Reference Tracking Optimization With Obstacle Avoidance via Task Prioritization for Automated Driving

Francesco Vitale and Claudio Roncoli

***Abstract*— Obstacle avoidance is a fundamental operation for automated driving and its formulation traditionally originates from robotics and decision making control fields. Given the high complexity required to compute an obstacle-free trajectory, this operation is usually demanded to a lower frequency planning layer that provides then a trajectory reference to be followed by a higher frequency control layer. As a result, whenever replanning is needed (for example, due to a new detected obstacle), the control layer must wait for a new planned trajectory to be generated. In this paper, we propose a novel methodology to approach obstacle avoidance already in the control layer, which allows a prompter response. In particular, we show how obstacle avoidance and reference tracking can be integrated, thus with no need to switch among different controllers, based on a null-space based behavioral control approach, implemented in a (possibly nonlinear) model predictive control scheme. We demonstrate practical implementation of the proposed methodology employing two different vehicle dynamic models and in four different (urban and highway) scenarios. Furthermore, we provide a sensitivity analysis to understand how parameters choice affects the automated vehicle behavior.**

***Index Terms*— Obstacle avoidance, null-space based control, model predictive control, automated driving, mixed traffic.**

I. Introduction

AUTOMATED Vehicles (AVs) are making their way through the automotive industry, while vast efforts are devoted to research for safety [1], [2], common practices, and new technologies [3]. Among the latter, a considerable amount of work is related to sensing the environment, *e.g.*, 3D object detection [4] and sensor fusion [5]. Likewise, advances in communication technologies, such as 5G [6], allow vehicular communications [7], [8] giving rise to Connected and Automated Vehicles (CAVs).

CAVs are embraced with enthusiasm because of their potential improvement in terms of energy efficiency, environmental sustainability [9], as well as safety impacts [10], [11]. At the same time, however, new and more complex challenges arise for the scientific and industrial community, such as how to integrate them with conventional (human-driven) vehicles, guarantee connectivity, update (physical and digital) infrastructures, integrate within traffic management strategies, and develop vehicle real-time planning and control strategies (see, *e.g.*, [12], [13], [14], [15]).

A typical architecture for AV operations consists in a hierarchical structure, where the trajectory/path planning layer, which aims at generating a feasible route given a goal and constraints dependent on the surrounding environment, is implemented in the upper layer; whereas, a reference tracking control layer, aiming at continuously minimizing the error with respect to the planned trajectory, is implemented in the lower layer [16].

Finding a collision-free path or trajectory for a mobile agent has always been a topic of concern, because of the increasing interest in mobile robots for various purposes such as warehousing, automation industry, rough terrain exploration, as well as AVs (see, *e.g.*, [17], [18], [19]). Typically, for applications such as AVs, similarly as for other mobile robots, Obstacle Avoidance (OA) is handled at the higher level of control in a hierarchical control system [20]. In such settings, collision avoidance is more often treated as a trajectory planning problem that provides the (lower level) control loop with a collision-free trajectory or path for Reference Trajectory Tracking (RTT). As a result, whenever a new obstacle is detected, or when a monitored dynamic obstacle has a discrepancy with respect to its predicted behavior, a trajectory replanning procedure is needed in the guidance loop. This implies that the agent's response to the surrounding environment is paced by the planning loop, which is typically slower than the low-level control loop, due to the high complexity of such process, caused by the non-linearity of the involved models and the non-convexity of the resulting search problems [21]. To address trajectory planning, various strategies, aiming at solving a collision-free trajectory problem, have been proposed in different fields of mobile robotics [22].

A widely used technique for OA is the Artificial Potential Field approach (see, *e.g.*, [23], [24], [25], [26]), which considers the agent moving in a field of forces, where the target position is modeled as a minimum, while obstacles are maxima. Obstacles may represent either objects that must be

Manuscript received 26 November 2021; revised 19 August 2022, 18 January 2023, and 9 August 2023; accepted 25 August 2023. Date of publication 14 September 2023; date of current version 2 February 2024. This work was supported by the Academy of Finland under Project ULTRA (328216) and under Project ALCOSTO (349327). An earlier version of this paper was presented at the 2021 European Control Conference (ECC) [DOI: 10.23919/ECC54610.2021.9654856]. The Associate Editor for this article was T. Q. Dinh. *(Corresponding author: Francesco Vitale.)*

The authors are with the Department of Built Environment, School of Engineering, Aalto University, 02150 Espoo, Finland (e-mail: francesco.vitale@aalto.fi; claudio.roncoli@aalto.fi).

Digital Object Identifier 10.1109/TITS.2023.3312999

avoided or rules that should be abided ([27], [28], [29]). This intuitive, yet elegant, method is not free from problems and, in [30], several issues were identified during experiments, mainly due to attraction to local minima and oscillatory behavior.

Moreover, search algorithms can be applied for obstacle-free path planning, with the drawback of having to cope with the high-dimensional states. The introduction of RRT [31] and subsequent variants, including, in particular, randomized algorithms [32], contributed to overcome such issues.

Another well-known search method for OA management relies on the idea of Velocity Obstacle [33]. This method consists in selecting avoidance maneuvers generated by agent velocities outside of the velocity obstacle set, which includes velocities that would lead to a collision with a given obstacle at some point in the future. The trajectory from start to goal is computed by searching a tree of feasible avoidance maneuvers, computed at discrete time intervals. Different variants of this technique have been further proposed, including [34], where the Reciprocal Velocity Obstacle concept was introduced for real-time multi-agent navigation. This new version assumes that the other agents react by running a similar collision-avoidance algorithm, guaranteeing safe and oscillation-free motion for all agents.

Reachability-based methods have also been investigated especially in an attempt to bridge the gap between safety and real-time performance (see, *e.g.*, [35], [36]). Another example is [37], where funnel libraries are utilized to compute a suitable trajectory while taking into account uncertainties.

Nonlinear Model Predictive Control (NMPC) [38] has been exploited in many research works (*e.g.*, [39], [40]), due to its ability in managing nonlinear constraints in optimization problems, given by the dynamics of the vehicle and the geometry of the OA task. In particular, for trajectory optimization, Mixed-Integer Linear Program (MILP) has been explored [41], where integer variables are used to define a geometrical area around an agent at a given time instant, not to be accessed by any other one. Nevertheless, integer variables in an optimization problem complicate the process of finding a suitable solution, thus the viability of such a method is subordinate to the specific solver in use and the problem size [42].

Neural networks inspired other approaches for path planning, considering both static and moving obstacles, aiming at making the decision process faster [43] and closer to optimal [44].

Recent research works were devoted to improve the performance of vehicles interacting with systems including uncertainty. Among these, various approaches were investigated, including: robust MPC [45], Gaussian probability distribution propagating along the nominal path [46], Markov processes [47], [48], Monte Carlo dropout and bootstrapping [49], and multi-modal constraint uncertainty [50].

Finally, reactive path following is an alternative approach for the OA problem, where decisions are made directly at the control lower level. This methodology usually comes with a behavior-switching logic [51] that selects which algorithm to trigger among a library of different options, including go-to-goal, collision avoidance, *etc.*

A novel approach, which slightly differs from traditional behavior-switching logic, was presented in [52], where the Null-Space-Based (NSB) behavioral control was employed to demonstrate how a proper composition of outputs based on single elementary behaviors may be effective. The key point is sorting tasks by their priority, so that the input of each task is projected into the null-space of the higher-priority tasks. A stability analysis of such a technique is provided in [53], while another remarkable result in this context is discussed in [54], where an input saturation management procedure was used for a non-holonomic mobile robot. This latter method derives from an effective intuition on how tasks input with different priorities can be properly saturated so that they: a) are compliant with the system physical constraints; and b) do not cause degradation of higher tasks performance [55].

In this paper, we propose a novel approach to achieve prompt, *i.e.*, real-time, reactions to unplanned situations in the context of vehicle control in road traffic. The proposed method is composed of two ingredients, namely: a) an NSB OA algorithm that determines proper velocities to perform evasive maneuvers whenever obstacles are detected; and b) the formulation of an NMPC problem where the underlying optimization problem is subject to proper constraints taking into account the velocities computed by the NSB OA algorithm. Our contribution to this paper is fourfold:

a) we design a novel methodology to integrate an NSB OA algorithm with reference tracking within an NMPC framework;
b) we propose a novel formulation to identify the minimum safety distance in a 2D environment;
c) we present how such a combination of techniques can be implemented with different (vehicle) dynamic systems to tackle different realistic road traffic scenarios;
d) we show the applicability of the proposed technique to react promptly to abrupt obstacles with no need for an immediate replanning of the reference trajectory.

In addition, it is worth noting that previous research works have typically tested the NSB approach for autonomous mobile robots whose physical and environmental constraints are less restricting than those of a CAV. Therefore, to the best of our knowledge, this is the first implementation of such a technique considering road vehicles. The proposed minimum safety distance gives the possibility to compute a 2D distance and to employ it without much of a computation burden. Indeed, a vast variety of spacing policies are available for the sole longitudinal control [56], and, hence, they are suitable for 1D operations. On the other hand, potential fields, which are usually employed in lane-changing decision-making (as for references above), consist of articulated formulations to be managed by the optimization problem. Conversely, our approach identifies a 2D spacing policy to return a simple constant value before computing an instance of the optimization problem. If such a minimum safety distance is violated, then the OA constraint is constructed. An earlier version of this work appeared in [57]. Here, we extend such work by i) considering both a kinematic and a dynamic bicycle model for the controlled vehicle; ii) including sensitivity analysis simulation experiments; iii) performing a comparison with a state-of-the-art approach; and iv) showing the advantage of using the proposed approach in terms of computation time, improving the presentation regarding practical implementation, and considering additional, more realistic traffic scenarios for different circumstances that might be encountered.

This paper is structured as follows. In Section II, we present the formulation of the proposed task priority management method and we describe how to apply it for obstacle avoidance; in Section III, we show how the two components of this method are integrated. In Section IV, we show how to implement our method for two different (vehicle) dynamic models; while in Section V, we present and discuss simulation results. Finally, Section VI summarizes the main findings of this paper and introduces future challenges.

II. Task Priority

In order to manage both trajectory tracking and collision avoidance operations according to priorities, we adopt the tasks formulation proposed in [52]. First, the general mathematical formulation of a task is given (Section II-A), then it is applied to the OA and stopping operations (Sections II-B and II-C). Sections II-D and II-E propose strategies to combine both operations, as well as some operating assumptions.

A. Mathematical Background

Let us introduce $\boldsymbol{\sigma}(t) \in \mathbb{R}^m$ as the task variable to be controlled and $\mathbf{q}(t) \in \mathbb{R}^n$ as the system configuration vector, where n and m are proper state and task dimensions. We can express the task vector as a function of the system configuration, namely

$$\boldsymbol{\sigma}(t) = f\left(\mathbf{q}(t)\right), \tag{1}$$

while its time derivative is

$$\dot{\boldsymbol{\sigma}}(t) = \frac{\partial f\left(\mathbf{q}(t)\right)}{\partial \mathbf{q}}\dot{\mathbf{q}}(t) = J\left(\mathbf{q}(t)\right)\dot{\mathbf{q}}(t), \tag{2}$$

where $J(\mathbf{q}(t)) \in \mathbb{R}^{m\times n}$ is the configuration-dependent task Jacobian matrix and $\dot{\mathbf{q}}(t) \in \mathbb{R}^n$ is the system velocity. We can then introduce the task error vector as

$$\tilde{\boldsymbol{\sigma}} = \boldsymbol{\sigma}_{\text{des}} - \boldsymbol{\sigma}, \tag{3}$$

where the dependencies are dropped for the sake of readability, and $\boldsymbol{\sigma}_{\text{des}} \in \mathbb{R}^m$ is the vector of desired values for the task related to the motion reference $\mathbf{q}_{\text{des}} \in \mathbb{R}^n$. In order to avoid the well-known problem of numerical drift, the system desired velocity can be expressed in a Closed Loop Inverse Kinematics (CLIK) version [58] as

$$\dot{\mathbf{q}}_{\text{des}} = J^{\#}\left(\dot{\boldsymbol{\sigma}}_{\text{des}} + \Lambda\tilde{\boldsymbol{\sigma}}\right), \tag{4}$$

where $J^{\#}$ may be either the transpose or the pseudoinverse of J, *i.e.*, either $J^{\#} = J^{\top}$ or $J^{\#} = J^{+} = J^{\top}(JJ^{\top})^{-1}$, depending on the adopted strategy, and $\Lambda \in \mathbb{R}^{m\times m}$ is a positive-definite gain matrix.

Let us turn our attention to a regulation problem where $N_{\text{T}} > 1$ tasks are to be fulfilled. Denoting tasks as $\boldsymbol{\sigma}_i$, characterized by $\dot{\boldsymbol{\sigma}}_i = 0$, for $i = 1, \ldots, N_{\text{T}}$, we can rewrite (4) as

$$\dot{\mathbf{q}}_{\text{des}} = J_1^{\#}\Lambda_1\tilde{\boldsymbol{\sigma}}_1 + \sum_{i=2}^{N_{\text{T}}} \bar{N}_{i-1}J_i^{\#}\Lambda_i\tilde{\boldsymbol{\sigma}}_i, \tag{5}$$

where

$$\bar{N}_{i-1} = I_{n\times n} - \bar{J}_{i-1}^{\#}\bar{J}_{i-1} \tag{6}$$

is a projector in the null space of $\bar{J}_{i-1}$, $I_{n\times n}$ is the $n \times n$ identity matrix, and $\bar{J}_{i-1}$ is the augmented task Jacobian, *i.e.*,

$$\bar{J}_{i-1} = \begin{bmatrix} J_1 \\ J_2 \\ \vdots \\ J_{i-1} \end{bmatrix}, \tag{7}$$

which allows us to perform a so-called internal motion, that is, we generate a motion of a robotic system without affecting that of primary tasks [59].

B. Obstacle Avoidance

The proposed method abstracts the concept of an obstacle, which may be whichever type of obstruction a vehicle may find in its planned path, be it either idle or moving, a vehicle or a person, a lane edge, or any other signal. The only care is to make sure that the OA task related to such an obstacle has a higher priority than the RTT task. Should the ego-vehicle detect collisions with more than one obstacle, we may want to consider: a) only the most *dangerous* one; b) all of them using (5) and treating RTT as the lowest priority task while the more *dangerous* an obstacle is, the higher priority it has in the OA algorithm; c) a combination of the previous strategies, *e.g.*, considering only some of the obstacles, *etc.*, where the closer to collision an obstacle is, the more *dangerous* it is considered. In Section V, whenever needed, we consider as a primary task the avoidance of another vehicle, while as a secondary task the avoidance of the road boundaries. Such a choice is motivated by solid results in the literature, including, *e.g.*, [52] and [54], thus, we do not focus further on this aspect. On the other hand, we do focus on the novel part, which involves the implementation of this approach in NMPC to perform our simulations.

The obstacle avoidance operation can be tackled by imposing an adequate distance between the ego-vehicle and the obstacle. Such a distance should be carefully computed taking into account the position, speed, and orientation of both. It should be stressed that minimum safety distances for vehicle control are typically constructed to perform longitudinal control operations, such as, *e.g.*, adaptive cruise control. Therefore, while well-known methods are addressing 1-D scenarios, we formulate here a novel method suitable for 2-D applications. To accomplish this, we consider two ellipses to define forbidden areas: one of them centered at the center of gravity of the ego-vehicle and the other one centered at the center of gravity of the obstacle. Ellipses are oriented so that the major axis lies on the yaw direction of the object they refer to. If the object is static, the yaw direction can be chosen

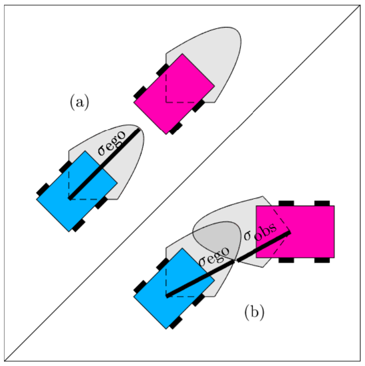


Fig. 1. (a) The ego-vehicle (blue) computes its minimum safety distance to the obstacle vehicle (red) as $\sigma_{des} = \sigma_{ego}$ only. (b) The ego-vehicle computes its minimum safety distance to the obstacle vehicle as $\sigma_{des} = \sigma_{ego} + \sigma_{obs}$, given a higher risk of collision.

arbitrarily, *e.g.*, the direction pointing towards the controlled vehicle. The semi-major and semi-minor axes of the ellipse are denoted as L_{ego} and l_{ego} when referred to the ego-vehicle, and L_{obs} and l_{obs} when referred to the obstacle, respectively. Omitting the dependence on the time step, we define $L_{ego} := 0.5\ s_0 + v_{ego}\Delta_t + 0.5\ a_{ego}\Delta_t^2$ and $l_{ego} := 0.5\ s_0$ for the ego-vehicle, and $L_{obs} := 0.5\ s_0 + v_{obs}\Delta_t + 0.5\ a_{obs}\Delta_t^2$ and $l_{obs} := 0.5\ s_0$ for the obstacle, where s_0 is a minimum safety distance, Δ_t is the time sample, v_{ego} and a_{ego} are the speed and acceleration of the ego-vehicle, respectively, and v_{obs} and a_{obs} are the speed and acceleration of the obstacle, respectively. Therefore, the resulting safety distance to be maintained (see Fig. 1 for a clarification) is

$$\sigma_{des} = \sum_{i\in\{\text{ego, obs}\}} \Phi\left(\frac{L_i l_i}{\sqrt{L_i^2 \sin^2\theta_i + l_i^2\cos^2\theta_i}}, \theta_{max}, s_0\right), \quad (8)$$

where $\Phi(\cdot)$ is a saturation function defined as

$$\Phi(x(\theta), \theta_{max}, s_0) = \begin{cases} x(\theta), & \text{if } |\theta| <= \theta_{max} \\ 0.5\ s_0, & \text{otherwise;} \end{cases} \quad (9)$$

variable θ_{ego} (θ_{obs}) denotes the angle between the position of the obstacle (ego-vehicle) and the orientation of the ego-vehicle (obstacle); and θ_{max} is a maximum range angle encompassing the feasible region of future input. Note that the formula provided as an argument to $\Phi(\cdot)$ in (8) is the distance between the center of an ellipse and the point on such ellipse positioned at a θ_i angle. In addition, notice that $\Phi(\cdot)$ gives as an output at least $0.5\ s_0$, namely half of the minimum safety distance, so that the summation in (8) will always result in $\sigma_{des} \geq s_0$.

We can now proceed formulating our tasks based on the quantities introduced above. Let $\mathbf{q} = \mathbf{p}_{ego} = \left[x_{ego}\ y_{ego}\right]^\intercal$ be the position coordinates of the ego-vehicle, namely its state configuration vector. We construct a task aiming at keeping a given distance from the obstacle, thus

$$\sigma = f(\mathbf{p}_{ego}) = \left\|\mathbf{p}_{ego} - \mathbf{p}_{obs}\right\|, \quad (10)$$

while the desired distance is, *e.g.*, σ_{des} as in (8). Knowing both σ and σ_{des}, we can compute $\tilde{\sigma}$ from (3). It is then straightforward to compute $\dot{\sigma}(t) = J(\mathbf{p}_{ego}(t))\dot{\mathbf{p}}_{ego}(t)$, where

$$J = \frac{\partial\sqrt{\left(\mathbf{p}_{ego}-\mathbf{p}_{obs}\right)^\intercal\left(\mathbf{p}_{ego}-\mathbf{p}_{obs}\right)}}{\partial\mathbf{p}_{ego}} = \frac{\mathbf{p}_{ego}^\intercal - \mathbf{p}_{obs}^\intercal}{\left\|\mathbf{p}_{ego}-\mathbf{p}_{obs}\right\|} = \hat{\mathbf{p}}_{oe}^\intercal \quad (11)$$

and $\hat{\mathbf{p}}_{oe}$ is the unit vector pointing from the position of the center of gravity of the obstacle to that of the ego-vehicle. The vector of velocities is then

$$\dot{\mathbf{q}}_{des} = \dot{\mathbf{p}}_{ego} = (\hat{\mathbf{p}}_{oe}^\intercal)^\#(\Lambda\tilde{\sigma}) \quad (12)$$

and the projector in the null space of the OA task is therefore

$$N = \left[I_{n\times n} - (\hat{\mathbf{p}}_{oe}^\intercal)^\#\hat{\mathbf{p}}_{oe}^\intercal\right]. \quad (13)$$

Recalling that the OA task should be considered with a higher priority, while still tending to accomplish the RTT task, we can define the control input as

$$\dot{\mathbf{p}}_{ego} = J_{obs}^\intercal\Lambda_{obs}\tilde{\sigma}_{obs} + N_{obs}J_{des}^\intercal\Lambda_{des}\tilde{\sigma}_{des}, \quad (14)$$

where J_{obs}, Λ_{obs}, $\tilde{\sigma}_{obs}$, and N_{obs} are the task Jacobian, gain, error to a desired behavior, and null projector of the OA task, respectively, while J_{des}, Λ_{des}, and $\tilde{\sigma}_{des}$ are the task Jacobian, gain, and error to a desired trajectory RTT task, respectively.

Hence, (14) summarizes the obstacle avoidance maneuver strategy that will be considered as a constraint in our NMPC problem (see Section III).

C. *Defining Stopping Maneuvers as Obstacle Avoidance*

Now we present an extension of (14) to handle the special case of defining a task when a stopping maneuver is clearly identified, *e.g.*, when a vehicle approaches a red traffic signal or any other contingency for which stopping is deemed necessary by sensors (see section II-D). Before we proceed, we present a graphical example illustrating how the above-described task priority management algorithm combines the OA and the RTT tasks. As we can observe in Fig. 2a, if the two tasks agree on the input to give, $[0, 1]^\intercal$ in the example, such an input remains unchanged. If the two tasks are completely discordant, namely the OA input direction points towards the anti-parallel direction $[0, -1]^\intercal$ with respect to the RTT desired input, then the RTT task is ignored, while not all of the final control inputs lie in the feasible region, denoted by the gray triangle. A first way to overcome this issue is to let the optimization problem in the subsequent NMPC find the control input within the feasible region. However, in such a way, there is no guarantee that the optimal solution is indeed the most appropriate for a given stopping scenario. A possible solution could employ a different projection method, like the one illustrated in Fig. 2b, where all the feasible inputs do not change, while only the unfeasible ones are projected to obtain a feasible and suitable control. As we can see, the completely discordant OA task control is projected onto the $\mathbf{0}$ velocity. Theoretically, this would mean that providing the completely discordant control input would be enough to effectively brake in front of a stopping signal. Of course, this strategy is inadmissible in a real-world scenario where not only do we need an infinite precision to provide a completely discordant control input, but we also need an infinite precision in the

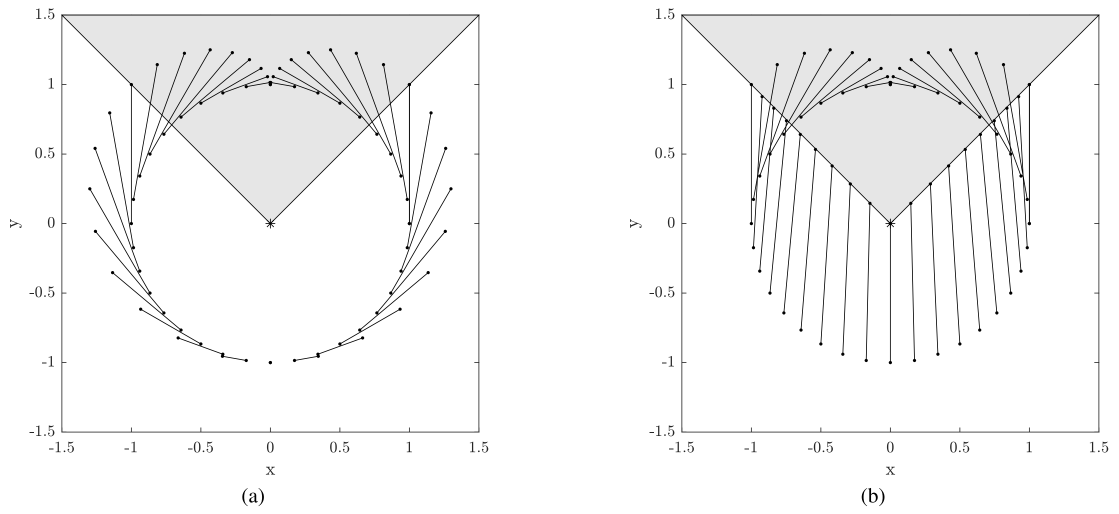


Fig. 2. Example of mapping of primary task velocities towards the final ones, given a secondary task velocity. The ego-vehicle (centered at $[0,0]^\top$) is required to drive with desired speed $[0,1]^\top$. All dots around its position are possible (OA yielded) primary task velocity unit vectors. (a) Lines joining OA velocities with other points are mappings of the primary task, considering the secondary one, to the resulting final input velocity as in (14). (b) Mapping of primary task velocities towards the final velocities and saturated to the feasible region, given a secondary task velocity pointing upward.

projection process as well. Nonetheless, we can still take advantage of this algorithm by assigning the higher priority task input to $\mathbf{0}$ and the null projector to the all-zeros matrix to force a braking maneuver.

D. Dealing With Complex Driving Scenarios

A combination of the previous two solutions to OA and stopping scenarios may be implemented to cope with a wide amount of cases. For example, suppose that we predefine a “giving way to vehicles from the right” rule. Then, whenever a vehicle from the right is *predicted* to intersect the ego-vehicle trajectory, it is possible to: a) apply the stopping algorithm as far as the vehicle is still on the right; and b) eventually apply the OA algorithm to safely continue on the desired direction.

E. Obstacles Measurement Assumptions

A prediction for future states of an obstacle is retrievable either by direct communication with it, *e.g.*, if vehicles are connected, or by some other prediction algorithm, considering, *e.g.*, the current state and constant speed and orientation for the following time steps.

Finally, we assume that sensors are able to provide preliminary useful information. For example, if the controlled vehicle sensed two obstacles too close to each other, so that it is not possible to pass between them, they should be considered as a single obstacle. This assumption, again, is quite reasonable, considering currently available technologies [5], [60], [61].

III. Proposed Algorithms

In this section, we introduce the proposed algorithms to implement the problem described above. The block diagram in Fig. 3 aims at clarifying the context where the proposed method is assumed to operate, which is described hereafter. The sensors block (S) provides internal measurements to the navigation block (N) and external measurements to the mission block (M); in this context, internal measurements include

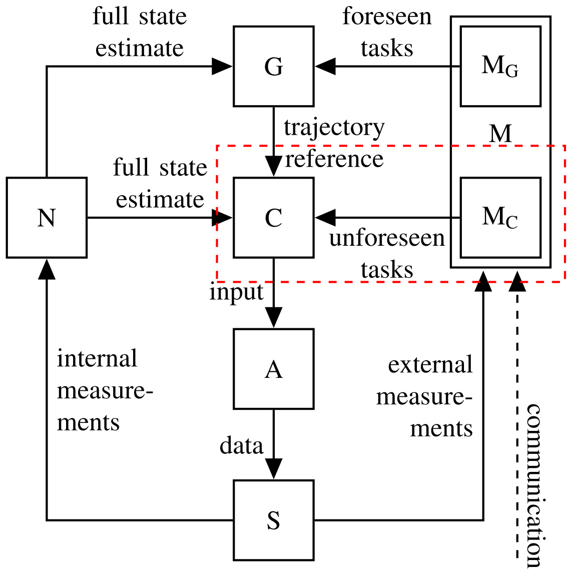


Fig. 3. Schematic block diagram of the proposed framework: guidance (G), control (C), navigation (N), mission (M), mission for guidance (M_G), mission for control (M_C), actuators (A), sensors (S). The blocks implemented in this work are grouped in the red dashed rectangle.

any quantity measured from the vehicle that are useful for vehicle state estimation, while external measurements are the ones originated from, *e.g.*, cameras, radars, and LiDARs. The navigation block (N) is assumed to implement an estimator that is used to reconstruct the full (vehicle) system state. The mission block (M) accommodates, in addition to external measurements, possible communication with connected vehicles infrastructures. It is divided into (M_C) and (M_G), which feed the control block (C) and the guidance block (G), respectively, with the corresponding required data. Block (G) employs the full vehicle state estimate, as well as the foreseen tasks (*i.e.*, tasks that are related to obstacles already detected that limit the configuration space), to compute a trajectory reference that is then fed to block (C). Block (C), besides the trajectory

Algorithm 1 Algorithm to Run at Each NMPC Step

```
Require: $\boldsymbol{\zeta}_i, i \in \{1, \dots, N_{\mathrm{PO}}\}$ possible obstacles states, and NMPC horizon $H$
Ensure: Updated vehicle state $\mathbf{z}$
  $O \leftarrow \emptyset$ ordered set of actual obstacles
  for $h \leftarrow 1$ to $H$ do
    for $k \leftarrow 1$ to $N_{\mathrm{PO}}$ do
      Compute $\sigma$ as in (10)
      Compute $\sigma_{des}$ as in (8)
      if $\sigma \le \sigma_{\mathrm{des}}$ then
        Insert $\boldsymbol{\zeta}_i$ into the proper position of $O$
      end if
    end for
    Let $N_{\mathrm{O}} \leftarrow |O|$
    Run Algorithm 2 (returns $\boldsymbol{\eta}_1 + \sum_{i=2}^{N_{\mathrm{O}}} \left(\bar{N}_{i-1}\boldsymbol{\eta}_i\right)$, $\bar{N}_{N_{\mathrm{O}}}$)
  end for
  Solve NMPC problem (15) instance at time $t$
  Apply control input $\mathbf{u}_1$ at time $t$
```

Algorithm 2 Obstacle Avoidance Algorithm

```
Require: $\mathbf{z}$ Vehicle states, $O$ ordered set of obstacles states, $N_{\mathrm{O}}$ cardinality of $O$, and $\Lambda_{\mathrm{obs}}$ gains
Ensure: $\Sigma := \boldsymbol{\eta}_1 + \sum_{i=2}^{N_{\mathrm{O}}} \left(\bar{N}_{i-1}\boldsymbol{\eta}_i\right)$, and $\bar{N}_{N_{\mathrm{O}}}$
  Let $\Sigma \leftarrow \mathbf{0}$
  Let $\bar{N}_{N_{\mathrm{O}}} \leftarrow I_{n\times n}$
  if $N_{\mathrm{O}} > 0$ then
    Extract $\boldsymbol{\zeta}_1$ from $O$ and use it to:
    Compute $J_1$ and $\tilde{\sigma}_1$ as in (11) and (3)
    Let $\Sigma \leftarrow J_1^{\mathsf{T}}\Lambda_1\tilde{\sigma}_1$
    Let $\bar{N}_{N_{\mathrm{O}}} \leftarrow$ as in equation (13)
    for $i \leftarrow 2$ to $N_{\mathrm{O}}$ do
      Extract $\boldsymbol{\zeta}_i$ from $O$ and use it to:
      Compute $J_i$ and $\tilde{\sigma}_i$ as in (11) and (3)
      Let $\Sigma \leftarrow \Sigma + \bar{N}_{i-1}J_i^{\mathsf{T}}\Lambda_i\tilde{\sigma}_i$
      Let $\bar{N}_{N_O} \leftarrow$ as in equation (6)
    end for
  end if
```

reference, employs the full state estimate and any unforeseen tasks (*i.e.*, sudden obstacles or any obstacle that was not considered in the trajectory planning), to generate the inputs implemented in the actuators block (A). Note that blocks (C) and ($\mathrm{M_C}$) run at a higher frequency than blocks (G) and ($\mathrm{M_G}$). Typical values for the time sample of the (G) block are of the order of tenths of a second, while for the (C) block, they are smaller up to one order of magnitude less (see, *e.g.*, [16], [32], [50], [62], [63], [64]). In this paper, we focus on block (C), for which Algorithm 1 is designed, and ($\mathrm{M_C}$), for which Algorithm 2 is designed. Since the problem is solved in an NMPC framework, both algorithms are supposed to run at each control step.

Algorithm 1 describes all the calculations to be performed at each control step. Vectors $\mathbf{z}$ and $\boldsymbol{\zeta}_i$ are the state vector of the ego-vehicle and of the i^{th} possible obstacle detected, respectively. In what follows, we consider the state (control) vector as the vector of all states (controls) stacked in time, *i.e.*, $\mathbf{z} = \left[z_1^1 \cdots z_1^j \cdots z_1^n\ z_2^1 \cdots z_k^j \cdots z_H^n\right]^{\mathsf{T}}$, where $j = 1, \dots, n$ denotes the different variables and $k = 1, \dots, H$ represents the different time of the receding horizon.

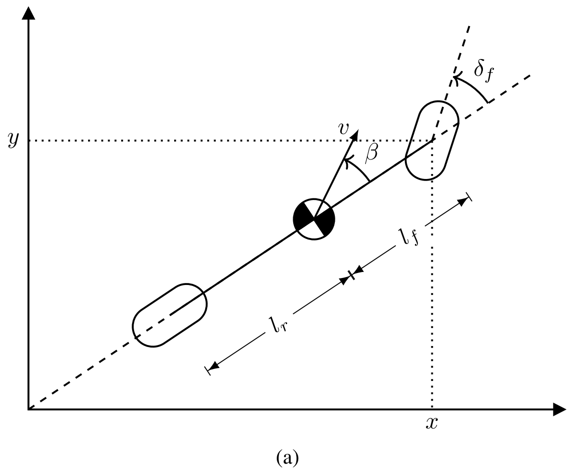
We formulate the NMPC problem as

$$
\begin{aligned}
\min_{\tilde{\mathbf{z}},\tilde{\mathbf{u}}}\ & \frac{1}{2}\tilde{\mathbf{z}}^{\mathsf{T}}Q\tilde{\mathbf{z}} + \frac{1}{2}\tilde{\mathbf{u}}^{\mathsf{T}}R\tilde{\mathbf{u}}\\
\text{s.t. } & \mathbf{z} \in \mathcal{Z}, \quad \bar{\mathbf{u}} \in \bar{\mathcal{U}},\\
& \mathbf{z}_{k+1} = f(\mathbf{z}_k, \mathbf{u}_k),\ \forall k = 1, \dots, H-1\\
& \mathbf{v} = \boldsymbol{\eta}_1 + \sum_{i=2}^{N_{\mathrm{O}}}\left(\bar{N}_{i-1}\boldsymbol{\eta}_i\right) + \bar{N}_{N_{\mathrm{O}}}\mathbf{u}_{\mathrm{aux}},
\end{aligned}
\tag{15}
$$

where the state error from a reference vector is denoted as $\tilde{\mathbf{z}} = \mathbf{z}_{\mathrm{des}} - \mathbf{z}$ and $\mathbf{z}_{\mathrm{des}}$ is the state reference vector. Input error from reference is given by $\tilde{\mathbf{u}} = \bar{\mathbf{u}}_{\mathrm{des}} - \bar{\mathbf{u}}$, where $\bar{\mathbf{u}} = [\mathbf{u}, \mathbf{u}_{\mathrm{aux}}]^{\mathsf{T}}$ is the augmented input vector calculated by solving the optimization problem, and $\bar{\mathbf{u}}_{\mathrm{des}} = [\mathbf{u}_{\mathrm{des}}, \mathbf{v}_{\mathrm{des}}]^{\mathsf{T}}$ is the augmented desired input vector. Vectors $\mathbf{u}_{\mathrm{aux}}$ and $\mathbf{v}_{\mathrm{des}}$ are an auxiliary vector and the vector of desired velocities, respectively. These two latter vectors are deliberately defined generically, since they can be adapted to the vehicle dynamic model in use (for examples, see Section IV). Again, we consider the velocity $\mathbf{v}_{\mathrm{des}}$ as the vector of stacked desired velocities over time. In the cost function, Q is a positive semidefinite matrix and R is a positive definite matrix. Note that weights for desired speed tracking are placed in the R matrix, while the Q matrix values may be null at the corresponding entries. In particular, we define $Q := \operatorname{diag}\left\{q_{z_1^1}, \cdots, q_{z_1^n}, q_{z_2^1}, \cdots, q_{z_H^n}\right\}$, where $q_{z_k^j}$ is the weight associated to state variable z_k^j, and, similarly, $R := \operatorname{diag}\left\{r_{\bar{u}_1^1}, \cdots, r_{\bar{u}_1^m}, r_{\bar{u}_2^1}, \cdots, r_{\bar{u}_H^m}\right\}$, where $r_{\bar{u}_k^j}$ is the weight associated to control variable $\bar{u}_k^j$. The first constraint guarantees that $\mathbf{z}$ and $\bar{\mathbf{u}}$ are in their respective feasible regions; the second constraint is a discretized version of the vehicle dynamic model; whereas the third constraint corresponds to the result of the OA algorithm, returned from Algorithm 2, where $\mathbf{v}$ is the set of stacked velocities over the time horizon as it is defined in the state vector (whatever way it is defined), $\boldsymbol{\eta}_i$ is the input yielded by the obstacle with priority i in the OA algorithm, and $\bar{N}_i$ is given by (6). Basically, we are allowing an auxiliary velocity vector $\mathbf{u}_{\mathrm{aux}}$ to track the desired speed $\mathbf{v}_{\mathrm{des}}$; then, projecting $\mathbf{u}_{\mathrm{aux}}$ into the null space of higher tasks will eventually result in the final velocity; note that the latter is part of the state vector. As a last remark, one may consider excessive to perform the OA constraint at each NMPC time step. Indeed, we could identify a *check horizon* $\bar{H}$ smaller than or equal to the control horizon. In such a case, the OA constraint would be applied from the detection of an obstacle to no more than $\bar{H}$ time instants in the NMPC instance. This aspect will be further investigated in a set of sensitivity analysis experiments presented in Section V-B.

IV. System Dynamics

We present here two vehicle models that can be incorporated within our method. In particular, Section IV-A describes a kinematic bicycle model and Section IV-B describes a dynamic



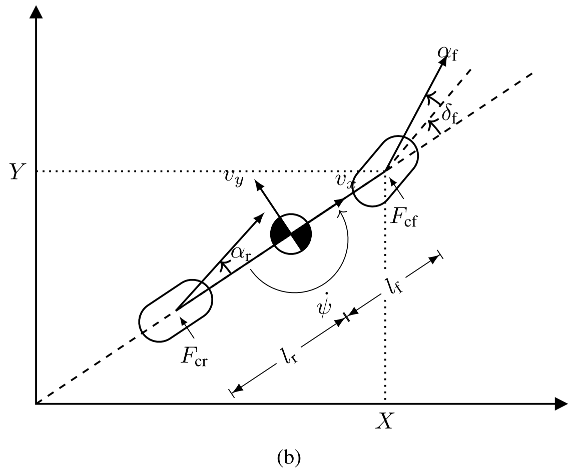


Fig. 4. (a) Kinematic bicycle model used in the simulations. (b) Dynamic bicycle model used in the simulations.

bicycle model. These dynamical models are chosen as examples to validate the proposed methodology. Indeed, both the kinematic and dynamic models are broadly used in literature for similar applications (see, *e.g.*, [12], [65]), and, in particular, the kinematic model is typically used for trajectory planning, while the dynamic model is more employed for reference tracking control. However, it is worth mentioning that also different, more complex, models, have been investigated in literature (see, *e.g.*, [64]) and a strength of the proposed method is that Problem (15) can be straightforwardly adaptable to them. Note that, in both formulations, time dependency is dropped for the sake of readability and, with abuse of notation, all operations are meant component-wise.

A. Kinematic Bicycle Model

We consider the following kinematic bicycle model, depicted in Fig. 4a, where (16)–(20) are from [66], and (21) describes the OA task, defined as in (5):

$$\dot{\mathbf{x}} = \mathbf{v}\cos(\boldsymbol{\psi} + \boldsymbol{\beta}) \tag{16}$$

$$\dot{\mathbf{y}} = \mathbf{v}\sin(\boldsymbol{\psi} + \boldsymbol{\beta}) \tag{17}$$

$$\dot{\mathbf{v}} = \mathbf{a} \tag{18}$$

$$\dot{\boldsymbol{\psi}} = \frac{\mathbf{v}}{l_\mathrm{r}}\sin(\boldsymbol{\beta}), \tag{19}$$

$$\text{with } \boldsymbol{\beta} = \tan^{-1}\left(\frac{l_\mathrm{r}}{l_\mathrm{f} + l_\mathrm{r}}\tan(\boldsymbol{\delta}_\mathrm{f})\right), \tag{20}$$

$$\begin{bmatrix}\mathbf{v}\cos\boldsymbol{\psi}\\ \mathbf{v}\sin\boldsymbol{\psi}\end{bmatrix} = \boldsymbol{\eta}_1 + \sum_{i=2}^{N_O}\bar{N}_{i-1}\boldsymbol{\eta}_i + \bar{N}_{N_O}\begin{bmatrix}\mathbf{v}_\mathrm{aux}\cos\boldsymbol{\psi}_\mathrm{aux}\\ \mathbf{v}_\mathrm{aux}\sin\boldsymbol{\psi}_\mathrm{aux}\end{bmatrix}, \tag{21}$$

where $(\mathbf{x}, \mathbf{y})$ is the ego-vehicle coordinate vector, $\mathbf{v}$ is the speed, and $\boldsymbol{\psi}$ is the heading angle; all variables are referred to a Cartesian global coordinate system. The angle between the vehicle velocity and its longitudinal axis is denoted as $\boldsymbol{\beta}$, while $\mathbf{a}$ is the acceleration along the same direction. The distance from the center of gravity of the vehicle to the front and rear axles are denoted as l_f and l_r, respectively. Finally, $\boldsymbol{\delta}_\mathrm{f}$ denotes the front steering angle, while the rear steering angle is assumed to be null. In (21), $\boldsymbol{\eta}_i$ is the velocity obtained to avoid the i^{th} obstacle, N_O is the number of obstacles to be avoided, $\mathbf{v}_\mathrm{aux}$ and $\boldsymbol{\psi}_\mathrm{aux}$ are the auxiliary inputs as already discussed, and $\bar{N}_i$ is the set of null projectors appropriately stacked according to other vectors. For this system, the state vector is $\mathbf{z} = \begin{bmatrix}\mathbf{x}^\mathsf{T} & \mathbf{y}^\mathsf{T} & \mathbf{v}^\mathsf{T} & \boldsymbol{\psi}^\mathsf{T}\end{bmatrix}^\mathsf{T}$ and the input vector is $\mathbf{u} = \begin{bmatrix}\mathbf{a}^\mathsf{T} & \boldsymbol{\beta}^\mathsf{T} & \mathbf{v}_\mathrm{aux}^\mathsf{T} & \boldsymbol{\psi}_\mathrm{aux}^\mathsf{T}\end{bmatrix}^\mathsf{T}$.

B. Dynamic Bicycle Model

We consider the following dynamic bicycle model, depicted in Fig. 4, where (22)–(27) are from [66], and (28) describes the OA task, defined as in (5):

$$\dot{\mathbf{X}} = \mathbf{v}_x\cos(\boldsymbol{\psi}) - \mathbf{v}_y\sin(\boldsymbol{\psi}) \tag{22}$$

$$\dot{\mathbf{Y}} = \mathbf{v}_x\sin(\boldsymbol{\psi}) + \mathbf{v}_y\cos(\boldsymbol{\psi}) \tag{23}$$

$$\dot{\boldsymbol{\psi}} = \frac{\mathbf{v}_x}{l_\mathrm{f} + l_\mathrm{r}}\tan(\boldsymbol{\delta}_\mathrm{f}) \tag{24}$$

$$m\dot{\mathbf{v}}_x = \mathbf{F}_x + m\mathbf{v}_y\dot{\boldsymbol{\psi}} - 2\mathbf{F}_\mathrm{cf}\sin(\boldsymbol{\delta}_\mathrm{f}) \tag{25}$$

$$m\dot{\mathbf{v}}_y = -m\mathbf{v}_x\dot{\boldsymbol{\psi}} + 2\left(\mathbf{F}_\mathrm{cf}\cos(\boldsymbol{\delta}_\mathrm{f}) + \mathbf{F}_\mathrm{cr}\right) \tag{26}$$

$$I_z\ddot{\boldsymbol{\psi}} = 2\left(l_\mathrm{f}\mathbf{F}_\mathrm{cf}\cos(\boldsymbol{\delta}_\mathrm{f}) - l_\mathrm{r}\mathbf{F}_\mathrm{cr}\right) \tag{27}$$

$$\begin{bmatrix}\mathbf{v}_x\\ \mathbf{v}_y\end{bmatrix} = \boldsymbol{\eta}_1 + \sum_{i=2}^{N_O}\bar{N}_{i-1}\boldsymbol{\eta}_i + \bar{N}_{N_O}\begin{bmatrix}\mathbf{v}_{x\mathrm{aux}}\\ \mathbf{v}_{y\mathrm{aux}}\end{bmatrix}, \tag{28}$$

where $(\mathbf{X}, \mathbf{Y})$ is the position vector in the global coordinate system and, analogously for $\boldsymbol{\psi}$ orientation, $\mathbf{v}_x$ and $\mathbf{v}_y$ are the longitudinal and lateral speed, while $\dot{\boldsymbol{\psi}}$ is the yaw rate. The distance from the front axle and rear axle to the center of gravity are l_f and l_r, respectively. The exerted force, the front tire force, and the rear tire force are, respectively, $\mathbf{F}_x$, $\mathbf{F}_\mathrm{cf}$, and $\mathbf{F}_\mathrm{cr}$, given $\mathbf{F}_\mathrm{cf} = C_f\left(\delta_\mathrm{f} - \arctan\left(\frac{\mathbf{v}_y + l_\mathrm{f}\dot{\boldsymbol{\psi}}}{\mathbf{v}_x}\right)\right)$, and $\mathbf{F}_\mathrm{cr} = -C_\mathrm{r}\arctan\left(\frac{\mathbf{v}_y - l_\mathrm{r}\dot{\boldsymbol{\psi}}}{\mathbf{v}_x}\right)$, where C_f and C_r are the cornering stiffness of the front and rear tire respectively. In (28), $\boldsymbol{\eta}_i$ is the velocity obtained to avoid the i^{th} obstacle, N_O is the number of obstacles to be avoided, $\mathbf{v}_{x\mathrm{aux}}$ and $\mathbf{v}_{y\mathrm{aux}}$ are the auxiliary inputs as already discussed, and $\bar{N}_i$ is the set of null projectors appropriately stacked according to other vectors. Here, the state vector is $\mathbf{z} = \begin{bmatrix}\mathbf{X}^\mathsf{T} & \mathbf{Y}^\mathsf{T} & \boldsymbol{\psi}^\mathsf{T} & \mathbf{v}_x^\mathsf{T} & \mathbf{v}_y^\mathsf{T} & \dot{\boldsymbol{\psi}}^\mathsf{T}\end{bmatrix}^\mathsf{T}$ and the input vector is $\mathbf{u} = \begin{bmatrix}\mathbf{a}^\mathsf{T} & \boldsymbol{\delta}_\mathrm{f}^\mathsf{T} & \mathbf{v}_{x\mathrm{aux}}^\mathsf{T} & \mathbf{v}_{y\mathrm{aux}}^\mathsf{T}\end{bmatrix}^\mathsf{T}$.

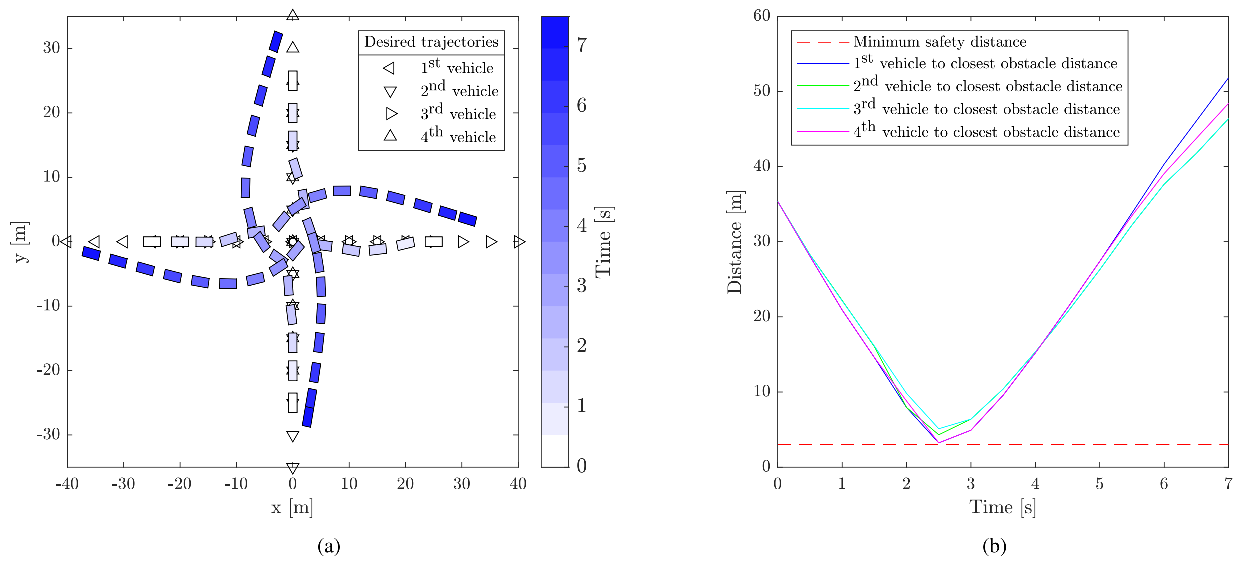


Fig. 5. (a) Four vehicles with intersecting desired trajectories deviate their paths to avoid collisions. Rectangles represent vehicles positions colored depending on the time step they refer to (starting instant = light blue, final instant = dark blue). (b) Distances between each vehicle and the relative closest obstacle throughout the simulation.

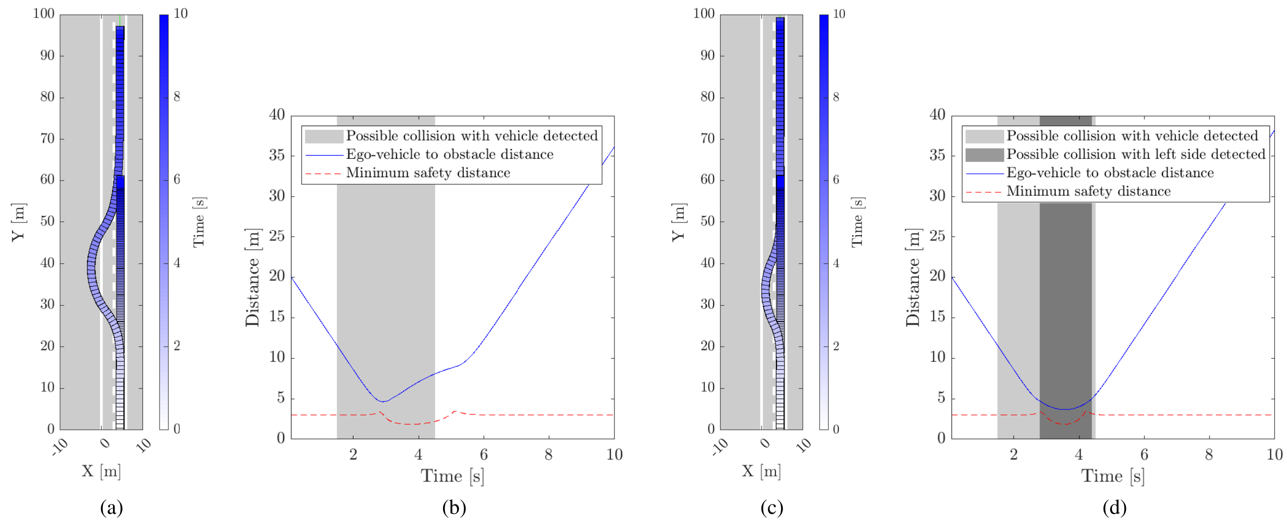


Fig. 6. (a) Overtaking maneuver considering only the vehicle in front as an obstacle. (b) The controlled vehicle and the obstacle vehicle never collide. (c) Overtaking considering both the vehicle in front and the left boundary of the road as obstacles. (d) The controlled vehicle never collides with the obstacle vehicle.

V. Simulation Experiments

In this section, we present a set of simulation experiments with the goal of evaluating the suitability and effectiveness of the proposed method for different road traffic scenarios. First, in Section V-A, we consider a warm-up scenario where four vehicles, aligned along a circumference, are moving towards the diametrically opposed point so that they are expected to intersect their trajectories in the middle of their path. Such a simulation, even though not realistic in traffic situations, is widely employed when testing obstacle avoidance algorithms (see, *e.g.*, [34]). Then, in Section V-B, we present a scenario characterized by an overtaking maneuver; in addition, for this scenario, we perform a sensitivity analysis to assess the impact of parameters' choice on the proposed algorithm, discuss the computational aspects of the proposed method, and present a comparison with a state-of-the-art method. After that, we include two additional scenarios, representative of different realistic traffic situations. In particular, in Section V-C, we investigate a giving way-turning-overtaking scenario; while, in Section V-D, we consider a lane drop in a multi-lane environment. Note that, besides the specific settings, these last scenarios incorporates many different behaviors that can be induced by our method, such as stopping, giving way, and overcoming a sequence of obstacles. Mastering these actions are, of course, indispensable for an AV to move safely in a typical (urban) road environment.

All simulation experiments are performed utilizing forward Euler discretized versions of the dynamical systems. Each NMPC step time instant is in the order of a tenth of a second, which is a commonly employed value for the dynamic bicycle model (see, *e.g.*, [65]). Whenever non-connected vehicles are involved in the following, the controlled vehicle assumes

TABLE I
PARAMETERS USED FOR THE DYNAMIC BICYCLE MODEL

l_f [m]	l_r [m]	m [kg]	I_z [kg · m^2]	C_f [N/rad]	C_r [N/rad]
1.4	1.6	2350	4132	80000	90000

TABLE II
ACCELERATION AND STEERING TO VARYING OF THEIR WEIGHTS

	$q_{\mathbf{a}} = q_{\boldsymbol{\delta}_{\mathrm{f}}} = 0.1$	$q_{\mathbf{a}} = q_{\boldsymbol{\delta}_{\mathrm{f}}} = 10$	$q_{\mathbf{a}} = q_{\boldsymbol{\delta}_{\mathrm{f}}} = 45$
$\lvert\mathbf{a}\rvert_\infty$	0.1666 m/s^2	0.0138 m/s^2	0.0062 m/s^2
$\lvert\boldsymbol{\delta}_{\mathrm{f}}\rvert_\infty$	0.1077 rad	0.0332 rad	0.0204 rad

they will maintain a constant speed and orientation for the prediction horizon used in the NMPC.

A. Scenario 1: Simple Crossing

In this warm-up simulation experiment, we consider four CAVs, which can communicate with one another to inform about their future positions. The model we implement here is the kinematic bicycle model (16)–(21). The desired trajectories are set in such a way to all intersect at $(0, 0)$. Fig. 5a shows how vehicles properly deviate from their desired trajectory to avoid collisions. The vertical color bar on the right side confirms that vehicles positions do not overlap at any time step. Notice that, in this case, only the closest obstacle was considered to perform the evasive maneuver and, therefore, no other traffic rule was considered; namely, vehicles were not forced to overtake on a given side or avoid going outside boundaries. The proper functioning of our algorithm is confirmed by inspecting Fig. 5, which shows that distances between each vehicle and its relative closest obstacle is always greater than a constant minimum safety distance.

B. Scenario 2: Overtaking Maneuver

An overtaking scenario is considered in this second simulation experiment. As a prediction within the NMPC, the controlled AV assumes that the other vehicle (obstacle) moves at constant speed and orientation. For this scenario, we employ the dynamic bicycle model (22)–(28) with parameters defined as in Table I.

First, we test a situation where the controlled vehicle considers the sole vehicle in front as an obstacle, which results in the ego-vehicle succeeding to perform the overtaking maneuver. Fig. 6a shows that the trajectories of the vehicles only intersect at different time instants (see the color bar on the right for a clarified view), while Fig. 6b shows that the distance between the two vehicles never goes below the minimum safety distance. However, in practice, road boundaries restrict the space where a vehicle is allowed to move. To address that, we include road boundaries as additional obstacles by considering a static (*i.e.*, with zero velocity) virtual obstacle on the left side of the controlled vehicle, which is present at any time instant for which a collision with the edge is detected. In this case, the ego-vehicle succeeds in performing the overtaking maneuver while still remaining within the road boundaries, as we can see from Figs. 6c and 6d. Indeed, first the ego-vehicle detects a possible collision with another vehicle, starting the overtaking maneuver as it identified there is available space on the left side. Then, when the orientation of the ego-vehicle reaches a value that could lead to a collision with the left road boundary, the latter is seen as a second obstacle and a corresponding constraint is activated. Once the vehicle on the right is not seen as an obstacle anymore, the overtaking maneuver is completed and the ego-vehicle returns to its nominal trajectory tracking behavior.

Sensitivity Analysis: It is important to understand how parameters affect the results obtained via the proposed algorithms. Therefore, we perform a sensitivity analysis to investigate the behavior of the proposed method for a set of significant parameters, evaluating their impact on the results in this scenario. In particular, we consider various values for the receding horizon size, the cost function weights, and the gain Λ.

In Fig. 7, we show different results obtained for different H and $\bar{H}$, namely, for how many steps the algorithm updates the OA constraint for an obstacle once it is detected. As expected, the time at which a collision with an obstacle is sensed is the same for all the simulations and the longer an obstacle is considered as a possible collision, the longer the OA constraint is triggered. On the other hand, it is interesting to notice that how long the left boundary is considered as a secondary obstacle does not show significantly different patterns. In any case, the minimum safety distance is always kept and the distance between the two vehicles differs very slightly for the set of parameters tested in our study.

Let us turn our attention to the cost function weights in Q and R. Note that positive weights should necessarily be set for $q_{\mathbf{X}}$, $q_{\mathbf{Y}}$, $r_{\mathbf{v}_{x\mathrm{aux}}}$, and $r_{\mathbf{v}_{y\mathrm{aux}}}$, since they are applied to the error related to the desired trajectory (*i.e.*, $\tilde{\mathbf{X}}$, $\tilde{\mathbf{Y}}$, $\tilde{\mathbf{v}}_{x\mathrm{aux}}$, and $\tilde{\mathbf{v}}_{x\mathrm{aux}}$) and, hence, allow a proper RTT operation. Therefore, we set to 1 the corresponding entries in Q and R and vary the others. The most interesting weights to probe are those for the acceleration $\mathbf{a}$ and steering angle $\boldsymbol{\delta}_{\mathrm{f}}$. We tested different values for them (smaller and bigger than 1 and various combinations). For the sake of brevity, in Fig 8, we show results obtained by only two particular configurations: one with acceleration and steering weights set to 0.1 (hence, smaller than the others), and one with acceleration and steering weights set to 10 (hence, bigger than the others). As expected, acceleration and steering angle are smaller if they have bigger weights, as can be observed from Figs. 8a, 8b, 8c, and 8d. On the other hand, when the weights set for the steering angle and acceleration are higher, the controlled vehicle drives less aggressively, it implements less pronounced orientations and, as a consequence, it perceives the left boundary as an obstacle for shorter time (Fig. 8f) than otherwise (Fig. 8e). Of course, if a too large penalty is set in the cost function, then the vehicle may not be able to avoid the obstacles as in the case depicted in Figs. 9 and 10, where $r_{\mathbf{a}}$ and $r_{\boldsymbol{\delta}_{\mathrm{f}}}$ are set to 45. Table II shows the maximum of the absolute value obtained for acceleration and steering angle to varying of their respective weights. Modifying other weights do not seem to strongly affect results.

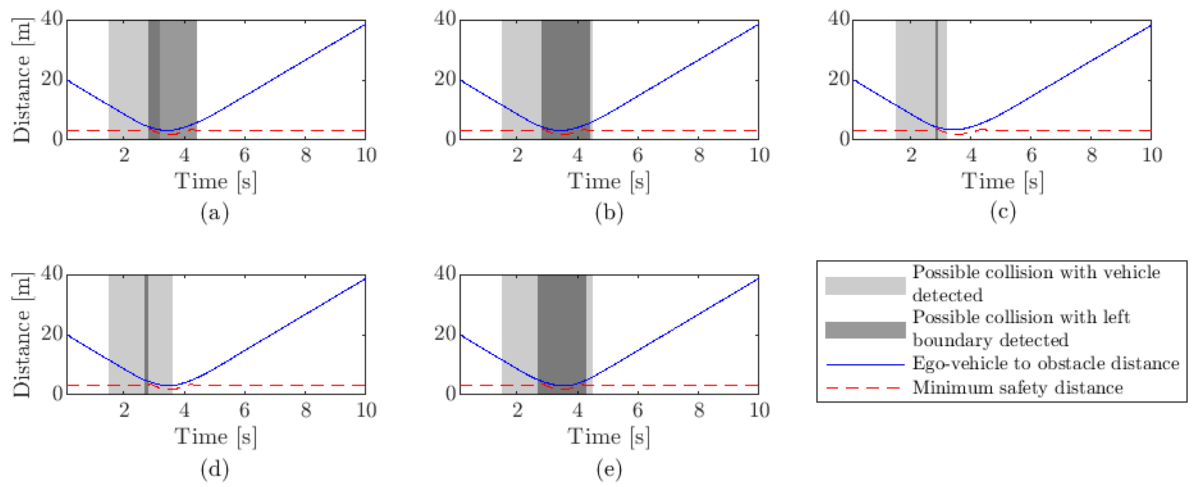


Fig. 7. Effects on obstacle detection and distance resulting by utilizing different control horizon H and *check horizon* $\bar{H}$; (a) $H = 1$ s, $\bar{H} = 0.5$ s. (b) $H = 1$ s, $\bar{H} = 1$ s. (c) $H = 2$ s, $\bar{H} = 0.5$ s. (d) $H = 2$ s, $\bar{H} = 1$ s. (e) $H = 2$ s, $\bar{H} = 2$ s.

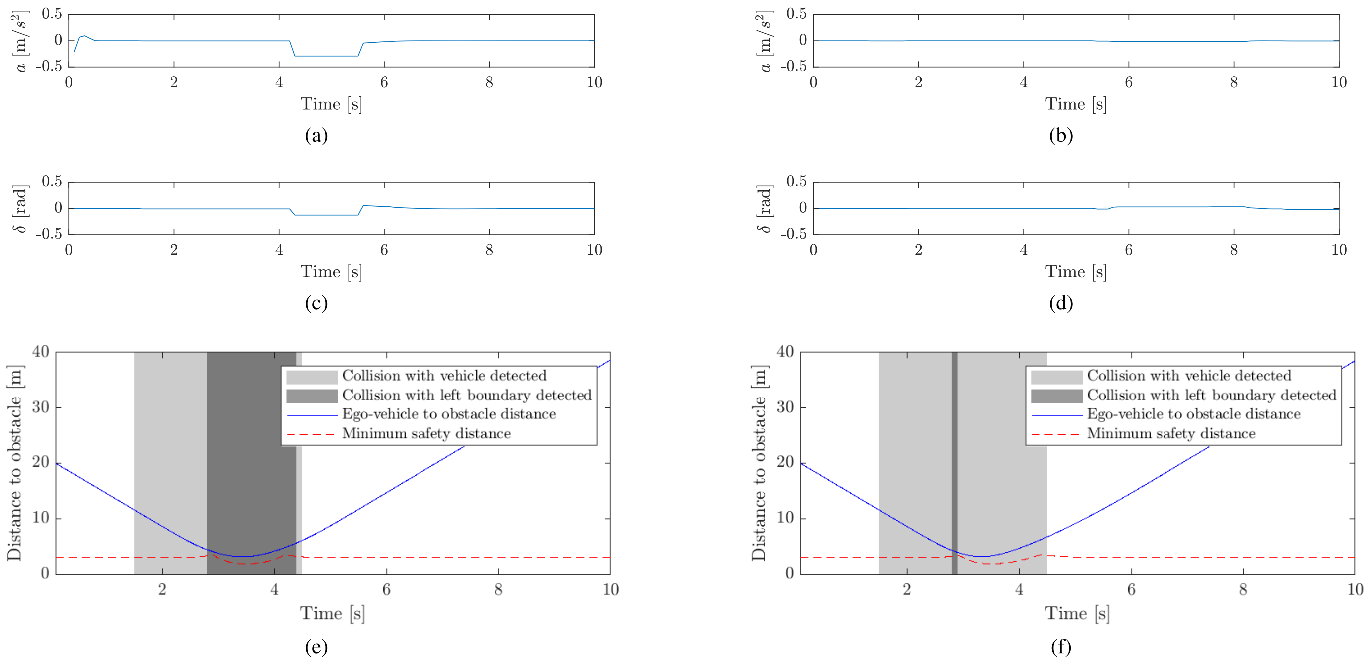


Fig. 8. Testing different weights for the cost function. (a) Acceleration when its corresponding weight is smaller than the others, *i.e.* $q_{\mathbf{X}} = q_{\mathbf{Y}} = r_{\mathbf{v}_{x\text{aux}}} = r_{\mathbf{v}_{y\text{aux}}} = 1$ and $r_{\mathbf{a}} = r_{\delta_{\text{f}}} = 0.1$. (b) Acceleration when its corresponding weight is larger than the others, *i.e.* $q_{\mathbf{X}} = q_{\mathbf{Y}} = r_{\mathbf{v}_{x\text{aux}}} = r_{\mathbf{v}_{y\text{aux}}} = 1$ and $r_{\mathbf{a}} = r_{\delta_{\text{f}}} = 10$. (c) Steering angle when its corresponding weight is smaller than the others, *i.e.* $q_{\mathbf{X}} = q_{\mathbf{Y}} = r_{\mathbf{v}_{x\text{aux}}} = r_{\mathbf{v}_{y\text{aux}}} = 1$ and $r_{\mathbf{a}} = r_{\delta_{\text{f}}} = 0.1$. (d) Steering angle when its corresponding weight is larger than the others, *i.e.* $q_{\mathbf{X}} = q_{\mathbf{Y}} = r_{\mathbf{v}_{x\text{aux}}} = r_{\mathbf{v}_{y\text{aux}}} = 1$ and $r_{\mathbf{a}} = r_{\delta_{\text{f}}} = 10$. (e) Distance to the obstacle when acceleration and steering have smaller weight than the others, *i.e.* $q_{\mathbf{X}} = q_{\mathbf{Y}} = r_{\mathbf{v}_{x\text{aux}}} = r_{\mathbf{v}_{y\text{aux}}} = 1$ and $r_{\mathbf{a}} = r_{\delta_{\text{f}}} = 0.1$. (f) Distance to the obstacle when acceleration and steering have bigger weight than the others, *i.e.* $q_{\mathbf{X}} = q_{\mathbf{Y}} = r_{\mathbf{v}_{x\text{aux}}} = r_{\mathbf{v}_{y\text{aux}}} = 1$ and $r_{\mathbf{a}} = r_{\delta_{\text{f}}} = 10$.

Lastly, we consider the gain parameter Λ, which is utilized in Algorithm 2. As a rule of thumb, it should not be too large, otherwise an excessive avoidance maneuver could lead the ego-vehicle outside the road, whereas it is more advisable to keep it smaller when an obstacle is in front, so that the OA operation can begin gently. Figs. 11a and 11b show what happens when the obstacle is in front, and Λ is set to 0.1 and 0.2, respectively, while always employing $\Lambda = 1$ when the obstacle is behind the ego-vehicle. Indeed, already for $\Lambda = 0.2$, the vehicle performs a too aggressive maneuver that leads it slightly out of the road. On the other hand, Fig. 12a and 12b, show results for $\Lambda = 0.01$ and $\Lambda = 100$, respectively, when the obstacle is behind the ego-vehicle, while $\Lambda = 0.1$ is employed when the obstacle is in front. In these cases, the gain does not seem to significantly affect the final results, even if we can observe that a slightly larger maneuver is performed for $\Lambda = 100$. This is probably due to the fact that the obstacle behind yields an OA velocity that is more in accordance with the RTT one, so that no big disruption is introduced. Based on our experimental results, it is reasonable to employ $\Lambda \leq 0.1$ when the obstacle is in front and $\Lambda \leq 1$ when it is behind. Thus, for the simulation shown in the rest of this

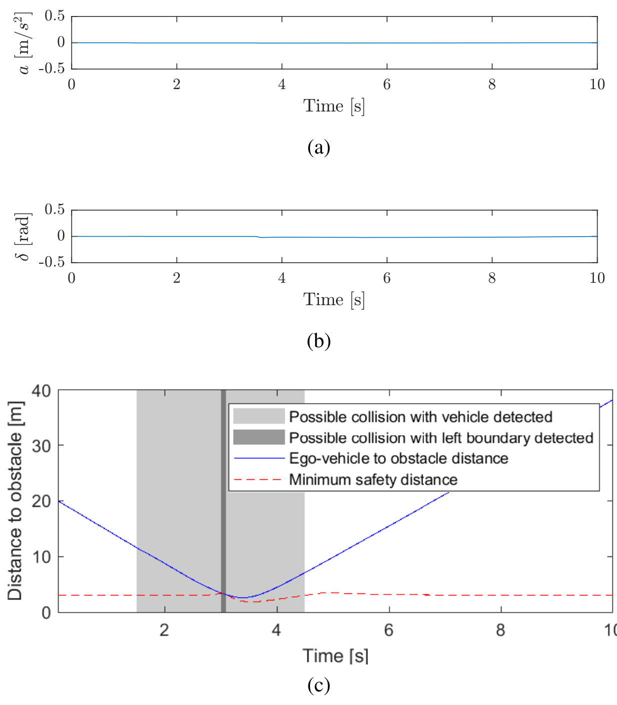


Fig. 9. Cost function weights set as $q_{\mathbf{X}} = q_{\mathbf{Y}} = r_{\mathbf{v}_{x\text{aux}}} = r_{\mathbf{v}_{x\text{aux}}} = 1$ and $r_{\mathbf{a}} = r_{\boldsymbol{\delta}_f} = 45$. (a) Changes in acceleration are barely noticeable. (b) Changes in steering are barely noticeable. (c) A collision with the obstacle vehicle does occur.

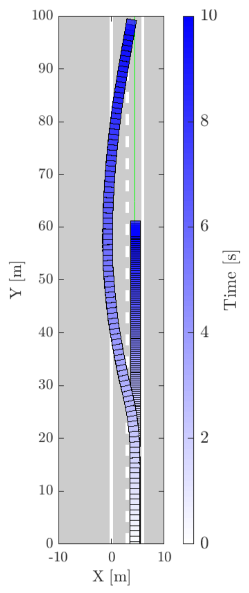


Fig. 10. If too high weights are set on the acceleration and steering, then the vehicle is unable to effectively avoid the obstacle and go back on track.

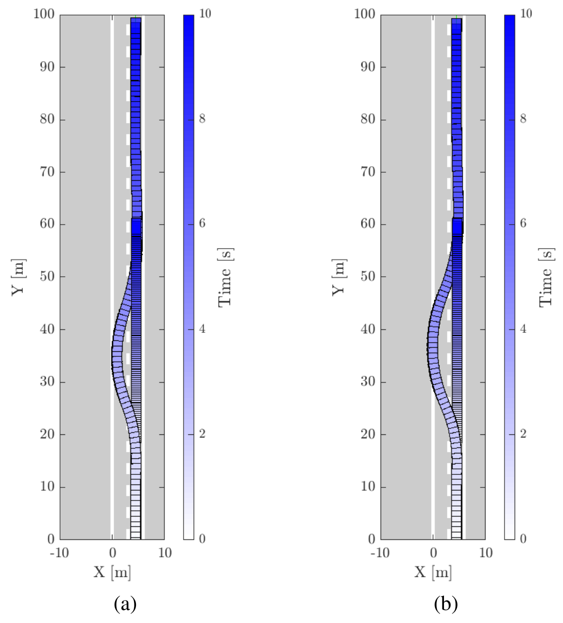


Fig. 11. Testing different Λ values when the obstacle is in front, while keeping $\Lambda = 1$ when the vehicle is behind. (a) $\Lambda = 0.1$ for the obstacle in front. (b) $\Lambda = 0.2$ for the obstacle in front.

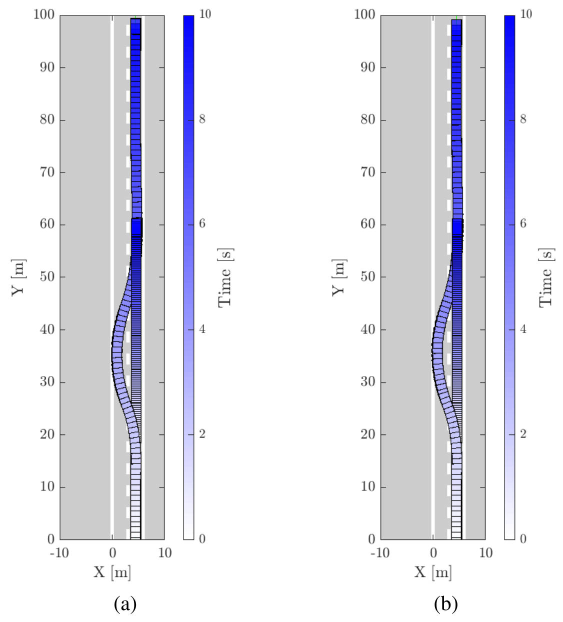


Fig. 12. Testing different Λ values when the obstacle is behind, while keeping $\Lambda = 1$ when the vehicle is in front. (a) $\Lambda = 0.01$ for the obstacle behind. (b) $\Lambda = 100$ for the obstacle behind.

paper, we employ $\Lambda = 0.05$ when the obstacle is in front of the ego-vehicle and $\Lambda = 0.5$ when the obstacle is behind the ego-vehicle, respectively.

Computational Efficiency: We used MATLAB and `nlmpc` (part of Model Predictive Control Toolbox [67]). For the first simulation, we considered a 2.5 s optimization window with a 0.5 s time sample. For the remaining simulations (excluding those for the sensitivity analysis, as already discussed) we considered a 1 s optimization window with a 0.1 s time sample. Simulations were performed on a machine with the following:

Processor: Intel(R) Core(TM) i5-8365U CPU @ 1.60 GHz 1.90 GHz

RAM: 16.0 GB (15.8 GB usable)

OS type: 64-bit Windows 10, x64-based processor

On an average, an NMPC step optimization plus OA took 60% more time than a simple RTT. Nevertheless, this result is not to be considered of much concern, since there are many solutions to improve the performance of NMPC [68]. In fact, many other papers have already focused on faster

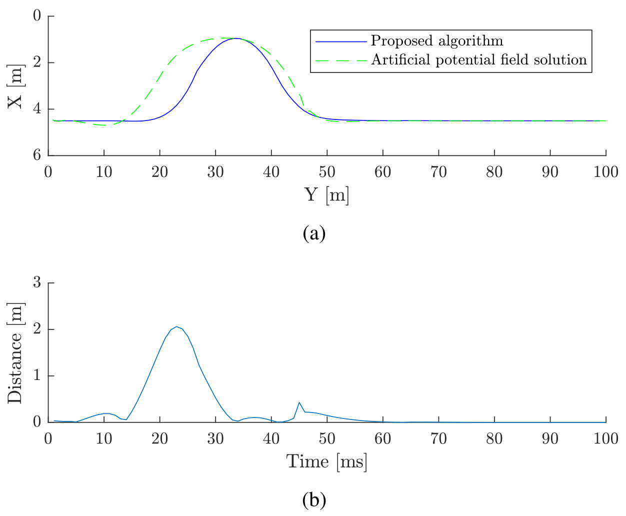


Fig. 13. Comparison between the proposed algorithm, considering an unforeseen obstacle, and an artificial potential field method, where the obstacle is known in advance. a) Most of the difference is evident during the first part of the overtaking maneuver, since the proposed algorithm is reacting to an unforeseen obstacle, while the artificial potential field algorithm is aware of it well in advance. b) The lateral position error, *i.e.* the distance between the *X* coordinate between the two algorithms is higher at the beginning and then goes back to zero.

implementation of NMPC on vehicle nonlinear dynamics even for smaller time steps (see, *e.g.*, [62], [63], [69]). Moreover, the proposed technique enjoys a remarkable generalization and it is applicable to different systems, for which we only presented the nonlinear bicycle models. Therefore, such algorithms would adapt to linearized models as well. Furthermore, even if our algorithms run in series by construction, some expedients can quicken the whole process. For example, if the time sample is sufficiently small, it could be an option to parallelize the two algorithms, so that the NMPC procedure at time instant t runs while data on obstacles that will be dangerous in time instant $t+1$ are being processed. Finally, using a dedicated machine, with optimized code using a more efficient programming language, such as C++, and the compiler would allow us to get efficient online performances.

Comparison With an Artificial Potential Field Method: An interesting evaluation of the proposed method can be given by comparing our solution with that obtained via a state-of-the-art trajectory planning method, where the latter optimizes the vehicle trajectory by knowing in advance information about obstacles. In particular, we consider an artificial potential field-based method, where the objective function penalizes the error from a desired trajectory and the input effort similarly to the proposed algorithm, while the closeness to obstacles (vehicles and roadside) are defined as in [26]. Figs. 13a and 13b show how most of the difference between the two solutions lies in the first phases of the overtaking maneuver. This result is as expected since, in the artificial potential field method, the presence of the obstacle is known a-priori, giving the possibility to consider the evasive maneuver since the beginning. Nevertheless, even though the simulation for the proposed algorithm is constructed so that the ego-vehicle learns about the presence of the obstacle only later, it is still capable to react to it and accurately track the desired trajectory.

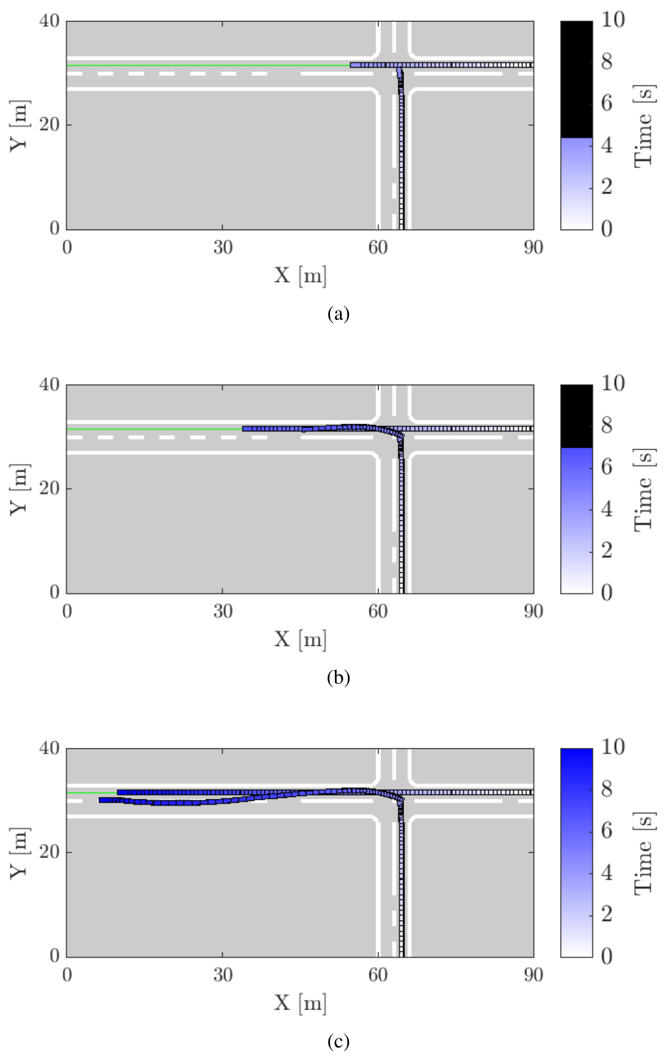


Fig. 14. Simulation of an intersection. (a) The ego-vehicle (coming from the down leg) decelerates to let the non-CAV vehicle go. (b) The ego-vehicle turns left and detects the non-CAV vehicle again. (c) The ego-vehicle performs an OA maneuver to overtake the slower obstacle.

In order to solve the mixed integer nonlinear programming problem resulting from the artificial potential field strategy, the optimization software APMonitor [70], [71] is utilized. Within our experiments, the solution took 4.206 s to be found, which means that, in case a sudden obstacle is encountered, a vehicle would need such a long time to re-plan a trajectory and then apply its own tracking control algorithm to follow it. On the other hand, our optimization problem, also coded in APMonitor for the sake of fair comparison, yielded a solution in 0.418 s, namely about 10 times faster, and this is already comprehensive of the tracking control capabilities, i.e., no further algorithm (and time) is needed.

C. Scenario 3: Giving Way - Turning - Overtaking

In this simulation experiment, we consider a four-leg intersection with two vehicles: a controlled vehicle willing to perform a left turn (hence whose planned desired trajectory exhibits a left turn) and a "traditional" manually-driven non-connected vehicle arriving from the right and willing to go

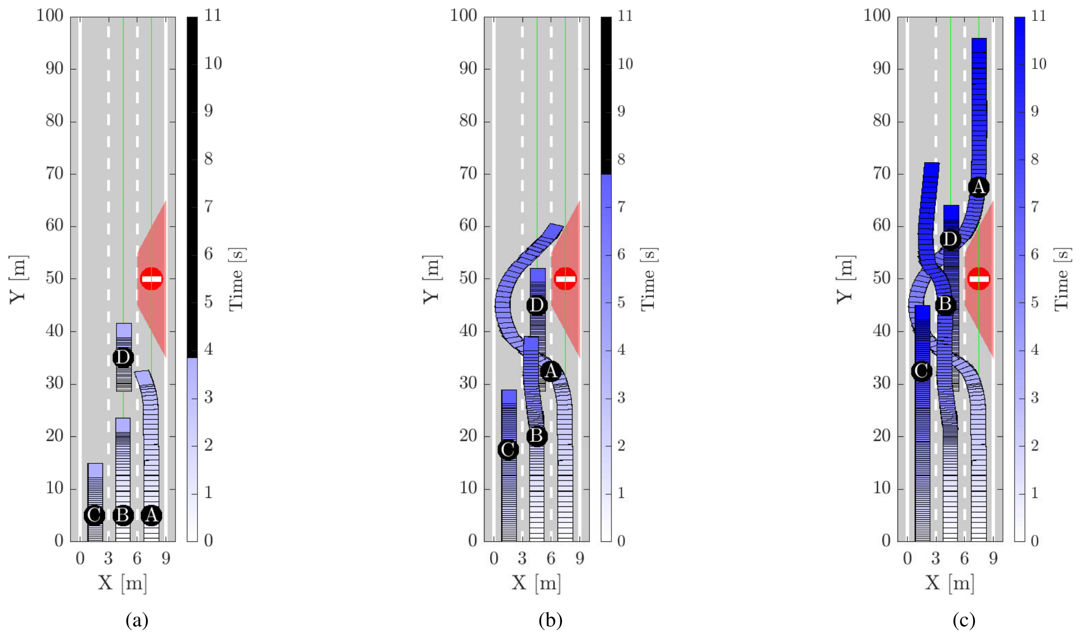


Fig. 15. CAVs A and B driving together with non-CAVs C and D. Desired trajectories for A and B are denoted by a green line. (a) Vehicle A senses a lane drop and starts a lane change maneuver. Vehicle B starts decelerating as there is no room for any other maneuver. (b) Vehicle A detects vehicle D and overtakes it. Vehicle B, once vehicle A has gone, accelerates. (c) Vehicle A is free to continue tracking its reference. Vehicle B detects the possibility to overtake vehicle D without colliding with vehicle C.

straight ahead (see also Fig. 14). The two techniques of stopping and obstacle avoidance are implemented here. First (Fig. 14a), the controlled vehicle senses the other vehicle approaching from its right side with a trajectory that may lead to a collision and, consequently, it decelerates, since there are no other options. Once the non-connected vehicle crosses the intersection (Fig. 14b), the controlled vehicle can freely accelerate and keep the desired trajectory again. When the two vehicles are getting too close to each other (Fig. 14c), the controlled vehicle performs an overtaking maneuver to avoid collision and continues on its way.

D. Scenario 4: Lane Drop

In this last simulation experiment, we consider a scenario in which an unforeseen lane drop, hence unplanned from the trajectory planning, is sensed. The setup consists of four vehicles: two of them are assumed to be connected and automated, *i.e.*, they are controlled by our proposed methodology and they are able to exchange information on their predicted states resulting from the NMPC process, while other two are assumed to be human-driven, moving at a constant speed and heading. In this example, we consider again the dynamic bicycle model (22)–(28) as a dynamic model for the CAVs. We present three different figures to illustrate the resulting behavior. Fig. 15a shows one of the CAVs, vehicle A (the rightmost one), sensing the lane drop; although its desired trajectory suggests to proceed straight, the controlled vehicle starts a lane-changing maneuver to avoid the obstacle in front of it. The second CAV, vehicle B, senses that vehicle A is going to enter its lane and intersect its path (moreover, in this case, they actually communicate to each other their intents). Vehicle B has, thus, an obstacle on the right, but it senses a non-CAV, vehicle C, on its left as well. In this case, since there is no space for an OA maneuver, vehicle B starts slowing down implementing a stopping maneuver (see Section II-C). In Fig. 15b, we observe that vehicle A senses another non-CAV vehicle (vehicle D) and, therefore, it starts an OA maneuver with respect to it, while still coping with the left side of the road. Meanwhile, vehicle B does not detect any obstacle, since vehicle A is not sensed anymore as an obstacle, and, consequently, it accelerates to keep up with its desired trajectory. Finally, Fig. 15 shows that vehicle A is again able to follow its reference trajectory, while vehicle B is approaching vehicle D and, insofar, engages an OA maneuver. Of course, this last maneuver is deemed safe since vehicle C is not sensed as a close obstacle.

Notice that this simulation is of great significance because it summarizes how the proposed technique behaves in case of: 1) stopping required, *i.e.* when vehicle B is surrounded by other vehicles; 2) giving way, again for vehicle B, while sensing vehicle A approaching; and 3) coping with different obstacles in sequence, namely vehicle A that senses a lane drop, communicates with vehicle B so that it can safely change lane, and overtakes vehicle D paying attention to the left side of the road.

VI. Conclusion

In this paper, we presented a novel approach for providing a prompt response to unforeseen contingencies in the control loop of a vehicle in an (urban) traffic context. In particular, we integrated a null-space-based behavioral control-like algorithm with reference tracking in a Nonlinear Model Predictive Control framework. Such a solution allows us to take advantage of the robustness of NMPC while running an obstacle avoidance algorithm in real-time. The validity of our approach was evaluated via a sensitivity analysis, identifying some reasonable bounds for the most interesting and relevant parameters coming into play. Since this strategy permits to react promptly to unexpected obstacles, the trajectory replanning process can safely take more time, as it normally requires, to generate a more convenient reference trajectory. Of course, this would allow the guidance operation to reach a solution closer to the optimal one. This work paves the way for future developments, besides the possibility to relax the requirements for trajectory planning in terms of computation time. Indeed, it would be interesting to test the proposed framework in a larger variety of more complex cases, both for urban and highway traffic, and even more interestingly with real-vehicle data, in order to, *e.g.*, fine-tune some of the parameters. Another interesting extension would be the development of a fully distributed version of the proposed method, so that (connected) vehicles are able to cooperate rather than to simply react to each other. Finally, some more features regarding sensors may be considered, especially investigating the preferred type of data, the amount of pre-processing that may be needed, as well as the robustness of the proposed method to noise and inaccuracy.

References

[1] I. Y. Noy, D. Shinar, and W. J. Horrey, "Automated driving: Safety blind spots," *Safety Sci.*, vol. 102, pp. 68–78, Feb. 2018.

[2] P. Junietz, U. Steininger, and H. Winner, "Macroscopic safety requirements for highly automated driving," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 2673, no. 3, pp. 1–10, Mar. 2019.

[3] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58443–58469, 2020.

[4] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, "A survey on 3D object detection methods for autonomous driving applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 10, pp. 3782–3795, Oct. 2019.

[5] Z. Wang, Y. Wu, and Q. Niu, "Multi-sensor fusion in automated driving: A survey," *IEEE Access*, vol. 8, pp. 2847–2868, 2020.

[6] G. A. Akpakwu, B. J. Silva, G. P. Hancke, and A. M. Abu-Mahfouz, "A survey on 5G networks for the Internet of Things: Communication technologies and challenges," *IEEE Access*, vol. 6, pp. 3619–3647, 2017.

[7] P. K. Singh, S. K. Nandi, and S. Nandi, "A tutorial survey on vehicular communication state of the art, and future research directions," *Veh. Commun.*, vol. 18, Aug. 2019, Art. no. 100164.

[8] H. Ullah, N. G. Nair, A. Moore, C. Nugent, P. Muschamp, and M. Cuevas, "5G communication: An overview of vehicle-to-everything, drones, and healthcare use-cases," *IEEE Access*, vol. 7, pp. 37251–37268, 2019.

[9] M. Taiebat, A. L. Brown, H. R. Safford, S. Qu, and M. Xu, "A review on energy, environmental, and sustainability implications of connected and automated vehicles," *Environ. Sci. Technol.*, vol. 52, pp. 11449–11465, Sep. 2018.

[10] Y. Lian, G. Zhang, J. Lee, and H. Huang, "Review on big data applications in safety research of intelligent transportation systems and connected/automated vehicles," *Accident Anal. Prevention*, vol. 146, Oct. 2020, Art. no. 105711.

[11] Z. Yao, R. Hu, Y. Jiang, and T. Xu, "Stability and safety evaluation of mixed traffic flow with connected automated vehicles on expressways," *J. Safety Res.*, vol. 75, pp. 262–274, Dec. 2020.

[12] J. Guanetti, Y. Kim, and F. Borrelli, "Control of connected and automated vehicles: State of the art and future challenges," *Annu. Rev. Control*, vol. 45, pp. 18–40, Jan. 2018.

[13] I. Papamichail et al., "Motorway traffic flow modelling, estimation and control with vehicle automation and communication systems," *Annu. Rev. Control*, vol. 48, pp. 325–346, Jan. 2019.

[14] Y. Zhang and C. G. Cassandras, "An impact study of integrating connected automated vehicles with conventional traffic," *Annu. Rev. Control*, vol. 48, pp. 347–356, Jan. 2019.

[15] S. Jin, D.-H. Sun, M. Zhao, Y. Li, and J. Chen, "Modeling and stability analysis of mixed traffic with conventional and connected automated vehicles from cyber physical perspective," *Phys. A, Stat. Mech. Appl.*, vol. 551, Aug. 2020, Art. no. 124217.

[16] O. Sharma, N. C. Sahoo, and N. B. Puhan, "Recent advances in motion and behavior planning techniques for software architecture of autonomous vehicles: A state-of-the-art survey," *Eng. Appl. Artif. Intell.*, vol. 101, May 2021, Art. no. 104211.

[17] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: A survey," *Robotica*, vol. 33, no. 3, pp. 463–497, Mar. 2015.

[18] P. Victerpaul, D. Saravanan, S. Janakiraman, and J. Pradeep, "Path planning of autonomous mobile robots: A survey and comparison," *J. Adv. Res. Dyn. Control Syst.*, vol. 9, no. 12, pp. 1535–1565, 2017.

[19] L. Claussmann, M. Revilloud, D. Gruyer, and S. Glaser, "A review of motion planning for highway autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 5, pp. 1826–1848, May 2020.

[20] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous UAV guidance," *J. Intell. Robotic Syst.*, vol. 57, nos. 1–4, pp. 65–100, Jan. 2010.

[21] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, Mar. 2016.

[22] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1135–1145, Apr. 2016.

[23] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous Robot Vehicles*. New York, NY, USA: Springer, 1986, pp. 396–404.

[24] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, "Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints," *IEEE Trans. Veh. Technol.*, vol. 66, no. 2, pp. 952–964, Apr. 2016.

[25] Y. Rasekhipour, A. Khajepour, S.-K. Chen, and B. Litkouhi, "A potential field-based model predictive path-planning controller for autonomous road vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 5, pp. 1255–1267, Oct. 2016.

[26] K. Makantasis and M. Papageorgiou, "Motorway path planning for automated road vehicles based on optimal control methods," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 2672, no. 19, pp. 112–123, Dec. 2018.

[27] M. T. Wolf and J. W. Burdick, "Artificial potential functions for highway driving with collision avoidance," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2008, pp. 3731–3736.

[28] R. Daily and D. M. Bevly, "Harmonic potential field path planning for high speed vehicles," in *Proc. Amer. Control Conf.*, Jun. 2008, pp. 4609–4614.

[29] J. Wang, J. Wu, and Y. Li, "The driving safety field based on driver–vehicle–road interactions," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 4, pp. 2203–2214, Aug. 2015.

[30] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 2, Apr. 1991, pp. 1398–1404.

[31] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011.

[32] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, May 2001.

[33] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Robot. Res.*, vol. 17, no. 7, pp. 760–772, Jul. 1998.

[34] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2008, pp. 1928–1935.

[35] S. Kousik, S. Vaskov, F. Bu, M. Johnson-Roberson, and R. Vasudevan, "Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots," *Int. J. Robot. Res.*, vol. 39, no. 12, pp. 1419–1469, 2020.

[36] S. Manzinger, C. Pek, and M. Althoff, "Using reachable sets for trajectory planning of automated vehicles," *IEEE Trans. Intell. Vehicles*, vol. 6, no. 2, pp. 232–248, Jun. 2021.

[37] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *Int. J. Robot. Res.*, vol. 36, no. 8, pp. 947–982, Jul. 2017.

[38] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control: Theory Algorithms*. London, U.K.: Springer-Verlag, 2017.

[39] M. Werling and D. Liccardo, "Automatic collision avoidance using model-predictive online optimization," in *Proc. 51st IEEE Conf. Decis. Control (CDC)*, Dec. 2012, pp. 6309–6314.

[40] A. Gray, M. Ali, Y. Gao, J. Hedrick, and F. Borrelli, "Semi-autonomous vehicle control for road departure and obstacle avoidance," in *Proc. IFAC Symp. Control Transp. Syst.*, 2012, pp. 1–6.

[41] A. Richards and J. P. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *Proc. Amer. Control Conf.*, vol. 3. Anchorage, AK, USA, May 2002, pp. 1936–1941.

[42] E. Klotz and A. M. Newman, "Practical guidelines for solving difficult mixed integer linear programs," *Surv. Oper. Res. Manage. Sci.*, vol. 18, nos. 1–2, pp. 18–32, Oct. 2013.

[43] R. Glasius, A. Komoda, and S. C. A. M. Gielen, "Neural network dynamics for path planning and obstacle avoidance," *Neural Netw.*, vol. 8, no. 1, pp. 125–133, Jan. 1995.

[44] N. H. Singh and K. Thongam, "Neural network-based approaches for mobile robot navigation in static and moving obstacles environments," *Intell. Service Robot.*, vol. 12, no. 1, pp. 55–67, Jan. 2019.

[45] R. Soloperto, J. Köhler, F. Allgöwer, and M. A. Müller, "Collision avoidance for uncertain nonlinear systems with moving obstacles using robust model predictive control," in *Proc. 18th Eur. Control Conf. (ECC)*, Jun. 2019, pp. 811–817.

[46] H. Banzhaf, M. Dolgov, J. Stellet, and J. M. Zöllner, "From footprints to beliefprints: Motion planning under uncertainty for maneuvering automated vehicles in dense scenarios," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 1680–1687.

[47] C. Hubmann, J. Schulz, M. Becker, D. Althoff, and C. Stiller, "Automated driving in uncertain environments: Planning with interaction and uncertain maneuver prediction," *IEEE Trans. Intell. Vehicles*, vol. 3, no. 1, pp. 5–17, Mar. 2018.

[48] J. Feng, C. Wang, C. Xu, D. Kuang, and W. Zhao, "Active collision avoidance strategy considering motion uncertainty of the pedestrian," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 4, pp. 3543–3555, Apr. 2022.

[49] B. Lutjens, M. Everett, and J. P. How, "Safe reinforcement learning with model uncertainty estimates," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 8662–8668.

[50] I. Batkovic, U. Rosolia, M. Zanon, and P. Falcone, "A robust scenario MPC approach for uncertain multi-modal obstacles," *IEEE Control Syst. Lett.*, vol. 5, no. 3, pp. 947–952, Jul. 2021.

[51] D. Liberzon, *Switching in Systems and Control*. Cham, Switzerland: Springer, 2003.

[52] G. Antonelli, F. Arrichiello, and S. Chiaverini, "The null-space-based behavioral control for autonomous robotic systems," *Intell. Service Robot.*, vol. 1, no. 1, pp. 27–39, Jan. 2008.

[53] G. Antonelli, "Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems," *IEEE Trans. Robot.*, vol. 25, no. 5, pp. 985–994, Oct. 2009.

[54] F. Arrichiello, S. Chiaverini, P. Pedone, A. A. Zizzari, and G. Indiveri, "The null-space based behavioral control for non-holonomic mobile robots with actuators velocity saturation," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 4019–4024.

[55] G. Indiveri, "Swedish wheeled omnidirectional mobile robots: Kinematics analysis and control," *IEEE Trans. Robot.*, vol. 25, no. 1, pp. 164–171, Feb. 2009.

[56] C. Wu, Z. Xu, Y. Liu, C. Fu, K. Li, and M. Hu, "Spacing policies for adaptive cruise control: A survey," *IEEE Access*, vol. 8, pp. 50149–50162, 2020.

[57] F. Vitale and C. Roncoli, "An MPC-based task priority management approach for connected and automated vehicles reference tracking with obstacle avoidance," in *Proc. Eur. Control Conf. (ECC)*, Jun. 2021, pp. 813–819.

[58] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators," *IEEE Trans. Robot. Autom.*, vol. 13, no. 3, pp. 398–410, Jun. 1997.

[59] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Cham, Switzerland: Springer, 2010.

[60] W. Shi, M. B. Alawieh, X. Li, and H. Yu, "Algorithm and hardware implementation for visual perception system in autonomous vehicle: A survey," *Integration*, vol. 59, pp. 148–156, Sep. 2017.

[61] Y. Ma, Z. Wang, H. Yang, and L. Yang, "Artificial intelligence applications in the development of autonomous vehicles: A survey," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 2, pp. 315–329, Mar. 2020.

[62] M. A. Abbas, R. Milman, and J. M. Eklund, "Obstacle avoidance in real time with nonlinear model predictive control of autonomous vehicles," *Can. J. Electr. Comput. Eng.*, vol. 40, no. 1, pp. 12–22, Winter 2017.

[63] W. Farag, "Complex track maneuvering using real-time MPC control for autonomous driving," *Int. J. Comput. Digit. Syst.*, vol. 9, no. 5, pp. 909–920, 2020.

[64] N. Chowdhri, L. Ferranti, F. S. Iribarren, and B. Shyrokau, "Integrated nonlinear model predictive control for automated driving," *Control Eng. Pract.*, vol. 106, Jan. 2021, Art. no. 104654.

[65] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2015, pp. 1094–1099.

[66] R. Rajamani, *Vehicle Dynamics and Control*. Cham, Switzerland: Springer, 2011.

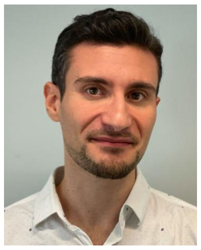
[67] *Model Predictive Control Toolbox*, MathWorks, Inc., Natick, MA, USA, 2020. [Online]. Available: https://se.mathworks.com/help/mpc/

[68] V. M. Zavala, C. D. Laird, and L. T. Biegler, "Fast implementations and rigorous models: Can both be accommodated in NMPC?" *Int. J. Robust Nonlinear Control*, vol. 18, no. 8, pp. 800–815, 2008.

[69] S. S. Oyelere, "The application of model predictive control (MPC) to fast systems such as autonomous ground vehicles (AGV)," *IOSR J. Comput. Eng.*, vol. 16, no. 3, pp. 27–37, 2014.

[70] J. D. Hedengren, R. A. Shishavan, K. M. Powell, and T. F. Edgar, "Nonlinear modeling, estimation and predictive control in APMonitor," *Comput. Chem. Eng.*, vol. 70, pp. 133–148, Nov. 2014.

[71] L. Beal, D. Hill, R. Martin, and J. Hedengren, "GEKKO optimization suite," *Processes*, vol. 6, no. 8, p. 106, 2018.



Francesco Vitale received the B.Eng. degree in information technology engineering and the M.Eng. degree in computer engineering from the University of Salento, Lecce, Italy, in 2016 and 2019, respectively. He is currently pursuing the Ph.D. degree in spatial planning and transportation engineering with the Department of Built Environment, School of Engineering, Aalto University, Espoo, Finland. His research interests include autonomous mobile robots, distributed optimization and control, and machine learning.

Claudio Roncoli received the Ph.D. degree from the University of Genoa, Italy, in 2013. He is currently an Associate Professor in transportation engineering with Aalto University, Finland. Before joining Aalto University, he was a Research Assistant with the University of Genoa, a Visiting Research Assistant with Imperial College London, U.K., and a Post-Doctoral Researcher with the Technical University of Crete, Greece. He has been involved in several national and international research projects as a principal investigator. His research interests include real-time traffic management, modeling, optimization, and the control of traffic systems with connected and automated vehicles, and smart mobility and intelligent transportation systems.