2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

Deriving Overtaking Strategy from Nonlinear Model Predictive Control for a Race Car

Alexander Buyval, Aidar Gabdulin, Ruslan Mustafin and Ilya Shimchik[1]

Abstract— **Car racing control is often assumed to take place in isolation, although the greatest strategic challenges arise in the presence of other cars. Our solution uses a combined nonlinear system model which includes the four-wheeled vehicle dynamics model and point-mass model of its opponent car. This proposed combined model allows the optimal control scheme to automatically find an overtaking strategy. The FORCES code generation tool to handle the consequent computational complexity of the Nonlinear Model Predictive Controller (NMPC). We demonstrate the practicality of the suggested method by performing real-time control in a racing simulation software.**

I. INTRODUCTION

Currently, there is a large set of methods and algorithms for control of autonomous cars, trajectory generation and path planning. A number of approaches consider planning and control as separate tasks: one subsystem generates a trajectory which satisfies the specified criteria and given constraints while the other generates control signals to keep the car on the desired path. The main disadvantage of two-component systems comes from the fact that the generated path may be physically impossible since during the off-line calculation the dynamic characteristics of the control object are not taken into account. Trajectory planning algorithms can be divided into two categories: algorithms from the first category include a model of the control object, and algorithms in the second category do not use any model at all. It is worth to mention that for a significant amount of tasks performed by autonomous vehicles it is not necessary to consider dynamics of a car. Keeping only car's kinematic properties is sufficient for trajectory planning as the main criteria are safety and collision-free path. Such tasks are often solved by applications of the grid / graph search techniques [14], [22]. However, when a car operates close to the limits of handling it is not enough to consider only its kinematics and control algorithms should take into account car dynamics as well. In racing, for example, it is important to achieve minimal lap times, which implies that the vehicle operates at maximum possible velocity. Kinematic model will not be able to capture important non-linearities that arise from tire-road interaction, which defines car behaviour. Such conditions require a highly detailed dynamic model of the car in order to plan the trajectory and to ensure its feasibility.

Usage of the detailed description of car dynamics exposes the problem of the algorithm computational complexity and the question of how to make the algorithm feasible for the real-time car control becomes highly important. One of the possible solutions for that is to do a path planning which incorporates car dynamics in off-line mode as described in the [10], then use a feedback controller to follow precomputed trajectory [13]. This approach solves the feasibility problem for static trajectory, but it is not applicable if the car trajectory is generated on-line. For example, in order to plan an overtaking manoeuvre, the car needs to know not only its own position but also positions of other dynamic obstacles (competitors in the context of racing), which are impossible to estimate in the off-line generation phase.

There are studies where the effectiveness of model predictive control (MPC) for the race car [7], [15], [20] have been demonstrated. In MPC the trajectory planning and execution are performed at each step, which allows to take into consideration both the car dynamics and the competitors' position. Control and dynamic optimization frameworks have been used, such as ACADO [9] or FORCES [5], to make the MPC algorithm perform in the real-time applications.

In [15] there is a comparison of two approaches for controlling 1:43 scaled RC car model: Hierarchical receding horizon controller and Model predictive contouring control (MPCC). Both approaches used a simplified non-linear dynamic model with lumped front and rear wheels, also known as a bicycle model. The lateral forces in tires have been modelled with Pacejka Tire model [1]. The experimental results in [15] have shown the applicability of MPCC for real-time applications on ARM Cortex A15 processor. The code was generated using FORCES Pro software [5]. The main disadvantage of MPCC approach is the way it represents dynamic and static obstacles. Particularly, the obstacles are described as positive or negative constraints on lateral deviation from the reference trajectory. Such representation requires defining explicitly from which side an overtaking manoeuvre is to be performed. In [15] authors suggest using additional techniques for selecting the side. Another disadvantage is that dynamic obstacles during each step are represented as static obstacles, making it impossible to predict the trajectories of these objects and include them into optimization.

A similar approach with the usage of bicycle model has been demonstrated in [20]. Instead of FORCES framework, the ACADO code generation tool has been used for the dynamic optimization. Experimental results on RC car has shown the feasibility of MPC in real time control.

In [7] authors used four wheeled car dynamic model, which includes the dynamics of wheels and distribution of

[1]Alexander Buyval, Aidar Gabdulin, Ruslan Mustafin and Ilya Shimchik are with the Laboratory of Intelligent Robotics System (LIRS), Innopolis University, Innopolis, Russia, email: {a.buyval, a.gabdullin, r.mustafin, i.shimchik}@innopolis.ru, https://university.innopolis.ru/en/research/robotics/lirs/

vertical forces. In this paper the control inputs are force moments for each wheel. Authors have shown the efficiency of ACADO toolkit for solving such optimization problems in real time.

In [3], a neural network has been applied to perform overtaking in different parts of a track. The overtaking behaviour was activated on top of the main driving behaviour at each time when an opponent became closer than a defined threshold. Loiacono et al. [16] used reinforcement learning to learn to overtake on a straight path and overtaking close to a turn. Onieva et al. [18] suggested to use a fuzzy system for car racing control with overtaking strategy, there is a comparison of presented overtaking strategy with others and the effectiveness of fuzzy logic for the task of overtaking has been shown. However, mentioned papers do not consider dynamics of the car and opponent, which makes overtaking impossible on high speeds.

Nikolce Murgovski and Jonas Sjoberg [17] studied the problem of optimal control of an autonomous vehicle to safely overtake a slow-moving vehicle. There are some assumptions which make the approach not applicable for a racing car: (I) within the prediction horizon the road is considered straight and the leading vehicle is considered to travel with a constant longitudinal speed; (II) the vehicle is modelled as a point mass linear system. The main contribution of this work in comparison to previous ones is utilizing collaborative vehicle and moving opponent nonlinear model to get optimal overtaking trajectory in critical handling conditions.

II. SYSTEM MODEL

As it was mentioned earlier, [15] and [20] used a simple bicycle model with lumped tires and 6 degrees of freedom (DOF). Such models are widely used, however, they do not consider such factors as weight transfer during cornering and do not allow to estimate independent tire-road friction coefficients for each wheel. On the other hand, the model used in [7] appears to be computationally expensive for the required planning horizon as it calculates dynamics and wheel vertical forces separately. As such, in our problem we implemented only the effects of weight transfer, leaving rotational dynamics out for performance reasons.

A. *Chassis Dynamics*

The vehicle chassis is modelled as a rigid body, described by its global position in the X-Y plane, its global orientation, and the corresponding velocities in a local X-Y-Z frame. Then the time-dependent dynamic system is eventually transformed into a track progress-dependent one in a similar fashion as presented in [7]. The chassis dynamic equations used in this paper are presented in (1a)-(1e)

$$m\dot{v}^x = F^x_{fr} + F^x_{fl} + F^x_{rr} + F^x_{rl} + m v^y \dot{\psi}, \tag{1a}$$

$$m\dot{v}^y = F^y_{fr} + F^y_{fl} + F^y_{rr} + F^y_{rl} - m v^x \dot{\psi}, \tag{1b}$$

$$\begin{aligned} I^z \ddot{\psi} =& a(F^y_{fl} + F^y_{fr}) - b(F^y_{rl} + F^y_{rr}) \\ &+ c(F^x_{fr} - F^x_{fl} + F^x_{rr} - F^x_{rl}), \end{aligned} \tag{1c}$$

$$\dot{x} = v^x \cos\psi - v^y \sin\psi, \tag{1d}$$

$$\dot{y} = v^x \sin\psi + v^y \cos\psi, \tag{1e}$$

where m denotes the mass and I^z the moment of inertia of the car. The geometric parameters of the car are characterized by a, b and c, in Fig. 1. The components of the tire contact forces are denoted by $F^x_{..}$ and $F^y_{...}$

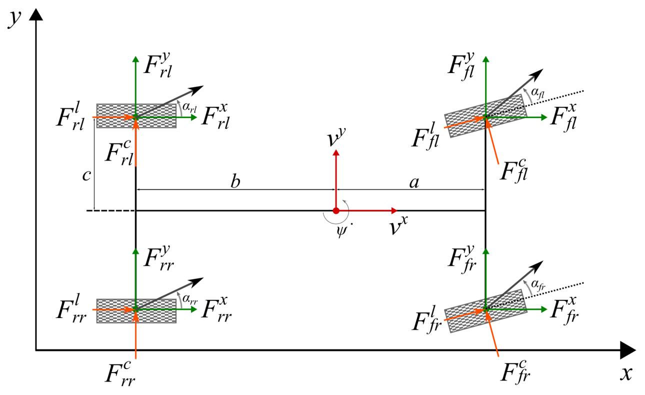

Fig. 1. The 4-wheel vehicle model in inertial coordinates

As we do not consider rotational dynamics of a wheel, combined Pacejka model cannot be applied to calculate friction force components as it was done in [7]. Additionally, we based our work on assumptions that the vehicle will be equipped with anti-locking braking system (ABS) and launch control system that will prevent longitudinal wheel slip. Given these assumptions, we used Pacejka model to calculate only lateral friction. The longitudinal estimation was done based on assumption that the vehicle is an electrically powered all-wheel-drive (AWD) car.

The equation for calculation longitudinal tyre forces is presented in (2).

$$F^l_{..} = ((C_{m1} + C_{m2} v_x) D_c + C_r + C_{ar} v_x^2 + C_b B_c)/4 \tag{2}$$

where C_{m1} and C_{m2} are parameters of the DC electric motor, C_b is brake system parameter, C_r and C_{ar} are parameters of rolling and air resistance forces respectively. The control inputs are the PWM duty cycle D_c of the electric drive train motor, normalized brake force B_c and the steering rate δ_r.

For system identification we did a number of tests that were conducted inside the simulator. Then we used an optimization-based approach for parameter identification based on sensory data that we received during the aforementioned tests. C_r and C_{ar} parameters were identified using a test in which the vehicle was given an initial velocity and all control inputs were set to zero. The engine parameters were identified using an acceleration test from stationary position with D_c set to a constant value. Similarly, C_b was identified with B_c set to a constant value. See [21] for details.

B. Vertical Forces Distribution

To consider such effects as aerodynamic downforce and weight transfer when cornering, we used (3a)-(3f) to recalculate vertical load acting on each wheel.

$$\Delta F^z_f = (\bar{F}^z_{fl} + \bar{F}^z_{fr})C_t\dot{\psi} \tag{3a}$$
$$F^z_{fl} = \bar{F}^z_{fl} - \Delta F^z_f + C_a v_x \tag{3b}$$
$$F^z_{fr} = \bar{F}^z_{fr} + \Delta F^z_f + C_a v_x \tag{3c}$$
$$\Delta F^z_r = (\bar{F}^z_{rl} + \bar{F}^z_{rr})C_t\dot{\psi} \tag{3d}$$
$$F^z_{rl} = \bar{F}^z_{rl} - \Delta F^z_r + C_a v_x \tag{3e}$$
$$F^z_{rr} = \bar{F}^z_{rr} + \Delta F^z_r + C_a v_x \tag{3f}$$

where $F^z_{..}$ are current vertical forces, $\bar{F}^z_{..}$ are normal vertical forces on each wheel, C_t is a coefficient of transfer load during turning and C_a is a coefficient of air dynamic pressure. Both were identified by corresponding numerical tests.

C. Spatial Reformulation of Dynamics

In this paper, we propose to use a track-dependent (spatial) dynamics instead of time-dependent vehicle dynamics, similar to [7], [11], [8]. Such approach allows a natural formulation of obstacles and road bounds under varying vehicle speed. Similar to [7] we project the inertial X-Y coordinates on a curve σ which could be the center-line of a road or optimized in offline mode trajectory. The ODE model is then stated w.r.t. the independent variable s, the parametrization of σ by its arc-length. Then we replaced the x,y, psi states by e^y and e^ψ. Fig. 2 describes the states in the new curvilinear coordinate system. We refer to [7] and [8] for more details on the track-dependent (spatial) dynamics.

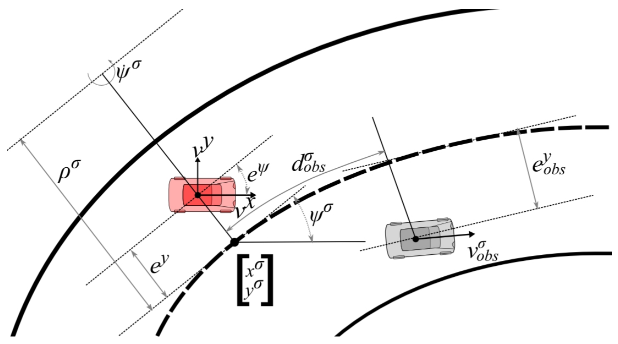

Fig. 2. The curvilinear coordinate system.

D. Additional states for overtaking

We propose to extend the list of state variables inside the system with the distance to the nearest dynamic obstacle along the path. This allows to consider obstacle's velocity along with our own dynamics when planning an optimal trajectory. The velocity itself can be obtained from the outer system (ex. V2I, V2V systems) or can be estimated onboard using Kalman filter. We leave this problem out of scope of this paper. Therefore, it is worth mentioning that we decided to not include full dynamical model of the obstacle for a number of reasons. First, inclusion of full model would result in an increase in the number of state variables, leading to dramatic growth in time it takes to solve the optimization problem. Second reason is that we have no data about obstacle's dynamical parameters which makes modeling difficult and largely inaccurate.

Additionally, we included two parameters that describe the lateral error and the velocity along the reference trajectory of the dynamic obstacle.

The final system dynamic equations used in this paper are presented in (4a)-(4h).

$$\dot{e}^y = (v^x \sin e^\psi + v^y \cos e^\psi)/\dot{s}, \tag{4a}$$
$$\dot{e}^\psi = \dot{\psi}/\dot{s} - \kappa^\sigma(s), \tag{4b}$$
$$\dot{v}^x = (F^x_{fr} + F^x_{fl} + F^x_{rr} + F^x_{rl} + mv^y\dot{\psi})/(m\dot{s}), \tag{4c}$$
$$\dot{v}^y = (F^y_{fr} + F^y_{fl} + F^y_{rr} + F^y_{rl} - mv^x\dot{\psi})/(m\dot{s}), \tag{4d}$$
$$\begin{aligned}\ddot{\psi} =&(a(F^y_{fl} + F^y_{fr}) - b(F^y_{rl} + F^y_{rr}) \\ &+ c(F^x_{fr} - F^x_{fl} + F^x_{rr} - F^x_{rl}))/(I^z\dot{s}),\end{aligned} \tag{4e}$$
$$\dot{t} = 1/\dot{s}, \tag{4f}$$
$$\dot{\delta} = \delta_r, \tag{4g}$$
$$\dot{d^\sigma_{obs}} = (-v^x \cos e^\psi + v^y \sin e^\psi + v^\sigma_{obs}(s))/\dot{s}, \tag{4h}$$

where d^σ_{obs} is distance to the closest obstacle along the reference path, $\kappa^\sigma(s)$ is the local curvature of the track, $v^\sigma_{obs}(s)$ is current and predicted velocity of the nearest car along track, δ is steering angle and $\dot{\delta}$ is steering rate.

III. NONLINEAR MPC FORMULATION

Based on the previous section we can write the state vector x and the vector u of continuous controls as:

$$x = (e^y, e^\psi, v^x, v^y, \dot{\psi}, t, \delta, d^\sigma_o)^T, u = (D_c, B_c, \delta_r)^T \tag{5}$$

If we use f to denote the right-hand-side function of the ODE system, the resulting optimal control problem is formulated in the following way:

$$\begin{aligned}\min_{x(),u()} \; t(s_f) + \int_{s_0}^{s_f} & \|x(\tau) - x_{ref}(\tau)\|^2_Q \\ & + \|u(\tau) - u_{ref}(\tau)\|^2_R \\ & + c_b e^{-((e^y(\tau) - e^y_{obs}(\tau))^2 + d^{\sigma 2}_{obs}(\tau))}\end{aligned} \tag{6a}$$
$$s.t. \dot{x}(s) = f(s, x(s), u(s), p(s)), \tag{6b}$$
$$x(s) \in [x_L(s), x_U(s)], \tag{6c}$$
$$\delta_r(s)v_x(s) \in [\delta_L, \delta_U], \tag{6d}$$
$$D_c \in [0,1], \tag{6e}$$
$$B_c \in [0,1], \tag{6f}$$
$$x(0) = x_o \tag{6g}$$

where $e^y_{obs}(\tau)$ is a lateral error of the closest car from the reference trajectory, $p(x)$ are real time parameters such as $\kappa^\sigma(s)$ and $v^\sigma_{obs}(s)$. The initial condition is denoted by $x_0 \in R$.

$x_L(s)$ and $x_U(s)$ denote lower and upper bounds for the n-dimensional state vector. The least-squares objective function (6a) aims at tracking a state reference trajectory x_{ref}, taking a control regularization term and obstacle avoiding term into account. Weighting matrices are given by $Q \in R_{n\times n}$ and $R \in R_{m\times m}$.

It is worth mentioning that $c_b e^{-((e^y(\tau)-e^y_{obs}(\tau))^2+d^{\sigma 2}_{obs}(\tau))}$ is a component of the objective function which implements the overtaking strategy inside NMPC. This component serves as a barrier function, with its value increasing as parameters reach their critical levels. The algorithm then attempts to find a trajectory that would either maximize d^{σ}_{obs}, or $e^y(\tau) - e^y_{obs}(\tau)$.

This specification for moving and static obstacles has a number of advantages from using dynamical constraints in the form of $[e^y_L(s), e^y_U(s)]$, introduced in [15]. First, NMPC has a freedom to choose which side to overtake a target obstacle. Second, this approach uses obstacle's velocity information to calculate the possibility of performing the overtaking. In case it is not possible to overtake at the given moment, the controller will match obstacle's velocity and avoid the collision. The presented approach is also more robust in terms of finding the solution, as there is a situation where forming dynamic constraints in the form of upper and lower bounds may lead to no solution (when the obstacle is too close to the vehicle, see Fig. 3 for details).

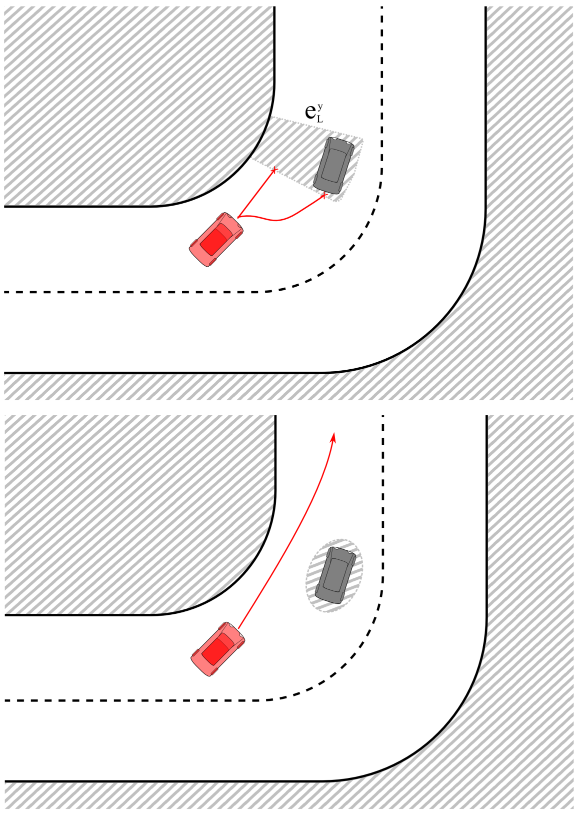

Fig. 3. top: Dynamic obstacle representation as bounds on eY. bottom: Dynamic obstacle representation in the form of a barrier function

IV. SOLUTION METHOD

We use the FORCES Pro Code Generation tool [5] that was already successfully used for high-speed nonlinear MPC of scale 1:43 RC cars in [15]. In particular, efficient interior point solvers for embedded systems, generated by FORCES [6], are employed to solve the resulting optimization problems. Tailored interior point solver, generated by FORCES is used to solve the resulting QP. After the QP has been solved, the first control input is applied to the system, and the process is repeated at the next sampling time. This approach for solving the NMPC problems corresponds to the basic version of the real-time iterations from [4]. The exported solver is provided as a static library and includes an implicit Runge-Kutta integrator of order 4, as well as a static memory allocation. We chose diagonal weighting matrices $Q = diag[0.7, 0.05, 10^{-6}, 10^{-6}, 10^{-6}, 10, 10^{-6}]$ and $R = diag[6, 2, 3 \times 10^{-3}]$. In addition, to decrease the solving time of the QP we initialize optimize variables with result values from previous MPC iteration.

V. SIMULATION EXPERIMENTS

We used Speed Dreams simulation software to test our approach. Speed Dreams has a similar architecture to another popular open source racing simulator TORCS [2], which has already been used in a number of scientific publications.

Fig. 4. Screenshot from a Speed-Dreams race during overtaking

To control a virtual car, we have implemented a ROS node over UDP interface, provided by Speed Dreams. Utilizing ROS allowed us to make debugging and logging easier. All simulation results were obtained on a PC with the following specs: Intel i5-6400 CPU at 2.7GHz under Ubuntu 14.04 and ROS Jade. The vehicle parameters are as follows: $m = 1000kg$, $I^z = 600kgm^2$, $a = 1.104m$, $b = 1.469m$, $c = 0.81m$. DC engine parameters were identified by tests mentioned above: $C_{m1} = 16500$, $C_{m2} = 135$, as well resistance parameters: $C_r = 2315$, $C_{ar} = 41$. Tyre parameters were identified during the ramp steer maneuver.

We have chosen 3 meters for a length of planning horizon step. This was motivated by the distance the car would travel during one iteration of MPC and was dictated by achievable computation times. Steps with greater length allow to plan further on the track with the same number of iterations, however, longer steps would also yield less precise results.

For testing purposes, we used a race track which contains both straight segments and sharp hairpin turn with an overall length of 3093 meters. Fig. 5 represents the trajectory of a test run on this track with a condition of no rivals presents and planning horizon length N = 25. We need to say that per reference trajectory we used the middle line of the track and maximum speed of 40 m/s.

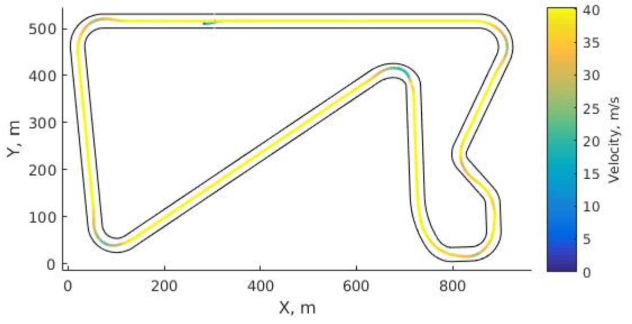

Fig. 5. Trajectory of a test run over using NMPC with N=25

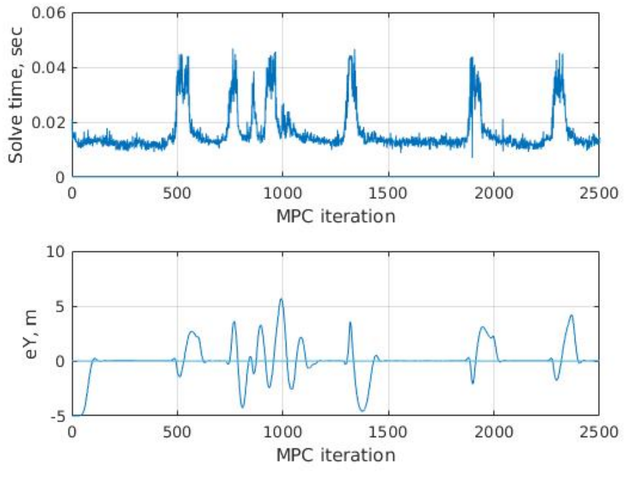

Fig. 6. Simulation results for NMPC with N=25

Additionally, we have executed several simulation experiments with a different number of steps in the planning horizon. The purpose of these experiments was to find the optimal number of steps which could allow car control at the rate of 25 Hz with a minimal lap time. Table I presents the results for two different number of steps. We should notice that the control rate of 25 Hz can only be reached with N=25 and the maximum speed of 40 m/s.

Fig. 6 represents the time it takes to solve the optimization problem compared to values of eY at corresponding MPC iteration number. In this figure, the spikes in solution time correspond to cases when the vehicle reaches boundary conditions on eY.

Fig. 7 shows the test runs on one particular turn with two different planning horizons and speed limits.

TABLE I

SOLVE TIME, MAXIMUM SPEED AND LAP TIME FOR DIFFERENT PLANNING HORIZON LENGTHS

N	t_{min}, s	t_{max}, s	t_{mean}, s	$V_{max}, m/s$	t_{lap}, s
25	0.008	0.045	0.014	40	79
40	0.018	0.076	0.031	53	73

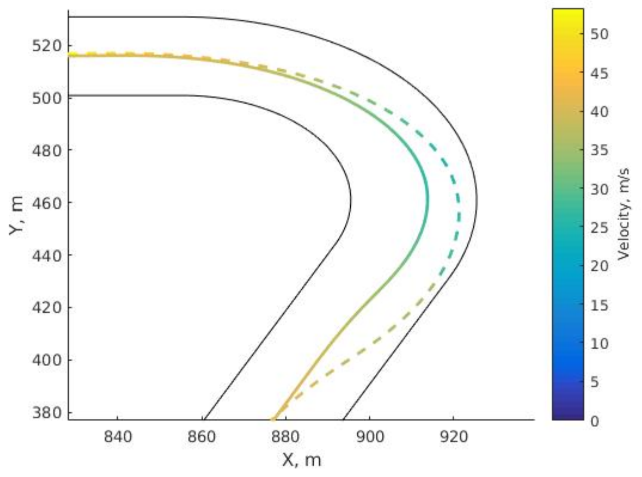

Fig. 7. Cornering trajectory with different settings for planning horizon length: solid line N=25, dashed line N=40

Additionally, we have conducted a number of tests inside the simulation, including overtaking of moving obstacles. To control the other vehicle we used a control method described in [12]. Middle lane of the track was used as a reference trajectory for both our and competitor's vehicle guaranteeing that the competitor's vehicle will obstruct our path. However, any other trajectory can be used. But calculating the such paths does not lie in the scope of our work.

Competitor's target velocity was limited to a constant value. Fig. 8 demonstrates overtaking on a straight segment. Numbers along trajectories show position in corresponding time moments for both opponents. Fig. 9 demonstrates overtaking on a more difficult segment of the track.

The video of the overtaking in different conditions in Speed Dreams is available through the following link: `https://youtu.be/ifh9GCkE8to`.

VI. CONCLUSION

We presented the Nonlinear MPC algorithm that allows generation of feasible trajectories in real-time. Our approach is capable of generating collision-free overtaking trajectories by taking both moving and static obstacles into account during the optimization. We used a combination of the controlled vehicle's dynamic model and the obstacle model. Experiments in the simulator have shown both real-time feasibility and effectiveness of the proposed method.

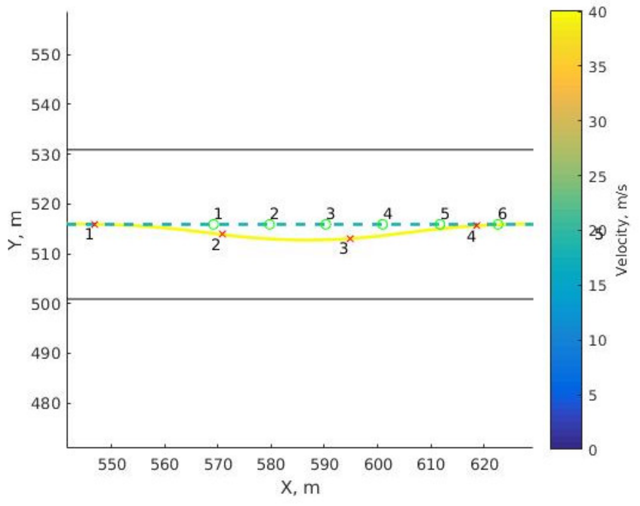

Fig. 8. Overtaking trajectory on a straight segment

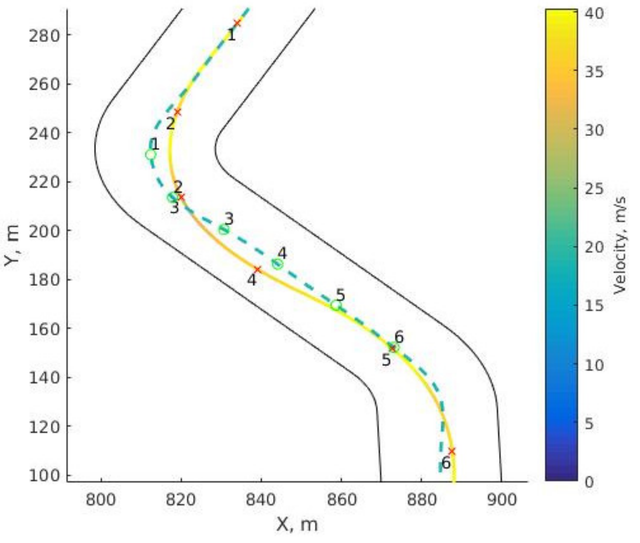

Fig. 9. Trajectory of overtaking on a segment between turns

VII. FUTURE WORK

Future work includes studying efficiency of using offline optimized reference trajectory. Also, we plan to automatically generate coefficients for the NMPC based on optimization techniques. We consider learning MPC approach for iterative tasks described in [19]. Moreover, we are going to deploy the algorithm on a real race car and have started testing it on a 1:10 scale RC car powered by Jetson TX1.

REFERENCES

[1] Egbert Bakker, Lars Nyborg, and Hans B Pacejka. Tyre modelling for use in vehicle dynamics studies. Technical report, SAE Technical Paper, 1987.

[2] Christophe Guionneau Christos Dimitrakakis Rémi Coulom Andrew Sumner Bernhard Wymann, Eric Espié. TORCS, The Open Racing Car Simulator. http://www.torcs.org, 2014.

[3] Luigi Cardamone, Daniele Loiacono, and Pier Luca Lanzi. Evolving competitive car controllers for racing games with neuroevolution. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1179–1186. ACM, 2009.

[4] Moritz Diehl, H Georg Bock, Johannes P Schlöder, Rolf Findeisen, Zoltan Nagy, and Frank Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4):577–585, 2002.

[5] Alexander Domahidi and Juan Jerez. FORCES Professional. embotech GmbH (http://embotech.com/FORCES-Pro), July 2014.

[6] Alexander Domahidi, Aldo U Zgraggen, Melanie N Zeilinger, Manfred Morari, and Colin N Jones. Efficient interior point methods for multistage problems arising in receding horizon control. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 668–674. IEEE, 2012.

[7] J. V. Frasch, A. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl. An auto-generated nonlinear mpc algorithm for real-time obstacle avoidance of ground vehicles. In *2013 European Control Conference (ECC)*, pages 4136–4141, July 2013.

[8] Yiqi Gao, Andrew Gray, Janick V Frasch, Theresa Lin, Eric Tseng, J Karl Hedrick, and Francesco Borrelli. Spatial predictive control for agile semi-autonomous ground vehicles. In *Proceedings of the 11th international symposium on advanced vehicle control*, 2012.

[9] B. Houska, H.J. Ferreau, and M. Diehl. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011.

[10] Nitin R Kapania, John Subosits, and J Christian Gerdes. A sequential two-step algorithm for fast generation of vehicle racing trajectories. *Journal of Dynamic Systems, Measurement, and Control*, 138(9):091005, 2016.

[11] F. Kehrle, J.V. Frasch, C. Kirches, and S. Sager. Optimal control of Formula 1 race cars in a VDrift based virtual environment. In S. Bittanti, A. Cenedese, and S. Zampieri, editors, *Proceedings of the 18th IFAC World Congress*, pages 11907–11912, Milan, Italy, August 28–September 2 2011.

[12] Krisada Kritayakirana and J Christian Gerdes. Autonomous vehicle control at the limits of handling. *International Journal of Vehicle Autonomous Systems*, 10(4):271–296, 2012.

[13] Krisada Kritayakirana and J Christian Gerdes. Using the centre of percussion to design a steering controller for an autonomous race car. *Vehicle System Dynamics*, 50(sup1):33–51, 2012.

[14] Yoshiaki Kuwata, Gaston A Fiore, Justin Teo, Emilio Frazzoli, and Jonathan P How. Motion planning for urban driving using rrt. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1681–1686. IEEE, 2008.

[15] Alexander Liniger, Alexander Domahidi, and Manfred Morari. Optimization-based autonomous racing of 1:43 scale rc cars. *Optimal Control Applications and Methods*, 36(5):628–647, 2015.

[16] Daniele Loiacono, Alessandro Prete, Pier Luca Lanzi, and Luigi Cardamone. Learning to overtake in torcs using simple reinforcement learning. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.

[17] Nikolce Murgovski and Jonas Sjöberg. Predictive cruise control with autonomous overtaking. In *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*, pages 644–649. IEEE, 2015.

[18] Enrique Onieva, Luigi Cardamone, Daniele Loiacono, and Pier Luca Lanzi. Overtaking opponents with blocking strategies using fuzzy logic. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 123–130. IEEE, 2010.

[19] Ugo Rosolia and Francesco Borrelli. Learning model predictive control for iterative tasks. *CoRR*, abs/1609.01387, 2016.

[20] R. Verschueren, S. De Bruyne, M. Zanon, J. V. Frasch, and M. Diehl. Towards time-optimal race car driving using nonlinear mpc in real-time. In *53rd IEEE Conference on Decision and Control*, pages 2505–2510, Dec 2014.

[21] Mario Zanon, Janick V Frasch, and Moritz Diehl. Nonlinear moving horizon estimation for combined state and friction coefficient estimation in autonomous driving. In *Control Conference (ECC), 2013 European*, pages 4130–4135. IEEE, 2013.

[22] Julius Ziegler and Moritz Werling. Navigating car-like robots in unstructured environments using an obstacle sensitive cost function. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 787–791. IEEE, 2008.