A Dynamics-Aware NMPC Method for Robot Navigation Among Moving Obstacles

Spyridon G. Tarantos(✉) and Giuseppe Oriolo

Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, via Ariosto 25, 00185 Roma, Italy
{tarantos,oriolo}@diag.uniroma1.it

Abstract. We present a novel method for mobile robot navigation among obstacles. Our approach is based on Nonlinear Model Predictive Control (NMPC) and uses a dynamics-aware collision avoidance constraint. The constraint, built upon the notion of *avoidable collision state*, considers not only the robot-obstacle distance but also their velocity as well as the robot actuation capabilities. To highlight the effectiveness of this constraint, we compare the proposed method with a version of the NMPC that uses a constraint purely based on distance information, showing that the first achieves better performance than the second, especially when the robot travels at higher speed among several moving obstacles. Results indicate that the method can work with relatively short prediction horizons and is therefore amenable to real-time implementation.

Keywords: Robot navigation · Mobile robots · Collision avoidance · NMPC

1 Introduction

In service applications, mobile robots must be able to navigate a variety of different environments. Apart from task fulfillment and kinodynamic feasibility, collision avoidance during motion is an essential requirement. This makes the use of real-time motion planning necessary.

In the literature, there is a huge number of motion planning methods. In [4] the authors categorize direct motion planning approaches as grid-based search methods [8,9], randomized probabilistic methods [15], artificial potential fields [14] and approaches based on the solution of an Optimal Control Problem (OCP) [1]. This last category includes NMPC, which is increasingly used as a motion generation technique. Using the robot dynamics as prediction model and through appropriate state and input constraints, NMPC solves an OCP at each time instant to compute an optimal motion which guarantees kinodynamic feasibility and collision avoidance. Moreover, it can be applied to both single

I. Petrovic et al. (Eds.): IAS 2022, LNNS 577, pp. 216–230, 2023.
https://doi.org/10.1007/978-3-031-22216-0_15

and multi-body robots with complex dynamics, a clear advantage over classical grid-based search approaches for navigation, such as the dynamic window approach [9]. Finally, thanks to the constant increase in computational capabilities and the use of solution approaches like real time iteration (RTI) [6] and appropriate software packages for the solution of the OCP like ACADO [13], it is possible to achieve real-time performance for NMPC.

Regarding the collision avoidance constraint, the structure of the environment affects its formulation. In some structured environments, collision avoidance can be often treated as a corridor navigation problem after proper reformulation of the equations of motion, see [11,16]. However, in the general case this is not possible. In [2], signed distance fields are used in order to generate collision-free trajectories. In [21], the authors proposed a reformulation of the generic collision avoidance constraint as a smooth nonlinear version more appropriate for numerical optimization. In [18], a soft collision avoidance constraint is introduced that considers obstacles described by general non-convex sets, while [7] emphasizes the importance of hard collision avoidance constraints in real-time NMPC.

It should be noted that the collision avoidance constraints in the aforementioned works are purely *distance-based*, without any consideration of the dynamic state of the robot. In an NMPC, the effectiveness of a distance-based collision avoidance method will critically depend on the length of the prediction horizon in relation to the robot actuation capabilities. The longer the prediction horizon, the earlier an imminent collision can be detected and averted without significant (sometimes prohibitive) actuation effort. On the other hand, real-time performance obviously does not allow arbitrarily long prediction horizons. In practice, since high-speed navigation requires the use of the robot dynamic model and high control frequency, the maximum achievable prediction horizon on typical robot processing platforms ends up being relatively short. Consequently, the use of a purely distance-based collision avoidance constraint may jeopardize the robot safety, since the danger of collision may be detected at a time when the robot does not have the necessary actuation power to prevent it.

To ensure an adequate look-ahead capability, one should consider both the robot state with respect to the environment and its actuation capabilities. In [3, 20], position, velocity and worst-case stopping time of the robot are considered in order to maintain a minimum clearance between the robot and the environment. In [10], the authors stress the importance of considering the whole robot state in order to guarantee safety and introduce the concept of *Inevitable Collision States* (ICS). In [17], the idea of ICS is exploited for anytime motion planning based on RRT. In [12], workspace obstacles are represented as velocity obstacles over a predefined time horizon, whose length depends on the robot state and actuation capabilities and is such that the robot can always avoid collision with the obstacle by a predefined maneuver. Finally, a forbidden velocity map is defined in [5] as the set of the robot prohibited velocities associated to each obstacle based on the state of the robot and its braking capabilities.

In this work, we take inspiration from the ICS concept to define the notion of *Avoidable Collision State* (ACS), i.e., a state from which the robot can avoid

collision with a certain obstacle. Building on this, we propose an NMPC method for real-time motion generation which enforces collision avoidance through a constraint which essentially requires the robot to be in an ACS at all times. Extensive simulations on a differential-drive robot navigating in both static and dynamic environments clearly show the superiority of the proposed NMPC method with respect to distance-based formulations, especially when the robot must navigate at high speed in environments cluttered with moving obstacles. In particular, it is shown that, thanks to its dynamics-aware nature, the novel constraint works well with relatively short prediction horizons, making real-time performance achievable even on low-cost computational platforms.

The paper is organized as follows. Our navigation problem is formulated in Sect. 2. In Sect. 3 we outline the proposed NMPC approach, while in Sect. 4 we formally define the concept of ACS and derive the associated collision avoidance constraint. Simulation results for a differential-drive robot are presented in Sect. 5. Finally, some concluding remarks are offered in Sect. 6.

2 Problem Formulation

Consider a robotic system whose configuration $\boldsymbol{q}$ (the vector of generalized coordinates) takes values in an n-dimensional configuration space $\mathcal{C}$. The robot is assumed to be subject to k Pfaffian nonholonomic constraints, expressed as $\boldsymbol{A}^T(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{0}$, with $\boldsymbol{A}^T(\boldsymbol{q}) \in \mathbb{R}^{k\times n}$. After some manipulation [19], the dynamics of the robot can be written in state-space format as follows

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \begin{pmatrix} \boldsymbol{G}(\boldsymbol{q})\boldsymbol{\nu} \\ \boldsymbol{M}^{-1}(\boldsymbol{q})(\boldsymbol{E}(\boldsymbol{q})\boldsymbol{u} - \boldsymbol{m}(\boldsymbol{q}, \boldsymbol{\nu})) \end{pmatrix}, \tag{1}$$

with the state defined as $\boldsymbol{x} = \begin{pmatrix} \boldsymbol{q} \\ \boldsymbol{\nu} \end{pmatrix}$, where $\boldsymbol{\nu}$ is the vector of the $m = n - k$ robot pseudovelocities, and the control input $\boldsymbol{u} \in \mathbb{R}^m$ is the vector of generalized forces applied by the m robot actuators. In the right hand side of (1), $\boldsymbol{G}(\boldsymbol{q})$ is a matrix whose columns span $\mathcal{N}(\boldsymbol{A}^T(\boldsymbol{q}))$, and we have set

$$\begin{aligned} \boldsymbol{M}(\boldsymbol{q}) &= \boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{B}(\boldsymbol{q})\boldsymbol{G}(\boldsymbol{q}) \\ \boldsymbol{m}(\boldsymbol{q}, \boldsymbol{\nu}) &= \boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{G}}(\boldsymbol{q})\boldsymbol{\nu} + \boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \\ \boldsymbol{E}(\boldsymbol{q}) &= \boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{S}(\boldsymbol{q}), \end{aligned}$$

being $\boldsymbol{B}(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ the inertia matrix, $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^n$ the vector of velocity and gravitational terms, and $\boldsymbol{S}(\boldsymbol{q}) \in \mathbb{R}^{n\times m}$ the matrix that maps the actuator forces to forces performing work on the generalized coordinates.

We emphasize that the above dynamic model is general and appropriately describes mobile robots as well as robot manipulators, mobile manipulators and so on. In the particular case of an unconstrained (free-flying) robot, we simply have $\boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{I}_n$ and $\boldsymbol{\nu} = \dot{\boldsymbol{q}}$.

The robot moves in an N-dimensional ($N = 2, 3$) workspace $\mathcal{W}$, populated by static and/or dynamic obstacles. We shall denote by $\mathcal{R}(\boldsymbol{q}) \subset \mathcal{W}$ the volume

occupied by the robot at configuration $\boldsymbol{q}$, and by $\mathcal{O}_j(t) \subset \mathcal{W}$ the volume occupied by the j-th obstacle at time t ($j = 1, 2, \dots$).

A navigation task is assigned to the robot in terms of a vector $\boldsymbol{y} \in \mathcal{Y}$, which describes the position of a representative point on the robot and is related to the configuration via a forward kinematic map $\boldsymbol{y} = \boldsymbol{k}(\boldsymbol{q})$. The task is to drive $\boldsymbol{y}$ to a desired position $\boldsymbol{y}_d$ in the workspace.

Our problem is to generate in real-time a motion that:

1. drives the robot from any starting configuration $\boldsymbol{q}_s$ to any configuration realizing the task, i.e., a configuration belonging to $\mathcal{C}_g$, where $\mathcal{C}_g = \{\boldsymbol{q} \in \mathcal{C} : \boldsymbol{k}(\boldsymbol{q}) = \boldsymbol{y}_d\}$;
2. is kinodynamically feasible, in the sense that it is consistent with model (1) and respects existing constraints on both $\boldsymbol{x}$ (e.g., joint and velocity limits) and $\boldsymbol{u}$ (e.g., torque bounds);
3. avoids collisions between the robot and the obstacles, i.e., $\mathcal{R}(\boldsymbol{q}) \cap \mathcal{O}_j(t) = \emptyset$, for all t and $j = 1, 2, \dots$.

As for the information available for solving the problem, it is assumed that the robot is always aware of its own state and that an on-board sensor provides the position and velocity of all obstacles located inside the sensor field of view.

3 The Proposed NMPC Approach

Our approach relies on the use of an NMPC algorithm for real-time motion generation. NMPC solves an OCP at each discrete time instant; to allow efficient numerical solution, each OCP must be reduced to a Nonlinear Program (NLP).

Denote by H the prediction horizon, by δ the sampling interval and by $N = H/\delta$ the number of control intervals in the prediction horizon. We also denote by $\boldsymbol{x}_{k|i}$ and $\boldsymbol{u}_{k|i}$ the predicted robot state and control input at the discrete time instant t_{k+i} computed at t_k. The decision variables of the NLP to be solved at time t_k are the control inputs $\{\boldsymbol{u}_{k|0}, \dots, \boldsymbol{u}_{k|N-1}\}$ and the states $\{\boldsymbol{x}_{k|0}, \dots, \boldsymbol{x}_{k|N}\}$.

Our objective is to drive the task error to zero while using the least possible control effort. Denoting the predicted task error at t_{k+i} as $\boldsymbol{e}_{k|i} = \boldsymbol{y}_d - \boldsymbol{y}_{k|i}$, with $\boldsymbol{y}_{k|i}$ the position of the representative point at t_{k+i}, the running cost can then be expressed as:

$$V_{k|i}(\boldsymbol{x}_{k|i}, \boldsymbol{u}_{k|i}) = \boldsymbol{e}_{k|i}^T \boldsymbol{Q} \boldsymbol{e}_{k|i} + \dot{\boldsymbol{y}}_{k|i}^T \boldsymbol{P} \dot{\boldsymbol{y}}_{k|i} + \boldsymbol{u}_{k|i}^T \boldsymbol{R}\, \boldsymbol{u}_{k|i},$$

while the terminal cost will be

$$V_{k|N}(\boldsymbol{x}_{k|N}) = \boldsymbol{e}_{k|N}^T \boldsymbol{Q}_N \boldsymbol{e}_{k|N} + \dot{\boldsymbol{y}}_{k|N}^T \boldsymbol{P}_N \dot{\boldsymbol{y}}_{k|N}.$$

Here, $\boldsymbol{Q}$, $\boldsymbol{P}$ and $\boldsymbol{R}$ are positive definite weighting matrices for the task error, the velocity of the representative point and the control effort throughout the prediction horizon, while $\boldsymbol{Q}_N$ and $\boldsymbol{P}_N$ are positive definite weighting matrices for the task error and the velocity of the representative point at the final time instant.

The NLP to be solved at time instant t_k is then:

$$\min_{\substack{\boldsymbol{u}_{k|0},\ldots,\boldsymbol{u}_{k|N-1},\\ \boldsymbol{x}_{k|0},\ldots,\boldsymbol{x}_{k|N}}} \sum_{i=0}^{N-1} V_{k|i}(\boldsymbol{x}_{k|i},\boldsymbol{u}_{k|i}) + V_{k|N}(\boldsymbol{x}_{k|N})$$

subject to:

$$\boldsymbol{x}_{k|0} - \boldsymbol{x}_k = 0$$
$$\boldsymbol{x}_{k|i+1} - \boldsymbol{F}(\boldsymbol{x}_{k|i},\boldsymbol{u}_{k|i}) = 0, \quad i = 0,\ldots,N-1$$
$$\boldsymbol{x}_{\min} \leq \boldsymbol{x}_{k|i} \leq \boldsymbol{x}_{\max}, \quad i = 0,\ldots,N$$
$$\boldsymbol{u}_{\min} \leq \boldsymbol{u}_{k|i} \leq \boldsymbol{u}_{\max}, \quad i = 0,\ldots,N-1$$
collision avoidance constraints at $t_k,\ldots,t_{k+N}$

where $\boldsymbol{x}_k$ represents the current robot state, $\boldsymbol{F}(\cdot,\cdot)$ represents the right-hand-side of a discrete state-space model of the robot obtained via numerical integration (typically under the assumption of piecewise-constant control inputs), while $\boldsymbol{x}_{\min}, \boldsymbol{x}_{\max}$ and $\boldsymbol{u}_{\min}, \boldsymbol{u}_{\max}$ are respectively the lower/upper bounds on the state variables and on the control inputs.

As for the collision avoidance constraint, which is the main contribution of this work, it is discussed in full detail in the next section.

4 Collision Avoidance

The proposed collision avoidance constraint hinges upon the notion of Avoidable Collision State (ACS), i.e., a robot state from which it is possible to avoid collisions. In this section we will first give a formal definition of what an ACS is, and then derive the corresponding collision avoidance constraint. Although all computations are presented in a 2-dimensional workspace, the extension to the 3-dimensional case (e.g., for application to UAVs) is straightforward.

4.1 Preliminaries

The notion of ACS is obstacle-specific, in the sense that it characterizes the possibility for the robot to avoid a certain obstacle. In view of this, we first need to identify the obstacles for which there is an actual danger of collision given the current state of the robot.

We are going to use bounding spheres for both the robot and the obstacles. In particular, we take the smallest sphere that contains the robot volume $\mathcal{R}$, and denote its radius by ρ and its center by R. The position vector of R in the world frame is denoted by $\boldsymbol{r}$ and is related to the robot configuration via a forward kinematic map $\boldsymbol{r} = \boldsymbol{\sigma}(\boldsymbol{q})$. For its velocity we have: $\dot{\boldsymbol{r}} = \boldsymbol{J}(\boldsymbol{q})\boldsymbol{\nu}$ where $\boldsymbol{J}(\boldsymbol{q}) = \partial\boldsymbol{\sigma}(\boldsymbol{q})/\partial\boldsymbol{q}\boldsymbol{G}(\boldsymbol{q})$. Similarly, we use a sphere of radius ρ_j to envelop the generic obstacle $\mathcal{O}_j$, denoting its center by O_j, the corresponding position vector by $\boldsymbol{o}_j$ and its velocity by $\dot{\boldsymbol{o}}_j$. Finally, let $\boldsymbol{n}_j$ be the unit vector pointing from R to O_j, and $\boldsymbol{r}_j = \boldsymbol{r} - \boldsymbol{o}_j$ be the relative position of R with respect to O_j, so that the corresponding relative velocity is $\dot{\boldsymbol{r}}_j$. Refer to Fig. 1 for illustration.

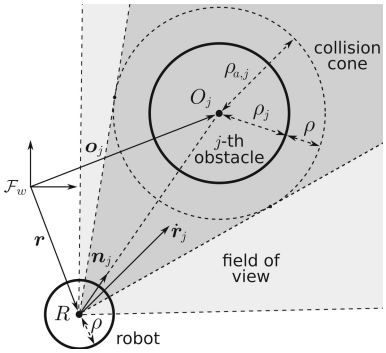

Fig. 1. Obstacle $\mathcal{O}_j$ is considered dangerous for the robot at a certain time if the relative velocity $\dot{\boldsymbol{r}}_j$ of the robot with respect to the obstacle lies inside the collision cone.

Now, augment the obstacle sphere by the radius of the robot sphere, denoting by $\rho_{a,j} = \rho_j + \rho$ the total radius. The *collision cone* is the cone defined by R and the tangents from R to the augmented obstacle. Obstacle $\mathcal{O}_j$ is *dangerous* at time t if $\dot{\boldsymbol{r}}_j(t)$ lies inside the collision cone (see Fig. 1). Simple geometrical arguments lead to the following condition for an obstacle to be dangerous:

$$h(\boldsymbol{x}, \boldsymbol{\xi}_j) = \boldsymbol{n}_j^T \frac{\dot{\boldsymbol{r}}_j}{\|\dot{\boldsymbol{r}}_j\|} - \frac{\sqrt{\|\boldsymbol{r}_j\|^2 - \rho_{a,j}^2}}{\|\boldsymbol{r}_j\|} \geq 0 \tag{2}$$

where $\boldsymbol{\xi}_j = \left(\boldsymbol{o}_j, \dot{\boldsymbol{o}}_j\right)$ is the j-th obstacle state.

4.2 Avoidable Collision States

By definition, to avoid an obstacle which is non-dangerous at time t the robot simply needs to keep its course. Therefore, we only need to characterize the possibility of avoiding obstacles that are dangerous at t. In particular, we will say that the robot is in an *Avoidable Collision State* (ACS) with respect to a dangerous obstacle if there exists at least one trajectory that originates from the current state, is kinodynamically feasible and avoids collision with the obstacle.

In principle, to conclude that a state is an ACS we should check all feasible trajectories emanating from it, until we find at least one that avoids collision. However, such a potentially exhaustive study is incompatible with a real-time application. We shall therefore look at one specific motion and ask ourselves if the robot can avoid collision during that motion. If the answer is positive, then it can be concluded that the current state is certainly an ACS.

In particular, in the presence of an obstacle $\mathcal{O}_j$ which moves along a given direction with constant speed and is dangerous at time t, we consider the robot moving in such a way that at each time instant $t' \geq t$ its relative velocity with respect to the obstacle, $\dot{\boldsymbol{r}}_j(t')$, projected on the original direction of collision, $\boldsymbol{n}_j(t)$, decreases with a constant rate α.

During this motion there will be eventually a time instant, denoted by t_v, in which the projection of the robot relative velocity on the original direction of collision will be zero, i.e. $\boldsymbol{n}_j^T(t)\dot{\boldsymbol{r}}_j(t_v) = 0$. A sufficient condition for this motion to be collision-free is:

$$\boldsymbol{n}_j^T(t)(\boldsymbol{o}_j(t') - \boldsymbol{r}(t')) \geq \rho_{a,j}, \quad \forall t' \in [t, t_v].$$

Denoting by $\gamma(t)$ the robot-obstacle clearance and considering that at time t the relation $\boldsymbol{n}_j^T(t)(\boldsymbol{o}_j(t) - \boldsymbol{r}(t)) = \gamma(t) + \rho_{a,j}$ holds, after simple substitution of $\rho_{a,j}$, the condition for collision-free motion becomes:

$$\boldsymbol{n}_j^T(t)(\boldsymbol{r}_j(t') - \boldsymbol{r}_j(t)) \leq \gamma(t), \quad \forall t' \in [t, t_v]. \tag{3}$$

Note that inequality (3) defines an admissible half-plane for the relative position of R with respect to O_j in which the robot has to remain for all $t' \in [t, t_v]$ (see Fig. 2).

In order to ensure that the considered motion can be implemented by the robot we will investigate whether the required deceleration of R, i.e. $\ddot{\boldsymbol{r}} = \boldsymbol{n}_j(t)\alpha$, projected on the robot actuation space lies within the actuation limits.

Throughout the considered motion, the position of R, projected on the original direction of collision $\boldsymbol{n}_j(t)$ is described at time $t' \geq t$ as

$$\boldsymbol{n}_j^T(t)\boldsymbol{r}(t') = \boldsymbol{n}_j^T(t)\boldsymbol{r}(t) + \boldsymbol{n}_j^T(t)\dot{\boldsymbol{r}}(t)(t' - t) + \frac{1}{2}\alpha(t' - t)^2 \tag{4}$$

and its projected velocity as

$$\boldsymbol{n}_j^T(t)\dot{\boldsymbol{r}}(t') = \boldsymbol{n}_j^T(t)\dot{\boldsymbol{r}}(t) + \alpha(t' - t). \tag{5}$$

As for the obstacle, the position of O_j at $t' \geq t$ projected on the original direction of collision is

$$\boldsymbol{n}_j^T(t)\boldsymbol{o}_j(t') = \boldsymbol{n}_j^T(t)\boldsymbol{o}_j(t) + \boldsymbol{n}_j^T(t)\dot{\boldsymbol{o}}_j(t' - t). \tag{6}$$

By solving the system of equations (3) for the equality, (4), (5) and (6), with $t' = t_v$, we obtain the minimum required deceleration of the robot in the original direction of collision:

$$\alpha = -\frac{1}{2}\frac{\left(\boldsymbol{n}_j^T(\dot{\boldsymbol{o}}_j - \dot{\boldsymbol{r}}(t))\right)^2}{\gamma(t)}.$$

Note that α depends on both the robot and the j-th obstacle state. Considering that the acceleration of R is:

$$\ddot{\boldsymbol{r}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{\nu}} + \dot{\boldsymbol{J}}(\boldsymbol{q})\boldsymbol{\nu} \tag{7}$$

and that from (1) we can express $\dot{\boldsymbol{\nu}}$ as:

$$\dot{\boldsymbol{\nu}} = \boldsymbol{M}^{-1}(\boldsymbol{q})(\boldsymbol{E}(\boldsymbol{q})\boldsymbol{u} - \boldsymbol{m}(\boldsymbol{q}, \boldsymbol{\nu})), \tag{8}$$

we can obtain the required control inputs for this deceleration by substituting $\ddot{\boldsymbol{r}} = \boldsymbol{n}_j(t)\alpha$ and (8) in (7). Using the Moore-Penrose pseudoinverse, we get

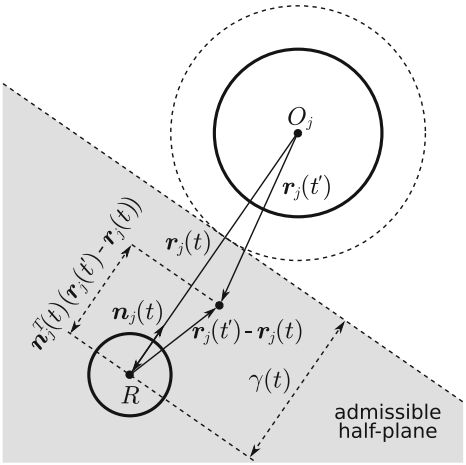

Fig. 2. The admissible half-plane for the relative position of robot R with respect to the obstacle O_j at time t', $t' \in [t, t_v]$.

the minimum norm control inputs needed in order to apply the deceleration $\ddot{\boldsymbol{r}} = \boldsymbol{n}_j(t)\alpha$ to R:

$$\boldsymbol{u}_\alpha(\boldsymbol{x}, \boldsymbol{\xi}_j) = \left(\boldsymbol{J}(\boldsymbol{q})\boldsymbol{M}^{-1}(\boldsymbol{q})\boldsymbol{E}(\boldsymbol{q})\right)^\dagger \boldsymbol{\beta}(\boldsymbol{x}, \boldsymbol{\xi}_j), \tag{9}$$

where

$$\boldsymbol{\beta}(\boldsymbol{x}, \boldsymbol{\xi}_j) = \boldsymbol{n}_j(t)\alpha - \dot{\boldsymbol{J}}(\boldsymbol{q})\boldsymbol{\nu} + \boldsymbol{J}(\boldsymbol{q})\boldsymbol{M}^{-1}(\boldsymbol{q})\boldsymbol{m}(\boldsymbol{q}, \boldsymbol{\nu}).$$

So the motion is kinodynamically feasible if the following condition is satisfied:

$$\boldsymbol{u}_{\min} \leq \boldsymbol{u}_\alpha(\boldsymbol{x}, \boldsymbol{\xi}_j) \leq \boldsymbol{u}_{\max}. \tag{10}$$

This condition can be used as a constraint in order to guarantee that the robot is always at an ACS.

Note that the ACS property for a state is strongly related to its being *safe* according to [10], i.e., not being an Inevitable Collision State (ICS). However, the two properties differ in two aspects:

- The number of obstacles considered by the property. An ICS is defined with respect to all the obstacles (or at least all the obstacles visible by the robot), a practice that obviously increases the computational time, while the ACS property is defined with respect to a specific (dangerous) obstacle;
- The way in which the property is established. To prove that a state is not ICS, in principle one has to search the whole control input set (or a finite subset [17]) to find a collision-free motion. On the other hand, to characterize a state as ACS we only look at the relative velocity of the robot with respect to the dangerous obstacle, and specifically investigate whether it is possible to stop motion along the robot-obstacle direction before collision.

4.3 Use of the ACS Condition in the NLP

Note that constraint (10) is suitable for enforcing collision avoidance since every robot motion that leads to collision will violate the constraint before the collision occurs. So we will use (10) in the proposed NLP applying it for each considered obstacle, for each time instant throughout the prediction horizon. In order to ensure that the constraint is inactive if non-dangerous obstacles are considered and to avoid using *if* statements, we multiply $\boldsymbol{u}_\alpha(\boldsymbol{x}, \boldsymbol{\xi}_j)$ as given by (9) by the sigmoid function $g(h(\boldsymbol{x}, \boldsymbol{\xi}_j)) = 1/(1 + e^{-\lambda h(\boldsymbol{x}, \boldsymbol{\xi}_j)})$, with λ being a constant value that tunes the steepness of the sigmoid function, getting:

$$\boldsymbol{u}_b(\boldsymbol{x}, \boldsymbol{\xi}_j) = g(h(\boldsymbol{x}, \boldsymbol{\xi}_j)) \left(\boldsymbol{J}(\boldsymbol{q})\boldsymbol{M}^{-1}(\boldsymbol{q})\boldsymbol{E}(\boldsymbol{q})\right)^\dagger \boldsymbol{\beta}(\boldsymbol{x}, \boldsymbol{\xi}_j).$$

So the constraint that will be applied in the NLP is:

$$\boldsymbol{u}_{\min} \leq \boldsymbol{u}_b(\boldsymbol{x}_{k|i}, \boldsymbol{\xi}_{j,k|i}) \leq \boldsymbol{u}_{\max}, \quad i = 0, \ldots, N, \quad j = 1, \ldots, n_o, \tag{11}$$

where $\boldsymbol{\xi}_{j,k|i}$ is the state of the j-th obstacle at t_{k+i} and n_o is the number of the considered obstacles in the NMPC.

5 Simulations

In order to show the effectiveness of the proposed method, we conducted a series of simulations in various static and dynamic environments. All the simulations were implemented in MATLAB on an Intel Core i9-9900K CPU at 3.60 GHz. For the numerical solution of the NLP required by our NMPC, we used the RTI scheme [6] implemented within the ACADO Toolkit [13].

The robotic platform used for our simulations is the differential-drive robot shown in Fig. 3, having length $l = 0.60$ m and width $b = 0.30$ m. The Center of Mass (CoM) is located at the geometric center of the vehicle, which is also the center R of the robot bounding sphere, whose radius is $\rho = \sqrt{l^2/4 + b^2/4}$. The rear wheels are located at a distance $d = 0.25$ m from the CoM, with a caster wheel in front for static balance. The vehicle mass and moment of inertia are respectively 50.0 kg and 1.41 kg·m^2, making this a rather heavy vehicle. The torque bounds for the wheel actuators are set to 2.5 Nm. The configuration vector of this robot is $\boldsymbol{q} = (x, y, \theta)$ and its dynamic model is that of a unicycle (e.g., see [19, Sect. 11.4]). The robot representative point for the navigation task is also placed at R. It is assumed that the robot is equipped with a line-of-sight sensor (such as a laser rangefinder) with an infinite field of view.

We wish to compare the proposed NMPC method, which relies upon a dynamics-aware (DA) collision avoidance constraint, with an NMPC that uses a purely distance-based (DB) collision avoidance constraint. In both methods, the NLP formulation is the same, apart from the collision avoidance constraint which in DB takes the form:

$$\left\|\boldsymbol{r}_{k|i} - \boldsymbol{o}_{j,k|i}\right\| \geq \rho_{a,j}, \quad i = 0, \ldots, N, \quad j = 1, \ldots, n_o,$$

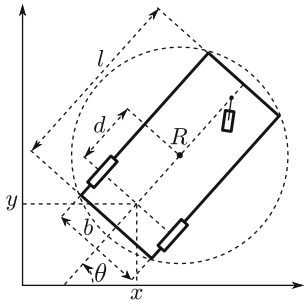

Fig. 3. The differential-drive robot used in simulation with its bounding sphere.

where $\boldsymbol{r}_{k|i}$ and $\boldsymbol{o}_{j,k|i}$ are the position of R and O_j at t_{k+i}, respectively. In both NMPCs we consider the $n_o = 5$ closest visible obstacles. The sampling interval for real-time control is $\delta = 31$ ms in both cases.

The two methods have been tested in 50 different environments (25 static and 25 dynamic), using 3 different values for the maximum driving velocity $v_{\max}$. The maximum steering velocity is always set at $\omega_{\max} = 20/3\, v_{\max}$ rad/s. The prediction horizons were chosen as large as possible to allow real-time performance for the two methods. In particular, for the DB method the prediction horizon is $H = 0.992$ s while for the DA method it is $H = 0.93$ s; in fact the DB method requires slightly less computations than the DA method.

We assess the performance of the two methods by taking into account the success rate as well as the quality of the motion according to the following criteria: (1) time t_g needed for the robot to reach the goal, (2) control effort $J_\tau = \int_0^{t_g} \|\boldsymbol{\tau}\|^2 dt$, (3) length l_p of the resulting path, (4) duration of the longest iteration $\delta_{\max}$ and (5) average iteration time $\bar{\delta}$.

Video clips of representative simulation results are shown in the accompanying video, also available at https://youtu.be/LS57E8jGoJk.

5.1 Static Environments

In the first group of simulations, the robot moves in 25 environments, each occupied by 10 static obstacles (e.g., see Fig. 4). The robot starts from $\boldsymbol{q}_s = (2, 2, \pi/3)$ and must reach $\boldsymbol{y}_g = (16, 15)$. Table 1 (top) summarizes the results of the DA and DB methods averaged over the 25 environments, for increasing values of the maximum driving velocity $v_{\max}$.

For lower $v_{\max}$ the two methods behave similarly having also the same success rate. An illustrative example of the robot motion with $v_{\max} = 0.9$ m/s is given in simulation 1 of the accompanying video. However, when $v_{\max} = 1.2$ m/s the success rate of the DB method drops to 88% while the DA method is hardly affected by the increase in the maximum velocity. Note that in this case, the

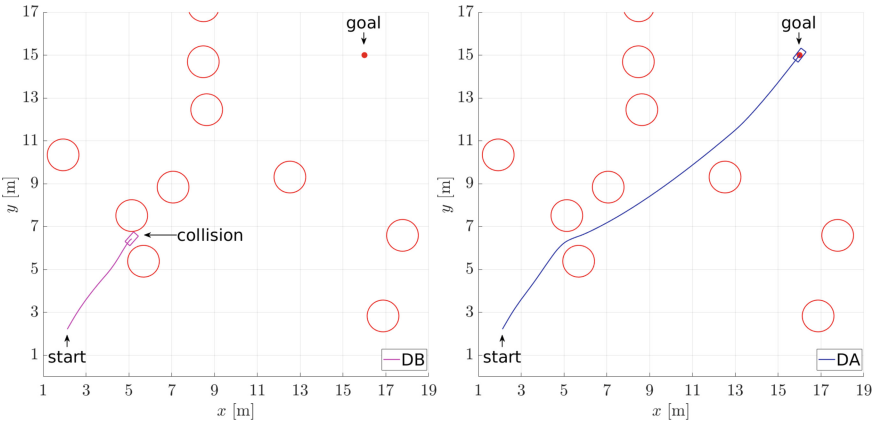

Fig. 4. Motions generated by the two methods in one of the static environments, with $v_{\max} = 1.2$ m/s. The DB method (left) cannot avoid collision with an obstacle, whereas DA (right) goes safely through the narrow passage and successfully reaches the goal. See also simulation 2 in the accompanying video.

Table 1. Averaged results over 25 environments for the proposed dynamics-aware (DA) method vs the distance-based (DB) method. Top: static environments, bottom: dynamic environments.

Static Environments						
	DB			DA		
$v_{\max}$ [m/s]; t_s [s]	0.9; 0.93	1.1; 1.116	1.2; 1.209	0.9; 0.93	1.1; 1.116	1.2; 1.209
success rate (%)	96	96	88	96	96	92
t_g [s]	27.16	24.29	25.07	26.63	24.76	23.98
J_τ [10^4 N^{2}m^{2}s]	1164.63	1495.56	1763.30	1046.58	1397.92	1601.14
l_p [m]	19.56	20.12	20.85	19.51	19.96	20.48
$\delta_{\max}$ [ms]	25.21	25.15	25.46	26.84	27.55	27.11
$\bar{\delta}$ [ms]	15.31	14.26	13.67	16.40	15.96	15.50

Dynamic Environments						
	DB			DA		
$v_{\max}$ [m/s]; t_s [s]	0.9; 0.93	1.1; 1.116	1.2; 1.209	0.9; 0.93	1.1; 1.116	1.2; 1.209
v_o [m/s]	0.45	0.55	0.6	0.45	0.55	0.6
success rate (%)	72	64	40	80	84	80
t_g [s]	29.65	25.52	24.38	30.61	29.78	27.19
J_τ [10^4 N^{2}m^{2}s]	1766.29	2001.42	2009.93	1625.94	2171.06	2510.07
l_p [m]	20.37	20.69	20.77	20.62	21.83	21.31
$\delta_{\max}$ [ms]	26.16	25.65	26.14	28.18	28.56	28.85
$\bar{\delta}$ [ms]	15.47	14.63	14.19	16.87	16.64	16.40

stopping time[1] t_s is significantly larger than the prediction horizon. This shows that under these conditions, unlike DA, a robot navigating with the DB method would require either a longer prediction horizon or greater braking capabilities in order to navigate safely. A representative example of the behaviour of the two methods is shown in Fig. 4 (also see simulation 2 in the accompanying video). In particular, it shows the trajectories generated by the two methods in one of the static environments with $v_{\max} = 1.2$ m/s. The DB method (left) runs into a collision while passing through a narrow passage, because the robot has accumulated too much speed in the initial part of the motion and avoiding the imminent collision would require more than the available torque (this corresponds to an unfeasibility for the NMPC). On the other hand, the DA method (right) starts the avoidance maneuver earlier thanks to the dynamics-aware constraint, which can detect a violation before the distance-based constraint does; therefore, DA is able to go through the narrow passage and reach the goal safely.

Looking at the other performance criteria of Table 1, one can observe that on the average the DA method produces trajectories that are slightly shorter and less energy-consuming than the DB method. As a counterpart, the DB method shows a slightly reduced duration of the longest iteration mainly due to the simplicity of the collision avoidance constraint.

5.2 Dynamic Environments

The second group of simulations take place in 25 dynamic environments, each occupied by 10 static and 10 moving obstacles. The moving obstacles are zigzagging at a speed $v_o = v_{\max}/2$, with direction changes of $\pi/3$ (toward the robot) after traveling a distance of 4.9 m. The results are presented in Table 1 (bottom).

The success rate of both methods is affected by the presence of the moving obstacles. However, the DA method is much more effective than the DB method, with success rates over 80% while DB goes as low as 40%.

Once again, for the lowest $v_{\max}$ the DB method has its higher success rate, yet lower than the DA method. In simulation 3 of the accompanying video we show the robot moving in a dynamic environment with $v_{\max} = 0.9$ m/s. Nevertheless, the success rate of the DB method decreases as $v_{\max}$ increases and plummets to 40% for $v_{\max} = 1.2$ m/s, revealing that the presence of moving obstacles in combination with fast robot motion affects significantly its performance. This is due to the fact that the maneuvers needed for collision avoidance become now more demanding, since a worst-case scenario may require the robot not only to stop, but also to accelerate in the opposite direction in order to avoid an imminent collision. Thus, by the time an imminent collision is detected in the DB case, there may not be enough actuation capability left to avoid the obstacle explaining why the DB method has such a low performance even in the case of the lowest maximum velocity. On the other hand, the DA method guarantees

[1] By stopping time t_s we denote the minimum time that a robot traveling on a straight line with speed $v_{\max}$ needs in order to stop.

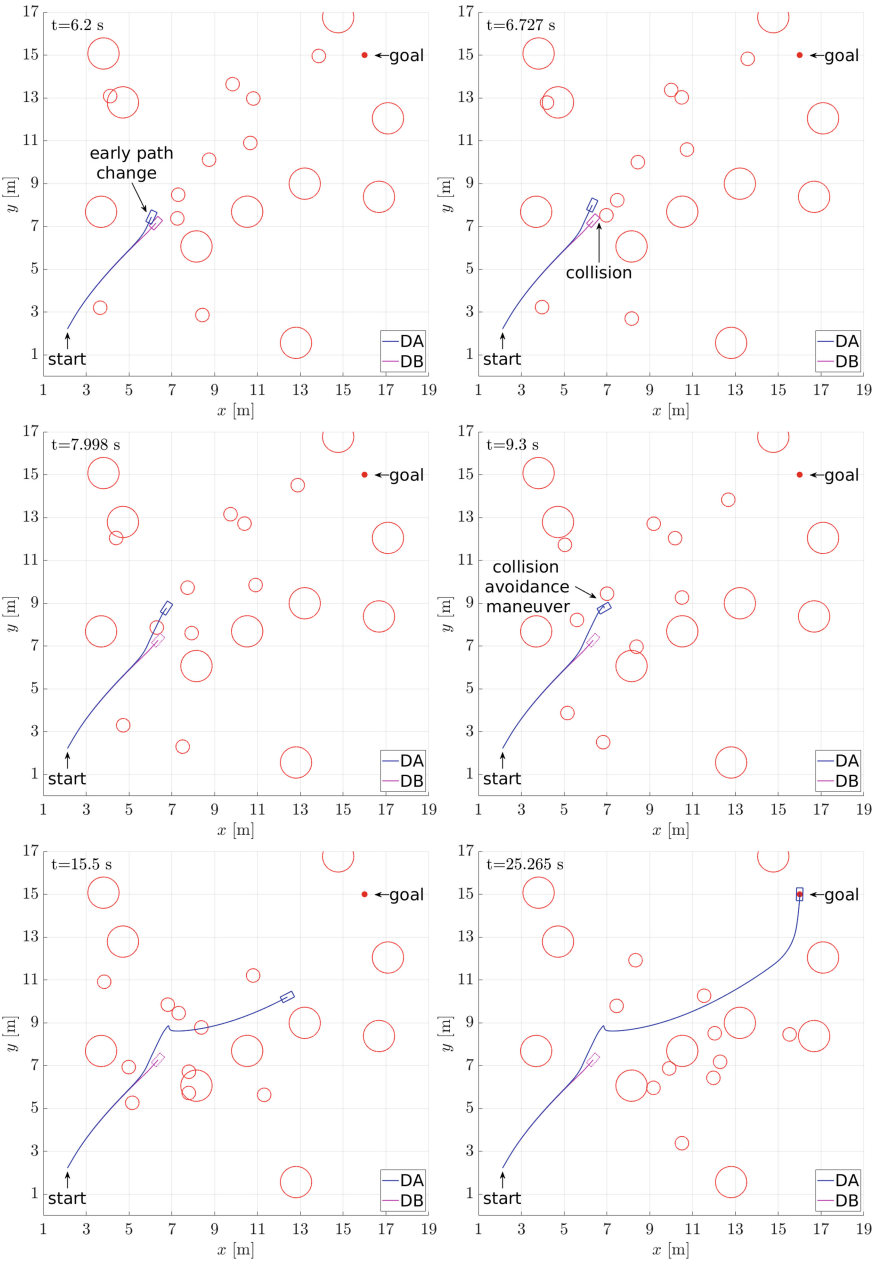

Fig. 5. Snapshots of the motion generated by the two methods in one of the dynamic environments, with $v_{\max} = 1.2$ m/s. As in the static environment, the DB method cannot avoid collision with an obstacle, whereas the DA method safely navigates to the goal. See also simulation 4 in the accompanying video.

that the robot is at all times in an ACS. As a result, the robot is able to veer off collision paths earlier.

For illustration, Fig. 5 (see also simulation 4 in the accompanying video) shows the behavior of the two methods in one of the dynamic environments through a series of snapshots. Again, we have here $v_{\max} = 1.2$ m/s. As in the static environment case, the DB method is not aware of the robot dynamic state, and therefore cannot prevent the collision with the moving obstacle. On the other hand, the DA method effectively reaches the goal avoiding nearby obstacles; note in particular how the early reaction in the presence of a moving obstacle averted the impending collision, and its elaborate avoidance maneuver in front of a moving obstacle combining a reverse motion and a quick reorientation.

In an attempt to further assess the effectiveness of the proposed method, we tested the performance of the two methods in a more cluttered version of the dynamic environment, increasing the number of moving obstacles to 15. The obstacles are zigzagging at a speed of 0.4 m/s, with direction changes of $\pi/3$ (toward the robot) after traveling a distance of 4.9 m. The robot maximum velocity is $v_{\max} = 1.2$ m/s. This simulation, whose results are only shown in simulation 5 of the accompanying video for compactness, confirms that the robot controlled by the dynamics-aware NMPC method safely reaches its goal even in this challenging environment, while the distance-based NMPC fails.

6 Conclusion

We have presented a novel real-time NMPC for robot navigation in environments populated by static and/or moving obstacles. Inspired by the concept of inevitable collision state, we defined the notion of ACS (avoidable collision state) and, based on this, formulated a hard constraint on the robot state guaranteeing that it can execute a collision avoidance trajectory in the presence of a dangerous obstacle. The method was compared with an NMPC that considers a purely distance-based collision avoidance constraint. The comparative simulations showed that the proposed method is generally more effective, especially when the robot navigates at high speed in cluttered dynamic environments.

Future work will be aimed at testing the proposed method on multi-body mobile robots (namely, mobile manipulators), which are already included in the class of robots considered in this paper. Another interesting possibility is to extend the proposed approach to flying robots.

References

1. Betts, J.: Survey of numerical methods for trajectory optimization. J. Guidance Control Dyn. **21**, 193–207 (1998)
2. Bonalli, R., Cauligi, A., Bylard, A., Pavone, M.: GuSTO: Guaranteed Sequential Trajectory optimization via Sequential Convex Programming. In: 2019 International Conference on Robotics and Automation, pp. 6741–6747 (2019)
3. Buizza Avanzini, G., Zanchettin, A.M., Rocco, P.: Constrained model predictive control for mobile robotic manipulators. Robotica, 1–20 (2017)

4. Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press (2005)
5. Damas, B., Santos-Victor, J.: Avoiding moving obstacles: the forbidden velocity map. In: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4393–4398 (2009)
6. Diehl, M., Bock, H., Schlöder, J.P., Findeisen, R., Nagy, Z., Allgöwer, F.: Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. J. Process Control **12**(4), 577–585 (2002)
7. Febbo, H., Jayakumar, P., Stein, J.L., Ersal, T.: Real-time trajectory planning for automated vehicle safety and performance in dynamic environments. J. Auton. Veh. Syst. 1(4) (2021)
8. Fiorini, P., Shiller, Z.: Motion planning in dynamic environments using velocity obstacles. Int. J. Rob. Res. **17**(7), 760–772 (1998)
9. Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. IEEE Rob. Autom. Mag. **4**(1), 23–33 (1997)
10. Fraichard, T., Asama, H.: Inevitable collision states. a step towards safer robots? In: 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 1, pp. 388–393 (2003)
11. Frasch, J.V., Gray, A., Zanon, M., Ferreau, H.J., Sager, S., Borrelli, F., Diehl, M.: An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles. In: 2013 European Control Conference, pp. 4136–4141 (2013)
12. Gal, O., Shiller, Z., Rimon, E.: Efficient and safe on-line motion planning in dynamic environments. In: 2009 IEEE International Conference on Robotics and Automation, pp. 88–93 (2009)
13. Houska, B., Ferreau, H.J., Diehl, M.: An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. Automatica **47**(10), 2279–2285 (2011)
14. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. In: 1985 IEEE International Conference on Robotics and Automation, vol. 2, pp. 500–505 (1985)
15. LaValle, S.M., James, J., Kuffner, J.: Randomized kinodynamic planning. Int. J. Rob. Res. **20**(5), 378–400 (2001)
16. Liniger, A., Domahidi, A., Morari, M.: Optimization-based autonomous racing of 1:43 scale RC cars. Optimal Control Appl. Methods **36**(5), 628–647 (2015)
17. Petti, S., Fraichard, T.: Safe motion planning in dynamic environments. In: 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2210–2215 (2005)
18. Sathya, A., Sopasakis, P., Van Parys, R., Themelis, A., Pipeleers, G., Patrinos, P.: Embedded nonlinear model predictive control for obstacle avoidance using PANOC. In: 2018 European Control Conference, pp. 1523–1528 (2018)
19. Siciliano, B., Sciavicco, L., Villani, L., Oriolo, G.: Robotics: Modelling, Planning and Control. Springer (2009)
20. Zanchettin, A.M., Ceriani, N.M., Rocco, P., Ding, H., Matthias, B.: Safety in human-robot collaborative manufacturing environments: metrics and control. IEEE Trans. Autom. Sci. Eng. **13**(2), 882–893 (2016)
21. Zhang, X., Liniger, A., Borrelli, F.: Optimization-based collision avoidance. ArXiv abs/1711.03449 (2017)