Driving Fast but Safe: On Enforcing Operational Limits of a NMPC System

Adam Gotlib, Krzysztof Jóskowiak, Piotr Libera, Marcel Kaliński, Jakub Bednarek and Maciej Majek
Students' Robotics Association, Faculty of Power and Aeronautical Engineering, Warsaw University of Technology, ul. Nowowiejska 24, 00-665 Warsaw, Poland

Keywords: Autonomous Driving, Control Systems, Nonlinear Model Predictive Control.

Abstract: In this paper, we present a novel approach to Model Predictive Control that allows to explore the largest possible portion of the state–space when still using a low–computational–complexity vehicle model. By introducing additional constraint for acceleration magnitude we are able to stay within the limits where the model gives accurate predictions, while driving with high velocity. This effects a behavior similar to one of a professional racing driver, as the controller is able to balance speed and curvature of the vehicle at any point in time.

1 INTRODUCTION

Control is one of the fundamental components of any robotics system. In case of autonomous vehicles, a common solution is to apply a steering feedback loop compensating for deviations from target trajectory, generated by a planning module. In this setting, planning is not considered a part of the control system, which is only responsible for execution of a predefined plan.

One of the variations of this method, which is steadily gaining popularity thanks to increasing performance of modern computer systems, involves replacing the feedback loop with receding window optimization. This approach, more widely known as Model Predictive Control, relies on a system model which is detailed enough to give accurate predictions on one hand, but simple enough to allow for real–time execution on the other. During each iteration, a single optimization problem is solved, resulting in a set of control inputs minimizing a specified cost function over a small prediction window. One of the biggest advantages of MPC is its great flexibility that allows to operate processes—such as automated driving—under complex constraints thanks to carefully designed cost functions (Allgöwer et al., 2004). For trajectory tracking goal, such a function might for example include deviation from the reference path at each prediction step (Faulwasser et al., 2009). In consequence, much better trajectories can be realized—and some issues, such as overshoot present with more classic controllers (Balaji et al., 2015), can be avoided altogether. However, strengths of this solution come with some drawbacks. MPC is only as good as the system model. Accurate predictions are crucial for the calculated inputs to be valid in the real world. On the other hand, the optimization algorithm introduces additional computational overhead depending on model complexity, much larger than of most common feedback controllers. Since execution has to be performed in real time, performance becomes a particularly important concern.

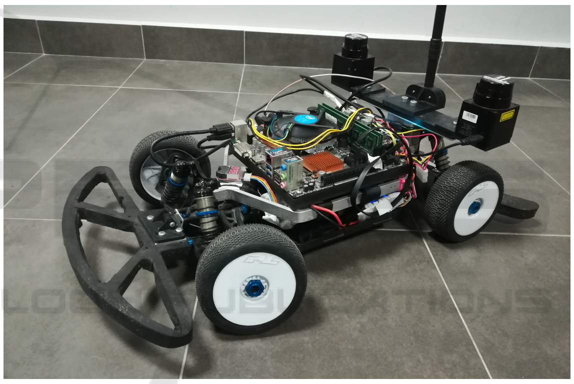
Figure 1: Selfie V2.1 UGV Platform.

This problem becomes especially evident when driving close to vehicle handling limits, e.g. during execution of high speed emergency maneuvers or racing. A lot of factors come into play, such as aerodynamic forces, load transfer, tire stiffness varying with temperature, coefficient of friction between tires and the road, etc. As velocities and accelerations increase, more and more nonlinearities are introduced in the system. This creates the need for more detailed models required to capture all nuances affecting behavior of the system.

Gotlib, A., Jóskowiak, K., Libera, P., Kaliński, M., Bednarek, J. and Majek, M.
Driving Fast but Safe: On Enforcing Operational Limits of a NMPC System.
DOI: 10.5220/0009425704970503
In *Proceedings of the 6th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2020)*, pages 497-503
ISBN: 978-989-758-419-0

1.1 Related Work

Literature offers numerous examples regarding path–tracking at vehicle limits. (Kritayakirana and Gerdes, 2012) show how a ‘g–g’ diagram can be used to generate a speed profile utilizing the maximum of available force of friction. This allows to employ advanced driving techniques, such as trail–braking and corner–on–exit, which involve balancing lateral and longitudinal accelerations between segments of the track with varying curvature.

In the area of optimal control, (Lam et al., 2010) introduce a cost function to trade off contouring accuracy against path speed. This approach has been successfully used in (Liniger et al., 2014) and (Kabzan et al., 2019b) to autonomously drive a car on a race track. Both use a high–fidelity dynamic bicycle model and additional constraints for tire forces and path deviation.

Data–driven modeling is also a technique that has found successful applications, largely thanks to the property that allows to capture various intricacies of the dynamic system, often hard to account for with purely analytic models. An example of this method can be found in (Williams et al., 2016), where a bayesian linear model is being fit to driving data generated under human operation. Use of such a model has been enabled by the presence of a custom optimization algorithm, making use of multi–threading capabilities of a Graphical Processing Unit (GPU) available on the platform. (Kabzan et al., 2019a) demonstrate a hybrid approach, where a base model is first used, which is then error–corrected based on measured inaccuracies in state prediction.

1.2 Contribution Overview

In this paper, we strive for a simple yet performant solution which is able to balance prediction accuracy with execution time. A low–complexity vehicle model is used, which has the additional advantage in that minimal preparation is required for parameter estimation. To ensure the system always stays in the safe part of the state space—by which we mean the subspace where model inaccuracies are still small enough to allow to reliably control the vehicle—additional constraint on acceleration is introduced, similar to the one used in (Kritayakirana and Gerdes, 2012). We intend best possible path-tracking speed on an Ackermann-type vehicle when operating with a model with low computational requirements and without using a pre-computed speed profile. The last constraint allows for better flexibility, should the system be extended in the future e.g. incorporate obstacle avoidance. Our solution runs in real-time on a modern Central Processing Unit (CPU), what is validated experimentally.

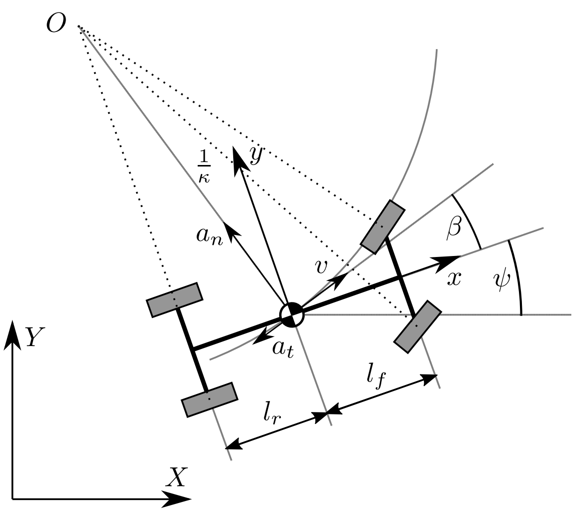

Figure 2: Kinematics of a double track vehicle model.

2 MODEL ANALYSIS

2.1 Kinematic Equations of Motion

Let us consider a vehicle with Ackermann steering, as depicted in figure 2. For computational simplicity, assume lateral velocity of each tire to be equal zero, so that path curvature κ depends solely on inclinations of the front wheels. This assumption is an approximation of the actual behavior of the system, as tires will deform under the influence of the force of friction – which in general scales along with acceleration, up to a maximum value dictated by the coefficient of friction and vertical load. It follows that this approximation can only be valid up to certain accelerations – an exact value to be evaluated experimentally.

State of the vehicle is characterized by its position X, Y, heading ψ, velocity v, and slip angle β, which together form a vector $\boldsymbol{x}$. Motion of the vehicle can be described by differential equations:

$$\dot{X} = v\cos\beta\cos\psi - v\sin\beta\sin\psi \tag{1a}$$

$$\dot{Y} = v\cos\beta\sin\psi + v\sin\beta\cos\psi \tag{1b}$$

$$\dot{\psi} = v\kappa \tag{1c}$$

where:

$$\kappa = \frac{\sin\beta}{l_r} \tag{2}$$

2.2 Acceleration Magnitude Squared

First derivative of position gives us velocity vector:

$$\boldsymbol{V} = \begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} \tag{3}$$

Note that based on equations 1a–1b, $\boldsymbol{V}$ can be defined as function of v, β, and ψ. By applying the chain rule, we can derive an expression for the acceleration vector $\boldsymbol{A}$:

$$\boldsymbol{A} = \dot{\boldsymbol{V}} = \underbrace{\begin{bmatrix} \frac{\partial \dot{X}}{\partial v} & \frac{\partial \dot{X}}{\partial \beta} & \frac{\partial \dot{X}}{\partial \psi} \\ \frac{\partial \dot{Y}}{\partial v} & \frac{\partial \dot{Y}}{\partial \beta} & \frac{\partial \dot{Y}}{\partial \psi} \end{bmatrix}}_{\boldsymbol{J_V}} \begin{bmatrix} \dot{v} \\ \dot{\beta} \\ \dot{\psi} \end{bmatrix} \tag{4}$$

where $\boldsymbol{J_V}$ is the Jacobian of $\boldsymbol{V}(v, \beta, \psi)$ with the following partial derivatives:

$$\frac{\partial \dot{X}}{\partial v} = \cos\beta\cos\psi - \sin\beta\sin\psi \tag{5a}$$

$$\frac{\partial \dot{Y}}{\partial v} = \cos\beta\sin\psi + \sin\beta\cos\psi \tag{5b}$$

$$\frac{\partial \dot{X}}{\partial \beta} = -v\sin\beta\cos\psi - v\cos\beta\sin\psi \tag{5c}$$

$$\frac{\partial \dot{Y}}{\partial \beta} = -v\sin\beta\sin\psi + v\cos\beta\cos\psi \tag{5d}$$

$$\frac{\partial \dot{X}}{\partial \psi} = -v\cos\beta\sin\psi - v\sin\beta\cos\psi \tag{5e}$$

$$\frac{\partial \dot{Y}}{\partial \psi} = v\cos\beta\cos\psi - v\sin\beta\sin\psi \tag{5f}$$

We can also calculate square of the acceleration magnitude with a quadratic form:

$$A^2 = \boldsymbol{A}^T\boldsymbol{A} = \begin{bmatrix} \dot{v} \\ \dot{\beta} \\ \dot{\psi} \end{bmatrix}^T \boldsymbol{J_V^T J_V} \begin{bmatrix} \dot{v} \\ \dot{\beta} \\ \dot{\psi} \end{bmatrix} \tag{6}$$

Note that:

$$\boldsymbol{J_V^T J_V} = \begin{bmatrix} \frac{\partial \dot{X}}{\partial v}\frac{\partial \dot{X}}{\partial v} + \frac{\partial \dot{Y}}{\partial v}\frac{\partial \dot{Y}}{\partial v} & \frac{\partial \dot{X}}{\partial v}\frac{\partial \dot{X}}{\partial \beta} + \frac{\partial \dot{Y}}{\partial v}\frac{\partial \dot{Y}}{\partial \beta} & \frac{\partial \dot{X}}{\partial v}\frac{\partial \dot{X}}{\partial \psi} + \frac{\partial \dot{Y}}{\partial v}\frac{\partial \dot{Y}}{\partial \psi} \\ \frac{\partial \dot{X}}{\partial v}\frac{\partial \dot{X}}{\partial \beta} + \frac{\partial \dot{Y}}{\partial v}\frac{\partial \dot{Y}}{\partial \beta} & \frac{\partial \dot{X}}{\partial \beta}\frac{\partial \dot{X}}{\partial \beta} + \frac{\partial \dot{Y}}{\partial \beta}\frac{\partial \dot{Y}}{\partial \beta} & \frac{\partial \dot{X}}{\partial \beta}\frac{\partial \dot{X}}{\partial \psi} + \frac{\partial \dot{Y}}{\partial \beta}\frac{\partial \dot{Y}}{\partial \psi} \\ \frac{\partial \dot{X}}{\partial v}\frac{\partial \dot{X}}{\partial \psi} + \frac{\partial \dot{Y}}{\partial v}\frac{\partial \dot{\psi}}{\partial \psi} & \frac{\partial \dot{X}}{\partial \beta}\frac{\partial \dot{X}}{\partial \psi} + \frac{\partial \dot{Y}}{\partial \beta}\frac{\partial \dot{Y}}{\partial \psi} & \frac{\partial \dot{X}}{\partial \psi}\frac{\partial \dot{X}}{\partial \psi} + \frac{\partial \dot{Y}}{\partial \psi}\frac{\partial \dot{Y}}{\partial \psi} \end{bmatrix} \tag{7}$$

and also:

$$\frac{\partial \dot{X}}{\partial v}\frac{\partial \dot{X}}{\partial v} + \frac{\partial \dot{Y}}{\partial v}\frac{\partial \dot{Y}}{\partial v} = 1 \tag{8a}$$

$$\frac{\partial \dot{X}}{\partial \beta}\frac{\partial \dot{X}}{\partial \beta} + \frac{\partial \dot{Y}}{\partial \beta}\frac{\partial \dot{Y}}{\partial \beta} = v^2 \tag{8b}$$

$$\frac{\partial \dot{X}}{\partial \psi}\frac{\partial \dot{X}}{\partial \psi} + \frac{\partial \dot{Y}}{\partial \psi}\frac{\partial \dot{Y}}{\partial \psi} = v^2 \tag{8c}$$

$$\frac{\partial \dot{X}}{\partial v}\frac{\partial \dot{X}}{\partial \beta} + \frac{\partial \dot{Y}}{\partial v}\frac{\partial \dot{Y}}{\partial \beta} = 0 \tag{8d}$$

$$\frac{\partial \dot{X}}{\partial v}\frac{\partial \dot{X}}{\partial \psi} + \frac{\partial \dot{Y}}{\partial v}\frac{\partial \dot{\psi}}{\partial \psi} = 0 \tag{8e}$$

$$\frac{\partial \dot{X}}{\partial \beta}\frac{\partial \dot{X}}{\partial \psi} + \frac{\partial \dot{Y}}{\partial \beta}\frac{\partial \dot{Y}}{\partial \psi} = v^2 \tag{8f}$$

By substituting from 7 and 8a–8f into eq. 6, we get:

$$\begin{aligned} A^2 &= \begin{bmatrix} \dot{v} \\ \dot{\beta} \\ \dot{\psi} \end{bmatrix}^T \begin{bmatrix} 1 & 0 & 0 \\ 0 & v^2 & v^2 \\ 0 & v^2 & v^2 \end{bmatrix} \begin{bmatrix} \dot{v} \\ \dot{\beta} \\ \dot{\psi} \end{bmatrix} \\ &= \dot{v}^2 + v^2\left(\dot{\beta}^2 + \dot{\psi}^2 + 2\dot{\beta}\dot{\psi}\right) \\ &= \dot{v}^2 + v^2\left(\dot{\beta} + \dot{\psi}\right)^2 \end{aligned} \tag{9}$$

Finally, substituting from equations 1c and 2:

$$A^2 = \dot{v}^2 + v^2\left(\dot{\beta} + v\frac{\sin\beta}{l_r}\right)^2 \tag{10}$$

3 CONTROLLER DESIGN

3.1 Path Tracking Scenario

For the purpose of this paper, we will consider an application where the path the vehicle is supposed to follow is given as a sequence of waypoints W_i, as depicted in fig. 3. Furthermore, we assume that the vehicle is positioned somewhat along the direction of the path, i.e. the difference between their headings lies within some small tolerance. This assumption is crucial to proper functioning of the system, and in practice could be enforced by a separate fail-safe mechanism that triggers a recovery procedure should the vehicle find itself across the path, or even in the opposite direction.

From this assumption, given dense enough placement of waypoints and kinematic viability of the underlying path (maximum curvature constraint), follows that there will be a range of waypoints for which we have increasing x values in vehicle coordinate frame. This in turn allows us to approximate a section of the path using spline interpolation $S(x)$. We will use this coordinate frame to denote deviations from the initial position as a triplet $(\hat{x}, \hat{y}, \hat{\psi})$. Let us also define lateral and heading errors with respect to the reference path as respectively:

$$\Delta\hat{y} = \hat{y} - S(\hat{x}) \tag{11a}$$

$$\Delta\hat{\psi} = \hat{\psi} - \arctan\left(S'(\hat{x})\right) \tag{11b}$$

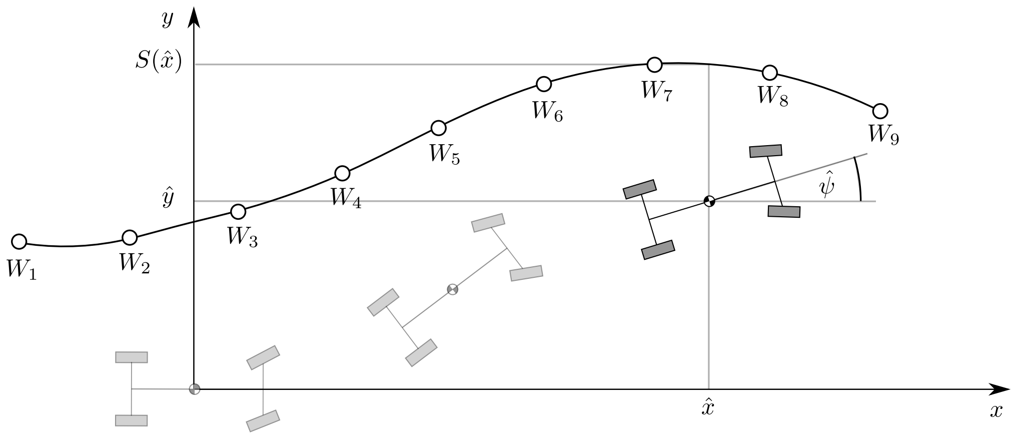

Figure 3: Vehicle Progression along a Reference Path.

3.2 Problem Formulation

Informally, we can state our task as one of following the path as closely and quickly as possible while staying within kinematic and dynamic limits of the vehicle. Around that description, we can build a more formal definition that can bring it to constrained optimization problem. The variables we want to optimize are given directly by equations 11a and 11b and state is defined in section 2.1. We can define a state-based cost function $C(\boldsymbol{x})$ as:

$$C(\boldsymbol{x}) = c_1 \cdot \Bigg(\underbrace{Y - S(X)}_{\Delta\hat{y}}\Bigg)^2 + c_2 \cdot \Bigg(\underbrace{\psi - \arctan\left(S'(X)\right)}_{\Delta\hat{\psi}}\Bigg)^2 + c_3 \cdot (v - v_{max})^2 \tag{12}$$

where c_1, c_2, c_3, and v_{max} are configurable parameters. Viability constraints can be represented as restrictions on allowed values for v, β, $\dot{v}$, $\dot{\beta}$, and A (eq. 10).

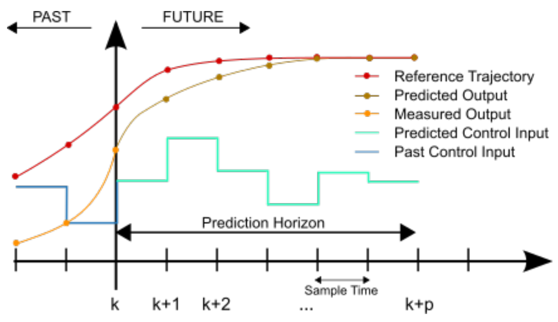

Figure 4: A Basic Working Principle of Model Predictive Control (M. Behrendt, Wikimedia Commons).

3.3 Receding Time Window Optimization

With system model defined in section 2, cost function and constraints defined in section 3.2, we can now proceed to construct the controller. The main concept behind MPC has been shown in figure 4. From system state and inputs sampled at time instants k an input sequence over a prediction horizon p is generated that results in an optimal output trajectory. The first value of this sequence is used as current input until state is measured again and the whole process is repeated.

We will refer to $\boldsymbol{X}_i$ and $\boldsymbol{U}_i$ as state and input sequences respectively, defined as:

$$\boldsymbol{X}_i = \begin{bmatrix} X_i & Y_i & \psi_i & v_i & \beta_i \end{bmatrix}^T \tag{13a}$$

$$\boldsymbol{U}_i = \begin{bmatrix} \dot{v}_i & \dot{\beta}_i \end{bmatrix}^T \tag{13b}$$

We will denote predictions as $\boldsymbol{X}_i^*$ and $\boldsymbol{U}_i^*$ to distinguish from actual history. We will also use a state transition function $\boldsymbol{f}$ that is an Euler–method–based discretization of state equations from section 2.1:

$$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) = \boldsymbol{x} + \dot{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{u})\Delta t \tag{14}$$

with sampling time Δt. The optimization can thus be written as:

$$\begin{aligned} \boldsymbol{V}_k^* = \min_{\boldsymbol{X}_{k+j}^*, \boldsymbol{U}_{k+j}^*} & \sum_{j=1}^{p} C\left(\boldsymbol{X}_{k+j}^*\right) \\ \text{s.t.} \quad & \boldsymbol{X}_{k+j+1}^* = \boldsymbol{f}\left(\boldsymbol{X}_{k+j}^*, \boldsymbol{U}_{k+j}^*\right), \\ & \boldsymbol{X}_{k+1}^* \quad = \boldsymbol{f}\left(\boldsymbol{X}_k, \boldsymbol{U}_k\right) \\ & v_{min} \leqslant v_{k+j} \leqslant v_{max}, \\ & v_{min} \leqslant \beta_{k+j} \leqslant v_{max}, \\ & v_{min} \leqslant \dot{v}_{k+j} \leqslant v_{max}, \\ & v_{min} \leqslant \dot{\beta}_{k+j} \leqslant v_{max}, \\ & v_{min} \leqslant A_{k+j} \leqslant v_{max}, \\ & j = 1, \ldots, p \end{aligned} \tag{15}$$

4 EXPERIMENTAL SETUP

To validate viability of the proposed solution, an experimental ride has been conducted using a roboticized vehicle based on a 1:8 scale car chassis, depicted in figure 1. Conceptual data flow in the entire system has been shown in figure 5. The following paragraphs describe in more details how it was implemented.

4.1 Hardware

The vehicle is based on Team Associated RC8B3.1e chassis. Actuations are provided via a high power Brushless Direct Current (BLDC) motor controlled by a dedicated Electronic Speed Control (ESC) module and a servomotor connected to the steering system. The frame has been modified to hold an ASRock Mini-ITX motherboard with an Intel i3 CPU, 8 GB RAM, and 128GB SSD. The car has been equipped with two Hokuyo URG-04LX-UG01 laser scanners, a small STM32 board for computer–ESC communication, a Wi-Fi module, an Inertial Measurement Unit (IMU), and a radio receiver for manual control. The setup is powered by four-cell, 14.8V lithium-polymer battery. The system measures 37,5 cm x 50 cm and weighs approximately 3 kg.

4.2 Software

Overall software architecture follows the one described in (Gotlib et al., 2019). The main on–board computer is running Ubuntu 16.04 configured with Robot Operating System (Quigley et al., 2009). During execution, laser scans combined with odometry data and a pre–built map of the environment are used as inputs to Adaptive Monte Carlo Localization (AMCL) system. Resulting position data, together with current velocity, steering, as well as waypoint of a reference path, are used to fill initial state for MPC implementation. Optimization is performed according to equation 15 by the use of IPOPT solver (Wächter and Biegler, 2006). Target acceleration and slip angle rate are integrated; resulting speed is used as a setpoint for ESC, and the angle is mapped to a respective servomotor position.

4.3 Preparation Procedure

Additional steps need to be taken before the vehicle is able to drive in autonomous mode. During an offline session, the car under manual operation covers the track to create a virtual map using Cartographer package (Hess et al., 2016). Then a reference path is generated with respect to that map and both are entered to the system.

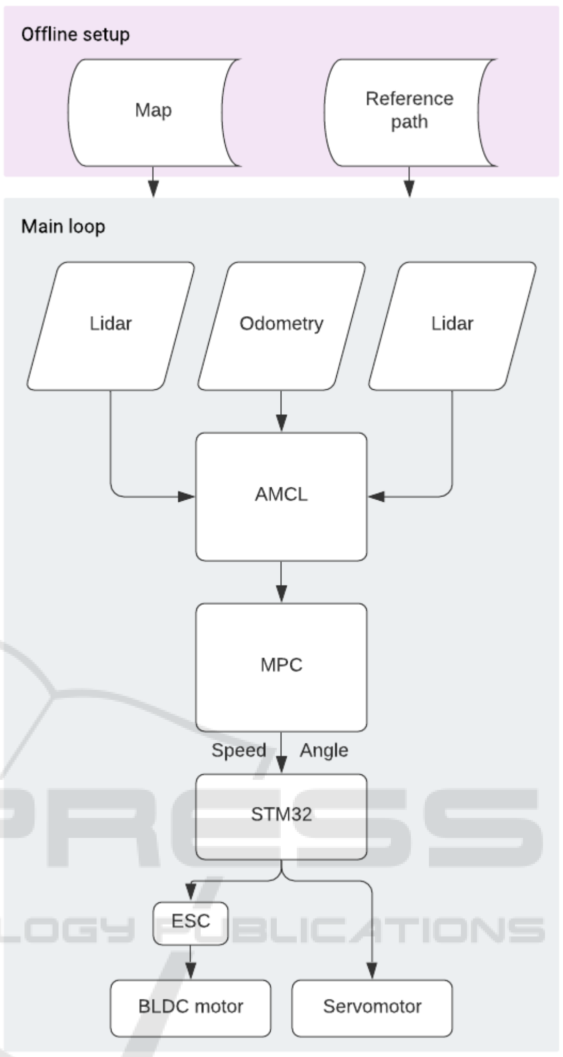

Figure 5: Data Flow in the Autonomous System.

5 RESULTS

Figure 6 visualizes data gathered from a 4–lap course around an example track. We can see a mismatch between the reference path and actual trajectory. The reason for that is that the controller tries to minimize curvature, similarly to what a driver does when following a racing line instead of the centerline of the track. Resulting trade–off between speed and accuracy can be regulated using coefficients c_1, c_2, and c_3 in the cost function.

The system is characterized by good repeatability. We can also see that the controller is able to make

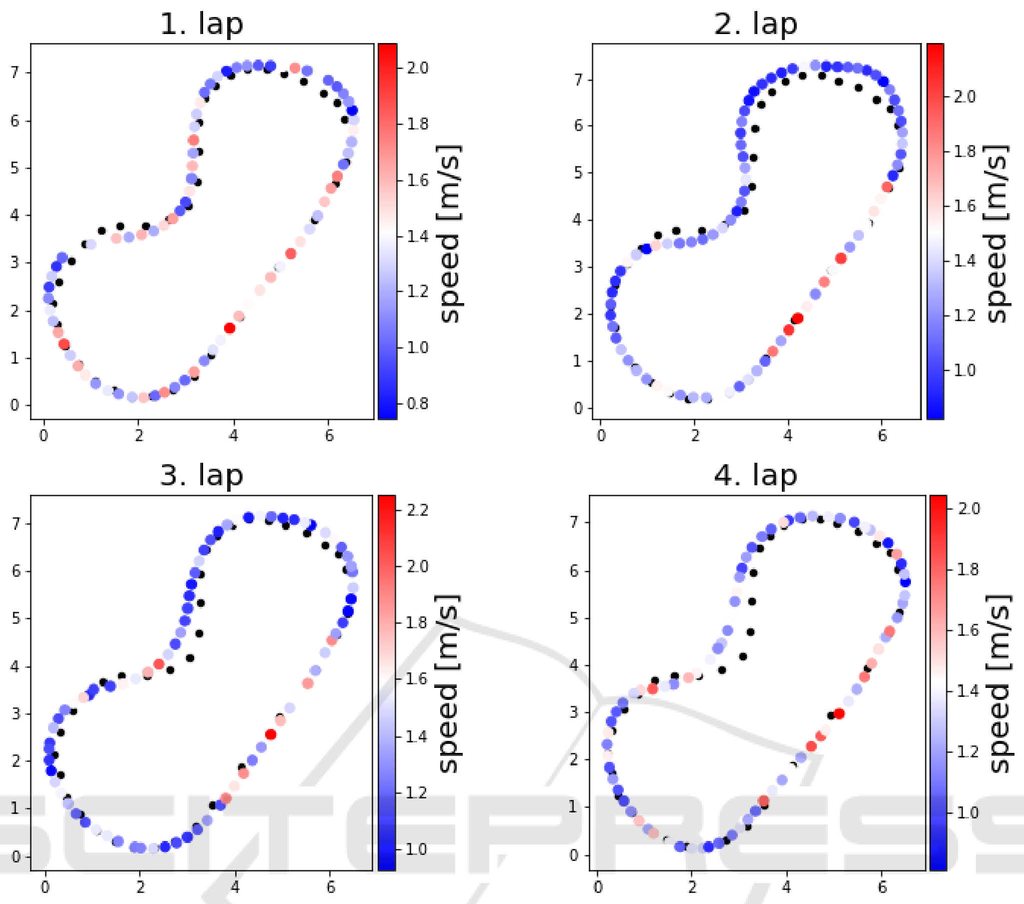

Figure 6: Reference (in Black) and Actual Path of the Vehicle during a 4–Lap Course around a Test Track.

correct decisions regarding vehicle velocity despite not being given any pre–computed speed profile. Figure 7 offers further insights into dynamics of the system. Distribution of accelerations peaks just below the limit circle (defined as a parameter), which means the requested actuations operate near the optimum during pure braking, cornering, and acceleration, but also in transient states, i.e. trail–braking and throttle–on–exit. However, what can be seen in comparison with the distribution of accelerations measured using on–board IMU, these actuations are not executed perfectly. They still peak near the limit circle, slightly exceeding it, but the spread in lateral direction is much larger, and in the longitudinal – much smaller. This most probably impacts negatively on overall performance.

6 CONCLUSIONS

All in all, despite issues with practical realization of requested actuations by the test platform, the vehicle operates correctly, what makes the approach presented here a valid basis for more advanced applications in the future. The biggest improvement can be brought by closer inspection of low–level control responsible for actuating the vehicle. Additional features, such as Contouring Control (Liniger et al., 2014) can be added to further extend capabilities of the system.

ACKNOWLEDGEMENTS

Authors are members of Students' Robotics Association (Koło Naukowe Robotyków) at the Faculty of Power and Aeronautical Engineering, Warsaw University of Technology.

Research done in cooperation with the Faculty of Power and Aeronautical Engineering within the framework of "Najlepsi z Najlepszych 4.0!" program.

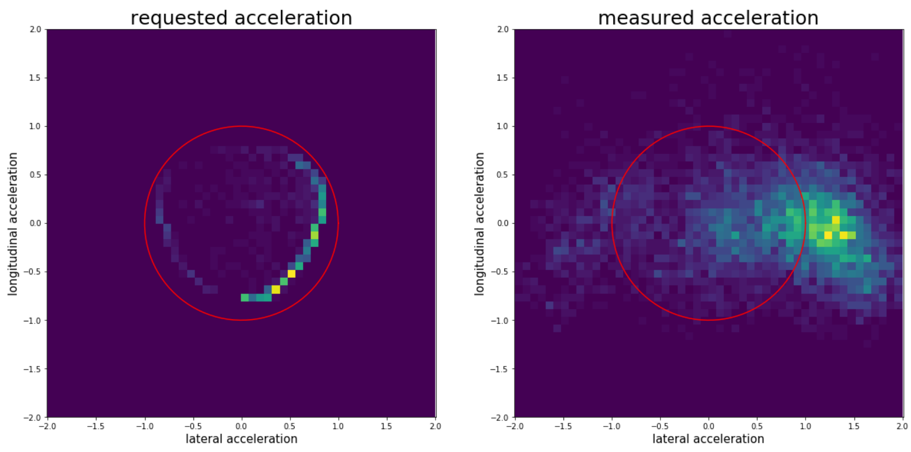

Figure 7: Distribution of Accelerations during the Test Drive.

REFERENCES

Allgöwer, F., Findeisen, R., and Nagy, Z. (2004). Nonlinear model predictive control: From theory to application. *J. Chin. Inst. Chem. Engrs*, 35:299–315.

Balaji, V., Balaji, M., Chandrasekaran, M., khan, M. A., and Elamvazuthi, I. (2015). Optimization of pid control for high speed line tracking robots. *Procedia Computer Science*, 76:147 – 154. 2015 IEEE International Symposium on Robotics and Intelligent Sensors (IEEE IRIS2015).

Faulwasser, T., Kern, B., and Findeisen, R. (2009). Model predictive path-following for constrained nonlinear systems. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 8642–8647.

Gotlib, A., Łukojć, K., and Szczygielski, M. (2019). Localization-based software architecture for 1:10 scale autonomous car. In *2019 International Interdisciplinary PhD Workshop (IIPhDW)*, pages 7–11.

Hess, W., Kohler, D., Rapp, H., and Andor, D. (2016). Real-time loop closure in 2d lidar slam. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278.

Kabzan, J., Hewing, L., Liniger, A., and Zeilinger, M. N. (2019a). Learning-Based Model Predictive Control for Autonomous Racing. *IEEE Robotics and Automation Letters*, 4(4):3363–3370.

Kabzan, J., Valls, M., Reijgwart, V., Hendrikx, H., Ehmke, C., Prajapat, M., Bühler, A., Gosala, N., Gupta, M., Sivanesan, R., Dhall, A., Chisari, E., Karnchanachari, N., Brits, S., Dangel, M., Sa, I., Dube, R., Gawel, A., Pfeiffer, M., and Siegwart, R. (2019b). Amz driverless: The full autonomous racing system.

Kritayakirana, K. and Gerdes, J. C. (2012). Autonomous vehicle control at the limits of handling.

Lam, D., Manzie, C., and Good, M. (2010). Model predictive contouring control. In *49th IEEE Conference on Decision and Control (CDC)*, pages 6137–6142.

Liniger, A., Domahidi, A., and Morari, M. (2014). Optimization-based autonomous racing of 1:43 scale rc cars. *Optimal Control Applications and Methods*, 36(5):628–647.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. (2009). Ros: an open-source robot operating system. volume 3.

Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.*, 106(1):25–57.

Williams, G., Drews, P., Goldfain, B., Rehg, J. M., and Theodorou, E. A. (2016). Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440.