Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

TS-MPC for Autonomous Vehicle using a Learning Approach ⋆

Eugenio Alcala [*,**] **Olivier Sename** [***] **Vicenç Puig** [*,**]
Joseba Quevedo [**]

[*] *Institut de Robòtica i Informàtica Industrial (CSIC-UPC). Carrer Llorens Artigas, 4-6, 08028 Barcelona (email: eugenio.alcala@upc.edu).*
[**] *Supervision, Safety and Automatic Control Research Center (CS2AC) of the Universitat Politécnica de Catalunya, Campus de Terrassa, Gaia Building, Rambla Sant Nebridi, 22, 08222 Terrassa, Barcelona.*
[***] *Univ. Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab, 38000 Grenoble, France.*

Abstract: In this paper, Model Predictive Control (MPC) and Moving Horizon Estimator (MHE) strategies using a data-driven approach to learn a Takagi-Sugeno (TS) representation of the vehicle dynamics are proposed to solve autonomous driving control problems in real-time. To address the TS modeling, we use the Adaptive Neuro-Fuzzy Inference System (ANFIS) approach to obtain a set of polytopic-based linear representations as well as a set of membership functions relating in a non-linear way the different linear subsystems. The proposed control approach is provided by racing-based references of an external planner and estimations from the MHE offering a high driving performance in racing mode. The control-estimation scheme is tested in a simulated racing environment to show the potential of the proposed approaches.

Keywords: Takagi-Sugeno approach, Model predictive control, Autonomous vehicles, Data-driven identification, Learning control

1. INTRODUCTION

In recent years, the amount of learning-based applications has increased immensely. Particularly, in the autonomous driving field, we have witnessed new approaches as the end-end driving where the goal consists on guiding the vehicle by means of using learning algorithms and input sensors data. In Sallab et al. (2017), a deep reinforcement learning framework is proposed that takes raw sensor measurements as inputs and outputs driving actions. In Bojarski et al. (2016), authors use a Convolutional Neural Network (CNN) to obtain the appropriate steering signal from the images of a single front camera.

Nowadays, from a control point of view, several strategies are starting to use learning tools to improve their capabilities while guaranteeing overall system stability. We have witnessed an advance in this field reaching learning techniques to adjust controllers, identify some parameters inside models or even control non-linear systems. In Lefèvre et al. (2015b,a), some solutions for controlling the longitudinal velocity of a car based on learning human behaviour are presented. Also, a Model Predictive Control (MPC) technique using deep CNN to predict cost functions from the camera input images is developed in Drews et al. (2017). In Rosolia and Borrelli (2017); Rosolia et al. (2017); Rosolia and Borrelli (2019), the authors propose a reference-free iterative MPC strategy able to learn from previous laps information.

Most of the last approaches were based on learning some policies to drive the vehicle independently of a physical model. In this work, we are interested on learning a realistic and accurate representation of the system dynamics to improve the control performance. In Kabzan et al. (2019), authors use a simple starting vehicle model which is enhanced on-line by learning the model error using Gaussian process regression and measured vehicle data.

In this paper, we propose the use of ANFIS approach, that is an adaptive neuro-fuzzy inference system, to learn the vehicle model. In the same manner as artificial neural networks, it works as a universal approximator (Jang, 1993). The main purpose of using ANFIS is to learn an input-output mapping based on input data. This tool has been widely used in other engineering fields (Ndiaye et al., 2018; Jaleel and Aparna, 2019).

The main contribution of this work is to model accurately a non-linear system as a structured Takagi-Sugeno (TS) representation of the vehicle by means of using machine learning tools and input data. In particular, this paper takes advantage of the properties of ANFIS tool to learn

⋆ The authors wish to thank the support received by the Spanish national project SCAV (ref. MINECODPI2017-88403-R and the European project SMART Project (ref. num. EFA153/16 Interreg Cooperation Program POCTEFA 2014-2020). The first author is supported by a FI AGAUR grant (ref 2017 FI B00433).

Fig. 1. Real picture of the vehicle used for simulation

a data-driven TS system which will be later used by a predictive optimal control to solve the driving problem.

The paper is structured as follows. Section 2 presents the testing vehicle used in simulation. Section 3 details the proposed learning-based method and its main components. Section 4 formulates the control and estimation problems. Section 5 introduces its application to a case study to assess the methodology, as well as various performance results. Finally, Section 6 presents several conclusions about the method suitability.

2. TESTING VEHICLE

The Berkeley Autonomous Race Car (Gonzales et al., 2016) (BARC[1]) is a development platform for autonomous driving to achieve complex maneuvers. This is a 1/10 scale RWD electric remote control (RC) vehicle (see Figure 1) that has been modified to operate autonomously. Mechanically speaking, this has been modified with some decks to protect the on-board electronics and sensors.

The non-linear model used in this chapter for simulating the BARC vehicle is presented as

$$\begin{aligned}
\dot{v}_x &= a_r + \frac{-F_{yf}\sin\delta - \mu g}{m} + \omega v_y \\
\dot{v}_y &= \frac{F_{yf}\cos\delta + F_{yr}}{m} - \omega v_x \\
\dot{\omega} &= \frac{F_{yf} l_f \cos\delta - F_{yr} l_r}{I} \\
\alpha_f &= \delta - \tan^{-1}\left(\frac{v_y}{v_x} - \frac{l_f \omega}{v_x}\right) \\
\alpha_r &= -\tan^{-1}\left(\frac{v_y}{v_x} + \frac{l_r \omega}{v_x}\right) \\
F_{yf} &= d\sin\left(c\tan^{-1}(b\alpha_f)\right) \\
F_{yr} &= d\sin\left(c\tan^{-1}(b\alpha_r)\right)
\end{aligned} \quad , \qquad (1)$$

where the dynamic vehicle variables v_x, v_y and ω represent the body frame velocities, i.e. linear in x, linear in y and angular velocities, respectively. The control variables δ and a are the steering angle at the front wheels and the longitudinal acceleration vector on the rear wheels, respectively. F_{yf} and F_{yr} are the lateral forces produced in front and rear tires, respectively. This considers the simplified "Magic Formula" model for simulating lateral tire forces where the parameters b, c and d define the shape of the curve. Front and rear slip angles are represented as α_f and α_r, respectively. m and I represent the vehicle mass and inertia and l_f and l_r are the distances from the vehicle center of mass to the front and rear wheel axes, respectively. μ and g are the static friction coefficient and the gravity constant, respectively. All the dynamic vehicle parameters are properly defined in Table 1.

[1] http://www.barc-project.com/

Table 1. Dynamic model parameters

Parameter	Value	Parameter	Value
l_f	0.125 m	l_r	0.125 m
m	1.98 kg	I	0.03 $kg\ m^2$
d	7.76	c	1.6
b	6.0	μ	0.1

In this work, with the aim of improving the simulation, Gaussian noise has been introduced in the measured variables as

$$n_{(\cdot)} \sim N(0, Co_{(\cdot)}) \qquad (2)$$

where $Co_{(\cdot)}$ is the signal covariance.

3. LEARNING THE TS MODEL

In this section, we present the modeling methodology used for obtaining the TS representation of the vehicle dynamic model. ANFIS is an adaptive neuro-fuzzy inference machine that is used for learning a particular structure from input-output data (Jang, 1993). More in detail, this modeling tool configures a neural network that learns from IO data the dynamic behaviour of the vehicle using back propagation technique and also employing the Recursive Least Squares (RLS) method for adjusting additional parameters.

The methodology consists on providing a dataset to the modeling algorithm (ANFIS). This is composed by vehicle states and inputs that represents a set of particular driving maneuvers guaranteeing rich enough data. Then, after a learning-based procedure, this provides a set of linear parameters, also known as consequent parameters, and a set of premise parameters or non-linear parameters that define the set of membership functions (MF) that provide the non-linear relationships between the different linear polynomials. One typical membership function is the generalized Gaussian function.

However, obtaining the TS representation of a system by means of using this resulting parameters is not trivial. The procedure is based on performing some inverse steps that ANFIS internally does. To address this problem we have to follow a set of reformulating steps. First, due to ANFIS algorithm can be only used for Multi-Input Single-Output (MISO) systems where just an output variable can be considered. Then, we split the system in MISO sub-systems obtaining as many sub-systems as state variables have the system. Our dynamic vehicle model is a third order system so three sub-systems will be obtained and three learning procedures will be carried out. Once the algorithm has computed conveniently the consequent and premise parameters for each one of the

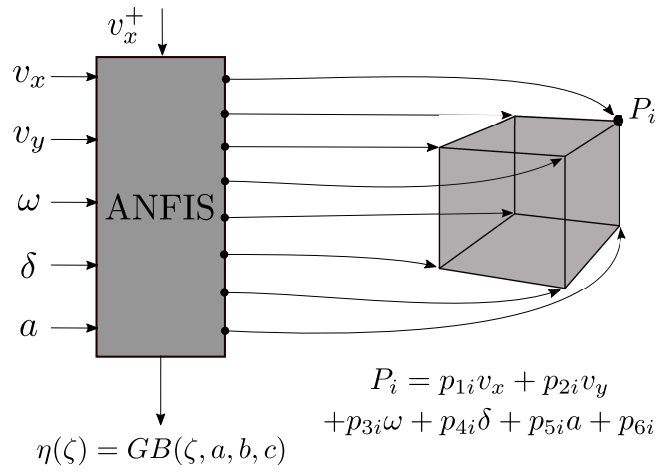

Fig. 2. Polytopic TS learning scheme for the v_x sub-system case

MISO sub-systems, we build the polytopic TS state-space representation for each one of these sub-systems. To do this, first, the polynomial representation of each sub-system is formulated as

$$\begin{aligned} &P_i = p_{1i}v_x + p_{2i}v_y + p_{3i}\omega + p_{4i}\delta + p_{5i}a + p_{6i} \\ &\forall i = 1, ..., N_v, \end{aligned} \tag{3}$$

where P_i stands for a linear polynomial representation of the dynamics of a sub-system at a particular states configuration, p_{ji}, $\forall j = 1, ..., N_\zeta$, are the consequent parameters obtained from ANFIS where N_ζ stands for the number of scheduling variables (see Figure 2) and N_v represents the number of polytopic vertexes.

Then, simply by reorganising the terms in (3) as

$$P_i = [\, p_{1i} \;\; p_{2i} \;\; p_{3i} \,]\, x + [\, p_{4i} \;\; p_{5i} \,]\, u + [\, p_{6i} \,] \tag{4}$$

the polynomial structure is transformed into the discrete-time state-space representation given by

$$x_i^+ = A_i x + B_i u + C_i \;, \forall i = 1, ..., N_v, \tag{5}$$

where, in order to easy the comprehension from a control point of view, P_i is represented as the sub-system i variable at the next discrete step (x_i^+) with symbol $^+$ representing the $k+1$ discrete step. A_i, B_i and C_i define the so-called *vertex systems*, $x = [\, v_x \;\; v_y \;\; \omega \,]^T$ and $u = [\, \delta \;\; a \,]^T$.

At this point, we use the obtained premise parameters to formulate the membership function. One of the most used is the generalized Gaussian Bell function (GB). This is defined by three parameters (a, b and c) as follows

$$\begin{aligned} &\eta_m(\zeta_o) = \frac{1}{1 + \frac{\zeta_o - c_{mo}}{a_{mo}}^{2b_{mo}}}, \\ &\forall m = 1, ..., N_{MF}, \forall o = 1, ..., N_\zeta, \end{aligned} \tag{6}$$

where ζ represents the ANFIS input vector of variables (from now on we will refer to them as scheduling variables) and N_{MF} and N_ζ represent the number of MF per scheduling variable and the number of scheduling variables, respectively. For a common case where N_{MF} is two, then, the normalized weights (μ_{N_i}) are computed following

$$\mu_i(\zeta) = \prod_{j=1}^{N_\zeta} \xi_{ij}(\eta_0, \eta_1), \forall i = 1, ..., N_v, \tag{7}$$

where $\xi_{ij}(\cdot)$ corresponds to any of the weighting functions that depend on each rule i. Then, using

$$\mu_{N_i}(\zeta) = \frac{\mu_i(\zeta)}{\sum_{j=1}^{N_v} \mu_j(\zeta)}, \forall i = 1, ..., N_v, \tag{8}$$

the normalized weights are obtained. Note that, each scheduling variable ζ_o is known and varies in a defined interval $\zeta_o \in [\underline{\zeta_o}, \overline{\zeta_o}] \in \mathbb{R}$. Finally, the polytopic TS model for each sub-system is represented as

$$x_j^+ = \sum_{i=1}^{N_v} \mu_{N_{ji}}(\zeta)(A_{ji}x + B_{ji}u + C_{ji}) \;, \forall j = 1, ..., N_G \;, \tag{9}$$

where N_G is the number of sub-systems.

Finally, for this work, we consider the third order dynamic system presented in (1), which implies $N_G = 3$. Then, the overall TS system is represented as

$$x^+ = \sum_{i=1}^{N_v} \begin{bmatrix} \mu_{N_{1i}} \\ \mu_{N_{2i}} \\ \mu_{N_{3i}} \end{bmatrix} \left(\begin{bmatrix} A_{1i} \\ A_{2i} \\ A_{3i} \end{bmatrix} x + \begin{bmatrix} B_{1i} \\ B_{2i} \\ B_{3i} \end{bmatrix} u + \begin{bmatrix} C_{1i} \\ C_{2i} \\ C_{3i} \end{bmatrix} \right). \tag{10}$$

From now on, with the aim of an easier reading, the system representation in (10) will be expressed as

$$x_{k+1} = \sum_{i=1}^{N_v} \mu_{N_i}(\zeta_k)(A_i x_k + B_i u_k + C_i) \;. \tag{11}$$

4. TS CONTROL AND ESTIMATION

In this section, we present the formulations for the MPC and MHE techniques using the TS formulation (see Figure (3)).

4.1 TS-MPC Design

When using a system dependent on some scheduling variables (TS system), computing the prediction of states behaviour in a certain horizon can be a challenging task, sometimes leading to errors in the instantiation since the real future behaviour is unknown.

In this work, we propose the use of data coming from two different locations to approximate in a better way the predictive instantiation and avoid a lack of convergence in the optimal procedure. On the one hand, data coming from a planner is used which represents the desired states behaviour for tracking the desired trajectory. On the other hand, predicted states from the past optimal realisation are also used to improve the TS model instantiation.

The model used in this section is the one presented in (11) where the vector of scheduling variables is defined as $\zeta := [\, v_x \;\; v_y \;\; \omega \;\; \delta \;\; a \,]$. The use of this model allows to formulate the MPC problem as a quadratic optimization problem that is solved at each time k to determine the control actions considering that the values of x_k and u_{k-1} are known

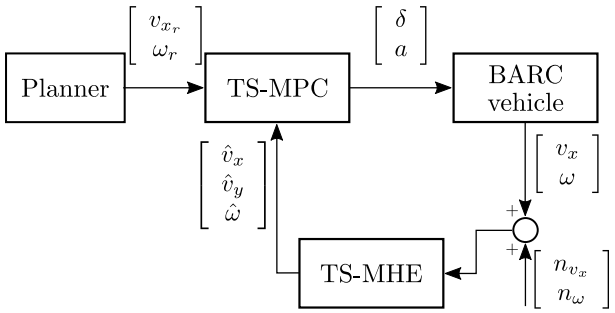

Fig. 3. Schematical view of the simulation set-up

$$
\begin{aligned}
\min_{\Delta U_k} \quad & \mathrm{J_k} = \sum_{i=0}^{H_p-1} \Big((r_{k+i} - x_{k+i})^T Q (r_{k+i} - x_{k+i}) \\
& + \Delta u_{k+i} R \Delta u_{k+i} \Big) + x_{k+H_p}^T P x_{k+H_p} \\
\text{s.t.} \quad & x_{k+i+1} = \sum_{j=1}^{N_v} \mu_{N_j}(\zeta_k)(A_j \hat{x}_{k+i} + B_j u_{k+i} + C_j) \\
& u_{k+i} = u_{k+i-1} + \Delta u_{k+i} \\
& \Delta U_k \in \Delta\Pi \\
& U_k \in \Pi \\
& x_{k+H_p} \in \chi \\
& y_e \in [\underline{y_e}, \overline{y_e}] \\
& x_{k+0} = \hat{x}_k \ ,
\end{aligned}
\tag{12}
$$

where $\Pi = \{u_k | A_u u_k = b_u, u_k \geq 0\}$ and $\Delta\Pi = \{\Delta u_k | A_{\Delta u} \Delta u_k = b_{\Delta u}, \Delta u_k \geq 0\}$ constraint the system inputs and their variations, respectively.

State vector is $x = [\, v_x \ v_y \ \omega \,]^T$, $\hat{x}$ is the estimated state vector, $r = [\, v_{x_r} \ 0 \ \omega_r \,]^T$ is the reference vector provided by the trajectory planner, $u = [\, \delta \ a \,]^T$ is the control input vector and H_p is the prediction horizon. The tuning matrices $Q \in \mathbb{R}^{3x3}$ and $R \in \mathbb{R}^{2x2}$, are positive definite in order to obtain a convex cost function. The closed loop stability is guaranteed by introducing $P \in \mathbb{R}^{3x3}$ and χ which represent the terminal set and the terminal constraint, respectively. Both are computed following the design presented in Alcala et al. (2019). Note that the time discretization is embedded inside the identification procedure such that the learned TS system is already in discrete-time.

4.2 TS-MHE Design

For the vehicle presented in Section 2, vehicle lateral velocity (v_y) is an unmeasurable variable and a necessary state to perform the closed-loop control of the vehicle. In this paper, we solve the estimation problem using the MHE approach. The aim of the MHE is to compute the current dynamic states by means of running a constrained optimization, using a set of past data and employing a system model for computing the current state. At this point, using the presented TS model in (11), we can run a quadratic optimization similar than in TS-MPC algorithm for estimating the current dynamic states as follows

$$
\begin{aligned}
\min_{\hat{X}_k} \quad & \mathrm{J_k} = \sum_{i=-H_p}^{0} \left(w_{k+i}^T Q w_{k+i} + s_{k+i}^T R s_{k+i} \right), \\
\text{s.t.} \quad & \hat{x}_{k+i+1} = \sum_{j=1}^{N_v} \mu_{N_j}(\zeta_k)(A_j \hat{x}_{k+i} + B_j u_{k+i} + C_j) \\
& + w_{k+i} \\
& y_{k+i} = C \hat{x}_{k+i} + s_{k+i} \\
& \hat{X}_k \in X_d \\
& \forall i = -H_p, ..., 0 \ ,
\end{aligned}
\tag{13}
$$

that is solved online for

$$
\hat{X}_k = \begin{bmatrix} \hat{x}_{k-H_p+1} \\ \hat{x}_{k-H_p+2} \\ \vdots \\ \hat{x}_{k+1} \end{bmatrix} \in \mathbb{R}^{H_p \times s} \ , \tag{14}
$$

where X_d is the constraint region for the dynamic states and its defined as $X_d = \{x_k | A_x x_k = b_x, x_k \geq 0\}$. H_p stands for the past data horizon and s the number of states. Matrices $Q = Q^T \in \mathbb{R}^{3x3}$ and $R = R^T \in \mathbb{R}^{2x2}$, are positive definite to generate a convex cost function and w and s represent the error of estimation and the process noise, respectively. State and input vectors are $\hat{x} = [\, v_x \ v_y \ \omega \,]^T$ and $u = [\, \delta \ a \,]^T$. Note that, unlike MPC technique, MHE strategy performs an optimization taken into account a window of past vehicle data.

5. RESULTS

The data-driven model identification carried out by the proposed approach is used to learn a state-space TS formulation of the real vehicle dynamics. In Figure 4, the membership functions learned for each input after the offline identification procedure are shown in the left side.

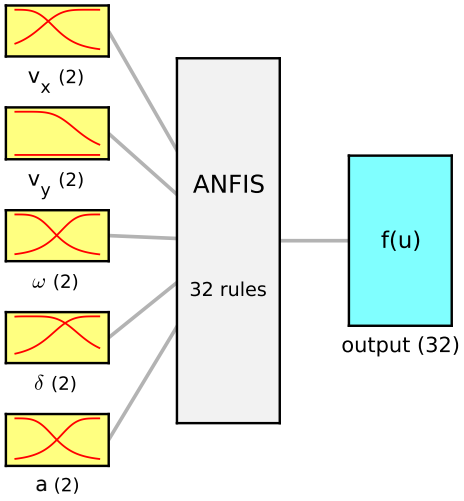

Fig. 4. Input-Output scheme for the v_x sub-system case

These represent the fuzzy rules that will be used online for obtaining the current state-space representation. Note that, since the discretization time is embedded in the input data of the learning procedure, the selection of a different sampling time is not allowed.

The way of evaluating the MPC and MHE strategies using the data-driven approach presented is through a

simulation scenario. In this, a racing situation is proposed where the autonomous scheme presented in Figure 3 is simulated.

First, at every sampling period, i.e. 30 Hz, the racing planner provides the references for the control strategy such that the vehicle will have to behave in a racing driving mode, what directly implies a more challenging control problem. Then, the TS-MHE optimal problem presented in (13) is solved for estimating the current vehicle vector state using past vehicle measurements. The next step is to instantiate the TS model matrices for the prediction stage using the approach presented in Section 2. Note that, both the planning evolution information as the previous optimal prediction are used to achieve a good guess of the future values of the scheduling vector ζ.

At this point, the quadratic optimal problem (12) is solved using the estimated state variables and the references coming from the trajectory planner. Once the optimal control actions (δ and a) are computed they are applied to the simulation vehicle presented in (1). As a consequence, the vehicle change its state and this is measured by the net of sensors. Besides, with the aim of adding more realistic conditions to the problem, white Gaussian noise magnitudes are added to measured states with zero mean and covariances

$$Co_{v_x} = 1 \times 10^{-6} \ , \ Co_{\omega} = 4 \times 10^{-8} \ . \tag{15}$$

Both TS-MPC and TS-MHE algorithms are coded in MATLAB framework. Yalmip and GUROBI (Gurobi Optimization, 2015) are used for solving a quadratic optimization problem and running on a DELL inspiron 15 (Intel core i7-8550U CPU @ 1.80GHzx8). In the controller, the tuning aims to minimize the longitudinal and angular velocity while computing smooth control actions. The diagonal terms of the weighting matrices in the cost function and prediction horizon of (12), found by iterative tuning until the desired performance is achieved, are

$$\begin{aligned} Q &= 0.65 \ [\, 0.4 \ \ 10^{-6} \ \ 0.6 \,], \\ R &= 0.35 \ [\, 0.7 \ \ 0.3 \,], \\ H_p &= 6 \ . \end{aligned} \tag{16}$$

The TS-MPC input constraints are defined as

$$A_u = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} , \ b_u = \begin{bmatrix} 0.249 \\ 0.249 \\ 4 \\ 1 \end{bmatrix} , \tag{17a}$$

$$A_{\Delta u} = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} , \ b_{\Delta u} = \begin{bmatrix} 0.05 \\ 0.05 \\ 0.5 \\ 0.5 \end{bmatrix} \ . \tag{17b}$$

In the estimator, the tuning aims to minimize the process noise while guessing the right value of v_y by using the TS model. The diagonal terms of the weighting matrices in the cost function and past horizon of (13), found by iterative tuning until the desired performance is achieved, are

$$\begin{aligned} Q &= 0.5 \ [\, 0.25 \ \ 0.5 \ \ 0.25 \,], \\ R &= 0.5 \ [\, 0.5 \ \ 0.5 \,], \\ H_p &= 10 \ . \end{aligned} \tag{18}$$

The TS-MHE state region is defined in the polytope

$$A_x = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \end{bmatrix} , \ b_x = \begin{bmatrix} 2.7 \\ -0.1 \\ 0.12 \\ 0.12 \\ 1.96 \\ 1.96 \end{bmatrix} . \tag{19a}$$

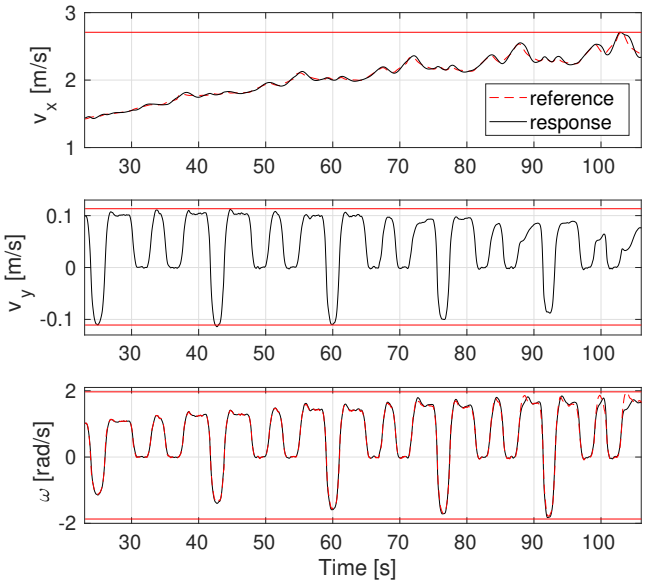

Fig. 5. Vehicle states throughout the simulation. Horizontal red lines represent the upper and lower limits

Figure 5 shows both the reference and the response for each one of the velocity variables. Note that, the vehicle lateral velocity (v_y) cannot be measured and hence, the signal presented is the estimated one. It can be seen that the controller is able to perfectly track the proposed references although having little troubles when driving in racing mode, i.e. after 85 seconds. Horizontal red lines represent the polytope boundaries for each one of the scheduling variables that in this approach coincide with the state and input vehicle variables. Note that, this limits are imposed in the learning stage by the maximum and minimum values of the input signals, i.e. scheduling variables.

In Figure 6, the optimal control actions are shown as well as their discrete time variations which are the ones minimized in the cost function of (12). Note that the steering angle reaches the upper and lower limits at some points while the rear wheel acceleration moves in a wider range.

Finally, after observing a good tracking performance in last figures, we present the elapsed time per iteration of the TS-MPC in Figure 7. It can be seen that, using a prediction horizon of 6 steps, the quadratic solver is able to obtain an average of 4.8 ms. This is one of the most remarkable results of this approach.

6. CONCLUSIONS

In this paper, a learning-based approach for identifying the dynamics of the vehicle and formulating them as a TS representation has been presented. Then, a TS-MPC strategy

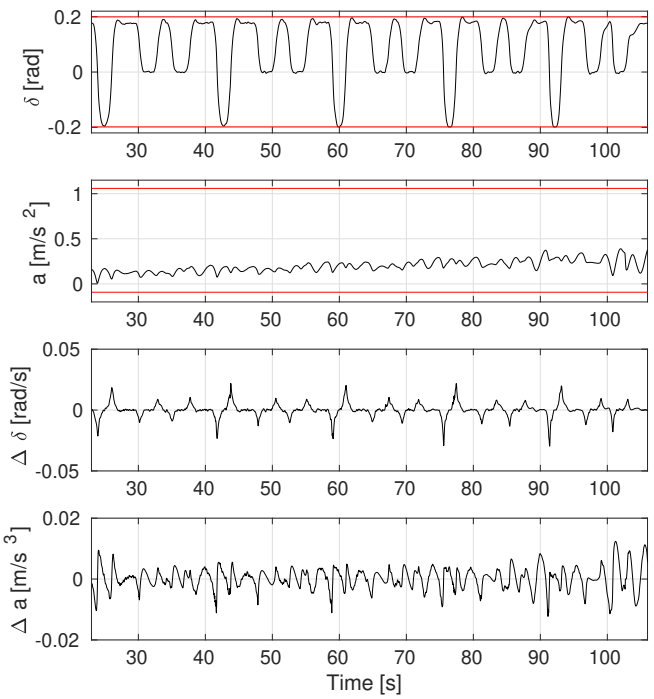

Fig. 6. Control actions and their time derivative variables throughout the simulation. Horizontal red lines represent the upper and lower limits

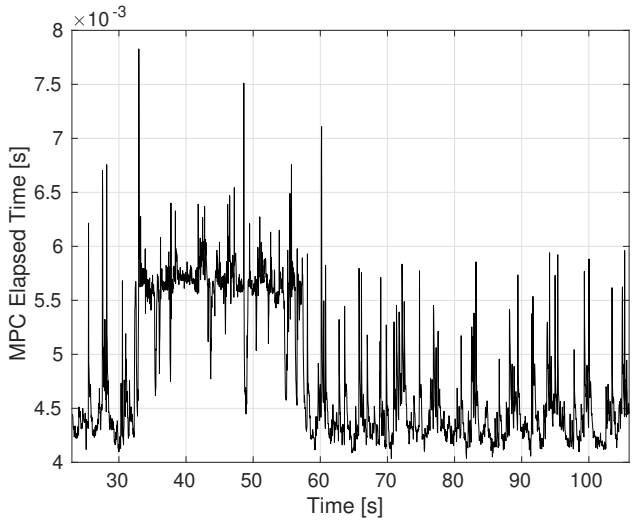

Fig. 7. Computational time required by the TS-MPC throughout the simulation

has been proposed as the approach to solve autonomous driving control problems under realistic conditions in real-time. In addition, using racing-based references provided by an external planner the controller makes the vehicle to perform in racing mode. The control strategy has been tested in simulation showing high performance potential in both reference tracking and computational time. However, this approach share the limitation of learning-based procedures where you can only do what you learn.

REFERENCES

Alcala, E., Cayuela, V.P., and Casin, J.Q. (2019). Ts-mpc for autonomous vehicles including a ts-mhe-uio estimator. *IEEE Transactions on Vehicular Technology*.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., et al. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

Drews, P., Williams, G., Goldfain, B., Theodorou, E.A., and Rehg, J.M. (2017). Aggressive deep driving: Model predictive control with a cnn cost model. *arXiv preprint arXiv:1707.05303*.

Gonzales, J., Zhang, F., Li, K., and Borrelli, F. (2016). Autonomous drifting with onboard sensors. In *Advanced Vehicle Control: Proceedings of the 13th International Symposium on Advanced Vehicle Control (AVEC'16), September 13-16, 2016, Munich, Germany*, 133.

Gurobi Optimization, I. (2015). Gurobi optimizer reference manual. *URL http://www. gurobi. com*.

Jaleel, E.A. and Aparna, K. (2019). Identification of realistic distillation column using hybrid particle swarm optimization and narx based artificial neural network. *Evolving Systems*, 10(2), 149–166.

Jang, J.S. (1993). Anfis: adaptive-network-based fuzzy inference system. *IEEE transactions on systems, man, and cybernetics*, 23(3), 665–685.

Kabzan, J., Hewing, L., Liniger, A., and Zeilinger, M.N. (2019). Learning-based model predictive control for autonomous racing. *IEEE Robotics and Automation Letters*, 4(4), 3363–3370.

Lefèvre, S., Carvalho, A., and Borrelli, F. (2015a). Autonomous car following: A learning-based approach. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, 920–926. IEEE.

Lefèvre, S., Carvalho, A., and Borrelli, F. (2015b). A learning-based framework for velocity control in autonomous driving. *IEEE Transactions on Automation Science and Engineering*, 13(1), 32–42.

Ndiaye, A., Tankari, M.A., Lefebvre, G., et al. (2018). Adaptive neuro-fuzzy inference system application for the identification of a photovoltaic system and the forecasting of its maximum power point. In *2018 7th International Conference on Renewable Energy Research and Applications (ICRERA)*, 1061–1067. IEEE.

Rosolia, U. and Borrelli, F. (2017). Learning model predictive control for iterative tasks. a data-driven control framework. *IEEE Transactions on Automatic Control*, 63(7), 1883–1896.

Rosolia, U. and Borrelli, F. (2019). Learning how to autonomously race a car: a predictive control approach. *arXiv preprint arXiv:1901.08184*.

Rosolia, U., Carvalho, A., and Borrelli, F. (2017). Autonomous racing using learning model predictive control. In *2017 American Control Conference (ACC)*, 5115–5120. IEEE.

Sallab, A.E., Abdou, M., Perot, E., and Yogamani, S. (2017). Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017(19), 70–76.