2022 IEEE International Conference on Robotics and Automation (ICRA)
May 23-27, 2022. Philadelphia, PA, USA

Learning Model Predictive Control for Quadrotors

Guanrui Li*, Alex Tunchez*, and Giuseppe Loianno

***Abstract*— Aerial robots can enhance their safe and agile navigation in complex and cluttered environments by efficiently exploiting the information collected during a given task. In this paper, we address the learning model predictive control problem for quadrotors. We design a learning receding–horizon nonlinear control strategy directly formulated on the system nonlinear manifold configuration space $SO(3) \times \mathbb{R}^3$. The proposed approach exploits past successful task iterations to improve the system performance over time while respecting system dynamics and actuator constraints. We further relax its computational complexity making it compatible with real-time quadrotor control requirements. We show the effectiveness of the proposed approach in learning a minimum time control task, respecting dynamics, actuators, and environment constraints. Several experiments in simulation and real-world setup validate the proposed approach.**

I. Introduction

Micro Aerial Vehicles (MAVs) such as quadrotors have become very popular platforms to help humans solve a wide range of time-sensitive problems in constrained outdoor and indoor environments including logistics, search and rescue for post-disaster response, and more recently during COVID-19 pandemic reconnaissance and monitoring. These time-sensitive tasks would require robots to make fast decisions and agile maneuvers in uncertain, cluttered, and dynamic environments by intelligently exploiting the environment information to improve their performances over time. In this work, we investigate a Learning Model Predictive Control (LMPC) for quadrotors exploiting past successful task iterations to improve its task performance over time while respecting system dynamics and actuator constraints.

Several works have investigated the use of MPC for quadrotor control with perception and actuator constraints [1]–[4]. Other works [5], [6] improve MPC performance by refining the system dynamics in a data-driven fashion. Conversely, approaches such as [7] approximate the system dynamics directly using neural networks. However, these methods are quite computationally expensive since they rely on a sampling approach that is generally performed in parallel using Graphics Processing Units (GPUs). Iterative learning control techniques [8] have been successfully combined with MPC in a Batch Model Predictive Control (BMPC) approach to control chemical processes [9] and refine their performances over multiple task iterations. In [10], [11] a learning MPC approach is proposed and applied to ground vehicles. In this approach, the vehicle collects the states and their corresponding costs, across multiple successful iterations of the same task. The vehicle learns from the collected data to explore new ways to decrease cost in the same task as long as it maintains the ability to reach a state that has already been demonstrated to be safe during previous iterations. The approach does not require a reference trajectory as in previously mentioned works; thus, it is especially versatile and useful during tasks where the desired trajectory is not known or difficult to compute due to the system complexity or parameter uncertainty. This is the typical case of drone racing competitions [12]–[15] that have recently inspired researchers to design autonomy algorithms with the goal to grant vehicles the ability to execute agile maneuvers with superior performances compared to human controlled vehicles. Therefore, inspired by [10], [11], we propose an LMPC approach for quadrotors. Common multi-rotor platforms including quadrotors evolve on the nonlinear manifold configuration space $SO(3) \times \mathbb{R}^3$ making the LMPC problem substantially different and more complex for these types of systems compared to ground vehicles. We address the challenges of building a safety set that includes members of the rotation group $SO(3)$. Also, we consider an appropriate numerical integration approach for the group elements to ensure that the forward integration results adhere to the $SO(3)$ structure once employed in the discrete MPC formulation. In addition, we carefully add several design considerations to make the approach compatible with real-time control requirements of small aerial robots and show its feasibility in a learning minimum time control problem.

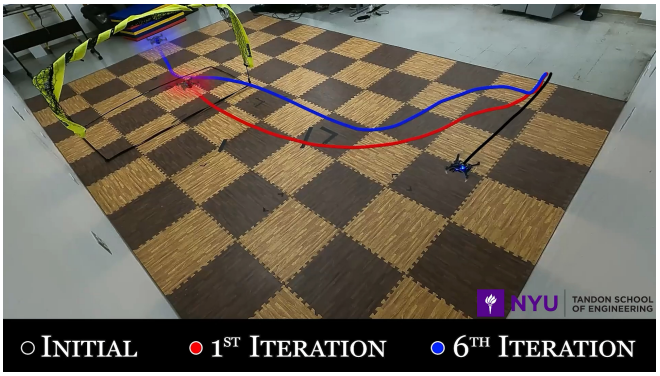

Fig. 1: The LMPC iterates over the same task while learning minimum time trajectories and improving its performances.

The contribution of this paper is threefold. First, we propose an LMPC for quadrotors. We tackle the challenges related to transitioning this class of MPC to multi-rotor

*These authors contributed equally.

The authors are with the New York University, Tandon School of Engineering, Brooklyn, NY 11201, USA. email: {lguanrui, atunchez, loiannog}@nyu.edu.

This work was supported by the NSF CAREER Award 2145277, the NSF CPS Grant CNS-2121391, the Technology Innovation Institute, Qualcomm Research, Nokia, and NYU Wireless.

aerial systems. The data collected across multiple iterations incorporates elements of the nonlinear manifold $SO(3)$. Furthermore, this aspect also makes the numerical MPC integration substantially more complex since at each integration step it is necessary to guarantee that the obtained elements still lie in the rotation group. Second, we relax the computational complexity of the proposed approach making it suitable for real-time quadrotor control applications. Finally, we show a specific instance of the proposed method in a learning minimum time control task. Several simulation and experimental results show the ability of our approach to successfully exploit collected data to improve system execution time performances.

The paper is organized as follows. In Section II, we review the MPC and LMPC approaches. In Section III, we specifically address the challenges and show how to design LMPC for quadrotors, whereas, in Section IV, we consider a particular instance of this approach employed to solve a minimum time control task. Section V presents our experimental results and Section VI concludes the paper.

II. Overview

In this section, we provide a brief review of the MPC formulation and how it transforms into LMPC.

A. Model Predictive Control

MPC is a predictive control method that finds a sequence of system inputs, $\mathbf{U} = \{\mathbf{u}_0, \mathbf{u}_1, \cdots, \mathbf{u}_{N-1}\}$ with $\mathbf{u}_k \in \mathbb{R}^{n_u}$, within a fixed time horizon of N steps. It optimizes a given objective function – with a running and terminal cost $h(\cdot,\cdot)$ and $Q(\cdot)$ respectively – while accounting for constraints and system dynamics

$$\begin{aligned} \min_{\mathbf{u}_0,\mathbf{u}_1,\cdots,\mathbf{u}_{N-1}} \; & \sum_{k=0}^{N-1} h\left(\mathbf{x}_k, \mathbf{u}_k\right) + Q\left(\mathbf{x}_N\right), \\ \text{s.t. } & \mathbf{x}_{k+1} = \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right), \; \forall k = 0, \cdots, N-1 \\ & \mathbf{x}_0 = \mathbf{x}(t_0), \\ & \mathbf{g}\left(\mathbf{x}_k, \mathbf{u}_k\right) \leq 0, \end{aligned} \tag{1}$$

where $\mathbf{x}_{k+1} = \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right)$ represents the system dynamics and $\mathbf{x}_k \in \mathbb{R}^{n_x}$ is the system state. The optimization occurs with initial condition $\mathbf{x}_0$ while respecting system dynamics $\mathbf{f}(\mathbf{x}, \mathbf{u})$ and additional state and input constraints $\mathbf{g}\left(\mathbf{x}, \mathbf{u}\right)$.

B. Learning Model Predictive Control

In this section, we review the LMPC, an iterative learning model predictive control method that can improve task performance over many trials [10]. The task is repeated at each iteration starting from the same initial state, $\mathbf{x}_s$. The task is considered complete when the system reaches a global terminal state, $\mathbf{x}_F$, without violating constraints. During each task iteration, the system records the state and cost and uses the recorded data to ensure control convergence to a locally optimal solution. We denote state and cost recorded during the j^{th} iteration as the j^{th} closed-loop trajectory. In the following, we first introduce preliminary concepts for LMPC such as safety set and its corresponding terminal cost function. Then we will introduce the LMPC formulation.

1) Safety Set: The states and inputs for a trajectory at each j^{th} iteration is defined as

$$\mathbf{x}^j = \left[\mathbf{x}_0^j, \mathbf{x}_1^j, ..., \mathbf{x}_{T^j}^j\right], \; \mathbf{u}^j = \left[\mathbf{u}_0^j, \mathbf{u}_1^j, ..., \mathbf{u}_{T^j-1}^j\right], \tag{2}$$

where T^j is the time stamp when the task is completed at the j^{th} iteration. The recorded data points from each iteration generate a sampled safety set

$$SS^j = \left\{ \bigcup_{i \in M^j} \bigcup_{t=0}^{T^i} x_t^i \right\}, \tag{3}$$

where M^j is a set of indexes that represents the iterations that successfully completed the task. In short, SS^j is a set for the j^{th} iteration which contains the recorded state trajectories from previous successfully completed tasks.

2) Cost Function: The cost-to-go for the state $\mathbf{x}_t^j$ at time t of the j^{th} iteration in the safety set is defined as

$$J^j_{t \to T^j}(\mathbf{x}_t^j) = \sum_{k=t}^{T^j} h\left(\mathbf{x}_k^j, \mathbf{u}_k^j\right). \tag{4}$$

The cost-to-go for any state can be determined with eq. (4) which is needed to satisfy LMPC formulation as discussed in Section II-B.3. Furthermore, the optimal cost of a given state is obtained from the minimum cost-to-go across all previous successful iterations

$$Q^j(\mathbf{x}) = \left\{ \begin{matrix} \min\limits_{(i,t) \in F^j(\mathbf{x})} J^i_{t \to T^i}(\mathbf{x}), & \text{if } \mathbf{x} \in SS^j \\ +\infty, & \text{if } \mathbf{x} \notin SS^j \end{matrix} \right\}, \tag{5}$$

$$F^j(\mathbf{x}) = \left\{(i,t) | i \in [0,j], t \geq 0 \text{ with } \mathbf{x} = \mathbf{x}_t^i, \text{ for } \mathbf{x}_t^i \in SS^j\right\}.$$

Eq. (5) assigns every state the respective minimum cost-to-go across all iterations.

3) LMPC Formulation: The LMPC builds upon the general MPC formulation as described in Section II-A by appending a safety set constraint for the terminal state and assigning eq. (5) as the terminal cost

$$\begin{aligned} J^{LMPC,j}_{t \to t+N}\left(\mathbf{x}_t^j\right) = \min_{\mathbf{u}_{t|t}, \ldots, \mathbf{u}_{t+N-1|t}} & \sum_{k=t}^{t+N-1} h\left(\mathbf{x}_{k|t}, \mathbf{u}_{k|t}\right) \\ & + \; Q^{j-1}\left(\mathbf{x}_{t+N|t}\right), \end{aligned} \tag{6a}$$

$$\text{s.t. } \mathbf{x}_{k+1|t} = \mathbf{f}\left(\mathbf{x}_{k|t}, \mathbf{u}_{k|t}\right), \; \forall k \in [t, \cdots, t+N-1], \tag{6b}$$

$$\mathbf{g}\left(\mathbf{x}_{k|t}, \mathbf{u}_{k|t}\right) \leq 0, \tag{6c}$$

$$\mathbf{x}_{t+N|t} \in SS^{j-1}, \tag{6d}$$

$$\mathbf{x}_{t|t} = \mathbf{x}_t^j. \tag{6e}$$

The optimal control problem in eq. (6) computes a solution for the j^{th} iteration at a given time stamp t of the task, over a finite horizon N. The eqs. (6b), (6c), (6e) define the system dynamics, state, and input constraints, and initial condition of the system, respectively. The safety set constraint in eq. (6d) forces the terminal state $\mathbf{x}_{t+N|t}$ to visit a discrete safety set state in SS^{j-1}. In principle, this guarantees a closed-loop solution that pushes the system to a final state $\mathbf{x}_F$.

III. LMPC for Quadrotors

In this section, we address the challenges related to designing LMPC for quadrotors. These systems evolve on

complex nonlinear manifold configuration spaces $SO(3) \times \mathbb{R}^3$ thus making the formulation and solution of the LMPC substantially more complex than ground vehicles studied in [10]. This induces the need for appropriate numerical integration techniques of the system dynamics preserving the structure of the configuration space over time. Furthermore, the controller should run at an acceptable rate for real-time control of the quadrotor. The discrete property of the safety set, SS^j, makes the LMPC a Nonlinear Mixed Integer Programming (NMIP) problem due to eq. (6d). This formulation is computationally expensive and incompatible with real-time applications. Hence in the following, we propose a convex approximation approach in Section III-C.1 in order to transfer the safety set constraints to a linear constraint and speed up the optimization. In addition, the size of the safety set SS^j is $n_x \times (\sum_{i=0}^{j} \tilde{f} T^i)$ and depends on the size of a state n_x, sample frequency $\tilde{f}$, and time T^i of completion for successful i^{th} iteration. Based on the formula, we see that the size of the safety set can grow very quickly across iterations. Therefore, it is beneficial to create a sparser safety set for computational reasons. We propose to select a subset of SS^j as a local safety set to reduce the computational burden. Finally, we consider the limitation of the hardware platform. We incorporate in our formulation actuator constraints to guarantee the task feasibility as well as safety of the quadrotor platform.

A. System Dynamics

The system overview is presented in Fig. 2. The relevant variables used in our paper are stated in Table I. We use two different coordinate frames with axes $\{\mathbf{e}_x^{\mathcal{I}}, \mathbf{e}_y^{\mathcal{I}}, \mathbf{e}_z^{\mathcal{I}}\}$ and $\{\mathbf{e}_x^{\mathcal{B}}, \mathbf{e}_y^{\mathcal{B}}, \mathbf{e}_z^{\mathcal{B}}\}$, to denote the inertial frame and the quadrotor's body frame, respectively. The relevant variables for our model are defined in Table I. We consider $\mathbf{x} = [\mathbf{x}_Q^\top, \dot{\mathbf{x}}_Q^\top, \mathbf{q}^\top]^\top$ and $\mathbf{u} = [f, \mathbf{\Omega}^\top]^\top$ the quadrotor state and input vectors, respectively. The state vector includes the quaternion $\mathbf{q} = [q_w, q_x, q_y, q_z]^\top$ to describe the quadrotor's orientation in the $\mathcal{I}$ frame. We employ this orientation parameterization instead of a rotation matrix so that we can reduce our state space size and consequently speed up our LMPC as mentioned in Section II.

The system dynamics are

$$\frac{d}{dt}\mathbf{x}_Q = \dot{\mathbf{x}}_Q, \quad \frac{d}{dt}\dot{\mathbf{x}}_Q = \frac{1}{m_Q}\left(f\boldsymbol{R}\mathbf{e}_z^{\mathcal{I}} - g\mathbf{e}_z^{\mathcal{I}}\right), \tag{7}$$

$$\frac{d}{dt}\mathbf{q} = \frac{1}{2}\Lambda\left(\mathbf{\Omega}\right)\cdot\mathbf{q}, \tag{8}$$

where $\Lambda\left(\mathbf{\Omega}\right)$ is a skew-symmetric matrix of the quadrotor angular velocity [16] with $\mathbf{\Omega} = [\Omega_x, \Omega_y, \Omega_z]^\top$.

TABLE I: Notation table.

$\mathcal{I}, \mathcal{C}, \mathcal{B}$	inertial, camera, robot frame
$m_Q \in \mathbb{R}$	mass of robot
$\mathbf{x}_Q, \dot{\mathbf{x}}_Q, \ddot{\mathbf{x}}_Q \in \mathbb{R}^3$	position, velocity, acceleration of robot in $\mathcal{I}$
$\boldsymbol{R} \in SO(3)$	rotation matrix of robot with respect to $\mathcal{I}$
$\mathbf{\Omega} \in \mathbb{R}^3$	angular velocity of robot in $\mathcal{B}$
$f \in \mathbb{R}$	total thrust
$g \in \mathbb{R}$	gravity constant

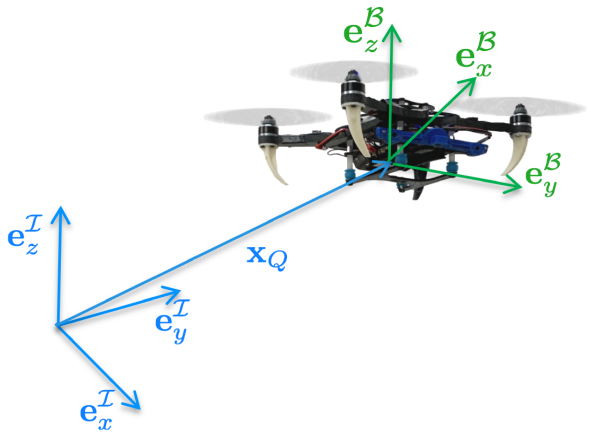

Fig. 2: System convention with $\mathcal{I}$ and $\mathcal{B}$ denoting inertial and body frames respectively.

The total thrust is the lift generated by each of the rotors $f = f_1 + f_2 + f_3 + f_4$ along $\mathbf{e}_z^{\mathcal{B}}$. Subsequently, it is necessary to apply a rotation induced by $\mathbf{q}$ onto $\mathbf{e}_z^{\mathcal{I}}$ by using

$$\mathbf{q} \odot \mathbf{e}_z^{\mathcal{I}} = \boldsymbol{R}\mathbf{e}_z^{\mathcal{I}}. \tag{9}$$

B. Discretization and Numerical Integration

The eqs. (7)-(8) represent the continuous system dynamics of a quadrotor which can be written as $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$. In order to incorporate them in the discrete time LMPC formulation, we apply 4^{th} order Runge-Kutta method to numerically integrate $\dot{\mathbf{x}}$ over the sampling time dt given the state $\mathbf{x}_k$ and input $\mathbf{u}_k$ as

$$\mathbf{x}_{k+1} = \mathbf{f}_{RK4}\left(\mathbf{x}_k, \mathbf{u}_k, dt\right). \tag{10}$$

The key challenge here is the numerical integration and discretization techniques over $SO(3)$ for eq. (8). Techniques like the Runge-Kutta method do not necessarily adhere to the $SO(3)$ structure [17]. As a result, the quaternion's norm can become non-unit because of round off error from numerical integration at each time step, which will also induce non-orthogonal column vectors in the corresponding rotation matrix. Extra efforts can be made to circumvent numerical drift by applying a unit constraint. It will add extra constraints to the optimization problem causing additional computational complexity. Another way is to normalize the quaternion after each integration. However, this cannot be done over a horizon in MPC because the result of any intermediate step within the Runge-Kutta calculation is not guaranteed that the unit quaternion represents a $SO(3)$ element. To bypass these strategies, we employ a non-unit quaternion in eq. (8) and its mapping to a rotation matrix is

$$\boldsymbol{R} = \frac{\mathbf{Q}}{\|\mathbf{q}\|_2^2}, \tag{11}$$

where $\mathbf{Q}$ is defined in [17]. The only remaining requirement is to avoid the quaternion getting too close to the zero norm in eq. (11). It can be shown that eq. (8), which maps angular velocity to the derivative of a unit quaternion, is a particular solution of the minimum-norm solution for the derivative of a non-unit quaternion [17]. Therefore, to avoid the quaternion magnitude getting close to 0, we can add to eq. (8) any linear combination of vectors in the nullspace of the weighting

matrix of the aforementioned minimum norm problem which turns out to be $\mathbf{q}$ and tune the coefficients. Alternatively, a one time re-scaling step can also be applied.

C. LMPC Relaxation

1) Convex Safety Set: Inspsired by [11], we approximate the safety set SS^j by taking its convex hull CS^j as

$$CS^j = Conv(SS^j) = \left[\mathbf{x}^0, \cdots, \mathbf{x}^j\right] \boldsymbol{\lambda}^\top, \tag{12}$$

where

$$\begin{aligned} &\boldsymbol{\lambda} = \left[\lambda_0^0, \lambda_1^0, \cdots, \lambda_{T^0}^0, ..., \lambda_0^j, \lambda_1^j, \cdots, \lambda_{T^j}^j\right], \\ &\boldsymbol{\lambda} \geq 0, \ \ \|\boldsymbol{\lambda}\|_1 = 1. \end{aligned} \tag{13}$$

Eq. (12) represents the barycentric approximation of SS^j. In this way, any state in the convex hull can be written as a convex combination of points in the safety set. Each element in $\boldsymbol{\lambda}$ corresponds to a positive weighted scalar value for each state in the convex hull. Similarly, an approximation of the terminal cost function can be derived. Therefore, we obtain

$$\tilde{Q}^j(\mathbf{x}) = Conv(Q^j(\mathbf{x})) = \min_{\boldsymbol{\lambda}\geq 0} \left[J^j_{0\to T^0}(\mathbf{x}^0_0), J^j_{1\to T^0}(\mathbf{x}^0_1), \right. \\ \left. ..., J^j_{0\to T^j}(\mathbf{x}^j_0), ...\right] \boldsymbol{\lambda}^\top. \tag{14}$$

Using the convex hull of the safety set transforms the NMIP into a Quadratic Program with linear constraints defined by eq. (12) and eq. (13).

It should be noted that for sake of simplicity, in eq. (12) we used the same notation to represent the weighted average operations for both the state variables in the Euclidean space and the variables in the rotation group $SO(3)$. However, in fact, for $SO(3)$ elements, it cannot be achieved in the same way as for elements in the Euclidean space. The notion of Karcher mean [18] choosing $L2$-norm as metric among two rotation elements guarantees to find the convex set. However, the procedure does not have a closed-form solution and does not guarantee the existence of a unique mean [18], thus making it difficult to incorporate it in the MPC. However in our task, the rotations are distributed around the identity element of $SO(3)$ group and we can simply compute the convex safety set $CS^j_{\mathbf{x}_q}$ for the rotation part by lifting all the sample in the tangent space since they lie in the same parameter subgroup [19] as

$$CS^j_{\mathbf{x}_q} = \exp\left(\sum_{l=0}^{P} \lambda_l \log\left(\mathbf{x}_{q,l}\right)\right)$$

where the exp and log operation is defined in [16], $\mathbf{x}_q$ refers to the quaternion vector as a subset of its corresponding state vector, and l indicates a selected state in CS^j among P number of states.

2) Local Safety Set: In order to further reduce the computational load, we chose to reduce the size of the safety set by choosing a subset, SS^j_t, of SS^j at each time t

$$SS^j_t = \left\{ \bigcup_{i=p}^{j} \bigcup_{k\in K^i} \mathbf{x}^i_k \right\}. \tag{15}$$

$K^i = \{k^i_1, \cdots, k^i_n\}$ is a set of time stamps for the corresponding states in the i^{th} iteration, where $i \in \{p, ..., j\}$ and

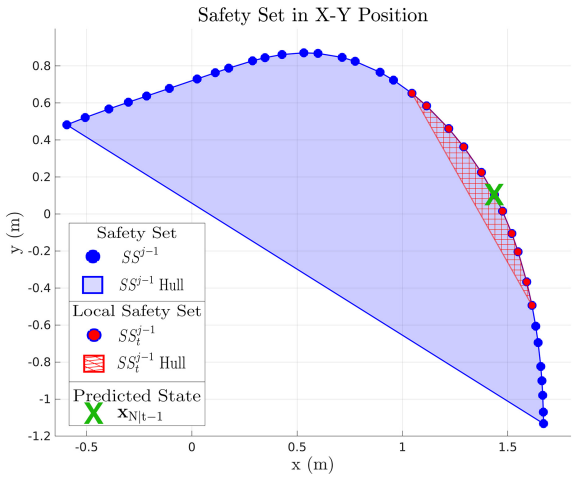

Fig. 3: Illustrative example on creating a local safety set in the X-Y position only. At time $t-1$ in j^{th} iteration, the LMPC forecasts states over the horizon, N, including the terminal predicted state $\mathbf{x}_{N|t-1}$ (green $\times$ in this figure). The local safety set, SS^{j-1}_t, at time t (red circles with blue outline) is made up of $\mathbf{x}_{N|t-1}$'s nearest neighbors in SS^{j-1}.

$p \in [0, j-1]$ is an integer that determines how many previous iterations to consider in the safety set. All the sets K^i have an equal size of n. The time index in K^i corresponds to the n-nearest neighbor states around the predicted terminal state from the predicted trajectory at $t-1$. Fig. (3) illustrates how the n-nearest neighbors are selected for the local safety set. Furthermore, using eq. (12), we approximate the subset by taking the convex hull

$$CS^j_t = Conv(SS^j_t) = \left[\mathbf{x}^p_{k^p_1}, ..., \mathbf{x}^p_{k^p_n} ..., \mathbf{x}^j_{k^j_1}, ...\mathbf{x}^j_{k^j_n}\right] \boldsymbol{\lambda}^\top_s = \mathbf{x}. \tag{16}$$

where

$$\|\boldsymbol{\lambda}_s\|_1 = 1, \ \ \boldsymbol{\lambda}_s = \left[\lambda^p_{k^p_1}, ..., \lambda^j_{k^j_n}\right]. \tag{17}$$

Eqs. (15)-(17) create a sparse convex hull with non-negative $\boldsymbol{\lambda}_s$ elements.

3) LMPC Problem Formulation: Finally, the relaxed LMPC for quadrotor system can be fully defined as following

$$\tilde{J}^{LMPC,j}_{t\to t+N}(\mathbf{x}^j_t) = \min_{\boldsymbol{\lambda}_s, \mathbf{u}_{t|t}, \cdots, \mathbf{u}_{t+N-1|t}} \sum_{k=t}^{t+N-1} h(\mathbf{x}_{k|t}, \mathbf{u}_{k|t}) \\ + \ \tilde{Q}^{j-1}_t(\mathbf{x}_{t+N|t}), \tag{18a}$$

$$\text{s.t. } \mathbf{x}_{k+1|t} = \mathbf{f}_{RK4}(\mathbf{x}_{k|t}, \mathbf{u}_{k|t}), \ \forall k \in [t, t+N-1], \tag{18b}$$

$$\mathbf{g}(\mathbf{x}_{k|t}, \mathbf{u}_{k|t}) \leq 0, \ \ \mathbf{x}_{t|t} = \mathbf{x}^j_t \tag{18c}$$

$$\mathbf{x}_{t+N|t} \in CS^{j-1}_t, \ \ \|\boldsymbol{\lambda}_s\|_1 = 1, \ \ \boldsymbol{\lambda}_s \geq 0. \tag{18d}$$

where

$$\tilde{Q}^{j-1}_t(\mathbf{x}_{t+N|t}) = \min_{\boldsymbol{\lambda}_s \geq 0} [J^p_{k^p_1\to T^p}(\mathbf{x}^p_{k^p_1}), \ \ J^p_{k^p_2\to T^p}(\mathbf{x}^p_{k^p_2}), ... \\ J^j_{k^j_1\to T^j}(\mathbf{x}^j_{k^j_1}), ...]\boldsymbol{\lambda}^\top_s. \tag{19}$$

Our system considers hardware limitations by constraining the control inputs

$$f_{min} \leq f \leq f_{max}, \tag{20}$$

$$\boldsymbol{\Omega}_{min} \leq \boldsymbol{\Omega} \leq \boldsymbol{\Omega}_{max}, \tag{21}$$

where f_{min} and f_{max} are the maximum and minimum thrusts, whereas $\boldsymbol{\Omega}_{min}$ and $\boldsymbol{\Omega}_{max}$ are the maximum and minimum angular velocity, respectively.

IV. Learning Optimal Time Control

In this section, we show a particular instance of the LMPC approach to learning optimal time trajectories. This approach can be leveraged for autonomous racing tasks by naturally discovering the minimum lap time through reference-free iterations. Each task begins with a quadrotor at $\mathbf{x}_s$ which maneuvers to a predefined goal $\mathbf{x}_G$. A track is created by setting intermediate waypoints and corridors to the goal. An initial safety set, SS^0, is created by flying steadily and suboptimally through the track. The LMPC formulation in eq. (6) is relaxed as discussed in Section III and the following cost function

$$\sum_{k=t}^{t+N-1} \left[\mathbb{1}\left(\mathbf{x}_{k|t}\right) + \mathbf{u}_{k|t}^{\top} \mathbf{R}_u \mathbf{u}_{k|t} \right] + \tilde{Q}_t^{j-1}\left(\mathbf{x}_{t+N|t}\right), \tag{22}$$

where

$$\mathbb{1}\left(\mathbf{x}\right) = \begin{Bmatrix} 1, & \text{if } \mathbf{x}_Q \neq \mathbf{x}_G \\ 0, & \text{Else} \end{Bmatrix}. \tag{23}$$

The running cost is defined by two main components. First, it includes a binary cost which depends on whether the quadrotor has reached the goal position. This cost represents the minimum time control. Second, there is a penalty applied on the inputs to minimize the control effort with $\mathbf{R}_u$ as constant diagonal matrix to tune the penalty on the control. It is important to note that lower values in $\mathbf{R}_u$ favor a minimum-time solution, but this may yield aggressive control and with possible non-smooth control inputs. Therefore, there is a trade-off between these two objectives and tuning the weights is useful to achieving desired performances. While the binary cost can be implemented in simulation, it proves time consuming in real-time application. Thus, eq. (22) is approximated with a sigmoid function

$$h(\mathbf{x}_{k|t}, \mathbf{u}_{k|t}) = \frac{\left\|\mathbf{x}_{Q,k|t} - \mathbf{x}_G\right\|_2^2}{\sqrt{\left\|\mathbf{x}_{Q,k|t} - \mathbf{x}_G\right\|_2^4 + 1}} + \mathbf{u}_{k|t}^{\top} \mathbf{R}_u \mathbf{u}_{k|t}. \tag{24}$$

Furthermore, a track must be defined for the racing case. This can be done with corridors that are defined between two waypoints. A linear constraint is added to the LMPC formulation to guarantee the quadrotor stays within the track. The position of two distinct consecutive waypoints is defined as $\mathbf{r}_w, \mathbf{r}_{w+1}$, respectively. The direction from the first waypoint to the quadrotor is defined as $\mathbf{r}_i = \mathbf{x}_Q - \mathbf{r}_w$. The normalized direction between each waypoint is $\hat{\mathbf{r}}_c = \frac{\mathbf{r}_{w+1} - \mathbf{r}_w}{\|\mathbf{r}_{w+1} - \mathbf{r}_w\|_2}$. $\mathbf{r}_n$ is the difference between $\mathbf{r}_i$ and the projection of $\mathbf{r}_i$ onto $\hat{\mathbf{r}}_c$ as

$$\mathbf{r}_n = \left[\mathbf{I} - \hat{\mathbf{r}}_c \hat{\mathbf{r}}_c^{\top}\right] \left(\mathbf{x}_Q - \mathbf{r}_w\right). \tag{25}$$

Physically, $\mathbf{r}_n$ represents the quadrotor's distance to the center axis of the corridor in space. Therefore, a single corridor constraint applies bounds on $\mathbf{r}_n$

$$-\boldsymbol{\delta} \leq \mathbf{r}_n \leq \boldsymbol{\delta}. \tag{26}$$

Therefore, eq. (26) is a linear constraint which matches the form of eq. (18c) as

$$\mathbf{b}_{min} \leq \mathbf{A}\mathbf{x}_Q \leq \mathbf{b}_{max}. \tag{27}$$

V. Experiments

We propose several simulations and experiments with a quadrotor to validate the proposed LMPC during a learning minimum time trajectory task.

A. Task Overview

We consider an optimal time control problem as specified in Section IV. The task is successfully accomplished if the quadrotor can pass through a given gate while staying within a given track and stop after passing it. Fig. 1 illustrates an example of the proposed task. This is obtained considering an L-shape track, which is constructed using the approach in Section. IV both in simulations and real-world experiments. We first provide the quadrotor with a feasible and slow reference trajectory for the quadrotor MPC controller to track, like the one shown in the black color in the Figs. 4 and 6. During the trajectory tracking, the quadrotor records its state history to build the initial safety set. Subsequently, we run the LMPC controller and repeat the same process by sending the quadrotor to the same start position at the end of each task iteration. The only information provided to LMPC is the safety set and the corresponding cost-to-go over the recorded safety set. The LMPC will then start to find the optimal time trajectories that minimize the travel time while respecting the system dynamics, actuator constraints, and track constraints. The attached multimedia material provides several additional experiments as well. We solve the proposed LMPC problem in eq. (18) with cost as eq. (22) via Sequential Quadratic Programming (SQP) using ACADOS [20], [21] as a solver. For the LMPC controller parameters, we choose the prediction time horizon the discretized time step dt as 0.1 s, the horizon length $N = 10$ and corridor width $\boldsymbol{\delta} = 0.8$ m.

B. Environments

1) Simulation: For simulation, we use a custom simulator available in the lab developed in ROS[1] with full system dynamics simulated using 4^{th} order Runge-Kutta method.

2) Real World: The real-world experiments are performed in an indoor testbed with a flying space of $10 \times 6 \times 4$ m^3 at the ARPL lab at the New York University. We leverage a Vicon[2] motion capture system at 100 Hz for control purposes. The quadrotor platform setup is similar to our previous work [22]. The control and estimation frameworks are developed in ROS. The proposed LMPC method can run on-board at 100 Hz on a common laptop.

[1] www.ros.org
[2] www.vicon.com

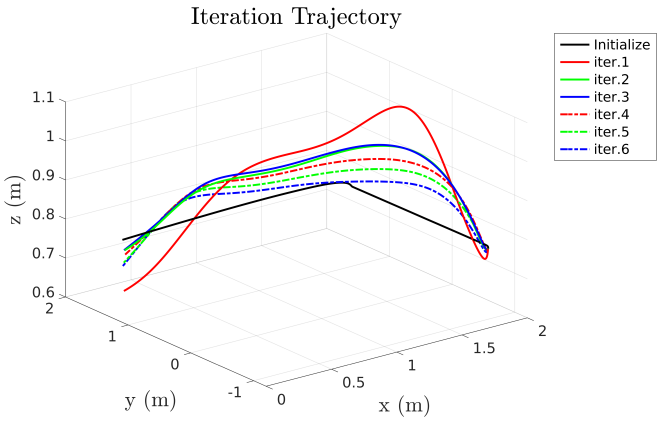

Fig. 4: Iteration trajectories for the L-shaped track in simulation.

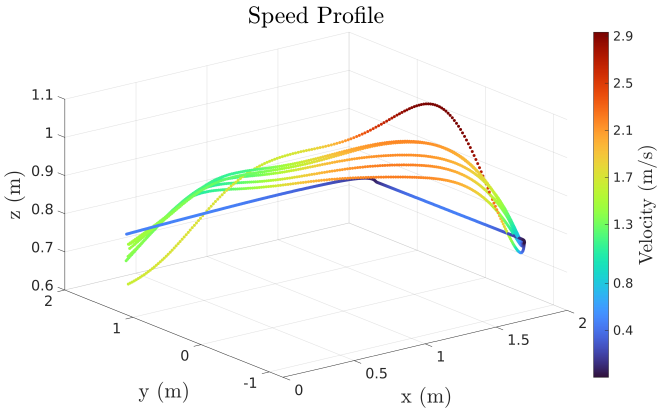

Fig. 5: Trajectory and velocity profiles across several iterations for the L-shaped track in simulation.

TABLE II: The total travel time (s) of different iterations in simulation and real-world experiments

Iter	Init	1	2	3	4	5	6
Real	10.0002	3.5291	3.6993	3.9895	3.6296	3.7494	3.7292
Sim	14.1291	2.0900	2.4000	2.4300	2.4600	2.5299	2.5800

C. Results

We show the simulation and real-world test results of a quadrotor traveling through an L-shape track with the LMPC. The results of the simulation are shown in Figs. 4 and 5 whereas the real-world experiments in Figs. 6 and 7. The travel time for each iteration is reported in Table II. As we can observe from the plots, both in the simulation and real-world experiments, the quadrotor can explore the track and find a locally optimal time trajectory which ends up a much faster trajectory than the initial one. In Table II, we observe that as the iterations continue, the LMPC converges similar and stable travel time along the track. This proves that the LMPC can utilize the recorded states and costs in the past iteration and explore new faster trajectories for the quadrotor to accomplish the task. However, we notice that the final lap time varies around a given value after convergence. We believe this behavior is mostly due to a mismatch between real and modeled dynamics including our approximation to first-order attitude dynamics in eq. (8).

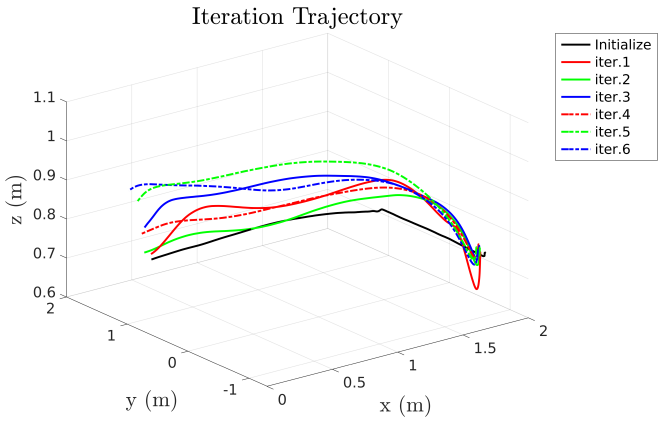

Fig. 6: Iteration trajectories for the L-shaped track in real-world experiments.

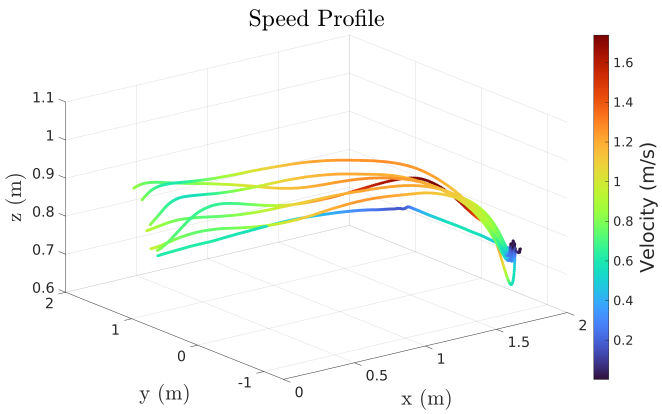

Fig. 7: Trajectory and velocity profiles across several iterations for the L-shaped track in real-world experiment.

VI. Conclusion

In this paper, we presented an LMPC for quadrotors. We addressed the challenges associated with the system evolution on a nonlinear manifold configuration space which requires careful considerations in the LMPC problem formulation as well as in its forward numerical time integration. We showed how to reduce its computational complexity to make it compatible with the stringent requirements for real-time control of quadrotors as well as its usefulness in a learning minimum time trajectory problem.

Future works will investigate the trade-offs between the incorporation of the dynamics till rotor speeds in the proposed approach to improve the system's performance, agility, and the corresponding increase in computational complexity which may affect the system real-time performances. We will leverage Bayesian machine learning techniques to refine the system dynamics incorporating unmodelled dynamical effects across multiple runs thus allowing us to further push the system performances and agility limits. We will also consider employing this method for drone racing tasks extending the proposed experiments to a full racing track. Finally, we will investigate the use of different cost functions and how the availability of reference trajectories can potentially be exploited to improve the task performances.

REFERENCES

[1] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "PAMPC: Perception-aware model predictive control for quadrotors," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–8.

[2] D. Bicego, J. Mazzetto, M. Farina, R. Carli, and A. Franchi, "Nonlinear model predictive control with enhanced actuator model for multi-rotor aerial vehicles with generic designs," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 100, p. 1213–1247, 2020.

[3] M. Jacquet and A. Franchi, "Motor and perception constrained nmpc for torque-controlled generic aerial vehicles," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 518–525, 2021.

[4] G. Li*, A. Tunchez*, and G. Loianno, "PCMPC: Perception-Constrained Model Predictive Control for Quadrotors with Suspended Loads using a Single Camera and IMU," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 2012–2018.

[5] P. Bouffard, A. Aswani, and C. Tomlin, "Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 279–284.

[6] G. Torrente, E. Kaufmann, P. Föhn, and D. Scaramuzza, "Data-driven mpc for quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3769–3776, 2021.

[7] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic mpc for model-based reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1714–1721.

[8] D. Bristow, M. Tharayil, and A. Alleyne, "A survey of iterative learning control," *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 96–114, 2006.

[9] J. H. Lee and K. S. Lee, "Iterative learning control applied to batch processes: An overview," *Control Engineering Practice*, vol. 15, no. 10, pp. 1306–1318, 2007, special Issue - International Symposium on Advanced Control of Chemical Processes (ADCHEM).

[10] U. Rosolia and F. Borrelli, "Learning model predictive control for iterative tasks. a data-driven control framework," *IEEE Transactions on Automatic Control*, vol. 63, no. 7, pp. 1883–1896, 2018.

[11] ——, "Learning model predictive control for iterative tasks: A computationally efficient approach for linear system," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3142–3147, 2017, 20th IFAC World Congress.

[12] H. Moon, J. Martinez-Carranza, T. Cieslewski, M. Faessler, D. Falanga, A. Simovic, D. Scaramuzza, S. Li, M. Ozo, C. De Wagter, G. Croon, S. Hwang, S. Jung, H. Shim, H. Kim, M. Park, T.-C. Au, and s.-j. Kim, "Challenges and implemented technologies used in autonomous drone racing," *Intelligent Service Robotics*, vol. 12, 04 2019.

[13] W. Guerra, E. Tal, V. Murali, G. Ryou, and S. Karaman, "Flightgoggles: Photorealistic sensor simulation for perception-driven robotics using photogrammetry and virtual reality," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 6941–6948.

[14] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "AlphaPilot: Autonomous Drone Racing," in *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020.

[15] K. Mohta, M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Zhu, J. Delmerico, K. Karydis, N. Atanasov, G. Loianno, D. Scaramuzza, K. Daniilidis, C. J. Taylor, and V. Kumar, "Fast, autonomous flight in gps-denied and cluttered environments," *Journal of Field Robotics*, vol. 35, no. 1, pp. 101–120, 2018.

[16] J. Solà, "Quaternion kinematics for the error-state kalman filter," *ArXiv*, vol. abs/1711.02508, 2017.

[17] C. Rucker, "Integrating rotations using nonunit quaternions," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2979–2986, 2018.

[18] R. Hartley, J. Trumpf, Y. Dai, and H. Li, "Rotation averaging," *International Journal of Computer Vision*, vol. 103, pp. 267–305, 2012.

[19] M. Moakher, "Means and averaging in the group of rotations," *SIAM Journal on Matrix Analysis and Applications*, vol. 24, 04 2002.

[20] R. Verschueren, G. Frison, D. Kouzoupis, N. van Duijkeren, A. Zanelli, R. Quirynen, and M. Diehl, "Towards a modular software package for embedded optimization," in *Proceedings of the IFAC Conference on Nonlinear Model Predictive Control (NMPC)*, 2018.

[21] R. Verschueren, G. Frison, D. Kouzoupis, N. van Duijkeren, A. Zanelli, B. Novoselnik, J. Frey, T. Albin, R. Quirynen, and M. Diehl, "acados—a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, 2021.

[22] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 404–411, April 2017.