Model Predictive Control with Collision Avoidance for Unknown Environment

Daniel Silvestre and Guilherme Ramos

***Abstract*—This paper proposes a model predictive control framework to design autonomous rendezvous operations for terrestrial and space missions in the scope of spacecraft and drones, in the presence of obstacles in an unknown environment. Vehicles equipped with a LiDAR collecting points associated with the obstacles are considered. By proposing an efficient method to compute the smallest ellipse that contains the collected LiDAR points, this work is able to correct, in run time, the vehicle trajectory to avoid collision and reach the rendezvous target. Finally, simulations of the proposed approach to show its effectiveness are presented.**

***Index Terms*—Autonomous vehicles; Predictive control for linear systems**

I. Introduction

A crucial and challenging problem for many terrestrial and space missions is the autonomous operation of spacecraft and drones. A relevant operation in these missions is the design of rendezvous. The classical way of designing rendezvous missions trajectories is offline, where the maneuvers are performed in an open-loop, with the need for online trajectory corrections from possible trajectory deviations or obstacles that come across [1], [2]. The advances in the computational power and efficiency of on-board processing units give rise to an increasing interest in applying Model Predictive Control (MPC) to these rendezvous guidance problems [3]–[10].

The application of rendezvous or exploration involving Unmanned Aerial Vehicles (UAVs) can often resort to consensus algorithms [11], [12] that generate waypoints to be followed, or other algorithms based on flocking rules [13], [14], which can further be improved with obstacle avoidance mechanisms based on reachability tools [15] or through a better design of the agent interactions [16], [17]. However, these strategies provide very computationally efficient methods at the expense of deviating from optimal trajectories such as when considering MPC for search and coverage missions like in [18]. Given that the MPC setup is based on optimization, it is simple to add battery energy constraints, terminal position sets, etc.

G. Ramos (guilherme.ramos@tecnico.ulisboa.pt) is with Dept. of Computer Science and Engineering, Instituto Superior Técnico, University of Lisbon, Portugal and Instituto de Telecomunicações, 1049-001 Lisbon, Portugal.

D. Silvestre is with School of Science and Technology from the NOVA University of Lisbon (FCT/UNL), 2829-516 Caparica, Portugal, with COPELABS from the Lusófona University, and also with the Institute for Systems and Robotics, Instituto Superior Técnico, University of Lisbon. dsilvestre@isr.tecnico.ulisboa.pt

This work was partially supported by the Portuguese Fundação para a Ciência e a Tecnologia (FCT) through Institute for Systems and Robotics (ISR), under Laboratory for Robotics and Engineering Systems (LARSyS) project UIDB/50009/2020, through project PCIF/MPG/0156/2019 FirePuma and through COPELABS, University Lusófona project UIDB/04111/2020.

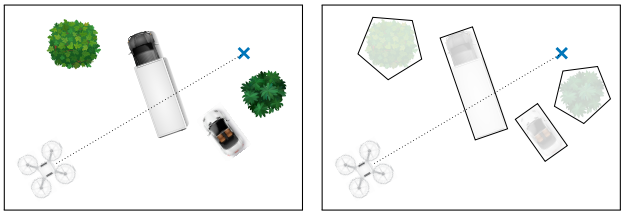
MPC and its distributed versions have been extensively studied both for linear and nonlinear models [19]. These methodologies are quite efficient assuming that the problem was adequately formulated with collision avoidance given as sets to be avoided like in [20]. In the case one opts by modeling the obstacles as convex polytopes, the work in [21] considers each hyperplane constraint and encodes them into a mixed integer formulation of the MPC problem. Such an approach is amenable to considering multiple polytopes by combining the different options but has two main disadvantages: i) it poses the inherent assumption that the obstacles are known a priori since converting between a point representation to a hyperplane is costly to perform in real-time; ii) a mixed-integer program is also computationally intensive to solve, which can be made worse once we have to include nonlinear dynamics for the vehicles.

In more recent work, the authors of [22] have encoded the obstacles by enforcing a given safety distance. This is done by adding inequality constraints that match a hyperplane separation between the state and the obstacle with the estimated heading. Such a formulation to avoid the use of mixed-integer programming resembles our approach of considering hyperplanes for some given points in the trajectories but adds two variables and three constraints for each pair of estimated states in the prediction horizon and obstacle. Moreover, it requires maintaining an estimation of the obstacle that is not available when the UAV is equipped with LiDAR sensors and we do not know the obstacle dynamics.

In [2], the authors improved on strategies using hyperplanes by considering spherical objects and attaching a tangent hyperplane constraint for each point in the estimation horizon. Then, the MPC problem is solved sequentially with constraints recomputed from the previous iteration. This approach improves on [22] but poses extra computational complexity on each sampling time as a series of MPC programs have to be solved. Moreover, computing the tangent hyperplanes has a closed-form expression for circular or spherical obstacles but it can be conservative in a LiDAR-based approach.

A last option proposed in the literature is the use of Control Barrier Functions (CBFs). The authors in [23] have proposed a solution to the problem that resorts to solving a quadratic optimization in each discrete-time slot by including a CBF constraint for a single obstacle. Such an approach also requires a priori knowledge of the obstacle and, even though quite trivial to define for circular or ellipsoidal objects, it is not amenable to polytopic obstacles or point-based readings.

Main contributions: This works proposes an efficient method that takes the collected LiDAR points as input and computes hyperplanes tangent to the smallest ellipse containing all the



(a) Original environment. (b) Possible environment set representation.

Fig. 1: A depiction of how obstacles can be modeled using sets to be added to the Model Predictive Control (MPC) optimization.

measured points. These hyperplanes are then added to an MPC formulation that is guaranteed to avoid collisions by solving two different optimization problems. All computations can be performed online and without any prior knowledge of the obstacle shape or location.

Preliminaries and Notation: The first n positive integers are denoted by $[n]$, i.e., $[n] = \{1, \dots, n\}$ and the set of integers from i to j ($i \leq j$) by $[i : j]$, i.e., $[i : j] = \{i, i+1, \dots, j\}$. Given a set $S \subset \mathbb{R}^2$, the translation of S by vector $v = (v_x, v_y) \in \mathbb{R}^2$ is denoted as $S + v$, where $S + v = \{(x, y) + (v_x, v_y) : (x, y) \in S\}$.

II. Model Predictive Control with Collision Avoidance for Known Environment

This section starts by formalizing a simpler version of the tackled problem to build the necessary definitions for the more challenging case of an unknown environment. Consider a vehicle and a set of $M \geq 0$ static (not moving) obstacles also described by closed curves $\{O_i\}_{i=1}^{M}$, with $O_i \subset \mathbb{R}^2$, as illustrated in Fig. 1 for $M = 4$. Additionally, assume that each O_i is limited, i.e., for some $\varepsilon_i > 0$ it follows that $O_i \subseteq B_{\varepsilon_i}$, where $B_r = \{x : d(x, \mathbf{0}) \leq r\}$ and d is the Euclidean distance. Given a target rendezvous position of the space $\mathbb{X} \in \mathbb{R}^2$, the goal is to devise a trajectory for the drone to follow such that it avoids the obstacles and arrives at the target place $\mathbb{X}$ until a time limit $\tau > 0$ while optimizing a given cost function (e.g. the fuel consumption of the drone.) This paper assumes that the drone can be modeled as a Linear Time-Invariant (LTI) model given by:

$$x[k+1] = Ax[k] + Bu[k]$$

where $x[k] \in \mathbb{R}^n$ is the state and $u[k] \in \mathbb{R}^m$ is the actuation. Notice that the linear model is a simplification that assumes a fully-actuated drone instead of having the acceleration dependent on the thrust and the attitude of the body. Nevertheless, if the MPC enforces a smooth and sufficiently slow acceleration, a low-level controller can track the generated trajectory with for instance an ArduPilot. Therefore, the state x is composed of position and velocity such that $x[k] = \begin{bmatrix} p[k]^\intercal & v[k]^\intercal \end{bmatrix}^\intercal$. The controller is based on the following optimization problem $\mathbf{P_1^K}$ for a fixed horizon N:

$$\begin{aligned}
\underset{\mathbf{x}[.],\mathbf{u}[.]}{\text{minimize}} \quad & J(\mathbf{x}[.], \mathbf{u}[.], \mathbb{X}) \\
\text{subject to} \quad & \mathbf{x}[0] = x[k] \\
& \mathbf{x}[\ell+1] = A\mathbf{x}[\ell] + B\mathbf{u}[\ell], \ \ell \in [0 : N-1] \\
& \mathbf{x}[\ell] \in \mathcal{X}, \ \ell \in [0 : N] \\
& \mathbf{u}[\ell] \in \mathcal{U}, \ \ell \in [0 : N-1] \\
& \mathbf{p}[\ell] \cap O_i = \emptyset, \ \ell \in [0 : N], \ i \in [M]
\end{aligned}$$

where $\mathbf{x}$ and $\mathbf{u}$ are optimization variables to match the predicted state and actuation over the horizon N, and the notation $\mathbf{x}[.]$ means the concatenation of the sequence of variables $\mathbf{x}$. Moreover, the sets $\mathcal{X}$ and $\mathcal{U}$ translate constraints on the feasible trajectories performed by the vehicle such as limited mission plane or velocity vectors, and the cost function $J(\mathbf{x}[.], \mathbf{u}[.], \mathbb{X})$ should translate the objective for the mission. For rendezvous, a possible definition of the cost function is

$$J(\mathbf{x}[.], \mathbf{u}[.], \mathbb{X}) = \|\mathbf{x}[.] - 1_{N+1} \otimes \mathbb{X}\|_Q + \|\mathbf{u}[.]\|_R$$

where $\|v\|_A := v^\intercal A v$. The problem in (1) is not convex due to the obstacle constraints, which poses difficulties when solving it by the onboard processors of a drone. In this paper, given that the MPC is linear, a solution that maintains the same level of algorithmic complexity when adding the obstacles constraints is pursued. The remainder of this section, for completeness, takes advantage of the known environment to introduce two strategies from the literature.

A. Obstacle avoidance with Sequential Linear Programming

A solution to problem $\mathbf{P_1^K}$ could trivially be pursued by modeling it as a mixed integer program if the sets are given as polytopes (a binary variable for each facet). However, for a single obstacle modeled as a circle ($O_1 = (\mathbf{p}_{\text{obstacle}}, r)$, for center $\mathbf{p}_{\text{obstacle}}$ and radius r), one could resort to the strategy in [2] that introduces a sequential linear programming approach that can roughly be described as introducing a hyperplane tangent to the projection of each point of the previous trajectory on the surface of the circle.

Algorithm 1 Sequential Convex Programming for MPC with obstacle avoidance

1: **Input:** $x[k]$, $\mathbb{X}$, $O_1 := (\mathbf{p}_{\text{obstacle}}, r)$, **maxTol**, **maxIter**
2: **Output:** $u^\star[k]$
3: $\mathbf{x}^{(0)}[.], \mathbf{u}^{(0)}[.]$ = solve (1) without (1) constraint
4: $u^\star[k] = \mathbf{u}^{(0)}[0]$
5: **if** $\mathbf{x}^{(0)}[.]$ not feasible in (1) **then**
6: $\quad \kappa = 0$
7: $\quad \mathbf{tol} = 2\mathbf{maxTol}$
8: $\quad \mathcal{I} = \emptyset$
9: $\quad$ **while tol** $\geq$ **maxTol** and $\kappa \leq$ **maxIter do**
10: $\quad\quad \mathcal{C} = \emptyset$
11: $\quad\quad$ **for all** $\ell = 1, \cdots, N$ **do**
12: $\quad\quad\quad$ **if** $\ell \in \mathcal{I}$ or $\|\mathbf{p}^{(\kappa)}[\ell] - \mathbf{p}_{\text{obstacle}}\|_2 \leq r$ **then**
13: $\quad\quad\quad\quad \mathbf{q} = \frac{\mathbf{p}^{(\kappa)}[\ell] - \mathbf{p}_{\text{obstacle}}}{\|\mathbf{p}^{(\kappa)}[\ell] - \mathbf{p}_{\text{obstacle}}\|_2}$
14: $\quad\quad\quad\quad \mathbf{p}_{\text{nearest}} = \mathbf{p}_{\text{obstacle}} + r\mathbf{q}$
15: $\quad\quad\quad\quad \mathcal{C} = \mathcal{C} \cup \mathbf{q}^\intercal \mathbf{p}^{(\kappa)}[\ell] \geq \mathbf{q}^\intercal \mathbf{p}_{\text{nearest}}$
16: $\quad\quad\quad\quad$ **if** $\ell \notin \mathcal{I}$ **then**
17: $\quad\quad\quad\quad\quad \mathcal{I} = \mathcal{I} \cup \ell$
18: $\quad\quad\quad\quad$ **end if**
19: $\quad\quad\quad$ **end if**
20: $\quad\quad$ **end for**
21: $\quad\quad \kappa = \kappa + 1$
22: $\quad\quad \mathbf{x}^{(\kappa)}[.], \mathbf{u}^{(\kappa)}[.]$ = solve (1) with $\mathcal{C}$ instead of (1)
23: $\quad\quad \mathbf{tol} = \|\mathbf{x}^{(\kappa)}[.] - \mathbf{x}^{(\kappa-1)}[.]\|_2$
24: $\quad\quad u^\star[k] = \mathbf{u}^{(\kappa)}[0]$
25: $\quad$ **end while**
26: **end if**

A visual depiction is given in Fig. 2 for the first iteration of Algorithm 1. The unconstrained trajectory resulting from line 3 is shown in blue with the points corresponding to the indices in

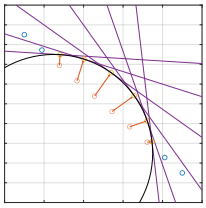


Fig. 2: Example trajectory (blue - feasible - and orange - infeasible), constraint boundaries generated by the algorithm for the infeasible point (purple) and the nearest points on the circle to each of the infeasible points of the trajectory (yellow).

set $\mathcal{I}$ shown in orange. These infeasible points will generate the purple hyperplanes (line 15) to be considered based on the closest points to the surface computed in line 14, shown in yellow. Remark that the description is required to be a circle for the projection step to be easy to calculate.
Next, an upper-bound on the computational complexity of Algorithm 1 is presented.

Proposition 1. *The computational complexity of Algorithm 1 is* $\mathcal{O}(s\,\textbf{\textit{maxIter}})$*, where* s *is the complexity of solving* (1). $\diamond$

Notice that the computational complexity of solving 1 depends on the selected solver.

B. Obstacle avoidance with Control Barrier Functions

In the mathematics field of constrained optimization, a barrier function (BF) is a continuous function whose value on a point augments to infinity as the point approaches the boundary of an optimization problem feasible region [24]. The main goal of these functions is to be used to replace inequality constraints via a penalizing term in the objective function that is often easier to handle. Another possible approach to address $\mathbf{P}_1^K$ is by using the so-called control barrier functions (CBFs) [23], [25], a particular case of BFs. CBFs allow specifying safe and unsafe sets, which, in turn, may be used to solve quadratic programs encoding the obstacle-avoiding problem. When there is a single obstacle defined as a circle, finding a suitable CBF is rather trivial but such a procedure does not generalize for multiple obstacles and partial views of the obstacles.
Obstacle avoidance using CBFs can use, for instance, the technique in [23] corresponding to the solution of the following Quadratic Program (QP) with a suitable Lyapunov function V and a CBF h:

$$\begin{aligned} u^*[x] = \underset{u\in\mathbb{R}^m,\,\delta\in\mathbb{R}}{\arg\min}\ & \frac{1}{2}\|u\|^2 + \frac{1}{2}p\delta^2 \\ \text{s.t.}\quad L_fV + L_gVu + \gamma(V) &\leq \delta \\ L_fh + L_ghu + \alpha(h) &\geq 0 \end{aligned}$$

with $p > 0$ and functions $\gamma \in \mathcal{K}$, $\alpha \in \mathcal{K}_\infty$ render the feasible set forward invariant. The notation L_f and L_g are the Lie derivative operators applied to the nonlinear dynamics affine in the actuation given by $\dot{x} = f(x) + g(x)u$. Notice that, even though our system is assumed to be running in discrete-time, the constraints in the QP only relate to the current time instant and do not pose any difficulty. This controller can have undesired equilibria but in [23] is shown how to deal with that problem by introducing a rotation to the Lyapunov function. Such details are omitted herein since for generic initial conditions, those limit cases will not happen.
For the very specific case of a circular obstacle defined as $O_1 = (\mathbf{p}_{\text{obstacle}}, r)$, a trivial choice for a CBF can be $h(x) = \|x - \mathbf{p}_{\text{obstacle}}\|_2^2 - r^2$ which satisfies all necessary conditions. However, both the SLP and the CBF-based QPs require a circular description known a priori of the obstacle which conflicts with the objective of this paper of having the drone acquiring LiDAR information as it moves through the mission plane and would add considerable conservatism when the obstacle is not circular.

III. Model Predictive Control with Collision Avoidance for Unknown Environment

In a more realistic scenario, only partial information about the obstacles given by onboard sensors that are corrupted by noise may be available. In other words, at time $k \geq 0$, only $\mathcal{P}(O_i, k)$ a portion of obstacle O_i is known. This paper assumes that the vehicle accomplishing the mission is equipped with a LiDAR (light detection and ranging device). Such a sensor, associated with the vehicle position, can be abstracted by the tuple $\mathcal{L} = (v_\mathcal{L}, r_\mathcal{L}, \theta_\mathcal{L}, \varepsilon_\mathcal{L})$, where $v_\mathcal{L}$ is a unit vector indicating the main direction that the LiDAR is pointed to, $r_\mathcal{L} > 0$ denotes the radius of reach, $\theta_\mathcal{L}$ denotes the angle covered by the LiDAR (together, $r_\mathcal{L}$ and $\theta_\mathcal{L}$ define the field-of-view (FoV) of the LiDAR), and $0 \leq \varepsilon_\mathcal{L} \ll r_\mathcal{L}$ the measurement error, see Fig. 3.

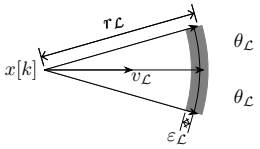


Fig. 3: LiDAR $\mathcal{L} = (v_\mathcal{L}, r_\mathcal{L}, \theta_\mathcal{L}, \varepsilon_\mathcal{L})$ range of observations.

With a slight abuse of notation, the cloud of points collected by the LiDAR $\mathcal{L}$ at time instance $k \in \mathbb{N}$ is denoted by $\mathcal{C}_\mathcal{L}(k)$ and is illustrated in Fig. 4. The convex hull of $\mathcal{C}_\mathcal{L}(k)$ defines a portion of an obstacle identified by the LiDAR. Hence, the problem $\mathbf{P}_1^\mathbf{U}$ to be solved is the following.

$$\begin{aligned} \underset{\mathbf{x}[.],\mathbf{u}[.]}{\text{minimize}}\quad & J(\mathbf{x}[.], \mathbf{u}[.], \mathbb{X}) \\ \text{subject to}\quad & \mathbf{x}[0] = x[k] \\ & \mathbf{x}[\ell+1] = A\mathbf{x}[\ell] + B\mathbf{u}[\ell],\ \ell \in [0:N-1] \\ & \mathbf{x}[\ell] \in \mathcal{X},\ \ell \in [0:N] \\ & \mathbf{u}[\ell] \in \mathcal{U},\ \ell \in [0:N-1] \\ & \mathbf{p}[\ell] \cap \text{cvxHull}(\mathcal{L}_k) = \emptyset. \end{aligned}$$

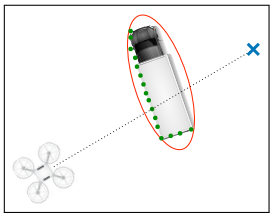


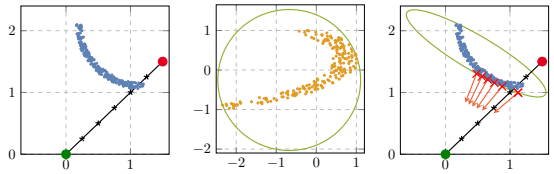
Fig. 4: Illustration of the points $\mathcal{C}_\mathcal{L}(k)$ (green) collected by the LiDAR and the desired ellipsoid that overbounds the obstacle.

The attentive reader could be tempted to follow the typical approach in the literature of finding an ellipse through the minimization of the volume, which can be written as convex program or even through the use of Khachiyan's Algorithm. However, such methods are at least quadratic in complexity. Thus, this paper presents a very efficient method with linear complexity that is specifically tailored to the problem at hand, which is presented in pseudo-code in Algorithm 2.

Algorithm 2 Find smallest ellipse enclosing a set of LiDAR points and closest ellipse orthogonal vectors to each trajectory point that are closest to the initial trajectory point

1: **input**: cloud of LiDAR points $D \subset \mathbb{R}^2$, drone trajectory points $T \subset \mathbb{R}^2$, number of points N to define the border of the ellipse that encloses D
2: **output**: cloud of points defining the border of an ellipse E, and set S of ellipse tangents for the closets ellipse points to each trajectory point
3: **compute:** $\mu = \frac{1}{|D|}\sum_{p\in D} p$
4: ▷ set the mean point μ as the origin $(0,0)$
5: **set:** $D' = D - \mu$
6: **find:** the point $p \in D$ with maximum norm
7: **compute:** $\theta = -\frac{\arccos(p)}{\|p\|_2}$
8: **set:** $\Theta = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$
9: ▷ rotate the points according to the main direction to be align with the axis
10: **set:** $D'' = \{q \cdot \Theta \,:\, q \in D'\}$
11: **set:** $x_{\max} = \max\{x \,:\, (x,y) \in D''\}$ and $y_{\max} = \max\{y \,:\, (x,y) \in D''\}$
12: **set:** $x_{\min} = \min\{x \,:\, (x,y) \in D''\}$ and $y_{\min} = \min\{y \,:\, (x,y) \in D''\}$
13: ▷ re-scale the points to have similar range in both axis
14: **set:** $D''' = \left\{\left(\frac{x-x_{\min}}{x_{\max}}, \frac{y-y_{\min}}{y_{\max}}\right) \,:\, (x,y) \in D''\right\}$
15: ▷ find circle that encloses the points
16: **compute:** $c = (c_x, c_y)$ and r, the center and radius of the smallest enclosing circle for the points in D'' using [26]
17: ▷ compute the set of points that define the border of the ellipse
18: **set:** $E' = \left\{r.(\cos(\vartheta) + c_x, \sin(\vartheta) + c_y) \,:\, \vartheta \in \{0, \frac{2\pi}{N}, \ldots, 2\pi\}\right\}$
19: ▷ transform the border of the ellipse back to enclose the input cloud of points
20: **set:** $E'' = \{(x_{\max}.x + x_{\min}, y_{\max}.y + y_{\min}) \,:\, (x,y) \in E'\}$
21: **set:** $E = \left\{q \cdot \Theta^{-1} + \mu \,:\, q \in E''\right\}$
22: ▷ for each point of the trajectory, compute the tangent vector to the ellipse in the ellipse point closer to the trajectory point
23: **set:** $S = \emptyset$
24: **for** each $p \in T$ **do**
25: **compute:** p', the closest point in E to point p that is closest to the first trajectory point
26: **set:** $S = S \cup \{v\}$, where v is the orthogonal vector to the ellipse intersecting point p'
27: **end for**

The algorithm starts by centering the cloud of points at the origin (lines 3 and 5) followed by a rotation to align them with the axis (lines 6 to 10). Then a re-scale of the space is done to have the same dispersion over each axis in lines 11 to 14. After finding the smallest circle, it revert to the original coordinates to get the ellipse (lines 16 to 21). The constraints can then be found by computing tangent hyperplanes to points that correspond to the projection of the desired trajectory waypoints on the boundary of the ellipse (last for cycle in the algorithm). For a visualization of the algorithm steps see Fig. 5 where Fig. 5(a) is the initial input cloud of points followed by the rescaling of the points in Fig. 5(b) and the final ellipse after inverting the rescaling with the orthogonal vectors to the tangent hyperplanes being shown in Fig. 5(c).



(a) (b) (c)

Fig. 5: Illustration of Algorithm 2. The trajectory starts at the origin, "●", is represented by the black points, and the rendezvous location is "●" (Figures (a) and (c)). In blue, the LiDAR set of points are shown (Figures (a) and (c)). Figure (b) has the normalized LiDAR set of points, in yellow, to encounter the closest enclosing circle, in green. Finally, the enclosing ellipse that result from the previous circle is in Figure (c), depicted in green, and the orthogonal vectors are depicted in orange (and start from the red ellipse points).

Next, we give the computational complexity of Algorithm 2.

Proposition 2. *The computational complexity of Algorithm 2 is* $\mathcal{O}(\max\{|D|, |T|^2, N\})$. ◇

Proof. The computational complexity of Algorithm 2 is the sum of the computational complexities of each step of the algorithm. Steps 3–14 are bounded by $\mathcal{O}(|D|)$. Step 16 also has cost $\mathcal{O}(|D|)$ [26]. Steps 18–21 have computational complexity of $\mathcal{O}(N)$. Finally, steps 23–27 have computational complexity of $\mathcal{O}(|T|^2)$. Thus, the total computational complexity is bounded by $\mathcal{O}(\max\{|D|, |T|^2, N\})$. □

The constraints found in Algorithm 2 can be combined with the SLP optimization by changing line 21 of Algorithm 1 to have the following MPC for iteration κ, and name it $\mathbf{P}^{\mathbf{U}}_{\mathbf{UNSAFE}}$:

$$
\begin{aligned}
\underset{\mathbf{x}^{(\kappa)}[.],\mathbf{u}^{(\kappa)}[.]}{\text{minimize}} \quad & J(\mathbf{x}^{(\kappa)}[.], \mathbf{u}^{(\kappa)}[.], \mathbb{X}) \\
\text{subject to} \quad & \mathbf{x}^{(\kappa)}[0] = x[k] \\
& \mathbf{x}^{(\kappa)}[\ell+1] = A\mathbf{x}^{(\kappa)}[\ell] + B\mathbf{u}^{(\kappa)}[\ell],\ \ell \in [0:N-1] \\
& \mathbf{x}^{(\kappa)}[\ell] \in \mathcal{X},\ \ell \in [0:N] \\
& \mathbf{u}^{(\kappa)}[\ell] \in \mathcal{U},\ \ell \in [0:N-1] \\
& \mathbf{q}^{(\kappa)\intercal}_{\text{ell}}[\ell]\mathbf{p}^{(\kappa)}[\ell] \geq \mathbf{q}^{(\kappa)\intercal}_{\text{ell}}[\ell]\mathbf{p}^{(\kappa)}_{\text{ell}}[\ell],\ \ell \in \mathcal{C}.
\end{aligned}
$$

where, as in Algorithm 1, the set $\mathcal{C}$ corresponds to the indices of all points that were inside the ellipse and the vectors $\mathbf{q}^{(\kappa)}_{\text{ell}}[\ell]$ and $\mathbf{p}^{(\kappa)}_{\text{ell}}[\ell]$ are the output of Algorithm 2 shown in orange and red in Fig. 5(c), respectively.

Notice that the solution of (3) is not a safe controller in the sense that the generated trajectory could travel to a location in the unknown space from which the MPC cannot recover in a future instant. For that reason, let us also define the following optimization with variables $\mathbf{x}^{(\kappa)}_s[.], \mathbf{u}^{(\kappa)}_s[.]$ and a safe horizon H deemed as the problem $\mathbf{P}^{\mathbf{U}}_{\mathbf{SAFE}}$:

$$
\begin{aligned}
\underset{\mathbf{x}^{(\kappa)}_s[.],\mathbf{u}^{(\kappa)}_s[.]}{\text{minimize}} \quad & J(\mathbf{x}^{(\kappa)}_s[.], \mathbf{u}^{(\kappa)}_s[.], \mathbb{X}) \\
\text{subject to} \quad & \mathbf{x}^{(\kappa)}_s[0] = x[k] \\
& \mathbf{x}^{(\kappa)}_s[\ell+1] = A\mathbf{x}^{(\kappa)}_s[\ell] + B\mathbf{u}^{(\kappa)}_s[\ell],\ \ell \in [0:N-1] \\
& \mathbf{x}^{(\kappa)}_s[\ell] \in \mathcal{X},\ \ell \in [0:N] \\
& \|\mathbf{p}^{(\kappa)}_s[\ell] - p[k]\|_2 \leq r_{\mathcal{L}},\ \ell \in [0:N] \\
& \mathbf{u}^{(\kappa)}_s[\ell] \in \mathcal{U},\ \ell \in [0:N-1] \\
& \mathbf{q}^{(\kappa)\intercal}_{\text{ell}}[\ell]\mathbf{p}^{(\kappa)}_s[\ell] \geq \mathbf{q}^{(\kappa)\intercal}_{\text{ell}}[\ell]\mathbf{p}^{(\kappa)}_{\text{ell}}[\ell],\ \ell \in \mathcal{C} \\
& \|\mathbf{x}^{(\kappa)}_s[\ell] - \mathbf{x}^{(\kappa)}[\ell]\|_2 \leq \epsilon,\ \ell \in [0:H] \\
& \|\mathbf{u}^{(\kappa)}_s[\ell] - \mathbf{u}^{(\kappa)}[\ell]\|_2 \leq \epsilon,\ \ell \in [0:H]
\end{aligned}
$$

for a sufficiently small $\epsilon > 0$. The problem $\mathbf{P}^{\mathbf{U}}_{\mathbf{SAFE}}$ in (4) adds three extra constraints to enforce that the entire trajectory lies within the observable space of the LiDAR and also an overlap of H steps between the safe and unsafe trajectories. That way, the vehicle can safely implement whatever is given as the input of (3) and switch back to the trajectory in (4) if needed. The two strategies may be combined as detailed in Algorithm 3, denoting a trajectory encompassing both the $\mathbf{x}(\kappa)[.]$ and $\mathbf{u}(\kappa)[.]$ variables over the entire horizon N by $T^{(k)}$ for the iteration k and adding a subscript to distinguish between the safe and unsafe MPCs.

Algorithm 3 Proposed Safe MPC with guaranteed Collision Avoidance

1: **Input:** $x[k]$, $\mathbb{X}$, $T^{(k-1)}_{\text{safe}}$
2: **Output:** $u^\star[k]$
3: **set:** feasible = false
4: **compute:** an unsafe trajectory $T^{(k)}_{\text{unsafe}}$ using Algorithm 1 with the MPC $\mathbf{P}^{\mathbf{U}}_{\mathbf{UNSAFE}}$ problem and set feasible = $\mathbf{true}$ if it had a feasible solution
5: **if** feasible **then**
6: **update:** $u^\star[k]$ from trajectory $T^{(k)}_{\text{unsafe}}$
7: **compute:** a safe trajectory $T^{(k)}_{\text{safe}}$ using Algorithm 1 with the MPC $\mathbf{P}^{\mathbf{U}}_{\mathbf{SAFE}}$ problem
8: **else**
9: **update:** $u^\star[k]$ from trajectory $T^{(\kappa-1)}_{\text{safe}}$
10: **compute:** a safe trajectory $T^{(k)}_{\text{safe}}$ using Algorithm 1 with the MPC $\mathbf{P}^{\mathbf{U}}_{\mathbf{SAFE}}$ problem using $T^{(k-1)}_{\text{safe}}$ as the base trajectory
11: **end if**

Remark 1. *In practice, solving $\mathbf{P}^{\mathbf{U}}_{\mathbf{SAFE}}$ may be avoided by encoding in $\mathcal{X}$ velocity constraints such that the drone can safely stop within the specified H horizon without leaving the observed radius of the LiDAR.*

IV. Theoretical Guarantees

This section shows two results that give the necessary safety guarantees to the proposed MPC approach. The first result shows that there are no collisions with the obstacles within the observed space (next lemma) by showing there is a hyperplane separating all LiDAR points from the trajectory points.

Lemma 1. *Given a point cloud $\mathcal{C}_{\mathcal{L}}(k)$ collected by the LiDAR $\mathcal{L}$ at the time $k \geq 0$, representing a partial view of an obstacle border $\partial O \subset \mathbb{R}^2$, if $p[k] \notin O$, $\mathbf{P}^{\mathbf{U}}_{\mathbf{UNSAFE}}$ is feasible, and for all predicted positions $\ell \in [N]$ we have $\|\mathbf{p}^{(\kappa)}[\ell] - p[k]\|_2 \leq r_{\mathcal{L}}$, then, $\mathbf{p}^{(\kappa)}[.] \notin O$.* $\diamond$

Proof. Let us consider that all points in $C_{\mathcal{L}}(k)$ belong to the ellipse $\mathbb{E}$ produced by Algorithm 2. The feasibility of $\mathbf{P}^{\mathbf{U}}_{\mathbf{UNSAFE}}$ implies that all points in the horizon $\mathbf{p}^{(\kappa)}[\ell]$ satisfy the inequalities $\mathbf{q}^{(\kappa)\mathsf{T}}_{\text{ell}}[\ell]\mathbf{p}^{(\kappa)}[\ell] \geq \mathbf{q}^{(\kappa)\mathsf{T}}_{\text{ell}}[\ell]\mathbf{p}^{(\kappa)}_{\text{ell}}[\ell]$ for hyperplanes tangent to the ellipse defined by the vectors $\mathbf{q}^{(\kappa)\mathsf{T}}_{\text{ell}}[\ell]$. Thus, given the assumption that for all $\ell \in [N]$ we have $\|\mathbf{p}^{(\kappa)}[\ell] - p[k]\|_2 \leq r_{\mathcal{L}}$, the hyperplanes serve as certificates separating the set O from each point $\mathbf{p}^{(\kappa)}_{\text{ell}}[\ell]$. Therefore, suppose, by contradiction, that there is a point $p \in C_{\mathcal{L}}(k)$ such that $p \notin \mathbb{E}$. This would entail that the affine transformation of p, p', is not in the same affine transformation of $\mathbb{E}$ that transforms back the ellipse into a circle $\mathbb{C}$ (i.e., $p' \notin \mathbb{C}$). Therefore, step 16 of Algorithm 2 would not obtain a circle that encloses all the points (including p'). This contradicts the correction of the algorithm in [26], thus proving the conclusion since all points in $C_{\mathcal{L}}(k)$ belong to the ellipse $\mathbb{E}$. $\square$

Lemma 1 asserts that within the observable region, as long as the unsafe MPC is feasible, there is no collision with the obstacle part that is viewed by the LiDAR. The next result shows that if $\mathbf{P}^{\mathbf{U}}_{\mathbf{UNSAFE}}$ is not feasible, the overall control strategy in Algorithm 3 is collision free.

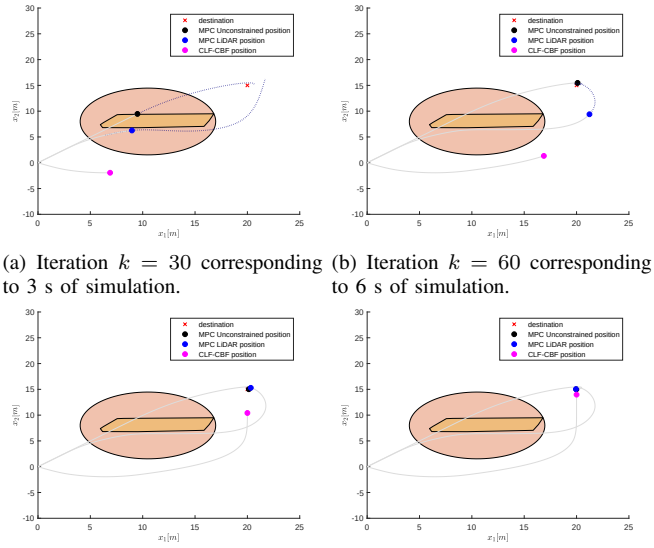
Theorem 1. *Consider a vehicle equipped with a LiDAR performing a rendezvous mission in an unknown environment using Algorithm 3 for which it is always possible to solve $\mathbf{P}^{\mathbf{U}}_{\mathbf{SAFE}}$ with $H > 1$. If $p[0] \notin O$, then $p[k] \notin O, \forall k > 0$.* $\diamond$

Proof. Let us consider at time step $k > 0$ that $p[k] \notin O$. If $\mathbf{P}^{\mathbf{U}}_{\mathbf{UNSAFE}}$ is feasible, from Lemma 1 it follows that $p[k+1] \notin O$. If $\mathbf{P}^{\mathbf{U}}_{\mathbf{UNSAFE}}$ is not feasible, given line 9 in Algorithm 3, the vehicle will switch to trajectory $T^{(k-1)}_{\text{safe}}$, which always exists with $H > 1$ using the statement of the theorem. Moreover, $T^{(k-1)}_{\text{safe}}$ is such that $\|\mathbf{p}^{(\kappa)}_s[\ell] - p[k-1]\|_2 \leq r_{\mathcal{L}}$ for all values of $\ell \in [N]$ and, using Lemma 1 entails that $p[k+1]$ resulting from using $T^{(k-1)}_{\text{safe}}$ will also satisfy $p[k+1] \notin O$. By induction, the conclusion follows. $\square$

Remark 2. *To consider a vehicle that is not simply represented as a point in space, the value $\varepsilon_{\mathcal{L}}$ used in Algorithm 2 can be increased to account for the vehicle radius. In practice, the assumptions in Theorem 1 can be enforced by setting up a maximum velocity in the safe MPC that matches the free space that the vehicle is allowed to stop before exiting the measuring radius of the LiDAR.* $\diamond$

V. Illustrative Examples

This section provides simulation results showing the trajectories achieved with the proposed solution and compared against a QP using the CLF-CBF framework. The dynamics is described by a discrete-time double integrator with a sampling time of 0.1 s. The rendezvous point is assumed to be in the point $\begin{bmatrix}20 & 15\end{bmatrix}^\mathsf{T}$ with the vehicle having a max velocity of 2.5 m/s and an acceleration of 1.5 m/s^2 and the simulation running for 12 s. The obstacle is randomly selected and then overbounded by a circle with such information assumed to be known *a priori* only for the design of the CLF-CBF controller. The results of the simulation are presented in Fig. 6 with the trajectories being shown in each subfigure for every multiple of 3 s. The generated trajectory by the current proposal is quite close to the obstacle and respects the maximum noise level for each point. In contrast, the CLF-CBF approach is even more conservative than following the known circle that overbounds the obstacle. Moreover, it takes around the 12 seconds to reach the rendezvous whereas the proposed MPC took 9 s. Both the unconstrained MPC and the QP arising from the CLF-CBF took under 0.02 s in each iteration whereas the proposed MPC is typically spending around 0.1 s. These values point towards the feasibility in practice of the proposed method as the simulations are being run in Matlab on an HP machine with a Intel Core i7-8550U CPU @ 1.80GHz and 12 GB of memory resorting to Yalmip as the language to model optimization



(a) Iteration $k = 30$ corresponding to 3 s of simulation.

(b) Iteration $k = 60$ corresponding to 6 s of simulation.

(c) Iteration $k = 90$ corresponding to 9 s of simulation.

(d) Iteration $k = 120$ corresponding to 12 s of simulation.

Fig. 6: Simulation of 12 s of a drone executing the CLF-CBF controller in comparison with the proposed MPC and the unconstrained version when in the presence of a polytopic obstacle for which is known a circular overbound. The dashed line in from of the drone serves to show the predicted horizon of the trajectory.

problems. Once implemented in C or C++, the method can fit in the typical values used for the clock of ArduPilots [27].

VI. Conclusions & Future Research

This work presented a model predictive control framework to design autonomous rendezvous operations in the presence of obstacles in an unknown environment. To this end, vehicles equipped with a LiDAR were considered, and an efficient and effective method to compute the smallest ellipse containing the collected points was presented. The generated ellipses allow us to encode the obstacles using linear inequalities in the optimization. Finally, simulations of the proposed method to demonstrate its effectiveness are shown.

Avenues for further research include extending the proposed method to cope with sets of more than one obstacle, and the design of efficient approaches to compute ellipsoids in $\mathbb{R}^3$ to extend the proposed method, for instance using the exact formula to compute the convex hull of collected obstacle's points as in [28]. Moreover, running an experiment with a boat equipped with a laser is the next step to have real experimental data validating the proposed method.

References

[1] W. Fehse, *Automated rendezvous and docking of spacecraft*. Cambridge university press, 2003, vol. 16.

[2] A. Botelho, B. Parreira, P. N. Rosa, and J. M. Lemos, *Predictive Control for Spacecraft Rendezvous*. Springer.

[3] E. N. Hartley, "A tutorial on model predictive control for spacecraft rendezvous," in *2015 European Control Conference (ECC)*. IEEE, 2015, pp. 1355–1361.

[4] A. Richards and J. How, "Performance evaluation of rendezvous using model predictive control," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2003, p. 5507.

[5] S. Di Cairano, H. Park, and I. Kolmanovsky, "Model predictive control approach for guidance of spacecraft rendezvous and proximity maneuvering," *International Journal of Robust and Nonlinear Control*, vol. 22, no. 12, pp. 1398–1427, 2012.

[6] E. N. Hartley, P. A. Trodden, A. G. Richards, and J. M. Maciejowski, "Model predictive control system design and implementation for spacecraft rendezvous," *Control Engineering Practice*, vol. 20, no. 7, pp. 695–713, 2012.

[7] A. Weiss, M. Baldwin, R. S. Erwin, and I. Kolmanovsky, "Model predictive control for spacecraft rendezvous and docking: Strategies for handling constraints and case studies," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 4, pp. 1638–1647, 2015.

[8] R. Vazquez, F. Gavilan, and E. F. Camacho, "Model predictive control for spacecraft rendezvous in elliptical orbits with on/off thrusters," *IFAC-PapersOnLine*, vol. 48, no. 9, pp. 251–256, 2015.

[9] S. Zhu, R. Sun, J. Wang, J. Wang, and X. Shao, "Robust model predictive control for multi-step short range spacecraft rendezvous," *Advances in Space Research*, vol. 62, no. 1, pp. 111–126, 2018.

[10] M. Leomanni, G. Bianchini, A. Garulli, and R. Quartullo, "Sum-of-norms periodic model predictive control for space rendezvous," *IEEE Transactions on Control Systems Technology*, vol. 30, no. 3, pp. 1311–1318, 2021.

[11] D. Silvestre, J. P. Hespanha, and C. Silvestre, "Broadcast and gossip stochastic average consensus algorithms in directed topologies," *IEEE Transactions on Control of Network Systems*, vol. 6, no. 2, pp. 474–486, 2019.

[12] G. Ramos, D. Silvestre, and C. Silvestre, "General resilient consensus algorithms," *International Journal of Control*, vol. 95, no. 6, pp. 1482–1496, 2022.

[13] R. Ribeiro, D. Silvestre, and C. Silvestre, "Decentralized control for multi-agent missions based on flocking rules," in *CONTROLO 2020*, J. A. Gonçalves, M. Braz-César, and J. P. Coelho, Eds. Cham: Springer International Publishing, 2021, pp. 445–454.

[14] ——, "A rendezvous algorithm for multi-agent systems in disconnected network topologies," in *2020 28th Mediterranean Conference on Control and Automation (MED)*, 2020, pp. 592–597.

[15] D. Silvestre, P. Rosa, J. P. Hespanha, and C. Silvestre, "Set-consensus using set-valued observers," in *American Control Conference (ACC), 2015, Chicago, Illinois, USA.*, July 2015.

[16] ——, "Finite-time convergence policies in state-dependent social networks," in *2015 American Control Conference (ACC)*, 2015, pp. 1041–1046.

[17] ——, "Stochastic and deterministic state-dependent social networks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 2, pp. 911–926, 2022.

[18] G. Gramajo and P. Shankar, "An efficient energy constraint based uav path planning for search and coverage," *International Journal of Aerospace Engineering*, vol. 2017, 2017.

[19] E. Camponogara, D. Jia, B. Krogh, and S. Talukdar, "Distributed model predictive control," *IEEE Control Systems Magazine*, vol. 22, no. 1, pp. 44–52, 2002.

[20] A. Thibbotuwawa, G. Bocewicz, G. Radzki, P. Nielsen, and Z. Banaszak, "Uav mission planning resistant to weather uncertainty," *Sensors*, vol. 20, no. 2, p. 515, 2020.

[21] F. Stoican, T.-G. Nicu, and I. Prodan, "A mixed-integer mpc with polyhedral potential field cost for obstacle avoidance," in *2022 American Control Conference (ACC)*, 2022, pp. 2039–2044.

[22] S. H. Nair, E. H. Tseng, and F. Borrelli, "Collision avoidance for dynamic obstacles with uncertain predictions using model predictive control," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 5267–5272.

[23] M. F. Reis, A. P. Aguiar, and P. Tabuada, "Control barrier function-based quadratic programs introduce undesirable asymptotically stable equilibria," *IEEE Control Systems Letters*, vol. 5, no. 2, pp. 731–736, 2020.

[24] Y. Nesterov *et al.*, *Lectures on convex optimization*. Springer, 2018, vol. 137.

[25] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 6271–6278.

[26] N. Megiddo, "Linear-time algorithms for linear programming in r^3 and related problems," *SIAM Journal on Computing*, vol. 12, no. 4, pp. 759–776, 1983.

[27] J. P. Hespanha, "Tenscalc: A toolbox to generate fast code to solve nonlinear constrained minimizations and compute nash equilibria," *Mathematical Programming Computation*, vol. 14, no. 3, pp. 451–496, 2022.

[28] D. Silvestre, "Accurate guaranteed state estimation for uncertain lpvs using constrained convex generators," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 4957–4962.