POLITECNICO DI TORINO
Repository ISTITUZIONALE

Learning model predictive control for quadrotors minimum-time flight in autonomous racing scenarios

*Original*
Learning model predictive control for quadrotors minimum-time flight in autonomous racing scenarios / Calogero, L., Mammarella, M., Dabbene, F.. - ELETTRONICO. - 56:(2023), pp. 1063-1068. (22nd World Congress of the International Federation of Automatic Control (IFAC) Yokohama (Japan) 09/07/2023-14/07/2023) [10.1016/j.ifacol.2023.10.1705].

*Availability:*
This version is available at: 11583/2984072 since: 2023-11-24T11:56:34Z

*Publisher:*
Elsevier

*Published*
DOI:10.1016/j.ifacol.2023.10.1705

*Terms of use:*

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

*Publisher copyright*

(Article begins on next page)

30 October 2026

Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 56-2 (2023) 1063–1068

# Learning Model Predictive Control for Quadrotors Minimum-Time Flight in Autonomous Racing Scenarios

**Lorenzo Calogero** *, **Martina Mammarella** **, **Fabrizio Dabbene** **

* *Department of Electronics and Telecommunications, Politecnico di Torino, Turin, Italy (e-mail: lorenzo.calogero@polito.it)*
** *CNR-IEIIT, Turin, Italy (e-mail: {martina.mammarella, fabrizio.dabbene}@ieiit.cnr.it)*

**Abstract:** In this paper, we design a Learning Model Predictive Control (LMPC) algorithm for quadrotors autonomous racing. The proposed algorithm allows to define a highly customizable 3D race track, in which multiple types of obstacles can be inserted. The controller is then able to autonomously find the best trajectory minimizing the quadrotor lap time, by learning from data coming from previous flights within the track, ensuring also the avoidance of all the obstacles therein. We also present novel relaxation approaches for the LMPC optimization problem, that allow to reduce it from a mixed-integer nonlinear program to a quadratic program. The LMPC algorithm is tested via several software-in-the-loop simulations, showing that the algorithm has learned to fly the quadrotor aggressively and dexterously, managing to both find the minimum-time trajectory and avoid the obstacles inside the track.

Copyright © 2023 The Authors. This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0/)

*Keywords:* iterative learning and repetitive control systems, model predictive control, autonomous vehicles control, motion planning, unmanned aerial vehicles.

## 1. INTRODUCTION

Unmanned Aerial Vehicles (UAVs), such as quadrotors, are aircrafts becoming increasingly employed to assist humans in a wide range of tasks, typically in constrained and cluttered environments (Li et al., 2022). In this context, the field of autonomous drone racing has emerged, with the aim of fostering research in drone agile navigation, to develop new and increasingly performing control schemes. Racing drones are typically required to fly across a series of gates, placed on a closed track, avoiding collisions and minimizing the lap time (Penicka et al., 2022). In this scenario, the aim of this paper is to investigate the use of Learning Model Predictive Control (LMPC) for quadrotors autonomous racing.

LMPC is a novel iterative learning control technique that exploits past information, coming from previous executions of the given control task, to autonomously improve its performance over time (Rosolia and Borrelli, 2017a). For our purposes, we develop a LMPC algorithm controlling the quadrotor motion within a closed 3D race track, in which multiple obstacles can be inserted. The algorithm, through its learning capabilities, is able to autonomously find the best trajectory minimizing the quadrotor lap time and also to avoid all the obstacles therein. Being MPC one of the most flexible and versatile control techniques for MIMO systems, several works have investigated its use for quadrotors control in constrained environments, with the addition of features such as perception capability (Li et al., 2020) (Bicego et al., 2020) or data-driven refinement of the system model (Torrente et al., 2021); approaches such as those of Penicka et al. (2022) and Han et al. (2021), instead, plan an optimal minimum-time trajectory in advance and use it as reference. The advantages of LMPC, with respect to the above approaches, are that it can be employed online to autonomously find the minimum-time trajectory, without the need of trajectory planning; moreover, collected data directly improves the control performances over time and is not limited to the identification of the system model.

LMPC has been primarily applied to ground vehicles (Rosolia and Borrelli, 2019) (Brunke, 2020), with limited research on its use for quadrotors (Li et al., 2022). Our study achieves all the relevant results from these previous works, while also introducing several novelties. Specifically, our work employs Frenet coordinates to describe the quadrotor motion, allowing for a highly customizable track description, requiring only linear constraints. Additionally, we present a novel relaxation approach, based on convex piecewise-linear interpolation, allowing to reduce the LMPC optimization problem to a quadratic program.

The remainder of the paper is structured as follows. In Section 2, we provide an overview on the general theoretical formulation of LMPC while in Section 3 we present the quadrotor model in Frenet coordinates. In Section 4 we show how to adapt LMPC for the purpose of controlling quadrotors in the drone racing scenario. Simulations of the LMPC algorithm for quadrotors are reported in Section 5, showing its functionality and good performances. Main conclusions are drawn in Section 6.

2405-8963 Copyright © 2023 The Authors. This is an open access article under the CC BY-NC-ND license.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2023.10.1705

## 2. LMPC THEORETICAL FORMULATION

LMPC is an MPC-based iterative learning control strategy that is able to autonomously improve its performances, by executing the control task multiple times and collecting data from each of these iterations. In this section, we recall the main features of LMPC, first proposed by Rosolia and Borrelli (2017b).

### 2.1 Learning Model Predictive Control (LMPC)

Let us consider a generic discrete-time dynamical system

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{u} \in \mathbb{R}^m$ are the system state and input, respectively. Each $j$-th LMPC iteration generates a state trajectory $\boldsymbol{X}^j$ and a corresponding input sequence $\boldsymbol{U}^j$, defined as

$$\boldsymbol{X}^j = (\boldsymbol{x}_0^j, \boldsymbol{x}_1^j, ..., \boldsymbol{x}_{T_j}^j) \tag{2a}$$

$$\boldsymbol{U}^j = (\boldsymbol{u}_0^j, \boldsymbol{u}_1^j, ..., \boldsymbol{u}_{T_j-1}^j) \tag{2b}$$

where $j$ is the iteration number and $T_j$ is the time instant at which the task is completed.

LMPC is formulated as *repetitive*, meaning that the $j$-th iteration starts from the last state of the previous iteration, i.e., $\boldsymbol{x}_0^j = \boldsymbol{x}_{T_{j-1}}^{j-1}, \forall j \geq 1$; only iteration 0 starts from an initial state $\boldsymbol{x}_S$, i.e., $\boldsymbol{x}_0^0 = \boldsymbol{x}_S$. The task is considered complete (at time $T_j$) when the system reaches a goal state $\boldsymbol{x}_F$ without violating any constraint.

At each iteration, the LMPC algorithm collects the states of the generated trajectory, with their related costs, and uses such data to improve the control performances of the next iteration. The main elements of the LMPC algorithm (Rosolia and Borrelli, 2017b) are briefly recalled next.

*Sampled safe set*

The states of each generated trajectory are collected within the *sampled safe set*, which is defined as

$$SS^j = \left\{ \bigcup_{i \in G^j} \bigcup_{k=0}^{\infty} \boldsymbol{x}_k^i \right\} \tag{3a}$$

$$G^j = \left\{ i \in [0, j] : \lim_{k \to \infty} \boldsymbol{x}_k^i = \boldsymbol{x}_F \right\} \tag{3b}$$

where $SS^j$ is a discrete set storing all the states $\boldsymbol{x}_k$, with $k \in [0, \infty]$, composing the state trajectories generated by successful iterations of the LMPC algorithm, up to iteration $j$. In particular, $SS^0$ is initialized with a first feasible trajectory, that is sub-optimal, and is generated by means of a basic reference tracking control method (in our case, classic MPC).

*Terminal cost function*

We define the *cost-to-go* of the trajectory $j$ at time $k$ as

$$J_{[k,\infty]}^j(\boldsymbol{x}_k^j) \equiv J_k^j = \sum_{t=k}^{\infty} h(\boldsymbol{x}_t^j, \boldsymbol{u}_t^j) \tag{4}$$

where $h$ is the stage cost function. The cost-to-go corresponds to the cumulative cost of the part of the $j$-th trajectory starting from state $\boldsymbol{x}_k^j$. When $k = 0$, the cost-to-go is denoted as *iteration cost* ($J_0^j$) and it corresponds to the cost of the whole trajectory, i.e., it quantifies the control algorithm performance at each $j$-th iteration.

Then, we define the *terminal cost function* $Q^j(\boldsymbol{x})$, over the $SS^j$, as

$$Q^j(\boldsymbol{x}) = \begin{cases} \min\limits_{(i,k) \in F^j(\boldsymbol{x})} J_{[k,\infty]}^i(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in SS^j \\ +\infty & \text{if } \boldsymbol{x} \notin SS^j \end{cases} \tag{5a}$$

$$F^j(\boldsymbol{x}) = \left\{ (i,k) \in [0, j] \times [0, \infty] : \boldsymbol{x} = \boldsymbol{x}_k^i \text{ for } \boldsymbol{x}_k^i \in SS^j \right\}. \tag{5b}$$

According to this definition, the terminal cost function $Q^j(\boldsymbol{x})$ assigns, to every state $\boldsymbol{x}$ in $SS^j$, the minimum cost-to-go along the trajectories in $SS^j$ departing from $\boldsymbol{x}$.

*LMPC formulation*

The LMPC optimization problem is constructed according to the MPC formulation, where at time $k$ of iteration $j$, $SS^{j-1}$ is the terminal set and $Q^{j-1}(\boldsymbol{x})$ is the terminal cost function, i.e.,

$$(\boldsymbol{X}_k^{j^*}, \boldsymbol{U}_k^{j^*}) = \underset{\boldsymbol{X}_k, \boldsymbol{U}_k}{\operatorname{argmin}} \sum_{t=0}^{N-1} h(\boldsymbol{x}_{t|k}, \boldsymbol{u}_{t|k}) + Q^{j-1}(\boldsymbol{x}_{N|k}) \tag{6a}$$

s.t.

$$\boldsymbol{x}_{t+1|k} = \boldsymbol{f}(\boldsymbol{x}_{t|k}, \boldsymbol{u}_{t|k}),\ t = [0, \dots, N-1] \tag{6b}$$

$$\boldsymbol{x}_{0|k} = \boldsymbol{x}_k^j \tag{6c}$$

$$\boldsymbol{x}_{t|k} \in \mathcal{X},\ \ \boldsymbol{u}_{t|k} \in \mathcal{U}, \quad t = [0, \dots, N-1] \tag{6d}$$

$$\boldsymbol{x}_{N|k} \in SS^{j-1} \tag{6e}$$

where $\boldsymbol{x}_{t|k}$ and $\boldsymbol{u}_{t|k}$ are the state and input predicted $t$ steps ahead at time $k$, respectively.

The addition of the terminal cost function and constraint in (6) guarantees LMPC asymptotically stability and recursively feasibility (Rosolia and Borrelli, 2017b). Moreover, as proved in Rosolia and Borrelli (2017b), the formulation (6) ensures that: $(i)$ between two successive iterations, the iteration cost is non-increasing; $(ii)$ the trajectories tend to converge to the solution of the infinite-horizon version of (6).

## 3. QUADROTOR MODELLING

The quadrotor dynamical model is described in (Das et al., 2009) as the following set of differential equations

$$\begin{cases} \ddot{x} = \frac{1}{m}(c_\phi s_\theta c_\psi + s_\phi s_\psi) u_1 - \frac{\beta_x}{m}\dot{x} \\ \ddot{y} = \frac{1}{m}(c_\phi s_\theta s_\psi - s_\phi c_\psi) u_1 - \frac{\beta_y}{m}\dot{y} \\ \ddot{z} = -g + \frac{1}{m} c_\phi c_\theta u_1 - \frac{\beta_z}{m}\dot{z} \\ \ddot{\phi} = \frac{I_y - I_z}{I_x}\dot{\theta}\dot{\psi} + \frac{1}{I_x} u_2 \\ \ddot{\theta} = \frac{I_z - I_x}{I_y}\dot{\phi}\dot{\psi} + \frac{1}{I_y} u_3 \\ \ddot{\psi} = \frac{I_x - I_y}{I_z}\dot{\phi}\dot{\theta} + \frac{1}{I_z} u_4 \end{cases} \tag{7}$$

where $s_x \equiv \sin(x)$ and $c_x \equiv \cos(x)$. In (7), $(x, y, z)$ are the Cartesian coordinates of the quadrotor center of mass (CoM) position in the inertial frame, $(\phi, \theta, \psi)$ are the orientation angles of the body frame with respect to the inertial one, $m$ and $(I_x, I_y, I_z)$ are the mass and principal moments of inertia of the quadrotor, respectively, $\beta_x, \beta_y, \beta_z$ are the air drag force coefficients, and $(u_1, u_2, u_3, u_4) = (f, \tau_\theta, \tau_\phi, \tau_\psi)$ are the control inputs,

corresponding to the quadrotor thrust force and torques related to each orientation angle. respectively. This model is later discretized using the forward Euler method with constant time step $T$.

### 3.1 Frenet coordinates

In order to define the linear track boundary constraints (6d), we convert the quadrotor model (7) from Cartesian to Frenet coordinates, which are a coordinate system that describes the position of a point $P$ on the plane with respect to a reference curve $\boldsymbol{\gamma}$, called Frenet curve, and by means of two coordinates, as shown in Fig. 1:

- the signed curvilinear abscissa $s$, i.e., the length of the curve $\boldsymbol{\gamma}$ from its origin to the orthogonal projection of $P$ on $\boldsymbol{\gamma}$;
- the signed distance $d$ from $\boldsymbol{\gamma}$, i.e., the lateral distance between $P$ and its orthogonal projection on $\boldsymbol{\gamma}$.

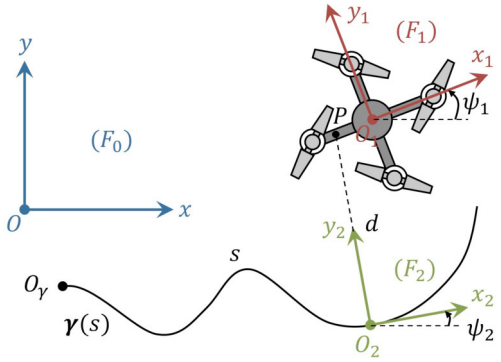



Fig. 1. Frenet coordinates.

Let us assume that $P$ to coincide with the CoM of the quadrotor. Then, the Cartesian-to-Frenet conversion is defined in (Morin and Samson, 2008) as follows, i.e.,

$$\begin{cases} \dot{s} = \frac{1}{1-K(s)d}\left(\dot{x}\cos\psi_2 + \dot{y}\sin\psi_2\right) \\ \dot{d} = -\dot{x}\sin\psi_2 + \dot{y}\cos\psi_2 \\ \dot{\psi}_2 = \frac{K(s)}{1-K(s)d}\left(\dot{x}\cos\psi_2 + \dot{y}\sin\psi_2\right) \end{cases} \tag{8}$$

where $K(s)$ is the curvature function of $\boldsymbol{\gamma}$. Through Frenet coordinates, the track shape is embedded within the quadrotor model (7) by means of the function $K(s)$, and the track width is then defined with a simple bound constraint on the state $d$ (6d), assuming that $\boldsymbol{\gamma}$ coincides with the track centerline.

### 3.2 Affine time-variant model for LMPC prediction

The quadrotor model equations, that are inserted in the LMPC optimization problem as equality constraints (6b), are linearized, at each time instant $k$, around the current operating point of the system (i.e., the current state $\boldsymbol{x}_k^j$ and the previous input $\boldsymbol{u}_{k-1}^j$). Hence, the prediction model (6b) becomes an Affine Time-Variant (ATV) model of the form

$$\begin{aligned} &\boldsymbol{x}_{t+1|k} = \boldsymbol{A}_k\boldsymbol{x}_{t|k} + \boldsymbol{B}_k\boldsymbol{u}_{t|k} + \boldsymbol{c}_k, \quad t = [0,\dots,N-1] \\ &\boldsymbol{A}_k = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}(\boldsymbol{x}_k, \boldsymbol{u}_{k-1}), \boldsymbol{B}_k = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}(\boldsymbol{x}_k, \boldsymbol{u}_{k-1}) \\ &\boldsymbol{c}_k = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_{k-1}) - \boldsymbol{A}_k\boldsymbol{x}_k - \boldsymbol{B}_k\boldsymbol{u}_{k-1}. \end{aligned} \tag{9}$$

## 4. LMPC FOR QUADROTORS

In this section, we describe the adaptation of the LMPC algorithm for the purpose of controlling racing quadrotors. The methods presented here represent a new contribution to the LMPC algorithm of Rosolia and Borrelli (2017b), in terms of relaxation procedures and application to tasks in constrained environments.

### 4.1 LMPC relaxation

We report here a series of relaxations for the LMPC optimization problem (6), that allow to reduce it from a mixed-integer nonlinear to a quadratic program.

*Sampled safe set relaxation*

A presented in (Rosolia and Borrelli, 2019), the sampled safe set $SS^j$ can be relaxed into its convex hull, denoted as *convex safe set* $CS^j$, through the barycentric approximation of $SS^j$, i.e.,

$$\begin{aligned} &CS^j = \text{conv}(SS^j) = (\boldsymbol{X}^0, ..., \boldsymbol{X}^j)\boldsymbol{\lambda}^\top \\ &\boldsymbol{\lambda} = (\lambda_0^0, \lambda_1^0, ..., \lambda_{T_0}^0, ..., \lambda_0^j, \lambda_1^j, ..., \lambda_{T_j}^j), \\ &\boldsymbol{\lambda} \geq 0, \quad ||\boldsymbol{\lambda}||_1 = 1. \end{aligned} \tag{10}$$

Then, being $CS^j$ a convex polytope, we can express it with its H-representation, obtaining a linear and convex definition of the the terminal constraint (6e).

*Terminal cost function relaxation*

Since we need to extend the terminal cost function $Q^j(\boldsymbol{x})$ over the continuous convex set $CS^j$, we can we compute the barycentric approximation also for $Q^j$, i.e., $P^j$, which is defined by (Rosolia and Borrelli, 2019) as follows

$$\begin{aligned} &P^j(\boldsymbol{x}) = \text{conv}(Q^j(\boldsymbol{x})) = \\ &= \min_{\boldsymbol{\lambda}\geq 0} \left(J^0_{[0,T_0]}(\boldsymbol{x}_0^0), J^0_{[1,T_0]}(\boldsymbol{x}_1^0), ..., J^j_{[0,T_j]}(\boldsymbol{x}_0^j), ...\right)\boldsymbol{\lambda}^\top \\ &\boldsymbol{\lambda} = (\lambda_0^0, \lambda_1^0, ..., \lambda_{T_0}^0, ..., \lambda_0^j, \lambda_1^j, ..., \lambda_{T_j}^j), \\ &\boldsymbol{\lambda} \geq 0, \quad ||\boldsymbol{\lambda}||_1 = 1, \quad (\boldsymbol{X}^0, ..., \boldsymbol{X}^j)\boldsymbol{\lambda}^\top = \boldsymbol{x}. \end{aligned} \tag{11}$$

To practically include the relaxed terminal cost function in the LMPC optimization problem, we compute $P^j(\boldsymbol{x})$ by interpolating the states contained in $SS^j = \{\boldsymbol{X}^0, ..., \boldsymbol{X}^j\} = \{\boldsymbol{x}_0^0,\ \boldsymbol{x}_1^0,\ ...,\ \boldsymbol{x}_{T_j}^j\}$ and the corresponding values of the terminal cost function $Q^j(SS^j) = \{J_0^0, J_1^0, ..., J_{T_j}^j\}$ by means of a convex piecewise-linear function $f(\boldsymbol{x})$ (Magnani and Boyd, 2009), i.e., we consider $f$ as a $k$-term *max-affine* function of the form

$$f(\boldsymbol{x}) = \max\left(\boldsymbol{a}_1^\top\boldsymbol{x} + b_1, ..., \boldsymbol{a}_k^\top\boldsymbol{x} + b_k\right) \tag{12}$$

Max-affine functions can be visualized as an intersection of $k$ planes in the $\mathbb{R}^n \times \mathbb{R}$ space, of which we take the envelope (which corresponds to the max operation). Then, we set up a parametric fitting problem for $f$, where $\boldsymbol{\alpha} = (\boldsymbol{a}_1, ..., \boldsymbol{a}_k, b_1, ..., b_k) \in \mathbb{R}^{k(n+1)}$ is the vector of parameters to be determined through the interpolation. The fitting problem can be solved by means of the algorithm proposed in (Magnani and Boyd, 2009) are recalled hereafter:

- create $k$ partitions of the $m$ data points $\{(\boldsymbol{x}_0^0, J_0^0), \dots, (\boldsymbol{x}_{T_j}^j, J_{T_j}^j)\} \equiv \{(\boldsymbol{x}_1, J_1), \dots, (\boldsymbol{x}_m, J_m)\}$;

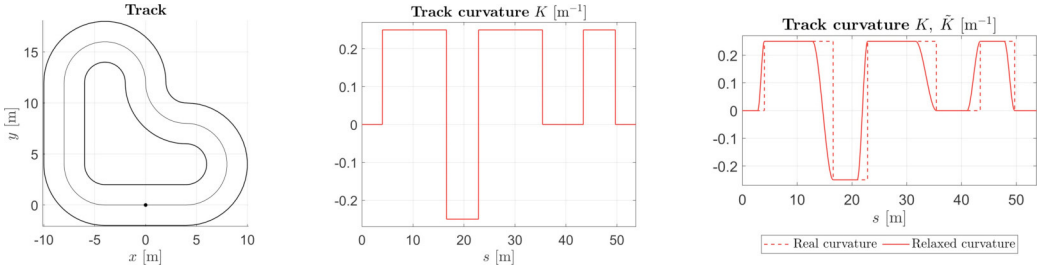



Fig. 2. Example of planar track, with its curvature function $K(s)$ and relaxed curvature function $\tilde{K}(s)$.

- on each $j$-th partition, solve analytically the linear least-squares fitting problem associated to the $j$-th affine function composing $f$, obtaining a first estimate of the coefficients $\boldsymbol{a}_j$ and $b_j$;
- update each partition on the base of the current coefficients value;
- iterate the algorithm until either the coefficients converge or a maximum number of iterations is reached.

For $j = 1, \dots, k$, let $P_j^{(l)}$ be a partition of the data indices $\{1, ..., m\}$ at the $l$-th iteration of the algorithm such that $P_j^{(l)} \subseteq \{1, ..., m\}$ and it holds that

$$\bigcup_j P_j^{(l)} = \{1, ..., m\}, \quad P_i^{(l)} \cap P_j^{(l)} = \emptyset \quad \text{for } i \neq j. \tag{13}$$

Denoting as $\boldsymbol{a}_j^{(l)}$ and $b_j^{(l)}$ the values of the parameters at the $l$-th iteration of the algorithm, the next values, i.e., $\boldsymbol{a}_j^{(l+1)}$ and $b_j^{(l+1)}$, are those that minimize

$$\sum_{i \in P_j^{(l)}} \left( \boldsymbol{a}^\top \boldsymbol{x}_i + b - J_i \right)^2 \tag{14}$$

which is the linear least-squares problem associated to the $j$-th affine function composing $f$, restricted to only the data in the $j$-th partition.

Using the new values of the coefficients, we update the partition to obtain $P_j^{(l+1)}$, by assigning $i$ to $P_j^{(l+1)}$ such that

$$f^{(l+1)}(\boldsymbol{x}_i) = \max_{s=1,\dots,k} \left( \boldsymbol{a}_s^{(l+1)^\top} \boldsymbol{x}_i + b_s^{(l+1)} \right) = \\ = \boldsymbol{a}_j^{(l+1)^\top} \boldsymbol{x}_i + b_j^{(l+1)}. \tag{15}$$

This means that $P_j^{(l+1)}$ is the set of indices $i$ for which the affine function $\boldsymbol{a}_j^{(l+1)^\top} \boldsymbol{x}_i + b_j^{(l+1)}$ is the maximum for the data point $\boldsymbol{x}_i$.

This algorithm is iterated, up to a maximum number of iterations, until the coefficients vector $\boldsymbol{\alpha}^{(l)}$ converges to a steady-state value or, equivalently, if the partitions $P_j^{(l)}$ do not change anymore after a certain iteration. Finally, the interpolated terminal cost function $f(\boldsymbol{x}) \equiv \tilde{P}^j(\boldsymbol{x})$ is inserted in (6a) by means of an additional optimization variable $c$ and an additional linear constraint as stated in (Magnani and Boyd, 2009), solving the following optimization problem

$$f(\boldsymbol{x}) = \tilde{P}^j(\boldsymbol{x}) = \min_c \ c \tag{16}$$
$$\text{s.t. } \ c \geq \boldsymbol{a}_s^{j^\top} \boldsymbol{x} + b_s^j, \ \ s = 1, ..., k.$$

### 4.2 Track definition

The 3D race track is vertically bounded and horizontally delimited by a planar track (Fig. 3). Among the planar tracks, we consider those composed by an arbitrary sequence of straight lines and circular curves, of any length and angle; in this way, $K(s)$ is a constant piecewise function (Fig. 2).

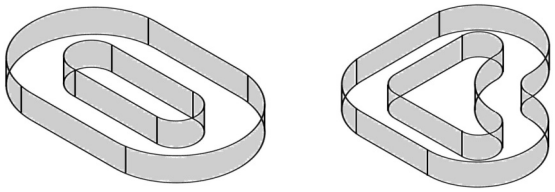


Fig. 3. Examples of race tracks.

To properly describe the shape of the track within the LMPC optimization problem, two technical devices are employed: *curvature propagation* and *curvature relaxation*. Curvature propagation consists in assuming $K(s)$ as constant and equal to its initial value $K(s_k^j) \equiv K$ within the LMPC optimization problem. This is done to avoid the need of analytically inserting the function $K(s)$ in the model equation constraints (6b). On the other hand, curvature relaxation, consists in relaxing the vertical edges of $K(s)$ with third-order polynomials, obtaining the relaxed function $\tilde{K}(s)$ (Fig. 2). This is done since the LMPC prediction is falsified by curvature propagation (i.e., being $K$ constant in the optimization problem), which may lead to bad control performances; by relaxing $K(s)$, the gradual change of the curvature edges allows the LMPC to better predict the future change of curvature, even if it is still considered as constant in each optimization problem.

### 4.3 Obstacle avoidance

*Obstacle avoidance* is an essential feature of the proposed algorithm since we are targeting drone racing. Specifically, three types of obstacles can be inserted in the track (Fig. 4). Such obstacles are described as local restrictions of the bounds (6d) on the states $z$ (altitude) and $d$ (lateral distance), and they are implemented exploiting the same propagation and relaxation method used for the track curvature.

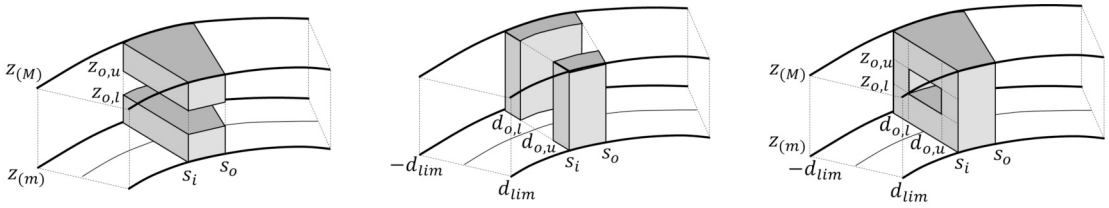



Fig. 4. Types of obstacles.

### 4.4 Cost function

The stage cost function $h$ of the LMPC optimization problem (6) is chosen to be quadratic

$$h(\boldsymbol{x}_{t|k}, \boldsymbol{u}_{t|k}, \boldsymbol{x}_{t-1|k}, \boldsymbol{u}_{t-1|k}) =$$
$$= \|\boldsymbol{x}_{t|k} - \boldsymbol{x}_F\|^2_{\boldsymbol{P}} \tag{17a}$$
$$+ \|\boldsymbol{x}_{t|k} - \boldsymbol{x}_r\|^2_{\boldsymbol{Q}} + \|\boldsymbol{u}_{t|k}\|^2_{\boldsymbol{R}} \tag{17b}$$
$$+ \|\boldsymbol{x}_{t|k} - \boldsymbol{x}_{t-1|k}\|^2_{\boldsymbol{Q}_\Delta} + \|\boldsymbol{u}_{t|k} - \boldsymbol{u}_{t-1|k}\|^2_{\boldsymbol{R}_\Delta} \tag{17c}$$

and it is composed by the following terms [1]:

- $\|\boldsymbol{x}_{t|k} - \boldsymbol{x}_F\|^2_{\boldsymbol{P}}$: it allows the algorithm to find the minimum-time trajectory, since it quantifies the distance from the goal state $\boldsymbol{x}_F$, corresponding to the finish line of the track (i.e., $s_F = L_{track}$), to the predicted state $\boldsymbol{x}_{t|k}$; thus, this cost term tends to minimize the travelled distance and, being the time step $T$ constant, also the lap time.
- $\|\boldsymbol{x}_{t|k} - \boldsymbol{x}_r\|^2_{\boldsymbol{Q}} + \|\boldsymbol{u}_{t|k}\|^2_{\boldsymbol{R}}$: it acts as reference tracking term (where $\boldsymbol{x}_r$ is the reference state) for all the other relevant states that are not present in (17a) (such as the altitude $z$ and the yaw angle $\psi$); it also penalizes the amplitude of the inputs.
- $\|\boldsymbol{x}_{t|k} - \boldsymbol{x}_{t-1|k}\|^2_{\boldsymbol{Q}_\Delta} + \|\boldsymbol{u}_{t|k} - \boldsymbol{u}_{t-1|k}\|^2_{\boldsymbol{R}_\Delta}$: it penalizes the variation of states and inputs, forcing the quadrotor to follow a smoother trajectory, without abrupt changes in its velocity and acceleration.

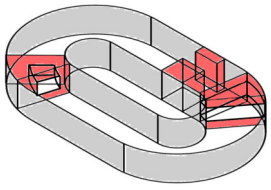


Fig. 5. Example of race track with obstacles.

## 5. SIMULATIONS AND RESULTS

In this section, we report the results of the simulations conducted on the LMPC algorithm for quadrotors. The algorithm is employed in software-in-the-loop simulations, in which it controls the quadrotor model (7), on a certain number of different tracks, which include also obstacles, and in various operative conditions.

### 5.1 Software implementation

The LMPC algorithm presented in Section 4 has been implemented in MATLAB® 2021a. For the formulation of the LMPC optimization problem, it has been used the third-party toolbox YALMIP, which provides a custom syntax and parser to express optimization problems in symbolic form (Löfberg, 2004). The simulations are run with an 11th Gen Intel® Core™ i7 CPU. The full MATLAB code implementing the LMPC algorithm for quadrotors, together with all the numerical data, is available in the following GitHub repository: `github.com/lorenzocalogero/LMPC_quadrotors`.

### 5.2 Simulations and results

We provide the results of three different case studies (Fig. 6): the first two (Fig. 6a-b) use the tracks shown in Fig. 3, which have no obstacles; the third one (Fig. 6c) uses the track shown in Fig. 5, which has three obstacles, one for each type (see Fig. 4). For each case study, we have: the quadrotor trajectories on the horizontal plane, overlaid by their colored velocity profile (Fig. 6-1); the trajectories on the vertical plane (Fig. 6-2); the iteration cost and lap time values with respect to the number of iterations (Fig. 6-3); and the quadrotor planar velocity with respect to its position in the track (Fig. 6-4).

From Fig. 6-1, we can observe that in each case study the LMPC has learned to fly the quadrotor aggressively, adopting an optimal shape for the trajectory and smartly increasing the velocity of the drone over the track. Specifically, from Fig. 6-4, we see that the velocity profiles steadily increase, up to triplicating the drone average speed. We also notice that the trajectories tend to converge, in accordance with the LMPC theorem cited in Section 2. Moreover, from Fig. 6-3, we notice that the iteration cost and lap time monotonically decrease as the number of iterations increases (with small steady-state oscillations due to the optimization problem relaxations), in accordance with the LMPC theorem of non-increasing iteration cost (see Section 2). The altitude $z$ (shown in Fig. 6-2), being the start and finish lines independent from it, is controlled through the term (17b) of the cost function (meaning that $z$ does not take part in the LMPC optimization); specifically, $z$ tracks (not strictly) the average altitude defined by the track vertical borders. Finally, in Fig. 6c-1, we see that the quadrotor is able to avoid any collision with the obstacles inserted in the track.

From a computational viewpoint, the maximum execution time of 1 step of the LMPC algorithm among all the simulations was equal to 28 ms. This value is lower than the chosen time step, i.e., $T = 0.1$ s, thus presumably resulting suitable for real-time applications.

[1] $\|\boldsymbol{x}\|^2_{\boldsymbol{A}}$ is equal to $\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x}$.

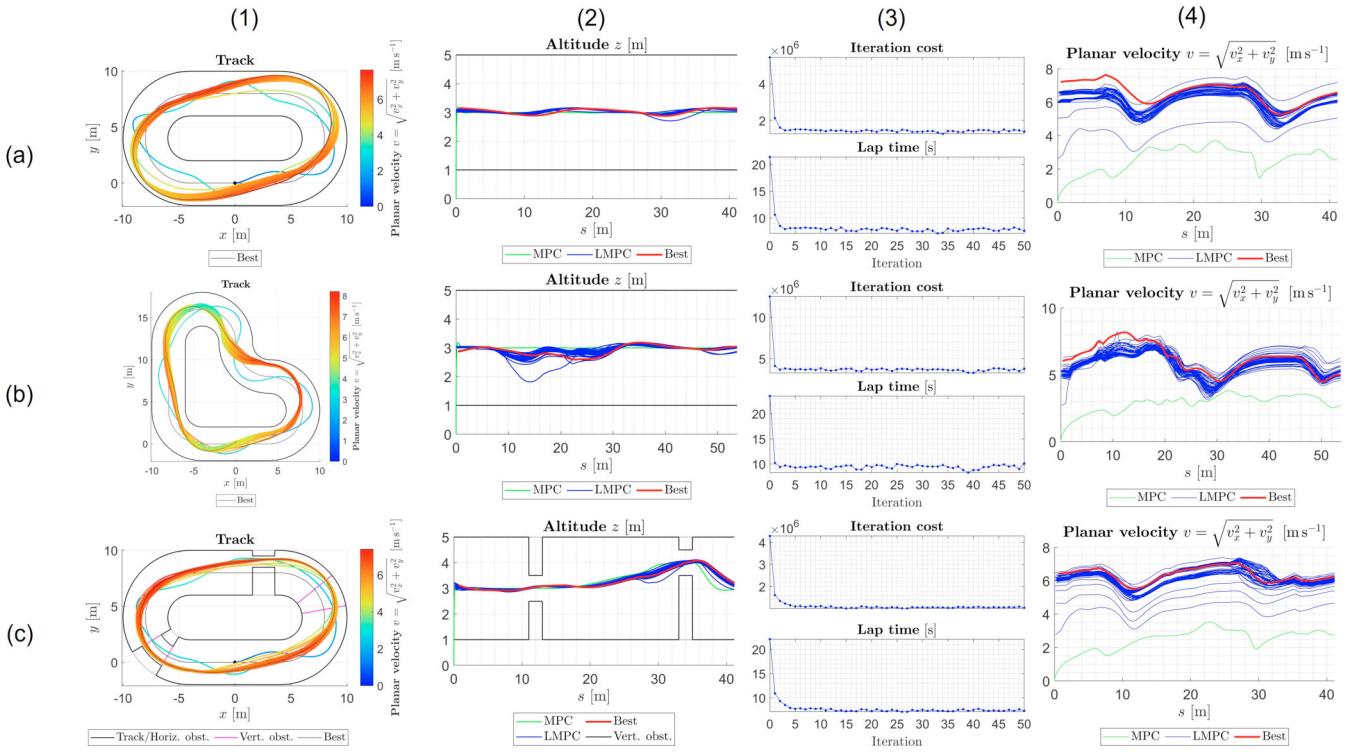



Fig. 6. Simulations of the LMPC algorithm for quadrotors.

## 6. CONCLUSIONS

In this paper, a LMPC algorithm for quadrotors autonomous racing has been presented. With respect to the basic LMPC algorithm, new relaxation techniques for the optimization problem have been developed, among which the convex piecewise-linear interpolation of the terminal cost function and the use of Frenet coordinates to obtain linear track constraints. Also, novel approaches for implementing the track, such as curvature/obstacles propagation and relaxation, have been developed. The proposed algorithm has been tested via software-in-the-loop simulations, proving its correct functionality and very good performances: the quadrotor has learned to fly aggressively, achieving lap time minimization and obstacle avoidance.

## REFERENCES

Bicego, D., Mazzetto, J., Carli, R., Farina, M., and Franchi, A. (2020). Nonlinear model predictive control with enhanced actuator model for multi-rotor aerial vehicles with generic designs. *Journal of Intelligent & Robotic Systems*, 100, 1213–1247.

Brunke, L. (2020). Learning Model Predictive Control for Competitive Autonomous Racing.

Das, A., Subbarao, K., and Lewis, F. (2009). Dynamic inversion with zero-dynamics stabilisation for quadrotor control. *IET Control Theory & Applications*, 3(3), 303–314.

Han, Z., Wang, Z., Pan, N., Lin, Y., Xu, C., and Gao, F. (2021). Fast-Racing: An Open-Source Strong Baseline for SE(3) Planning in Autonomous Drone Racing. *IEEE Robotics and Automation Letters*, 6(4), 8631–8638.

Li, G., Tunchez, A., and Loianno, G. (2022). Learning model predictive control for quadrotors. In *2022 IEEE International Conference on Robotics and Automation*, 5872–5878.

Li, S., Ozo, M.M.O.I., De Wagter, C., and De Croon, G.C.H.E. (2020). Autonomous drone race: A computationally efficient vision-based navigation and control strategy. *Robotics and Autonomous Systems*, 133.

Löfberg, J. (2004). YALMIP: A toolbox for modeling and optimization in MATLAB. In *2004 IEEE International Conference on Robotics and Automation*, 284–289.

Magnani, A. and Boyd, S.P. (2009). Convex piecewise-linear fitting. *Optimization and Engineering*, 10, 1–17.

Morin, P. and Samson, C. (2008). *Motion control of wheeled mobile robots*, 799–826. Springer.

Penicka, R., Song, Y., Kaufmann, E., and Scaramuzza, D. (2022). Learning minimum-time flight in cluttered environments. *IEEE Robotics and Automation Letters*, 7(3), 7209–7216.

Rosolia, U. and Borrelli, F. (2017a). Learning model predictive control for iterative tasks: A computationally efficient approach for linear system. *IFAC-PapersOnLine*, 50(1), 3142–3147.

Rosolia, U. and Borrelli, F. (2017b). Learning model predictive control for iterative tasks. A data-driven control framework. *IEEE Transactions on Automatic Control*, 63(7), 1883–1896.

Rosolia, U. and Borrelli, F. (2019). Learning how to autonomously race a car: A predictive control approach. *IEEE Transactions on Control Systems Technology*, 28(6), 2713–2719.

Sabatino, F. (2015). Quadrotor control: Modeling, nonlinear control design, and simulation.

Torrente, G., Kaufmann, E., Föhn, P., and Scaramuzza, D. (2021). Data-driven MPC for quadrotors. *IEEE Robotics and Automation Letters*, 6(2), 3769–3776.