A Real-Time NMPC Controller for Autonomous Vehicle Racing

Nan Li
LTCI, Télécom Paris
Institut Polytechnique de Paris
Palaiseau, France
nan.li@telecom-paris.fr

Eric Goubault
LIX, CNRS, Ecole Polytechnique
Institut Polytechnique de Paris
Palaiseau, France
goubault@lix.polytechnique.fr

Laurent Pautet
LTCI, Télécom Paris
Institut Polytechnique de Paris
Palaiseau, France
laurent.pautet@telecom-paris.fr

Sylvie Putot
LIX, CNRS, Ecole Polytechnique
Institut Polytechnique de Paris
Palaiseau, France
putot@lix.polytechnique.fr

***Abstract*—Non-linear model predictive control (NMPC) solves structured optimization problems under predetermined constraints. It results in an optimal control series in a multiple-step prediction horizon. However, the NMPC requires considerable computation time, making it difficult to implement on devices with limited resources. We focus on an NMPC-based controller used for autonomous vehicle racing. It is a typical representative of quickly evolving cyber-physical systems. In the single-vehicle racing mode, we propose a triggering method to enable the execution of long-horizon NMPC, which is desirable for achieving a better lap time. For the head-to-head racing mode, in order to react rapidly to the evolving surroundings, we propose a short-horizon NMPC-based control strategy with safe overtaking capability. These control strategies can be implemented within a limited time budget.**

***Keywords*—NMPC, real-time systems, autonomous vehicle racing**

I. Introduction

Non-linear model predictive control (NMPC) is widely used to control systems with non-linear constraints and dynamics. However, due to its high calculation costs, it is typically used in systems that change slowly. In this paper, we concentrate on its application in the autonomous race car system [1], in order to investigate the viability of running NMPC in a rapidly evolving system.

Sequential quadratic programming (SQP) is a popular technique to solve NMPC problems: starting from an initial value for system states, the original problem is sequentially and linearly approximated by QPs; the state value is updated after each iteration and finally converges by using Newton directions delivered by QPs. As in the work [2], NMPC is proposed to achieve the shortest lap time for autonomous race cars. In the work [3], the authors use SQP with only one QP step and separate the calculation into preparation/feedback phases, which is called the Real-time Iteration (RTI) method, to reduce computation cost. The underlying concept is that by shifting the result of the last NMPC calculation, the authors have a reasonable initialization for the next NMPC calculation. Instead of sequentially solving multiple QP sub-problems, the authors solve only one QP. The NMPC can still produce a solution relatively close to the one that it would have calculated with the traditional method, thanks to its efficient initialization.

We approach the problem of the computation time load in a different way. In single-vehicle racing mode, the race car evolves in a relatively static environment. We can observe that the resulting trajectory of the NMPC is significantly changed only when the vehicle is about to run into different track types, e.g. from a straight line to a sharp corner. This means that it is only necessary to recalculate the NMPC under this specific circumstance. Based on this fact, we propose a triggering condition to reduce total calling times to the NMPC solver while maintaining a sufficient level of system performance. This mechanism also allows us to run long-horizon NMPC without exceeding the given time budget. It is worth mentioning that the NMPC is calculated by the SQP method until its convergence. In this way, the convergence criteria of the NMPC can naturally guarantee that the system constraints are not violated, whereas the RTI method does not.

Triggering-based control is well considered in path tracking problems. For example, authors in [4] propose two triggering modes: self-triggering and event-triggering. The difference is that the event-triggered mode relies on the environment and requires continuous measurement of system state(s), while in the self-triggered mode, the next triggering point is pre-computed based on the current state and predicted control sequence. In this work, we use a triggering mode similar to self-triggering to handle our time-optimal autonomous racing problem in the single-vehicle racing mode. We will show that with this triggering method, the number of recalculation steps is largely reduced and the performance in terms of lap time is comparable to the conventional NMPC which recalculates every progress step.

In the head-to-head racing mode where the ego vehicle and one opponent vehicle run together, we propose a different approach. A quick reaction time, i.e. a short enough control period, is needed to avoid a potential collision with the opponent. Also, the procedure of SQP should always be completed to ensure the satisfaction of system constraints (especially collision-free constraints), thus the RTI method is not desirable for this racing mode. To cut down the computation time below a given time budget (or a pre-determined control period), we employ a short-horizon NMPC. In this way, we can recalculate

This work is supported by the project CIEDS/FARO.

the NMPC with high frequency. In head-to-head competition, the correctness of the result (which should be delivered within the given time budget) is of paramount importance. We may slightly degrade the lap time performance by using short-horizon NMPC, but it is worth making this trade-off to guarantee the functional correctness.

In order to integrate the constraint of non-collision between vehicles into the NMPC approach, we formulate the overtaking procedure as a sequence of choice of the ego vehicle. At each forecast stage, the ego vehicle can choose to overtake the leading vehicle either from the left or from the right. The overtaking procedure then consists in finding the combination of choices to overtake as quickly as possible. As reported in [5], mixed-integer programming (MIP) can be used to solve this combination problem for finding an overtaking strategy in the head-to-head racing scenario. However, the calculation time of NMPC combined to MIP is very expensive and it is difficult to employ in real-time scenarii. We propose a simplified control strategy for selecting between the left and right sides to achieve overtaking while also maintaining fast computation. Authors in [6] decide the overtaking side thanks to a learning-based method and only carry out the computation for the chosen side. In our work, the trajectories on both sides are computed in parallel (on different CPU cores) and the better one (which results in a faster progress time) is selected.

Contributions.

- For the single-vehicle racing mode, we propose a self-triggering method to reduce the number of recalculation steps of NMPC without violating the time budget constraint or considerably sacrificing system performance.
- For the head-to-head racing mode, we propose a control strategy with a fixed control period. This strategy featuring the overtaking ability is feasible for real-time scenarii.

Organization.

The article is structured as follows. In section II, we present the NMPC formulation and the conventional control framework that forms the basis for the parts that follow. In section III, we describe the issue that we intend to solve. We introduce our solutions to enable NMPC-based controllers to run under limited computation resources in section IV. Simulation results are reported in section V.

II. Background

In this section, we first present the NMPC formulation for the autonomous vehicle racing problem. We then describe how the NMPC's resulting optimal control is deployed in a conventional control framework.

A. The NMPC formulation

The whole system can be abstracted as the interaction between the plant (i.e. the race car) and the NMPC controller. We use the same NMPC formalization as in [5] which optimizes the vehicle's progress time in a curvilinear coordinate system to achieve the objective of time-optimal control. In this curvilinear coordinate system, the trajectory is parameterized by the arc-length s along the track's center line which serves as a reference. We define s as the vehicle's progress. The vehicle's center of gravity (CoG) has a projection point on this reference line. We note the deviation of the vehicle's CoG from the reference line as e_y. The difference between the vehicle's orientation and the tangent angle at the projection point is noted as e_ψ. The vehicle's position in the curvilinear coordinate is thus represented as (s, e_y, e_ψ). At progress step k in the prediction horizon, the vehicle's progress will be $s_k = s_0 + k \cdot \Delta s$, where s_0 is the current progress along the reference line and Δs is the step size. Since control in the NMPC horizon is piece-wise constant for each step, a longer Δs will reduce control flexibility but also safety guarantee. However, a shorter Δs will increase control complexity since more control steps are needed for realizing the same horizon length. In the following experiment, we set Δs to a similar value as the vehicle length to reach a compromise between safety and complexity.

We recall the NMPC formalization used in [5]:

$$\begin{aligned} \min_{u_i(s)} \quad & t_N \\ s.t. \quad & \xi'_{i+1} = f_{dyn}(\xi_i, u_i),\ i = 0, ..., N \\ & \xi_i \in [\xi^{\min}, \xi^{\max}],\ i = 0, ..., N \\ & u_i \in [u^{\min}, u^{\max}]\ i = 0, ..., N-1 \end{aligned} \tag{1}$$

where the state vector $\xi_i = [e_y, e_\psi, v_x, v_y, \omega, t, s, d, \delta]_i$, the control vector $u_i = [\Delta d, \Delta\delta]_i$. v_x, v_y are the longitudinal and lateral velocities. ω is the angular velocity of the vehicle's yaw angle. t is the progress time. d is the parameter of the motor engine. δ is the vehicle's steering angle. Δd and $\Delta\delta$ are the variation ratios of d and δ. In this NMPC formulation, the prediction horizon has a length of N. The optimization objective is the progress time at the end of the horizon: t_N. System states at different steps should respect the constraints on dynamic evolution. States and controls are also constrained by physical limitations in the form of intervals.

This paper uses the SQP technique to solve the above NMPC formulation. The procedure is explained as follows. We first provide guessed values for $\xi_i, i = 0, \ldots, N$ and $u_i, i = 0, \ldots, N-1$. These guesses result from previous calculations or specific methods such as those introduced in [7]. The initialization quality will have an impact on how fast the algorithm converges to the optimal result. Then we linearize the dynamics equation and constraints around the initialized state/control vector and solve the problem in the form of QP. After solving a single QP, we generate a new N-steps prediction for (ξ_i, u_i). If the corresponding solution does not reach the required precision, we linearize again the dynamics equation/constraints around the newly generated state/control vector, then resolve a new QP problem. We use the Karush–Kuhn–Tucker (KKT) condition as the precision indicator for denoting the optimality of the solution and the violation of constraints.

It takes several QP iterations for the NMPC framework to find a solution. We can set up the maximum number of iterations $N_{\text{iter}}^{\max}$ while the optimization can still terminate in advance once the desired KKT value is reached. The worst

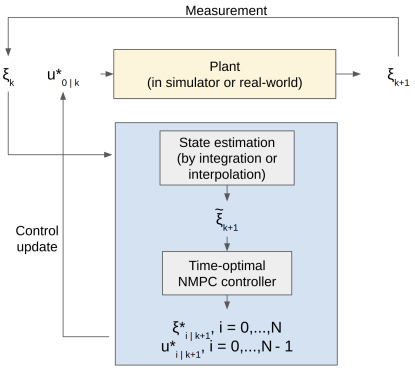


Fig. 1. The conventional NMPC control framework.

case execution time (WCET) of one QP iteration is an affine function of the horizon length: $\text{WCET}_{\text{per iter}}(N)$.

B. Conventional control framework

In conventional NMPC, the calculation is supposed to be done between steps discretized in terms of progress. The result of NMPC is released at the beginning of each step. At progress step k, the state series in the prediction horizon is noted as $\xi_{i|k}, i = 0, \ldots, N$ while the piece-wise constant control series is $u_{i|k}, i = 0, \ldots, N-1$. $\xi^*_{i|k}$ and $u^*_{i|k}$ represent the optimal result. In conventional NMPC, only the optimal control of the first predicted step $u^*_{0|k}$ is deployed, and the rest of the optimal control series is recalculated at progress step $k+1$.

As shown in Fig. 1, the system evolution and the NMPC calculation occur simultaneously. In the system evolution part, the vehicle moves forward using the optimal control input $u^*_{0|k}$. The system finally evolves to a new state ξ_{k+1} during the progress time $\Delta t_k^{\text{progress}} = t_{k+1} - t_k$. In the calculation part, we prepare the optimal control for the next evolution step. First, we measure the current state ξ_k and predict the state $\tilde{\xi}_{k+1}$ using the integration of system dynamics or a simple interpolation. Then the NMPC is calculated using time Δt_k^{calc}.

III. Motivation and Solution Design Objective

In this section, we first discuss some issues in the online implementation of NMPC-based controllers under the conventional control framework. This provides research motivations for designing a new framework to deploy NMPC-based controllers within a limited time budget. In the second part of this section, we describe the research problem and define a design objective for the intended solution.

A. Conventional NMPC issues in single-vehicle racing mode

Using the conventional control framework, the time budget requirement, $\Delta t_k^{\text{progress}} \geq \Delta t_k^{\text{calc}}$, is not always respected, especially in long-horizon NMPC. The computed trajectory may be obsolete taking into account the progress of the vehicle during the calculation.

We demonstrate this problem using the same experimental setting as in [5] running on a standard laptop with an Intel i7 processor (in this work, the code optimization level is set to be '-O3'). The experiment is performed in an offline way: we get the vehicle state at step k, then we calculate the optimal control series with NMPC; once the calculation finishes, we simulate the system state at step $k+1$ by an integration method using the control result of NMPC. Both a short-horizon length of $N = 15$ and a long-horizon length of $N = 30$ are tested. The two testing tracks studied in [5] take respectively 145 and 300 progress steps to run through.

TABLE I
Statistics for conventional NMPC calculation

		Track 1		Track 2	
horizon N		15	30	15	30
Per-step progress time [ms]	max	57.8	54.5	73.7	66.5
	min	17.2	17.3	12.8	12.2
	mean	33.2	32.7	33.9	33.4
Per-step calculation time [ms]	max	19.7	81.6	19.2	101.3
	min	2.4	8.1	1.2	3.8
	mean	6.8	27.0	6.4	25.2
QP iteration number	max	15	11	14	18
	min	2	2	1	1
	mean	4.4	4.3	3.9	4.0
Lap time [s]		4.852	4.773	10.189	10.064

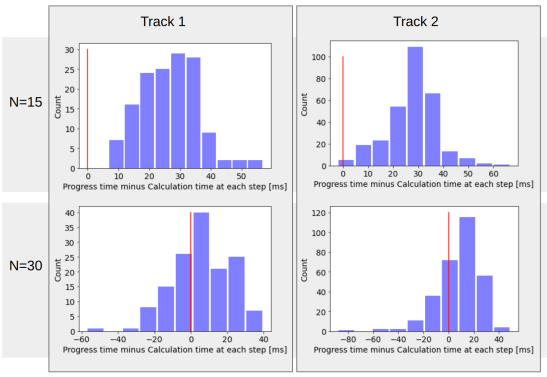


Fig. 2. The histograms for the value of $\Delta t = \Delta t_k^{\text{progress}} - \Delta t_k^{\text{calc}}$.

From Table I, we can observe that the average calculation time for $N = 30$ is about 4 to 5 times more than that for $N = 15$. To examine the constraint $\Delta t_k^{\text{progress}} > \Delta t_k^{\text{calc}}$ for each step, we display the value of $\Delta t = \Delta t_k^{\text{progress}} - \Delta t_k^{\text{calc}}$ in Fig. 2. The value of Δt should always be positive to allow online execution. We see that on track 1, the constraint is totally satisfied for $N = 15$ while this is not the case for certain steps with $N = 30$. On track 2, even with $N = 15$, there are 2 steps out of 300 where the calculation time exceeds the progress time by $2.2[ms]$ and $0.7[ms]$ respectively, while with $N = 30$ the constraint is violated at more steps. We find that longer horizon NMPC is computationally expensive. But its superior performance in terms of lap time (as shown in the last row in Table I) makes it attractive.

B. NMPC controller issues in head-to-head racing mode

The conventional control framework counts the control steps by measuring the progress s, e.g. the control step k is at the progress $s_k = s_0 + k \cdot \Delta s$. In this way, continuous monitoring for state s_k is required and related computation resources will be consumed. Since the monitoring is based on

vehicle self-localization, inaccurate measurement results are possible, particularly in complicated dynamic situations. The measurement inaccuracy of s_k may result in a delay in control release, which could potentially lead to a collision.

Another difficulty lies in the representation of constraints for avoiding collision between two vehicles. At each prediction step, the ego vehicle can be situated in 4 different positions relative to the opponent vehicle: left, right, behind, or ahead. The choice at each step can be formulated and solved as a Mixed Integer Problem (MIP). However, as reported in [5], combining NMPC and MIP results in a long execution time which is usually 2 to 4 times more, depending on the horizon length, than simple NMPC. As seen in Fig. 2, simple NMPC with $N = 15$ already encounters difficulty in finishing the calculation at certain steps on track 2. The additional complexity of MIP makes it more difficult to complete the calculation in time.

C. Problem description

In single-vehicle racing mode, a long-horizon NMPC is preferred. In section III-A, we described the problem to ensure the online execution of long-horizon NMPC. The proposed method should keep the optimality featured by long-horizon NMPC while also ensuring that the calculation time is shorter than the progress time. We should also pay attention to the QP iteration numbers of the SQP procedure of the proposed method, since it indicates the required effort to reach the convergence of NMPC.

In head-to-head racing mode, we make more effort to ensure that the vehicle can respond in a safe and flexible manner to the surroundings. In section III-B, we described the problem to design an online NMPC-based controller that is feasible (the calculation should be done within the given time budget), safe (no collision happens) and realistic (the ability for overtaking behavior should be enabled).

IV. Proposed solution

In this section, we formally propose a triggering-based NMPC recalculation method for single-vehicle racing mode and a realistic control strategy for head-to-head competition mode. Both approaches are intended to make it possible for NMPC-based controllers to function with constrained computational resources.

A. Single-vehicle racing mode

1) m-step recalculation: To ensure that we always have $\Delta t_k^{\text{progress}} > \Delta t_k^{\text{calc}}$, we propose to deploy m-steps of optimal control instead of the 1-step control in the conventional control framework. The progress time is therefore $\Delta t_k^{\text{progress}} = \Delta t_{k\to k+m}^{\text{progress}} = t_{k+m} - t_k, 1 \le m \le N-1$. We select a minimum step number m_1 to satisfy the time budget needed by the calculation. To save even more computational resources, we can reuse $m_2, m_2 \ge m_1$ steps in the NMPC prediction horizon until the recalculation is triggered by other factors. In the following part, we introduce a triggering condition that is related to system performance in terms of lap time. Other

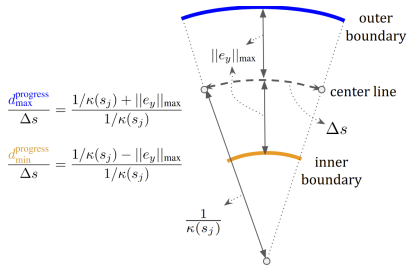


Fig. 3. The estimation for vehicle's traveling path.

factors such as the difference between the predicted state and the actual state at step $k+i, 1 \le i \le N-1$ can also be used as the triggering condition, which we will study in future work. In this work, we suppose that there is no external disturbance and model mismatch error that could cause this type of state difference.

2) Curvature change based triggering: In single-vehicle racing mode, human drivers have been observed to adjust their intended course only when they are aware that they are about to enter a track segment that is different from the one they had originally planned for (e.g. from a straight line to a corner, or inverse). In our design, the curvature of the track's center line $\kappa(s)$ is available, which depends on the progress s along the center line and indicates the track's segment type. Intuitively, we scan the curvature change to decide the triggering point for recalculation.

At step k, we check the recalculation step $k+i, i = 1, \dots, N-1$ which will be triggered by curvature changing. The end of the potential prediction horizon will be at step $k+i+N$. If there is a significant difference between the track curvatures at step $k+i+N$ and step $k+N$ (which is the end of the current prediction horizon), the recalculation of NMPC might provide us with a different optimal trajectory; otherwise, the resulting trajectory should not be significantly different from the previous result. Meanwhile, as mentioned in the last section, we should make sure that the progress time between the current step k and step $k+i$ is long enough to finish the recalculation in time budget C. The condition is formally expressed as finding the first $i = m, i \in [1, \dots, N-1]$ such that:

$$\begin{aligned} &||\kappa(s_{k+i+N}) - \kappa(s_{k+N})|| \ge \epsilon_\kappa \\ &\Delta t_{k\to k+i}^{\text{progress}} = t_{k+i} - t_k \ge C \end{aligned} \tag{2}$$

The first term is meant to improve the system performance in terms of lap time since it indicates the potential change of the optimal trajectory, while the second term is a necessary condition for the time budget to be respected.

3) Approximation for the triggering condition: It is logically difficult to calculate the second term in (2) since the progress time at step $k+i$, t_{k+i}, is an unknown future state variable. However, we can make a conservative estimation of the vehicle's travel path: assuming that it travels along the track's inner boundary, the vehicle will take the shortest route which results in the shortest progress time. As shown in Fig.

3, we use a simple geometric relation to estimate the progress time between steps j and $j+1$:

$$\Delta t_{\text{j, min}} = d_{\text{min}}^{\text{progress}} / v_{\text{max}}^{\text{ego}} = \Delta s \cdot (1 - ||e_y||_{\text{max}} \cdot \kappa(s_j)) / v_{\text{max}}^{\text{ego}} \tag{3}$$

The triggering condition is therefore reformulated as finding the first $i = m, i \in [1, \dots, N-1]$ such that:

$$\begin{aligned} ||\kappa(s_{k+i+N}) - \kappa(s_{k+N})|| &\geq \epsilon_\kappa \\ \Delta t_{k \to k+i}^{\text{progress}} \geq \sum\nolimits_{j=1}^{i} \Delta t_{\text{j, min}} &\geq C \end{aligned} \tag{4}$$

4) Initialization related issues: Skipping recalculation points makes it harder to come up with an initialization strategy. In conventional NMPC, we have the optimal solution at progress step $k-1$, i.e. $\xi_{i|k-1}^*, u_{i|k-1}^*$, but only the control of the first step $u_{0|k-1}^*$ in the prediction horizon will be applied. We recalculate the NMPC at progress step k. Its state and control in the first $N-1$ prediction steps can be initialized by the previous result (it is called "state/control shifting", which is a key component of the RTI method):

$$\begin{aligned} \xi_{i|k} &= \xi_{i+1|k-1}^*, i = 0, \dots, N-1 \\ u_{i|k} &= u_{i+1|k-1}^*, i = 0, \dots, N-2 \end{aligned} \tag{5}$$

The state/control values at the last prediction step can be a direct copy from the second last prediction step: $\xi_{N|k} = \xi_{N-1|k}$ and $u_{N-1|k} = u_{N-2|k}$. Another option is to only copy the control at the last prediction step $u_{N-1|k} = u_{N-2|k}$ and estimate the last step state by forward simulation: $\xi_{N|k} = f_{\text{Runge–Kutta}}^{\text{integration}}(\xi_{N-1|k}, u_{N-1|k})$.

With our method, the state/control shifting is only feasible for the first $N-m$ steps. For the remaining m steps, we currently use an all-zero strategy except for several exceptions, e.g. the velocity is set to be a constant non-zero value inside its value range. Although no initialization-related errors appeared in the following experiments, this naive initialization method can take more QP iterations to converge than other approaches. In the worst case, it might not even ensure convergence. We discuss in detail this influence of initialization on QP iteration numbers in section V. A smarter initialization method should be studied in further research.

B. Head-to-head racing mode

1) Fixed control period: As discussed in section III-B, continuous monitoring of the progress s costs additional computation resources. We propose to release the calculation of optimal control in head-to-head racing mode at a fixed frequency. The associated timing function is relatively accurate and involves very little calculation. The conventional control framework discussed in section II-B is therefore modified as follows. The system evolution is triggered by time period T^{ctrl} instead of progress length Δs. The NMPC is calculated in the same manner as in the conventional framework and we note the optimal control series for time instant t as $u_{i|t}^*, i = 0, \dots, N-1$. The control input will be updated as $u^*(t \to t + T^{\text{ctrl}}) = [u_{i|t}^*], 1 \leq i \leq N_T$ where $N_T \leq N-1$ & $t_{N_T-1}^* - t \leq T^{\text{ctrl}}$ & $t_{N_T}^* - t > T^{\text{ctrl}}$.

2) Constraints on control period: With the appearance of the opponent vehicle, we should take into account the dynamic evolution of both vehicles. In other words, the constraints are based on the opponent vehicle's expected trajectory, which might become invalid as time passes and might lead to functional incorrectness. For safety reasons, we enforce a safety distance d_{min} that allows the ego vehicle to have sufficient braking space and time to react. In an extreme case, the ego vehicle follows closely to the opponent with the maximum speed: the braking distance is ${v_{\text{max}}^{\text{ego}}}^2 / 2a_{\text{max}}$. Before receiving the braking command from the control update, the ego vehicle continues to move forward. The safety distance between the front of the ego vehicle and the rear of the opponent is thus defined as $d_{\text{min}} = {v_{\text{max}}^{\text{ego}}}^2 / 2a_{\text{max}} + v_{\text{max}}^{\text{ego}} \cdot T_{\text{max}}^{\text{ctrl}}$. We shouxld thus update the control signal in a period shorter than $T_{\text{max}}^{\text{ctrl}} = (d_{\text{min}} - {v_{\text{max}}^{\text{ego}}}^2 / 2a_{\text{max}}) / v_{\text{max}}^{\text{ego}}$ to prevent a collision.

In addition to meeting the above requirement of functional correctness, we should also make sure that the controller has sufficient time to finish the calculation even in the worst case. The minimum control period is estimated as the same with the time budget: $T_{\text{min}}^{\text{ctrl}} = C$, while $C = \text{WCET}_{\text{per iter}}(N) \cdot N_{\text{iter}}^{\text{max}}$. Finally we have the constraint: $T_{\text{min}}^{\text{ctrl}} \leq T^{\text{ctrl}} \leq T_{\text{max}}^{\text{ctrl}}$, i.e. $\text{WCET}_{\text{per iter}}(N) \cdot N_{\text{iter}}^{\text{max}} \leq T^{\text{ctrl}} \leq (d_{\text{min}} - {v_{\text{max}}^{\text{ego}}}^2 / 2a_{\text{max}}) / v_{\text{max}}^{\text{ego}}$.

3) Short-horizon NMPC: To make the ego vehicle react quickly enough to the environment, we propose to use short-horizon NMPC, which reduces $\text{WCET}_{\text{per iter}}(N)$ by shortening N. As an example shown in Table I, long-horizon ($N = 30$) has only relatively small advantage compared to short-horizon ($N = 15$) in terms of lap time: 1.6% and 1.2% better on the two tracks. In head-to-head mode, extra progress time to react to the dynamic surrounding (for example emergency braking for avoiding potential collision with the opponent vehicle) is much more than this lap time difference. Therefore, using short-horizon NMPC is rather acceptable in this mode.

4) Simplified overtaking strategy: In both the real-world racing scenarii and numerical simulations using NMPC and MIP, we can observe that overtaking usually occurs in successive steps and only on one single side (either left or right). We can thus simplify the control strategy as one-side overtaking, which allows us to use only NMPC and therefore reduce computation time.

The general idea of our control strategy design is that, if two vehicles are far away from each other, we use a simple NMPC same as the one in the single-vehicle racing mode for its forward (FW) advancement. Otherwise, in the prediction horizon of the ego vehicle, we find the first step that might overlap with the opponent. Then we set up the constraints for the next M steps ($M \leq N$) for left and right side overtaking (LO and RO), and solve them in parallel. To ensure that this simplified overtaking attempt is safe, we propose the following technique in the situation where both sides overtaking is impossible: if two vehicles still keep a safe distance, we let the ego vehicle run at the same speed as the opponent vehicle and continuously steer towards it, i.e. follow (FO) the opponent vehicle; otherwise the ego vehicle brakes. More details are given in Algo. 1. D_1 represents the distance threshold to

trigger the overtaking strategy; D_2 is the safe distance in terms of e_y.

Algorithm 1 Control strategy in head-to-head racing

Input: the position and orientation of both vehicles.
Output: the control command for the ego vehicle.

prepare an NMPC problem formulation FW by ignoring the opponent vehicle
if not ($s_0^{\text{ego}} \leq s_0^{\text{opp}}$ and $s_0^{\text{opp}} - s_0^{\text{ego}} \leq D_1$) **then**
  solve NMPC FW problem
else
  find the first prediction step $i, 0 \leq i \leq N$ which satisfies: dist$(s_i^{\text{ego}} - s_i^{\text{opp}}) \leq d_{\text{min}}$
  set up LO: ${e_y}_i^{\text{opp}} + D_2 \leq {e_y}_j^{\text{ego}} \leq {e_y}^{\text{max}}, j = i, \ldots, i+M$
  set up RO: ${e_y}^{\text{min}} \leq {e_y}_j^{\text{ego}} \leq {e_y}_i^{\text{opp}} - D_2, j = i, \ldots, i+M$
  solve LO and RO problems in parallel
  if one of these converges to a feasible solution **then**
    select the one with faster progress time
  else
    if dist$(s_0^{\text{ego}} - s_0^{\text{opp}}) \leq d_{\text{min}}$ **then**
      brake by letting: $v^{\text{ego}} = 0, \delta^{\text{ego}} = 0$
    else
      calculate $\Delta\psi$ difference between the ego's yaw and the angle towards the opponent vehicle
      suppose that the vehicle's path follows the kinematic model: $\delta^{\text{ego}} = \arctan(\Delta\psi / T^{\text{ctrl}} \cdot l^{\text{vehicle}} / v^{\text{ego}})$
      $v^{\text{ego}} = v^{\text{opp}}$
      apply $v^{\text{ego}}, \delta^{\text{ego}}$ to perform FO strategy
    end if
  end if
end if

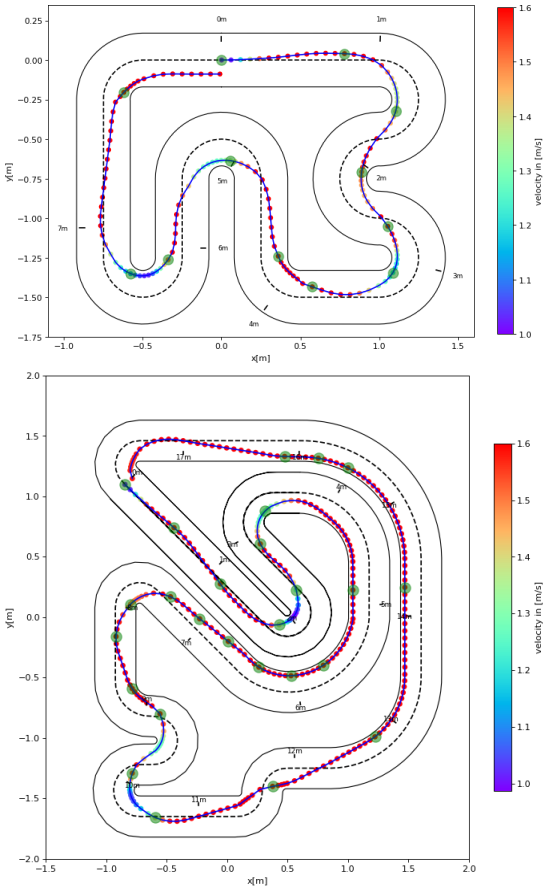


Fig. 4. Trajectories using our triggering mechanism (track 1 and 2). Green dots represent progress points triggering NMPC recalculation events; red points conventional NMPC recalculation events. Colorbars represent speed.

V. Simulation Results

In this section, we present simulation results for both racing modes. For single-vehicle racing mode, the vehicle dynamics and identification parameters are the same as those in [5]. In the experiment for head-to-head racing, we use a more realistic simulator that allows two cars to interact with one another. The vehicle model chosen in the simulator has a real-world counterpart, making it convenient to apply the same algorithm to an actual race car in future work.

A. Single-vehicle racing mode

1) Experimental setup: Using the proposed triggering condition (4) and the same experimental setting as in [5] with the code optimization level set to '-O3', we run NMPC with prediction horizon $N = 30$ on both testing tracks. The experiment is conducted on a standard laptop featuring an Intel i7 processor and 32 GB of RAM under Ubuntu 18.04.

A small value of ϵ_κ in the triggering condition (4) makes the triggering condition more sensitive, thus, more calculations will be performed. A large value of ϵ_κ makes it less sensitive and we might miss the recalculation points that lead to the global optimal trajectory. We select $\epsilon_\kappa = 10\% \cdot (\kappa^{\max} - \kappa^{\min})$ as a compromise according to the general curvature condition on both tracks. Another important parameter to be determined is the time budget C. According to statistics in Table I, we find that the average calculation time of one QP iteration is about $6.3[ms]$ and the total QP iteration number per step is usually smaller than 20. In the following experiment, we bound the QP iteration number below 20, and set the KKT value to 1×10^{-4}. This KKT value indicates the precision error. In terms of vehicle position, it means that the maximum precision error is no more than $0.01[cm]$, which is relatively small and acceptable given that the vehicle size is about $6[cm]$. We also give a 20% margin to estimate WCET as time budget: $C = (1 + 20\%) \cdot \Delta t^{\text{max observed time}} = (1 + 20\%) \cdot (20 \cdot 6.3[ms]) \sim 150[ms]$.

The resulting trajectory is shown in Fig. 4. We find that: on track 1 (resp. track 2), NMPC is triggered 12 (resp. 26) times for recalculation, while it is recalculated 145 (resp. 300) times in conventional NMPC.

2) Respect of the time budget: We first verify whether the second term in (4) is satisfied for all recalculation steps, i.e. whether the progress time is always long enough to allow the NMPC recalculation to be completed.

Fig. 5 gives an intuitive comparison of progress time and calculation time at each step. Table III shows detailed statisti-

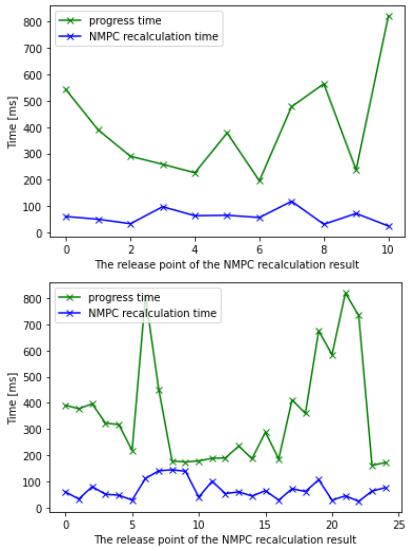


Fig. 5. Progress time and NMPC recalculation time. (track 1 - track 2)

TABLE II
LAP TIME [S] FOR DIFFERENT METHODS ON BOTH TRACKS

		Track 1	Track 2
standard NMPC	N = 15	4.852	10.189
	N = 30	4.773	10.064
triggering method	N = 30	4.775	10.075

TABLE III
STATISTICS FOR NMPC WITH TRIGGERING METHOD ($N = 30$)

		Track 1	Track 2
Per-step progress time [ms]	max	820.0	819.2
	min	196.5	161.8
	mean	398.2	360.4
Per-step calculation time [ms]	max	118.8	144.6
	min	25.4	24.3
	mean	62.0	68.5
QP iteration number	max	20	20
	min	6	4
	mean	10.2	10.4

cal information. The minimum progress time on track 1 (resp. track 2) is $196.5[ms]$ (resp. $161.8[ms]$), which is longer than $C = 150[ms]$. The maximum calculation time is $118.8[ms]$ (resp. $144.6[ms]$), which hasn't exceeded C. As a result, the computation timing restriction is respected.

3) Lap time: We also check the performance in terms of lap time. By comparing the lap times between different methods in Table II, we find that the proposed method has a slightly worse lap time than the conventional NMPC method with $N = 30$ but is still better than the one with $N = 15$. It shows that the first term of the triggering condition (4) relaunches the NMPC efficiently, i.e. in a sporadic manner without deviating too much from the global optimal trajectory.

4) Initialization related issue: We notice that in Table III, several NMPC recalculations reach the maximum iteration number. To be precise, on track 1 (resp. track 2), 1 out of 12 (resp. 3 out of 26) recalculation points reaches the maximum iteration number, finishing with the KKT value

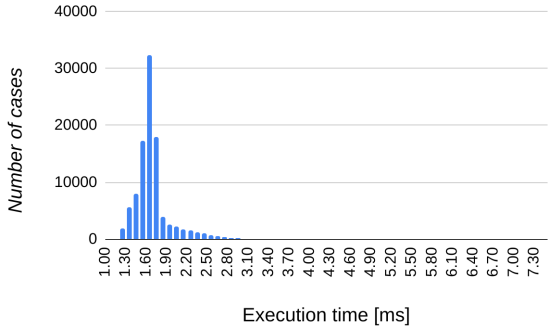


Fig. 6. Histogram of "NMPC FW" per-iteration execution time.

1.2×10^{-4} (resp. 2.3×10^{-4}). As discussed in section V-A1, the maximum precision error is limited to $0.012[cm]$ (resp. $0.023[cm]$), which is relatively small and acceptable.

If we compare Tables III with I, we find average QP iteration numbers of the proposed triggering methods are almost twice as high as in the conventional NMPC scenario, resulting in a per-step calculation time about twice as long. This can be explained by the fact that the conventional NMPC benefits from a better initialization method, state/control shifting, as discussed in section IV-A4. With our naive initialization, the proposed triggering method needs more QP iterations to make the NMPC algorithm converge. To further reduce the QP iterations number and thus the total calculation time, we intend to design a better initialization method in future work.

In conclusion, this experiment demonstrates the effectiveness of the proposed method: long-horizon NMPC has a sufficient time budget to complete the computation; the optimality is preserved even though we skipped many recalculation steps.

B. Head-to-head racing mode

1) Experimental setup: A ROS node [8] runs on the NVIDIA Jetson TX2, featuring two CPUs and one GPU, as the ego vehicle's controller. The onboard operating system is a tailed version of Ubuntu (JetPack SDK 4.6 + Linux 4 Tegra 32.6.1) with PREEMPT_RT patch. The CPU frequency is set to the maximum value: 2.0 GHz. We set the code optimization option to '-O3' for compilation.

We perform the experiment with a Hardware-in-the-Loop configuration: a simulator named f1tenth_gym_ros [9] runs on the laptop and interacts with the controller running on the Jetson TX2 via a USB cable. The simulator sends the odometry information to the controller which sends back the vehicle speed/steering angle as the command input. The ping test indicates that the highest latency is less than $1[ms]$, which meets our experimental requirements. The track used in the simulator has the same shape as in [2], but is 10 times larger to accommodate the size of our experiment vehicle.

2) WCET Measurements: According to results from several preliminary experiments on the testing track, we find appropriate values for the horizon length and maximum iteration number: $N = 10$, $N_{\text{iter}}^{\max} = 10$. We use both values as parameters to test the NMPC controllers on 100000 random sampling

TABLE IV
CONTROLLER EXECUTION TIME ON JETSON TX2 ([MS])

	NMPC			FO
	LO per iter	RO per iter	FW per iter	
max	7.333	7.380	7.322	0.027
99%*	2.928	2.929	2.822	0.007
mean	1.710	1.694	1.688	0.002

*99% means the value at the 99th percentile in the distribution

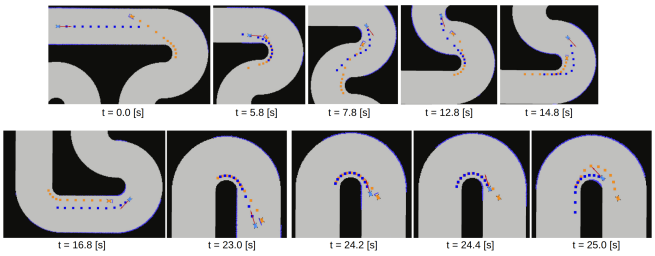


Fig. 7. Typical scenario in head-to-head competition using our control strategy. Ego vehicle in blue; opponent vehicle in orange. Dotted lines are planned optimal trajectories.

cases and obtain a WCET distribution as in Fig. 6. Detailed statistics about the execution time of the different controller components are provided in Table IV. We use an optimistic estimation of the WCET: $\text{WCET}_{\text{per iter}}(10) \approx 3.0[ms]$.

3) NMPC Parameters: We recall the criteria discussed in section IV-B for choosing control period: $\text{WCET}_{\text{per iter}}(N) \cdot N_{\text{iter}}^{\max} = C \leq T^{\text{ctrl}} \leq (d_{\min} - v_{\max}^{\text{ego}\,2}/2a_{\max})/v_{\max}^{\text{ego}}$. The safety distance between the two vehicles' edges is set to be the same as the vehicle's length: $d_{\min} = 0.58[m]$. The maximum velocity is $v_{\max}^{\text{ego}} = 2[m/s]$ and the maximum braking acceleration is $a_{\max} = 8[m/s^2]$. The upper bound for the control period is calculated as $165[ms]$. We have the lower bound for the control period, i.e. the time budget: $C = \text{WCET}_{\text{per iter}}(10) \cdot 10 = 30[ms]$. To react to the environment as quickly as possible, we set the control period equal to this lower bound: $T^{\text{ctrl}} = C = 30[ms]$.

4) Overtaking Behavior: Using the predetermined value of T^{ctrl}, we run the proposed control strategy (Algo. 1) in the simulator. A typical scenario is shown in Fig. 7. At the beginning, the ego vehicle is behind the opponent: $s_0^{\text{ego}} = 0.0[m], e_{y0}^{\text{ego}} = 0.0[m], e_{\psi 0}^{\text{ego}} = 0.0[rad]$ and $s_0^{\text{opp}} = 7.0[m], e_{y0}^{\text{opp}} = 1.0[m], e_{\psi 0}^{\text{opp}} = -0.5[rad]$. At $t = 5.8[s]$, neither LO nor RO strategies give a feasible solution, but the distance between the two vehicles is still safe. The ego vehicle then follows the opponent vehicle: as seen in the second frame of Fig. 7, the previously planned optimal trajectory (blue dotted line) is no longer employed because it is invalid at this time and the ego vehicle uses FO strategy instead. At time $t = 7.8[s]$, only LO strategy is feasible (no space for RO). At $t = 12.8[s]$, both LO and RO strategies are feasible, the ego vehicle decides to attempt RO at this step since the predicted progress time is shorter than the one on the left side. At $t = 14.8, 16.8, 23.0[s]$, the ego vehicle uses LO strategy and succeeds at $t = 24.2[s]$. From now on, the ego vehicle becomes the leader, it is thus the opponent vehicle's responsibility to avoid collision. Since two vehicles are close enough to each other, the emergency braking of the opponent vehicle is triggered. The opponent vehicle's position remains nearly unchanged between $t = 24.2[s]$ and $24.4[s]$, proving that it succeeds to brake.

In summary, as seen in the simulation: the safety (collision-free) of the system is ensured by the emergency braking mechanism (lines 12-13 in Algo. 1); the execution time is guaranteed to be under the given time budget based on the WCET estimation; the overtaking is enabled, and by taking into consideration the possibility of both left and right side overtaking, we get a relatively optimal control strategy.

VI. CONCLUSION

To allow the NMPC-based controller to operate within a limited time budget, we proposed two distinct methods. In single-vehicle racing mode, we defined a triggering-based method to implement a long-horizon NMPC on board. The triggering mechanism enables the long-horizon NMPC to have a sufficient time budget for completing the computation. The optimality is preserved even though we skipped recalculation steps. In head-to-head racing mode, frequent control updates should be maintained to respond to the changing surroundings. We came up with a fixed-period short-horizon NMPC approach that enables overtaking behavior while ensuring the safety of the vehicle.

To handle more realistic scenarii in single-vehicle racing mode, we shall take into account the environment disturbance and the vehicle model mismatch error. We also consider implementing our control strategy on a real-world race car.

REFERENCES

[1] J. Betz, H. Zheng, A. Liniger, U. Rosolia, P. Karle, M. Behl, V. Krovi, and R. Mangharam, "Autonomous vehicles on the edge: A survey on autonomous vehicle racing," *IEEE Open Journal of Intelligent Transportation Systems*, 2022.

[2] R. Verschueren, M. Zanon, R. Quirynen, and M. Diehl, "Time-optimal race car driving using an online exact hessian based nonlinear mpc algorithm," *ECC 2016*, pp. 141–147, 2016.

[3] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM Journal on control and optimization*, vol. 43, no. 5, pp. 1714–1736, 2005.

[4] Z. Sun, L. Dai, K. Liu, D. V. Dimarogonas, and Y. Xia, "Robust self-triggered mpc with adaptive prediction horizon for perturbed nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 11, pp. 4780–4787, 2019.

[5] N. Li, E. Goubault, L. Pautet, and S. Putot, "Autonomous racecar control in head-to-head competition using mixed-integer quadratic programming," in *Opportunities and Challenges with Autonomous Racing, ICRA*, 2021.

[6] J. Bhargav, J. Betz, H. Zehng, and R. Mangharam, "Deriving spatial policies for overtaking maneuvers with autonomous vehicles," in *International Conference on COMmunication Systems & NETworkS (COMSNETS)*. IEEE, 2022, pp. 859–864.

[7] S. M. Safdarnejad, J. D. Hedengren, N. R. Lewis, and E. L. Haseltine, "Initialization strategies for optimization of dynamic systems," *Computers & Chemical Engineering*, vol. 78, pp. 39–50, 2015.

[8] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[9] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam, "F1tenth: An open-source evaluation environment for continuous control and reinforcement learning," in *NeurIPS 2019 Competition and Demonstration Track*. PMLR, 2020, pp. 77–89.