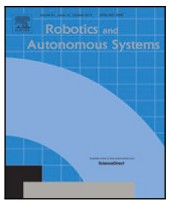
Robotics and Autonomous Systems 172 (2024) 104582

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot



Dynamics-aware navigation among moving obstacles with application to ground and flying robots

Spyridon G. Tarantos [a], Tommaso Belvedere [b], Giuseppe Oriolo [b,*]

[a] Center of AI & Robotics (CAIR) and Engineering Division, New York University Abu Dhabi, Abu Dhabi, United Arab Emirates
[b] Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, Via Ariosto 25, 00185 Rome, Italy

ARTICLE INFO

Keywords:
Robot navigation
Mobile robots
Collision avoidance
NMPC

ABSTRACT

We present a novel method for navigation of mobile robots in challenging dynamic environments. The method, which is based on Nonlinear Model Predictive Control (NMPC), hinges upon a specially devised constraint for dynamics-aware collision avoidance. In particular, the constraint builds on the notion of *avoidable collision state*, taking into account the robot actuation capabilities in addition to the robot–obstacle relative distance and velocity. The proposed approach is applied to both ground and flying robots and tested in a variety of static and dynamic environments. Comparative simulations with an NMPC using a purely distance-based collision avoidance constraint confirm the superiority of the dynamics-aware version, especially for high-speed navigation among moving obstacles. Moreover, the results indicate that the method can work with relatively short prediction horizons and is therefore amenable to real-time implementation.

1. Introduction

The ability of a robot to navigate safely in a dynamic environment is crucial in both service and field applications and makes real-time motion planning necessary. In the literature, there are various motion planning methods appropriate for robot navigation, among them the Artificial Potential Fields (APFs) [1], methods like the *velocity obstacles* [2] or the *dynamic window approach* [3], as well as online versions of sampling-based motion planners [4].

Recently, NMPC has become a rather attractive approach for mobile robot navigation in environments populated by static and/or moving obstacles. Solving an Optimal Control Problem (OCP) at each control cycle and by using the robot dynamic model as prediction model along with appropriate state and input constraints, NMPC can generate kinodynamically feasible motions that adapt to the changes of the environments, while since the OCP is defined over a time horizon, the resulting motion have also a notion of look-ahead. Finally, the constant increase of the available computational capabilities, the development of tailored algorithms like the Real-Time Iteration [5] (RTI) and of appropriate software for embedded optimization [6], enable real-time performance for NMPC.

In order to ensure that the generated motion will be collision-free, one has to include in the NMPC appropriate collision avoidance constraints. Typically, such a constraint aims to ensure that the volume of the robot $\mathcal{R}$ and the obstacle $\mathcal{O}$ do not overlap, i.e.,

$$\mathcal{R} \cap \mathcal{O} = \emptyset. \tag{1}$$

However, in principle, condition (1) is not appropriate for numerical optimization [7] and one has to resort to appropriate reformulations and/or approximations.

If the considered environment is structured, one can leverage its particular features and simply bind the robot state within appropriate limits creating a collision-free corridor [8,9]. However, this is not always possible, and explicit collision avoidance constraints have to be included. In [7] the authors propose a smooth nonlinear version of condition (1), appropriate for numerical optimization, while in [10] the proposed constraint considers obstacles described by general non-convex sets. In [11,12] *signed distance fields* are used in order to generate collision-free trajectories, while in [13–16] condition (1) is approximated by enveloping the volumes of the robot and the obstacles with bounding geometries.

It should be noted that the aforementioned collision avoidance constraints are based on purely distance information neglecting completely the dynamic state of the robot. When employed in an NMPC, the effectiveness of a distance-based collision avoidance constraint strongly depends on the length of the prediction horizon, in relation to the robot actuation capabilities. Intuitively, the longer the prediction horizon, the earlier an imminent collision can be detected and averted without significant actuation effort, that in some cases can become prohibitive. Nevertheless, the length of the prediction horizon is limited by the real-time performance. In practice, the maximum achievable prediction

* Corresponding author.
E-mail addresses: spyridon.tarantos@nyu.edu (S.G. Tarantos), belvedere@diag.uniroma1.it (T. Belvedere), oriolo@diag.uniroma1.it (G. Oriolo).

https://doi.org/10.1016/j.robot.2023.104582

Available online 17 November 2023

horizon on typical robot processing platforms can become relatively short, especially in cases of high-speed navigation which requires not only the use of the robot full dynamics but also high control frequency. In such cases, the use of a purely distance-based constraint puts the robot safety in jeopardy, since an imminent collision may be detected at a time when the robot does not have the necessary actuation power to prevent it.

In principle, one can attribute look-ahead capabilities to the collision avoidance constraint by considering both the whole robot state with respect to the obstacles and the robot actuation capabilities. This idea has been already exploited in some classical motion planning methods (e.g., [2,3,17,18]). The importance of considering the whole robot state in order to guarantee safety has been also stressed in [19] where the concept of *Inevitable Collision States* (ICS) was first introduced. In [20], the idea of ICS is exploited by an anytime motion planning approach based on Rapidly-exploring Random Trees [21]. Nevertheless, these methods require a search in the input and state space, which is not consistent with the nature of the NMPC. A collision avoidance constraint based on the robot–obstacle relative state and the worst-case stopping time of the robot has been applied to an MPC in [22].

Inspired by the concept of ICS, we 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-based method for robot navigation, which drives the robot to an assigned goal. To enforce kinodynamic feasibility of the resulting motion, we consider as prediction model the full dynamic model of the robot, equipped with state and input constraints that reflect its hardware limitations. Collision avoidance is guaranteed by including a constraint that requires the robot to be in an ACS at all times. The resulting dynamics-aware navigation approach is improved by the integration of velocity fields that prevent the method from getting trapped in local minima by guiding the robot around obstacles.

Although the proposed navigation method can in principle be applied to any robotic platform, in this work we showcase its performance through the application to a wheeled robot and a flying robot. Wheeled robots are the most common mobile robotic platforms; moreover, they are typically subject to nonholonomic constraints, which essentially reflect on their dynamic models being underactuated. Therefore, they represent an important benchmark for high-speed navigation techniques in general, and for our dynamics-aware method in particular. On the other hand, validation on flying robots is also challenging in view of their complex dynamics, intrinsic 3D nature and agile maneuvrability.

In the simulations, the performance of the proposed method will be compared with a version of the NMPC in which the collision avoidance constraint is purely distance-based, confirming the superiority of the dynamics-aware formulation for high-speed navigation among moving obstacles.

The proposed navigation method is the evolution of our approach first introduced in [23]. In particular, we add here the following contributions:

- Integration of velocity fields in order to improve the performance of the method in the vicinity of the obstacles.
- New extensive simulations in environments populated by many obstacles moving in formation.
- Application and validation of the proposed method on a flying robot.

The paper is organized as follows. The navigation problem is formulated in Section 2. In Section 3 we outline the proposed NMPC approach, while in Section 4 we formally define the concept of ACS and derive the associated collision avoidance constraint. In Section 5 we offer some guidelines for the implementation of the proposed method. Simulation results for a differential-drive robot are presented in Section 6, while in Section 7 we illustrate the application of the proposed method to flying robots. In Section 8 we integrate velocity fields to the proposed method in order to improve the quality of the resulting maneuvers. Finally, some concluding remarks are offered in Section 9.

2. Problem formulation

Consider a robotic system whose generalized coordinates are collected in a configuration vector $\boldsymbol{q}$ taking values in an n-dimensional configuration space $\mathcal{C}$. For the sake of generality, we assume that the robot is subject to $k \geq 0$ nonholonomic constraints, expressed in Pfaffian form as $\boldsymbol{A}^T(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{0}$, with $\boldsymbol{A}^T(\boldsymbol{q}) \in \mathbb{R}^{k\times n}$.

The robot kinematics can be expressed as

$$\dot{\boldsymbol{q}} = \boldsymbol{G}(\boldsymbol{q})\boldsymbol{\nu}, \tag{2}$$

being $\boldsymbol{G}(\boldsymbol{q}) \in \mathbb{R}^{n\times m}$ a matrix whose columns span the null space of $\boldsymbol{A}^T(\boldsymbol{q})$, with $m = n - k$, and $\boldsymbol{\nu} \in \mathbb{R}^m$ the robot *pseudovelocities*. In the absence of nonholonomic constraints, $\boldsymbol{\nu}$ and $\dot{\boldsymbol{q}}$ coincide, i.e., $\boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{I}_n$.

The robot dynamics can be expressed in Lagrange form as

$$\boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{S}(\boldsymbol{q})\boldsymbol{u} + \boldsymbol{A}(\boldsymbol{q})\lambda, \tag{3}$$

where $\boldsymbol{B}(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ is the inertia matrix, $\boldsymbol{n}(\boldsymbol{q},\dot{\boldsymbol{q}}) \in \mathbb{R}^n$ are the velocity and gravitational terms, $\boldsymbol{u} \in \mathbb{R}^{n_u}$ are the generalized forces exerted by the n_u robot actuators, $\boldsymbol{S}(\boldsymbol{q}) \in \mathbb{R}^{n\times n_u}$ is the matrix mapping $\boldsymbol{u}$ to generalized forces performing work on $\boldsymbol{q}$, and $\boldsymbol{A}(\boldsymbol{q})\lambda$ are the reaction forces generated by the nonholonomic constraints, with $\lambda \in \mathbb{R}^k$ the associated Lagrange multipliers. Manipulating (2) and (3) we can obtain the robot state-space reduced model as [24]

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) = \begin{pmatrix} \boldsymbol{G}(\boldsymbol{q})\boldsymbol{\nu} \\ \boldsymbol{M}^{-1}(\boldsymbol{q})(\boldsymbol{E}(\boldsymbol{q})\boldsymbol{u} - \boldsymbol{m}(\boldsymbol{q},\boldsymbol{\nu})) \end{pmatrix}, \tag{4}$$

with the state defined as $\boldsymbol{x} = (\boldsymbol{q}, \boldsymbol{\nu})$ and

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

The robot operates in a workspace $\mathcal{W}$ with dimensions n_w ($n_w = 2$ or 3), populated by fixed and/or moving obstacles. We denote by $\mathcal{R}(\boldsymbol{q}) \subset \mathcal{W}$ the volume occupied by the robot at configuration $\boldsymbol{q}$ and by $\mathcal{O}(t) \subset \mathcal{W}$ the volume occupied by the obstacles at time t.

A navigation task is assigned to the robot in terms of a set of variables $\boldsymbol{y} \in \mathcal{Y}$, which describe the position of a representative point[1] on the robot and are related to the configuration via the forward kinematic map $\boldsymbol{y} = \boldsymbol{k}(\boldsymbol{q})$.

The navigation problem considered in this paper consists in generating in *real-time* a motion that starts from any initial configuration and:

(1) drives the robot representative point to an assigned desired position $\boldsymbol{y}_d$;
(2) is *kinodynamically feasible*, in the sense that it is consistent with model (4) and satisfies the existing constraints on both $\boldsymbol{x}$ (e.g., joint and velocity limits) and $\boldsymbol{u}$ (e.g., torque bounds);
(3) is collision-free, i.e., $\mathcal{R}(\boldsymbol{q}(t)) \cap \mathcal{O}(t) = \emptyset$, for all t.

For the solution of this problem, we will assume that the robot is at all times aware of its own state $\boldsymbol{x}$, while position and velocity of the obstacles are measured by appropriate on-board sensors.

3. The proposed NMPC approach

In order to generate the desired motion, we rely on the use of a real-time NMPC algorithm. At each control cycle, NMPC solves a constrained OCP defined along a finite time horizon H. Typically, for its

[1] The task can be defined differently. For example, one may include in $\boldsymbol{y}$ the orientation of certain bodies of the robot, or even set directly $\boldsymbol{y} = \boldsymbol{q}$. The proposed method can be applied without any modification, provided that the error vector is computed accordingly.

numerical solution, the OCP has to be transcribed into a discrete-time, finite-dimensional Nonlinear Program (NLP).

Let us denote by δ the sampling interval used for the discretization and by $N = H/\delta$ the number of control intervals resulting within the prediction horizon. For the NLP to be solved at the generic time instant t_k, let us denote by $\boldsymbol{x}_{k|i}$ and $\boldsymbol{u}_{k|i}$ the predicted robot state and control inputs vector at the discrete time instant t_{k+i}, computed at t_k. Let us also denote by $\boldsymbol{y}_{k|i}$ and $\dot{\boldsymbol{y}}_{k|i}$ the predicted position and velocity of the representative point of the robot at time t_{k+i}. We define the predicted task error at t_{k+i} as $\boldsymbol{e}_{k|i} = \boldsymbol{y}_d - \boldsymbol{y}_{k|i}$. Considering that our objective is to drive the task error to zero, possibly using a minimum control effort, the running and terminal costs can be expressed[2], respectively, as

$$V_{k|i}(\boldsymbol{x}_{k|i}, \boldsymbol{u}_{k|i}) = \boldsymbol{e}_{k|i}^T \boldsymbol{Q} \boldsymbol{e}_{k|i} + \dot{\boldsymbol{y}}_{k|i}^T \boldsymbol{P} \dot{\boldsymbol{y}}_{k|i} + \boldsymbol{u}_{k|i}^T \boldsymbol{R}\, \boldsymbol{u}_{k|i} \tag{5}$$

$$V_{k|N}(\boldsymbol{x}_{k|N}) = \boldsymbol{e}_{k|N}^T \boldsymbol{Q}_N \boldsymbol{e}_{k|N} + \dot{\boldsymbol{y}}_{k|N}^T \boldsymbol{P}_N \dot{\boldsymbol{y}}_{k|N}. \tag{6}$$

Here, $\boldsymbol{Q}$, $\boldsymbol{P}$ and $\boldsymbol{R}$ are weighting matrices of appropriate dimensions 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 weighting matrices for the first two quantities at the final time instant.

The NLP to be solved at time instant t_k is

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

subject to:

$$\boldsymbol{x}_{k|0} - \boldsymbol{x}_k = \boldsymbol{0}$$

$$\boldsymbol{x}_{k|i+1} - \boldsymbol{F}(\boldsymbol{x}_{k|i}, \boldsymbol{u}_{k|i}) = \boldsymbol{0}, \quad i = 0, \dots, N-1$$

$$\boldsymbol{x}_{\min} \le \boldsymbol{x}_{k|i} \le \boldsymbol{x}_{\max}, \quad i = 0, \dots, N$$

$$\boldsymbol{u}_{\min} \le \boldsymbol{u}_{k|i} \le \boldsymbol{u}_{\max}, \quad i = 0, \dots, N-1$$

collision avoidance constraints at $t_k, \dots, t_{k+N}$,

with $\boldsymbol{X}_k = \{\boldsymbol{x}_{k|0}, \dots, \boldsymbol{x}_{k|N}\}$, $\boldsymbol{U}_k = \{\boldsymbol{u}_{k|0}, \dots, \boldsymbol{u}_{k|N-1}\}$, $\boldsymbol{x}_k$ the state of the robot at time t_k and $\boldsymbol{F}(\cdot,\cdot)$ the discrete-time model of the robot obtained via numerical integration of (4) under the assumption of piecewise-constant control inputs. The vectors $\boldsymbol{x}_{\min}, \boldsymbol{x}_{\max}$ 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.

The solution of the NLP consists of the optimal state and input sequences $\boldsymbol{X}_k^*$ and $\boldsymbol{U}_k^*$, from which the first control action $\boldsymbol{u}_{k|0}^*$ (i.e., the one associated to the interval $[t_k, t_k + \delta]$) is extracted and applied to the robot.

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 give a formal definition of what an ACS is, and then derive the corresponding collision avoidance constraint that will be included in our NLP. 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. In fact, the method will be later applied to both ground and flying robots.

[2] This cost function represents a minimal implementation for our navigation task, but specific terms relevant to the robotic platform of choice might be added, e.g., a penalty on the orientation for flying robots or on joint velocities for mobile manipulators.

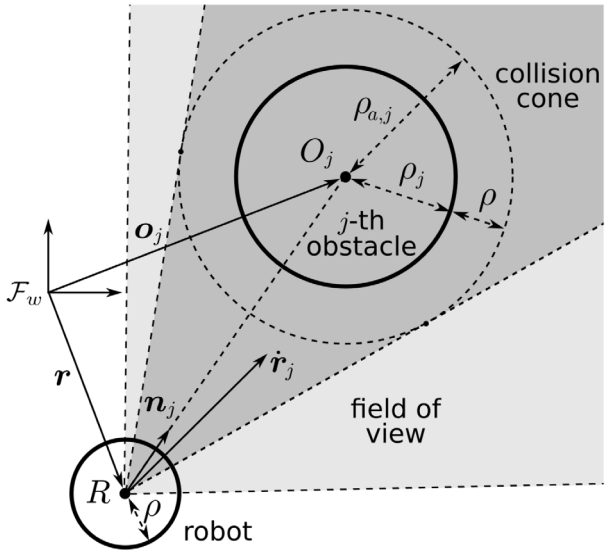


Fig. 1. Obstacle $\mathcal{O}_j$ is considered dangerous for the robot at a certain time if the relative velocity $\dot{\boldsymbol{r}}_j$ of the robot with respect to the obstacle lies inside the collision cone.

4.1. Preliminaries

First, we are going to use bounding spheres for both the robot and the obstacles in order to approximate their occupancy volumes.[3] 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 $\mathcal{O}_j \subset \mathcal{O}$, denoting its center by O_j, the corresponding position vector by $\boldsymbol{o}_j$ and its velocity by $\dot{\boldsymbol{o}}_j$. The state of the generic obstacle is denoted by $\boldsymbol{\xi}_j = (\boldsymbol{o}_j, \dot{\boldsymbol{o}}_j)$. Finally, let $\boldsymbol{n}_j = (\boldsymbol{o}_j - \boldsymbol{r})/\|\boldsymbol{o}_j - \boldsymbol{r}\|$ 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.

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. For the definition of the ACS we will assume that the obstacles move along a given direction with constant speed. Note that although this assumption represents only an approximation of the motion of the obstacle when it is not moving with constant velocity, we emphasize how, within the scope of a collision avoidance constraint, its accuracy increases as the distance between the robot and the obstacle decreases.

In order to establish a criterion to evaluate whether an obstacle is dangerous, we are going to use the concept of *collision cone* [2]. In particular, by augmenting 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. The generic obstacle $\mathcal{O}_j$ is *dangerous* at time t if at the same time instant $\dot{\boldsymbol{r}}_j$ lies inside the collision cone (see Fig. 1). Simple

[3] Note that although a bounding sphere is a good approximation of the occupancy volume of the robot and the obstacle and a reasonable choice under a safety viewpoint in high-speed navigation, in cluttered environments it might be necessary to consider multiple bounding spheres to approximate the occupancy volume of a body. However, an increase in the number of the bounding spheres would also increase the number of collision avoidance constraints and thus the computational complexity of the NMPC.

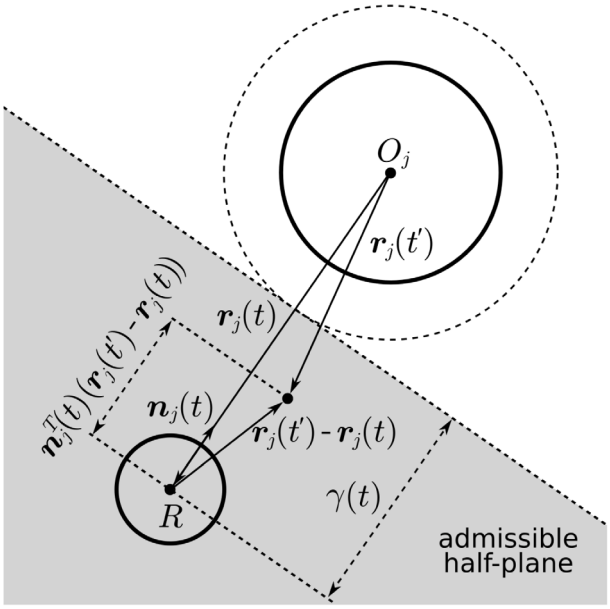


Fig. 2. The admissible half-plane for the relative position of robot R with respect to the obstacle O_j at time t', $t' \in [t, t+\Delta t]$.

geometrical arguments lead to the following condition for an obstacle to be dangerous:

$$h(\boldsymbol{x}, \boldsymbol{\xi}_j) = \boldsymbol{n}_j^T \frac{\dot{\boldsymbol{r}}_j}{\|\dot{\boldsymbol{r}}_j\|} - \frac{\sqrt{\|\boldsymbol{r}_j\|^2 - \rho_{a,j}^2}}{\|\boldsymbol{r}_j\|} \geq 0. \tag{7}$$

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.

Let us consider the robot at time t with the position and velocity of R being $\boldsymbol{r}(t)$ and $\dot{\boldsymbol{r}}(t)$ respectively. At the same time instant let us also consider a dangerous obstacle $\mathcal{O}_j$ being in state $\boldsymbol{\xi}_j(t) = (\boldsymbol{o}_j(t), \dot{\boldsymbol{o}}_j)$. To avoid the imminent collision, it is sufficient to consider a collision-free motion at the end of which the relative velocity of the robot with respect to the obstacle projected on the original direction of collision is zero, i.e.,

$$\boldsymbol{n}_j^T(t)\dot{\boldsymbol{r}}_j(t+\Delta t) = 0, \tag{8}$$

where $\boldsymbol{n}_j(t)$ reflects the original direction of collision while Δt is the duration of the motion. So we consider the robot moving for a time interval $[t, t+\Delta t]$ in such a way that at each time instant $t' \in [t, t+\Delta t]$ its relative velocity with respect to the obstacle projected on the original direction of collision, $\boldsymbol{n}_j^T(t)\dot{\boldsymbol{r}}_j(t')$, decreases with constant rate α until it eventually becomes zero. Note that the duration of the motion and thus the rate α need to be appropriate in order to maintain the motion collision-free.

In order to determine the value of α, let us first consider a sufficient condition for the motion to be collision-free:

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

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

$$\boldsymbol{n}_j^T(t)(\boldsymbol{r}_j(t') - \boldsymbol{r}_j(t)) \leq \gamma(t) \quad \forall t' \in [t, t+\Delta t]. \tag{9}$$

Note that inequality (9) 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 $t' \in [t, t+\Delta t]$ (see Fig. 2).

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

$$\boldsymbol{n}_j^T(t)\boldsymbol{r}(t') = \boldsymbol{n}_j^T(t)\boldsymbol{r}(t) + \boldsymbol{n}_j^T(t)\dot{\boldsymbol{r}}(t)(t'-t) + \frac{1}{2}\alpha(t'-t)^2 \tag{10}$$

and its projected velocity as

$$\boldsymbol{n}_j^T(t)\dot{\boldsymbol{r}}(t') = \boldsymbol{n}_j^T(t)\dot{\boldsymbol{r}}(t) + \alpha(t'-t). \tag{11}$$

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

$$\boldsymbol{n}_j^T(t)\boldsymbol{o}_j(t') = \boldsymbol{n}_j^T(t)\boldsymbol{o}_j(t) + \boldsymbol{n}_j^T(t)\dot{\boldsymbol{o}}_j(t'-t). \tag{12}$$

From (8), (9), (10), (11) and (12) and for $t' = t + \Delta t$ we get for the duration of motion that

$$\Delta t \leq \frac{2\gamma(t)}{\boldsymbol{n}_j^T(t)(\dot{\boldsymbol{r}}(t) - \dot{\boldsymbol{o}}_j)}. \tag{13}$$

For $t' = t + \Delta t$, by substituting (8) and (11) in (13) and considering also that $\alpha < 0$ we get the condition for the required deceleration in the direction of collision that need to be satisfied in order to maintain the motion collision-free:

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

In order to ensure that the considered motion can be implemented by the robot, we will investigate whether the required deceleration projected on the robot actuation space lies within the actuation limits. In particular, we want to ensure that the robot is able to enforce at least the minimum required deceleration in the original direction of collision that is

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

Note that $\bar{\alpha}$ depends on both the robot and the j-th obstacle state.

Considering that the acceleration of R is

$$\ddot{\boldsymbol{r}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{\nu}} + \dot{\boldsymbol{J}}(\boldsymbol{q})\boldsymbol{\nu} \tag{14}$$

and that from (4) we can express $\dot{\boldsymbol{\nu}}$ as

$$\dot{\boldsymbol{\nu}} = \boldsymbol{M}^{-1}(\boldsymbol{q})(\boldsymbol{E}(\boldsymbol{q})\boldsymbol{u} - \boldsymbol{m}(\boldsymbol{q}, \boldsymbol{\nu})), \tag{15}$$

we can obtain the required control inputs for this deceleration by substituting $\ddot{\boldsymbol{r}} = \boldsymbol{n}_j(t)\bar{\alpha}$ and (15) in (14). Using the Moore–Penrose pseudoinverse, we get the minimum norm control inputs needed in order to apply the deceleration $\ddot{\boldsymbol{r}} = \boldsymbol{n}_j(t)\bar{\alpha}$ to R, that is

$$\boldsymbol{u}_{\bar{\alpha}}(\boldsymbol{x}, \boldsymbol{\xi}_j) = \left(\boldsymbol{J}(\boldsymbol{q})\boldsymbol{M}^{-1}(\boldsymbol{q})\boldsymbol{E}(\boldsymbol{q})\right)^{\dagger} \boldsymbol{\beta}(\boldsymbol{x}, \boldsymbol{\xi}_j), \tag{16}$$

where

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

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

$$\boldsymbol{u}_{\min} \leq \boldsymbol{u}_{\bar{\alpha}}(\boldsymbol{x}, \boldsymbol{\xi}_j) \leq \boldsymbol{u}_{\max}. \tag{17}$$

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 [19], 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 [20]) 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 (17) 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 (17) 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 $\boldsymbol{u}_{\tilde{a}}(\boldsymbol{x}, \boldsymbol{\xi}_j)$ as given by (16) by the sigmoid function $g(h(\boldsymbol{x}, \boldsymbol{\xi}_j)) = 1/(1 + e^{-\kappa h(\boldsymbol{x}, \boldsymbol{\xi}_j)})$, with κ 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

$$\boldsymbol{u}_{\min} \leq \boldsymbol{u}_b(\boldsymbol{x}_{k|i}, \boldsymbol{\xi}_{j,k|i}) \leq \boldsymbol{u}_{\max}, \quad \begin{matrix} i = 0, \dots, N, \\ j = 1, \dots, n_o, \end{matrix} \tag{18}$$

where n_o is the number of obstacles considered by the NMPC and $\boldsymbol{\xi}_{j,k|i}$ is the predicted state of the j-th obstacle at t_{k+i}. Given the state $\boldsymbol{\xi}_{j,k} = (\boldsymbol{o}_{j,k}, \dot{\boldsymbol{o}}_{j,k})$ of the j-th obstacle at time t_k, its predicted state at time t_{k+i} is obtained using the model $\boldsymbol{\xi}_{j,k|i} = (\boldsymbol{o}_{j,k} + i\delta\dot{\boldsymbol{o}}_{j,k}, \dot{\boldsymbol{o}}_{j,k})$. Clearly, any inaccuracy in the considered prediction model introduces uncertainty. However, the fast re-planning of the NMPC in combination with the relatively short prediction horizon can account for an unexpected change in the obstacle motion.

Finally, it should be noted that the proposed collision avoidance constraint is inherently more conservative than its purely distance-based counterparts as it considers in addition to the robot–obstacle distance, their relative velocity and the robot actuation capabilities. However, one can always add conservativeness in order to enhance the robot and environment safety, by simply increasing the radius of the bounding sphere of the obstacle.

5. Implementation guidelines

The implementation of the proposed method will significantly affect its performance. In this section, we will focus on three main aspects, namely the tuning of the weighting matrices, the initialization of the NMPC solution, and the compensation of the delay introduced by the solution of the NLP, and propose some simple guidelines.

1) Tuning the weighting matrices: The choice of the weighting matrices depends on many factors, including the actuation capabilities of the robot, its inertial properties, the number and velocity of the obstacles, and the size of the prediction horizon. In our simulations, these matrices were determined after a campaign of trials on representative scenarios, following some simple guidelines:

- $\boldsymbol{Q}_N$ should be (in norm) at least one order of magnitude larger than $\boldsymbol{Q}$. With this choice the robot will be ultimately driven to the goal, but still allowed to move temporarily away from it if this is required for collision avoidance.
- $\boldsymbol{P}$ and $\boldsymbol{P}_N$ should be one order of magnitude smaller than $\boldsymbol{Q}$ and $\boldsymbol{Q}_N$, respectively. This allows the robot to move at high speed when it is far from the goal, while providing an appropriate damping (and therefore avoiding oscillations) in the vicinity of the goal.
- $\boldsymbol{R}$ should be chosen in such a way that in (5) the first term dominates over the last term. This makes it possible for the robot to perform aggressive motions, e.g., in order to quickly approach to goal or avoid obstacles.

2) Initialization of the NMPC solution: At each control instant t_k, the NLP solver used by the NMPC is initialized at the solution of time t_{k-1} appropriately shifted in time. This has the effect of reducing the number of iterations that the solver has to perform to converge to a solution, effectively decreasing the computation time.

3) Delay compensation: In Section 3, we implicitly assumed an instantaneous solution of the NLP at time t_k in order to obtain the control action that will be applied to the system for the time interval $[t_k, t_k + \delta]$. However, in practice, the computational time required for the solution of the NLP can be significant, even comparable to the sampling time δ. In order to deal with this delay that can potentially jeopardize the robot and environment safety (especially at high speeds), we employ a *compensation by prediction* approach [25].

In particular, let us consider an upper bound $D \leq \delta$ for the maximum computational time for the NLP.[4] Instead of t_k, we trigger the solution of the NLP at time $t_k - D$ using an estimation of the state that the robot and the obstacles will have at t_k. For the estimation, we integrate the prediction models of the robot and the obstacles for the time interval $[t_k - D, t_k]$ using their state at $t_k - D$ and the control $\boldsymbol{u}^*_{k-1|0}$ that acts on the robot. It is clear that the accuracy of this estimation depends on the accuracy of the prediction models for the robot and the obstacles. This is also true for the performance of the NMPC in general. In any case, thanks to its fast re-planning on a control horizon, the NMPC has an intrinsic anticipatory behavior that alleviates the effect of possible inaccuracies.

There are alternative ways to deal with the delay, e.g., increasing the integration timestep to achieve a higher control frequency [26], employing high-frequency sensitivity-based approximations of the NLP to be applied in between samples [27], or adopting an event-triggered NMPC approach to reduce the total amount of computations, solving the NLP only when the triggering condition is met [28].

6. Application to ground robots

To show the effectiveness of the proposed method, we will first test it on a differential-drive robot. In this section, we will first present the considered robot and then show the results of the conducted simulations.

6.1. Differential-drive robot

The considered differential-drive robot is illustrated in Fig. 3. The length and width of the vehicle are respectively $l_1 = 0.60$ m and $l_2 = 0.30$ m. The two driving wheels of the robot have radius $r = 0.10$ m. The robot is also equipped with a caster wheel for mechanical balance. We assume that the projection of the centroid of each driving wheel on the ground corresponds to the contact point between the ground and the wheel. The length of the line segment connecting the centroids of

[4] In practice one can obtain D by considering a conservative upper bound of the maximum computational times reported after a simulation campaign on different scenarios.

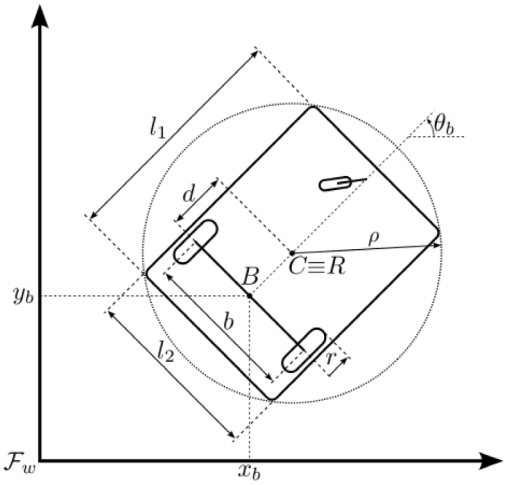


Fig. 3. The considered differential-drive robot. With dashed line is illustrated the bounding circle that is used for the collision avoidance constraint.

the two driving wheels is $b = 0.30$ m. We denote by B the midpoint of this line segment. The robot center of mass is located at C which lies on the sagittal axis of the vehicle at a distance $d = 0.25$ m from B and coincides with the geometric center of the robot. The vehicle mass and its moment of inertia are respectively $m_c = 50$ kg and $I_c = 1.14$ kg m^2, making it a rather heavy vehicle. We consider the robot to be controlled at the wheel torque level with torque bounds set to 2.5 Nm.

The vector of the robot generalized coordinates is

$$\boldsymbol{q} = (x_b, y_b, \theta_b) \in \mathbb{R}^3,$$

with x_b and y_b being the Cartesian coordinates of B in the world frame $\mathcal{F}_w$ and θ_b being the orientation of the vehicle and consequently of the driving wheels,

The kinematic constraint of the considered robot emanates from its contact with the ground through its wheels. The ground-wheel friction, which for simplicity we assume to be adequate to prevent slippage, prevents the driving wheels from moving along the direction of their common axis, enforcing the following nonholonomic constraint

$$\underbrace{\begin{pmatrix}\sin\theta_b & -\cos\theta_b & 0\end{pmatrix}}_{\boldsymbol{A}^T(\boldsymbol{q})\in\mathbb{R}^{1\times 3}} \underbrace{\begin{pmatrix}\dot{x}_b\\ \dot{y}_b\\ \dot{\theta}_b\end{pmatrix}}_{\dot{\boldsymbol{q}}} = 0. \tag{19}$$

Regarding the robot kinematics (2), we define the vector of pseudovelocities as $\boldsymbol{\nu} = (\upsilon, \omega)$, with υ and ω being respectively the robot *driving* and *steering velocity*,[5] and

$$\boldsymbol{G}(\boldsymbol{q}) = \begin{pmatrix}\cos\theta_b & 0\\ \sin\theta_b & 0\\ 0 & 1\end{pmatrix}.$$

Regarding the robot dynamics (3), we consider the control input vector $\boldsymbol{u} = (\tau_r, \tau_l)$ with τ_r and τ_l being the torques on the right and left driving wheels respectively, and $\boldsymbol{B}(\boldsymbol{q})$, $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and $\boldsymbol{S}(\boldsymbol{q})$ being respectively

$$\boldsymbol{B}(\boldsymbol{q}) = \begin{pmatrix} m_c & 0 & -m_c d\sin\theta_b \\ 0 & m_c & m_c d\cos\theta_b \\ -m_c d\sin\theta_b & m_c d\cos\theta_b & I_c + m_c d^2 \end{pmatrix}$$

$$\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \begin{pmatrix} -m_c d\cos\theta_b \dot{\theta}_b^2 \\ -m_c d\sin\theta_b \dot{\theta}_b^2 \\ 0 \end{pmatrix}$$

$$\boldsymbol{S}(\boldsymbol{q}) = \begin{pmatrix} 1/r\cos\theta_b & 1/r\cos\theta_b \\ 1/r\sin\theta_b & 1/r\sin\theta_b \\ b/(2r) & -b/(2r) \end{pmatrix}.$$

As concerns the navigation task, we consider as representative the point C of the robot and so

$$\boldsymbol{y} = \begin{pmatrix} x_b + d\cos\theta_b \\ y_b + d\sin\theta_b \end{pmatrix}.$$

Finally, regarding the collision avoidance constraint, the bounding sphere that envelops the robot degrades to a bounding circle (see in Fig. 3 the bounding circle illustrated with a dashed line). In our case, the center R of the smallest circle coincides with the geometric center of the robot and consequently with the point C, and thus

$$\boldsymbol{r} = \boldsymbol{y}.$$

The radius of the bounding circle is $\rho = 0.34$ m.

6.2. Simulations

In order to show the effectiveness of the proposed method, we conducted a series of simulations in which the differential-drive robot has to navigate various static and dynamic environments. The simulations were implemented in MATLAB run on an Intel Core i9-9900K CPU at 3.60 GHz. For the NMPC, and in order to achieve real-time performance, we used the RTI method [5], implemented within the MATLAB interface of the ACADO Toolkit [29]. In the considered simulations we assume that the robot obtains information about the state of the obstacles (relative position and velocity) via an onboard laser rangefinder with an infinite field-of-view (FOV).

Along with the effectiveness of the proposed method, we would also like to highlight the superiority of the dynamics-aware collision avoidance constraint over its distance-based counterparts. For this purpose, in the simulations, we will compare the proposed NMPC method, with the dynamics-aware (DA) collision avoidance constraint, with a version of it that considers a purely distance-based (DB) collision avoidance constraint of the form

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

where $\boldsymbol{r}_{k|i}$ and $\boldsymbol{o}_{j,k|i}$ are respectively the position of R and O_j at t_{k+i}. For both methods, the obstacles considered for collision avoidance are the $n_o = 5$ closest to the robot within its sensor FOV. In the simulations we consider the sampling interval for real-time control to be $\delta = 31$ ms in both cases. Regarding the prediction horizons, we chose the larger value possible that does not violate the real-time performance. Specifically, the prediction horizon for the DB method is set to $H = 0.992$ s while for the DA method is set to $H = 0.93$ s.

The simulations are conducted 25 static and 25 dynamic environments. In each environment, the two methods are tested using three different values for the maximum driving velocity $\upsilon_{\max}$ while the steering velocity is set to $\omega_{\max} = 20/3\,\upsilon_{\max}$ rad/s. The performance of the two methods is assessed according to the following criteria: (1) success rate, (2) time t_g needed for the robot to reach the goal, (3) control effort $J_\tau = \int_0^{t_g} \|\boldsymbol{\tau}\|^2 dt$, (4) length l_p of the resulting path, (5) duration of the longest iteration $\delta_{\max}$ and (6) average iteration time $\bar{\delta}$. Note that a simulation is considered successful if the robot reaches the goal without experiencing any collision and the maximum computational time does not exceed the limit of 31 ms.

The accompanying video, available also at https://youtu.be/nRV4UYqYbuA, contains video clips of selected simulations.

[5] As driving velocity we define the velocity of the robot along its longitudinal axis, while as steering we define the angular velocity around the vertical axis at B.

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

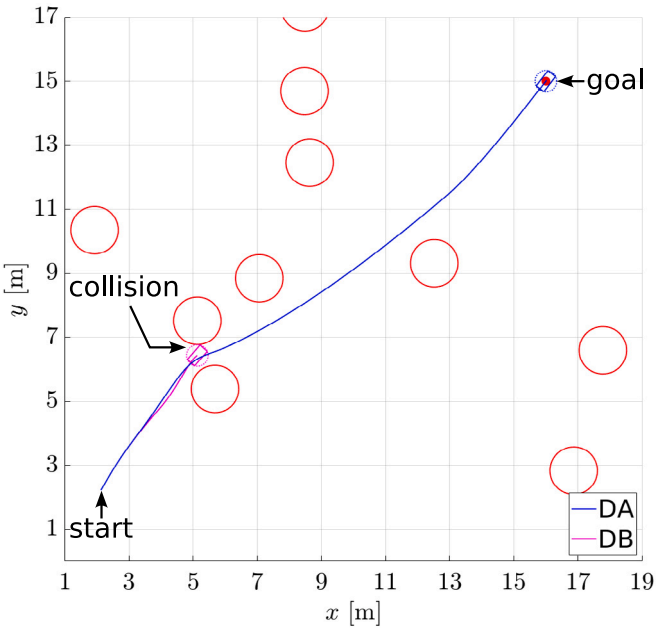


Fig. 4. Motions generated by the two methods in one of the static environments, with $v_{\max} = 1.2$ m/s. The DB method cannot avoid collision with an obstacle, whereas DA goes safely through the narrow passage and successfully reaches the goal. See also simulation 2 in the accompanying video.

6.2.1. Static environments

In the first simulation campaign, the robot has to navigate into 25 environments occupied by 10 randomly placed static obstacles (see Fig. 4). In all the simulations the starting configuration of the robot is $\boldsymbol{q}_s = (2, 2, \pi/3)$ while the goal position is $\boldsymbol{y}_g = (16, 15)$. Table 1 (top) reports, for increasing values of the maximum driving velocity $v_{\max}$, the simulation results of the two methods averaged over the 25 environments.

From the results, it is evident that the two methods have similar performance for lower $v_{\max}$, reporting the same success rate. In simulation 1 of the accompanying video, we offer an illustrative example of the robot motion for $v_{\max} = 0.9$ m/s. However, by increasing the maximum driving velocity to $v_{\max} = 1.2$ m/s, one can notice the success rate of the DB method dropping to 88%, while at the same time, DA is hardly affected by this increase. An indication for the cause of this behavior can be the stopping time[6] t_s reported on Table 1, that in the case of $v_{\max} = 1.2$ m/s is significantly larger than the prediction horizon. This suggests that a robot navigating under the DB method with the given maximum driving velocity would require either a longer prediction horizon or greater braking capabilities in order to remain safe. On the contrary, this is not necessary for the DA method that works effectively, even with a shorter prediction horizon than the DB method. Fig. 4 illustrates a representative example of the different behavior of the two methods for $v_{\max} = 1.2$ m/s (see also simulation 2 in the accompanying video). In particular, it shows the DB method running into a collision during its attempt to pass through a narrow passage formed by two adjacent obstacles. On the other hand, the DA method reacts almost 1 s earlier to the presence of the obstacle, thanks to the dynamics-aware collision avoidance constraint that can detect a violation earlier than the distance-based constraint (see in Fig. 5 how the control inputs of the two methods start to differ at $t \approx 2$ s). This early reaction enables DA to go through the narrow passage and reach the goal safely.

From the rest of the performance criteria of Table 1, one can deduce that the trajectories resulting from the DA method are on average slightly shorter and less energy-consuming than those of its distance-based counterpart. Nevertheless, the simplicity of the distance-based collision avoidance constraint leads to a reduced duration of the longest iteration for the DB method compared to those reported for the DA method.

6.2.2. Dynamic environments

In the second simulation campaign, the robot is called to navigate in 25 environments. Each environment is occupied by 10 static and 10 moving obstacles. The considered moving obstacles travel at a constant speed $v_o = v_{\max}/2$ along straight paths. After traveling a distance of 2.45 m, each obstacle changes its direction of motion by 60° towards the robot. Again, as starting configuration of the robot we set $\boldsymbol{q}_s = (2, 2, \pi/3)$ and as goal position $\boldsymbol{y}_g = (16, 15)$.

The results for increasing values of the maximum driving velocity are reported in Table 1 (bottom). From those, it is clear that the presence of the moving obstacles affects the performance of both methods. Nevertheless, the success rate of the DA method is maintained over 80%, showing its superior performance over the DB method, whose success rate goes as low as 40%.

Similar to the static, in the dynamic environment the DB method has its higher success rate for the lowest $v_{\max}$, which is, however,

[6] By the term *stopping time* t_s we denote the minimum time required for a robot traveling on a straight line with maximum driving velocity $v_{\max}$ in order to stop.

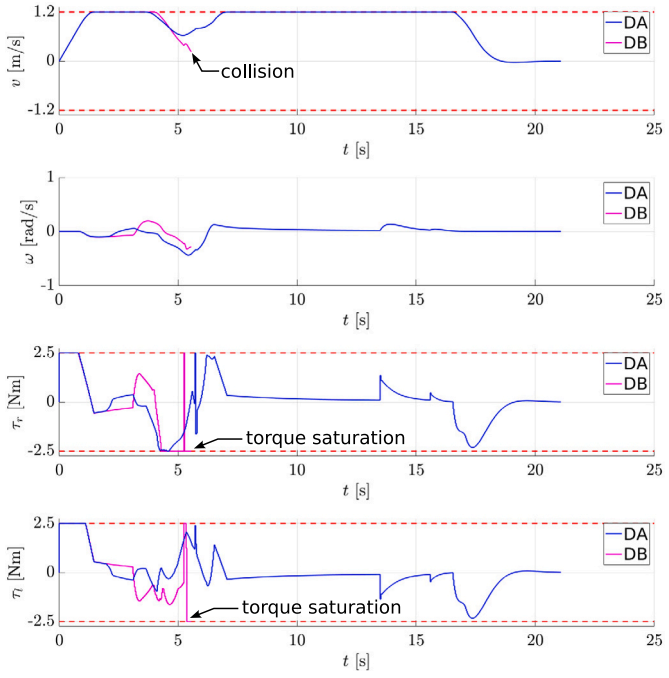


Fig. 5. The driving and steering velocity profiles and the control inputs generated by the two methods in the static environment, with $v_{\max} = 1.2$ m/s.

lower than the success rate reported for the DA method. An illustrative example of the robot moving in one of the dynamic environments with $v_{\max} = 0.9$ m/s is given in simulation 3 of the accompanying video.

It is clear that in the dynamic environment, the maneuvers needed to be performed by the robot in order to avoid an imminent collision with a moving obstacle are more demanding, as in a worst-case scenario the robot may not only have to stop but also accelerate in the opposite direction in order to avoid it. A maneuver like this might require more than the actuation capabilities available if the danger of an imminent collision is not detected on time. This explains why the performance of the DB method decreases with the increase of $v_{\max}$, plummeting to 40% for $v_{\max} = 1.2$ m/s as well as why it has such a low performance even in the case of the lowest maximum velocity. On the other hand, the DA method guaranteeing that the robot is at all times in an ACS, enables the robot to veer off collision paths earlier.

The snapshots of Fig. 6 illustrate a representative example of the behavior of the two methods for $v_{\max} = 1.2$ m/s in one of the considered dynamic environments (the full motion of the robot is offered in simulation 4 of the accompanying video). In Fig. 7 we also offer the velocity profiles and the control inputs generated throughout the robot motion.

Once again, the DB is unable to safely navigate the robot to the goal, as it leads to collision with one of the moving obstacles. On the contrary, the DA method is able to avert imminent collisions with the obstacles of the environment by reacting earlier to their presence, reaching the goal safely. Note in particular how in Fig. 7 the control inputs of DA started to differ from those of the DB method in the presence of the moving obstacle and how this late reaction of the DB method to the same obstacle led to torque saturation, which was, however, not enough to prevent the imminent collision. One can also appreciate the elaborate avoidance maneuver performed by the DA method in front of another moving obstacle combining a reverse motion and a quick reorientation.

Apart from the aforementioned simulation campaigns and in an attempt to further assess the effectiveness of the proposed method, we tested the performance of the two methods in a series of challenging dynamic environments. The first one is a more cluttered version of the already considered dynamic environments consisting of 10 static and 15 moving obstacles that travel at a speed of 0.4 m/s, with direction changes of 60° towards the robot every 2.45 m. In the considered simulation, the robot maximum driving velocity is set to $v_{\max} = 1.2$ m/s. The results that are only shown in simulation 5 of the accompanying video, show that the proposed method is effective also in this more challenging dynamic environment, with its distance-based counterpart failing once again to navigate safely.

The two methods were also tested in environments in which the obstacles move in formation. The first one consists of 36 obstacles, arranged in 6 groups. In each group, the obstacles form a straight line, while the gap between each other is slightly larger than the diameter of the bounding circle of the considered mobile robot. The obstacles move at speed $v_o = 0.65$ m/s, with the obstacles of the same group having the same direction of motion. This direction changes by 60° toward the robot every 3.8 s. In the simulation, the robot starts from initial configuration $\boldsymbol{q}_s = (2, 2, \pi/3)$ and is called to reach the goal position $\boldsymbol{y}_g = (16, 15)$ while the maximum permitted driving velocity is set to $v_{\max} = 1.3$ m/s. Snapshots of the motion resulting by the two methods are offered in Fig. 8, while Fig. 9 shows the velocity profiles and the control inputs generated throughout the robot motion (for a video with the full robot motion see simulation 6 in the accompanying video). In Fig. 8 we can see that the DB method was not able to navigate safely in this dynamic environment, colliding at its first encounter with a moving obstacle. On the other hand, the DA method was able to avoid the collision with the same obstacle by reacting earlier to its presence (see in Fig. 9 the associated control inputs). Although this behavior of the proposed method was already evident from the previous simulations, here one can appreciate how the DA method behaves when the robot is called to pass through a moving narrow passage (created by the obstacles moving in formation). In particular, at time $t = 13.671$ s, and in view of the narrow passage, the robot tends to align its heading direction with the direction of motion of the group of obstacles attempting to mirror their motion. This maneuver permitted the robot to place itself between the two obstacles (see snapshot at time $t = 15.19$ s), and by properly arranging its steering and driving velocity, to navigate through the narrow passage. In the same way, the robot passes through the next narrow passage formed by the following group of obstacles, reaching, in the end, the goal safely. It should be noted that throughout the robot motion the maximum computational time was 28.3 ms.

The second environment of this type consists of 32 moving obstacles arranged in such a way that they form 4 circles. Again the clearance between two obstacles of the same formation is large enough to permit the bounding circle of the robot to pass through it. The obstacles are again moving at a speed $v_o = 0.65$ m/s with the common direction of motion of each group changing every 3.8 s by 60° toward the robot. In Fig. 10 we offer snapshots of the motion resulting from the two methods in the considered environment, while the full motion along with the velocity profiles and the control inputs are offered in simulation 7 of the accompanying video. Once again, the DB method is unable to navigate safely in the dynamic environment and collides at the very beginning of the robot motion with one of the obstacles. On the contrary, the DA method started immediately a collision avoidance maneuver that permitted the robot to pass through the narrow passage formed between two moving obstacles. Note that in order to do so the robot had to mirror the motion of the two obstacles. The robot follows the same strategy in order to exit from the circular formation. In this way, the proposed dynamics-aware method was able to avoid all the imminent collisions and navigate safely in this challenging dynamic environment.

7. Application to flying robots

The proposed method has been also tested on a flying robot in order to showcase its applicability to fast highly maneuverable systems

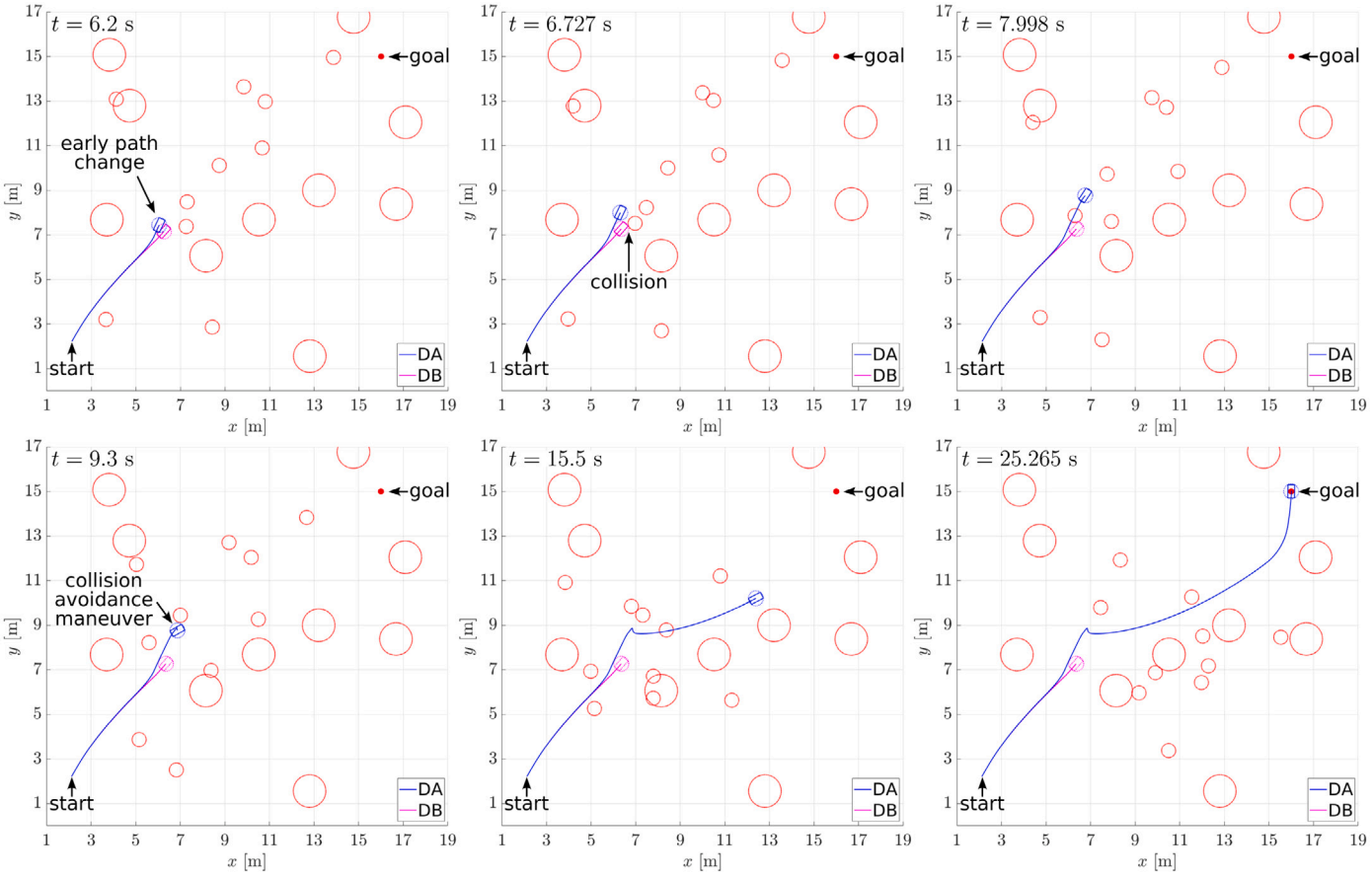


Fig. 6. Snapshots of the motion generated by the two methods in one of the dynamic environments, with $v_{\max} = 1.2$ m/s. As in the static environment, the DB method cannot avoid collision with an obstacle, whereas the DA method safely navigates to the goal. See also simulation 4 in the accompanying video.

while maintaining real-time performance. As we already mentioned, the collision avoidance constraint and the related discussion can be easily extended to the 3-dimensional case, so we will directly examine how the proposed method performs in a dynamic environment.

7.1. Quadrotor

The vehicle we consider is the quadrotor UAV illustrated in Fig. 11, whose equations of motion correspond to that of a single rigid body (SRB) under the action of the forces/torques generated by the rotors. We assume that the robot center of mass C is located at the geometric center of the vehicle B. The vehicle mass and its body-fixed moment of inertia are respectively $m_c = 0.033$ kg and $\boldsymbol{I}_c = \text{diag}\{I_{xx}, I_{yy}, I_{zz}\}$, with $I_{xx} = I_{yy} = 1.395 \cdot 10^{-5}$ kg m^2 and $I_{zz} = 2.173 \cdot 10^{-5}$ kg m^2.

The position and orientation of the quadrotor are described respectively by the Cartesian coordinates of the center of mass $(x_c, y_c, z_c) \in \mathbb{R}^3$ and by the RPY Euler angles $(\phi, \theta, \psi) \in \mathbb{R}^3$ expressing the orientation of the body-fixed frame $\mathcal{F}_b$ with respect to the inertial world frame $\mathcal{F}_w$. The robot configuration is then described by the generalized coordinates

$$\boldsymbol{q} = (x_c, y_c, z_c, \phi, \theta, \psi) \in \mathbb{R}^6.$$

We find convenient to express the translational dynamics in the world frame $\mathcal{F}_w$, and the rotational dynamics in the body frame $\mathcal{F}_b$. With this choice, the velocity vector is defined as

$$\boldsymbol{\nu} = (\boldsymbol{v}_c, \boldsymbol{\omega}_b) \in \mathbb{R}^6,$$

with $\boldsymbol{v}_c = (\dot{x}_c, \dot{y}_c, \dot{z}_c)$ being the linear velocity and $\boldsymbol{\omega}_b = (\omega_{bx}, \omega_{by}, \omega_{bz})$ the angular velocity in $\mathcal{F}_b$.

Regarding the vector of control inputs, we consider $\boldsymbol{u} = (T, \tau_\phi, \tau_\theta, \tau_\psi) \in \mathbb{R}^4$, composed of the total thrust T and of the three torques in the body frame $\mathcal{F}_b$, obtained from the cumulative effect of the aerodynamic forces and torques generated by the rotors. In our simulations, we will assume the thrust T to be bounded to $[0, 0.8]$ N, the torques τ_ϕ and τ_θ to $[-2.5 \cdot 10^{-3}, 2.5 \cdot 10^{-3}]$ Nm and τ_ψ to $[-2 \cdot 10^{-3}, 2 \cdot 10^{-3}]$ Nm.

We can express the quadrotor dynamics in the form (4), with

$$\boldsymbol{G}(\boldsymbol{q}) = \begin{pmatrix} \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times1} & \boldsymbol{0}_{3\times1} & \boldsymbol{0}_{3\times1} \\ \boldsymbol{0}_{1\times3} & 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ \boldsymbol{0}_{1\times3} & 0 & \cos\phi & -\sin\phi \\ \boldsymbol{0}_{1\times3} & 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{pmatrix}$$

$$\boldsymbol{M}(\boldsymbol{q}) = \text{diag}\{m_c, m_c, m_c, I_{xx}, I_{yy}, I_{zz}\}$$

$$\boldsymbol{m}(\boldsymbol{q}, \boldsymbol{\nu}) = \begin{pmatrix} 0 \\ 0 \\ -m_c g \\ (I_{zz} - I_{yy})\omega_{by}\omega_{bz} \\ (I_{xx} - I_{zz})\omega_{bx}\omega_{bz} \\ (I_{yy} - I_{xx})\omega_{bx}\omega_{by} \end{pmatrix}$$

$$\boldsymbol{E}(\boldsymbol{q}) = \begin{pmatrix} -\sin\psi\sin\phi - \cos\psi\sin\theta\cos\phi & \boldsymbol{0}_{1\times3} \\ \sin\phi\cos\psi - \sin\psi\sin\theta\cos\phi & \boldsymbol{0}_{1\times3} \\ -\cos\theta\cos\phi & \boldsymbol{0}_{1\times3} \\ \boldsymbol{0}_{3\times1} & \boldsymbol{I}_{3\times3} \end{pmatrix}.$$

For the collision avoidance constraint, we consider a bounding sphere with radius $\rho = 0.1$ m and centered at a reference point R displaced by a small distance d_r above the center of the quadrotor:

$$\boldsymbol{r} = \begin{pmatrix} x_r \\ y_r \\ z_r \end{pmatrix} = \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} + \boldsymbol{R}(\phi, \theta, \psi) \begin{pmatrix} 0 \\ 0 \\ -d_r \end{pmatrix},$$

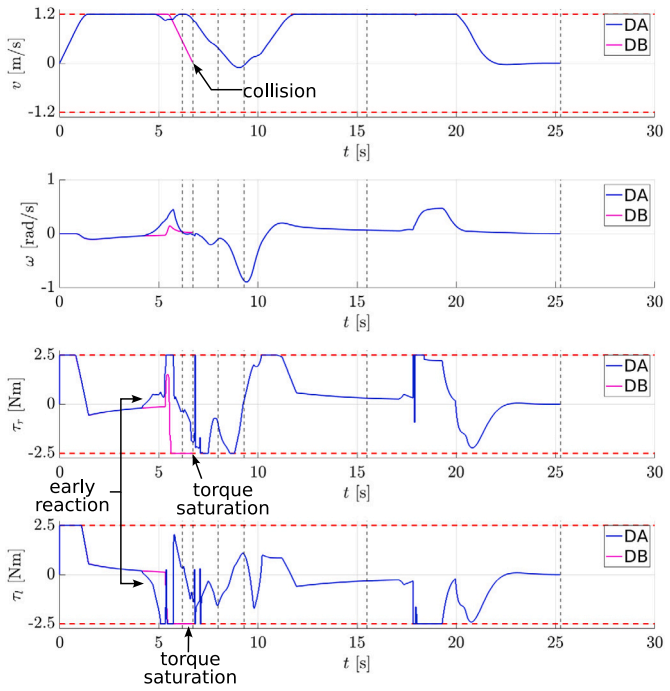


Fig. 7. The driving and steering velocity profiles and the control inputs generated by the two methods in the dynamic environment, with $v_{\max} = 1.2$ m/s. The dashed vertical lines indicate the time instants of the snapshots.

with $\boldsymbol{R}(\phi, \theta, \psi)$ being the rotation matrix obtained from the Euler angles. Concerning the navigation task, we express the error vector with respect to the same point R, so the position vector is defined as $\boldsymbol{y} = \boldsymbol{r}$.

7.2. Simulations

The proposed method has been also tested in a 3D dynamic environment where a quadrotor has to navigate to a predefined goal while avoiding moving obstacles. Specifically, we have considered the Crazyflie 2.1 quadrotor model as described in Section 7.1.

The simulations have been performed in MATLAB, with the NMPC being transcribed using CasADi [30] in a multiple shooting formulation, while the resulting NLP is solved using the Ipopt solver [31]. Given the available time budget fixed by the sampling time $\delta = 20$ ms, we have set the solver to perform 6 iterations for each control step, warm-starting it with an initialization based on the solution obtained at the previous cycle, in a similar fashion to the RTI method. The time horizon H for the optimization is set to 0.5 s. With these settings, the optimization is able to run in real-time, with a maximum computation time $\delta_{\max} = 17$ ms for each control cycle.

The proposed scenario consists of four obstacles with a radius of 0.2 m, three of which are moving with a constant speed $\|\boldsymbol{v}_o\| \in [0.7, 1.6]$ m/s, while the robot typically moves with a higher velocity, resulting in a motion much faster than those in the previous simulations. The robot starts at $\boldsymbol{q}_s = (0, 0, 0, 0, 0, 0)$ in hovering state and has to reach a goal $\boldsymbol{y}_g = (5, 1, 1)$ m. In Fig. 12 we report snapshots of the resulting motion, that show the quadrotor that avoids a series of obstacles that progressively obstruct its path toward the goal. The results show that the vehicle is able to perform fast maneuvers around the obstacles. The robot slows down only marginally while performing the most demanding obstacle avoidance maneuvers (i.e., while avoiding the first two obstacles at $t \approx 1$ s and $t \approx 2$ s) maintaining a linear velocity $\|v_c\|$ above 1.1 m/s and 1.5 m/s. Moreover, a velocity of over 2 m/s is maintained while navigating around the fourth and last obstacle (see the velocity profile in Fig. 13). The dynamic nature of the motion is also shown in Fig. 14, where we can observe how after avoiding the first obstacle (second snapshot) the robot performs a quick pitch to accelerate forward. Finally, in Fig. 15 we report the profile of the control inputs, where it is shown how the proposed method utilizes all of the available thrust range and saturates the roll and pitch torques τ_ϕ and τ_θ during the first instants of motion.

An animation of the generated motion can be found in simulation 8 of the accompanying video, along with additional simulation results (simulation 9) for a second scenario that considers five moving obstacles, which we omit here for compactness.

8. Integration of velocity fields

In this section, we work out and validate an extension of the basic dynamics-aware navigation method based on the integration of velocity fields.

8.1. Velocity fields

The above result show that the proposed robot navigation method can be effectively used in both static and dynamic environments, even in rather challenging conditions. However, in the course of our simulation campaign, there were cases in which the robot almost stopped in front of a fixed obstacle before circumventing it. Similarly, in the vicinity of moving obstacles, we noticed the robot backtracking before it starts circumventing them. A reason for this behavior can be the antagonistic nature of the terms in the cost function associated with the task error and the control effort. Although this behavior does not directly affect the effectiveness of the proposed method, one would clearly like to avoid it as the time needed for the robot to reach its goal increases. Obviously, appropriate tuning could improve this behavior, however, this is an environment-dependent process.

One could argue that APFs, very popular in robotics, provide a direct action aimed at steering the robot away from nearby obstacles; whereas optimization-based navigation methods, as those proposed on NMPC, only see the obstacles through constraints violation. This suggests the idea of integrating APFs into NMPCs schemes to combine the efficient short-range collision avoidance of the former with the superior lookahead capability of the latter.

Here, we are going to accordingly modify the proposed method taking inspiration from the concept of *vortex fields* [32]. The vortex fields were initially introduced as a local minima-free extension to APFs substituting the repulsive field assigned to the obstacles with flows that rotate around them. By following these flows the robot is able to circumvent the obstacles without getting trapped in local minima.

Based on this idea, from the obstacles that are considered for collision within the NMPC, we are going to pick the one that is more likely to violate the collision avoidance constraint (18). To this obstacle, we will assign a velocity field that rotates around it. In this way, the robot can circumvent the obstacle by simply aligning its velocity with the field. To enable this robot action, we will consider the velocity of the field as a reference velocity for a representative point of the robot, in our case point C, and we will include an additional term in the cost functions (5) and (6) that penalize deviations from this reference. This term will continue influencing the solution of the NMPC until the considered obstacle stops being dangerous or a solution that circumvents the obstacle has been found.

Specifically, consider the robot being at state $\boldsymbol{x}_k$ at time instant t_k and the NLP to be solved at the same time instant. Collect the n_o obstacles that are considered by the NLP in the set $\bar{\mathcal{O}}_k = \{\mathcal{O}_1(t_k), \dots, \mathcal{O}_{n_o}(t_k)\}$. Denote by $\mathcal{O}_v \in \bar{\mathcal{O}}_k$ the obstacle to which we assign the velocity field. Since we aim to assign the velocity field to the obstacle in $\bar{\mathcal{O}}_k$ that is more likely to violate the collision avoidance constraint (18), $\mathcal{O}_v$ is defined according to

$$\mathcal{O}_v = \arg \max_{\mathcal{O}_j(t_k) \in \bar{\mathcal{O}}_k} \left\| \frac{\boldsymbol{u}_b(\boldsymbol{x}_k, \boldsymbol{\xi}_{j,k}) - \frac{\boldsymbol{u}_{\max} + \boldsymbol{u}_{\min}}{2}}{\boldsymbol{u}_{\max} - \boldsymbol{u}_{\min}} \right\|_\infty ,$$

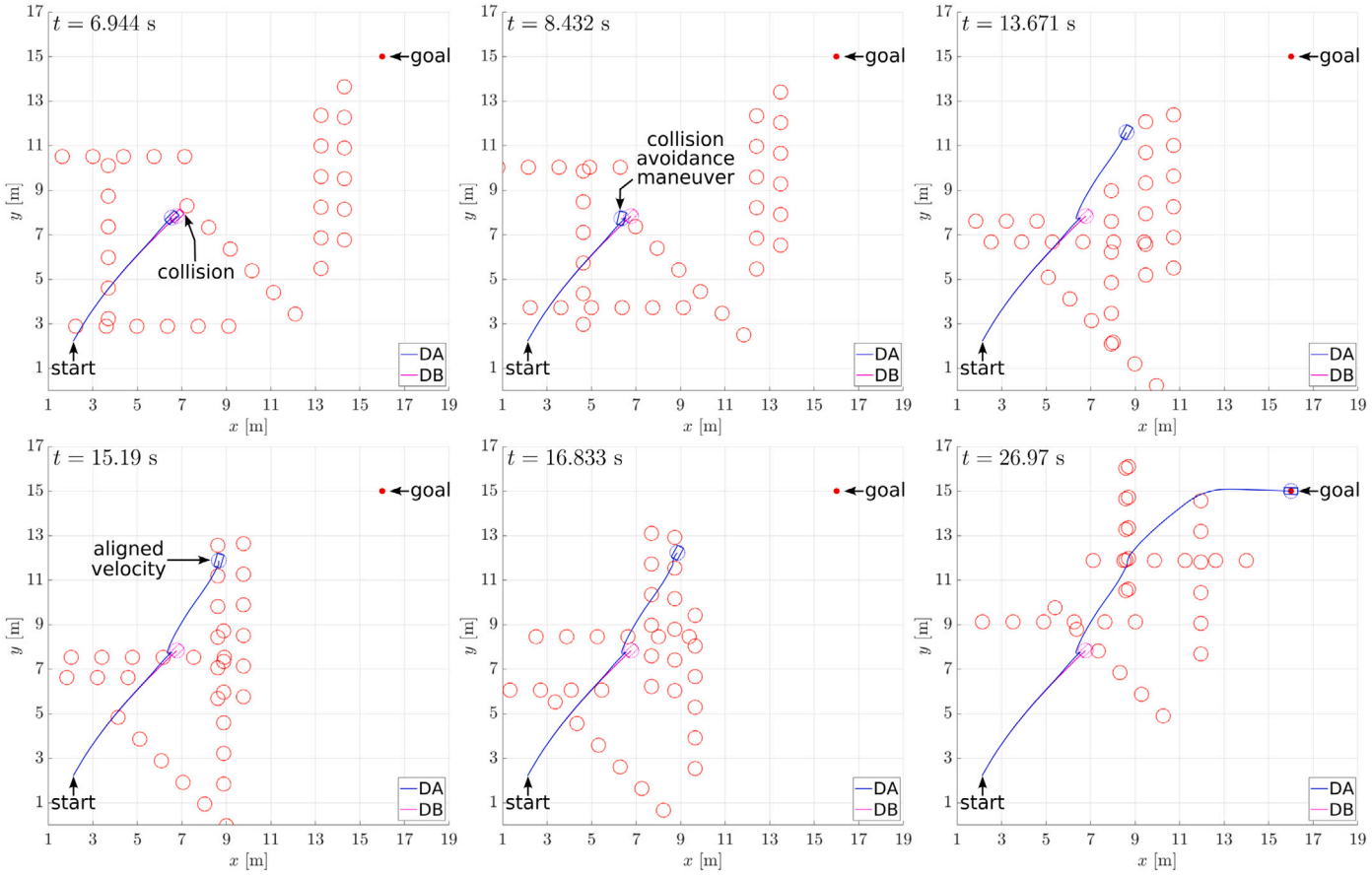


Fig. 8. Snapshots of the motion generated by the two methods in an environment occupied by obstacle moving in linear formation, with $v_{\max} = 1.3$ m/s. See also simulation 6 in the accompanying video.

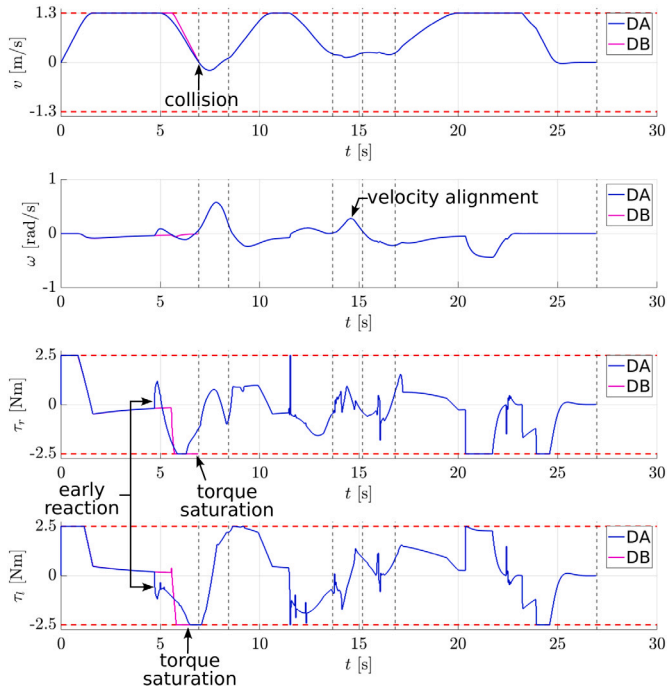


Fig. 9. The velocity profiles and the control inputs generated by the two methods in the environment occupied by obstacle moving in linear formation, with $v_{\max} = 1.3$ m/s.

with $\boldsymbol{\xi}_{j,k}$ being the state of the j-th obstacle at the considered time instant and the division between the vectors being the element-wise one. By O_v we denote the center of the bounding circle of $\mathcal{O}_v$ and by $\boldsymbol{o}_v$ the associated position vector.

The velocity field will be formed around the bounding circle of obstacle $\mathcal{O}_v$. If $\boldsymbol{p} = (x, y)$ is the position vector of a point P in the workspace, we define the velocity field around the obstacle as

$$\boldsymbol{\zeta}_v(\boldsymbol{p}, \boldsymbol{o}_v) = \pm\mu \begin{pmatrix} \frac{\partial \|\boldsymbol{p}-\boldsymbol{o}_v\|}{\partial y} & -\frac{\partial \|\boldsymbol{p}-\boldsymbol{o}_v\|}{\partial x} \end{pmatrix}^T,$$

where μ is a user-defined parameter. The sign indicates the rotation of the field, with the '+' corresponding to a counterclockwise (CCW) rotation and the '-' to a clockwise (CW) one. Since we aim to align the velocity of the robot representative point C with the velocity of the field, we will penalize its deviation by adding in the running (5) and terminal cost (6) of the NLP, respectively, the terms

$$(\dot{\boldsymbol{y}}_{k|i} - \boldsymbol{\zeta}_v(\boldsymbol{y}_{k|i}, \boldsymbol{o}_{v,k|i}))^T \boldsymbol{S} (\dot{\boldsymbol{y}}_{k|i} - \boldsymbol{\zeta}_v(\boldsymbol{y}_{k|i}, \boldsymbol{o}_{v,k|i}))$$

$$(\dot{\boldsymbol{y}}_{k|N} - \boldsymbol{\zeta}_v(\boldsymbol{y}_{k|N}, \boldsymbol{o}_{v,k|N}))^T \boldsymbol{S}_N (\dot{\boldsymbol{y}}_{k|N} - \boldsymbol{\zeta}_v(\boldsymbol{y}_{k|N}, \boldsymbol{o}_{v,k|N})),$$

where $\boldsymbol{S}$ and $\boldsymbol{S}_N$ the associated weighting matrices and $\boldsymbol{o}_{v,k|i}$ the position of O_v at the predicted time t_{k+i}. Note that $\boldsymbol{S}$ and $\boldsymbol{S}_N$ are chosen to be at least one order of magnitude smaller than $\boldsymbol{Q}$ and $\boldsymbol{Q}_N$ respectively. With this choice, we aim to prevent the terms associated with the velocity fields from dominating the cost function and thus the solution.

There are two points to be discussed regarding the integration of the velocity field:

- the selection of its direction of rotation;
- the relaxation of its influence to the solution.

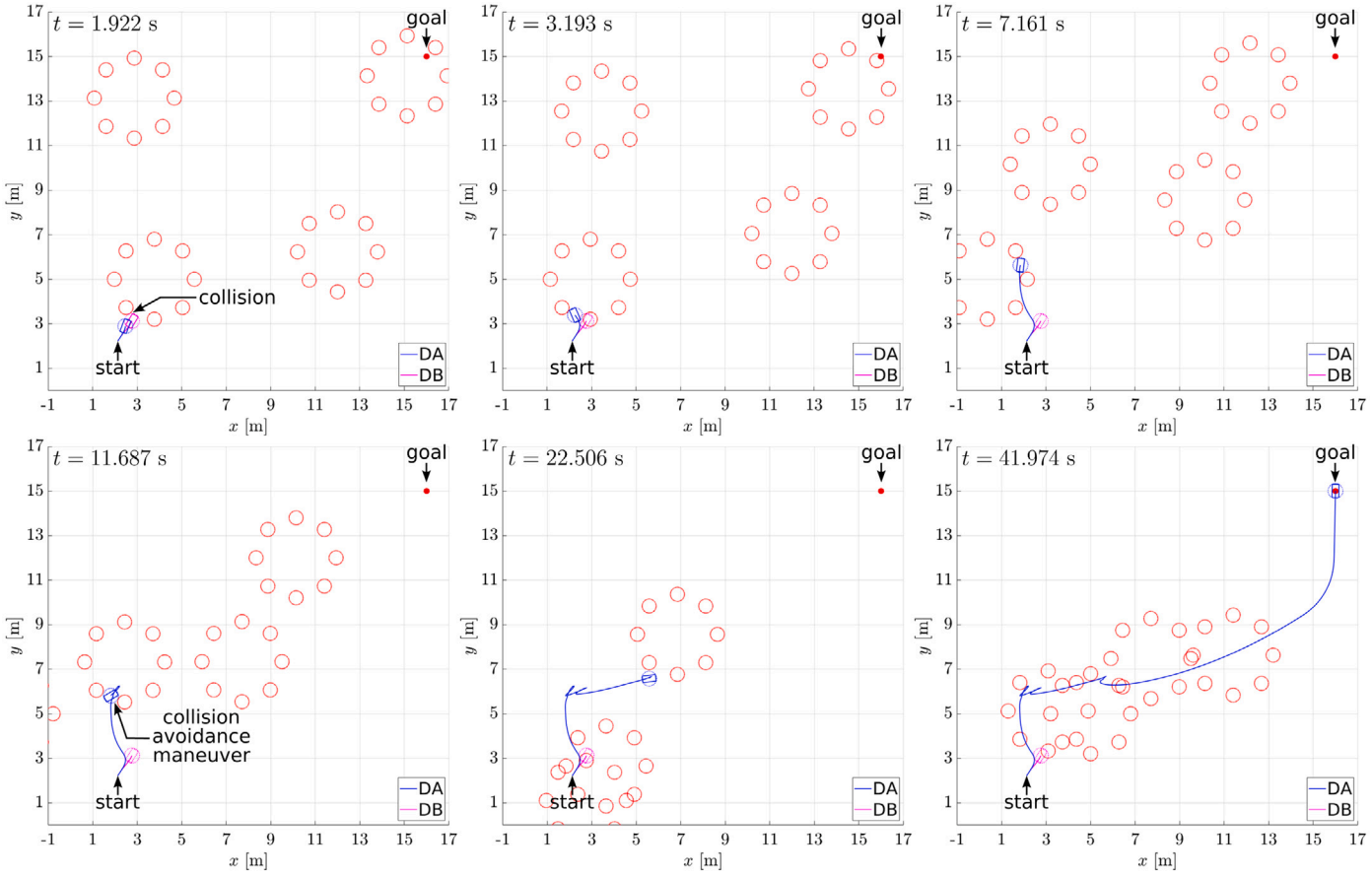


Fig. 10. Snapshots of the motion generated by the two methods in an environment occupied by obstacle moving in circular formation, with $v_{\max} = 1.3$ m/s. See also simulation 7 in the accompanying video.

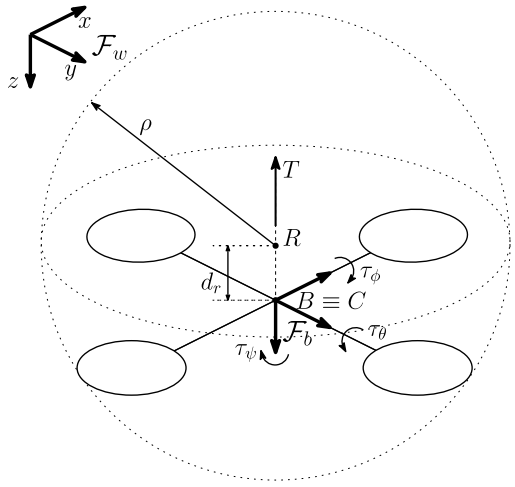


Fig. 11. The considered quadrotor UAV. With dashed line is illustrated the bounding sphere that is used for the collision avoidance constraint.

Direction of rotation: We remind that in this work we use as an initial guess for the NLP to be solved at time t_k the solution of the NLP that was solved at the previous control cycle at time t_{k-1}. If this initial guess corresponds to a motion that tends to go around the obstacle from a specific direction, it is reasonable to say that at the current control cycle, we would like to continue following the same direction. So for the direction of the velocity field at t_k, we take into account the motion resulting from the solution of the NLP at t_{k-1}, and specifically the predicted path of the representative point C. If $\boldsymbol{X}^*_{k-1}$ is the optimal state sequences obtained from the solution of the NLP at time t_{k-1}, by applying the forward kinematic map $\boldsymbol{y} = \boldsymbol{k}(\boldsymbol{x})$ to its elements we can express the predicted path of C at time t_{k-1} as the sequence $\boldsymbol{Y}^*_{k-1} = \{\boldsymbol{y}^*_{k-1|0}, \boldsymbol{y}^*_{k-1|1}, \dots, \boldsymbol{y}^*_{k-1|N}\}$. Considering now that given an accurate prediction model of the robot, the vector $\boldsymbol{y}^*_{k-1|1}$ corresponds to the position vector of C at t_k, we can use the vector $\boldsymbol{y}^*_{k-1|N} - \boldsymbol{y}^*_{k-1|1}$ as an indication of the intended direction of the robot motion with respect to the obstacle (see Fig. 16 for illustration). So the CCW direction for the velocity field, and thus the '+' sign, will be considered if the following condition is satisfied

$$(\boldsymbol{y}^*_{k-1|N} - \boldsymbol{y}^*_{k-1|1}) \times \boldsymbol{n}_v \geq \boldsymbol{0}, \tag{20}$$

where $\boldsymbol{n}_v$ is the unit vector pointing from C to O_v at t_k and is defined as

$$\boldsymbol{n}_v = \frac{\boldsymbol{o}_{v,k} - \boldsymbol{y}^*_{k-1|1}}{\|\boldsymbol{o}_{v,k} - \boldsymbol{y}^*_{k-1|1}\|}.$$

In a different case, the CW direction, and thus the '-' sign, will be considered.

Relaxation of field influence: We want the velocity field to influence the solution only if (i) the obstacle is in the vicinity of violating the collision avoidance constraint and (ii) a solution that circumvents the obstacle has not already been found. If $\boldsymbol{\xi}_{v,k}$ is the state at t_k of the obstacle to which the velocity field is assigned, then the field will actually influence the solution if the following conditions hold

$$\left\| \frac{\boldsymbol{u}_b(\boldsymbol{x}_k, \boldsymbol{\xi}_{v,k}) - \frac{\boldsymbol{u}_{\max} + \boldsymbol{u}_{\min}}{2}}{\boldsymbol{u}_{\max} - \boldsymbol{u}_{\min}} \right\|_\infty \geq \bar{\tau}, \tag{21}$$

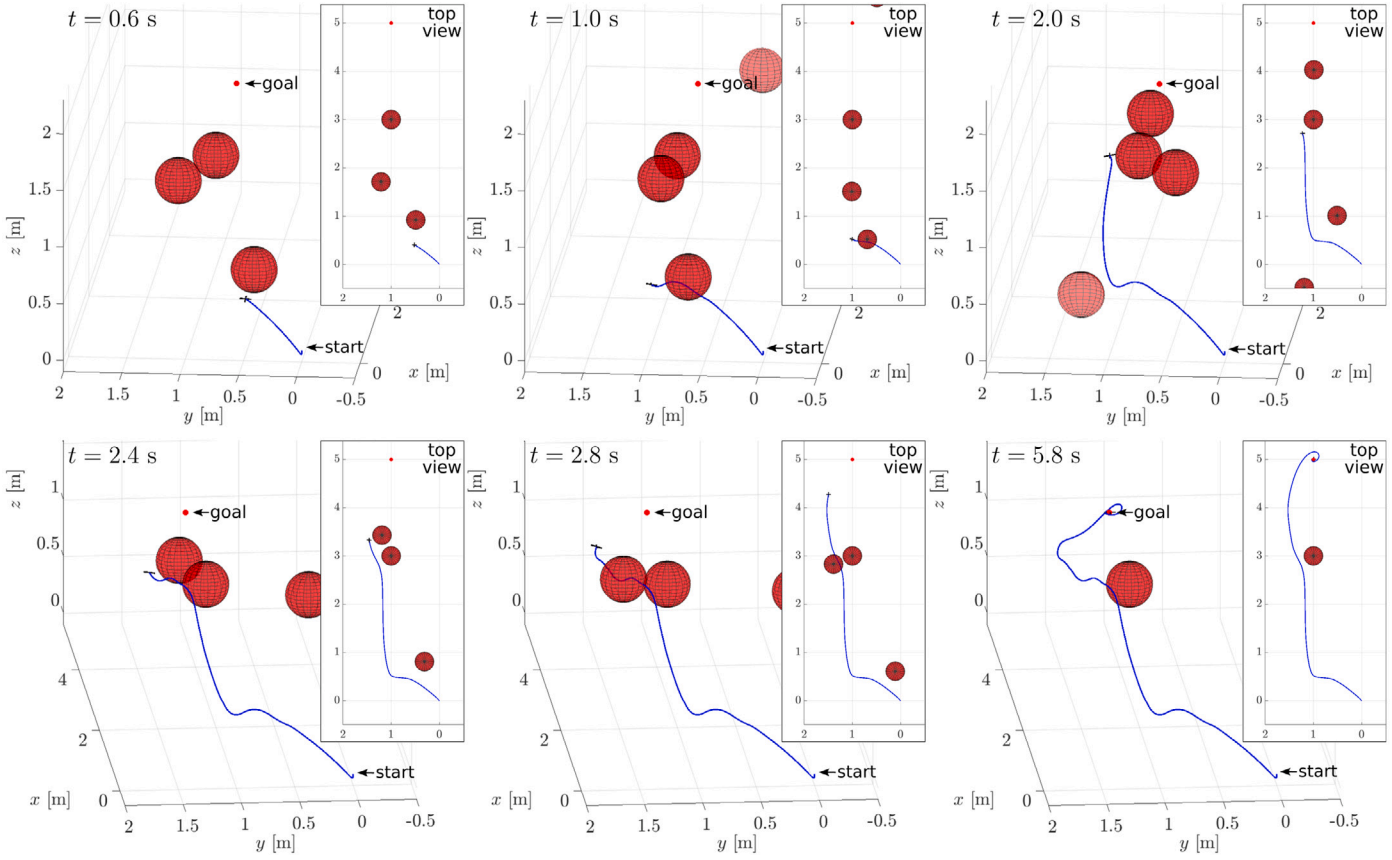


Fig. 12. Snapshots of the motion generated by the proposed method. The vehicle successfully navigates around the obstacles to reach the goal. Note that the top and bottom images are rendered with two different camera views to provide a clearer perspective. See also the accompanying video.

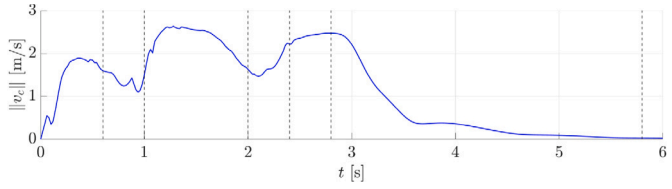


Fig. 13. The linear velocity profile generated by the proposed method. The robot maintains a high speed even while avoiding the obstacles.

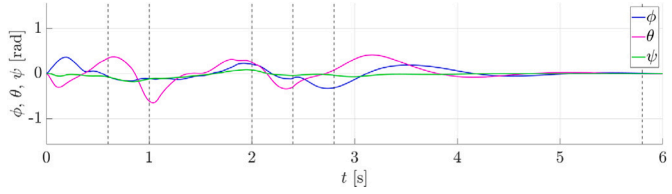


Fig. 14. The Euler angles profiles describing the attitude of the quadrotor during the simulation.

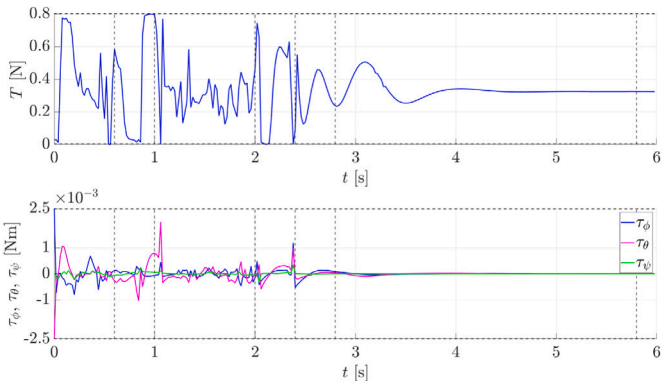


Fig. 15. The thrust (top) and torque (bottom) inputs generated by the proposed method. Note the saturation of the thrust (both lower and upper bounds) occurring multiple times during the simulation and of two of the torque components during the first instants of motion.

$$\|\boldsymbol{o}_{v,k|N-1} - \boldsymbol{y}^*_{k-1|1}\| - \boldsymbol{\eta}^T(\boldsymbol{y}_{k-1|N} - \boldsymbol{y}^*_{k-1|1}) \geq 0, \tag{22}$$

with

$$\boldsymbol{\eta} = \frac{\boldsymbol{o}_{v,k|N-1} - \boldsymbol{y}^*_{k-1|1}}{\|\boldsymbol{o}_{v,k|N-1} - \boldsymbol{y}^*_{k-1|1}\|}$$

and $\bar{\tau}$ is a user-defined value.

Condition (21) ensures that the field influences the solution only when the obstacle is dangerous enough to violate the collision avoidance constraint, while it also sets a threshold under which the field is not necessary to be activated. Note that the smaller the value of $\bar{\tau}$ the earlier the robot will start circumventing the obstacle.

Regarding condition (22), it ensures that if the initial guess to the NLP, obtained from the solution at t_{k-1}, corresponds to a motion that is able to circumvent the predicted obstacle at the end of the prediction horizon, then the velocity field will not influence the solution at the current control cycle. More precisely, we consider a trajectory to

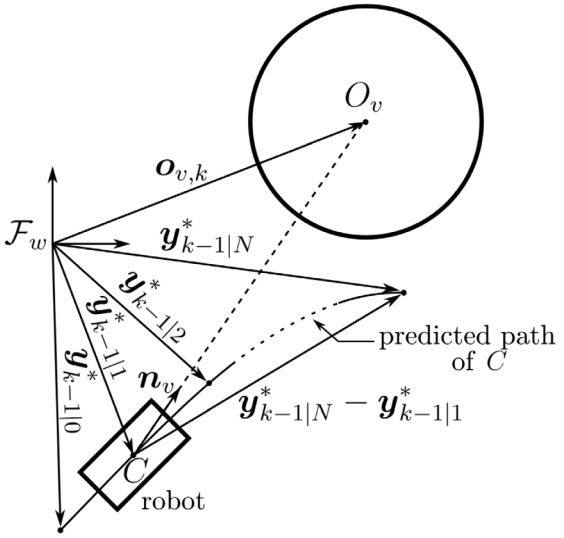


Fig. 16. The rotation of the velocity field at t_k will be CCW, if the predicted path of the representative point C at time t_{k-1} tends to go around the obstacle from its right, as illustrated in the figure. Alternatively, the rotation will be CW.

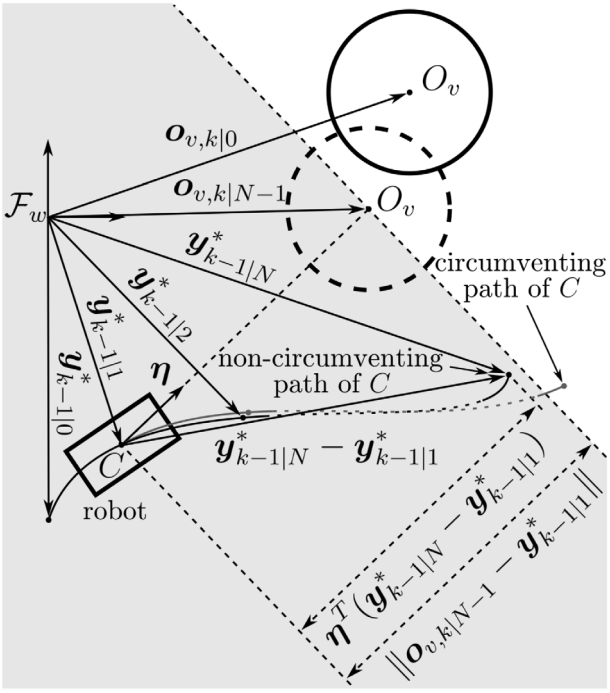


Fig. 17. The gray area represents the half-plane in which C needs to lie inside at the predicted time instant t_{k+N-1} in order for the field to be active at t_k. Note that if C lies outside the half-plane at t_{k+N-1} then we consider that the computed motion at t_{k-1} circumvents the obstacle (gray line) and thus the field will be deactivated.

circumvent the obstacle if it is such that the minimal arc between the projection of point C at time t_k and at time t_{k+N-1} on the bounding circle of the obstacle spans an angle greater than 90°. So, considering the predicted position of the center O_v of the obstacle at time t_{k+N-1}, condition (22) defines a half-plane in which if the point C lies in at time t_{k+N-1}, the minimal arc on the bounding circle of the obstacle is lower than 90° and thus the predicted motion does not circumvent the obstacle (see Fig. 17 for illustration). Violation of condition (22) would mean that a path that circumvents the obstacle has already been found.

Note that if any of the conditions (21) and (22) is violated, then we prevent the field from influencing the solution by setting $\boldsymbol{S} = \boldsymbol{S}_N = \boldsymbol{0}_{2\times 2}$, until the conditions are both satisfied again.

8.2. Simulations

To show the advantages of this modified version of the proposed method (DA-V), we will compare it with the original method (DA) in one of the dynamic environments in which we observed its particular behavior. Note that for this simulation we consider the differential-drive robot of Section 6.1, however, the modification is general and can be applied to flying robots as well. For the considered scenario, the robot maximum driving velocity is set to $v_{\max} = 0.9$ m/s, while the moving obstacles travel at constant speed $v_o = 0.45$ m/s with direction changes of 60° toward the robot. For both methods, the prediction horizon is set at $H = 0.93$ s. For the DA-V we set $\mu = 3$ and $\bar{\tau} = 0.25$. In Fig. 18 we show snapshots of the resulting robot motion under the two methods (see also simulation 10 in the accompanying video). The first snapshot illustrates the situation in which the DA method almost stops in front of a static obstacle before it starts the maneuver to circumvent it (see also the associated velocity profile in Fig. 19). On the other hand, DA-V under the influence of the velocity field initiates the maneuver to circumvent the obstacle earlier and so it does not have to decelerate significantly. The field stops influencing the solution when a circumventing motion is found as it is evident after $t = 9.331$ s when the robot has already stopped rotating around the obstacle. Again in the presence of the moving obstacle, the influence of the velocity field is also evident with DA-V rotating CW around it. On the other hand, we can see that under the DA method, the robot backtracks in front of the same obstacle and performs a fast reorientation in order to avoid it. This leads the robot to follow a longer path than the one followed by the DA-V and to reach the goal 5 s later.

Finally, note that the benefits of the proposed modification come without an increase in the computational cost, as the maximum reported computational time was around 27 ms for both methods.

9. Conclusion

The navigation of robots in dynamic environments can be challenging, especially when they travel at a high speed. In this article, we presented a novel real-time NMPC approach suitable for this kind of scenarios. In an attempt to avoid using collision avoidance constraints based on purely distance information, 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 proposed method applied to a differential-drive robot was tested extensively in both static and dynamic environments, while its performance was compared with the one of an NMPC that considers a purely distance-based collision avoidance constraint. The results indicate the effectiveness of the proposed method and its superiority over its distance-based counterpart, especially when the robot navigates at high speed in cluttered dynamic environments.

To further assess its performance, the method was applied to a flying robot moving in dynamic environments. The simulation results revealed the effectiveness of the proposed method, showing also that it is suitable for agile robotic platforms.

Finally, the modification of the proposed method with the integration of the velocity field improved its performance in the vicinity of the obstacles without adding any significant computational cost.

Future work will aim at the experimental validation of the proposed method in human-crowded environments, as well as its application to multi-body robots (e.g., mobile manipulators).

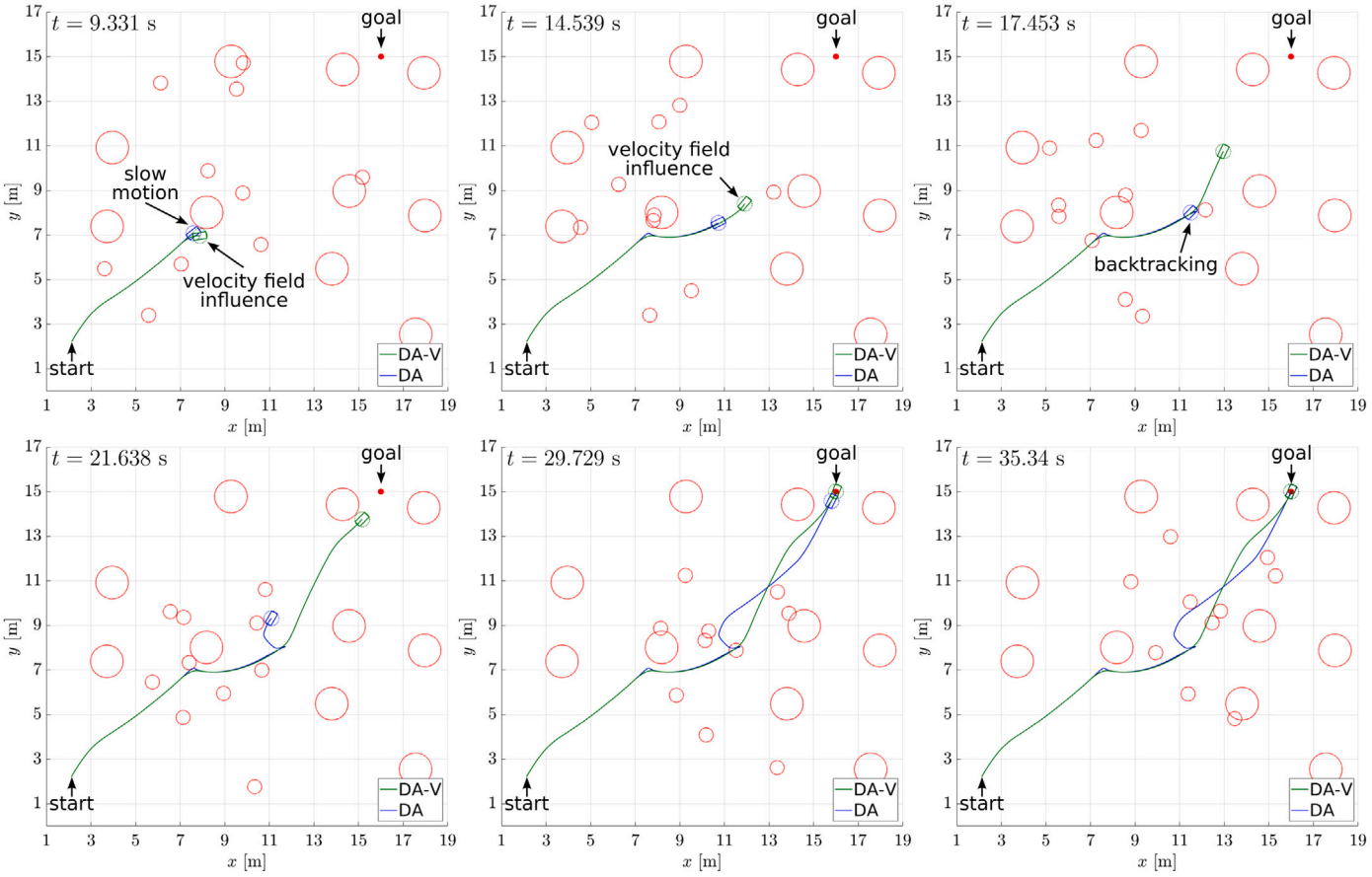


Fig. 18. Snapshots of the motion generated by the proposed method DA and the modified version with the velocity field integration DA-V in one of the dynamic environments, with $v_{max} = 0.9$ m/s.

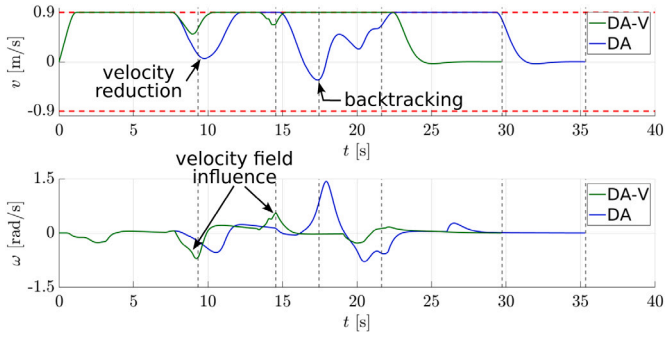


Fig. 19. The velocity profiles generated by DA and DA-V in one of the dynamic environments, with $v_{max} = 0.9$ m/s.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This work was performed while Spyridon G. Tarantos was a Ph.D. student at the Department of Computer, Control and Management Engineering, Sapienza University of Rome.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2023.104582.

References

[1] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, in: 1985 IEEE Int. Conf. on Robotics and Automation, Vol. 2, 1985, pp. 500–505.
[2] P. Fiorini, Z. Shiller, Motion planning in dynamic environments using velocity obstacles, Int. J. Robotics Res. 17 (7) (1998) 760–772.
[3] D. Fox, W. Burgard, S. Thrun, The dynamic window approach to collision avoidance, IEEE Robot. Autom. Mag. 4 (1) (1997) 23–33.
[4] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, J.P. How, Real-time motion planning with applications to autonomous urban driving, IEEE Trans. Control Syst. Technol. 17 (5) (2009) 1105–1118.
[5] M. Diehl, H. Bock, J.P. Schlöder, R. Findeisen, Z. Nagy, F. Allgöwer, Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations, J. Process Control 12 (4) (2002) 577–585.
[6] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, M. Diehl, acados – A modular open-source framework for fast embedded optimal control, Math. Program. Comput. (2021).
[7] X. Zhang, A. Liniger, F. Borrelli, Optimization-based collision avoidance, IEEE Trans. Control Syst. Technol. 29 (3) (2021) 972–983.
[8] J.V. Frasch, A. Gray, M. Zanon, H.J. Ferreau, S. Sager, F. Borrelli, M. Diehl, An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles, in: 2013 European Control Conference, 2013, pp. 4136–4141.

[9] A. Liniger, A. Domahidi, M. Morari, Optimization-based autonomous racing of 1:43 scale RC cars, Optim. Control Appl. Methods 36 (5) (2015) 628–647.
[10] A. Sathya, P. Sopasakis, R. Van Parys, A. Themelis, G. Pipeleers, P. Patrinos, Embedded nonlinear model predictive control for obstacle avoidance using PANOC, in: 2018 European Control Conference, 2018, pp. 1523–1528.
[11] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, P. Abbeel, Motion planning with sequential convex optimization and convex collision checking, Int. J. Robotics Res. 33 (9) (2014) 1251–1270.
[12] R. Bonalli, A. Cauligi, A. Bylard, M. Pavone, GuSTO: Guaranteed sequential trajectory optimization via sequential convex programming, in: 2019 Int. Conf. on Robotics and Automation, 2019, pp. 6741–6747.
[13] H. Febbo, J. Liu, P. Jayakumar, J.L. Stein, T. Ersal, Moving obstacle avoidance for large, high-speed autonomous ground vehicles, in: 2017 American Control Conference, 2017, pp. 5568–5573.
[14] C. Jewison, R.S. Erwin, A. Saenz-Otero, Model predictive control with ellipsoid obstacle constraints for spacecraft rendezvous, IFAC-PapersOnLine 48 (9) (2015) 257–262, 1st IFAC Workshop on Advanced Control and Navigation for Autonomous Aerospace Vehicles ACNAAV'15.
[15] H. Febbo, P. Jayakumar, J.L. Stein, T. Ersal, Real-Time Trajectory Planning for Automated Vehicle Safety and Performance in Dynamic Environments, J. Autonomous Veh. Syst. 1 (4) (2021) 041001.
[16] B. Brito, B. Floor, L. Ferranti, J. Alonso-Mora, Model predictive contouring control for collision avoidance in unstructured dynamic environments, IEEE Robot. Autom. Lett. 4 (4) (2019) 4459–4466.
[17] O. Gal, Z. Shiller, E. Rimon, Efficient and safe on-line motion planning in dynamic environments, in: 2009 IEEE Int. Conf. on Robotics and Automation, 2009, pp. 88–93.
[18] B. Damas, J. Santos-Victor, Avoiding moving obstacles: the forbidden velocity map, in: 2009 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2009, pp. 4393–4398.
[19] T. Fraichard, H. Asama, Inevitable collision states. a step towards safer robots? in: 2003 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Vol. 1, 2003, pp. 388–393.
[20] S. Petti, T. Fraichard, Safe motion planning in dynamic environments, in: 2005 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2005, pp. 2210–2215.
[21] S.M. LaValle, J.J. Kuffner, Jr., Randomized kinodynamic planning, Int. J. Robotics Res. 20 (5) (2001) 378–400.
[22] G. Buizza Avanzini, A.M. Zanchettin, P. Rocco, Constrained model predictive control for mobile robotic manipulators, Robotica 36 (1) (2018) 19–38.
[23] S.G. Tarantos, G. Oriolo, A dynamics-aware NMPC method for robot navigation among moving obstacles, in: Int. Conf. on Intelligent Autonomous Systems, 2022, pp. 129–143.
[24] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, Robotics: Modelling, Planning and Control, Springer, London, 2009.
[25] J. Rawlings, D. Mayne, M. Diehl, Model Predictive Control: Theory, Computation, and Design, Nob Hill Publishing, 2017.
[26] S. Sun, A. Romero, P. Foehn, E. Kaufmann, D. Scaramuzza, A comparative study of nonlinear mpc and differential-flatness-based control for quadrotor agile flight, IEEE Trans. Robot. 38 (6) (2022) 3357–3373.
[27] L. Biegler, X. Yang, G. Fischer, Advances in sensitivity-based nonlinear model predictive control and dynamic real-time optimization, J. Process Control 30 (2015) 104–116.
[28] A. Eqtami, D.V. Dimarogonas, K.J. Kyriakopoulos, Novel event-triggered strategies for model predictive controllers, in: 2011 50th IEEE Conference on Decision and Control and European Control Conference, IEEE, 2011, pp. 3392–3397.
[29] B. Houska, H.J. Ferreau, M. Diehl, An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range, Automatica 47 (10) (2011) 2279–2285.
[30] J.A.E. Andersson, J. Gillis, G. Horn, J.B. Rawlings, M. Diehl, CasADi – A software framework for nonlinear optimization and optimal control, Math. Program. Comput. 11 (1) (2019) 1–36.
[31] A. Wächter, L.T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, Math. Program. 106 (1) (2006) 25–57.
[32] C. De Medio, G. Oriolo, Robot obstacle avoidance using vortex fields, in: S. Stifter, J. Lenarčič (Eds.), Advances in Robot Kinematics, Springer Vienna, Vienna, 1991, pp. 227–235.

Spyridon G. Tarantos received his Diploma in Mechanical Engineering in 2017 from National Technical University of Athens, Greece, and his Ph.D. in Automatic Control in 2023 from Sapienza University of Rome, Italy. He is currently Post-Doctoral Associate at New York University Abu Dhabi. His research interests lie in the area of robotics and control with an emphasis on optimization-based motion generation.

Tommaso Belvedere received the bachelor's degree in Electronics Engineering and the master's degree in Control Engineering, respectively in 2018 and 2020, from Sapienza University of Rome, Italy, where he is currently working toward the Ph.D. degree in Control Engineering. His research interests focus on optimization-based control of mobile robotic systems.

Giuseppe Oriolo (S'89-M'92-SM'02-F'16) received his Ph.D. degree in Control Engineering in 1992 from Sapienza University of Rome, Italy. He is currently with the Department of Computer, Control and Management Engineering (DIAG) of the same university, where he is a Full Professor of automatic control and robotics and the director of the DIAG Robotics Lab. His research interests are in the general area of planning and control of robotic systems. Prof. Oriolo has been Associate Editor of the IEEE Transactions on Robotics and Automation from 2001 to 2005 and Editor of the IEEE Transactions on Robotics from 2009 to 2013. He is a Fellow of the IEEE.