Multi-phase NMPC Strategy for Safe Navigation in Unknown Environments using Polynomial Zonotopes

Iuro B. P. Nascimento * **Luciano C. A. Pimenta** *,** **Guilherme V. Raffo** *,**

* *Graduate Program in Electrical Engineering, Universidade Federal de Minas Gerais, Brazil, (e-mail: iuro@ufmg.br).*
** *Department of Electronic Engineering, Universidade Federal de Minas Gerais, Brazil (e-mail: {lucpim, raffo}@ufmg.br).*

Abstract: This study presents a novel approach for guiding robots through cluttered and unknown environments. The approach utilizes multi-phase non-linear model predictive control (NMPC) and polynomial zonotopes (PZs) to describe collision-free areas (CFAs). Laser sensor data is processed to obtain the CFA, which is divided into convex subregions. These subregions are then transformed into PZs, which provide constraints for the optimal control problem (OCP) of the NMPC. Compared to conventional half-space representations, PZs require fewer constraints, resulting in a more efficient method for describing the same polytopes. The proposed approach employs a multi-phase method that divides the trajectory into segments and applies individual subregion convex constraints to each segment. This result is a reduction in the number of constraints per segment. Numerical experiments with a wheeled mobile robot are conducted to demonstrate the effectiveness of the proposed approach.

Keywords: NMPC; Obstacle Avoidance; MPC; Mobile Robot;

1. INTRODUÇÃO

Mobile Robots have gained significant attention due to their various applications, including assistive robots (Losey et al., 2020), search and rescue missions (Niroui et al., 2019), and inspection of power lines (Perez-Jimenez et al., 2020). In many applications, the environment is known, and a common navigation approach involves a two-layer solution. Trajectory planning can be obtained offline using sampling-based techniques such as probabilistic roadmaps (Kavraki et al., 1996) or the asymptotically optimal versions (Karaman and Frazzoli, 2011). A trajectory tracking control system is then required to execute the planned trajectory.

When dealing with unknown environments with unknown obstacle positions and shapes, using the two-layer solution demands the constant recalculation of the trajectory due to sensing new obstacles as the robot navigates. An alternative approach is the model predictive control (MPC), where at each time step an optimization problem combines both trajectory planning and control to obtain an optimal trajectory and control signals according to a performance index and constraints on states and inputs such as the system dynamics, obstacles, and physical limits of the system. The first control signal is then applied to the system and the optimization problem is computed again at the next time step.

There are multiple methods for incorporating a collision-free area (CFA) into the optimal control problem (OCP) for model predictive control (MPC) (Ioan et al., 2019; Liu et al., 2017; Nascimento et al., 2019). Common methods to embed the CFA have been either by imposing a constraint that describes the CFA (Ioan et al., 2019), or by adding a cost penalty in the cost functional (Santos et al., 2023), the so-called soft constraints. The representation complexity, number of constraints, and accuracy depend on how the safe regions are described. Usually, there is a trade-off between description accuracy and computational efficiency, since more accurate descriptions may involve either more constraints, non-convex constraints, or even the use of integer variables to select a set of constraints. While simpler descriptions may be adequate in sparsely populated environments, they may not be effective in cluttered environments where narrow passages can become obstructed, and the optimization may become unfeasible due to conservatism.

Many algorithms in the literature use the MPC framework for obstacle avoidance in mobile robots. Some works have described the safe region/obstacles only as a point and develop a control scheme upon this description. In Marzat et al. (2017), a micro aerial vehicle (MAV) trajectory has

⋆ This work was partially supported by the project INCT (National Institute of Science and Technology) under the grants CNPq (National Council for Scientific and Technological Development) 465755/2014-3 and FAPESP (São Paulo Research Foundation) 2014/50851-0, and by the Brazilian agencies Coordination for the Improvement of Higher Education Personnel - Brazil (CAPES) through the Academic Excellence Program (PROEX) and grant number 88887.136349/2017-00, CNPq (grant numbers 315258/2020-9, 315695/2020-0, and 407063/2021-8), and FAPEMIG under grant APQ-03090-17.

been tracked using an MPC strategy. First, the OCP is solved without obstacles. Next, the discretized trajectory is checked for collisions, and in case there is a collision, a new OCP is solved using potential fields as penalty cost in the cost functional. The obstacles are described in a grid map, and only the closest obstacle point is considered in the second MPC. Santos et al. (2021) have proposed a Nonlinear Model Predictive Control (NMPC) approach for tracking a position in cluttered environments. The authors incorporated a repulsive potential field as a cost function in the OCP by considering the point with the shortest distance to each obstacle detected by a 3D LIDAR. Additionally, they have included an attractive potential field as a cost to guide the trajectory towards the goal position and avoid obstacles.

Other works have described obstacles as circles or ellipses. Pereira et al. (2021) have employed an NMPC approach to enable aggressive maneuvering of a quadrotor UAV in cluttered environments. The authors modeled both the obstacles and the UAV as ellipses to formulate the NMPC strategy. To account for the UAV's attitude and enable navigation through narrow passages, the authors utilized an algebraic ellipsoidal set approach as soft constraints in the optimization problem. Drozdova et al. (2016) have used a laser scanner to detect obstacles and describe them as ellipses. The ellipse equations have been used in inequality constraints in the OCP. In Zhang et al. (2019), the obstacles have been approximated as circles, and two circular zones have been added around the object in order to obtain constraints on inputs to limit angular velocity and linear velocity.

Some works have described obstacles or the safe region as polytopes, as in Jardine and Givigi (2018), in which linear constraints have been generated in the XY plane to approximate the safe region around obstacles by a convex polytope. Ioan et al. (2019) have developed an NMPC scheme for navigation in multi-obstacle environments. To achieve this, they have utilized a *half-space representation* (H-rep) of the CFA known as hyperplane arrangements (HA). The authors have first approximated the obstacles to zonotopes, a centrally symmetric polytope representation, and then applied the HA method to obtain convex cells in H-rep. To solve the OCP, a mixed-integer (MI) formulation has been used. However, MI formulations may suffer from a higher computational burden due to their combinatorial nature. Liu et al. (2017) have navigated an autonomous ground vehicle (AGV) in 2D unknown environments using a non-linear MPC. Obstacles have been described as polygons, which are processed to obtain linear convexified constraints using H-rep. Nascimento et al. (2019) have utilized a 3D LIDAR to capture obstacle information for 3D quadrotor Unmanned Aerial Vehicle (UAV) navigation. We have approximated the CFA into polyhedra, divided the non-convex CFA into convex subregions and used the subregions' half-space representation in an NMPC multi-phase formulation.

This work proposes the use of *polynomial zonotope* (PZ) (Kochdumper and Althoff, 2020) to describe safe regions for an NMPC. PZ is a set representation that uses fewer constraints than H-rep, leading to better computational efficiency than H-rep works such as (Liu et al., 2017; Ioan et al., 2019), while maintaining the representation equivalent to a polytope. Besides, it avoids problems such as unfeasibility due to excess of conservativeness and obstruction of passages that occurs in works such as (Ioan et al., 2019; Drozdova et al., 2016; Zhang et al., 2019). In a similar manner to the work of Liu et al. (2017), this work divides the safe region into convex subregions and explores different possible exits from cluttered parts of an environment. Furthermore, while some works resort to mixed-integer formulations (Ioan et al., 2019) that have combinatorial nature and are computationally expensive, this works uses a multi-phase formulation of the OCP in order to obtain a solution that is more computationally efficient.

2. PRELIMINARIES

A *convex polytope* P with r vertices $v_i \in \mathbb{R}^n$ in vertex representation (V-rep) is described as

$$P \triangleq \left\{ \sum_{i=1}^{r} \lambda_i v_i \mid \lambda_i \geq 0,\ \sum_{i=1}^{r} \lambda_i = 1 \right\}. \tag{1}$$

The convex polytope P in *half-space representation* (H-rep) is written as

$$P \triangleq \{ \boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \}, \tag{2}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ with n being the set dimension, $\boldsymbol{A} \in \mathbb{R}^{r \times n}$ and $\boldsymbol{b} \in \mathbb{R}^r$ with r constraints.

A *constrained zonotope* (CZ) (Scott et al., 2016) is a representation (called CG-rep) of a convex set in the form

$$CZ \triangleq \left\{ \boldsymbol{c} + \sum_{i=1}^{p} \alpha_i \boldsymbol{G}(\cdot, i) \;\middle|\; \sum_{i=1}^{p} \alpha_i \boldsymbol{A}(\cdot, i) = \boldsymbol{b},\ \alpha_i \in [-1,\ 1] \right\}, \tag{3}$$

where the vector $\boldsymbol{c} \in \mathbb{R}^n$ is the center of the set with n being the set dimension, the matrix $\boldsymbol{G} \in \mathbb{R}^{n \times p}$ has p generators, the matrix $\boldsymbol{A} \in \mathbb{R}^{r \times n}$ and the column vector $\boldsymbol{b} \in \mathbb{R}^r$ form the equality constraints with r constraints. CZs can represent any convex polytope and has the advantage of forming linear constraints. However, equality constraints can substantially increase the number of constraints. CZs have explicit and efficient solutions of operations like linear map, Minkowski sum, and intersection.

A *sparse polynomial zonotope* (PZ) (Kochdumper and Althoff, 2020) is a compact efficient representation of a polynomial zonotope in the form

$$PZ \triangleq \left\{ \boldsymbol{c} + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{\boldsymbol{E}(k,i)} \right) \boldsymbol{G}(\cdot, i) + \sum_{j=1}^{q} \beta_j \boldsymbol{G}_I(\cdot, j) \;\middle|\; \alpha_k, \beta_j \in [-1,\ 1] \right\}, \tag{4}$$

called PG-rep. The matrix $\boldsymbol{G} \in \mathbb{R}^{n \times h}$ is the matrix of the h dependent generators, the i-th dependent column generator is $\boldsymbol{G}(., i)$, $\boldsymbol{G}_I \in \mathbb{R}^{n \times q}$ is the matrix of q independent generators, the j-th independent column generator is $\boldsymbol{G}_I(., j)$, the exponential matrix is $\boldsymbol{E} \in \mathbb{R}^{p \times h}$, the scalars α_k are the p dependent factors, and the scalars β_j are the q independent factors. PZs have exact efficient solutions to the linear map, Minkowski sum, exact addition, quadratic map, and convex hull. A PZ is written in compact form as $PZ \triangleq \langle \boldsymbol{c}, \boldsymbol{G}, \boldsymbol{G}_I, \boldsymbol{E} \rangle_{PZ}$. Polytopes in several representations such as V-rep, H-rep, and zonotopes can be converted

exactly into PZs. To convert a polytope in V-rep to PG-rep, the vertices can be converted to PZ using the vertex as the center of the PZ, and empty matrices for generators.

It is worth mentioning that PZs are polynomial representations that can depict not only convex polytopic sets but also non-convex and non-polytopic sets. Compared to polytopes represented in H-rep, PZs generated by PG-rep provide constraints that are proportional to the number of set dimensions, rather than the number of polytope facets. This makes PZs more scalable in cluttered environments.

3. NMPC STRATEGY FOR SAFE ROBOT NAVIGATION

A non-linear continuous-time system described by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{5}$$

where $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^n$ is the state vector, $\boldsymbol{u} \in \mathcal{U} \subset \mathbb{R}^m$ is the input vector, and $\boldsymbol{f} : R^n \times R^m \rightarrow R^n$ is a non-linear function that describes the motion of a robot navigating in an unknown environment. The objective is to design an NMPC strategy to safely navigate the system (5) through an unknown environment in order to reach a target position as fast as possible.

The control system proposed is composed of two main tasks, as it is shown in Figure 1: (i) *Constraint Generation* (CG); and (ii) *NMPC*.

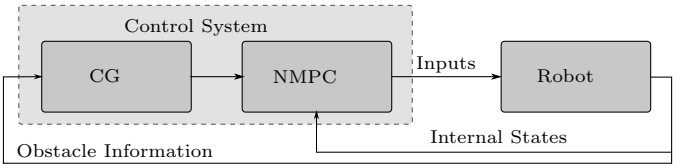

Figure 1. Control Structure.

3.1 Constraint Generation Task

The CG task aims to process the sensor outputs to obtain proper constraints to the NMPC OCP. The laser sensor (LIDAR, Light Detection and Ranging) scans its surroundings at regular angle intervals to obtain the distance from obstacles. At each angle α_i, the sensor sends a beam to obtain a distance d_i that can be either a distance to an obstacle or the maximum range $R_{\max}$ of the sensor. The scanning area reached by the sensor to $R_{\max}$ is called the sensor field of vision (FOV). In the 2D scanning, the LIDAR scans n samples, forming points in polar coordinates, which are converted to cartesian coordinates, yielding a polytope P of n points $\boldsymbol{\xi}_i = [x_i, y_i]$. The polytope P is the safe region (SR) or CFA.

Figure 2 (a) shows an example of LIDAR output with two in gray and the possible openings that the system can pass through from the current robot position $\boldsymbol{\xi}_0$ to reach the target position $\boldsymbol{\xi}_{\text{goal}}$.

The CG is composed of four subtasks:

1) Safe Region Creation: The algorithm begins by converting the LIDAR output into a polytope, as outlined earlier in this section. The subsequent step involves eliminating redundant points to form the SR polytope. Redundant

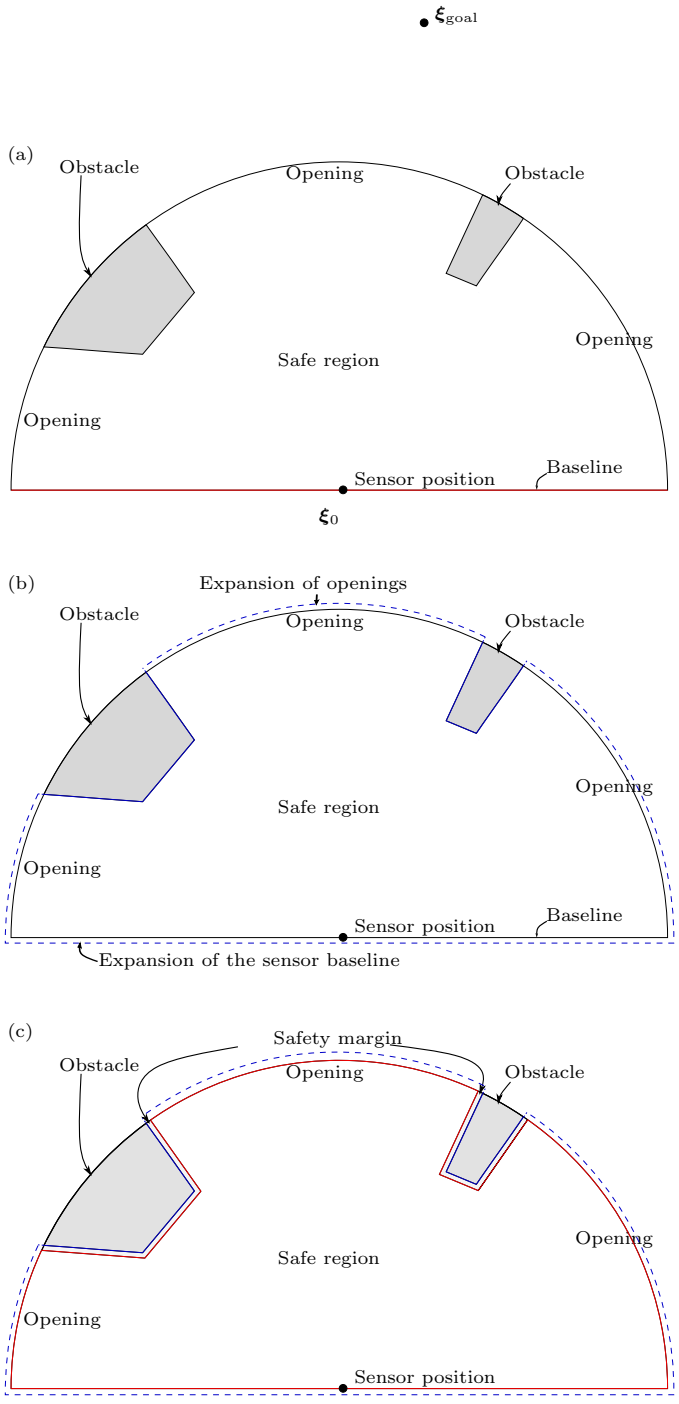

Figure 2. Obstacle polytope expansion. (a) Original polytope with the sensor baseline in red; (b) the expansion of the openings and the sensor baseline (dashed blue line); (c) The blue polytope is deflated into the red polytope.

points are removed using the line simplification algorithm developed by Ramer (1972). The algorithm takes as input the list of points comprising the 2D polytope (either a polygon or a polygonal chain) and starts by creating a line segment between the first and last points, p_1 and p_2, respectively. Next, it selects the test point, p_t, that is the farthest point from the line segment $\overline{p_1 p_2}$. If the distance between p_t and $\overline{p_1 p_2}$ is less than a specified threshold distance, ϵ, then p_t and all points on the line

segment are removed. If p_t is located at a distance greater than ϵ from the line segment, it is retained and two new line segments, $\overline{p_1 p_t}$ and $\overline{p_t p_2}$, are created. The algorithm iteratively repeats these steps for each new line segment until all line segments contain only two points.

2) Addition of a Safety Margin: In order to provide a safety margin, we utilize Vatti clipping algorithm (Vatti, 1992) to expand and contract polytopes. Vatti clipping algorithm utilizes boolean operations to shift the border segments of a polytope to a desired distance. The length of the expansion is calculated to compensate for measurement and approximation errors, as well as the size of the robot. Consequently, the robot can be treated as a single point in the OCP. It is important to note that only the obstacles have to be expanded to create a safe margin around them and not the openings. Therefore, we expand the openings and sensor baseline and contract the resulting polytope to keep them in the same position, and at the same time, we contract the obstacles, which is equivalent to an expansion of the obstacles into the safe region. An example is shown in Figure 2.

3) Safe Region Decomposition: After obtaining the 2D polytope P that represents the safe region, the subsequent step is to partition the region into convex subregions. This approach offers the benefit of convexifying the constraints and allowing a simple environment description. The optimal convex partition algorithm presented by Greene (1983) employs dynamic programming to determine the minimum number of convex partitions of the original polytope.

4) Obtaining the subregion constraints: Since the output of the decomposition consists of convex polytope subregions in V-rep, it is necessary to first convert each polytope from V-rep to PG-rep using Algorithm 1 of Kochdumper and Althoff (2019). Assuming that all subregions are convex, we convert each vertex to PG-rep and compute the convex hull of the resulting PZs to form the subregion. In order to obtain the constraints for a point $\boldsymbol{\xi}_k$ to be inside a set $PZ_l = \langle \boldsymbol{c},\ \boldsymbol{G},\ [],\ \boldsymbol{E} \rangle_{PZ}$, we have to find any values of $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_p]$ that yields

$$\begin{aligned} &\boldsymbol{c} - \boldsymbol{\xi}_k + \sum_{i=2}^{h} \left(\prod_{k=1}^{p} \alpha_k^{\boldsymbol{E}(k,i)} \right) \boldsymbol{G}(\cdot, i) = \boldsymbol{0}_{n \times 1}, \\ &\alpha_k \in [-1,\ 1]\,, \end{aligned} \tag{6}$$

with $\boldsymbol{\xi}_k = \boldsymbol{C}\boldsymbol{x}_k \in \mathbb{R}^n$ being the position of the robot, $\boldsymbol{C} = [\boldsymbol{I}_n,\ \boldsymbol{0}_{n \times N_x - n}]$, and n being the set dimension. The scalar N_x is the size of the state vector $\boldsymbol{x}$. In 2D environments, $\boldsymbol{\xi} = [x\ y]^T$. The α_k variables are added to the OCP as decision variables. The β_k variables and the $\boldsymbol{G}_I$ matrix are not included when converting convex polytopes from V-rep to PG-rep, since the operations of converting vertices to PZ and computing the convex hull can be executed without adding independent generators. If there exists any $\boldsymbol{\alpha}$ that make (6) feasible, then $\boldsymbol{\xi}_k$ is inside PZ_l

3.2 NMPC Strategy

Given the constraints generated by the CG task, the NMPC solves the optimal control problem by using a non-linear optimization solver to obtain an optimal trajectory and the corresponding optimal control signal. The trajectory is constrained into the safe region, which is generally non-convex when obstacles are present, consequently, a non-convex OCP may have local optima solutions. One way to observe the possible local solutions is by referring to Figure 3, which depicts three known as openings. The openings consist of the points where the LIDAR does not detect any obstacle and are potential escape regions to reach the target $\boldsymbol{\xi}_{\text{goal}}$ (as shown in Figure 2). The OCP must select appropriate constraints to keep the problem feasible and enable the NLP solver to find the optimal solution, guiding the system safely through the safe region towards one of the openings. However, reaching different openings requires passing through different subregions, which in turn requires using different constraints for different openings.

For example, to reach opening 1 of Figure 3, the system only needs to pass through subregion 1, whereas to reach opening 3, the system must pass through subregions 1, 2, and 4. With different constraints, there are different local solutions and, then we can formulate different OCPs, one for each opening. In general, we formulate and solve p OCPs with p solutions if we have p openings. The best solution is selected based on the lowest cost according to the OCP cost functional criteria.

In order to select the subregions and constraints to use for one OCP, we form an adjacency graph of the subregions. Each node is a subregion and each edge is a physical connection of two subregions. The graph for the LIDAR output of Figure 3 is depicted in Figure 4. A graph search

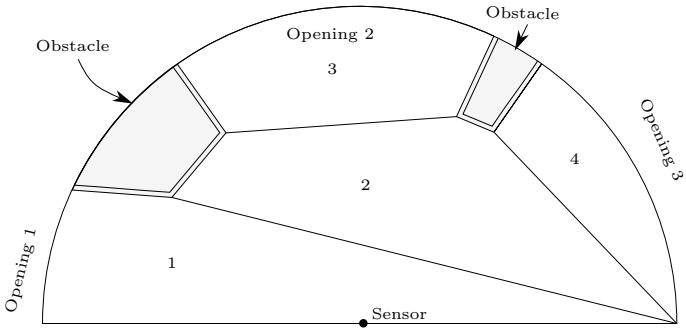

Figure 3. Subregions and openings formed by processing the LIDAR output.

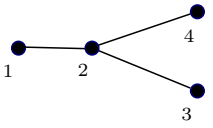

Figure 4. Adjacency graph obtained from the processed LIDAR output of Fig. 3.

algorithm is typically employed to find a path from a subregion to an opening. For instance, in order to reach opening 3, one can follow the subregion path $1 \rightarrow 2 \rightarrow 4$. The OCP formulation utilizes constraints derived from subregions 1, 2, and 4. However, imposing these constraints on all the discretized trajectory points is impractical since the points cannot be simultaneously contained within all three subregions. Consequently, the trajectory is divided

into segments or phases, and the OCP is formulated as a multi-phase OCP. Specifically, in this case, the first segment of points is constrained into subregion 1, the second segment into subregion 2, and the third segment into subregion 4.

4. NMPC FORMULATION

The OCP formulation is defined as

$$\underset{\boldsymbol{x},\boldsymbol{u},\boldsymbol{\alpha},T_1,\ldots,T_N}{\text{minimize}} \quad J = \mathfrak{T}\left(\boldsymbol{x}^{(N)}(T_N), \boldsymbol{x}_{\text{goal}}, T_N\right) + \sum_{i=1}^{N}\left[\int_{T_{i-1}}^{T_i} \mathfrak{I}\left(\boldsymbol{x}^{(i)}(t), \boldsymbol{u}^{(i)}(t), \boldsymbol{x}_{\text{goal}}\right) dt\right], \tag{7}$$

$$\underset{\forall i=1,\ldots,N}{\text{subject to}} \quad \dot{\boldsymbol{x}}^{(i)}(t) = \boldsymbol{\mathcal{V}}\left(\boldsymbol{x}^{(i)}(t), \boldsymbol{u}^{(i)}(t)\right), \tag{8}$$

$$\boldsymbol{x}_{min} \leq \boldsymbol{x}^{(i)}(t) \leq \boldsymbol{x}_{max}, \tag{9}$$

$$\boldsymbol{u}_{min} \leq \boldsymbol{u}^{(i)}(t) \leq \boldsymbol{u}_{max}, \tag{10}$$

$$\boldsymbol{x}^{(i)}(t) \in \boldsymbol{\mathcal{S}}^{(i)}(\boldsymbol{\alpha}^{(i)}), \tag{11}$$

$$\boldsymbol{x}^{(i)}(T_{i-1}) = \boldsymbol{x}^{(i-1)}(T_{i-1}), \tag{12}$$

$$t \in [T_{i-1}, T_i],\ T_{i-1} < T_i, \tag{13}$$

$$\text{subject to} \quad \boldsymbol{\mathcal{F}}\left(\boldsymbol{x}^{(N)}(T_N),\right) \leq 0, \tag{14}$$

$$T_0 = 0,\ T_N = T_p, \tag{15}$$

where Eq. (7) is the cost functional, Eqs. (8) to (13) are the constraints of each phase i, $\forall i = 1, \ldots, N$ with N phases, and Eqs. (14) and (15) are terminal constraints. The superscript (i) on a state and control signal variables indicates that the variable belongs to phase i. Eq. (8) is the phase i dynamical equations of the system, Eqs. (9) and (10) are limits on state and control signals that can be derived from the physical limitations of the system, Eqs. (11) is the i-th subregion constraints using the Eqs. (6). For each time point $t_k \in [T_{i-1},\ T_i]$ in phase i, the state $\boldsymbol{x}^{(i)}(t_k)$ is constrained using different $\boldsymbol{\alpha}_k^{(i)}$ variables. Eqs. (12) are the continuity constraints that connect the last state of phase $i-1$ with the first state of the next phase i to maintain the continuity of the dynamical Eqs. (8) between sequential phases. Eq. (13) ensures the monotonicity of the phase times (T_i). T_P is the time horizon equal to the final time (T_N).

4.1 Cost Functional

The cost functional contains a terminal cost $\mathfrak{T}$ composed of a weighted time horizon cost to penalize the final horizon time T_P, and a cost that penalizes the distance from the last trajectory point to the target point ($\boldsymbol{x}_{\text{goal}}$), which yields

$$\mathfrak{T}(\boldsymbol{x}(T_P), \boldsymbol{x}_{goal}, T_p) \triangleq\ w_t T_N + \|\boldsymbol{x}(T_N) - \boldsymbol{x}_{\text{goal}}\|^2_{\boldsymbol{P}}, \tag{16}$$

where $\boldsymbol{P}$ is obtained by solving a Ricatti algebraic equation based on the linearized model of the system with $\boldsymbol{Q}$ and $\boldsymbol{R}$ being the stage cost weighting matrices. The stage cost $\boldsymbol{\mathfrak{J}}$ penalizes the distance from state to the target and the control signal energy in the segment, resulting in

$$\boldsymbol{\mathfrak{J}}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{x}_{goal}) \triangleq \|\boldsymbol{x}(t) - \boldsymbol{x}_{\text{goal}}\|^2_{\boldsymbol{Q}} + \|\boldsymbol{u}(t)\|^2_{\boldsymbol{R}}. \tag{17}$$

4.2 Terminal Constraints

The terminal constraints vary depending on whether the system's target position is inside the sensor FOV or not. If the target $\boldsymbol{x}_{\text{goal}}$ is outside of the sensor FOV of the sensor, the terminal constraint is

$$R_{\max} - \delta \leq \|\boldsymbol{\xi}^{(N)}(T_N) - \boldsymbol{\xi}^{(1)}(T_0)\| \leq R_{\max},$$

where δ is a tolerance value to allow the terminal region to be close to the sensor range in an opening, with $\boldsymbol{\xi}(t) = [x(t)\ y(t)]^T$. If the target is inside the sensor FOV, the constraint is

$$\mathcal{B}(\boldsymbol{\xi}_{\text{goal}}, \sigma) \triangleq \{\boldsymbol{\xi} : \|\boldsymbol{\xi} - \boldsymbol{\xi}_{\text{goal}}\|_\infty \leq \sigma\}.$$

5. RESULTS

section presents numerical results that demonstrate the effectiveness of the proposed control strategy. The experiments were conducted on a personal computer equipped with an AMD Ryzen 7 3700U CPU with a clock speed of 2.3 GHz and 20 GB of RAM. The simulation was performed using Simulink and MATLAB R2023a, with the robotics toolbox emulating the LIDAR sensor. The CasADi toolbox (Andersson et al., 2019) was used to build the OCPs, which also provides exact derivatives and Hessians. The non-linear dynamical equations of the system were simulated in Simulink. For several tasks such as the polytope partition, we employed the CGAL library (Hert, 2021), while the Clipper library (Wein et al., 2021) was used to incorporate the safety margin into the obstacles.

To solve each OCP corresponding to each opening, we adopt a parallel approach at each sample time. Each OCP is transcribed to a non-linear programming problem using the direct hp-adaptive pseudospectral method (Darby et al., 2011). Subsequently, the interior-point algorithm with a filter line search, implemented in IPOPT (Wächter and Biegler, 2006), is used to solve the NLP problem.

5.1 Wheeled Mobile Robot in a 2D Environment

To evaluate the controller's performance, a wheeled mobile robot is navigated within a 2D environment with the objective of reaching a target position. The layout of the environment map forms a square, with each side measuring 20m in length. Throughout the environment, rectangular obstacles are positioned at random, featuring varying dimensions. The lengths of the obstacle's sides are determined randomly, ranging from 1m to 1.6m, following a uniform distribution. The center of each obstacle is selected at random within the map's range, adhering to a uniform distribution.

The robot starts at $\boldsymbol{x}(0) = [0.1\ 0.1\ \pi/2]^T$, and the target is at $\boldsymbol{x}_{\text{goal}} = [20\ 20\ \pi/2]^T$. A differential-drive robot kinematic model (Choset et al., 2005) is used in this study. The model considers a tracking point located at a distance of d in front of the geometric center of the robot, and the equations of motion can be expressed as

$$\begin{bmatrix}\dot{x}(t)\\ \dot{y}(t)\\ \dot{\psi}(t)\end{bmatrix} = \begin{bmatrix}(c\psi - d\,s\psi)/2 & (c\psi - d\,s\psi)/2\\ (s\psi + d\,c\psi)/2 & (s\psi + d\,c\psi)/2\\ 1/R & -1/R\end{bmatrix}\begin{bmatrix}v_R(t)\\ v_L(t)\end{bmatrix}, \tag{18}$$

where $c\psi = \cos(\psi(t))$, $s\psi = \sin(\psi(t))$, $v_R(t)$ and $v_L(t)$ are the right and left wheels linear velocities, respectively, $\boldsymbol{x}(t) = [x(t)\ y(t)\ \psi(t)]^T$ is the state vector of the system, with x and y being the position of the robot, ψ the

orientation of the robot, R the distance between the robot wheels, and $\boldsymbol{u}(t) = [v_R(t)\ v_L(t)]^T$ the input vector.

5.2 Results and Discussion

The disturbances considered in this simulation are: i) the 2D LIDAR with 0.01 m of uncertainty of measurement of distances, and ii) an additive white noise of maximum amplitude $[0.01\ 0.01\ 0.0175]^T$. The simulation parameters are: $R = 1$ m, $\sigma = 0.1$ m, $\delta = 0.5$ m, $\epsilon = 0.1$, $R_{\text{LIDAR}} = 5$ m, $w_t = 0.1$, $\boldsymbol{Q} = \text{diag}(1/5^2, 1/5^2, 1/(40\pi^2))$, $\boldsymbol{R} = \text{diag}(1/(2)^2, 1/(2)^2)$, $\boldsymbol{P} = \text{diag}(1, 1, 1)$, $U_{\max} = 1$ m/s, $U_{\min} = -1$ m/s.

Figures 5 to 7 show the results. In Figure 5, is illustrated the robot trajectory during the experiment. The robot passes comfortably through the obstacles due to the safety margin. Figure 6 shows the position and velocities. The oscillatory behavior of the velocities are due to the minimum time term in the cost functional, which makes the controller aggressive in order to reach the target as fast as possible. The oscilations also are present in the control signals (see Figure 7), but they stay close to the actuator limit of 1 m/s for the most part of the simulation, with control signals decreasing as the robot approaches the target point. At the end, the obstacles are out of the FOV, and the control signals stop oscillating and decrease as they approach the terminal region. After the robot reaches the terminal region defined by (14), the simulation is stopped.

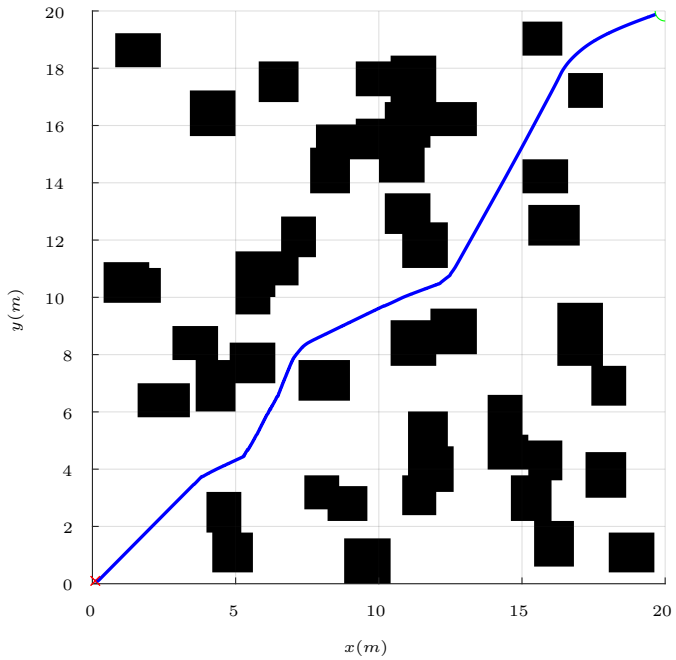

Figure 5. Robot trajectory during simulation. The black areas represent the obstacles, the red × is the starting point, and the green circle is the target area around the target point. The blue line is the trajectory obtained in simulation.

6. CONCLUSIONS

The NMPC strategy developed in this work successfully enabled a mobile robot to navigate through a cluttered 2D environment. The robot was able to comfortably avoid obstacles, thanks to the safety margin, although some oscillations were observed due to the inclusion of a minimum time term in the cost functional. The multi-phase capability of the strategy enabled convex constraints to be selected for different segments of the trajectory, which allowed the convexification of obstacle constraints and eliminated the need for a combinatorial approach for constraint selection. In future work, we intend to obtain experimental results with a real wheeled mobile robot, besides to adapt the strategy to control Unmanned Aerial Vehicles in 3D environments.

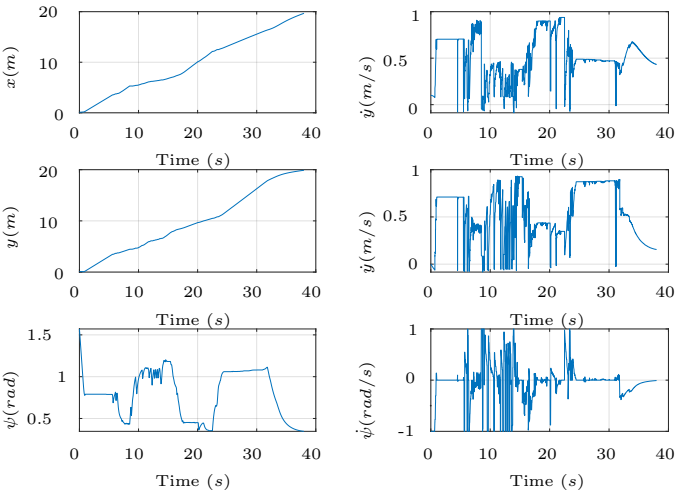

Figure 6. Position and velocities during the experiment.

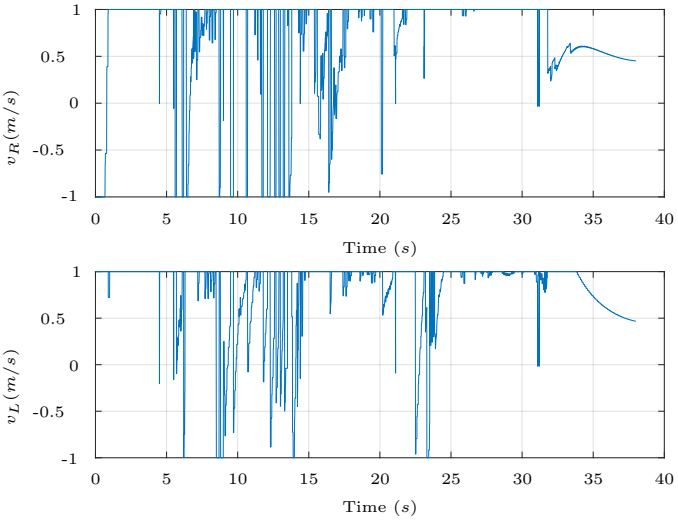

Figure 7. Control signals applied to the robot during the experiment.

REFERENCES

Andersson, J.A., Gillis, J., Horn, G., Rawlings, J.B., and Diehl, M. (2019). CasADi: A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1), 1–36.

Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G.A., and Burgard, W. (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT press.

Darby, C.L., Hager, W.W., and Rao, A.V. (2011). An hp-adaptive pseudospectral method for solving optimal

control problems. *Optimal Control Applications and Methods*, 32(4), 476–502.

Drozdova, E., Hopfgarten, S., Lazutkin, E., and Li, P. (2016). Autonomous driving of a mobile robot using a combined multiple-shooting and collocation method. *IFAC-PapersOnLine*, 49(15), 193–198.

Greene, D.H. (1983). The decomposition of polygons into convex parts. *Computational Geometry*, 1, 235–259.

Hert, S. (2021). 2D polygon partitioning. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.3 edition.

Ioan, D., Olaru, S., Niculescu, S.I., Prodan, I., and Stoican, F. (2019). Navigation in a multi-obstacle environment. From partition of the space to a zonotopic-based MPC. In *2019 18th European Control Conference (ECC)*, 1772–1777. IEEE.

Jardine, P.T. and Givigi, S.N. (2018). A robust model-predictive guidance system for autonomous vehicles in cluttered environments. *IEEE Systems Journal*, 13(2), 2034–2045.

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The nternational journal of robotics research*, 30(7), 846–894.

Kavraki, L.E., Svestka, P., Latombe, J.C., and Overmars, M.H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4), 566–580.

Kochdumper, N. and Althoff, M. (2019). Representation of polytopes as polynomial zonotopes. *arXiv preprint arXiv:1910.07271*.

Kochdumper, N. and Althoff, M. (2020). Sparse polynomial zonotopes: A novel set representation for reachability analysis. *IEEE Transactions on Automatic Control*.

Liu, J., Jayakumar, P., Stein, J.L., and Ersal, T. (2017). Combined speed and steering control in high-speed autonomous ground vehicles for obstacle avoidance using model predictive control. *IEEE Transactions on Vehicular Technology*, 66(10), 8746–8763.

Losey, D.P., Srinivasan, K., Mandlekar, A., Garg, A., and Sadigh, D. (2020). Controlling assistive robots with learned latent actions. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 378–384. IEEE.

Marzat, J., Bertrand, S., Eudes, A., Sanfourche, M., and Moras, J. (2017). Reactive MPC for autonomous MAV navigation in indoor cluttered environments: Flight experiments. *International Federation of Automatic Control, IFAC*, 50(1), 15996–16002.

Nascimento, I.B., Ferramosca, A., Piment, L.C., and Raffo, G.V. (2019). NMPC Strategy for a Quadrotor UAV in a 3D Unknown Environment. In *2019 19th International Conference on Advanced Robotics (ICAR)*, 179–184. IEEE.

Niroui, F., Zhang, K., Kashino, Z., and Nejat, G. (2019). Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments. *IEEE Robotics and Automation Letters*, 4(2), 610–617.

Pereira, J.C., Leite, V.J., and Raffo, G.V. (2021). Nonlinear model predictive control on SE (3) for quadrotor aggressive maneuvers. *Journal of Intelligent & Robotic Systems*, 101(3), 1–15.

Perez-Jimenez, M., Montes-Grova, MA., Ramon-Soria, P., Arrue, BC., and Ollero, A. (2020). POSITRON: Lightweight active positioning compliant joints robotic arm in power lines inspection. In *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, 729–736. IEEE.

Ramer, U. (1972). An iterative procedure for the polygonal approximation of plane curves. *Computer graphics and image processing*, 1(3), 244–256.

Santos, M.A., Ferramosca, A., and Raffo, G.V. (2021). Tracking nonlinear model predictive control for obstacle avoidance. In *2021 Latin American Robotics Symposium (LARS), 2021 Brazilian Symposium on Robotics (SBR), and 2021 Workshop on Robotics in Education (WRE)*, 30–35. IEEE.

Santos, M.A., Ferramosca, A., and Raffo, G.V. (2023). Nonlinear Model Predictive Control Schemes for Obstacle Avoidance. *Journal of Control, Automation and Electrical Systems*. doi:10.1007/s40313-023-01024-2.

Scott, J.K., Raimondo, D.M., Marseglia, G.R., and Braatz, R.D. (2016). Constrained zonotopes: A new tool for set-based estimation and fault detection. *Automatica*, 69, 126–136.

Vatti, B.R. (1992). A generic solution to polygon clipping. *Communications of the ACM*, 35(7), 56–63.

Wächter, A. and Biegler, L.T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1), 25–57.

Wein, R., Baram, A., Flato, E., Fogel, E., Hemmer, M., and Morr, S. (2021). CGAL: 2D minkowski sums. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.3 edition.

Zhang, J., Zhang, S., and Gao, R. (2019). Discrete-time predictive trajectory tracking control for nonholonomic mobile robots with obstacle avoidance. *International Journal of Advanced Robotic Systems*, 16(5), 1729881419877316.