Autonomous Racing with Multiple Vehicles using a Parallelized Optimization with Safety Guarantee using Control Barrier Functions

Suiyi He\textsuperscript{1,*}, Jun Zeng\textsuperscript{2,*}, and Koushil Sreenath\textsuperscript{2}

Abstract—This paper presents a novel planning and control strategy for competing with multiple vehicles in a car racing scenario. The proposed racing strategy switches between two modes. When there are no surrounding vehicles, a learning-based model predictive control (MPC) trajectory planner is used to guarantee that the ego vehicle achieves better lap timing performance. When the ego vehicle is competing with other surrounding vehicles to overtake, an optimization-based planner generates multiple dynamically-feasible trajectories through parallel computation. Each trajectory is optimized under a MPC formulation with different homotopic Bezier-curve reference paths lying laterally between surrounding vehicles. The time-optimal trajectory among these different homotopic trajectories is selected and a low-level MPC controller with control barrier function constraints for obstacle avoidance is used to guarantee the system’s safety-critical performance. The proposed algorithm has the capability to generate collision-free trajectories and track them while enhancing the lap timing performance with steady low computational complexity, outperforming existing approaches in both timing and performance for an autonomous racing environment. To demonstrate the performance of our racing strategy, we simulate with multiple randomly generated moving vehicles on the track and test the ego vehicle’s overtaking maneuvers.

I. INTRODUCTION

A. Motivation

Recently, autonomous racing is an active subtopic in the field of autonomous driving research. In autonomous racing, the ego car is required to drive along a specific track with an aggressive behavior, such that it is capable of competing with other agents on the same track. By overtaking other leading vehicles and moving ahead, the ego vehicle can finish the racing competition with a smaller lap time. While the behavior of overtaking other vehicles has been studied in autonomous driving on public roads, however, these techniques are not effective on a race track. This is because autonomous vehicles are guided by dedicated lanes on public roads to succeed in lane follow and lane change behaviors, while the racing vehicles compete in the limited-width tracks without guidance from well-defined lanes. Existing work focuses on a variety of algorithms for autonomous racing, but most of them could not provide a time-optimal behavior with high update frequency in the presence of other moving agents on the race track. In order to generate racing behaviors for the ego racing car, we propose a racing algorithm for planning and control that enables the ego vehicle to maintain time-optimal maneuvers in the absence of local vehicles, and fast overtake maneuvers when local vehicles exist.

B. Related Work

In recent years, researchers have been focusing on planning and control for autonomous driving on public roads. For competitive scenarios like autonomous lane change or lane merge, both model-based methods \cite{1} and learning-based methods \cite{2} have been demonstrated to generate the ego vehicle’s desired trajectory. Similarly, control using model-based methods \cite{3}–\cite{5} and learning-based methods \cite{6} has also been developed. However, the criteria to evaluate planning and control performance are different for car racing compared to autonomous driving on public roads. For autonomous racing \cite{7}, when the ego racing car competes with other surrounding vehicles, most on-road traffic rules are not effective. Instead of maneuvers that offer a smooth and safe ride, aggressive maneuvers that push the vehicle to its dynamics limit \cite{8} or even beyond its dynamics limit \cite{9} are sought to win the race. In order to quickly overtake...
surrounding vehicles, overtake maneuvers with tiny distances between the cars and large orientation changes are needed. Moreover, due to the bigger slip angle caused by changing the steering orientation more quickly during racing, more accurate dynamical models should be used for the design of planners and controllers for autonomous racing. We next enumerate the related work in several specific areas.

1) Planning Algorithms: For autonomous racing, the planner is desired to generate a time-optimal trajectory. Although some work using convex optimization problems [15], [22]–[27] or Bayesian optimization (BO) [28] reduces the ego vehicle’s lap time impressively, either no obstacles [15], [23]–[28] or only static obstacles [22] are assumed to be on the track. When moving vehicles exist on the track, nonlinear dynamic programming (NLP) [19], graph-search [12] and game theory [13], [14], [29] based approaches have demonstrated their capabilities to generate collision-free trajectories. Additionally, in order to improve the chance of overtaking, offline policies are learnt for the overtake maneuvers at different portions of a specific track [30]. However, these approaches don’t solve all challenges. For instance, work in [14], [19], [29], [30] does not take lap timing enhancement into account. In [12], the ego vehicle is assumed to compete on a straight track with one constant-speed surrounding vehicle. These assumptions are relatively simple for a real car racing competition. In [13], it is assumed that the planner knows the other vehicle’s strategy and the complexity of the planner increases excessively when multiple vehicles compete with each other on the track.

2) Control Algorithms: Researchers focus on enhancing performance of the ego vehicle by achieving its speed and steering limits through better control design, e.g., obtaining the optimal lap time by driving fast. The majority of existing work focuses on developing controllers with no other vehicles on the track. The learning-based controllers [16]–[18], [31] leverage the control input bounds to achieve optimal performance in iterative tasks. Model-free methods like Bayesian optimization (BO) [32], Gaussian processes (GPs) [10], deep neural networks (DNN) [33], [34] and deep reinforcement learning (DRL) [11], [35] have also been exploited to develop controllers that result in agile maneuvers for the ego car. To deal with other surrounding vehicles, DRL has also been used in [36] to control the ego vehicle during overtake maneuvers. Recently, model predictive based controllers (MPC) with nonlinear obstacle avoidance constraints have become popular to help the ego vehicle avoid other vehicles in the free space. A nonconvex nonlinear optimization based controller is implemented in [37] to help the ego vehicle avoid static obstacles. Researchers in [38] use mixed-integer quadratic programs (MIQP) to help the ego vehicle compete with one moving vehicle. In [20], GPs was applied to formulate the distance constraints of a stochastic MPC controller with a kinematic bicycle model. However, large slip angles under aggressive maneuvers will cause a mismatch between the real dynamics model and the kinematic model used in the controller, resulting in the controller being unable to guarantee the system’s safety in some cases. In [21], a safety-critical control design by using control barrier functions is proposed to generate a collision-free trajectory without a high-level planner, where infeasibility could arise due to the high nonlinearity of the optimization problem. Moreover, due to the lack of a trajectory planner, deadlock could happen very often during overtake maneuvers, such as in [20], [21]. A comparison of various approaches and their features are enumerated in TABLE I.

As mentioned above, all the previous work on planning and control design for autonomous racing could not enhance the lap timing performance and simultaneously compete with multiple vehicles. Inspired by the work on iterative learning-based control and optimization-based planning, we propose a novel racing strategy to resolve the challenges mentioned above with a steady low computational complexity.

C. Contribution

The contributions of this paper are as follows:

- We present an autonomous racing strategy that switches between a learning-based MPC trajectory planner (in the absence of surrounding vehicles) and an optimization-based homotopic trajectory planner with a low-level safety-critical controller (when the ego vehicle competes with surrounding vehicles).
- The learning-based MPC approach guarantees time-optimal performance in the absence of surrounding vehicles. When the ego vehicle competes with surrounding vehicles, multiple homotopic trajectories are optimized in parallel with different geometric reference paths and the best time-optimal trajectory is selected to be tracked with an optimization-based controller with obstacle avoidance constraints.
- We validate the robust performance together with steady low computational complexity of our racing strategy in numerical simulations where randomly moving vehicles are generated on a simulated race track. It is shown that our proposed strategy allows the ego vehicle to succeed in overtaking tasks without deadlock when there are multiple vehicles moving around the ego vehicle. We also demonstrate that our strategy would work for various racing environments.

II. BACKGROUND

In this section, we revisit the vehicle model and learning-based MPC for iterative tasks. The learning-based MPC will be used as the trajectory planner when no surrounding vehicles exist.

A. Vehicle Model

In this work, we use a dynamic bicycle model with decoupled Pacejka tire model under Frenet coordinates. The system dynamics is described as follows,

\[
\dot{x} = f(x, u),
\]
for better tracking performance. The dynamics (4) will be used for the racing controller design, where acceleration at vehicle’s center of gravity $a$ and steering angle $\delta$ are obtained at the local equilibrium point $\bar{x}$. The longitudinal velocity, lateral velocity and yaw rate, heading angle error between vehicle and the track’s center line is denoted by $v$, $y$ and $\psi$, respectively denoted by $v_x$, $v_y$ and $\omega_z$. In this paper, this model (1) is applied for precise numerical simulation using Euler discretization with sampling time 0.001s (1000Hz). Through linear regression from the simulated reference path, an affine time-invariant model

$$x_{t+1} = A(x) x_t + B(x) u_t$$

will be used in the trajectory planner to avoid excessive complexity from nonlinear optimization, where $\bar{x}$ represents the equilibrium point for the linearized dynamics. On the other hand, an affine time-varying model

$$x_{t+1} = A_t(\bar{x}k)x_t + B_t(\bar{x}k)u_t + C_t(\bar{x}k),$$

is obtained at the local equilibrium point $\bar{x}k$ on reference trajectory with iterative data which is close to $x_t$. This dynamics (4) will be used for the racing controller design for better tracking performance.

**Table I:** A comparison of recent work on autonomous racing and their attributes. Lap timing indicates if the lap timing performance is considered. Static and moving agent indicates if other static or moving agents are considered. Update frequency indicates the optimization update frequency. Learning-based approaches like GP don’t have this attribute. Planner indicates if the approach has the planning part. Dynamics accuracy indicates the dynamics model used in the controller, with “Yes”, “No”, “N/A” representing dynamic model, kinematic model and model-free.

<table>
<thead>
<tr>
<th>Approach</th>
<th>GP</th>
<th>DRL</th>
<th>Graph-Search</th>
<th>Game Theory</th>
<th>Model-Based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publication</td>
<td>[10]</td>
<td>[11]</td>
<td>[12]</td>
<td>[13]</td>
<td>[14]</td>
</tr>
<tr>
<td>Lap Timing</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Static Agent</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Moving Agent</td>
<td>No</td>
<td>No</td>
<td>One</td>
<td>One</td>
<td>Multiple</td>
</tr>
<tr>
<td>Update Frequency (Hz)</td>
<td>N/A</td>
<td>N/A</td>
<td>15</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>Planner</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Dynamics Accuracy</td>
<td>N/A</td>
<td>N/A</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

**Fig. 2:** Autonomous Racing Strategy: The system dynamics is identified through offline data collection via recursive tasks. For online deployment, when no surrounding vehicles exist, the learning-based MPC trajectory planner is executed to guarantee time-optimal trajectories. When there are surrounding vehicles, the best time-optimal trajectory is chosen among the $n+1$ trajectories that are optimized in parallel with each optimization carried out for a particular homotopic trajectory around the $n$ surrounding cars. The chosen trajectory is then tracked with a safety-critical model predictive based controller.

**B. Iterative Learning Control**

A learning-based MPC [16], which improves the ego vehicle’s lap timing performance through iterative tasks, will be used in this paper. This has the following components:

1) **Data Collection:** The learning-based MPC optimizes the lap timing through historical states and inputs from iterative tasks. To collect initial data, a simple tracking controller like PID or MPC can be used for the first several laps. During the data collection process, after the j-th iteration (lap), the controller will store the ego vehicle’s closed-loop states and inputs as vectors. Meanwhile, through offline calculation, every point of this iteration will be associated with a cost, which describes the time to finish the lap from this point.

2) **Online Optimization:** After the initial laps, the learning-based MPC optimizes the vehicle’s behavior based on collected data. At each time step, the terminal constraint is formulated as a convex set. This convex set includes the states that can drive the ego vehicle to the finish line in the previous laps. By constructing the cost function to create a minimum-time problem, an open-loop optimized trajectory can be generated. Since the cost function is based on the previous states’ timing data, the vehicle is able to drive to the finish line with time that is no greater than the time from the same position during previous laps. As a result, the ego vehicle will reach the time-optimal performance after several laps.

More details of this method can be found in [16]. In our work, this approach will be used for trajectory planning when the ego vehicle has no surrounding vehicles. This helps with
better lap timing in the absence of surrounding vehicles. Notice that the data for iterative learning control will be collected through offline simulation with no obstacles on the track, as shown in Fig. 2

III. RACING ALGORITHM

Having introduced the background on vehicle modeling and learning-based MPC, we will next present an autonomous racing strategy that can help the ego vehicle enhance lap timing performance while overtaking other moving vehicles.

A. Autonomous Racing Strategy

There are two tasks in autonomous racing: enhancing the lap timing performance and competing with other vehicles. To deal with these two tasks, our proposed strategy will switch between two different planning strategies. When there are no surrounding vehicles, trajectory planning with learning-based MPC is used to enhance the timing performance through historical data. When there are surrounding vehicles, an optimization-based trajectory planner optimizes several homotopic trajectories in parallel around the surrounding vehicles and a collision-free time-optimal trajectory is selected among these, which is then tracked by a low-level MPC controller. By adding obstacle avoidance constraints to the low-level controller, it has the ability to guarantee the system’s safety. The racing strategy is summarized in Fig. 2. More details can be found in the full version [40].

B. Overtaking Planner

To determine if a surrounding vehicle is in the ego vehicle’s range of overtaking, the following condition must be satisfied:

$$-\epsilon l \leq s_{c,i} - s_c \leq \epsilon l + \gamma |v_x - v_{x,i}|$$

where $s_c$ and $s_{c,i}$ are ego vehicle’s and $i$-th surrounding vehicle’s current traveling distance, $v_x$ and $v_{x,i}$ are ego vehicle’s and $i$-th surrounding vehicle’s longitudinal speed. $l$ indicates the vehicle’s length. $\epsilon$ and $\gamma$ are safety-margin factor and prediction factor which we can tune for different performance.

As shown in Fig. 3 when there are $n$ vehicles in the ego vehicle’s range of overtaking, there exists $(n+1)$ potential areas with each area leading to paths with a different homotopy that the ego vehicle can use to overtake these surrounding vehicles. These $n+1$ areas are the one below the $n$-th vehicle, the one above the 1st vehicle, and the ones between each group of adjacent vehicles. We then solve $n+1$ groups of optimization-based trajectory planning problems in parallel, enabling steady low computational complexity even when competing with different numbers of surrounding vehicles. To reduce each optimization problem’s computational complexity, geometric paths with a distinct homotopy class that laterally lay between vehicles or vehicle and track boundary (black dashed curves in Fig. 3) are used as reference paths in the optimization problems. By comparing the optimization problems’ costs, the optimal trajectory is selected from the $n+1$ optimized solutions. For example, as the case shown in Fig. 3 the dashed orange line in area 2 will be selected since it avoids surrounding vehicles and finishes the overtake maneuver with a smaller time. The function to minimize during the selection is shown as follows,

$$J_s(x_t) = \min_{x_t} -K_s(s_{c,t+n} - s_{c,t}) - \sum_{k=1}^{N_p} ((s_{c,t+k} - s_{c,t+k+1})^2 + (e_y + e_{y,t+k+1})^2 - \gamma^2 + \beta^2) + b$$

where $K_s$ is a scalar used in metric for timing and $b$ is a non-zero penalty cost if the new potential area of overtaking is different from the area of overtaking in the last time step. A bigger value of $K_s$ is applied such that the ego vehicle is optimized to reach a farther point during the overtake maneuver, which results in a shorter overtaking time since the planner’s prediction horizon and sampling time are fixed. Additionally, the other terms in (6) prevents the ego vehicle from changing direction abruptly during an overtake maneuver and guarantees the ego vehicle’s safety.

Bezier-curves are widely used in path planning algorithms in autonomous driving research [41]–[43] because it is easy to tune and formulate. Third-order Bezier-curves are used in this work. Each Bezier-curve is interpolated from four control points, including shared start and end points with two additional intermediate points, shown in Fig. 3. Specifically, the start point for the Bezier curve is the ego vehicle’s current position and end point is on the time-optimal trajectory generated from learning-based MPC planner. The selection of the end point makes the vehicle’s state as close as possible to the time-optimal trajectory after the overtake behavior. To make all curves smoother and have no or fewer conflicts with surrounding vehicles, the other two control points will be between the track’s boundary and the vehicle for Areas 1 and $n+1$, or between two adjacent vehicles as shown in Fig. 3. These two intermediate control points will have the same lateral deviation from the center line. The key advantage of our selection of control points is that the interpolated geometric curve won’t cross the connected lines.
between control points with its convexity, as shown in Fig. [3]. This property makes our reference paths collision-free with respect to the surrounding vehicles in most cases, which speeds up the computational time of the trajectory generation in each area.

The details of the optimization formulation for trajectory generation will be illustrated in the next section.

### C. Trajectory Generation

After illustrating the planning strategy, this subsection will present details about the optimization problem used for trajectory generation for each potential area with different homotopic paths that the ego vehicle can use to overtake the surrounding vehicles.

The optimization problem is formulated as follows,

$$\arg\min_{x_{t+Np|t}, u_{t+Np|t-1}} p(x_{t+N|t}) + \sum_{k=0}^{N_p-1} q(x_{t+k|t})$$

s.t. $x_{t+k|t} = A x_{t+k-1|t} + B u_{t+k-1|t}, k = 0, ..., N_p-1,$

$$x_{t+k|t} \in \mathcal{X}, u_{t+k|t} \in \mathcal{U}, k = 0, ..., N_p-1,$$

$$g(x_{t+k|t}) \geq d + \epsilon, k = 0, ..., N_p-1,$$

where (7b), (7c), (7d) are constraints for system dynamics, state/input bounds and initial condition. The system dynamics constraint describes the affine linearized model described in [3]. The cost function (7a) is composed of three parts along the horizon of length $N_p$, the terminal cost $p(x_{t+N|t})$, the stage cost $q(x_{t+k|t})$ and the state/input changing rate cost $r(x_{t+k|t}, u_{t+k|t})$. The construction of cost function and constraints in the optimization will be presented in detail in the following subsections.

1) **Terminal Cost:** The terminal cost is related to the ego vehicle’s traveling distance along the track during the overtaking process, and is given by

$$p(x_{t+N|t}) = K_d(s_{t+N|t} - s_{c_e}).$$

This compares the open-loop predicted traveling distance at the $N$-th step $s_{t+N|t}$ with the ego vehicle’s current traveling distance $s_{c_e}$. This works as the cost metric for timing during the overtaking process.

2) **Stage Cost:** The stage cost introduces the lateral position difference between the open-loop predicted trajectory and other two paths along the horizon,

$$q(x_{t+k|t}) = ||x_{t+k|t} - x_R(s_{c_e})||_Q^2 + ||x_{t+k|t} - x_T(s_{c_e})||_Q^2,$$

where $x_R$ and $x_T$ are the reference path and time-optimal trajectories respectively in Frenet coordinates. The reference path $x_R$ is the Bezier-curve in the corresponding area. The time-optimal trajectory $x_T$ is generated by the learning-based MPC trajectory planner used on a track without other agents, as discussed in Sec. [2-E]. Note that $s_{c_e}$ is an initial guess for the traveling distance at the $k$-th step, which is equal to $s_{c_e} = s_{c_e} + v_{x_e}k \Delta t$, where a constant longitudinal speed is assumed along the prediction horizon.

3) **State/Input Changing Rate Cost:** To make the predicted trajectory smoother, the state/input changing rate cost $r(x_{t+k|t}, u_{t+k|t})$ is formulated as follows:

$$r(x_{t+k|t}, u_{t+k|t}, x_{t+k-1|t}, u_{t+k-1|t})$$

$$= ||x_{t+k|t} - x_{t+k-1|t}||_{R_1}^2 + ||u_{t+k|t} - u_{t+k-1|t}||_{R_2}^2$$

4) **Obstacle Avoidance Constraint:** In order to generate a collision-free trajectory, collision avoidance constraint (7e) is added in the optimization problem. To reduce computational complexity, only linear lateral position constraint will be added when the ego vehicle overlaps with other vehicles longitudinally. The inequality $|s_{c_i}(t) + v_{x_i}(t)k \Delta t - s_{c_e}(t + k)| < l + \epsilon$ will be used to check if the ego vehicle is overlapping with other vehicles longitudinally along the horizon. In (7e), $g(x) = |e_{y,i} - e_{y_e}|$ shows the lateral position difference, $l$ and $d$ are the vehicle’s length and width, and $\epsilon$ is a safe margin.

After parallel computation, the optimized trajectory $x_{t+N|t}^*$ with the minimum cost $J(x_t)$ discussed in [6] will be selected from among the $n + 1$ groups of optimization problems. It will be tracked by the MPC controller introduced in [II-D].

### D. Overtaking Controller

After introducing the algorithm for trajectory generation, a low-level tracking controller with model predictive control used for overtaking will be discussed in this part. The constrained optimization problem is described as follows:

$$\arg\min_{\ddot{u}_{t+Np-1|t}, \ddot{u}_{t+Np-1|t}} \sum_{k=0}^{N_p-1} \tilde{q}(\ddot{x}_{t+k|t}, \ddot{u}_{t+k|t}) + p_d(1 - \omega_k)^2$$

s.t. $\ddot{x}_{t+k+1|t} = A t+k|t| \ddot{x}_{t+k|t} + B t+k|t| \ddot{u}_{t+k|t}$

$$+ C t+k|t, k = 0, ..., N_c-1,$$

$$\ddot{x}_{t+k|t} \in \mathcal{X}, \ddot{u}_{t+k|t} \in \mathcal{U}, k = 0, ..., N_c-1,$$

$$\ddot{x}_{t+k|t} = \ddot{x}_t, \ddot{x}_{t+k|t} \geq \gamma x_{c_e} h(\ddot{x}_{t+k|t}), k = 0, ..., N_c-1,$$

where (11b), (11c), (11d) describe the constraints for system dynamics $\ddot{u}$, state/input bounds and initial conditions, respectively. The term $\tilde{q}(\ddot{x}_{t+k|t}, \ddot{u}_{t+k|t}) = ||\ddot{x}_{t+k|t} - x_{t+k|t}||_Q^2 + ||\ddot{u}_{t+k|t}||_Q^2$ represents the stage cost, which tracks the desired trajectory $x_{t+N|t}^*$ optimized by the trajectory planner from Sec. [II-C]. Equation (11c) with $0 \leq \gamma < 1$ represents the discrete-time control barrier function constraints [44] with relaxation ratio $\omega_k$, which could guarantee the system’s safety by guaranteeing $h(\ddot{x}_{t+k|t}) > 0$ along the horizon with forward invariance. In this project, $h(\ddot{x}_{t+k|t}) = (s_{c_e} - s_{c_i})^2 + (e_{y,i} - e_{y_e})^2 - l^2 - d^2$ is used to represent the distance between the ego vehicle and other vehicles. The optimization (11) allows us to find the optimal control $u^*_t = \ddot{u}_{t|t}$ in a manner similar to MPC.

### IV. RESULTS

Having illustrated our autonomous racing strategy in the previous section, we now show the performance of proposed...
TABLE II: Time taken to overtake the leading vehicle travelling at different speeds. For each group of speed range of the leading vehicles, 100 cases were simulated. The ego vehicle starts from the track’s origin. One other vehicle starts from a random position in the range of 10m ≤ sc,i ≤ 30m. The mean, min and max values show the average overtaking time, minimum overtaking time and maximum overtaking time for the corresponding group. In general, it takes more time to overtake faster moving vehicles on this track since they spend lesser time on the straight segments.

<table>
<thead>
<tr>
<th>Speed Range [m/s]</th>
<th>0 - 0.4</th>
<th>0.4 - 0.8</th>
<th>0.8 - 1.2</th>
<th>1.2 - 1.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean [s]</td>
<td>1.613</td>
<td>2.312</td>
<td>3.857</td>
<td>13.095</td>
</tr>
<tr>
<td>min [s]</td>
<td>0.8</td>
<td>1.2</td>
<td>1.8</td>
<td>3.5</td>
</tr>
<tr>
<td>max [s]</td>
<td>3.6</td>
<td>5.2</td>
<td>21.6</td>
<td>36.1</td>
</tr>
</tbody>
</table>

TABLE III: Overtaking success rate for the ego vehicle after one lap. For each group of speed range of the leading vehicles, 100 cases were simulated. The ego vehicle starts from the track’s origin. Other vehicles start from a random position in the range of 5m ≤ sc,i ≤ 15m. One to three leading cars were simulated.

<table>
<thead>
<tr>
<th>Speed Range [m/s]</th>
<th>0 - 0.4</th>
<th>0.4 - 0.8</th>
<th>0.8 - 1.2</th>
<th>1.2 - 1.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single</td>
<td>100 %</td>
<td>100 %</td>
<td>96 %</td>
<td>84 %</td>
</tr>
<tr>
<td>Two</td>
<td>100 %</td>
<td>100 %</td>
<td>98 %</td>
<td>66 %</td>
</tr>
<tr>
<td>Three</td>
<td>100 %</td>
<td>98 %</td>
<td>84 %</td>
<td>30 %</td>
</tr>
</tbody>
</table>

We can find that when more than one surrounding vehicle exists, much more space would be occupied by other vehicles. As a result, the ego vehicle might not have enough space to accelerate to high speed to pass surrounding vehicles. Although in these cases, the ego vehicle can not overtake all surrounding vehicles after one lap, our proposed racing strategy can still guarantee the ego vehicle’s safety along the track.

During our simulation, the mean solver time for our planner for single, two or three surrounding vehicles is 39.21ms, 39.41ms and 40.23ms. We also notice that when the number of surrounding vehicles is larger than three, the steady complexity still holds but the track becomes too crowded for the ego vehicle to achieve a high success rate of the overtaking maneuver. This validates the steady low computational complexity of our proposed planning strategy thanks to the parallel computation for multiple trajectory optimizations.

C. Racing Without Other Vehicles

As discussed in Sec. III-A when there are no other surrounding vehicles, the ego vehicle adopts the learning-based MPC formulation for trajectory generation and control. In this paper, the learning-based MPC uses historical data from two previous laps and the initial data is calculated offline before the racing task. For the same setup as shown in Fig. 1 the ego vehicle is simulated to race without other agents. It’s found that the ego vehicle slows down when there are surrounding vehicles. This is because the overtaking maneuver happens in a hairpin curve and the curve’s apex is occupied by other moving vehicles, resulting in less space being available for the ego vehicle and thus causing it to slow down to avoid a potential collision. After it passes all surrounding vehicles, the ego vehicle goes back to drive at its speed and steering limit to achieve time-optimal behavior.

V. Conclusion

In this paper, we have presented an autonomous racing strategy that enables an ego vehicle to enhance its lap timing performance while overtaking other moving vehicles. We have verified the performance of our proposed algorithm through numerical simulation, where several surrounding vehicles are simulated to start from random positions with random speeds on a track. Moreover, interaction between the ego vehicle and other surrounding vehicles will be considered in the future work. For instance, autonomous racing strategies such as blocking cars from overtaking are envisaged for the future.
REFERENCES


