Multitrajectory Model Predictive Control for Safe UAV Navigation in an Unknown Environment

Danilo Saccani, *Graduate Student Member, IEEE*, Leonardo Cecchin, and Lorenzo Fagiano, *Senior Member, IEEE*

***Abstract*— The problem of navigating an unmanned aerial vehicle (UAV) in an unknown environment is addressed with a novel model predictive control (MPC) formulation, named multitrajectory MPC (mt-MPC). The objective is to safely drive the vehicle to the desired target location by relying only on the partial description of the surroundings provided by an exteroceptive sensor. This information results in time-varying constraints during the navigation among obstacles. The proposed mt-MPC generates a sequence of position set points that are fed to control loops at lower hierarchical levels. To do so, the mt-MPC predicts two different state trajectories, a safe one and an exploiting one, in the same finite horizon optimal control problem (FHOCP). This formulation, particularly suitable for problems with uncertain time-varying constraints, allows one to partially decouple constraint satisfaction (safety) from cost function minimization (exploitation). Uncertainty due to modeling errors and sensors noise is taken into account as well, in a set membership (SM) framework. Theoretical guarantees of persistent obstacle avoidance are derived under suitable assumptions, and the approach is demonstrated experimentally out-of-the-laboratory on a prototype built with off-the-shelf components.**

***Index Terms*— Learning for control, model predictive control (MPC), safe autonomous navigation, uncertainty quantification, unmanned aerial vehicles (UAVs).**

I. Introduction

IN THE last decade, technological advancements have allowed unmanned aerial vehicles (UAVs) to become more and more common in our everyday life [1], [2]. Great research progress has been made across multiple areas, showing that relatively cheap civil drones can take off, carry out complex missions, and land without any human intervention. At the same time, autonomous UAV missions belong to the spectrum of safety-critical applications, where the use of algorithms that do not account for uncertainty and robust constraint satisfaction can lead to catastrophic effects, such as injury to people, loss or harm to property/equipment, or environmental damage. The design of motion planning algorithms in safety-critical applications must trade-off a risk-aware approach, which guarantees the safety of the system and the environment, and the exploitation of the vehicle capabilities without falling into a too conservative behavior. We consider here the problem of guaranteeing collision-free autonomous navigation from an initial point to a target position in an unknown environment. This problem presents several challenges: from rather usual requirements, such as the compliance with actuator constraints and vehicle dynamics [3], to the need to guarantee collision avoidance despite uncertainties both in the environment and in the system dynamics and sensing capabilities, at the same time keeping computational complexity small enough to enable the real-time applicability of the algorithm in real-world tasks.

A. Related Work

When a description of the environment is available, different approaches can be found in the literature to provide a collision-free path [4]. Sampling-based techniques, such as potential field methods [5], [6], cell decomposition [7], or roadmaps (e.g., A* [8], rapidly exploring random trees [9]), have been widely studied, providing a reliable way to perform offline path planning.

To include the vehicle's dynamics and manage constraints, model predictive control (MPC) [10], [11] has been investigated as well. Robust MPC approaches have been widely explored to consider disturbances and/or model mismatch. Robustness is usually achieved by tightening the constraints in the optimization, as shown in [12], [13], and [14]. Bemporad et al. [10] and Richards and How [15] use an integer variable to account for the intersection of the predicted trajectory with an obstacle, obtaining in this way a mixed-integer quadratic or linear program (MIQP or MILP) to be solved at each time step in a receding horizon fashion. By under-approximating the free space with convex polytopes, instead, it is possible to solve the problem without using integer variables (see, e.g., [16], [17]). The resulting linear, but time-variant constraints allow one to use a linear time-varying MPC formulation to compute a dynamically feasible and collision-free trajectory. The main drawback of these approaches is the need for a suitable discretization of the environment and the increasing complexity of the problem

Manuscript received 21 January 2022; revised 13 June 2022; accepted 30 September 2022. This work was supported in part by the Italian Ministry of University and Research (MUR), PRIN 2017 program, under Grant 201732RS94 "Systems of Tethered Multicopters," and in part by the European Union's Horizon 2020 Research and Innovation Program, under the Marie Skłodowska–Curie Grant 953348 "Embedded Learning and Optimization for the Next Generation of Smart Industrial Control Systems (ELO-X)." Recommended by Associate Editor F. Dabbene. *(Corresponding author: Lorenzo Fagiano.)*

Danilo Saccani and Lorenzo Fagiano are with the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, 20133 Milan, Italy (e-mail: danilo.saccani@polimi.it; lorenzo.fagiano@polimi.it).

Leonardo Cecchin is with Robert Bosch GmbH, 70465 Stuttgart, Germany (e-mail: leonardo.cecchin@de.bosch.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCST.2022.3216989.

Digital Object Identifier 10.1109/TCST.2022.3216989

with the number of obstacles and/or convex regions used in the approximation. The inner approximation with convex sets of safe regions has been widely studied as well. Several optimization-based approaches have been proposed (see, e.g., [16], [18], [19]) where the position of known obstacles is exploited to obtain the largest convex set in the free area. These approaches, however, do not guarantee the belonging of the current vehicle's position to the obtained set, and they require an a priori description of the environment, as a set of convex polytopes or as a map. On the contrary, in this work, we assume that no map is initially available and exploit only local sensor measurements.

Indeed, when the environment is partially or totally unknown, or it could be different from the prior information, the system must rely on local information collected by the available sensors to compute a feasible trajectory. Different approaches can be found in the literature that exploit directly the available information provided by exteroceptive sensors either to build a local map of the environment [20], [21] or for reactive obstacle avoidance (see, e.g., [22], [23]). Liu et al. [20] use convex optimization to derive piecewise polynomial trajectories for the navigation of an UAV in a partially unknown environment. A 3-D light detection and ranging (LiDAR) is used to build a local map, which is employed to compute convex connected polyhedra modeling the obstacle-free space and considered as linear inequality constraints in a quadratic program (QP) for trajectory optimization. However, the trajectory planning approaches available in the literature, despite providing an obstacle-free trajectory, only consider constraints on kinematic quantities and do not include, in the problem setup, the vehicle's dynamical response and the related uncertainty, due to, e.g., model mismatch or external disturbances. Other existing works dealing with this problem adopt reinforcement learning (RL) methods [24], [25], which employ directly the sensor measurements. All these methods often obtain good performance and the planning of obstacle-free trajectories but with little regard to safety guarantees, here considered in the form of constraint satisfaction and persistent obstacle avoidance. Other approaches exploit prior knowledge about the system to ensure safety, by combining optimal control and learning (see, e.g., [26], [27], [28]). This problem is particularly challenging, since the feasible path must be replanned online, as new parts of the environment are discovered. In this case, the control logic has to balance two conflicting aspects: *safety*, that is to avoid the online discovered obstacles, and *exploitation*, that is to reach the desired target in short time.

To deal with this balancing issue, in this work, we propose a novel MPC approach that we find particularly suitable for time-varying systems or constraints, named multitrajectory MPC (mt-MPC). To trade-off safety and exploitation in an intuitive way, the mt-MPC considers different future state trajectories in the same finite horizon optimal control problem (FHOCP), enabling a partial decoupling between constraint satisfaction (safety) and cost function minimization (exploitation). In [21] and [29], a control approach is proposed for the trajectory planning of a UAV equipped with a sensor that detects the surrounding partially unknown environment. The approach also relies in this case on a multitrajectory concept and proposes the application to several practical scenarios. On the other hand, the approach only considers a triple integrator as dynamics of the vehicle and does not provide theoretical guarantees on constraints satisfaction, which is one of the objectives of this work. A related MPC formulation has been proposed in [30], where the FHOCP trades-off the behavior of a nominal and a contingency model of a self-driving car. While Alsterda et al. [30] try to find a contingency maneuver in case of an unexpected change in the system, here, we consider multiple trajectories to exploit as much as possible the current knowledge of the environment and to reduce the conservatism of a guaranteed collision-free approach. Finally, a preliminary version of the mt-MPC approach appeared in our recent work [31], where a 2-D navigation problem is considered and a simulation study is presented. In this article, we deliver many additional contributions, as described next.

B. Contributions

We propose and demonstrate experimentally the use of mt-MPC to drive safely a multicopter drone, equipped with an exteroceptive sensor, to a goal point in an a priori unknown environment. The control structure is hierarchical: at low level, state-feedback controllers stabilize the vehicle's trajectories and track the set points provided by the high-level mt-MPC. The navigation in an unknown environment leads to time-varying constraints, such that standard receding horizon strategies do not guarantee recursive feasibility anymore. To address this problem, we propose a modified receding horizon implementation and prove the existence of a feasible trajectory at each time step, hence persistent obstacle avoidance, under the assumption of time invariant environment. To guarantee this property also in the presence of model-plant mismatch and disturbances, we quantify the model uncertainty in terms of bounds on the prediction error by exploiting a set membership (SM) framework [32], [33]. Thus, we address both environment uncertainty and model uncertainty, at the same time providing a method to quantify the latter from experimental data. All these aspects are relevant in real-world applications, yet they are rarely considered altogether in previous contributions, where the focus is either on the environment or on robust control starting from a given uncertainty model (e.g., disturbance bounds or model sets) without mentioning how this is derived. The resulting MPC law is a dynamic state-feedback one, in contrast with most of the literature where a static state-feedback controller is obtained. In addition, we also address two application-specific problems: the need to derive a convex under-approximation of the free space around the drone exploiting only local sensor measurements, in order to formulate the FHOCP as a convex QP, and the need to navigate around obstacles that stand between the drone and its target. Regarding the approximation of the free space, we present an approach that is computationally efficient and guarantees that the drone belongs to the derived set, which is needed to formally guarantee obstacle avoidance. Compared with our preliminary work [31], the main novel contributions are as follows: the quantification of model uncertainty from data, the development and theoretical analysis of a mt-MPC approach

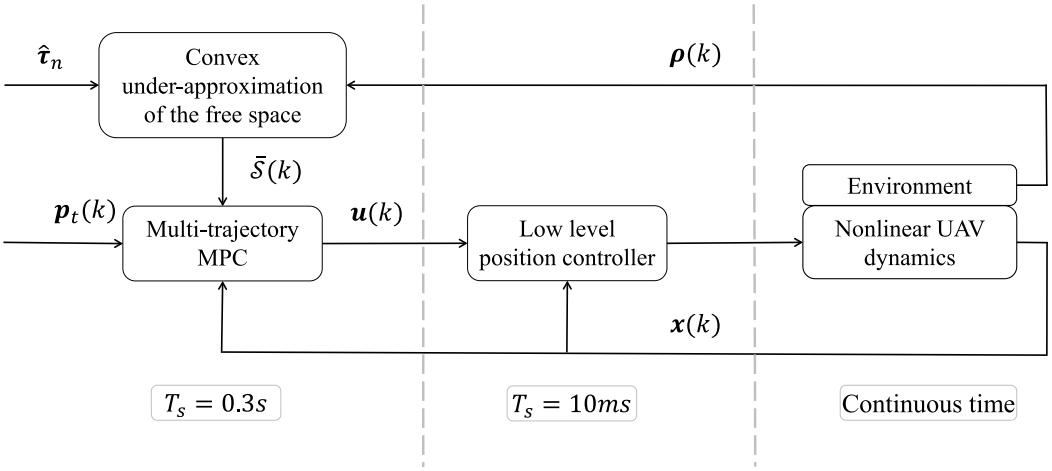


Fig. 1. Hierarchical structure of the UAV control system and of the proposed approach: an existing low-level position controller tracks the reference signals provided by a high-level navigation controller. The latter elaborates the sensor readings $\boldsymbol{\rho}(k)$ (Section II-B) and exploits a bound $\hat{\boldsymbol{\tau}}_n$ on the trajectory uncertainty (Section III-C) to build a convex under-approximation of the feasible space $\bar{\mathcal{S}}(k)$ (Section III-D), which is employed, together with a simplified, control-oriented model of the drone (Section III-A), by the state-feedback mt-MPC strategy to compute the reference position $\boldsymbol{u}(k)$ (Section IV). The sampling frequencies employed in our experimental tests are indicated as well.

that guarantees robust obstacle avoidance against such uncertainty, the extension to 3-D motion, and the experimental test on a real-world, out-of-the-laboratory drone prototype.

This article is organized as follows. After an overview of the system and the problem formulation in Section II, in Section III, we describe the employed control-oriented model, characterize the model uncertainty, and present the approach to derive a convex under-approximation of the free space, in order to plan safe trajectories. Then, Section IV is concerned with the mt-MPC formulation and its properties. Finally, Sections V and VI provide the simulation and experimental results, respectively, and Section VII concludes this article.

C. Notation

We denote with $t \in \mathbb{R}$ the continuous time variable, with $k \in \mathbb{Z}$ the discrete time index with sampling time T_s, with $\mathbb{N}$ the set of positive integers, and $\mathbb{N}_a^b = \{n \in \mathbb{N} \mid a \leq n \leq b\}$. Bold symbols indicate vectors, and $\cdot^T$ is the matrix transpose operation. $0^{a\times b}$ and $\mathbb{I}^a$ denote a matrix of zeros with a rows and b columns and the a-by-a identity matrix, respectively. $||\boldsymbol{v}|| = (\boldsymbol{v}^T\boldsymbol{v})^{1/2}$ denotes the two norm of vector $\boldsymbol{v}$, and $||\boldsymbol{v}||_A = (\boldsymbol{v}^T A\boldsymbol{v})^{1/2}$ denotes the two norm of vector $\boldsymbol{v}$ weighted by matrix $A \geq 0$ (positive semidefinite). The notations $\tilde{\cdot}$, $\hat{\cdot}$ represent a sample and an estimate of a given variable, respectively. The sum of two sets $\mathcal{A}$, $\mathcal{B} \subseteq \mathbb{R}^n$, denoted as $\mathcal{A} \oplus \mathcal{B}$, is $\{\boldsymbol{a} + \boldsymbol{b} \mid \boldsymbol{a} \in \mathcal{A},\ \boldsymbol{b} \in \mathcal{B}\}$, while $\mathcal{A} \ominus \mathcal{B}$ is $\{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} + \boldsymbol{b} \in \mathcal{A},\ \forall \boldsymbol{b} \in \mathcal{B}\}$. For an ordered set of points $\mathcal{S}^v = \{\mathcal{S}_1^v, \ldots, \mathcal{S}_{n_v}^v\} \in \mathbb{R}^3$, we denote with $chull(\mathcal{S}^v)$ their convex hull. Finally, let $\mathcal{E}(W) = \{\boldsymbol{e} : \boldsymbol{e}^T W \boldsymbol{e} \leq 1\}$ be the ellipsoidal set centered at the origin with shape matrix $W = W^T > 0$.

II. System and Environment Models, Problem Formulation

The three main ingredients defining the problem at hand are as follows: the autonomous system, the exteroceptive sensor used to gather information about the environment, and the environment itself. These elements are presented in the Sections II-A–II-C culminating in a more precise problem formulation.

A. Multicopter Vehicle and Preliminary Dataset

We consider an inertial, right-handed coordinate system (x_f, y_f, z_f) with origin at ground level, (x_f, y_f) coordinates defining a plane parallel to the ground, and z_f coordinate positive above ground level. The autonomous vehicle features a hierarchical control structure. From the point of view of the high-level controller, the control input is the variable

$$\boldsymbol{u}(k) = \left[u^{x_f}(k),\ u^{y_f}(k),\ u^{z_f}(k)\right]^T = \boldsymbol{p}_{\text{ref}}(k) \in \mathbb{R}^3 \qquad (1)$$

which is the position reference provided to the low-level controller; see Fig. 1. The latter is in charge of tracking such a position reference. Note that any low-level controller (e.g., the one of a commercial flight controller, as in our experimental tests) can be considered, as long as it is able to stabilize the drone's trajectories and to track the given reference with good performance (possible tracking errors due to disturbances are captured by our method to quantify uncertainty, described in Section III-C). Neglecting the attitude dynamics that are managed by the low-level control loops, we can represent the feedback-controlled drone as a nonlinear time-invariant system featuring as state the vehicle position $\boldsymbol{p}(k)$ and velocity $\boldsymbol{v}(k)$

$$\boldsymbol{x}(k) = \left[p^{x_f}(k),\ p^{y_f}(k),\ p^{z_f}(k),\ v^{x_f}(k),\ v^{y_f}(k),\ v^{z_f}(k)\right]^T \quad (2)$$

and as input $\boldsymbol{u}(k)$; thus, we have $\boldsymbol{x}(k) \in \mathbb{R}^6$ and $\boldsymbol{u}(k) \in \mathbb{R}^3$. We also denote with $\bar{\boldsymbol{a}} \in \mathbb{R}^3$ and $\bar{\boldsymbol{v}} \in \mathbb{R}^3$ the vehicle's maximum acceleration and velocity, respectively. If the drone's yaw is relevant, for example, to point a given sensor to a desired direction, this can be easily added as further state. Here, for simplicity, we focus on missions that require the drone to reach a given target position without any yaw angle specification. We also assume that a finite number of measured pairs $(\tilde{\boldsymbol{u}}(k),\ \tilde{\boldsymbol{x}}(k))$ are available from preliminary tests, for

example, carried out by a human operator or, as in our experimental application, by automated step reference sequences. We denote the collected dataset as follows:

$$\tilde{\mathcal{M}} \doteq \left\{ (\tilde{\boldsymbol{u}}(j), \tilde{\boldsymbol{x}}(j)) \quad \forall j \in \mathbb{N}_0^{N_s} \right\} \tag{3}$$

where $N_s + 1$ is the number of input–output pairs in the dataset. We assume that these data are affected both by process disturbance (e.g., wind) and measurement noise.

Let $\boldsymbol{x}\ (k+j; \bar{\boldsymbol{u}})$, $j > 0$, be the state trajectory of the nonlinear system at time $k+j$, obtained by applying the constant reference position $\bar{\boldsymbol{u}}$ starting from the initial condition $\boldsymbol{x}(k)$. Let the scalar $r > 0$ be a given distance bound. We consider the following assumption on the feedback-controlled vehicle.

Assumption 1: There exists a convex set $\mathcal{P}_W$, such that

$$\forall r \quad \forall \boldsymbol{x}(k)\ \forall \bar{\boldsymbol{u}} : ||\boldsymbol{p}(k) - \bar{\boldsymbol{u}}|| \leq r,\ \exists \underline{j}(r) : \\ \times \left(\boldsymbol{x}(k+j; \bar{\boldsymbol{u}}) - \begin{bmatrix} \bar{\boldsymbol{u}} \\ \boldsymbol{0}^{3\times 1} \end{bmatrix} \right) \in \mathcal{P}_W \quad \forall j \geq \underline{j}(r).$$

Assumption 1 states that for any given distance r, there exists a time instant $\underline{j}(r)$, such that for all constant references $\bar{\boldsymbol{u}}$ that are closer than r to the starting position $\boldsymbol{p}(k)$, the vector of position tracking errors at time $k+j$ belongs to the convex set $\mathcal{P}_W, \forall j > \underline{j}(r)$.

This assumption is related to the stability of the trajectories of the system at hand and to the boundedness of exogenous process disturbances. In the considered application, it is a reasonable assumption, considering the presence of a stabilizing, reference-tracking low-level position controller. The convex set $\mathcal{P}_W$ can be estimated directly from the dataset (3), as shown in our experimental results in Section VI.

B. Sensor Setup

We assume to receive, at each time step k, the measurements of vehicle's position $\boldsymbol{p}(k)$ and velocity $\boldsymbol{v}(k)$ together with the readings of an exteroceptive sensor able to partially detect the surrounding obstacles. The vehicle position and velocity can be measured with the use of a global positioning system (GPS) device or estimated exploiting exteroceptive sensors and simultaneous localization and mapping (SLAM) approaches [34]. As for the exteroceptive sensor, we assume to receive a 3-D point cloud providing a discretization of the environment around the drone, as shown in Fig. 2. In practice, this can be achieved with 3-D LiDAR sensors and/or stereo-cameras. Without loss of generality, we assume to receive a grid of points lying on directions that are equally spaced over a unit-sphere centered at the drone's position, and we denote with $\boldsymbol{\rho}(k) \in \mathbb{R}^M$ the vector readings provided at time k by the sensor. The length of $\boldsymbol{\rho}(k)$ is $M = M_a M_e = (2\pi/\alpha_a)(\pi/\alpha_e)$, where α_a and α_e are the azimuth and elevation angular resolutions, respectively. Denoting with i_a and i_e two indexes spanning the sampled azimuth and elevation values, each measurement corresponds to a vector

$$\boldsymbol{s_i}(k) \\ = \rho_i(k) \begin{bmatrix} \cos(i_e\alpha_e)\cos(i_a\alpha_a) \\ \cos(i_e\alpha_e)\sin(i_a\alpha_a) \\ \sin(i_e\alpha_e) \end{bmatrix},\ i_a \in \mathbb{N}_0^{M_a-1},\ i_e \in \mathbb{N}_0^{M_e-1} \tag{4}$$

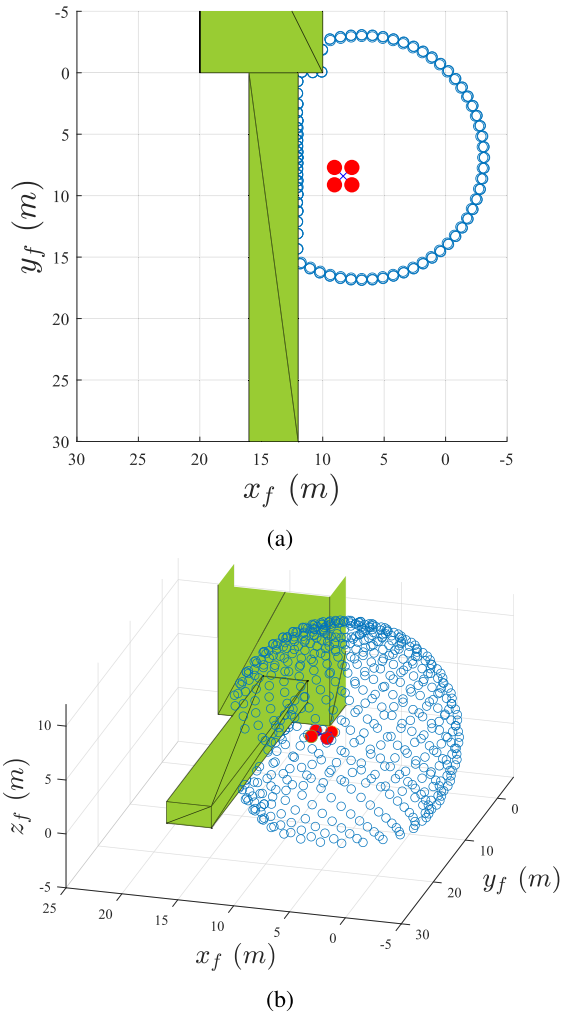


Fig. 2. Example of the point cloud obtained with the exteroceptive sensor. (a) 2-D view on the plane parallel to ground and containing the UAV position. (b) 3-D view.

where $\rho_i(k)$ is the ith entry of vector $\boldsymbol{\rho}(k)$ and $i = M_a i_e + i_a$. We finally denote with r_L the range of the sensor, assumed for simplicity to be the same along any direction.

C. Environment Model and Problem Formulation

We consider a time-invariant environment composed of N_o, generally non-convex, 3-D obstacles with variable cross section; see Fig. 3 for an example. In this framework, each obstacle can be described as a compact set $O_i \in \mathbb{R}^3$, $i = 1, \ldots, N_o$. We define the overall obstacles' set $\mathcal{O}$ as follows:

$$\mathcal{O} \doteq \bigcup_{i=1}^{N_o} O_i. \tag{5}$$

We are now in position to formalize the problem considered in this work. Given the dataset $\tilde{\mathcal{M}}$, identify a "control-oriented" model of the feedback-controlled vehicle together with a bound on the prediction error with respect to the real system. Then, exploiting these information, design a high-level discrete-time navigation logic that makes use of the sensor measurements $\boldsymbol{\rho}(k)$ and the state feedback $\boldsymbol{x}(k)$ to

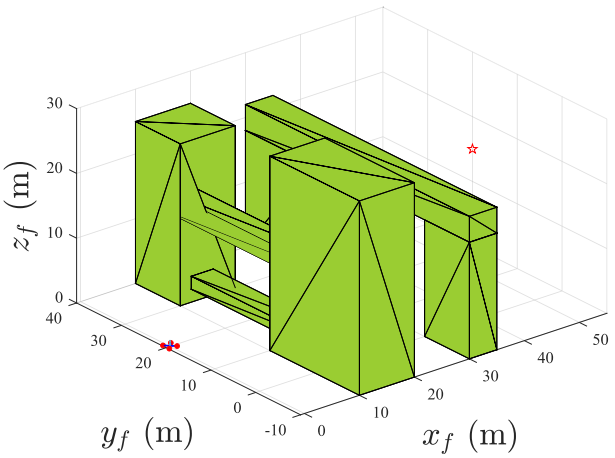


Fig. 3. Example of the considered environment.

compute, at each time step k, the input $\boldsymbol{u}(k)$ in order to move the UAV from a given initial position $\boldsymbol{p}(0) \notin \mathcal{O}$ toward a target position $\boldsymbol{p}_t(k) \in \mathbb{R}^3$ in a safe way, i.e., such that $\boldsymbol{p}(k) \notin \mathcal{O}, \forall k \geq 0$ despite the various sources of uncertainty.

III. Control-Oriented Model and Convex Approximation of the Environment

Two key elements to address the problem at hand are a suitable model of the system together with the associated uncertainty consistent with the dataset (3) and the exploitation of sensor measurements to describe the surrounding environment. We describe next these elements, which will be employed in the mt-MPC formulation described in Section IV.

A. Control-Oriented Model

The considered system is composed of the nonlinear dynamics of the drone in closed loop with the low-level position controller. From the point of view of the high-level navigation strategy, linear dynamics well approximate the behavior of the closed-loop system. Furthermore, the choice of a linear time-invariant (LTI) model allows us to cast the optimal control problem as a QP, that we can solve in real time also on relatively low-power hardware, as shown in Section VI. To obtain the control-oriented model, we consider the following continuous time double integrator with state feedback:

$$\begin{bmatrix}\dot{\boldsymbol{p}}(t)\\ \dot{\boldsymbol{v}}(t)\end{bmatrix} = \begin{bmatrix}0^{3\times3} & \mathbb{I}^3\\ -K_{\text{vel}}K_{\text{pos}} & -K_{\text{vel}}\end{bmatrix}\begin{bmatrix}\boldsymbol{p}(t)\\ \boldsymbol{v}(t)\end{bmatrix} + \begin{bmatrix}0^{3\times3}\\ K_{\text{vel}}K_{\text{pos}}\end{bmatrix}\boldsymbol{u}(t)$$
$$\boldsymbol{p}(t) = \underbrace{\begin{bmatrix}\mathbb{I}^3 & 0^{3\times3}\end{bmatrix}}_{C_p}\begin{bmatrix}\boldsymbol{p}(t)\\ \boldsymbol{v}(t)\end{bmatrix}$$
$$\boldsymbol{v}(t) = \underbrace{\begin{bmatrix}0^{3\times3} & \mathbb{I}^3\end{bmatrix}}_{C_v}\begin{bmatrix}\boldsymbol{p}(t)\\ \boldsymbol{v}(t)\end{bmatrix} \tag{6}$$

and we denote with

$$\boldsymbol{a}(t) = K_{\text{vel}}\big(K_{\text{pos}}(\boldsymbol{u}(t) - \boldsymbol{p}(t)) - \boldsymbol{v}(t)\big) \tag{7}$$

the drone acceleration vector. $K_{\text{pos}} \in \mathbb{R}^{3\times3}$ and $K_{\text{vel}} \in \mathbb{R}^{3\times3}$ are suitable gain matrices representing the position and velocity feedback loops that can be tuned by carrying out a system identification procedure to obtain a closed-loop system response as close as possible to the one of the actual nonlinear system. We then convert (6) to discrete time with sampling time T_s, obtaining the desired control-oriented model of the form

$$\boldsymbol{x}(k+1) = A\big(K_{\text{pos}}, K_{\text{vel}}\big)\boldsymbol{x}(k) + B\big(K_{\text{pos}}, K_{\text{vel}}\big)\boldsymbol{u}(k) \tag{8a}$$
$$\boldsymbol{p}(k) = C_p\boldsymbol{x}(k) \tag{8b}$$
$$\boldsymbol{v}(k) = C_v\boldsymbol{x}(k) \tag{8c}$$

where the input $\boldsymbol{u}(k)$ and the state $\boldsymbol{x}(k)$ of the position-controlled system are defined in (1) and (2). Thus, the LTI model (8) features $\boldsymbol{u}(k)$ as input, consistently with our setup, and its state includes both position and velocity, making it possible to easily include, in the predictive control strategy, constraints on these quantities and on the accelerations as well, through the linear, static equation (7). In turn, acceleration bounds can be chosen to account for the maximum limits on propellers' thrust and roll/pitch angles that are used to maneuver, thus achieving coherence between the LTI control-oriented model and the performance limits of the actual nonlinear system.

We consider the following assumption on the nominal model.

Assumption 2:

$$\forall \bar{\boldsymbol{u}} \quad \forall \boldsymbol{x}(k) : \left(\boldsymbol{x}(k) - \begin{bmatrix}\bar{\boldsymbol{u}}\\ \boldsymbol{0}^{3\times1}\end{bmatrix}\right) \in \mathcal{P}_W, \ \exists D = D^T > 0.$$

1) $A(K_{\text{pos}}, K_{\text{vel}})\boldsymbol{x}(k) + B(K_{\text{pos}}, K_{\text{vel}})\bar{\boldsymbol{u}} \in \mathcal{E}(D)$.
2) $\mathcal{P}_W \subseteq \mathcal{E}(D)$.
3) $\mathcal{E}(D) \subseteq \mathcal{V}$.

Here, $\mathcal{V}$ is the set of admissible states accounting for the drone's acceleration and velocity limits.

In practice, Assumption 2 is satisfied if the control-oriented model is asymptotically stable, and it can be verified by taking the sublevel set of the Lyapunov function $V(\boldsymbol{x}) = \boldsymbol{x}^T D\boldsymbol{x}$ that contains the polytope $\mathcal{P}_W$ in its interior (this condition can be checked via Linear Matrix Inequalities; see, e.g., [35]) and such that state constraints are satisfied for all points in it. Suitable convex programs can be used to check this condition as well, as shown in Section VI.

Remark 1: In order to obtain a convex QP in the final implementation, in the following, we will consider the convex polytopic outer approximations of the ellipsoidal set $\mathcal{E}(D)$, denoted as $\mathcal{P}_D$. Alternatively, one can retain an ellipse, which leads to a convex quadratically constrained QP (QCQP).

B. Identification of the Control-Oriented Model

Exploiting the dataset (3), we identify the gain matrices K_{pos} and K_{vel} of the control-oriented model. According to best practices [36], we divide the dataset in two parts: one is used for the identification phase (identification set), while performance is assessed on the remaining part (validation set). We carry out the identification with a simulation error

method (SEM)

$$\begin{aligned}\min_{K_{\text{pos}},K_{\text{vel}}} \quad & \sum_{k=1}^{N_s} \|\hat{\boldsymbol{x}}(k)-\tilde{\boldsymbol{x}}(k)\|^2 \\ \text{s.t.} \quad & \hat{\boldsymbol{x}}(0)=\tilde{\boldsymbol{x}}(0) \\ & \hat{\boldsymbol{x}}(k+1)=A\big(K_{\text{pos}},K_{\text{vel}}\big)\hat{\boldsymbol{x}}(k) \\ & +B\big(K_{\text{pos}},K_{\text{vel}}\big)\tilde{\boldsymbol{u}}(k) \forall k\in\mathbb{N}_0^{N_s-1}. \end{aligned} \tag{9}$$

The use of a simulation error criterion and the fact that $A(K_{\text{pos}}, K_{\text{vel}}) \in \mathbb{R}^{6\times 6}$ and $B(K_{\text{pos}}, K_{\text{vel}}) \in \mathbb{R}^{6\times 3}$ depend nonlinearly on K_{pos} and K_{vel} make (9) a non-LP (NLP).

Remark 2: For the sake of notational simplicity, from now on, we denote simply with A and B the matrices $A(K_{\text{pos}}, K_{\text{vel}})$ and $B(K_{\text{pos}}, K_{\text{vel}})$ with the parameters identified by solving the NLP (9).

C. Derivation of Prediction Uncertainty Bounds

The identified model is a linear, low dimensional approximation of the nonlinear system dynamics. To obtain a navigation logic able to robustly guarantee safety, we, thus, need to derive bounds on the prediction error due to linearization, process and measurement disturbances, and neglected dynamics. On the other hand, the uncertainty bounds shall be not too conservative, to avoid unnecessary performance degradation. In most contributions on robust MPC, these bounds are initially given together with the model, but, in practice, they have to be derived from data. We provide here a systematic procedure to do this, in an SM framework. Since we are interested in prediction bounds on the drone's position, let us consider the output equation (8b).

The n-steps-ahead predictor of the control-oriented model (8) for the ith output, denoted as $\hat{p}_{n,i}$, is

$$\begin{aligned}\hat{p}_{n,i}(k) = {} & C_{p,i}A^n\tilde{\boldsymbol{x}}(k) \\ & +C_{p,i}\sum_{j=1}^{n}A^{j-1}B\tilde{\boldsymbol{u}}(k+n-j)\end{aligned} \tag{10}$$

where $C_{p,i}$ is the ith row of the output matrix C_p in (8b) and $i = 1, 2, 3$. We define the measured regressor $\tilde{\boldsymbol{\varphi}}_n \in \mathbb{R}^{6+3n}$ as follows:

$$\tilde{\boldsymbol{\varphi}}_n(k)=\begin{bmatrix}\tilde{\boldsymbol{x}}(k)^T & \tilde{\boldsymbol{u}}(k)^T & \tilde{\boldsymbol{u}}(k+1)^T & \cdots & \tilde{\boldsymbol{u}}(k+n-1)^T\end{bmatrix}^T \tag{11}$$

and the vector of the identified parameters for the ith output $\hat{\boldsymbol{\theta}}_{n,i} \in \mathbb{R}^{6+3n}$ is

$$\hat{\boldsymbol{\theta}}_{n,i}=\begin{bmatrix}C_{p,i}A^n & C_{p,i}A^{n-1}B & C_{p,i}A^{n-2}B & \cdots C_{p,i}AB & C_{p,i}B\end{bmatrix}^T. \tag{12}$$

We can then reorganize the dataset $\tilde{\mathcal{M}}$ for each n-steps-ahead prediction $n = 1, \ldots, N$, where $N \in \mathbb{N}$ is the considered prediction horizon, by collecting sampled regressors and corresponding output values

$$\tilde{\mathcal{M}}_{n,i}\doteq\Big\{\big(\tilde{\boldsymbol{\varphi}}_n(k),\,\tilde{p}_{n,i}(k)\big)\quad \forall k\in\mathbb{N}_0^{N_s}\Big\} \tag{13}$$

where $\tilde{p}_{n,i}(k)$ is the ith component of the sampled n-steps-ahead position, available in the dataset.

Then, (10) can be compactly rewritten as follows:

$$\hat{p}_{n,i}(k)=\tilde{\boldsymbol{\varphi}}_n(k)^T\hat{\boldsymbol{\theta}}_{n,i}. \tag{14}$$

We can now estimate the error bound between the measured position and the corresponding n-steps-ahead prediction given by a linear model via the following LP:

$$\underline{\lambda}_{n,i}=\min_{\boldsymbol{\theta}_{n,i},\ \lambda\in\mathbb{R}^+}\lambda \tag{15a}$$

$$\begin{aligned}\text{s.t. } & |\tilde{p}_{n,i}-\tilde{\boldsymbol{\varphi}}_n^T\boldsymbol{\theta}_{n,i}| \\ & \le\lambda \quad \forall\big(\tilde{\boldsymbol{\varphi}}_n,\,\tilde{p}_{n,i}\big)\in\tilde{\mathcal{M}}_{n,i}.\end{aligned} \tag{15b}$$

This LP is always feasible and returns a positive value when the constraints generated by the data (15b) are informative enough to support from below the bound λ; otherwise, we have $\lambda = 0$. As proven in [33], due to the finiteness of the dataset $\tilde{\mathcal{M}}_{n,i}$, the computed value of $\underline{\lambda}_{n,i}$ is an under-approximation of the global (with respect to all possible regressors) error bound $\bar{\varepsilon}_{n,i}$. To estimate the latter, we, thus, include a scaling factor $\mu > 1$

$$\hat{\bar{\varepsilon}}_{n,i}=\mu\underline{\lambda}_{n,i},\qquad \mu>1. \tag{16}$$

We are now in position to define the feasible parameter set (FPS), i.e., the set of all the possible parameters of a predictor of the form (14) compatible with the available information. This set is the following convex polytope:

$$\Theta_{n,i}\doteq\big\{\boldsymbol{\theta}_{n,i}:|\tilde{p}_{n,i}-\tilde{\boldsymbol{\varphi}}_n^T\boldsymbol{\theta}_{n,i}|\le\hat{\bar{\varepsilon}}_{n,i}\ \forall\big(\tilde{\boldsymbol{\varphi}}_n,\,\tilde{p}_{n,i}\big)\in\tilde{\mathcal{M}}_{n,i}\big\}. \tag{17}$$

The FPS can be finally used to compute, for each prediction step n, the wanted worst case prediction error bound associated with the identified model (14) with parameters $\hat{\boldsymbol{\theta}}_{n,i}$

$$\tau_{n,i}=\max_{k=0,\ldots,N_s}\max_{\boldsymbol{\theta}\in\Theta_{n,i}}\big|\tilde{\boldsymbol{\varphi}}_n(k)^T\big(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}_{n,i}\big)\big|+\hat{\bar{\varepsilon}}_{n,i}. \tag{18}$$

Namely, this bound is the worst case discrepancy between the prediction provided by the identified linear model and that of any other model that is consistent with the data up to the error bound (16). Since also $\tau_{n,i}$ is an under-approximation of the actual bound, a second scaling factor $\eta > 1$ is introduced to account for the finite dataset

$$\hat{\tau}_{n,i}=\eta\left(\max_{k=0,\ldots,N_s}\max_{\boldsymbol{\theta}\in\Theta_{n,i}}\big|\tilde{\boldsymbol{\varphi}}_n(k)^T\big(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}_{n,i}\big)\big|\right)+\hat{\bar{\varepsilon}}_{n,i},\quad \eta>1. \tag{19}$$

The estimation procedure is carried out for each row of matrix C_p in (8b), eventually obtaining the 3-D worst case prediction error bound $\hat{\boldsymbol{\tau}}_n$.

Remark 3: Larger scaling factors μ and η result in larger uncertainty bounds to take into account possible new data that may invalidate the prior assumptions and/or estimated bounds. In a real application, it is easy to observe when these factors are too small by checking if the FPS (17) becomes empty when new data (i.e., additional inequalities) are considered. On the other hand, to understand if μ and η are too large, one can evaluate empirically the conservativeness with ad hoc tests or directly by analyzing the data of the system in operation and comparing it with the bounds, as we show in Section VI.

Algorithm 1 Convex Under-Approximation of the Free Space

Input: $\boldsymbol{p}(k), \mathcal{L}(k), r_L, d_{step}, \beta_a, \beta_e, n_v$
Result: $\mathcal{S}(k)$
1 $d_{min}(k) \leftarrow \min_{i=0,\dots,M-1} |\boldsymbol{s}_i(k)|$
2 **for** $i = 0 : n_v - 1$ **do**
3 $\quad \mathcal{S}_i^v \leftarrow \boldsymbol{p}(k) + d_{min}(k) \begin{bmatrix} \cos(i\ \beta_e)\cos(i\ \beta_a) \\ \cos(i\ \beta_e)\sin(i\ \beta_a) \\ \sin(i\ \beta_e) \end{bmatrix}$;
4 $\quad \gamma_i \leftarrow 0$
5 **end**
6 **for** $i = 0 : n_v - 1$ **do**
7 $\quad$ **while** $\gamma_i \neq 1$ **do**
8 $\quad\quad \mathcal{S}_i^{v,+} \leftarrow \mathcal{S}_i^v + d_{step} \begin{bmatrix} \cos(i\ \beta_e)\cos(i\ \beta_a) \\ \cos(i\ \beta_e)\sin(i\ \beta_a) \\ \sin(i\ \beta_e) \end{bmatrix}$;
9 $\quad\quad$ **if** $\mathcal{L}(k) \notin chull(\mathcal{S}^{v,+}) \wedge (\|\mathcal{S}_i^{v,+} - \boldsymbol{p}(k)\|_2 < r_L)$ **then**
10 $\quad\quad\quad \mathcal{S}^v \leftarrow \mathcal{S}^{v,+}$;
11 $\quad\quad$ **else**
12 $\quad\quad\quad \gamma_i \leftarrow 1$;
13 $\quad\quad$ **end**
14 $\quad$ **end**
15 **end**
16 $\mathcal{S}(k) \leftarrow chull\{\mathcal{S}^v\}$

D. Convex Under-Approximation of the Free Space

Let us denote with $\mathcal{L}(k) \doteq \{\boldsymbol{s}_0(k), \dots, \boldsymbol{s}_{M-1}(k)\} \in \mathbb{R}^3$ the M readings (4) of the exteroceptive sensor at time k. Let $d_{step} > 0$ be a user-defined distance. Moreover, we consider the user-selected quantities $\beta_a > \alpha_a$ and $\beta_e > \alpha_e$, i.e., azimuth and elevation angular resolution intervals defining a number n_v of equally spaced candidate vertices on the unit sphere centered at the drone position. Then, the proposed routine to build a convex under-approximation of the obstacle-free region is given by Algorithm 1.

First (lines 1–5), a regular polyhedron with n_v vertices lying on the ball $\mathcal{B}(\boldsymbol{p}(k), d_{\min}(k)) = \{\boldsymbol{w} \in \mathbb{R}^3 : \|\boldsymbol{w} - \boldsymbol{p}(k)\| = d_{\min}(k)\}$ is built. Then (lines 6–15), an arbitrarily chosen vertex is radially translated outward with respect to the center by the user-defined quantity d_{step}, and a convex hull of the new sequence of vertices is computed. If the obtained hull does not contain any points of $\mathcal{L}(k)$ and its vertices are closer than the maximum detection range r_L, the polyhedron $\mathcal{S}^v$ is updated (line 10), and the next vertex is considered for the expansion. Otherwise, the last expansion is removed and the position of that vertex is blocked (line 12). The cycle stops when all the auxiliary variables γ_i are true (line 7), which means that either all vertices are blocked or they have reached the maximum distance r_L from the drone.

When the process is completed for all the vertices, the algorithm returns the wanted polytope $\mathcal{S}(k)$. To take into account the size of the drone, its maximum encumbrance is removed from the sensor readings. As an example, Fig. 4 shows a few iterations of Algorithm 1.

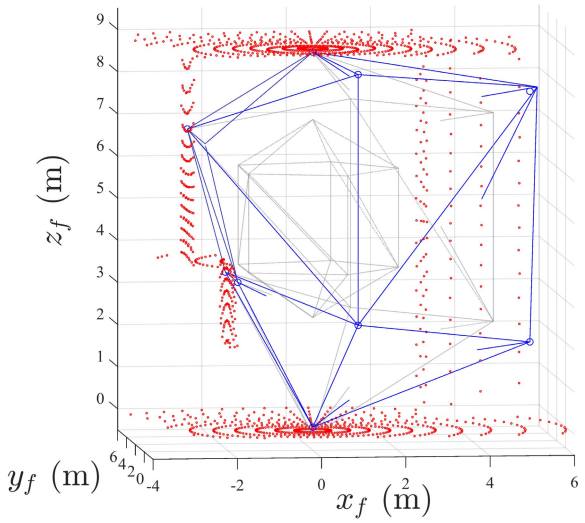


Fig. 4. Example of construction of the convex under-approximation of the free region: intermediate steps (dashed-dotted gray lines) and final result (blue lines). Exteroceptive sensor readings are shown as red points.

Remark 4: Differently from other approaches in the literature, Algorithm 1 ensures by construction that the current position of the drone, $\boldsymbol{p}(k)$, is always in the interior of $\mathcal{S}(k)$, i.e., in the safe set. Moreover, this approach returns a valid convex under-approximation of the free space also when stopped at an intermediate step, which is an advantage when a strict real-time implementation is needed. The parameters β_a and β_e can be tuned to trade-off the polytope accuracy with the required computational time, their limit being the resolution of the sensor.

IV. Multitrajectory MPC

When the environment is unknown and the vehicle has to rely only on real-time local information, a common approach to guarantee safety in a receding horizon framework is to consider a trajectory able to stop the vehicle inside the obstacle-free set $\mathcal{S}(k)$ within the prediction horizon $N \in \mathbb{N}$; see, e.g., [37]. This can be achieved imposing an admissible steady state, or, as in our case, a safe terminal set at the end of the predicted trajectory. Furthermore, a robust approach must be adopted to account for the model uncertainty. To this regard, robust and stochastic MPCs have been widely studied [38]. In these approaches, the optimal trajectory is conservatively computed to include all, or a statistically representative part of, possible uncertainty realizations. However, in this case, the use of such approaches can lead to a too conservative behavior. Moreover, the safe set $\mathcal{S}(k)$ considered at each time step k is a convex under-approximation of the free space that generally changes over time and can possibly evolve in a favorable way for the sake of pursuing the given target. Therefore, on the one hand, we want to robustly guarantee safety at each time step, ensuring that there exists a maneuver able to keep the vehicle in a safe area before a collision occurs. On the other hand, we want to limit the conservativeness of the approach considering a possible, hopefully favorable, evolution of the safe set at the next time step. The proposed mt-MPC technique

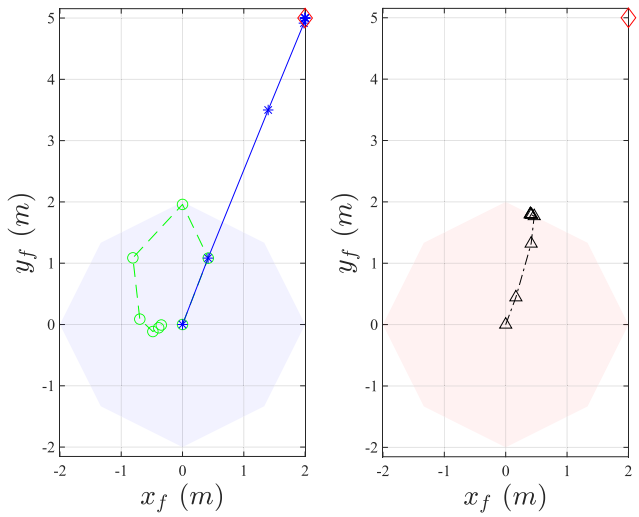


Fig. 5. Example illustrating the predicted state evolution with the multi-trajectory concept (left) versus a single-trajectory one (right). Blue line with "*": exploiting trajectory. Green dashed line with "○": safe trajectory. Black dashed-dotted line with "△": single-trajectory approach. Red "◇": target. Colored polytopes represent the safe sets $\mathcal{S}(k)$.

aims at managing these conflicting objectives, by planning two trajectories: a "safe" one, guaranteed to be contained in the safe set and to reach a safe state also considering the model uncertainty, and an "exploiting" one, which assumes that the current constraints are too conservative and can, thus, violate them. The two trajectories feature the same input at the current step and separate only afterward in the prediction. To better illustrate this concept, let us consider the 2-D example in Fig. 5, showing on the left the multitrajectory approach and on the right a standard, single-trajectory one. The trajectories shown are computed by solving an FHOCP using the control-oriented model (8) and aiming to reach a target beyond the safe set. The single-trajectory approach forces all the predicted trajectory to lie within the safe set, minimizing the average distance from the target. This solution is optimal for the current safe set, but it does not consider a possible favorable evolution of the latter at the next time step. On the other hand, the multitrajectory approach plans a much better (yet currently unfeasible) exploiting trajectory, but still retaining a safe alternative in case the constraints' set does not expand toward the target. Since the approach is implemented in receding horizon, the potential advantage is apparent by comparing the position reached at the first predicted time step by the two approaches; see Fig. 5.

Let us denote with $\boldsymbol{x}^I_{i|k}$ the state of the exploiting trajectory of system (8) at time $k+i$ predicted at time k, and with $\boldsymbol{x}^{II}_{i|k}$ that of the safe trajectory. Considering a finite horizon $N \in \mathbb{N}$, such that $N \geq \underline{j}(r_L)$ (see Assumption 1), we introduce the two input sequences

$$\boldsymbol{U}^I = \begin{bmatrix} \boldsymbol{u}^{I^T}_{0|k} & \boldsymbol{u}^{I^T}_{1|k} & \cdots & \boldsymbol{u}^{I^T}_{N-1|k} \end{bmatrix}^T \tag{20}$$

$$\boldsymbol{U}^{II} = \begin{bmatrix} \boldsymbol{u}^{II^T}_{0|k} & \boldsymbol{u}^{II^T}_{1|k} & \cdots & \boldsymbol{u}^{II^T}_{N-1|k} \end{bmatrix}^T \tag{21}$$

pertaining to the exploiting and safe trajectories, respectively. We consider the following cost function to track a target $\boldsymbol{p}_t(k)$:

$$J\big(\boldsymbol{x}(k), \boldsymbol{U}, \boldsymbol{p}_t(k)\big) = \sum_{i=1}^{N} \|\boldsymbol{p}^I_{i|k} - \boldsymbol{p}_t(k)\|^2_Q \tag{22}$$

where $Q \in \mathbb{R}^{3\times 3}$ is a symmetric positive-definite weighting matrix, and vector $\boldsymbol{U} = [\boldsymbol{U}^{I^T}\ \boldsymbol{U}^{II^T}]^T \in \mathbb{R}^{3(2N)}$.

To include in the FHOCP the worst case prediction error bound $\hat{\boldsymbol{\tau}}_n$ computed in Section III-C, we tighten along the horizon N the set $\mathcal{S}(k)$ computed with Algorithm 1. To this end, let us define the uncertainty hyper-rectangles at each prediction step n as follows:

$$\mathcal{T}_n = \left\{\boldsymbol{p} \in \mathbb{R}^3 : |\boldsymbol{p}| \leq \hat{\boldsymbol{\tau}}_n\right\} \quad \forall n \in \mathbb{N}_1^{N-1} \tag{23}$$

$$\mathcal{T}_N = \left\{\boldsymbol{p} \in \mathbb{R}^3 : |\boldsymbol{p}| \leq \max_{n=1,\dots,N} \hat{\boldsymbol{\tau}}_n\right\} \tag{24}$$

where all inequalities are elementwise.

Then, the polytope $\mathcal{S}(k)$ is tightened along the horizon as follows:

$$\mathcal{S}_i(k) = \mathcal{S}(k) \ominus \mathcal{T}_i \quad \forall i \in \mathbb{N}_1^N \tag{25}$$

leading to the sequence of convex polytopes

$$\bar{\mathcal{S}}(k) = \left\{\mathcal{S}_i(k) \quad \forall i \in \mathbb{N}_1^N\right\}. \tag{26}$$

We are now in position to formulate the multitrajectory FHOCP, denoted as $\mathscr{P}(\boldsymbol{x}(k),\ \bar{\mathcal{S}}(k),\ \boldsymbol{p}_t(k))$

$$\min_{\boldsymbol{U}} \quad J\big(\boldsymbol{x}(k), \boldsymbol{U}, \boldsymbol{p}_t(k)\big) \tag{27a}$$

$$\text{s.t.} \quad \boldsymbol{u}^I_{0|k} = \boldsymbol{u}^{II}_{0|k} \tag{27b}$$

$$\boldsymbol{x}^{I,II}_{0|k} = \boldsymbol{x}(k) \tag{27c}$$

$$\boldsymbol{x}^{I,II}_{i+1|k} = A\boldsymbol{x}^{I,II}_{i|k} + B\boldsymbol{u}^{I,II}_{i|k} \quad \forall i \in \mathbb{N}_0^{N-1} \tag{27d}$$

$$\boldsymbol{p}^{I,II}_{i|k} = C_p\boldsymbol{x}^{I,II}_{i|k} \quad \forall i \in \mathbb{N}_0^N \tag{27e}$$

$$\boldsymbol{v}^{I,II}_{i|k} = C_v\boldsymbol{x}^{I,II}_{i|k} \quad \forall i \in \mathbb{N}_0^N \tag{27f}$$

$$-\bar{\boldsymbol{v}} \leq \boldsymbol{v}^{I,II}_{i|k} \leq \bar{\boldsymbol{v}} \quad \forall i \in \mathbb{N}_0^N \tag{27g}$$

$$-\bar{\boldsymbol{a}} \leq \boldsymbol{a}^{I,II}_{i|k} \leq \bar{\boldsymbol{a}} \quad \forall i \in \mathbb{N}_0^{N-1} \tag{27h}$$

$$\boldsymbol{p}^{II}_{i|k} \in \mathcal{S}_i(k) \quad \forall i \in \mathbb{N}_1^N \tag{27i}$$

$$\boldsymbol{u}^{II}_{i|k} = \boldsymbol{u}^{II}_{i-1|k} \quad \forall i \in \mathbb{N}_{N-1-\underline{j}(r_L)}^{N-1} \tag{27j}$$

$$\boldsymbol{p}^{II}_{N|k} \in \mathcal{S}_N \ominus \mathcal{P}_D \tag{27k}$$

where all equalities and inequalities are elementwise, the predicted acceleration $\boldsymbol{a}_{i|k}$ pertaining to the exploiting and safe trajectories is defined as $\boldsymbol{a}^{I,II}_{i|k} = K_{\text{vel}}(K_{\text{pos}}(\boldsymbol{u}^{I,II}_{i|k} - \boldsymbol{p}^{I,II}_{i|k}) - \boldsymbol{v}^{I,II}_{i|k})$, and $\bar{\boldsymbol{a}}$ and $\bar{\boldsymbol{v}}$ are the maximum acceleration and velocity vectors, respectively. Constraints (27c)–(27h) are meant to be applied to both safe and exploit trajectories. The FHOCP (27) is a convex QP that, if feasible, can be solved efficiently for a global minimizer. We denote its solution as $\boldsymbol{U}^*(\boldsymbol{x}(k), \bar{\mathcal{S}}(k), \boldsymbol{p}_t(k)) = [\boldsymbol{U}^{I^{*T}}\ \boldsymbol{U}^{II^{*T}}]^T$ and the corresponding optimal predicted state trajectories, with all elements stacked in single column vectors, as $\boldsymbol{X}^{I*}(\boldsymbol{x}(k), \bar{\mathcal{S}}(k), \boldsymbol{p}_t(k)) \in \mathbb{R}^{6N}$ and $\boldsymbol{X}^{II*}(\boldsymbol{x}(k), \bar{\mathcal{S}}(k), \boldsymbol{p}_t(k)) \in \mathbb{R}^{6N}$. The optimal control problem results to be divided in two predictions: 1) the trajectory $\boldsymbol{X}^I$, considered in the cost function, pointing to the desired reference and 2) the trajectory $\boldsymbol{X}^{II}$, which, instead, satisfies

Algorithm 2 Multitrajectory MPC

Input: $\boldsymbol{x}(k), \mathcal{S}(k), \boldsymbol{p}_t(k)$
Result: $\boldsymbol{u}(k)$
if $\mathscr{P}(\boldsymbol{x}(k), \bar{\mathcal{S}}(k), \boldsymbol{p}_t(k))$ *is feasible* **then**
  $l(k+1) = k$
  $m(k+1) = 1$
  $\boldsymbol{u}(k) \leftarrow$ first control input in $\boldsymbol{U}^*(\boldsymbol{x}(k), \bar{\mathcal{S}}(k), \boldsymbol{p}_t(k))$
else
  if $\mathscr{P}(\boldsymbol{x}(k), \bar{\mathcal{S}}(l(k)), \boldsymbol{p}_t(k))$ *is feasible* **then**
    $l(k+1) = l(k)$
    $m(k+1) = 1$
    $\boldsymbol{u}(k) \leftarrow$ first control input in $\boldsymbol{U}^*(\boldsymbol{x}(k), \bar{\mathcal{S}}(l(k)), \boldsymbol{p}_t(k))$
  else
    $l(k+1) = l(k)$
    if $m(k) < N-1$ **then**
      $m(k+1) = m(k)+1$
    else
      $m(k+1) = m(k)$
    end
    $\boldsymbol{u}(k) = \boldsymbol{u}^{II}_{m(k)|l(k)}$
  end
end

tightened state and terminal constraints (27i), (27j), and (27k) ensuring the existence of an obstacle-free trajectory.

At any time k, let us denote with $l(k) < k$ the latest sampling instant, such that the FHOCP $\mathscr{P}(\boldsymbol{x}(l(k)), \bar{\mathcal{S}}(l(k)), \boldsymbol{p}_t(l(k)))$ was feasible and with $m(k) \in [1, N-1]$ a counter used inside our algorithm. Then, we propose the following receding horizon strategy.

Since, for a time-invariant environment, the safe set generated by Algorithm 1 depends only on $\boldsymbol{x}(k)$, the mt-MPC approach results in a dynamic, state-feedback control law with internal states $l(k)$ and $m(k)$ and input $\boldsymbol{p}_t(k)$

$$\begin{aligned} m(k+1) &= \zeta(\boldsymbol{x}(k), l(k), m(k)) \\ l(k+1) &= \xi(\boldsymbol{x}(k), l(k)) \\ \boldsymbol{u}(k) &= \kappa\big(\boldsymbol{x}(k), l(k), m(k), \boldsymbol{p}_t(k)\big) \end{aligned} \tag{28}$$

where functions $\zeta : \mathbb{R}^6 \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, $\xi : \mathbb{R}^6 \times \mathbb{N} \to \mathbb{N}$, and $\kappa : \mathbb{R}^6 \times \mathbb{N} \times \mathbb{N} \times \mathbb{R}^3 \to \mathbb{R}^3$ are implicitly defined by Algorithm 2. The resulting closed-loop system is

$$\begin{aligned} m(k+1) &= \zeta(\boldsymbol{x}(k), l(k), m(k)) \\ l(k+1) &= \xi(\boldsymbol{x}(k), l(k)) \\ \boldsymbol{x}(k+1) &= \boldsymbol{x}\big(k; \kappa\big(\boldsymbol{x}(k), l(k), m(k), \boldsymbol{p}_t(k)\big)\big). \end{aligned} \tag{29}$$

In Algorithm 2, the role of variable $l(k)$ and of the corresponding set $\bar{\mathcal{S}}(l(k))$ is to keep track of the last polytopic approximation of the free space that yielded a feasible problem. This is introduced to cope with the time-varying nature of the safe convex set $\mathcal{S}(k)$. The role, instead, of variable $m(k)$ is to select, if needed, suitable control inputs in the safe trajectory to guarantee that an input leading to an obstacle-free trajectory can be issued notwithstanding the measurement noise and prediction uncertainty. Such a guarantee holds, however, only when unknown but static obstacles are considered, as in our problem. In the presence of time-varying obstacles, additional assumptions and different approaches would be required, currently subject of our research.

The mt-MPC approach guarantees an obstacle-free trajectory, as shown by the following result.

Lemma 1: Assume that the FHOCP (27) at time $k = 0$ is feasible and that $\boldsymbol{p}(0) \notin \mathcal{O}$, i.e., the drone is initially in the obstacle-free region. Moreover, assume that for all $\boldsymbol{x}(k)$ and all sequences $[\boldsymbol{u}^{II^T}_{0|k}, \boldsymbol{u}^{II^T}_{1|k}, \ldots, \boldsymbol{u}^{II^T}_{N-1|k}]^T$, we have that the estimated uncertainty bounds are not violated, i.e., $\boldsymbol{p}^{II}(k+n) \in \boldsymbol{p}^{II}_{n|k} \oplus \hat{\boldsymbol{\tau}}_n$, $\forall n \le N$. Then, the trajectory of the close loop system (29) is such that $\boldsymbol{p}(k) \notin \mathcal{O}$, $\forall k > 0$.

Proof: At $k = 0$, problem $\mathscr{P}(\boldsymbol{x}(0), \bar{\mathcal{S}}(0), \boldsymbol{p}_t(0))$ is solved, $m(1)$ is set to 1, and $l(1)$ is set to 0. For any $k \ge 0$, let us denote with $\boldsymbol{U}^{II*} = [\boldsymbol{u}^{II*^T}_{0|k}, \boldsymbol{u}^{II*^T}_{1|k}, \ldots, \boldsymbol{u}^{II*^T}_{N-1|k}]^T$ the optimal safe input sequence computed by the mt-MPC algorithm, be it by solving $\mathscr{P}(\boldsymbol{x}(k), \bar{\mathcal{S}}(k), \boldsymbol{p}_t(k))$ or $\mathscr{P}(\boldsymbol{x}(k), \bar{\mathcal{S}}(l(k)), \boldsymbol{p}_t(k))$ (see Algorithm 2), with $\boldsymbol{x}^{II*}_{j|k}$ the jth element of the safe trajectory and $\boldsymbol{p}^{II*}_{j|k} = C_p \boldsymbol{x}^{II*}_{j|k}$ the corresponding position.

Then, at each $k > 0$, there are three possibilities.

1) If $\mathscr{P}(\boldsymbol{x}(k), \bar{\mathcal{S}}(k), \boldsymbol{p}_t(k))$ is feasible, then at time $k+1$, we have $\boldsymbol{p}(k+1) \in \boldsymbol{p}^{II*}_{1|k} \oplus \hat{\boldsymbol{\tau}}_1 \in \mathcal{S}(k)$; see (27d), (27i), and (25).
2) Conversely, if $\mathscr{P}(\boldsymbol{x}(k), \bar{\mathcal{S}}(k), \boldsymbol{p}_t(k))$ is not feasible, but problem $\mathscr{P}(\boldsymbol{x}(k), \bar{\mathcal{S}}(l(k)), \boldsymbol{p}_t(k))$ is feasible, the latter is solved. Thus, in this case, we have $\boldsymbol{p}(k+1) \in \boldsymbol{p}^{II*}_{1|k} \oplus \hat{\boldsymbol{\tau}}_1 \in \mathcal{S}(l(k))$.
3) Finally, if both $\mathscr{P}(\boldsymbol{x}(k), \bar{\mathcal{S}}(k), \boldsymbol{p}_t(k))$ and $\mathscr{P}(\boldsymbol{x}(k), \bar{\mathcal{S}}(l(k)), \boldsymbol{p}_t(k))$ are not feasible, we apply the $m(k)$th element in the tail of the safe optimal input sequence obtained with the last feasible problem $\mathscr{P}(\boldsymbol{x}(l(k)), \bar{\mathcal{S}}(l(k)), \boldsymbol{p}_t(l(k)))$, i.e., $\boldsymbol{u}^{II*}_{m(k)|l(k)}$.

Since $\mathscr{P}(\boldsymbol{x}(l(k)), \bar{\mathcal{S}}(l(k)), \boldsymbol{p}_t(l(k)))$ satisfies constraint (27i), we have $\boldsymbol{p}(k+1) \in \boldsymbol{p}^{II*}_{m(k)+1|l(k)} \oplus \hat{\boldsymbol{\tau}}_{m(k)+1} \in \mathcal{S}(l(k))$, $\forall m(k) \le N-1$. Therefore, we have that, in all cases 1)–3), $\boldsymbol{p}(k+1)$ belongs to a set $\mathcal{S}(j)$, with $j \le k$. Now, by construction (see Algorithm (1)), whenever $\boldsymbol{p}(j) \notin \mathcal{O}$, then the corresponding set $\mathcal{S}(j)$ is an under-approximation of the obstacle-free region, i.e., $\mathcal{S}(j) \cap O_i = \emptyset$, $\forall i = 1, \ldots, N_o$, meaning that in all cases 1)–3), we have $\boldsymbol{p}(k+1) \notin \mathcal{O}$. We, thus, demonstrated that $\boldsymbol{p}(k) \notin \mathcal{O} \Rightarrow \boldsymbol{p}(k+1) \notin \mathcal{O}$. The result is then proven by induction, considering that $\boldsymbol{p}(0) \notin \mathcal{O}$ by assumption. Finally, if case **C)** is encountered repeatedly, the last feasible safe set point may be applied until the end of the horizon. In this case, the drone state ends up in a positively invariant set because of constraint (27k). Then, $\boldsymbol{p}(k) \in \mathcal{S}_N(l(k)) \ominus \mathcal{P}_D$, leading to a new feasible problem $\mathscr{P}(\boldsymbol{x}(k), \bar{\mathcal{S}}(l(k)), \boldsymbol{p}_t(k))$ [case 2)], where constraints (27c)–(27k) are satisfied by the trivial safe trajectory $\boldsymbol{U}^{II} = [\boldsymbol{u}^{II*^T}_{N-1|l(k)}, \ldots, \boldsymbol{u}^{II*^T}_{N-1|l(k)}]^T$ by Assumption 2. □

Remark 5: The FHOCP (27), from an implementation point of view, can be simplified by imposing a terminal equality constraint instead of (27j), imposing that the last step of the safe trajectory is a steady state and, then, artificially extending the horizon by a time interval larger than $\underline{j}(r_L)$. In this case,

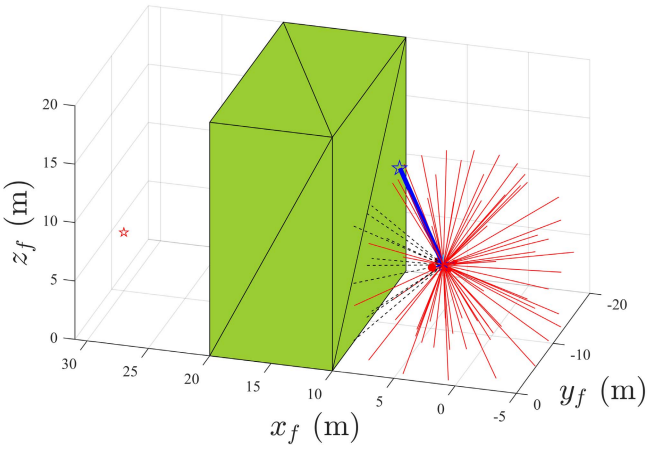


Fig. 6. Example of situation where the drone might stop in front of an obstacle, and the proposed temporary target shifting is adopted. Blue star: $\hat{\boldsymbol{p}}_t(k)$. Red star: $\boldsymbol{p}_t(k)$. Black dashed lines: sensor measurements with index $i \notin \mathcal{I}(k)$. Red solid lines: sensor measurements with index $i \in \mathcal{I}(k)$.

to take into account the model mismatch, (24) should be modified to include the horizon $i = 1, \dots, N + \underline{j}(r_L)$.

A. Temporary Target Shifting Strategy

A common problem in optimization-based autonomous navigation without mapping is the possibility that the system is in front of an obstacle that is between the current drone position and the target, as shown in Fig. 6. In such a situation, if no countermeasure is taken, the drone might simply stop at a locally optimal position, because any allowed lateral movement would temporarily imply a growth of distance from the target. To avoid this situation, we propose a strategy named temporary target shifting. Let $\boldsymbol{f}_t = \boldsymbol{p}_t(k) - \boldsymbol{p}(k)$ be the vector connecting the drone to the target. When the sensor readings whose angular position is closest to that of direction $\boldsymbol{f}_t$ do not detect any obstacle (up to the maximum range r_L), the original target $\boldsymbol{p}_t(k)$ is used. Otherwise, a temporary target $\hat{\boldsymbol{p}}_t(k)$ is chosen as follows. Consider the set of indexes corresponding to the sensor readings reporting the maximum distance

$$\mathcal{I}(k) = \left\{ \bar{j} : \bar{j} = \arg \max_{i=0,\dots,M-1} \boldsymbol{\rho}_i(k) \right\}. \tag{30}$$

Then, the temporary target is obtained as the sensor reading, among those with index $\bar{j} \in \mathcal{I}$, that is closest to the target

$$\hat{\boldsymbol{p}}_t(k) = \min_{\bar{j} \in \mathcal{I}} \| \boldsymbol{s}_{\bar{j}}(k) - \boldsymbol{f}_t \|. \tag{31}$$

This strategy is illustrated in Fig. 6 as well. The temporary target $\hat{\boldsymbol{p}}_t(k)$ is then held constant until the drone reaches it within a certain tolerance, or until the target direction becomes obstacle-free again, whichever condition happens first. This choice avoids situations where the drone starts moving back and forth behind an obstacle, because the temporary target is periodically switched between the two visible edges. If more than one solution to (31) exists, we take the one with smallest index $\bar{j}$. This approach yields good results in most cases with bounded obstacles, but still does not guarantee that the target is eventually reached if the obstacles' shapes are too complicated. In those cases, a mapping strategy shall be added, to incrementally explore the environment and save information on it until finding the path to the target. Mapping is outside the scope of this work, but can be well combined with our approach.

TABLE I

PARAMETERS EMPLOYED IN THE NUMERICAL SIMULATIONS WHERE m_d IS THE MASS OF THE VEHICLE, l_d IS THE HALF OF THE DISTANCE BETWEEN TWO OPPOSITE MOTORS, I_{xx}, I_{yy}, AND I_{zz} ARE THE DIAGONAL COMPONENTS OF THE INERTIA MATRIX I, AND c_d AND c_t ARE THE DRAG AND THRUST COEFFICIENTS, RESPECTIVELY

Multi-copter drone			
m_d	6 kg	l_d	0.525 m
$I_{xx,yy}$	0.1663 kg m^2	I_{zz}	0.27 kg m^2
c_d	10^{-5} Nm s^2/rad^2	c_t	9.99 10^{-5} N s^2/rad^2
Simulated exteroceptive sensor			
r_L	10 m	α_a, α_e	30°
Control-oriented model (6)			
T_s	0.3 s	K_{pos}	0.6 $\mathbb{I}^3$
$K_{vel} =$	1.597366	−0.460821	1.464074 10^{-2}
	0.526193	1.581678	8.223714 10^{-2}
	−4.485497 10^{-2}	−1.060711 10^{-2}	2.318620
$\bar{\boldsymbol{v}}$	$[2\ 2\ 2]^T$ m/s	$\bar{\boldsymbol{a}}$	$[5\ 5\ 5]^T$ m/s^2
Algorithm 1			
β_a, β_e	60°	d_{step}	0.2 m
FHOCP (27)			
Q	2 $\mathbb{I}^3$	N	10

V. SIMULATION RESULTS

The effectiveness of the proposed approach has been evaluated first via numerical simulations according to the layout reported in Fig. 1. In this case, the nonlinear drone dynamics with the low-level position controller have been simulated with the model described in [39]. The employed system and control parameters are reported in Table I. The QP (27) is solved using MATLAB[1] `quadprog` running on a Quad-Core Intel Core i7 (3.6 GHz, 32 GB) on MATLAB 2020b under MS Windows. Fig. 7 shows the UAV during navigation in a typical simulation test where the convex under-approximation $\mathcal{S}(k)$, the "safe" trajectory, and the "exploiting" one can be easily distinguished. The approach exhibits good performance, driving quickly the drone to its target without collisions. On average, in our tests, the numerical solution of the QP (27) required about 45 ms per sampling step and about 0.1 s in addition to run Algorithm 1. As mentioned in Remark 4, the latter can also be safely interrupted if a strict real-time implementation is needed, at the cost of smaller safe set.

We compared the mt-MPC approach with a standard MPC, which still employs Algorithms 1 and 2, however, with a single trajectory in the FHOCP, subject to all the operational and safety constraints. We used the same tuning parameters in the cost function for the two approaches (see Table I). In the single-trajectory MPC, the following FHOCP is considered:

$$\min_{\boldsymbol{U}} \quad \sum_{i=1}^{N} ||\boldsymbol{p}_{i|k} - \boldsymbol{p}_t(k)||_Q^2$$

[1]Registered trademark.

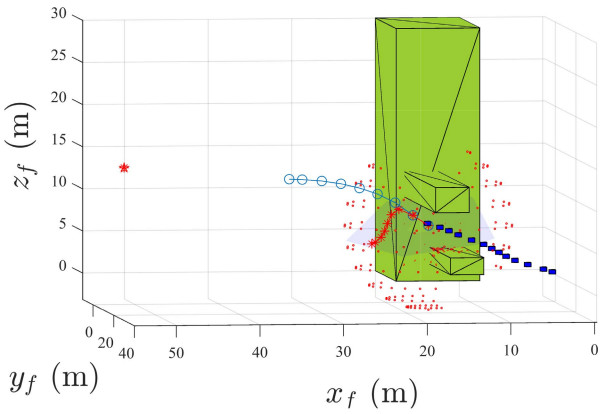


Fig. 7. Simulation of the mt-MPC approach. Light blue polytope: set $\mathcal{S}(k)$. Red points: exteroceptive sensor measurements. Red dashed line with "$*$": safe trajectory. Blue solid line with "$\circ$": exploiting trajectory. Red star: target $\boldsymbol{p}_t$. Blue rectangles: past drone positions. Obstacles beyond the LiDAR's field of view are not shown.

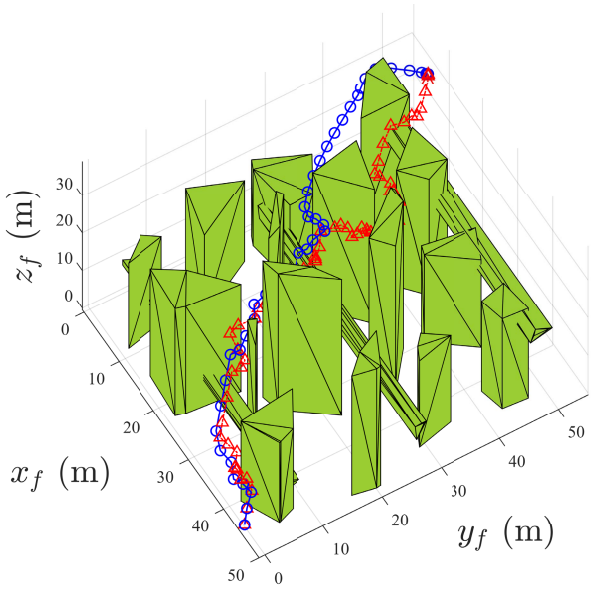


Fig. 8. Simulation results. Trajectories obtained with mt-MPC (blue solid line with "$\circ$") and MPC (dashed-dotted red line with "$\triangle$").

$$
\begin{aligned}
\text{s.t.}\quad & \boldsymbol{x}_{0|k} = \boldsymbol{x}(k) \\
& \boldsymbol{x}_{i+1|k} = A\boldsymbol{x}_{i|k} + B\boldsymbol{u}_{i|k} \quad \forall i \in \mathbb{N}_0^{N-1} \\
& \boldsymbol{p}_{i|k} = C_p \boldsymbol{x}_{i|k} \quad \forall i \in \mathbb{N}_0^{N} \\
& \boldsymbol{v}_{i|k} = C_v \boldsymbol{x}_{i|k} \quad \forall i \in \mathbb{N}_0^{N} \\
& -\bar{\boldsymbol{v}} \leq \boldsymbol{v}_{i|k} \leq \bar{\boldsymbol{v}} \quad \forall i \in \mathbb{N}_0^{N} \\
& \boldsymbol{p}_{i|k} \in \mathcal{S}_i(k) \quad \forall i \in \mathbb{N}_0^{N} \\
& -\bar{\boldsymbol{a}} \leq \boldsymbol{a}_{i|k} \leq \bar{\boldsymbol{a}} \quad \forall i \in \mathbb{N}_0^{N-1} \\
& \boldsymbol{u}_{i|k} = \boldsymbol{u}_{i-1|k} \quad \forall i \in \mathbb{N}_{N-1-\underline{j}(r_L)}^{N-1} \\
& \boldsymbol{x}_{N|k} \in \mathcal{S}_N \ominus \mathcal{P}_D.
\end{aligned}
$$

To ensure persistent obstacle avoidance, we applied again Algorithm 2. Fig. 8 shows a comparison between the resulting closed-loop trajectories. Both drive the UAV to its target without collisions, however, obtaining different paths. For a more thorough comparison, we ran a series of $N_{\text{sim}} = 1000$ problems with randomly generated initial state, obstacles, and target, but all presenting a layout qualitatively similar to that of Fig. 8, i.e., where the drone has to traverse an area with many unknown obstacles. In each test i, we measured the cumulative closed-loop tracking error $J_i = \sum_{k=0}^{T_i} ||\boldsymbol{p}(k) - \boldsymbol{p}_{t_i}||^2$.
Furthermore, we considered the following average quantity as performance indicator:

$$\bar{J} = \frac{1}{N_{\text{sim}}} \sum_{i=1}^{N_{\text{sim}}} J_i.$$

Finally, we also compared the computational effort required to solve the FHOCP in the two cases. The obtained results indicate that the mt-MPC improves the average tracking error (-7.2%) with respect to the standard MPC technique, however, with a higher computational time [$+32\%$ to solve the QP (27)]. The presented results, together with the simulations in a 2-D scenario presented in [31], thus, confirm that the approach has good potential in terms of performance improvement, at the cost of higher computational effort in this application.

VI. Experimental Results

To demonstrate the performance of the presented mt-MPC, we implemented it on an autonomous multicopter drone and carried out experiments out-of-the-laboratory in our test site.

A. Test Site and Prototype Drone

We ran the experiments in an outdoor facility of Politecnico di Milano at Spino d'Adda ($45.4° N$, $9.5° E$), near Milan, Northern Italy. A view of the site test is shown in Fig. 9(a). We conducted experiments with a DJI $S1000+$ octocopter, shown in Fig. 9(b). The frame has a diagonal wheelbase of 1045 mm with eight motors that rotate at 400 rpm/V.

The drone is equipped with two planar LiDAR SICK $TiM5xx$ series sensors, each one with 270° range fused together to obtain a 360° field of view. Each sensor has a scanning frequency of 15 Hz, a range of $r_L = 10$ m, and an angular resolution of $\alpha_s = 0.33°$. Due to the planar nature of the sensors used, we fixed the reference vertical position to a constant value in the tests. The low-level position controller is provided by a DJI $A3$ flight control unit equipped with a built-in inertial measurement unit (IMU) featuring a standard GPS compass. We implemented the high-level controller on an Odroid-XU4 embedded system, featuring an octa-core Exynos 5422 big.LITTLE processor running Linux Ubuntu 18 and robot operating system (ROS) Melodic. The flight controller has an ROS interface via the DJI onboard SDK, allowing the Odroid to send reference commands and receiving sensors feedback via ROS to/from the $A3$ unit.

B. ROS Implementation

An overview of nodes and topics communication through an ROS graph diagram is shown in Fig. 10. The two LiDARs provide to the `/s1000_interface` node their measurements at a frequency of 15 Hz. The node elaborates the measurements and publishes a message of type

(a) (b)

Fig. 9. (a) View of the test site at Spino d'Adda, Lombardy, Italy. (b) DJI S1000+ octocopter during field test.

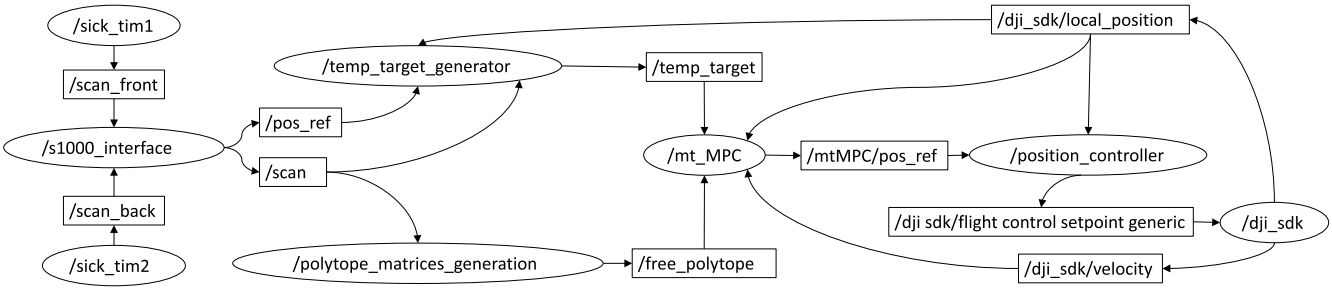


Fig. 10. ROS simplified graph diagram showing various nodes (ellipses) and topics (rectangles) involved in the approach.

`LaserScan` with the merged measurements, together with a `PoseStamped` message defining the desired final position target. The `/temp_target_generator` node is in charge of providing the temporary target to the `/mt_MPC` node following the procedure described in Section IV-A at a frequency of 5 Hz. The node exploits the LiDAR measurements, the final target, and the GPS position of the drone published by the `/dji_sdk` node at a frequency of 50 Hz. The LiDAR measurements are also exploited by the `/polytope_matrices_generation` node, where Algorithm 1 is implemented, that publishes the polytope $\mathcal{S}(k)$ with a custom message `PolytopeMatricesStamped` at a frequency of 5 Hz. Finally, the `/mt_MPC` node receives the target, the free polytope, and the state feedback, and it executes Algorithm 2 and publishes a position reference at a frequency of 3 Hz. The latter is then sent to a `/position_controller` node that publishes a velocity reference to the `/dji_sdk` node. The ROS implementation of the approach is available at `https://github.com/DaniloSaccani/mt_MPC`.

C. Model Identification and Error Bound Estimation

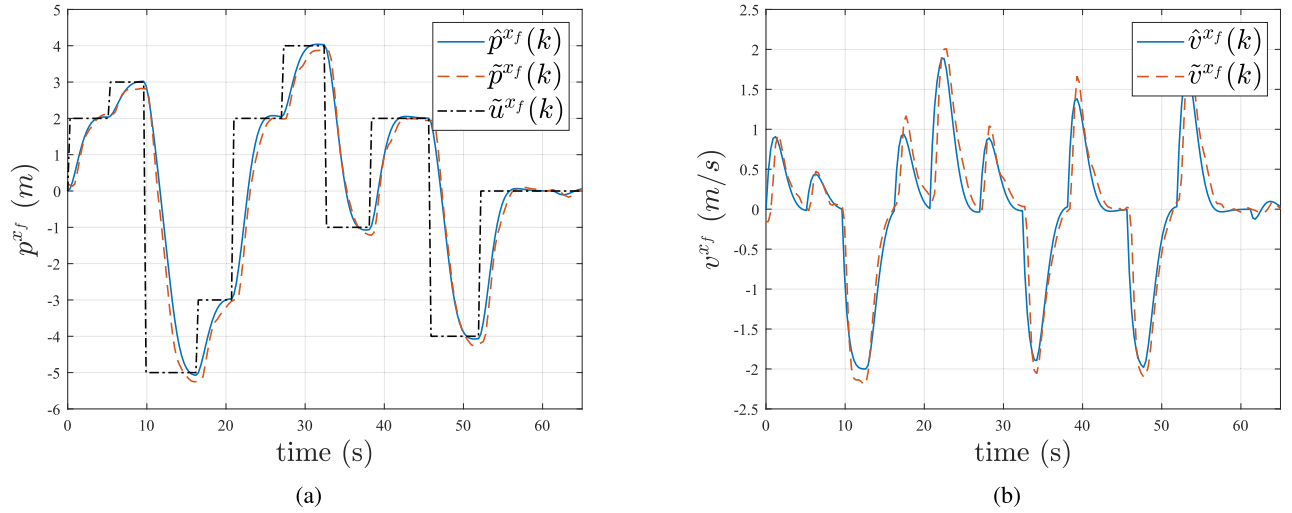
To test the presented approach, we have collected position step responses in closed loop, recording the position and velocity of the vehicle with a sampling frequency of 100 Hz, while the high-level control unit was sending predefined position references to the low-level DJI $A3$ flight controller. We identified the control-oriented model (8) as described in Section III-B. Fig. 11 shows a comparison between the measured $\tilde{p}^{x_f}$ and $\tilde{v}^{x_f}$ and the estimated model. Then, we exploited the dataset to estimate the worst case prediction error bound $\hat{\boldsymbol{\tau}}_n$ considering the scaling factors $\mu = \eta = 1.02$. Fig. 12 shows the obtained values of $\underline{\lambda}_{n,i}$ (15) and the worst case prediction error bound $\hat{\tau}_{n,i}$ (19) along x_f and y_f for an horizon $N = 30$ in the validation dataset. As it can be noticed, the worst case simulation error bound $\hat{\tau}_{n,i}$ has a maximum value of about 3 m, which is reasonable considering the GPS noise, the presence of little wind, and the model-plant mismatch. The constant $\underline{j}(r_L)$, that represents the settling time of the low-level position controller to reach a target placed at a distance r_L, can be roughly estimated from the ratio $(r_L/\bar{v})$ and in our case is $\underline{j}(r_L) = 16$. We estimated the set $\mathcal{P}_W$ from the data by taking the convex hull of the steady-state samples of the tracking error, as shown in Fig. 13

$$\mathcal{P}_W = \text{chull}\Big(\tilde{\boldsymbol{x}}(k+j;\tilde{\boldsymbol{u}}) - \Big[\tilde{\boldsymbol{u}}\mathbf{0}^{3\times 1}\Big]\Big).$$

As pointed out in Section II-A, in practice, Assumption 1 is satisfied if the low level controller is properly tuned, and it is able to stabilize the drone's trajectories and to track the given reference.

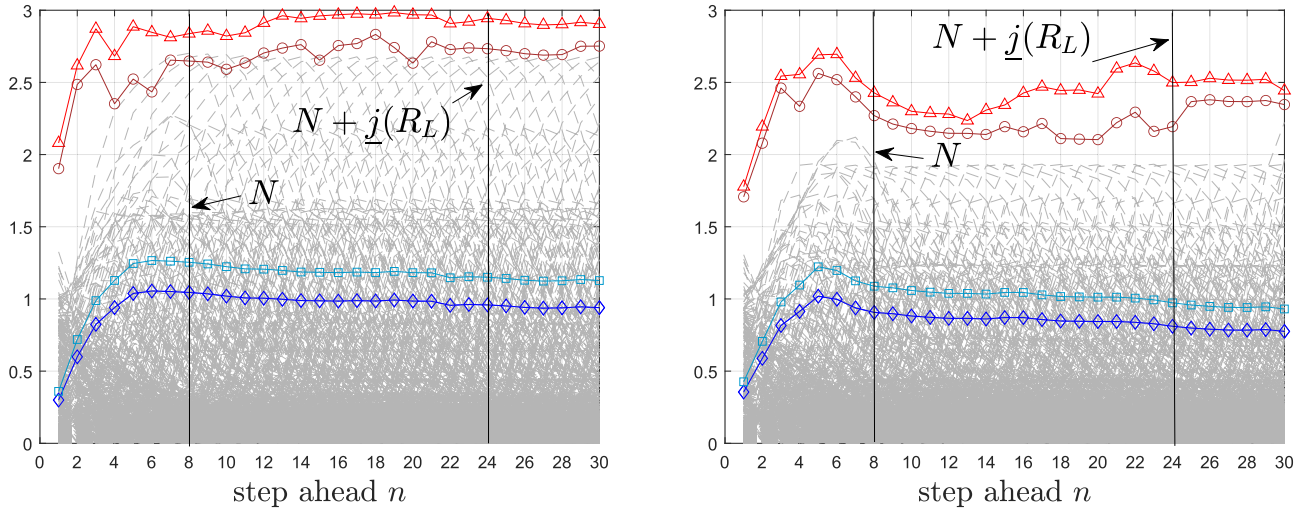
D. MPC Implementation

We translated the developed algorithms in Python and integrated them within ROS [40]. To obtain a strictly convex optimization problem, which improves the solution speed and



(a) (b)

Fig. 11. Comparison of the identified "control-oriented" model (blue solid line) and the validation dataset (orange dashed line). Position in (a) and velocity in (b) along the direction x_f. In (a), the black dashed-dotted line is the position reference along x_f, i.e., u^{x_f}.



(a) (b)

Fig. 12. Estimated error between the measured output and the generic n-steps-ahead predictor $\underline{\lambda}_{n,1}$ (blue line with "◇"), $\hat{\bar{\varepsilon}}_{n,1}$ considering a scaling factor $\mu = 1.2$ (light blue line with "□"). Estimated worst case prediction error bound $\tau_{n,1}$ (red line with "○") and $\hat{\tau}_{n,1}$ considering the scaling factors $\eta = 1.02$ and $\mu = 1.02$ (light red line with "△") along (a) direction x_f and (b) y_f. Gray dashed lines show the trajectories of the estimation error in the validation dataset.

numerical stability, we included in the cost function (22) a term that penalizes the rate of change of the input

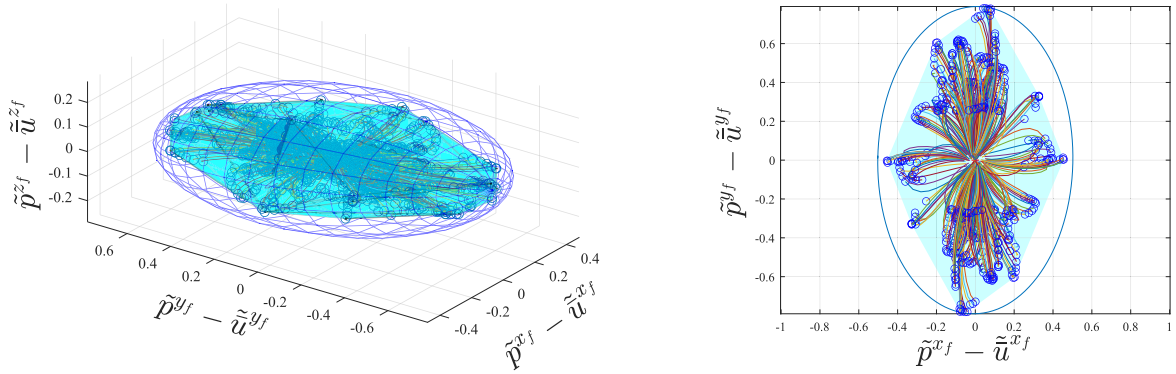
$$J\left(\boldsymbol{x}(k), \boldsymbol{U}, \boldsymbol{p}_t(k)\right) = \sum_{i=1}^{N} \|\boldsymbol{p}^I_{i|k} - \boldsymbol{p}_t(k)\|^2_Q + \sum_{i=0}^{N-1} \|\Delta \boldsymbol{u}^{I,II}_{i|k}\|^2_{\Delta_R} \tag{32}$$

where $\Delta_R \in \mathbb{R}^{3\times 3}$ is a suitable weight matrix for the input variation. To avoid unnecessary conservativeness in the input variation, we have chosen a weight Q much larger than Δ_R to prioritize the tracking of the desired target, i.e., $Q = 6 \cdot \mathbb{I}^3$ and $\Delta_R = 0.5 \cdot \mathbb{I}^3$. The set $\mathcal{P}_D$ has been selected as a polytopic outer-approximation of the ellipsoidal set $\boldsymbol{x}^T D \boldsymbol{x} \leq \delta$, where δ and D have been obtained with the following optimization problem that returns the minimum-volume sublevel set of a Lyapunov function $\{\boldsymbol{x} \in \mathbb{R}^6 | \boldsymbol{x}^T D \boldsymbol{x} \leq \delta\}$ containing the convex hull of the steady state error samples (33), as shown at the bottom of the next page. Fig. 13 shows the sets $\mathcal{P}_W$ and $\mathcal{E}(D)$ together with the steady state data considered ($\tilde{\boldsymbol{x}}(k+j; \tilde{\tilde{\boldsymbol{u}}}) - \begin{bmatrix} \tilde{\tilde{\boldsymbol{u}}} \\ \boldsymbol{0}^{3\times 1} \end{bmatrix}$), $\forall j > \underline{j}(r_L)$, and the evolution of the simulated tracking error according to the model (8). To assess the belonging of the ellipsoidal set $\mathcal{E}(D)$ to the admissible set $\mathcal{V}$ according to acceleration and velocity constraints as described in Assumption 2, the following two checks have been performed:

$$\boldsymbol{a}_{ck} < \bar{\boldsymbol{a}}, \qquad \boldsymbol{v}_{ck} < \bar{\boldsymbol{v}}$$

where $\boldsymbol{a}_{ck}$ and $\boldsymbol{v}_{ck}$ are the solutions of the following two problems, formulated componentwise:

$$\begin{aligned} \boldsymbol{a}_{ck} = \max_{\boldsymbol{x}(k)} \quad & |\boldsymbol{a}(k)| \\ \text{s.t.} \quad & \boldsymbol{x}(k) \in \mathcal{E}(D) \end{aligned}$$



(a) (b)

Fig. 13. Estimated sets $\mathcal{P}_W$ (light blue polytope) and $\mathcal{E}(D)$ (blue ellipsoid) projected onto the subspace (a) $\boldsymbol{v} = 0^{3\times1}$ and (b) $\boldsymbol{v} = \mathbf{0}^{3\times1}$, $p^{z_f} - u^{z_f} = 0$. Blue dots represent the steady-state errors $(\tilde{\boldsymbol{x}}(k+j;\tilde{\tilde{\boldsymbol{u}}}) - \begin{bmatrix}\tilde{\tilde{\boldsymbol{u}}} 0^{3\times1}\end{bmatrix})$, $\forall j > \underline{j}(r_L)$. The colored lines inside the polytopes are the tracking error trajectories of model (8) with initial condition inside $\mathcal{P}_W$ and subject to constant references $\tilde{\tilde{\boldsymbol{u}}}$.

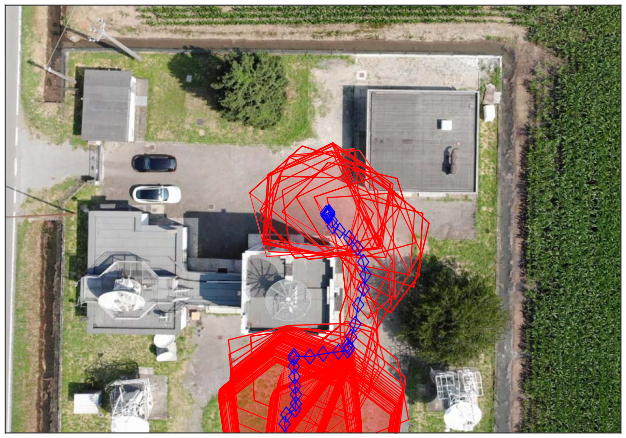


Fig. 14. Experimental test of the mt-MPC approach. Trajectory obtained with mt-MPC (blue line with "◊"). Safe set $\mathcal{S}(k)$ at different time steps k in red.

$$\begin{aligned} \boldsymbol{v}_{ck} = \max_{\boldsymbol{x}(k)} \quad & |\boldsymbol{v}(k)| \\ \text{s.t.} \quad & \boldsymbol{x}(k) \in \mathcal{E}(D) \end{aligned}$$

where $\boldsymbol{a}(k) = K_{\text{vel}}(K_{\text{pos}}(\boldsymbol{u}(k) - \boldsymbol{p}(k)) - \boldsymbol{v}(k))$ as described in (7).

Remark 6: Assumption 2 implies the stability of the identified control-oriented model. We checked this condition after the identification procedure, and, as highlighted in (33), we have considered the system's trajectory starting from steady-state error samples that satisfy the constraints obtained from experimental data.

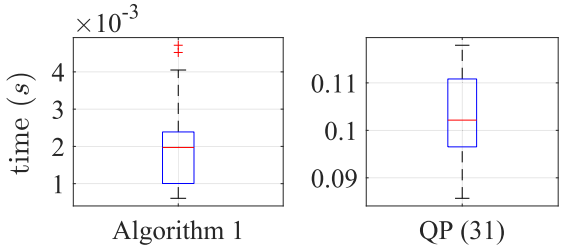


Fig. 15. Computational time required for Algorithm 1 and QP (27) during experimental tests.

As mentioned in Remark 5, we have replaced the constraint (27j) in the FHOCP (27) with a terminal zero-velocity constraint $\boldsymbol{v}_{N|k}^{II} = \mathbf{0}^{3\times1}$, and we have then artificially extended the horizon N by $\underline{j}(r_L)$ time steps when needed, to reduce the computational effort required to solve the problem.

Fig. 12 presents a visualization of the prediction horizon and of the uncertainty bounds until the terminal step $N + \underline{j}(r_L)$. The optimization problem (27) has been solved with Operator Splitting Quadratic Program (OSQP) solver [41] in the ROS node of the MPC law. We have properly selected the horizon N and the sampling time T_s in order to obtain a real-time implementation of the algorithm while still capturing the dynamic motion of the drone. In particular, using $T_s = 0.3$s and $N = 8$, the average execution times per sampling step of the QP (27) is about 0.1 s, and about 20 ms in addition to run Algorithm 1 with the available 2-D LiDAR. Fig. 14 shows the trajectory obtained with the mt-MPC approach and some of

$$\begin{aligned} \min_{D,\delta} \quad & \log\det D^{-1} \\ \text{s.t.} \quad & A^T D A - D < 0, \\ & D > 0 \left(\tilde{\boldsymbol{x}}\left(k+j;\tilde{\tilde{\boldsymbol{u}}}\right) - \begin{bmatrix}\tilde{\tilde{\boldsymbol{u}}} \\ \mathbf{0}^{3\times1}\end{bmatrix}\right)^T D \left(\boldsymbol{x}\left(k+j;\tilde{\tilde{\boldsymbol{u}}}\right) - \begin{bmatrix}\tilde{\tilde{\boldsymbol{u}}} \\ \mathbf{0}^{3\times1}\end{bmatrix}\right) \leq \delta \quad \forall j > \underline{j}(r_L) \end{aligned} \tag{33}$$

the safe sets at different time steps during the tests. Videos of the experimental setup and mt-MPC trajectories computation are available at https://youtu.be/-_kOhl6AI68.

Fig. 15 shows the boxplots of the execution times during experimental tests. Note that the computation time required to solve Algorithm 1 is drastically smaller in this implementation than in the one required in the simulations presented in Section V, due to the planar nature of the considered exteroceptive sensor.

VII. Conclusion

A novel MPC formulation, named mt-MPC, has been presented, where multiple trajectories are considered in the same optimization problem to trade-off conflicting objectives. The approach has been applied to the autonomous navigation of a multicopter drone in a priori unknown environment. An SM approach has been used to estimate the prediction error of a model obtained from measured data. A novel approach to approximate the feasible set with a convex polytope exploiting only real-time measurements of an exteroceptive sensor has been used, together with a strategy to guarantee obstacle avoidance in case of time-invariant environment and considering the computed prediction uncertainty. The obstacle-avoidance property has been theoretically proven and demonstrated experimentally. The experiments and simulation results show that mt-MPC can be implemented in real time on the considered low-cost hardware and safely navigate the system to destination, consistently with our theoretical analysis. Current research is aimed to apply the mt-MPC concept to reconfigurable and/or nonlinear systems, to include additional learning components in the problem, and to consider dynamic obstacles in the environment.

References

[1] S. A. Bagloee, M. Tavana, M. Asadi, and T. Oliver, "Autonomous vehicles: Challenges, opportunities, and future implications for transportation policies," *J. Mod. Transp.*, vol. 24, pp. 284–303, Dec. 2016.

[2] R. Bishop, "A survey of intelligent vehicle applications worldwide," in *Proc. IEEE Intell. Vehicles Symp.*, Oct. 2000, pp. 25–30.

[3] S. Tang and V. Kumar, "Autonomous flight," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 1, pp. 29–52, May 2018.

[4] S. Aggarwal and N. Kumar, "Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges," *Comput. Commun.*, vol. 149, pp. 270–299, Jan. 2020.

[5] T. Paul, T. R. Krogstad, and J. T. Gravdahl, "Modelling of UAV formation flight using 3D potential field," *Simul. Model. Pract. Theory*, vol. 16, no. 9, pp. 1453–1462, 2008.

[6] Y.-B. Chen, G.-C. Luo, Y.-S. Mei, J.-Q. Yu, and X.-L. Su, "UAV path planning using artificial potential field method updated by optimal control theory," *Int. J. Syst. Sci.*, vol. 47, no. 6, pp. 1407–1420, 2016.

[7] R. Geraerts, "Planning short paths with clearance using explicit corridors," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 1997–2004.

[8] Z. Chengjun and M. Xiuyun, "Spare a search approach for UAV route planning," in *Proc. IEEE Int. Conf. Unmanned Syst. (ICUS)*, Oct. 2017, pp. 413–417.

[9] M. Kothari and I. Postlethwaite, "A probabilistically robust path planning algorithm for UAVs using rapidly-exploring random trees," *J. Intell. Robot. Syst.*, vol. 71, pp. 231–253, Dec. 2013.

[10] A. Bemporad, C. A. Pascucci, and C. Rocchi, "Hierarchical and hybrid model predictive control of quadcopter air vehicles," in *Proc. ADHS*, 2009, pp. 14–19.

[11] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online UAV replanning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 5332–5339.

[12] D. Q. Mayne, M. M. Seron, and S. V. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, vol. 41, no. 2, p. 219–224, 2005.

[13] L. Chisci and G. Zappa, "Robustifying a predictive controller against persistent disturbances," in *Proc. Eur. Control Conf. (ECC)*, Aug. 1999, pp. 2419–2424.

[14] L. Chisci, J. A. Rossiter, and G. Zappa, "Systems with persistent disturbances: Predictive control with restricted constraints," *Automatica*, vol. 37, no. 7, pp. 1019–1028, 2001.

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

[16] A. Bemporad and C. Rocchi, "Decentralized linear time-varying model predictive control of a formation of unmanned aerial vehicles," in *Proc. IEEE Conf. Decis. Control Eur. Control Conf.*, Dec. 2011, pp. 7488–7493.

[17] M. A. Mousavi, Z. Heshmati, and B. Moshiri, "LTV-MPC based path planning of an autonomous vehicle via convex optimization," in *Proc. 21st Iranian Conf. Electr. Eng. (ICEE)*, May 2013, pp. 1–7.

[18] S. Sharma, "QCQP-tunneling: Ellipsoidal constrained agent navigation," in *Proc. IASTED Int. Conf. Robot.*, 2011, doi: 10.2316/P.2011.752-010.

[19] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Algorithmic Foundations of Robotics XI*. Cham, Switzerland: Springer, 2015, pp. 109–124.

[20] S. Liu et al., "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1688–1695, Feb. 2017.

[21] J. Tordesillas, B. T. Lopez, M. Everett, and J. P. How, "FASTER: Fast and safe trajectory planner for navigation in unknown environments," *IEEE Trans. Robot.*, vol. 38, no. 2, pp. 922–938, Apr. 2021.

[22] S. Hrabar, "Reactive obstacle avoidance for rotorcraft UAVs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2011, pp. 4967–4974.

[23] J. Park and N. Cho, "Collision avoidance of hexacopter UAV based on LiDAR data in dynamic environment," *Remote Sens.*, vol. 12, no. 6, p. 975, Mar. 2020.

[24] S. Sharma and M. E. Taylor, "Autonomous waypoint generation strategy for on-line navigation in unknown environments," in *Proc. IROS Workshop Robot Motion Planning, Online, Reactive, Real-Time*, Jul. 2012, pp. 37–48.

[25] V. J. Hodge, R. Hawkins, and R. Alexander, "Deep reinforcement learning for drone navigation using sensor data," *Neural Comput. Appl.*, vol. 333, pp. 1–19, Mar. 2020.

[26] C. Greatwood and A. G. Richards, "Reinforcement learning and model predictive control for robust embedded quadrotor guidance and control," *Auto. Robots*, vol. 43, no. 7, pp. 1681–1693, Oct. 2019.

[27] S. Bansal, V. Tolani, S. Gupta, J. Malik, and C. Tomlin, "Combining optimal control and learning for visual navigation in novel environments," in *Proc. Conf. Robot Learn.*, 2020, pp. 420–429.

[28] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, Jul. 2021, Art. no. 109597.

[29] J. Tordesillas, B. T. Lopez, and J. P. How, "Faster: Fast and safe trajectory planner for flights in unknown environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 1934–1940.

[30] J. P. Alsterda, M. Brown, and J. C. Gerdes, "Contingency model predictive control for automated vehicles," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2019, pp. 717–722.

[31] D. Saccani and L. Fagiano, "Autonomous UAV navigation in an unknown environment via multi-trajectory model predictive control," in *Proc. Eur. Control Conf. (ECC)*, Jun. 2021, pp. 1577–1582.

[32] M. Milanese and C. Novara, "Unified set membership theory for identification, prediction and filtering of nonlinear systems," *Automatica*, vol. 47, no. 10, pp. 2141–2151, 2011.

[33] M. Lauricella and L. Fagiano, "Set membership identification of linear systems with guaranteed simulation accuracy," *IEEE Trans. Autom. Control*, vol. 65, no. 12, pp. 5189–5204, Dec. 2020, doi: 10.1109/TAC.2020.2970146.

[34] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, "Visual simultaneous localization and mapping: A survey," *Artif. Intell. Rev.*, vol. 43, no. 1, pp. 55–81, 2015.

[35] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*. Philadelphia, PA, USA: SIAM, 1994.

[36] L. Ljung, "System identification," in *Signal Analysis and Prediction* (Applied and Numerical Harmonic Analysis), A. Procházka, J. Uhlíř, P. W. J. Rayner, and N. G. Kingsbury, Eds. Boston, MA, USA: Birkhäuser, 1998, doi: 10.1007/978-1-4612-1768-8_11.

[37] T. Schouwenaars, J. How, and E. Feron, "Receding horizon path planning with implicit safety guarantees," in *Proc. Amer. control Conf.*, vol. 6, Jun. 2004, pp. 5576–5581.
[38] D. Mayne, "Robust and stochastic model predictive control: Are we going in the right direction?" *Annu. Rev. Control*, vol. 41, pp. 184–192, Jan. 2016.
[39] S. Formentin and M. Lovera, "Flatness-based control of a quadrotor helicopter via feedforward linearization," in *Proc. IEEE Conf. Decis. Control Eur. Control Conf.*, Dec. 2011, pp. 6171–6176.
[40] Stanford Artificial Intelligence Laboratory. *Robotic Operating System*. Accessed: Nov. 1, 2022. [Online]. Available: https://www.ros.org
[41] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," *Math. Program. Comput.*, vol. 12, no. 4, pp. 637–672, Dec. 2020, doi: 10.1007/s12532-020-00179-2.

Danilo Saccani (Graduate Student Member, IEEE) received the B.Sc. degree in mechanical engineering and the M.Sc. degree in automation and control engineering from the Politecnico di Milano, Milan, Italy, in 2014 and 2019, respectively, where he is currently pursuing the Ph.D. degree in information technology, area systems and control with the Department of Electronic, Information and Bioengineering.

His current research interests include modeling, optimization and control of process systems, unmanned aerial vehicles, and model predictive control.

Leonardo Cecchin received the M.Sc. degree in automation and control engineering from the Politecnico di Milano, Milan, Italy, in 2020.

From 2020 to 2021, he was a Research Fellow with the Politecnico di Milano, where he was involved in graph-based approaches for exploration and mapping with autonomous multicopter drones. Since 2021, he has been carrying out a Ph.D. with Bosch Corporate Research, Stuttgart, Germany, where he has involving in the framework of the Marie Curie Initial Training Network "ELO-X." His current research interests include adaptive model predictive control, numerical optimization methods, and applications to industrial and automotive systems.

Lorenzo Fagiano (Senior Member, IEEE) received the Ph.D. degree in information and systems engineering from the Politecnico di Torino, Turin, Italy, in 2009.

From 2010 to 2013, he was a Marie Curie Fellow with the University of California at Santa Barbara, Santa Barbara, CA, USA, and ETH Zürich, Zürich, Switzerland. From 2013 to 2016, he was a Scientist and a Senior Scientist with ABB Switzerland, Zürich, Corporate Research. He is currently an Associate Professor of automation and control engineering with the Politecnico di Milano, Milan, Italy. His research interests include constrained estimation and control, set membership methods, and applications to industrial, robotic, and energy systems.

Dr. Fagiano was a recipient of the 2019 European Control Award, the Mission Innovation Champion Award 2019 for Italy, two Marie Curie Individual Fellowships in 2009 and 2012, the 2011 IEEE TRANSACTIONS ON SYSTEMS TECHNOLOGY Outstanding Paper Award, and the 2010 ENI Award Debut in Research Prize. He was an Associate Editor of the IEEE TRANSACTIONS OF CONTROL SYSTEMS TECHNOLOGY from 2015 to 2020.

Open Access funding provided by 'Università degli Studi di Trieste' within the CRUI CARE Agreement