NMPC Strategy for a Quadrotor UAV in a 3D Unknown Environment

Iuro B. P. Nascimento[1], Antonio Ferramosca[2], Luciano C. A. Pimenta[1,3] and Guilherme V. Raffo[1,3]

***Abstract*— This work presents a Nonlinear Model Predictive Control strategy for a quadrotor UAV with obstacle avoidance capability in a 3D unknown environment with static obstacles. The system aims to reach the target in minimum time while avoiding obstacles and also to take into account the energy of states and inputs. Sensor information is processed to detect the obstacles and obtain the inequality constraints of an obstacle-free zone. Numerical results are presented to attest the performance of the system.**

I. Introduction

Unmanned Aerial Vehicles (UAVs) have received enormous attention from academy and industry. Many applications are carried out with UAVs, such as search and rescue, cartography, inspections of power lines, aerial photography, and others. These applications demand many different schemes of control, path planning, and obstacle avoidance in order to complete their tasks.

In some applications, the environment is known a priori. Consequently, the trajectory planning can be computed offline, requiring a control strategy capable of tracking it. In the literature, many algorithms can be found that perform the planning task, for instance, sampling algorithms as probabilistic roadmaps (PRM) [1], rapidly-exploring random trees (RRT) [2], among others. Sampling algorithms may have a high computational cost due to their probabilistic nature. However, they only need to be executed once and offline.

In other applications, the environment is unknown, which means that the location of possible obstacles and their shape are not known a priori. In this case, path planning would be performed at every change of the environment, what could lead to computational costly algorithms. Some approaches recompute only part of the path when the environment changes, as D^*[3]. They are more computationally efficient if compared to recomputing the entire path, but they are designed for incremental changes in the environment.

An alternative approach is the Model Predictive Control (MPC), which combines an optimal path planning and optimal control design into an unified optimization problem. MPC delivers an optimal control law by minimizing specified performance index constrained by the dynamical model of the system, obstacles and the limits of the system inputs.

There are many algorithms in the literature using the MPC framework for obstacle avoidance. In [4], a linear MPC is used to track a reference trajectory of a UAV, where depending on the risk of collision, a new optimal control problem (OCP) is solved with an extra cost to penalize the proximity with obstacles In [5], a robust MPC for UAVs performs obstacle avoidance by generating linear constraints in the XY plane by geometric calculations. In [6], zonotopic approximations of hyperplane arrangements [7] of an obstacle-free area in 2D environments are used as constraints of a linear MPC.

In [8], a Nonlinear MPC (NMPC) strategy is used for an Autonomous Ground Vehicle (AGV) navigating in 2D environments with obstacles represented as polygons, based on computational geometry algorithms, to obtain linear constraints. Since the OCP is nonlinear and non-convex, a local optimum solution is inevitable and dependent on the initial guess. [8] also explores more possible paths in order to search for better local minima solutions of the OCP.

This work proposes an NMPC strategy for a UAV in a 3D unknown environment. Differently from [8], we model obstacles as polyhedra and use boolean polyhedral operations to obtain an obstacle-free zone in a 3D scenario. A convex decomposition of polyhedra is performed to obtain tractable constraints and explore possible paths that can lead to local minima with a smaller cost. While [8] uses a cost function that is tailored to a specific AGV, we use a general quadratic cost that could be used with other systems.

This work is organized as follows. Section II shows the quadrotor UAV model. In Section III, we describe the control strategy. Section IV presents the OCP formulation, describing the generation of obstacles constraints and the OCP formulation. Section V presents numerical simulation results, and Section VI concludes the work.

II. Quadrotor UAV Mathematical Model

The dynamic equations of the quadrotor UAV are obtained from [9] and are given by

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{f}(\boldsymbol{q}) = \boldsymbol{B}_u(\boldsymbol{q})\boldsymbol{u}, \qquad (1)$$

where $\boldsymbol{q} = \begin{bmatrix}\boldsymbol{\xi}^T & \boldsymbol{\eta}^T\end{bmatrix}^T$, $\dot{\boldsymbol{q}} = \begin{bmatrix}\dot{\boldsymbol{\xi}}^T & \dot{\boldsymbol{\eta}}^T\end{bmatrix}^T$, with $\boldsymbol{\xi} = \begin{bmatrix}x & y & z\end{bmatrix}^T$ being the position of the quadrotor body frame

*This work was in part supported by the project INCT (National Institute of Science and Technology) under the grant CNPq (Brazilian National Research Council) 465755/2014-3, FAPESP, Brazil 2014/50851-0. This work was also partially supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Brazil (Finance Code 88887.136349/2017-00), CNPq, Brazil (grant numbers 426392/2016-7, 313568/2017-0, 311063/2017-9), FAPEMIG, Brazil (grant number APQ-03090-17), and the Agencia Nacional de Promoción Científica y Tecnologica (ANPCyT), Argentina (grant number PICT-2016-0283).

[1]I. B. P. Nascimento, L. C. A. Pimenta and G. V. Raffo are in the Graduate Program in Electrical Engineering, Federal University of Minas Gerais (UFMG), CEP 31270-901, Belo Horizonte, Minas Gerais, Brasil. (iuro@ufmg.br, lucpim@cpdee.ufmg.br, raffo@ufmg.br)

[2]A. Ferramosca is with CONICET - UTN Facultad Regional Reconquista, Santa Fe 3560, Argentina. (ferramosca@santafe-conicet.gov.ar)

[3] L. C. A. Pimenta and G. V. Raffo are also with the Department of Electronic Engineering, UFMG, and are members of the INCT for Cooperative Autonomous Systems Applied to Security and Environment.

origin w.r.t. the inertial frame $\mathscr{I}$, $\boldsymbol{\eta} = \begin{bmatrix}\phi & \theta & \psi\end{bmatrix}^T$ its orientation, where ϕ, θ, and ψ are the angles of the rotation of the body frame $\mathscr{B}$ w.r.t. $\mathscr{I}$ expressed in $\mathscr{B}$. The vectors $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ are the generalized coordinates and their time derivatives, respectively. Thrust forces generated by the propellers compose the input vector $\boldsymbol{u} = \begin{bmatrix}f_1 & f_2 & f_3 & f_4\end{bmatrix}^T$, and $\boldsymbol{M}(\boldsymbol{q})$ is the inertia matrix, derived from the kinetic energy of the quadrotor. $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is the Coriolis and centrifugal forces matrix, which is derived from the inertia matrix using the Christoffel symbols of the first kind, and the $\boldsymbol{g}(\boldsymbol{q})$ vector is composed of terms generated by the gravity force [10]. Assuming the state vector $\boldsymbol{x} = \begin{bmatrix}\boldsymbol{q}^T & \dot{\boldsymbol{q}}^T\end{bmatrix}^T$, the state equation is

$$\dot{\boldsymbol{x}} = \begin{bmatrix}\dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{q}}\end{bmatrix} = \begin{bmatrix}\dot{\boldsymbol{q}} \\ \boldsymbol{M}(\boldsymbol{q})^{-1}(-\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{f}(\boldsymbol{q}))\end{bmatrix}. \tag{2}$$

The equations are derived assuming the frame $\mathscr{B}$ with the origin at the quadrotor center of rotation. The input coupling matrix $\boldsymbol{B}_u(\boldsymbol{q})$ transforms the input signals represented in $\mathscr{B}$ to $\mathscr{I}$, where $\boldsymbol{B}_u(\boldsymbol{q}) = \boldsymbol{R_{rw}}\boldsymbol{B_a}$, $\boldsymbol{R_{rw}} = \begin{bmatrix}\boldsymbol{R}_{\mathscr{B}}^T & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W}_{\boldsymbol{\eta}}^T\end{bmatrix}^T$, and

$$\boldsymbol{B}_a(\boldsymbol{q}) = \begin{bmatrix} -\sin(\alpha_T) & 0 & \sin(\alpha_T) & 0 \\ 0 & -\sin(\alpha_T) & 0 & \sin(\alpha_T) \\ \cos(\alpha_T) & \cos(\alpha_T) & \cos(\alpha_T) & \cos(\alpha_T) \\ 0 & l\cos(\alpha_T) & 0 & -l\cos(\alpha_T) \\ -l\cos(\alpha_T) & 0 & l\cos(\alpha_T) & 0 \\ \frac{k_T}{b}\cos(\alpha_T) & -\frac{k_T}{b}\cos(\alpha_T) & \frac{k_T}{b}\cos(\alpha_T) & -\frac{k_T}{b}\cos(\alpha_T) \end{bmatrix}.$$

The rotation matrix $\boldsymbol{R}_{\mathscr{B}}$ represents the rotation from $\mathscr{B}$ to $\mathscr{I}$. The angle α_T is the inclination of the rotors towards to the origin of $\mathscr{B}$, the constant k_τ is a drag constant, the constant b is the thrust constant, assuming $f_i = b\ \Omega_i^2$ with Ω_i being the angular velocities of the propellers. The matrix $\boldsymbol{W}_\eta$ is the Euler matrix, and the constant l is the distance from a rotor to the origin of $\mathscr{B}$.

III. Control Strategy

A. Problem

Consider a UAV in an unknown environment. The problem consists of moving the UAV to a target pose as fast as possible. The environment may contain static obstacles with location, size, and shape not known a priori.

B. Control Scheme

The control strategy consists of two main tasks:

- generate the obstacle constraints;
- formulate and solve optimal control problems (OCPs).

The system schematic is illustrated in Fig. 1.

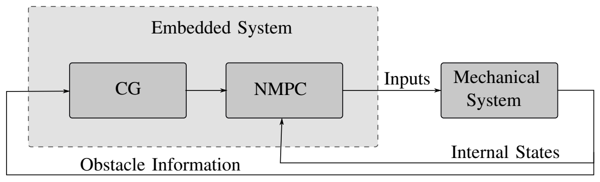

Fig. 1. System Schematic.

The constraint generation (CG) task obtains the constraints to maintain the UAV's position in a safe region, which is called the obstacle-free region. The NMPC task formulates an optimal control problem (OCP) with constraints. The

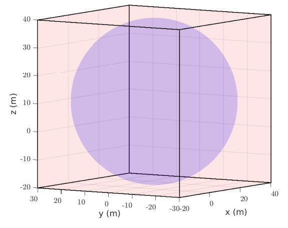

Fig. 2. The range sphere of the sensor in blue and the superscribed polyhedron in pink.

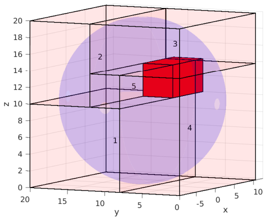

Fig. 3. The main polyhedron in pink, the five numbered subregions, the range sphere in blue, and the obstacle in red.

OCP solution will provide an optimal trajectory and inputs satisfying constraints, such as the safe region constraints, the mathematical model, state and input limits, and others. By satisfying these constraints, the generated optimal trajectory is feasible since it is free of a collision. These tasks will be described in detail in the next sections.

C. Obstacle Constraint Generation

The CG task generates constraints to maintain the optimal trajectory in a safe region. The safe region is obtained by processing sensor data. A camera or laser sensor is used to obtain the relative positions of obstacles.

In order to obtain the constraints, both the sensor field of view and the obstacles are modeled as polyhedra. The sensor has a range R, which is the maximum distance from the UAV such that the sensor is able to detect obstacles. A sphere of radius R represents the sensor field of view. This sphere is approximated as a cuboid polyhedron, as shown in Fig. 3. The entire sensor field of view is contained inside this main polyhedron. This is a conservative approximation; however, the trajectory terminal position is constrained to be near the sphere surface.

The CG task performs three main processing subtasks on the sensor data. The first task adds a safety margin to the obstacles. In the sequence, these polyhedra are removed from the main polyhedron, creating a safe region. Finally, the safe region is decomposed into convex subregions. Each subregion will have its constraints derived from the half-space representation of the polyhedron. These tasks are described as follows:

1) Safety Margin: The obstacles are expanded to create a safety margin. The safety margin is added to allow the UAV safely pass near the inflated obstacles. Increasing the obstacles by the UAV size allows us to consider it as a point in the optimal control problem. Thus, the safety margin D is the sum of the largest robot dimension, the maximum error of measurement, and the maximum error of approximation of obstacles as polyhedra. A Minkowski sum of a polyhedron with a sphere of radius D gives the offset polyhedron with its boundaries inflated by D. Since this operation would not return a polyhedron, a Minkowski sum of the polyhedron with a cube with side $2D$ is used as an approximation.

2) Safe region: The safe region is obtained by removing the obstacles polyhedra from the main polyhedron. Fig. 3

shows an example of a safe region with one obstacle inside the main polyhedron.

The creation of the safe region is accomplished by using a regularized set difference operation on the polyhedra since a set difference of two polyhedra would have as a result an open set or a polyhedron without some of its boundaries. The regularized set difference will contain all the polyhedron boundaries. Besides, this operator is defined as the closure of the interior of the normal set difference between 2 polyhedra.

In order to obtain the set difference, we represent the polyhedra as Nef Polyhedra and use the data structures and algorithms from [11]. A Nef polyhedron is defined by [12] as

Definition 1. *[12] A Nef Polyhedron in $\mathbb{R}^n$ is a point set $P \subset \mathbb{R}^n$ generated from a finite number of half-spaces using set complement and intersection operations.*

The algorithm stores cells for facets, vertices, edges, and volumes with information about the intersection between them. Labels may be used to mark various features of cells. Binary labels with on/off values indicate if the cell is contained in a polyhedron or not. This information allows the intersection and complement operations. Other operations on sets can be reduced to these operations.

3) Subregion Decomposition: The safe region is usually non-convex, and a function to describe this region is hard to be obtained. The safe region may present other problems:

- The formulated OCP may have many possible valid local minimum solutions that will not be explored by the solver.
- Most modern nonlinear programming (NLP) solvers work more efficiently with all constraints, along with the objective function being twice differentiable, and a safe region usually has edges and corners, which are not twice differentiable. These NLPs can work with numerical derivatives, but the higher numerical errors often make the solver have a slow convergence or not converge.

To deal with these problems, we decompose the safe region into convex polyhedra subregions. These polyhedra will form sets of constraints, one set per polyhedron or subregion. For each subregion, the hyperplane equations of the half-space representation are used as the constraints. These constraints are linear and twice differentiable as in equation $\boldsymbol{A}_i\boldsymbol{\xi}_j - \boldsymbol{d}_i \leq \boldsymbol{0}$, where $\boldsymbol{\xi}_j = \begin{bmatrix} x_j & y_j & z_j \end{bmatrix}^T$ is jth position in the discretized trajectory, $\boldsymbol{A}_i$ is an $M \times 3$ matrix of constraints of the subregion i, $\boldsymbol{d}_i$ and $\boldsymbol{0}$ are $M \times 1$ column vectors, M is the number of hyperplanes in the polyhedron, and $i = 1, 2 \ldots N$ with N subregions.

In order to decompose the safe region into convex polyhedra, the algorithm proposed in [13] is used. In a polyhedron, each edge is only shared by two facets. Non-convex polyhedra will have edges where the inner angles with respect to the polyhedron are reflex, or greater than 180^o, called notches. In each notch g_i with facets T_i and S_i, we cut the polyhedron in two with a facet R_i that shares the notch g_i as one of their line segments and forms angles with the facets T_i and R_i that are not reflex. After each cut, we form two new polyhedra by splitting the polyhedron in R_i. After all notches have been cut, all remaining polyhedra will be convex since all reflex angles have been divided. Fig. 3 shows the safe region decomposed into subregions.

In order to find free paths for the robot, all possible openings are considered as a possible solution. Openings are areas where the sensor detects no obstacle. They are obstacle-free and are possible exits from the safe region. Subregions 1, 3, and 4 of Fig. 4 have openings. For each opening, a path through the subregions is obtained from the current pose to the subregion opening. In order to decide which subregion should be traversed to reach the opening, a graph is created where each subregion is a node, and each edge (a graph's edge) represents the adjacency between subregions. Adjacent subregions share at least part of a plane. A path can be obtained using the Dijkstra shortest path algorithm [14].

This procedure can be better visualized with the output of a planar sensor, but the same can be executed with a 3D sensor output. As an example, Fig. 4 shows a safe region decomposed into convex 2D polyhedra.

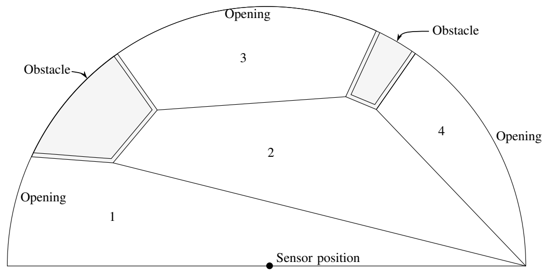

Fig. 4. 2D sensor output decomposed into four numbered subregions.

The start subregion is number 1, which also has an opening. Subregions 3 and 4 also have openings. Fig. 5 shows the adjacency graph.

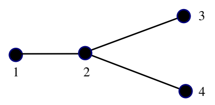

Fig. 5. Adjacency graph of subregions.

In this case, using a weight of 1 for each edge in the graph, the best possible paths to each opening will be: 1; $1 \rightarrow 2 \rightarrow 4$; and $1 \rightarrow 2 \rightarrow 3$. In order to get more direct paths, we estimate the distance to be traversed in each path to use as weights for the graph edges. For each subregion, one reference point is chosen. For each subregion, a convex combination of its vertices is used as a reference point. Using the initial point and the reference points as nodes, the weights for each edge are the distances between each node point.

D. Nonlinear MPC Strategy

Using the subregion information, a multi-phase OCP is formulated. A multi-phase OCP consists of dividing the trajectory into segments, where the dynamical equations and constraints may be different in each segment. In some problems, systems change along the trajectory, as the stages of spaceship launch. In other problems a different set of constraints may be needed at each phase. This formulation allows the use of different state constraints in each segment. For each path, the trajectory is divided into N segments, one per subregion that the trajectory has to go through to an opening. Each segment is subject to its respective subregion constraints. Additional constraints are used to ensure continuity of the solution at the transition of one phase to the next.

In order to determine the constraints for each phase, a graph search determines the subregion path from the initial subregion to each opening. For each opening, its constraints are used to formulate one OCP. Since each path can be processed independently, each OCP is formulated and solved in parallel. The solution with the smaller cost is chosen.

IV. Optimal Control Problem Formulation

In order to use the MPC framework, a multi-phase constrained OCP is formulated as follows

$$\begin{aligned}
&\underset{\boldsymbol{x},\boldsymbol{u},T_1,\ldots,T_N}{\text{minimize}} && J && (3)\\
&\underset{\forall i=1,\ldots,N}{\text{subject to}} && \dot{\boldsymbol{x}}^{(i)}(t) = \mathscr{V}\left(\boldsymbol{x}^{(i)}(t), \boldsymbol{u}^{(i)}(t)\right) && (4)\\
& && \boldsymbol{x}^{(i)}_{min}(t) \leq \boldsymbol{x}^{(i)}(t) \leq \boldsymbol{x}^{(i)}_{max}(t) && (5)\\
& && \boldsymbol{u}^{(i)}_{min}(t) \leq \boldsymbol{u}^{(i)}(t) \leq \boldsymbol{u}^{(i)}_{max}(t) && (6)\\
& && \mathscr{G}\left(\boldsymbol{x}^{(i)}(t)\right) \leq 0 && (7)\\
& && \boldsymbol{x}^{(i)}(T_{i-1}) = \boldsymbol{x}^{(i-1)}(T_{i-1}) && (8)\\
& && t \in [T_{i-1}, T_i],\ T_{i-1} < T_i && (9)\\
&\text{subject to} && \mathscr{F}\left[\boldsymbol{x}^{(N)}(T_N), \boldsymbol{x}^{(0)}(T_0)\right] \leq 0 && (10)\\
& && T_0 = 0,\ T_N = T_p && (11)
\end{aligned}$$

where (3) is the cost functional, with one cost per each of the N phases, and a terminal cost. Equations (4) to (9) are the per phase constraint equations. Equations (10) and (11) are constraints to enforce on all phases. (4) are the nonlinear dynamical equations of the UAV, with one set of equations (2) per phase. Equations (5) and (6) are bounds to the states and inputs. Equation (7) is the position constraint to avoid obstacles. To maintain continuity between phases, equations (8) are imposed to ensure a continuous transition. Equation (9) is necessary to enforce the phase instants of times to be monotonically increasing. Equation (10) imposes the initial state to be the current state of the system and imposes a terminal region on the final state. Finally, equation (11) sets the initial time to 0 and the final time of the last phase T_N to be equal to the time horizon T_p. The following sections describe in more detail the cost function and constraints.

A. Cost Function

The multi-phase cost function of equation (3) is given by

$$\begin{aligned}
J =& w_t T_p + \left\|\boldsymbol{x}^{(N)}(T_p) - \boldsymbol{x}_{goal}\right\|^2_{\boldsymbol{P}} \qquad (12)\\
&+ \sum_{i=1}^{N}\left[\int_{T_{i-1}}^{T_i}\left(\left\|\boldsymbol{x}^{(i)}(t) - \boldsymbol{x}_{goal}\right\|^2_{\boldsymbol{Q}} + \left\|\boldsymbol{u}^{(i)}(t) - \boldsymbol{u}_{\boldsymbol{eq}}\right\|^2_{\boldsymbol{R}}\right) dt\right].
\end{aligned}$$

There is one cost per phase, where it is penalized the energy of states that are far from the final target states ($\boldsymbol{x}_{goal}$) and the energy of inputs that are far from the equilibrium inputs ($\boldsymbol{u}_{\boldsymbol{eq}}$). The cost of the energy of inputs has two purposes: to avoid or minimize the chance of multiple solutions and to avoid a solution with aggressive inputs. In order to obtain a shorter final time, a cost is used to penalize T_P with weight w_t.

The weighting matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ can be chosen according to the desired performance. The matrix $\boldsymbol{P}$ is obtained by solving the algebraic Ricatti equation $\boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} - \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P} + \boldsymbol{Q} = \boldsymbol{0}$, where $\boldsymbol{A}$ and $\boldsymbol{B}$ are the linear state-space representation matrices of the linearized system of equations (2).

B. Path Constraints

Equation (7) represents the path constraints, which are the subregion constraints. Each subregion constraint is given by $\tilde{\boldsymbol{A}}_i\boldsymbol{x}_i - \boldsymbol{b}_i \leq \boldsymbol{0}$, where $\tilde{\boldsymbol{A}} = \begin{bmatrix}\boldsymbol{A}_i & \boldsymbol{0}_{M\times 9}\end{bmatrix}$.

C. State Continuity

In order to ensure that the solution will be continuous between phases, the dynamical equation (4) of phase $i-1$ requires that its final state be equal to the initial state of phase i, as stated in (8).

D. Terminal Constraints

In order to ensure that the terminal state is close to the border of the sensor field of view, the following constraints are added

$$R_{sensor} - \delta \leq \left\|\boldsymbol{\xi}^{(N)}(T_N) - \boldsymbol{\xi}^{(1)}(T_0)\right\| \leq R_{sensor}, \qquad (13)$$

where $\boldsymbol{\xi}^{(i)}(t) = \begin{bmatrix}x^{(i)}(t) & y^{(i)}(t) & z^{(i)}(t)\end{bmatrix}^T$, R_{sensor} is the sensor range, and δ is a small value to define the terminal region near to the end of the sensor field of view.

When the target position is within the sensor range, the following inequalities are used as terminal constraints.

$$\boldsymbol{\xi}_{goal} - \sigma \leq \boldsymbol{\xi}^{(N)}(T_P) \leq \boldsymbol{\xi}_{goal} + \sigma, \qquad (14)$$

where σ is a constant that indicates the distance from the goal position in the x, y, and z direction that will be in the terminal region. This terminal region constrains the last position $\boldsymbol{\xi}^{(N)}(T_P)$ into a cube with edge size of 2σ and with the goal on its center.

V. Results and Discussion

A. OCP Solution

At each sample time, the optimal control problem (OCP) (3)-(11) is solved, and the first control signal is applied to the system. In order to solve the OCP, it is used a direct method called HP-adaptive pseudospectral [15], which is used to transcribe the OCP to a nonlinear programming (NLP) problem and then solve with a primal-dual interior-point algorithm with a filter line search implemented in IPOPT [16].

The HP-adaptive pseudospectral is a direct collocation method that transcribes a continuous-time OCP into an NLP problem by approximating states and controls with a variable number of segments of a variable number of polynomial degrees. In this method, the dynamical equations of the system and virtual states are added as differential-algebraic constraints evaluated at the Legendre-Gauss-Radau (LGR) time points. In each segment, the states and controls are sampled at the LGR points using the Lagrange polynomial. The method adaptively changes the number of segments and degrees of the polynomials in order to reach a user-specified tolerance error in the middle of each two LGR points.

The resulting NLP is solved by the IPOPT, which handles large scale non-convex problems with constraints. The basic principle is to iteractively solve a barrier problem for a fixed value of Lagrange multipliers μ and decreasing the multiplier. Making μ converge to zero leads to the Karush-Kuhn-Tucker (KKT) conditions of the original problem be satisfied.

B. Results

Numerical results are obtained using Matlab R2018a and Simulink software. The dynamical model is simulated in Simulink, and the controller is implemented in Matlab using the Casadi [17] toolbox for the formulation of the OCP and for providing exact derivatives and Hessians by performing algorithm differentiation.

Two algorithms of the CGAL (Computational Geometry Algorithms Library) were used, the set difference [18] to remove all obstacles polyhedra from the main polyhedron, and [19] to decompose the safe regions into convex subregions.

The quadrotor model parameters were obtained from [9]: the mass m is 2.24 kg, the distance l from the center of mass to the rotors is 0.332 m, the thrust coefficient b is $9.5e^{-6}$ N s^2, the drag coefficient κ_τ is $1.7e^{-7}$ N m s^2, the gravity acceleration g is 9.81 m/s^2, the moments of inertia I_{xx}, I_{yy}, and I_{zz} are 0.0363 Kg m^2, 0.0363 Kg m^2, and 0.0615 Kg m^2, respectively. Also, the NMPC was synthesized with the following parameters: $w_t = 0.1$, $\delta = 0.5$, $D = 2l + 0.1 + 0.0173$, $\boldsymbol{Q} = \text{diag}([0.0006\ 0.0006\ 0.0006\ 0.0001\ 0.0001\ 0.0001\ 0.0025\ 0.0025\ 0.0025\ 0.0006\ 0.0006\ 0.0006])$, $\boldsymbol{R} = \text{diag}([0.0006944\ 0.0006944\ 0.0006944\ 0.0006944])$, and The sensor range R_{sensor} is $30m$. The matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ are initially chosen according to the Bryson's rule [20], in which each weight is chosen as the inverse of the square of the state or control being weighted. The matrix $\boldsymbol{P}$ is obtained by solving an algebraic Ricatti equation. The obstacle sensor error is 0.1 m, and the noise added to the state measurement was a white noise with maximum vector $\boldsymbol{n} = [0.01,\ 0.01,\ 0.01,\ 1.5\ 10^{-3},\ 1.5\ 10^{-3},\ 1.5\ 10^{-3},\ 5\ 10^{-3},\ 5\ 10^{-3},\ 5\ 10^{-3},\ 2\ 10^{-3},\ 2\ 10^{-3},\ 2\ 10^{-3}]$. The input limits are from 0 to 12 N, ϕ and θ are limited between $\pm\pi/2$, and ψ is limited to $\pm\pi$

Fig. 6 to 8 show the simulation results. Fig. 6 illustrates the trajectory of the UAV from the red circle (initial point at $\boldsymbol{\xi_0} = [10\ 0\ 10]$ m) to the green one (target point at $\boldsymbol{\xi_{goal}} = [100\ 0\ 20]$ m), from which it can be observed that the UAV successfully avoided the obstacles and reached the terminal region. The terminal region parameter σ is set equal to the l parameter of the quadrotor. The orientation of the UAV throughout the trajectory is shown in Fig. 8. In order to obtain the minimum time to reach the target in agreement with the cost functional (12), the quadrotor UAV performed aggressive maneuvers in some segments of the trajectory, requiring the roll angle to reach approximately its bound. This angular motion allowed the UAV to achieve high linear velocities due to the projection of the thrust forces towards the direction of the desired movement. Fig. 9 presents the applied control signals. Due to the minimum time requirement, the optimal control inputs reached their bounds at some periods in order to accomplish the task.

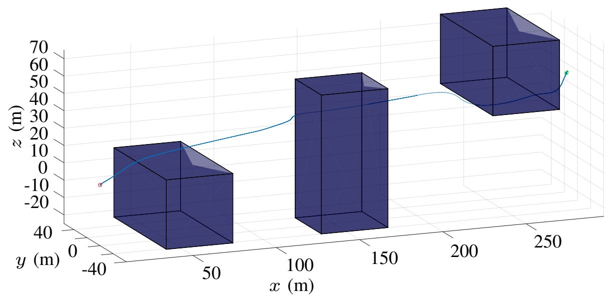

Fig. 6. Trajectory performed by the quadrotor UAV while avoiding the obstacle.

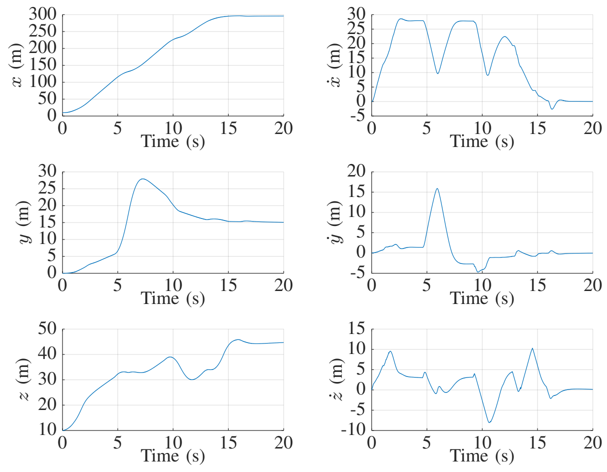

Fig. 7. Position and velocities.

VI. Conclusion and Future Works

This work proposed a multi-phase NMPC strategy for a quadrotor UAV in order to reach a target while avoiding obstacles in a 3D unknown environment with static obstacles. This task was required to be performed as fast as possible, taking into account the minimum energy of states and control inputs. Sensor information was used in the multi-phase capability in order to obtain feasible trajectories. An obstacle-free region was obtained from sensor information, and obstacles were modeled as polyhedra. The obstacle-free region was then divided into convex polyhedra to provide

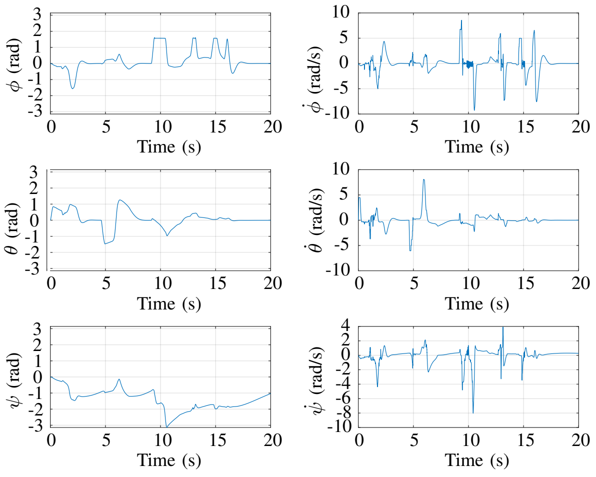

Fig. 8. Orientation and angular velocities.

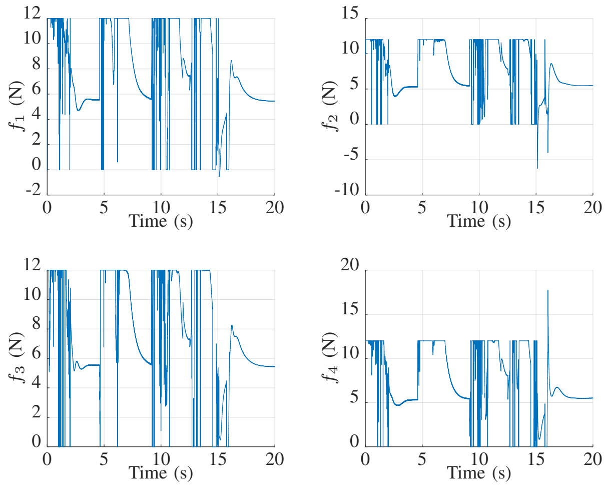

Fig. 9. Control signals - the applied thrusts.

constraints to the NMPC. Simulation results corroborated the proposed NMPC approach successfully.

The simulations are computationally expensive since a highly nonlinear and nonconvex optimization problem is being solved with the addition of linear path constraints. The partitions of the safe region are not optimum, and to the best of our knowledge, there is no algorithm to partition a nonconvex polyhedron into a minimum number of partitions. In cluttered environments, the number of partitions can be large, and the increased number of partitions may increase the computational cost significantly. Thin subregions may difficult the solver to find a feasible solution and decrease the convergence rate of the mesh-refinement algorithm [15]. The mesh-refinement algorithm can solve problems with regions with these characteristics, however the segment in the thin region has to start with a small number of collocated points.

Future works will deal with the implementation of a hardware-in-the-loop simulation using a 3D simulator such as Gazebo. In order to accomplish this goal, other approaches to the optimization, and even parallelization of the code will be considered.

References

[1] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.
[2] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Citeseer*, 1998.
[3] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 1994, pp. 3310–3317 vol.4.
[4] J. Marzat, S. Bertrand, A. Eudes, M. Sanfourche, and J. Moras, "Reactive mpc for autonomous mav navigation in indoor cluttered environments: Flight experiments," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 15 996–16 002, 2017.
[5] P. T. Jardine and S. N. Givigi, "A robust model-predictive guidance system for autonomous vehicles in cluttered environments," *IEEE Systems Journal*, vol. 13, no. 2, pp. 2034–2045, 2018.
[6] D. Ioan, S. Olaru, S.-I. Niculescu, I. Prodan, and F. Stoican, "Navigation in a multi-obstacle environment. from partition of the space to a zonotopic-based mpc," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 1772–1777.
[7] I. Prodan, F. Stoican, S. Olaru, and S.-I. Niculescu, "Enhancements on the hyperplanes arrangements in mixed-integer programming techniques," *Journal of Optimization Theory and Applications*, vol. 154, no. 2, pp. 549–572, 2012.
[8] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "Combined speed and steering control in high-speed autonomous ground vehicles for obstacle avoidance using model predictive control," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 10, pp. 8746–8763, 2017.
[9] G. V. Raffo, "Robust control strategies for a quadrotor helicopter: an underactuated mechanical system," Ph.D. dissertation, Universidad de Sevilla, 2011.
[10] M. W. Spong, S. Hutchinson, M. Vidyasagar, *et al.*, *Robot modeling and control*. Wiley New York, 2006, vol. 3.
[11] M. Granados, P. Hachenberger, S. Hert, L. Kettner, K. Mehlhorn, and M. Seel, "Boolean operations on 3d selective nef complexes: Data structure, algorithms, and implementation," in *European Symposium on Algorithms*. Springer, 2003, pp. 654–666.
[12] W. Nef, *Beiträge zur Theorie der Polyeder: mit Anwendungen in der Computergraphik*. Herbert Lang, 1978, vol. 1.
[13] B. Chazelle, "Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm," *SIAM Journal on Computing*, vol. 13, no. 3, pp. 488–507, 1984.
[14] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
[15] C. L. Darby, W. W. Hager, and A. V. Rao, "An hp-adaptive pseudospectral method for solving optimal control problems," *Optimal Control Applications and Methods*, vol. 32, no. 4, pp. 476–502, 2011.
[16] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, Mar 2006. [Online]. Available: https://doi.org/10.1007/s10107-004-0559-y
[17] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.
[18] P. Hachenberger and L. Kettner, "3D boolean operations on nef polyhedra," in *CGAL User and Reference Manual*. CGAL Editorial Board, 2018. [Online]. Available: https://doc.cgal.org/4.13/Manual/packages.html#PkgNef3Summary
[19] P. Hachenberger, "Convex decomposition of polyhedra," in *CGAL User and Reference Manual*. CGAL Editorial Board, 2018. [Online]. Available: https://doc.cgal.org/4.13/Manual/packages.html
[20] A. E. Bryson, *Applied Linear Optimal Control Hardback with CD-ROM: Examples and Algorithms*. Cambridge university press, 2002.