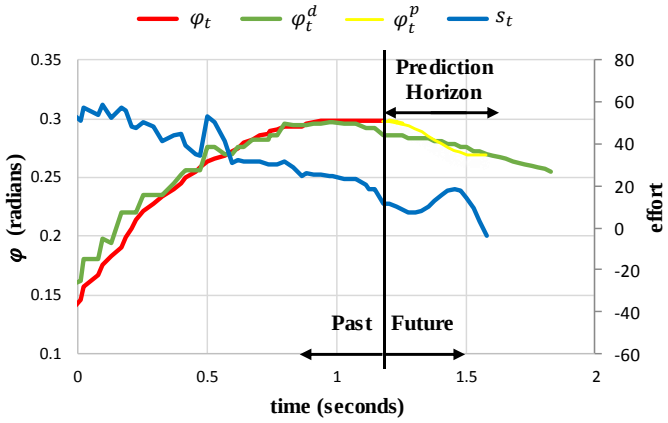

Fig. 7. Operation of the IARA's N-MPC Controller using real-world data. The vertical line splits the graph in two parts: Past and Future. In the Past side, the blue curve denotes executed s_t and the red curve denotes φ_t computed from measured c_t at k past time instants. In the Future side, the green 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).

At each N-SMP Prediction Cycle, a s_t taken from $K(.)$ (blue curve in the Future side of Fig. 7) 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. 7), using Equation (3). 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. 8 is a crop of Fig. 7 that shows more details of $K(.)$.

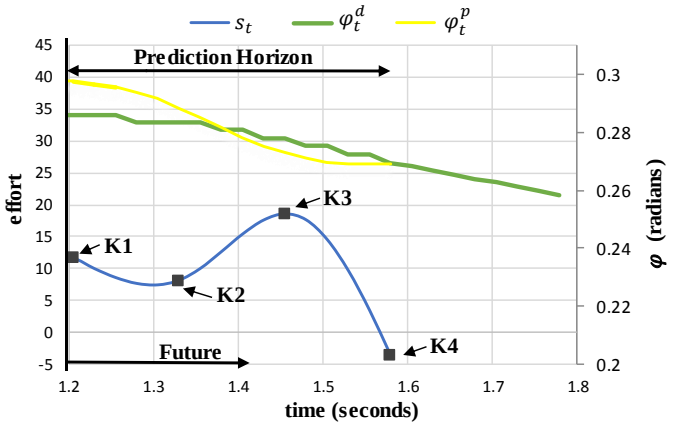

Fig. 8. Crop of Fig. 7 that shows more details of the $K(.)$. The y axes have changed sides and origin, but the data is still the same. Only the Future Horizon is shown and the parameters of $K(.)$ are indicated.

Although the N-SMP 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

$$\delta_t = \varphi_t - \varphi_t^p. \quad (4)$$

The use of δ_t allows for the compensation of the aforementioned plant modeling errors [6].

V. Experimental Evaluation

A. IARA's Hardware and Software

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.

IARA's software is composed of many modules and the five main ones are: *mapper* [17], *localizer* [18] [19], *motion planner* [3], *obstacle avoider* [4] and *controller*. N-MPC is part of the *controller* module. Fig. 9 shows a simplified block diagram of IARA's software and depicts how the *controller* module, and hence N-MPC, connects with the other five main modules. The *mapper* continuously computes a map of the environment around IARA [17]. The *localizer* continuously estimates IARA's state relative to the origin of the map [18] [19]. The *motion planner* continuously computes a trajectory from the current IARA's state to the next goal state [3]. The *obstacle avoider* continuously verifies and eventually changes the current trajectory in case it becomes necessary to avoid a collision [4]. Finally, the *controller* converts the control commands of the trajectories into acceleration, brake and steering efforts.

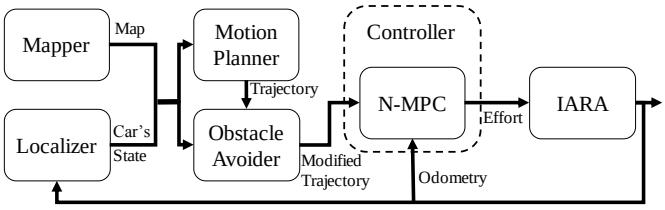

Fig. 9. Block diagram of the five main modules of the IARA's software

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 [7]; among others.

B. Experimental Methodology

To evaluate the performance of N-MPC, IARA was driven autonomously along two stretches of the ring road of the main campus of Federal University of Espírito Santo (*Universidade Federal do Espírito Santo* – UFES). Fig. 10 shows the Google Map view of the UFES main campus ring road. Fig. 10(a) shows the whole UFES main campus ring road, which has an extension of about 3.7 km, and Fig. 10(b) and Fig. 10(c) show the two stretches of the UFES main campus ring road where we conducted the experiments with N-MPC. As shown in Fig. 10(b), the first stretch comprises a sharp curve and, as shown in Fig. 10(c), the second stretch comprises a series of smoother curves.

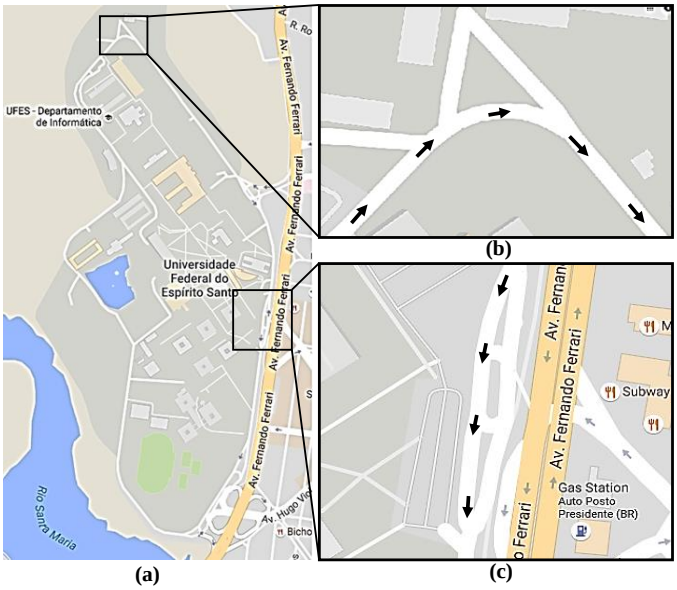

Fig. 10. (a) Google Map view of the UFES main campus ring road. (b) First stretch of the UFES main campus ring road, which comprises a sharp curve. (c) Second stretch of the UFES main campus ring road, which comprises a series of smoother curves.

IARA was driven autonomously along the two stretches of the UFES ring road with a maximum velocity of 37 km/h. We evaluated the difference between φ_t^d (desired steering angle)

and φ_t (measured steering angle) using the mean squared error (MSE) metric.

VI. Experimental Results

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. 10(b)) and second (Fig. 10(c)) stretches of UFES main campus ring road. In Fig. 11, we show graphically the performance of the PID controller in first stretch of the UFES main campus ring road (Fig. 10(b)). The MSE between φ_t^d (desired steering angle) and φ_t (measured steering angle) is 6.88×10^{-4} radians^2 (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 paper).

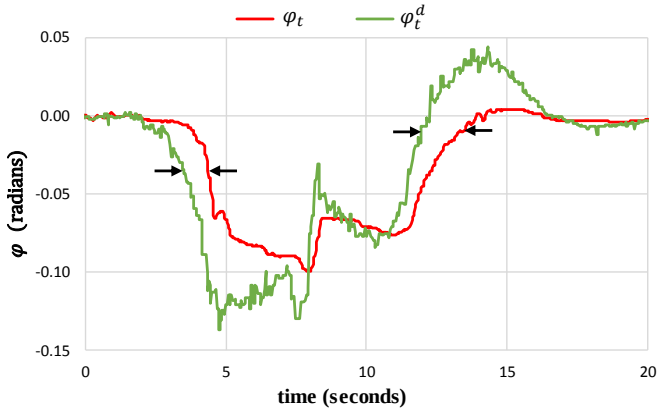

Fig. 11. Results of the performance evaluation of the PID steering controller while driving IARA's Steering Plant along the first stretch of the UFES main campus ring road. The green curve denotes φ_t^d and the red curve denotes φ_t.

In Fig. 12, we show graphically the performance of the N-MPC steering controller in first stretch of UFES main campus ring road. The MSE between φ_t^d and $\varphi_t$in this case is 5.3×10^{-5} radians^2, which represents an error reduction of more than one order of magnitude if compared with the PID error (This result corresponds to the period between 23 and 32 seconds of the video mentioned above).

It is important to note that the main cause of the errors of both controllers is the Steering Plant delay, as can be seen in Fig. 11 and Fig. 12. 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. This was made possible by our 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 IARA's Steering Plant to reduce the controller errors.

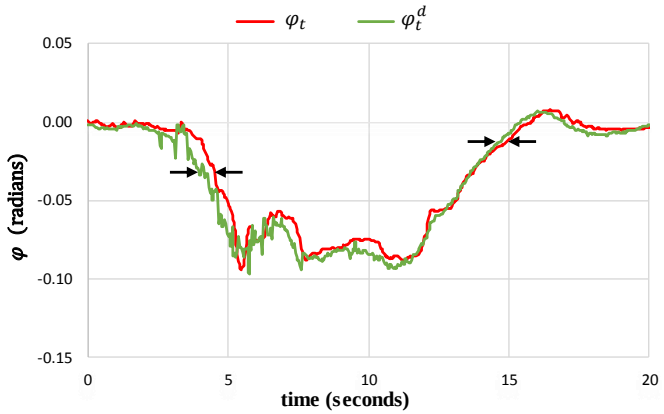

Fig. 12. Results of the performance evaluation of the N-MPC steering controller while driving the IARA's Steering Plant along the first stretch of the UFES main campus ring road. The green curve denotes φ_t^d and the red curve denotes φ_t.

Fig. 13 shows the results of the performance evaluation of the PID controller while driving IARA's Steering Plant along the second stretch of UFES main campus ring road (Fig. 10(c)). As in the previous graphs, in the graph of Fig. 13 the green curve denotes φ_t^d and the red curve denotes φ_t. The MSE between φ_t^d and φ_t in this graph is 7.1×10^{-4} radians^2 (This result corresponds to the period between 43 and 1:01 in the video).

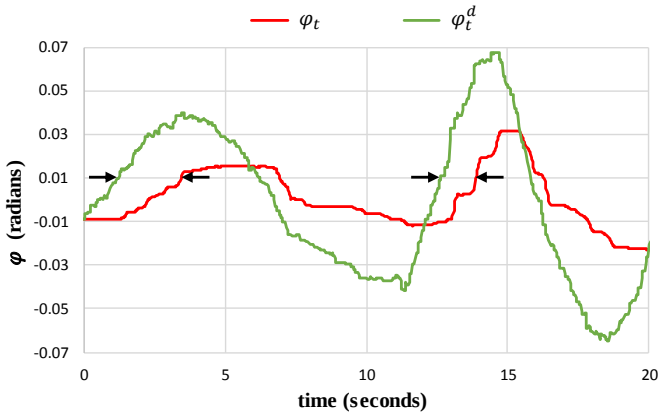

Fig. 13. Results of the performance evaluation of the PID steering controller while driving IARA's Steering Plant along the second stretch of the UFES main campus ring road. The green curve denotes φ_t^d and the red curve denotes φ_t.

Fig. 14 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. 10(c)). The MSE between φ_t^d and φ_t is 8.0×10^{-5} radians^2, which is again about one order of magnitude below the errors of the PID 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.

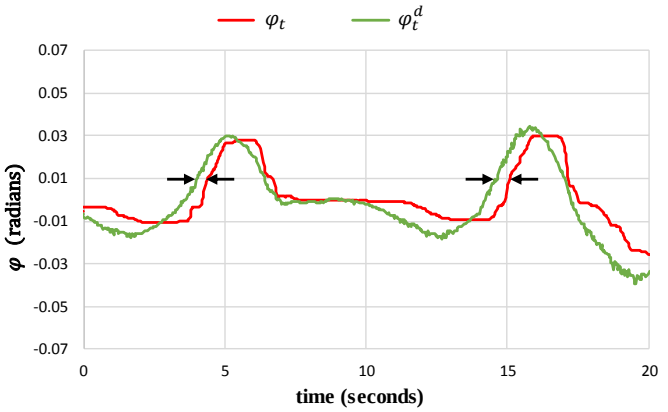

Fig. 14. Results of the performance evaluation of the N-MPC steering controller while driving IARA's Steering Plant along the second stretch of the UFES main campus ring road. The green curve denotes φ_t^d and the red curve denotes φ_t.

VII. Conclusions and Future Work

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. Our experimental results showed that N-MPC outperformed the PID control 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.

A possible direction for future work is the use of the N-MPC approach for implementing the IARA's velocity control subsystem. Another direction for future research is the integration of our motion planning subsystem with our N-MPC controller. Finally, 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 paper, N-SPM was trained with data acquired while using the PID steering controller).

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

[6] J. A. Rossiter, "Model-Based Predictive Control", CRC Press, 2004.

[7] 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.

[8] L. Ljung, "System Identification Toolbox: User's Guide", MathWorks Incorporated, 2014.

[9] 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.

[10] 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.

[11] 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. 2 44, 2012.

[12] 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.

[13] 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.

[14] 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.

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

[16] J. Angeles, "Dynamic Response of Linear Mechanical Systems: Modeling, Analysis and Simulation", Springer, 2012.

[17] F. Mutz, L. P. Veronese, T. Oliveira-Santos, E. Aguiar, F. A. Auat-Cheein 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.

[18] 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.

[19] 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.