Model Predictive Control of a Driving Robot for Testing of Advanced Driver Assist Systems

Mike Huang*[a]
mhuang@srisensor.com

***Qiu Haixuan*[b]**
qiuhaixuan@saicmotor.com

***Yang Chunyu*[b]**
yangchunyu@saicmotor.com

***Xia Lian*[b]**
xialian@saicmotor.com

***Lin Zhaomin*[b]**
linzhaomin@saicmotor.com

***Wang Yanqing*[b]**
wangyanqing@saicmotor.com

***Xu Zongqing*[b]**
xuzongqing@saicmotor.com

***Mingfu Tang*[c]**
mftang@srisensor.com

[a]Sunrise Instruments USA, Canton, Michigan, USA
[b]SAIC Motor R&D Innovation Headquarters, Shanghai, China
[c]Sunrise Instruments, Nanning, China

**Corresponding author*

Abstract

Advanced Driver Assist Systems (ADAS) are becoming more prevalent and more sophisticated in passenger vehicles, with features such as automatic lane keeping, pedestrian detection, and emergency breaking. In line with the increased production deployment of ADAS, testing of these systems are becoming more rigorous with more scenarios needing to be considered every year, see, for example, the ADAS testing conducted by Euro NCAP. To fit the need of placing test vehicles and environment factors in very specific and repeatable scenarios, physical driving robots are commonly used. While most current tests set up the test vehicle in near steady state conditions, e.g., constant speed and straight, in the future, more complex, possibly dynamic scenarios will need to be tested. This paper presents a Model Predictive Control (MPC) strategy for controlling a vehicle along dynamic paths and is easily deployable across different vehicles. Experiment results demonstrating the controller capability for both an electric and a conventional vehicle is presented.

Keywords: Model Predictive Control, Optimal Control, Robot Driver, Advanced Driver Assist Systems**.**

1. Introduction

In the following, a Model Predictive Control (MPC) for a driving robot will be developed for the purpose of testing ADAS systems. The robot consists of motors and linkages for pressing the test vehicle's pedal and brake and for steering and will drive the vehicle at specific speeds along specific paths, e.g., as defined by Euro NCAP test scenarios (Euro NCAP, 2020). Tight tracking of the test plan, as defined by target positions and speeds or target positions in time, is pertinent to successful and repeatable tests, particularly since the tests often involve coordinating multiple objects in addition to test vehicle, e.g., leading and following vehicles, cross traffic, and pedestrians.

MPC is a control methodology which uses a plant model to predict states and outputs into the future as a function of a sequence of future controls and current state. This sequence of future controls is then optimized with respect to a cost function, e.g., predicted tracking error, and constraints. Then the first element of the sequence is applied to the plant. The next sampling instant, the optimization problem is updated with the new state measurements / estimates and new targets, the future control sequence is re-optimized, and the first element of the control sequence is again applied to plant.

In the context of control of vehicle dynamics, most work tend to focus on lateral control and tire modelling. Rarely is lateral and longitudinal control considered together, possibly because tight speed tracking may be considered well handled by inner loop controllers. For example, Sun et al. (2019) and Wang et al. (2019) use bicycle and tire models for prediction but only optimize for a steering command while assuming speed is constant over the horizon. Tang et al. (2020) uses a bicycle and tire model for prediction and optimizes for a yaw rate target which is sent to a lower level PID controller for steering. In dynamic scenarios, e.g., accelerating while turning, these decoupled strategies will be suboptimal because yaw rate is fundamentally a nonlinear function of both the steering angle and vehicle speed.

For multi-variable control, Alcalá et al. (2019) uses a multi-layer control structure, first using a Linear Parameter Varying (LPV)-MPC for kinematic planning of speed and yaw rate targets, then using an LPV-LQR controller to obtain steering angle and acceleration from the speed and yaw rate targets. Ugo and Borelli (2020) built a MPC simultaneously controlling steering and acceleration with the ability learn optimal racing trajectories over a sequence of laps and was demonstrated on a RC car platform. Liao-McPherson et al. (2020) considers how to solve the constrained multi-variable (steering and drive force demand) MPC optimization problem in real time using a Fischer-Burmeister function to transform complementarity conditions into nonlinear equations.

None of these prior works consider the powertrain dynamics, i.e., pedal to vehicle acceleration, which we observed in vehicle testing to significantly degrade tracking performance and even stability, particularly if the

test vehicle is a conventional gasoline vehicle. It may be that for autonomous vehicles, tight speed tracking is not as important as steering around obstacles. Additionally, autonomous vehicles are commonly built on top of hybrid or electric vehicle platform, thus, not as exposed to delays in power generation. However, in other developments, e.g., for emissions and fuel economy testing, tight tracking of speed profiles is explicitly required. Pedal robots are sometimes used for these tests. A strategy using a recorded human drive for feed-forward combined with a PID could be used to control the vehicle speed. A more sophisticated strategy (Park et al., 2022) uses Reinforcement Learning (RL) to find a policy for adjusting time varying PI gains. However, to accommodate vehicle variations, e.g., drive modes or entirely different vehicles, a P gain table as a function of vehicle speed must be made first for each vehicle variation.

We have seen in our experiments that the lag / delay between the pedal and vehicle acceleration can be large, e.g., up to 400msec. on conventional vehicle tests. This greatly limits the performance of controllers if the model has no ability to express that lag. Adding on top of this, braking dynamics and acceleration dynamics are vastly different. A model free approach, e.g., PID, would also be sensitive to the lag, greatly limiting its performance. Furthermore, unlike MPC, PID cannot take advantage of knowing future speed targets, and, as a result, requires high gains which are not feasible in the presence of substantial lag. One may suggest a simple model of the powertrain, e.g., a first order filter. However, our experiments also indicate that this is not sufficient to obtain good, closed loop performance.

Our final MPC design, described herein, takes a data driven approach to modelling the vehicle. First, a neural network model of the powertrain is made. Specifically, we use a Long Short-Term Memory (LSTM) network (The MathWorks, Inc., 2022), which can represent general nonlinearities and has internal states capable of representing lags. We then co-optimize for the steering wheel angle and pedal and brake position as a function of the GPS-IMU measurements and target path.

The remainder of this section is organized as follows. Section 2 describes the vehicle model and model validation. Section 3 describes the MPC optimization problem to be solved at every sample time. Section 4 shows closed loop experiment results. Finally, Section 5 contains concluding remarks.

2. Vehicle Model

The vehicle model has the following physical states: longitudinal acceleration, a, longitudinal velocity, v, eastward position, x_E, northward position, y_E, yaw rate (clockwise), ω_z, and yaw, ψ. The controls are the normalized pedal, $0 \le u_p \le 1$, brake, $0 \le u_b \le 1$, and steering, $-1 \le u_s \le 1$. The evolution of the acceleration is modeled with a neural network, f_{nn},

$$\begin{bmatrix} a_{k+1} \\ c_{k+1} \\ h_{k+1} \end{bmatrix} = f_{nn}\left(a_k, v_k, u_{p,k}, u_{b,k}, c_k, h_k\right), \quad (1)$$

where k denotes a discrete time step (we utilize a 10Hz sampling rate) and c and h denote cell and hidden state vectors for the LSTM network respectively. The function, f_{nn}, consists of an input saturation layer, LSTM layer, and a fully connected output layer. The first layer l_1 adds a bias, β_{in}, and saturation to the brake and pedal inputs. This is used to capture a dead band arising from a non-rigid contact between the robot's push rod and the pedals,

$$l_{1,k} = \begin{bmatrix} a_k \\ v_k \\ max\left(u_{p,k} + \beta_{in,1}, 0\right) \\ max\left(u_{b,k} + \beta_{in,2}, 0\right) \end{bmatrix}. \quad (2)$$

The second layer is the LSTM layer with weights W_{LSTM} and R_{LSTM} and bias β_{LSTM} where

$$W_{LSTM} = \begin{bmatrix} W_i \\ W_f \\ W_g \\ W_o \end{bmatrix}, R_{LSTM} = \begin{bmatrix} R_i \\ R_f \\ R_g \\ R_o \end{bmatrix}, \beta_{LSTM} = \begin{bmatrix} \beta_i \\ \beta_f \\ \beta_g \\ \beta_o \end{bmatrix}. \quad (3)$$

Let $\sigma_c(x) = \tanh(x)$ and $\sigma_g(x) = (1 + e^{-x})^{-1}$ denote cell and gate activation functions and let $.\times$ denote an element-wise multiplication. The cell and hidden state update are computed from the following sequence of equations.

$$\begin{aligned} i_k &= \sigma_g\left(W_i l_{1,k} + R_i h_k + \beta_i\right), \\ f_k &= \sigma_g\left(W_f l_{1,k} + R_f h_k + \beta_f\right), \\ g_k &= \sigma_c\left(W_g l_{1,k} + R_g h_k + \beta_g\right), \\ o_k &= \sigma_g\left(W_o l_{1,k} + R_o h_k + \beta_o\right), \\ c_{k+1} &= f_k .\times c_k + i_k .\times g_k, \\ h_{k+1} &= o_k .\times \sigma_c(c_{k+1}). \end{aligned} \quad (4)$$

Finally, the acceleration at time $k+1$ is computed from a fully connected layer with weights, W_{fc}, and bias β_{fc},

$$a_{k+1} = W_{fc} h_{k+1} + \beta_{fc}. \quad (5)$$

Fitting of the weights $\beta_{in}, W_{LSTM}, R_{LSTM}, \beta_{LSTM}, W_{fc}$ and β_{fc} is executed with MathWork's Deep Learning Toolbox (The MathWorks, Inc., 2022).

The remaining state equations are simple and natural, with a discretization step size, Δt (100msec.),

$$\begin{aligned} v_{k+1} &= v_k + \Delta t a_k, \\ x_{E,k+1} &= x_{E,k} + \Delta t \cos\psi_k, \\ y_{E,k+1} &= y_{E,k} + \Delta t \sin\psi_k, \\ \omega_{z,k+1} &= \alpha_1 \omega_z + \alpha_2 v_k u_{s,k}, \\ \psi_{k+1} &= \psi_k + \Delta t \omega_{Z,k}, \end{aligned} \quad (6)$$

where α_1 and α_2 are coefficients, fit with linear least squares, giving the yaw rate response the ability to represent first order dynamics from the steering input.

The model was trained on less than 10 min. of driving data from a combination of human driving and an old version of a MPC controller. The driving scenarios include

straight line drives, circular drives and weaving through a series of cones. The maximum speed was 60kph due to limited size of the testing site. The number of cell and hidden states is eight each, i.e., $c, h \in \mathbb{R}^8$.

In addition to the neural network acceleration model, model results from a first order acceleration model and a static model will be shown for comparison. The first order acceleration model, a_{fo}, has the form

$$a_{fo,k+1} = c_0 + c_1 a_{fo,k} + c_2\, max\big(u_{p,k} + \beta_{in,1}, 0\big) + c_3\, max\big(u_{b,k} + \beta_{in,2}, 0\big) + c_4 v_k + c_5 v_k^2, \quad (7)$$

where c_0 through c_5 are fitted coefficients. The static model is the same as the first order model except with $c_1 = 0$. Figure 1 shows a comparison of the models against data from a straight-line drive scenario with a conventional vehicle. As expected, the static model greatly leads the acceleration response, e.g., see the time difference between the acceleration peaks between 38 sec. and 45 sec. While the first order model can roughly match the phase of the oscillations between 38 sec. and 45 sec., the neural net model is able to match the peak-to-peak amplitude better. The neural net model can also express a different brake response, between 55 sec. and 60 sec., from the pedal response whereas the first order model must use the same time constant for both pedal and brake.

Figure 2 shows the modeled yaw rate versus data where the vehicle is weaving through a series of cones. As can be seen, the simple yaw rate model is effective for matching the test data and, as will be shown in later, is sufficient for achieving low lateral path tracking error when used by the MPC.

3. Model Predictive Controller Design

In the following, the optimization problem associated with the MPC will be given and the solver strategy will be discussed.

First, because the pedal and brake should not be pressed at the same time, they can be combined into a single actuator for the purposes of prediction and optimization. Let the combined pedal and brake be denoted u_{pb} and let u_p and u_b in acceleration model, eq. (1), be replaced with $u_p = \max(u_{pb}, 0)$ and $u_b = \max(-u_{pb}, 0)$. Let u be a concatenation of controls, $u = \left[u_{pb}\ u_s\right]^T$. For prediction, $\beta_{in} = 0$ is utilized, i.e., removing the dead band in prediction, knowing that integral action will take care of static offsets and avoiding loss of local sensitivity to the controls in the MPC computations. Let x_k be a concatenation of discrete time states,

$$x_k = \left[a_k\ v_k\ x_{E,k}\ y_{E,k}\ \omega_{Z,k}\ \psi_k\ c_k^T\ h_k^T\ u_{k-1}^T\right]^T, \quad (8)$$

where u_{k-1} is the control applied at the previous time step and will be used to facilitate evaluating the rate of change of the control in the cost function to be minimized. The equations (1) – (6) and (8) can be compactly written into a system of nonlinear difference equations,

$$x_{k+1} = f(x_k, u_k). \quad (9)$$

We will define the number of states and number of controls as N_s and N_c respectively.

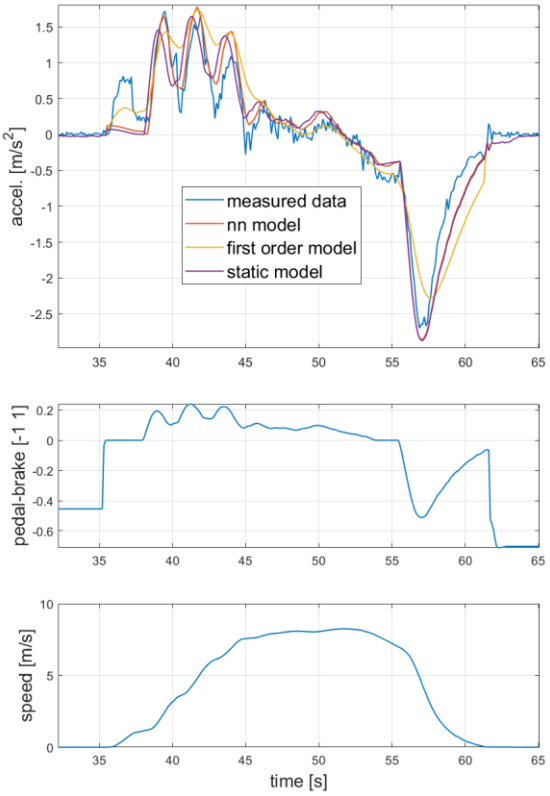

Fig. 1. Acceleration models versus measured data.

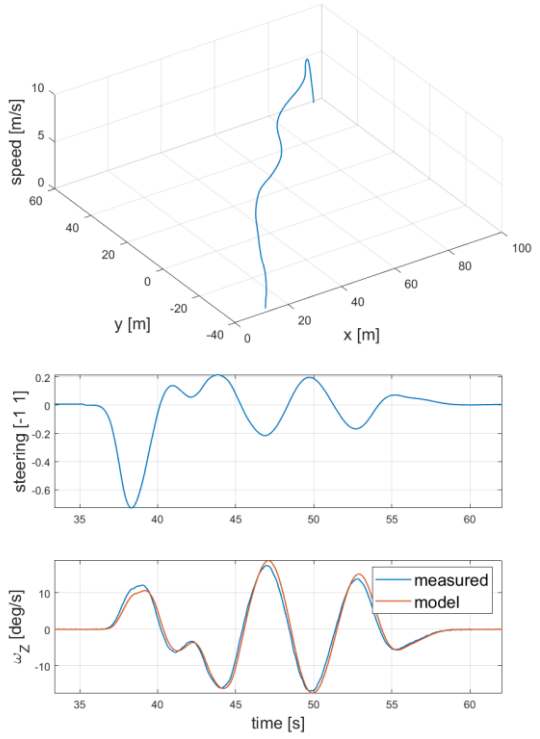

Fig. 2. Yaw rate model versus measured data.

The objective of the MPC is to track a sequence of positions as a function of time (or sample), i.e., the test

plan is defined through functions $x_E^{target}(t)$ and $y_E^{target}(t)$ which may be stored in a lookup table and interpolated. The optimization problem to be solved at each sample time is

$$\min_{u_0,\dots,u_{N-1}} \sum_{k=0}^{N-1} \left(x_{E,k} - x_{E,k}^{target}\right)^2 + \left(y_{E,k} - y_{E,k}^{target}\right)^2 + (u_k - u_{k-1})^T R (u_k - u_{k-1})$$

$$subject\ to\ the\ constraints:$$

$$x_0 = x(t),$$
$$x_{k+1} = f(x_k, u_k),$$
$$-1 \le u_k \le 1,$$
$$v_k \ge 0, \qquad (10)$$

where N is the horizon length and $x(t)$ is the measured state at sample time t with cell and hidden states obtained from evaluating equations (2) and (4) once per sample. The targets, $x_{E,k}^{target}$ and $y_{E,k}^{target}$ are obtained from sampling $x_E^{target}(t)$ and $y_E^{target}(t)$. The cost function in (10) penalizes the control effort through the control rate of change, $u_k - u_{k-1}$, giving a smooth control action and rider comfort and removes the need to compute a control target.

An alternative to the cost function in (10) would be to instead have terms to track speed and yaw targets (or x-y velocity components instead of yaw) and penalize lateral path offset and control effort. In our simulations and experiments, we have observed that this alternate version has difficulties at low speed, e.g., less than 10-15kph. Also, compared to (10), this alternate version has more tuning parameters, and requires additional integrators / adaptors / estimators for speed tracking, which also need to be tuned. In contrast, the MPC, (10), only has tuning on the control effort and speed tracking is inherited because position error is the integral of speed error, assuming the speed targets and position targets are consistent, i.e., $v^{target}(t) = \left[\dot{x}_E^{target}(t)\ \ \dot{y}_E^{target}(t)\right]^T$.

To assist the solving of the optimization problem (10) in real time, the inequality constraints are converted to an exterior penalty function

$$\gamma(u_k, v_k) = \gamma_1 \left|\left|max(u_k - 1, 0)\right|\right|_2^2 + \gamma_1 \left|\left|max(1 - u_k, 0)\right|\right|_2^2 + \gamma_2 \left|\left|v_k\right|\right|_2^2, \qquad (11)$$

where $\left|\left|x\right|\right|_2^2 = x^T x$ and γ_1 and γ_2 are tuning variables which can be set large relative to R, i.e., $\gamma_1, \gamma_2 \gg \left|\left|R\right|\right|$. In practice, values for γ_1 and γ_2 are easily selected in simulations and does not need to be changed afterward.

For convenience, let the incremental cost, l, be defined as

$$l(x_k, u_k) = \left(x_{E,k} - x_{E,k}^{target}\right)^2 + \left(y_{E,k} - y_{E,k}^{target}\right)^2 + (u_k - u_{k-1})^T R (u_k - u_{k-1}) + \gamma(u_k, v_k). \qquad (12)$$

The exterior penalized optimized problem to be solved in real time is

$$\min_{u_0,\dots,u_{N-1}} \sum_{k=0}^{N-1} l(x_k, u_k)$$

$$subject\ to\ the\ constraints:$$

$$x_0 = x(t),$$
$$x_{k+1} = f(x_k, u_k). \qquad (13)$$

From the optimization problem (13), necessary conditions for optimality are obtained through applying Karush-Kuhn-Tucker conditions and are as follows,

$$x_{k+1} = f(x_k, u_k) \quad \forall k \in \{0, \dots, N-1\}, x_0 = x(t), \qquad (14)$$

$$p_k = l_x(x_k, u_k) + f_x^T(x_k, u_k) p_{k+1} \quad \forall k \in \{1, \dots, N-1\},\ p_N = 0, \qquad (15)$$

$$l_u(x_k, u_k) + f_u^T(x_k, u_k) p_{k+1} = 0 \quad \forall k \in \{0, \dots, N-1\}, \qquad (16)$$

where l_x, l_u, f_x and f_u are $\partial l/\partial x$, $\partial l/\partial u$, $\partial f/\partial x$ and $\partial f/\partial u$ respectively. The co-states, $p_k \in \mathbb{R}^{N_s}$, are the Lagrange multipliers associated with state equation equality constraints. Equation (15) is referred to as the co-state equation and equation (16) is referred to as the stationarity condition.

Let X be the sequence of future states, $X = [x_0^T, .., x_{N-1}^T]^T \in \mathbb{R}^{(N-1)N_S}$ and P be the sequence of co-states, $P = [p_1^T, .., p_N^T]^T \in \mathbb{R}^{(N-1)N_S}$. The states, X, can be computed as a function of U and initial condition $x(t)$ through propagation of the state equations, i.e., $X = X(U, x(t))$. The co-states can be computed as a function of U and X through back-propagation of the co-state equations, i.e., $P = P(X, U)$. We define the residual, $F \in \mathbb{R}^{(N-1)N_c}$, as the left-hand side of the stationarity conditions,

$$F(X, U, P) = \begin{bmatrix} l_u(x_0, u_0) + f_u^T(x_0, u_0) p_1 \\ \dots \\ l_u(x_{N-1}, u_{N-1}) + f_u^T(x_{N-1}, u_{N-1}) p_N \end{bmatrix}. \qquad (17)$$

Note that $F(X, U, P) = F\big(X(U), U, P(X(U), U)\big) = F(U)$. Newton's method can now be used to find a sequence of U's such that $F(U) \to 0$, i.e., satisfying the necessary conditions. Let $U^i(t)$ denote the i-th iteration of U at sample time t. The Newton update is

$$U^{i+1}(t) = U^i(t) + \delta U, \qquad (18)$$

where δU is the Newton step. Specifically, the Forward Difference Conjugate Gradient (FDCG) algorithm (Algorithm 2.5.1 in Kelley, 1999) is used calculate δU starting from an initial point $U^i(t)$ and residual $F(U^i(t))$. The advantage of the FDCG algorithm is that it does not require computing the Jacobian, $J(U^i(t))$, of the residual around the point $U^i(t)$. Rather the Jacobian times vector product in the standard Conjugate Gradient (CG) algorithm (Algorithm 1.5.1 in Kelley, 1999) can be approximated by a forward difference calculation,

$$J\left(U^i(t)\right)\rho \approx \frac{\left(F(U^i(t)+\epsilon\rho)-F(U^i(t))\right)}{\epsilon}. \qquad (19)$$

Finally, a Real Time Iteration (RTI) strategy (Diehl et al. 2005) will be used, meaning that the optimization problem (13) will not be solved to completion, but rather, a fixed, finite number, N_{New}, of Newton steps (18) will be taken each sampling instant. The initial guess for the sequence of controls at the next sampling instant is the final iterate from the current sampling instant, i.e., $U^0(t+\Delta t) = U^{N_{New}}(t)$.

4. Experiment Results

The MPC used in experiments utilize a prediction horizon of 4.5 sec. (N = 45 steps), the sampling rate is 10Hz and 10 Newton steps are taken every sample. Figure 3 shows experiments with MPC controlling a conventional vehicle along a straight-line path. The target positions versus time were obtained from a human drive. Note that the red pedal-brake line is not used as a target for the control and is only there to show how the human drove the path. Both a MPC with the neural net powertrain model and a MPC with a first order model are shown, where the MPC with a first order model struggles to track the speed target while the MPC with neural net can track the target closely. Different tunings of the MPC with the first order model were not able to achieve satisfying speed tracking performance. A different configuration with a velocity and lateral offset type cost function was somewhat more successful but still did not perform as well as the MPC with neural net model.

The MPC with neural net in Fig. 3 gains some speed at the beginning of the experiment when the target is zero. This is due to down-sampling of the target drive and target interpolation and is not an issue of the MPC itself. At the transition between acceleration and braking, the MPC looks like it is a little slow to brake. This is due to the dead-band as the pedal comes off and as the brake engages which the MPC is currently configured to smoothly pass through.

Figures 4 and 5 show the MPC (with neural net powertrain model) controlling the conventional vehicle along a cone weaving path and along a circular path respectively. In these cases, the speed tracking is also satisfactory and the path error, as defined as the distance to the closest point on the target path, is less than 30 cm after the initial condition response. Note that the MPC softly stops to zero speed due to the $v \geq 0$ constraint. This causes some overshoot of the final target position (and is why the path error becomes large at the end of the path).

Finally, Fig. 6 shows the MPC driving an electric vehicle along a cone weaving path (with the model refit to electric vehicle data). As expected, the speed tracking and path error is good. The main thing to note is that after model refitting, no other parameters / tunings of the MPC were changed from the conventional vehicle settings. This shows the general applicability of the approach.

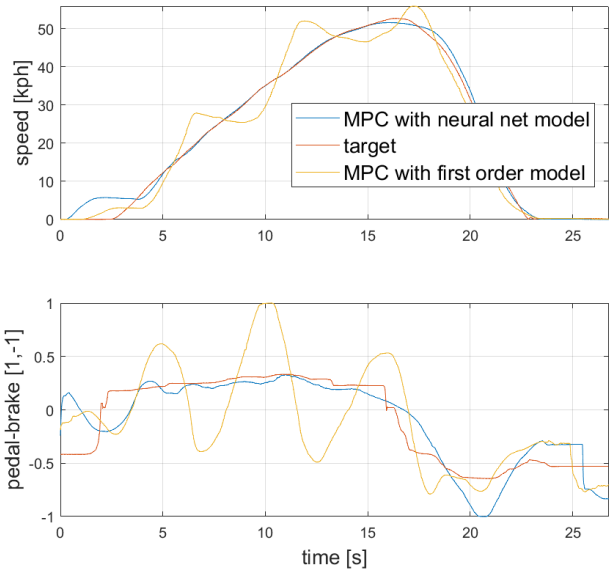

Fig. 3. Speed tracking performance on a conventional vehicle comparison between MPC with neural network model and MPC with a first order powertrain model on a straight-line drive.

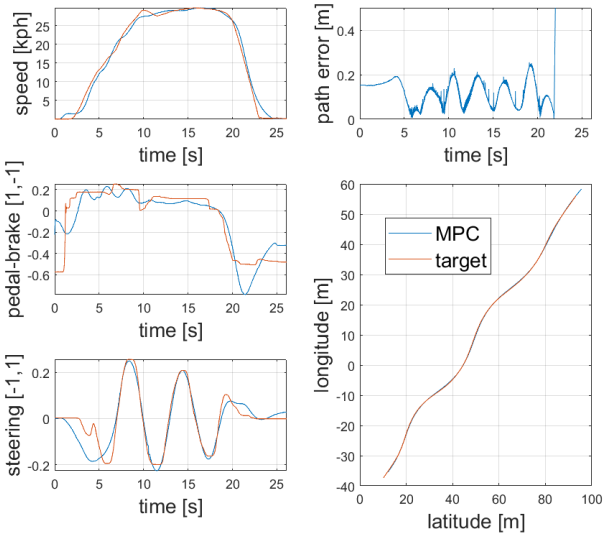

Fig. 4. Speed and path tracking performance on a conventional vehicle with the MPC when weaving through cones.

5. Conclusions

This paper describes a MPC for a driving robot to be used for testing ADAS systems. Specifically, this MPC utilizes a neural network, with LSTM layer, to model the acceleration dynamics of the vehicle. Being able to easily adapt to a wide variety of vehicles and powertrains is necessary for the driving robot, a capability not clearly seen in other robot driver or autonomous driving developments. Despite the model being data driven, little data is needed to fit the model, e.g., 10 min of driving data. Furthermore, no tuning of the controller was required

when swapping between the conventional and electric vehicles in our experiments.

In future work, we will be testing the controller in additional scenarios and higher speeds, specifically ensuring that Euro NCAP test requirements are met. We will also be considering the coordination of the test vehicle with other robot-controlled vehicles and objects.

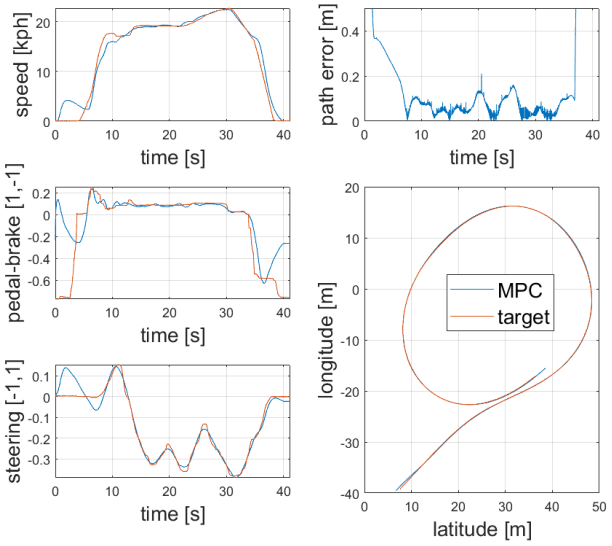

Fig. 5. Speed and path tracking performance on a conventional vehicle with the MPC when driving in a circle.

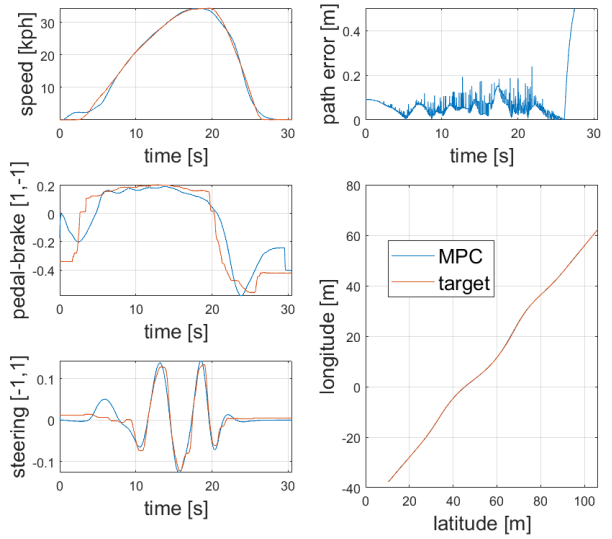

Fig. 6. Speed and path tracking performance on a electric vehicle with the MPC when weaving through cones.

6. References

Alcalá, E., Puig, V., Quevedo, J., 2019. LPV-MPC Control for Autonomous Vehicles. *IFAC PapersOnline*. Vol. 32, No. 28, pp. 106-113.

Diehl, M., Findeisen, R., Allgöwer, F., Bock, H. G., Schlöder, J. P., 2005. Nominal Stability of the Real-Time Iteration Scheme for Nonlinear Model Predictive Control. *IEEE Proceedings-Control Theory and Applications*, Vol. 152, No. 3, pp. 296-308.

Euro NCAP, 2020. 2020 Assisted Driving Tests, Retrieved from https://www.euroncap.com/en/vehicle-safety/safety-campaigns/2020-assisted-driving-tests/, July 13, 2022.

Kelley, C. T., 1999. Iterative Methods for Optimization. *Society for Industrial and Applied Mathematics*, Philadelphia.

Liao-McPherson, D., Huang, M., Zaseck, K., 2020. Smoothed and regularized Fischer-Burmeister solver for embedded real-time constrained optimal control problems in autonomous systems. *United States Patent*, No. US 10,739,768 B2.

The MathWorks, Inc. 2022. Deep Learning Toolbox. Retrieved from https://www.mathworks.com/products/deep-learning.html, July 14, 2022.

Park, J., Kim, H., Hwang, K., Lim, S., 2022. Deep reinforcement learning based dynamic proportional-integral (PI) gain auto-tuning method for a robot driver system. *IEEE Access*. Vol 10., pp. 31043-31057.

Sun, C., Zhang, X., Zhou, Q., Tian, Y., 2019. A model predictive control with switched tracking error for autonomous vehicle path tracking. IEEE Access. Vol. 7, pp. 53103 – 53114.

Tang, L., Yan, Fuwu, Y., Zou, B., Wang, K., Chen L., 2020. An improved kinematic model predictive control for high-speed path tracking of autonomous vehicles. *IEEE Access*. Vol 8., pp. 51400-51413.

Ugo, R., Borelli, F., 2020. Learning how to autonomously race a car: a predictive control approach. Direct control acceleration and steering angle. *IEEE Transactions on Control Systems Technology*, Vol. 28, No. 6, pp. 2713-2719.

Wang, H., Liu, B., Ping, X., An Q., 2019. Path tracking control for autonomous vehicles based on an improved MPC. *IEEE Access*. Vol. 7, pp. 161064-161073.