Real-Time Multi-Convex Model Predictive Control for Occlusion-Free Target Tracking With Quadrotors

HOUMAN MASNAVI[1], **VIVEK KANTILAL ADAJANIA**[2], **KARL KRUUSAMÄE**[1], **AND ARUN KUMAR SINGH**[1]

[1]Institute of Technology, University of Tartu, 51004 Tartu, Estonia
[2]University of Toronto Institute for Aerospace Studies (UTIAS), Toronto, ON M5S 1A1, Canada

Corresponding author: Arun Kumar Singh (arun.singh@ut.ee)

ABSTRACT This paper proposes a Model Predictive Control (MPC) algorithm for target tracking amongst static and dynamic obstacles. Our main contribution lies in improving the computational tractability and reliability of the underlying non-convex trajectory optimization. The result is an MPC algorithm that runs real-time on laptops and embedded hardware devices such as Jetson TX2. Our approach relies on novel reformulations for the tracking, collision, and occlusion constraints that induce a multi-convex structure in the resulting trajectory optimization. We exploit these mathematical structures using the split Bregman Iteration technique, eventually reducing our MPC to a series of convex Quadratic Programs solvable in a few milliseconds. The fast re-planning of our MPC allows for occlusion and collision-free tracking in complex environments even while considering a simple constant-velocity prediction for the target trajectory and dynamic obstacles. We perform extensive bench-marking in a realistic physics engine and show that our MPC outperforms the state-of-the-art algorithms in visibility, smoothness, and computation-time metrics.

INDEX TERMS Quadrotors, target-tracking, occlusion, collision avoidance, dynamic obstacles.

I. INTRODUCTION

Target tracking is one of the most popular and important applications of quadrotors. It forms an important component of aerial cinematography pipelines [1], [2]. It is also critical for perception-aware control wherein a quadrotor needs to navigate while always keeping certain features or fiducial markers in its field of view [3] for improved state estimation. Target-tracking algorithms can also be re-purposed for autonomous structural inspection purposes. While target tracking in free-space is relatively simple, occlusions stemming from static and dynamic obstacles pose a difficult challenge in cluttered environments. Moreover, often the occlusion avoidance is in direct conflict with the collision avoidance requirement. In other words, trajectories that are trivially collision-free lead to severe occlusion of the target.

The associate editor coordinating the review of this manuscript and approving it for publication was Juan Liu.

A. CORE CHALLENGE

In this paper, we focus only on motion planning and control aspects of the target tracking problem, assuming that a robust computer vision pipeline is available for estimating the position of the target and quadrotor's state. In this context, the core challenge stems from the complex nature of the occlusion/collision avoidance constraints. We can view these constraints as highly non-linear and non-convex functions of states/control if we adopt an optimization perspective. As a result, it is computationally challenging to include them within trajectory optimization or Model Predictive Control (MPC) algorithms. Although the solution process of non-convex optimizations is well understood, their application in real-time control and motion planning remains problematic. To be more precise, optimizers like Gradient Descent (GD), sequential quadratic programming(SQP), Gauss-Newton, etc., rely heavily on the quality of the initial guess or more precisely, how close it is to the optimal solution. Furthermore, their underlying numerical structure

ensures slow, conservative progress towards the optimal solution.[1] One possible approach to improve the computational tractability of non-convex optimizations is to leverage the partial convex structures in the problem. For example, problems with the so-called difference of convex structure are efficiently solvable through a heuristic called convex-concave procedure [4]. However, the partial convex structures are often not apparent, and one needs to employ carefully constructed reformulations to make these structures more explicit. Our proposed work is a step forward in this direction. Although we consider the specific application of target tracking with quadrotors, we expect the developed formulation to have a much broader impact (see Section VII).

B. CONTRIBUTION
1) ALGORITHMIC
For the first time, we present a multi-convex approximation of trajectory optimization associated with target tracking problems. We achieve this by rephrasing tracking and occlusion/collision avoidance constraints in a single unified form that reassembles a polar representation of the Euclidean distance (see Eqn. (10) and (16)). This is strikingly different from the models employed in existing works [5] (see Eqn. (2)) and [6] (see Eqn. (6)). We show that our chosen representation induces a multi-convex structure in these constraints (see Defn. 1). Thus, when we combine it with the Alternating Minimization (AM) [7] and split-Bregman technique [8], [9], our trajectory optimization decomposes into three smaller blocks: one convex Quadratic Programs (QP) and two parallel single-variable optimizations solvable in closed form. Importantly, all the individual blocks scale linearly with the number of obstacles. Leveraging all the aforementioned reformulations, our developed optimizer can produce diverse trajectories from arbitrary initial guesses in real-time. Finally, we construct a Model Predictive Control algorithm that uses our multi-convex optimizer in a receding horizon manner following real-time iteration paradigm [10]. Please refer Section II-E for a summary of contribution over author's prior work.

2) APPLIED
We provide an open-source implementation of our optimizer integrated with Gazebo physics Engine in Robot Operating System framework [11]: https://bit.ly/3fLI6zi. We have tested our optimizer extensively in different benchmarks and provide access to those as well. Our released implementation is expected to spur further development in this field.

3) STATE-OF-THE-ART PERFORMANCE
We perform two sets of benchmarking to validate the superiority of our proposed optimizer and the MPC algorithm built on top it. For the former, we compare our AM/split-Bregman

approach with state-of-the-art convex-concave procedure (CCP) [4]. We show that our optimizer is robust to initialization and outperforms CCP by more than two orders of magnitude in computation time while achieving comparable optimal cost. For the latter, we benchmark our multi-convex MPC against existing state-of-the-art algorithms using three metrics: visibility score (extent of occlusion constraint satisfaction), acceleration norm (optimal cost), and computation time. In environments with just static obstacles, we outperform recent work [12] in avoiding occlusion from the obstacles while using up to 30% less control effort. Our MPC's performance is a direct consequence of small computation time ($0.006s$) that is one order of magnitude lower than that of [12]. Among existing works, only a few consider occlusion from dynamic obstacles during tracking. The MPC approach of [5] can be regarded as the current state-of-the-art, and we show that our MPC also demonstrates substantial improvement over it in terms of the chosen metrics.

II. PROBLEM FORMULATION AND RELATED WORKS
In this section, we first present a generic formulation for target tracking with quadrotors and then use it to review the existing works and contrast them with the critical ideas of the proposed work. We begin by establishing the notation style used in our formulation.

A. SYMBOLS AND NOTATIONS
We will use normal-faced small case letters to represent scalars, while bold font variants represent vectors. We will represent matrices through bold-font upper case letters. The variable t will represent the time stamp of a variable. The upper case variant T will denote the transpose of a matrix. We summarize the main symbols in Table 1. We also define some symbols in their first place of use. We use a special construction at some places in the paper where a variable at different time-stamps is stacked to form a vector. For example, $\boldsymbol{\alpha}$ will be formed by stacking different $\alpha(t)$.

B. TRAJECTORY OPTIMIZATION FOR TARGET TRACKING
We can formalize target-tracking as the following optimization problem:

$$\min \sum_t \ddot{x}(t)^2 + \ddot{y}(t)^2 + \ddot{z}(t)^2, \tag{1a}$$

$$s_{min} \leq \left\| \begin{bmatrix} x(t) - x_r(t) \\ y(t) - y_r(t) \\ z(t) - z_r(t) \end{bmatrix} \right\|_2 \leq s_{max}, \tag{1b}$$

$$(x(t), y(t), z(t)) \in \mathcal{C}_b, \mathcal{C}_{free} \forall t. \tag{1c}$$

The cost function ensures smoothness of the computed trajectory by minimizing the norm of the acceleration at each time instant. The inequality constraints (1b) is the tracking constraint meant to ensure that the robot is within a distance (s_{min}, s_{max}) from the target at all times. Constraint (1c) forces the trajectory to lie in some feasible sets at all times. Of these sets, $\mathcal{C}_b$ is formed by the initial/final boundary conditions

---

[1]Non-convex optimizations have the so-called trust-region (or line-search) constraints that limit the progress that the optimizer can make towards the individual solutions between subsequent iterations.

TABLE 1. Important symbols.

$x(t), y(t), z(t), \psi(t)$	Position, heading angle of the mobile robot (quadrotor) at time t. See accompanying video
$x_{oi}(t), y_{oi}(t), z_{oi}(t)$	Position of the ith obstacle. The time stamp is relevant in case of dynamic obstacles.
u_j	A variable between 0 and 1 used to identify any point in the LOS trajectory.
$\alpha_{oi}(t, u_j), \beta_{oi}(t, u_j), d_{oi}(t, u_j)$	Variables associated with our collision and occlusion avoidance constraints. They depend on both time t and the LOS variable u_j Refer text for details.
$\alpha_r(t), \beta_r(t), d_r(t)$	Variables associated with our target tracking constraints. Refer text for details.
$x_r(t), y_r(t), z_r(t)$	Target position at time t
n, m, q	Number of obstacles in the environment, discrete samples of u_j, and planning steps, respectively

and the bounds on the positions and their derivatives up to second order (velocity and acceleration). We assume that $\mathcal{C}_b$ is convex and formed by a combination of affine equality and inequality constraints. The set $\mathcal{C}_{free}$ constrains the trajectory to be collision and occlusion-free at each time instant.

The main computation challenge of the above formulated trajectory optimization stems from the non-convexity of (1b) and the set $\mathcal{C}_{free}$. In the following, we review the solution approaches presented in existing works. We specifically aim to cover two aspects: (i) the occlusion/collision model or in other words the algebraic form of $\mathcal{C}_{free}$ and (ii) the algorithms employed to solve the target-tracking trajectory optimization problem. Note that some existing approaches like [13] simplify the problem by ignoring the effect of occlusion and focus only on collision avoidance while tracking. In our experimentation, we found that such an approach can work if the quadrotor equipped with a downward-looking camera is always allowed to hover over the target. But quadrotors with front-facing cameras need to take potential occlusion into account actively. Furthermore, occlusion avoidance also becomes crucial when the quadrotor needs to follow the target from a distance.

C. EXISTING OCCLUSION MODELS

Inspired by the GPU ray casting model [14], authors in [5] proposed a cost function to be used within their developed MPC to prevent occlusion from obstacles. Using the notation presented in Table 1, the occlusion cost of [5] is defined as
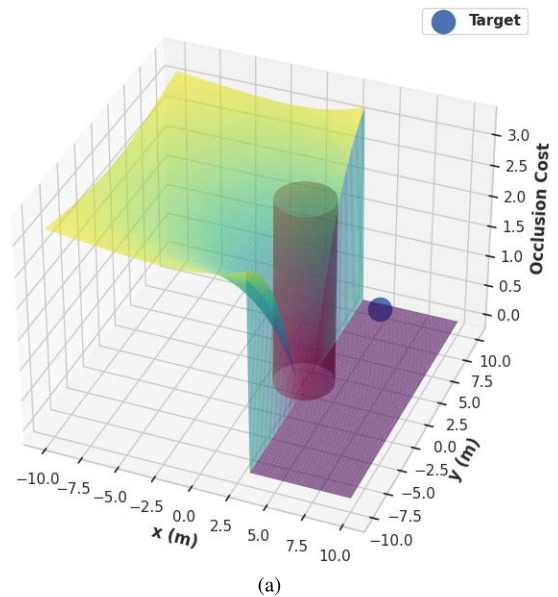
$$c_{occ} = \begin{cases} \|d_v\|, & \text{if } d_v > 0 \text{ and } p_{proj} > \mathbf{r}_{cti}^T \mathbf{r}_{cti} - 1. \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

$$d_v = \frac{p_{proj}}{\mathbf{r}_{ch}^T \mathbf{r}_{ch}}, \qquad p_{proj} = \mathbf{r}_{ch}^T \mathbf{r}_{cti}. \quad (3)$$
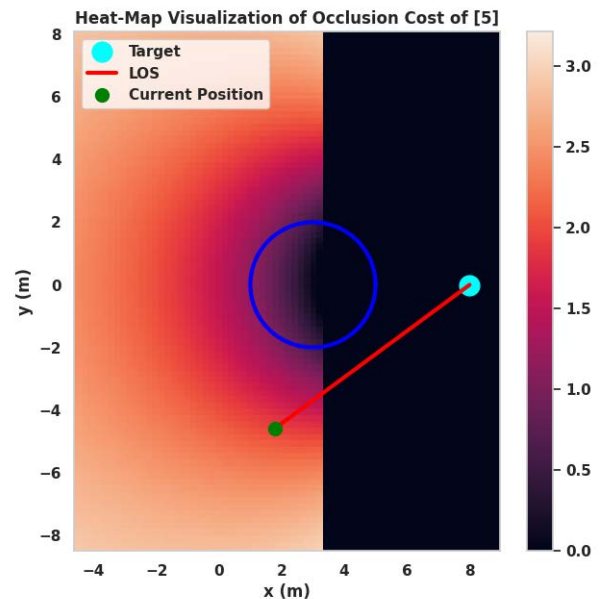
$$\mathbf{r}_{ch} = (x_r(t) - x(t), y_r(t) - y(t), z_r(t) - z(t)). \quad (4)$$

$$\mathbf{r}_{cti} = (x_{oi}(t) - x(t), y_{oi}(t) - y(t), z_{oi}(t) - z(t)). \quad (5)$$

(a)

(b)

FIGURE 1. Fig. (a): Visualization of the occlusion cost of [5] for a simple example with one obstacle. The target is a fixed point. As can be seen, the cost value increases radially from the center of the obstacle and away from the target. The cost surface does not explicitly depend on the geometry of the obstacle. Thus, some points which are occlusion-free are erroneously assigned high-cost value. This is exemplified in Fig. (b), wherein the point shown in green is occlusion-free with respect to the target (line-of-sight (red) does not intersect with the obstacle (blue)). But the occlusion-cost of this point is not zero. This conservative behavior of occlusion-cost (2) coupled with its non-smooth nature shown in Fig. (a) leads to sub-optimal behavior in the presence of multiple static and dynamic obstacles as the quadrotor is unable to find occlusion-free regions.

The variables $\mathbf{r}_{ch}, \mathbf{r}_{cti}$ are respectively the vectors to the target and the ith obstacle from the quadrotor's current position. The occlusion model of [5] first checks whether the obstacle blocks the target. That is, whether the target is behind the

obstacle and falls within the occlusion cone formed by the tangents drawn to the obstacle ellipsoid from the quadrotor center. If the conditions of occlusion are met, then the visibility cost given by (2) is invoked. Fig. 1(a) shows a visualization of the cost surface for a simple scenario with a cylindrical-shaped obstacle and a stationary target point. The following key observations can be made from Fig. 1(a). First, the occlusion cost does not depend on the geometry of the obstacle. It just increases radially from the center of the obstacle in a direction away from the target. This is an overly conservative model as a large part of the workspace that is indeed occlusion-free is assigned high-cost values (see Fig. 1(b)). As a result, (2) leads to sub-optimal performance when used within a trajectory optimizer. The second important thing to note is that the cost surface itself is non-smooth, indicating the computational difficulty of tractably optimizing over this cost function. Our experiments have also shown that it is difficult to reliably trade-off the occlusion cost of (2) with the costs stemming from collision avoidance.

An alternate occlusion model that directly works in the image space was proposed in [6]. Let $\boldsymbol{\mu}_r$ and $\boldsymbol{\mu}_{oi}$ denote the quadotor and obstacle projection on the 2D image, then occlusion can be defined in terms of euclidean separation between these image points in the following form:

$$\|\boldsymbol{\mu}_r - \boldsymbol{\mu}_{oi}\|_2^2 \geq r_{oi}^2, \tag{6}$$

where, r_{oi} is the semi-minor axis of the ellipse obtained by projecting the sphere shaped obstacles onto the image.

Many existing works derive occlusion avoidance constraints from the line of sight (LOS), i.e. a line connecting the quadrotor to the target at any given time instant [12], [15]–[17]. Intuitively, occlusion happens when the line of sight passes through an obstacle (see Fig. 3). Thus, occlusion avoidance is essentially collision avoidance for the line of sight trajectory constructed at each time instant (see Defn. 2). Authors in [12], [15], [17] use this insight to construct occlusion cost for their trajectory optimizers. Importantly, their cost representation can be derived from the generic signed distance field representation of the environment.

Our occlusion model also follows the LOS based reasoning. However, we differ from existing works [12], [15]–[17] in terms of chosen algebraic representation that presents strong computational benefits when integrated within a trajectory optimizer.

D. OPTIMIZATION ALGORITHMS

A standard approach of solving optimization of the form (1a)-(1c) is to use techniques like SQP that are readily available through software libraries like ACADO [18], NLOPT [19]. For example, this approach was followed in [5], [6]. In contrast, works like [15] use a customized form of first-order GD originally proposed for manipulation problems [20]. At the most fundamental level, both SQP and GD are based on first-order Taylor series expansion of the underlying constraint functions. Thus, the initialization of these optimizers requires us to guess where the Taylor expansion

will be computed at the first iteration. As mentioned earlier, this choice has a significant impact on the performance of the GD and SQP optimizers. A good initialization can be ensured by leveraging graph search techniques [21]. Authors in [12], [17] show how graph search technique can also be used to decompose (1a)-(1c) into hierarchy of smaller problems.

The fundamental difference between the above-cited works and our approach is that our custom optimizer never performs any linearization of the underlying non-linear tracking, occlusion/collision avoidance constraints. As a result, it does not require any sophisticated initialization for the states and controls. We always initialize the states/controls with their initial boundary conditions or nominal values in all our implementation.

E. CONTRIBUTION OVER AUTHOR's PRIOR WORK

Our occlusion/collision and tracking constraints reformulation builds on the collision avoidance constraints proposed in our prior work [22], [23]. In fact, the collision avoidance constraints of [22], [23] is a special case of the occlusion avoidance and tracking constraints proposed in the current work. Our proposed optimizer also differs from [22], [23] in the use of Lagrange multipliers. While the former follows the more conventional Alternating Direction Method of Multipliers (ADMM) template, our proposed optimizer is built on the split Bregman iteration technique [8], [9]. The prior work [23] was also restricted to offline trajectory optimization while our current work constructs a real-time MPC on top of the proposed optimizer. Finally, the current work considers a very different application of target-tracking as compared to our prior works [22], [23].

III. MATHEMATICAL PRELIMINARIES
A. MULTI-CONVEXITY AND ALTERNATING MINIMIZATION (AM)

Definition 1: Consider a function $g(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \boldsymbol{\xi}_3)$ whose argument can be split into three separate blocks of variables. The function $g(.)$ is said to be multi-convex (multi-affine) if fixing two of the variable blocks makes it convex (affine) with respect to the remaining variables. For example, fixing $(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2)$ makes g convex (affine) in $\boldsymbol{\xi}_3$ [24].

The above definition trivially extends to arbitrary number of blocks of variables.

1) ALTERNATING MINIMIZATION

Alternating (or Gauss Seidel) minimization is an efficient technique for optimizing over multi-convex functions. For example, the iteration for minimizing multi-convex function $g(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \boldsymbol{\xi}_3)$ has the following form [7]:

$$^{k+1}\boldsymbol{\xi}_1 = \arg\min_{\boldsymbol{\xi}_1} g(\boldsymbol{\xi}_1, {}^k\boldsymbol{\xi}_2, {}^k\boldsymbol{\xi}_3), \tag{7a}$$

$$^{k+1}\boldsymbol{\xi}_2 = \arg\min_{\boldsymbol{\xi}_2} g({}^{k+1}\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, {}^k\boldsymbol{\xi}_3), \tag{7b}$$

$$^{k+1}\boldsymbol{\xi}_3 = \arg\min_{\boldsymbol{\xi}_3} g({}^{k+1}\boldsymbol{\xi}_1, {}^{k+1}\boldsymbol{\xi}_2, \boldsymbol{\xi}_3). \tag{7c}$$

where k represents the iteration index. In each iteration, we optimize over each variable block in a sequence. While optimizing over a specific block, all other blocks of variables are held fixed at values obtained at either the previous iteration or that obtained at preceding steps of the same iteration.

2) A TOY EXAMPLE

Consider $g(x_1, x_2, x_3, x_4) = (x_1(x_2 + x_3)x_4 - 2)^2$. We can identify three blocks of variables $\boldsymbol{\xi}_1 = x_1$, $\boldsymbol{\xi}_2 = (x_2, x_3)$ and $\boldsymbol{\xi}_3 = x_4$ with respect to which $g(.)$ is multi-convex. Following (7a)-(7c) will reduce the minimization of $g(.)$ into a sequence of Quadratic Programs (QP).

Remark 1: Identifying the correct variables blocks with respect to which a function is multi-convex is non-trivial. Several layers of reformulation are often needed to make the multi-convex structure more explicit. Moreover, the chosen blocks have a substantial impact on the efficiency of the resulting AM-based optimizer.

B. TRAJECTORY PARAMETRIZATION

Our optimizer parametrizes the position trajectories in the following manner:

$$\begin{bmatrix} x(t_1) \\ x(t_2) \\ \cdots \\ x(t_n) \end{bmatrix} = \mathbf{P}\mathbf{c}_x, \quad \begin{bmatrix} \dot{x}(t_1) \\ \dot{x}(t_2) \\ \cdots \\ \dot{x}(t_n) \end{bmatrix} = \dot{\mathbf{P}}\mathbf{c}_x, \quad \begin{bmatrix} \ddot{x}(t_1) \\ \ddot{x}(t_2) \\ \cdots \\ \ddot{x}(t_n) \end{bmatrix} = \ddot{\mathbf{P}}\mathbf{c}_x. \quad (8)$$

where, $\mathbf{P}, \dot{\mathbf{P}}, \ddot{\mathbf{P}}$ are matrices formed with time-dependent basis functions (e.g polynomials) and $\mathbf{c}_x$ are the coefficients associated with the basis functions. Similar expressions can be written for $y(t), z(t)$ as well in terms of coefficients $\mathbf{c}_y, \mathbf{c}_z$, respectively.

IV. MAIN ALGORITHMIC RESULTS

In this section, we present our main algorithmic results: a multi-convex trajectory optimization algorithm for occlusion-free tracking and the resulting MPC. We begin by describing our main assumptions and the novel building blocks in the first few subsections.

A. ASSUMPTIONS

- We assume that the quadrotor is moving at moderate speeds for which kinematic models are sufficient. Due to the differential flatness property, the motion along x, y, z axes, and the yaw motion are all decoupled. These assumptions are standard in the existing literature on quadrotor motion planning, including those dealing specifically with target tracking [13] and drone-cinematography [25]. Finally, like [12], [25], we assume that the variation of pitch and roll angles have minimal effect on the visibility of the target. This is reasonable for most commercially available quadrotors that have field-of-view of around 90 degrees.

- We assume that the static and dynamic obstacles are modeled as ellipsoids (or cylinders in 2.5D). This is a

(a)

(b)
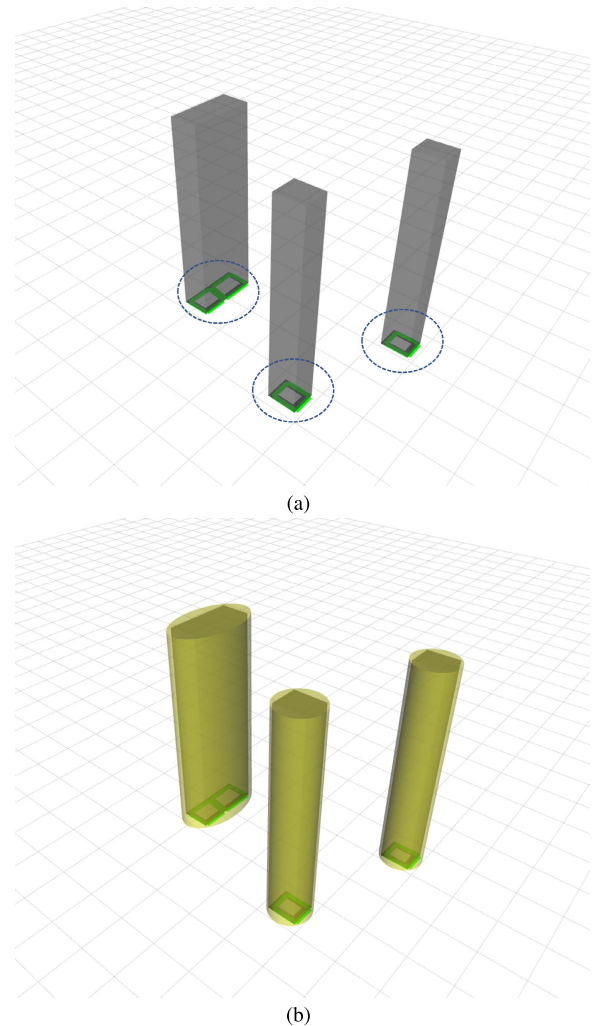
FIGURE 2. We can convert the cost map resulting from an occupancy grid representation of the environment to a set of polygons as shown in Fig. (a). These can be further converted to ellipsoidal representations as shown in Fig. (b) that can be handled efficiently by our optimizer.

fairly standard assumption for dynamic obstacles [5]. For static obstacles, we can leverage the so-called cost-map converter in Robot Operating System (ROS) [26] that can provide polygonal decomposition of occupancy maps that can be further decomposed into ellipsoidal representations (see accompanying video). An example of the aforementioned cost-map to ellipsoidal conversion is shown in Fig. 2(a)-2(b). It is important to point that many existing algorithms work with the same assumption as ours on the static obstacle form [6], [27].

- **Independent Camera Control:** Similar to works like [13], [25], we assume that the camera motion is decoupled from the quadrotors body motion. Thus, we can always align the camera of the quadrotor to the LOS vector. For a quadrotor moving in 2D with a front-facing, body-fixed camera, this boils down to aligning the yaw angle of the quadrotor at each instant to the LOS vector.

B. REFORMULATING TRACKING CONSTRAINTS

We rephrase the tracking constraints (1b) in the following manner:

$$\mathbf{f}_{tar} = \mathbf{0}, \quad s_{min} \leq d_r(t) \leq s_{max}. \tag{9}$$

$$\mathbf{f}_{tar} = \left\{ \begin{array}{l} x(t) - x_r(t) - d_r(t) \sin \beta_r(t) \cos \alpha_r(t) \\ y(t) - y_r(t) - d_r(t) \sin \beta_r(t) \sin \alpha_r(t) \\ z(t) - z_r(t) - d_r(t) \cos \beta_r(t) \end{array} \right\}. \tag{10}$$

Using the parametrization presented in (8), we can put $\mathbf{f}_{tar} = \mathbf{0}, \forall t$ in the following compact form:

$$\mathbf{A}_{tar}\mathbf{c}_x - \mathbf{b}_{tar}^x(\mathbf{d}_r, \boldsymbol{\alpha}_r, \boldsymbol{\beta}_r) = \mathbf{0}. \tag{11a}$$

$$\mathbf{A}_{tar}\mathbf{c}_y - \mathbf{b}_{tar}^y(\mathbf{d}_r, \boldsymbol{\alpha}_r, \boldsymbol{\beta}_r) = \mathbf{0}. \tag{11b}$$

$$\mathbf{A}_{tar}\mathbf{c}_z - \mathbf{b}_{tar}^z(\mathbf{d}_r, \boldsymbol{\alpha}_r, \boldsymbol{\beta}_r) = \mathbf{0}. \tag{11c}$$

$$\mathbf{b}_{tar}^x = \mathbf{x}_r + \mathbf{d}_r \cos \boldsymbol{\alpha}_r \sin \boldsymbol{\beta}_r. \tag{12a}$$

$$\mathbf{b}_{tar}^y = \mathbf{y}_r + \mathbf{d}_r \sin \boldsymbol{\alpha}_r \sin \boldsymbol{\beta}_r. \tag{12b}$$

$$\mathbf{b}_{tar}^z = \mathbf{z}_r + \mathbf{d}_r \cos \boldsymbol{\beta}_r. \tag{12c}$$

where, $\mathbf{A}_{tar} = \mathbf{P}$ and $\mathbf{x}_r, \mathbf{y}_r, \mathbf{z}_r$ are formed by stacking $x_r(t), y_r(t), z_r(t)$ at different time instants. Similar process is followed for constructing $\mathbf{d}_r, \boldsymbol{\alpha}_r, \boldsymbol{\beta}_r$ from their respective time-stamped scalar values.

C. COMBINED OCCLUSION/COLLISION AVOIDANCE CONSTRAINTS

Definition 2: The line of sight (LOS) trajectory at time t is a straight line connecting the robot's position and the target position. Mathematically, we can identify any point on this trajectory through the following equation:

$$(x_{los}, y_{los}, z_{los})(t, u_j)$$
$$= \begin{bmatrix} (1 - u_j)x(t) + u_j x_r(t) \\ (1 - u_j)y(t) + u_j y_r(t) \\ (1 - u_j)z(t) + u_j z_r(t) \end{bmatrix}, \quad \forall u_j \in [0, 1] \tag{13}$$

The different points on the LOS trajectory are identified by their respective u_j values. Using (13), we can define occlusion avoidance as collision avoidance for each point on the LOS trajectory. That is, we have

$$\frac{(x_{los}(t, u_j) - x_{oi}(t))^2}{a_i^2} + \frac{(y_{los}(t, u_j) - y_{oi}(t))^2}{b_i^2}$$
$$+ \frac{(z_{los}(t, u_j) - z_{oi}(t))^2}{c_i^2} \geq 1 \tag{14}$$

where, for ease of exposition, we have made the assumption that the ith obstacle is an axis-aligned ellipsoids. The size of the ellipsoids are given by the radius along each axis a_i, b_i, c_i which also incorporates the inflation due to the size of the quadrotor. Extension to rotated ellipsoids is straightforward and does not affect the multi-convex structure of our trajectory optimization or MPC in any manner. Inequality (14) ensures that each point on the LOS trajectory is outside of the obstacle ellipsoid.
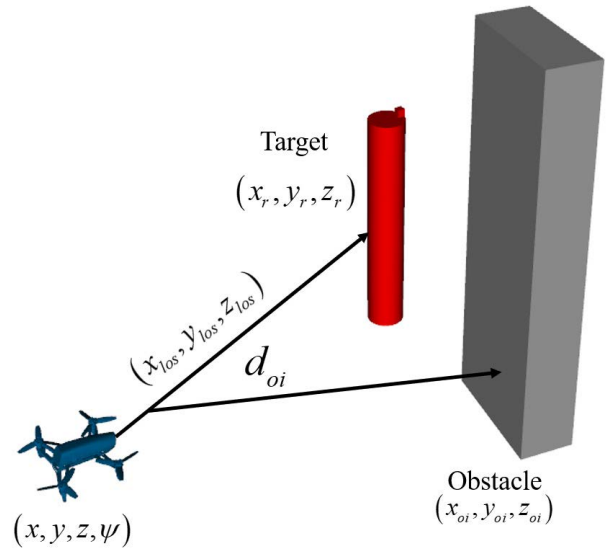
FIGURE 3. The figure explains the occlusion model used in our trajectory optimization. It is defined using the distance $d_{oi}(t, u_j)$ between the points on the LOS connecting the quadrotor with the target at any given time instant t. A $d_{oi}(t, u_j) < 0$ implies that the LOS at time t intersects with the obstacle, and consequently, the target is occluded from the quadrotor's camera at that instant.

Remark 2: Collision avoidance is contained in the occlusion-avoidance constraints. Specifically, evaluating (14) for $u_j = 0$ gives us the standard collision avoidance. Thus, (14) is enough for characterizing the set $\mathcal{C}_{free}$.

Inequalities (14) have the same form as the tracking constraints presented in (1b). Thus, we follow the same reformulation technique presented in the last subsection and rephrase (14) in the following form:

$$\mathcal{C}_{free} : \mathbf{f}_{occ} = \mathbf{0}, \quad d_{oi}(t, u_j) \geq 1, \ \forall t, u_j. \tag{15}$$

$$\mathbf{f}_{occ} = \left\{ \begin{array}{l} (x_{los} - a_i d_{oi} \sin \beta_{oi} \cos \alpha_{oi})(t, u_j) - x_{oi}(t) \\ (y_{los} - b_i d_{oi} \sin \beta_{oi} \sin \alpha_{oi})(t, u_j) - y_{oi}(t) \\ (z_{los} - c_i d_{oi} \cos \beta_{oi})(t, u_j) - z_{oi}(t) \end{array} \right\}. \tag{16}$$

Using the trajectory parametrization presented in (8), we can reduce $\mathbf{f}_{occ} = \mathbf{0}, \forall t, u_j$ to the following form:

$$\mathbf{A}_{occ}\mathbf{c}_x = \mathbf{b}_{occ}^x(\mathbf{d}_o, \boldsymbol{\alpha}_o, \boldsymbol{\beta}_o). \tag{17a}$$

$$\mathbf{A}_{occ}\mathbf{c}_y = \mathbf{b}_{occ}^y(\mathbf{d}_o, \boldsymbol{\alpha}_o, \boldsymbol{\beta}_o). \tag{17b}$$

$$\mathbf{A}_{occ}\mathbf{c}_z = \mathbf{b}_{occ}^z(\mathbf{d}_o, \boldsymbol{\alpha}_o, \boldsymbol{\beta}_o). \tag{17c}$$

where,

$$\mathbf{A}_{occ} = \begin{bmatrix} \mathbf{A}_u \\ \mathbf{A}_u \\ \vdots \\ \mathbf{A}_u \end{bmatrix}_{(\times n)}, \quad \mathbf{A}_u = \begin{bmatrix} \mathbf{P}(1 - u_1) \\ \mathbf{P}(1 - u_2) \\ \vdots \\ \mathbf{P}(1 - u_m), \end{bmatrix} \tag{18}$$

$$\mathbf{b}_{occ}^x = \widetilde{\mathbf{x}}_o + \mathbf{a}\mathbf{d}_o \cos \boldsymbol{\alpha}_o \sin \boldsymbol{\beta}_o, \tag{19a}$$

$$\widetilde{\mathbf{x}}_o = \begin{bmatrix} \widetilde{\mathbf{x}}_{o1} \\ \widetilde{\mathbf{x}}_{o2} \\ \vdots \\ \widetilde{\mathbf{x}}_{on} \end{bmatrix}, \quad \widetilde{\mathbf{x}}_{oi} = \begin{bmatrix} \mathbf{x}_{oi} + \mathbf{x}_r u_1 \\ \mathbf{x}_{oi} + \mathbf{x}_r u_2 \\ \vdots \\ \mathbf{x}_{oi} + \mathbf{x}_r u_m \end{bmatrix}$$

$$\mathbf{ad}_o \cos \boldsymbol{\alpha}_o \sin \boldsymbol{\beta}_o = \begin{bmatrix} a_1 \mathbf{d}_{o1} \cos \boldsymbol{\alpha}_{o1} \sin \boldsymbol{\beta}_{o1} \\ a_2 \mathbf{d}_{o2} \cos \boldsymbol{\alpha}_{o2} \sin \boldsymbol{\beta}_{o2} \\ \vdots \\ a_n \mathbf{d}_{on} \cos \boldsymbol{\alpha}_{on} \sin \boldsymbol{\beta}_{on} \end{bmatrix} \quad (19\text{b})$$

$$\mathbf{b}_{occ}^y = \widetilde{\mathbf{y}}_o + \mathbf{bd}_o \sin \boldsymbol{\alpha}_o \sin \boldsymbol{\beta}_o, \quad (20\text{a})$$

$$\widetilde{\mathbf{y}}_o = \begin{bmatrix} \widetilde{\mathbf{y}}_{o1} \\ \widetilde{\mathbf{y}}_{o2} \\ \vdots \\ \widetilde{\mathbf{y}}_{on} \end{bmatrix}, \quad \widetilde{\mathbf{y}}_{oi} = \begin{bmatrix} \mathbf{y}_{oi} + \mathbf{y}_r u_1 \\ \mathbf{y}_{oi} + \mathbf{y}_r u_2 \\ \vdots \\ \mathbf{y}_{oi} + \mathbf{y}_r u_m \end{bmatrix},$$

$$\mathbf{bd}_o \sin \boldsymbol{\alpha}_o \sin \boldsymbol{\beta}_o = \begin{bmatrix} b_1 \mathbf{d}_{o1} \sin \boldsymbol{\alpha}_{o1} \sin \boldsymbol{\beta}_{o1} \\ b_2 \mathbf{d}_{o2} \sin \boldsymbol{\alpha}_{o2} \sin \boldsymbol{\beta}_{o2} \\ \vdots \\ b_n \mathbf{d}_{on} \sin \boldsymbol{\alpha}_{on} \sin \boldsymbol{\beta}_{on} \end{bmatrix} \quad (20\text{b})$$

$$\mathbf{b}_{occ}^z = \widetilde{\mathbf{z}}_o + \mathbf{cd}_o \cos \boldsymbol{\beta}_o, \quad (21\text{a})$$

$$\widetilde{\mathbf{z}}_o = \begin{bmatrix} \widetilde{\mathbf{z}}_{o1} \\ \widetilde{\mathbf{z}}_{o2} \\ \vdots \\ \widetilde{\mathbf{z}}_{on} \end{bmatrix}, \quad \widetilde{\mathbf{z}}_{oi} = \begin{bmatrix} \mathbf{z}_{oi} + \mathbf{z}_r u_1 \\ \mathbf{z}_{oi} + \mathbf{z}_r u_2 \\ \vdots \\ \mathbf{z}_{oi} + \mathbf{z}_r u_m \end{bmatrix},$$

$$\mathbf{cd}_o \cos \boldsymbol{\beta}_o = \begin{bmatrix} c_1 \mathbf{d}_{o1} \cos \boldsymbol{\beta}_{o1} \\ c_2 \mathbf{d}_{o2} \cos \boldsymbol{\beta}_{o2} \\ \vdots \\ c_n \mathbf{d}_{on} \cos \boldsymbol{\beta}_{on} \end{bmatrix} \quad (21\text{b})$$

In (18), ($\times n$) signifies vertically stacking a matrix n times. For convenience, we recall that n is the number of obstacles in the environment while m is the number of discrete $u_j \in [0\ 1]$ chosen to identify points on the LOS trajectory. The vector $\boldsymbol{\alpha}_{oi}$ is formed by stacking $\alpha_{oi}(t, u_j)$ for all t and u_j. Similar construction follows for $\boldsymbol{\beta}_{oi}, \mathbf{d}_{oi}$. Furthermore, we stack different $\boldsymbol{\alpha}_{oi}, \boldsymbol{\beta}_{oi}, \mathbf{d}_{oi}$ across obstacle index i to construct $\boldsymbol{\alpha}_o, \boldsymbol{\beta}_o, \mathbf{d}_o$.

D. PROPOSED TRAJECTORY OPTIMIZATION
We are now in a position to use the reformulation of the tracking and occlusion/collision constraints to present the proposed trajectory optimization for target tracking.

$$\min \frac{1}{2} \boldsymbol{\xi}_1^T \mathbf{Q} \boldsymbol{\xi}_1. \quad (22\text{a})$$

$$\mathbf{A}\boldsymbol{\xi}_1 = \mathbf{b}(\boldsymbol{\xi}_2, \boldsymbol{\xi}_3). \quad (22\text{b})$$

$$\boldsymbol{\xi}_1 \in \mathcal{C}_{\xi_1}, \quad \boldsymbol{\xi}_3 \in \mathcal{C}_{\xi_3}. \quad (22\text{c})$$

where, $\boldsymbol{\xi}_1 = (\mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_z)$, $\boldsymbol{\xi}_2 = (\boldsymbol{\alpha}_r, \boldsymbol{\alpha}_o, \boldsymbol{\beta}_r, \boldsymbol{\beta}_o)$, $\boldsymbol{\xi}_3 = (\mathbf{d}_r, \mathbf{d}_o)$, and

$$\mathbf{A} = \begin{bmatrix} \begin{bmatrix} \mathbf{A}_{tar} \\ \mathbf{A}_{occ} \end{bmatrix} & 0 & 0 \\ 0 & \begin{bmatrix} \mathbf{A}_{tar} \\ \mathbf{A}_{occ} \end{bmatrix} & 0 \\ 0 & 0 & \begin{bmatrix} \mathbf{A}_{tar} \\ \mathbf{A}_{occ} \end{bmatrix} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}_{tar}^x \\ \mathbf{b}_{occ}^x \\ \mathbf{b}_{tar}^y \\ \mathbf{b}_{occ}^y \\ \mathbf{b}_{tar}^z \\ \mathbf{b}_{occ}^z \end{bmatrix}, \quad (23)$$

$$\mathbf{Q} = \begin{bmatrix} \ddot{\mathbf{P}}^T \ddot{\mathbf{P}} & 0 & 0 \\ 0 & \ddot{\mathbf{P}}^T \ddot{\mathbf{P}} & 0 \\ 0 & 0 & \ddot{\mathbf{P}}^T \ddot{\mathbf{P}} \end{bmatrix}. \quad (24)$$

Note that the cost function (22a) is just a matrix representation of the sum of squared acceleration presented in (1a), obtained through the trajectory parametrization (8). Similarly, the constraints (22c) are a reformulation of the feasible set $\mathcal{C}_b$. Specifically, the $\mathcal{C}_{\xi_1}$ constrains the coefficients $\mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_z$ to satisfy the boundary conditions on the trajectory and the bounds on positions, velocities, and accelerations. Mathematically, $\mathcal{C}_{\xi_1}$ is a combination of affine equality and inequality constraints. The set $\mathcal{C}_{\xi_3}$ is the feasible set for $\boldsymbol{\xi}_3$ that captures the constraints of $d_r(t)$ and $d_{oi}(t, u_j)$ (recall (9) and (15)).

The splitting of the variable blocks is done consciously to induce some useful structures in the problem and will be discussed shortly. Moreover, the fact that the cost function only depends on $\boldsymbol{\xi}_1$ will also prove useful.

E. SOLUTION BY SPLIT-BREGMAN ITERATION
Split-Bregman (SB) technique [8] solves optimization (22a)-(22c) by relaxing the non-convex equality constraints (22b) as l_2 penalties.

$$\min_{\boldsymbol{\xi}_1 \in \mathcal{C}_{\xi_1}, \boldsymbol{\xi}_3 \in \mathcal{C}_{\xi_3}} \frac{1}{2} \boldsymbol{\xi}_1^T \mathbf{Q} \boldsymbol{\xi}_1 - \langle \boldsymbol{\lambda}, \boldsymbol{\xi}_1 \rangle + \frac{\rho}{2} \left\| \mathbf{A}\boldsymbol{\xi}_1 - \mathbf{b}(\boldsymbol{\xi}_2, \boldsymbol{\xi}_3) \right\|_2^2. \quad (25)$$

The key point to note in SB technique is the introduction of a Lagrange multiplier $\boldsymbol{\lambda}$ that aims to balance the conflicting objective of minimizing the primary cost function (acceleration norm) and the residual of the equality constraints. Intuitively, $\boldsymbol{\lambda}$ appropriately weakens the effect of primary cost function to allow the optimizer to focus on reducing the constraint residual [9].

Remark 3: In (25), we have rolled all the non-convex constraints into the form of an augmented Lagrangian cost. The remaining constraints in our trajectory optimizer are simply convex bounds on the position, velocity, and acceleration. Thus, our reformulated problem is feasible by construction. This is particularly useful when dealing with infeasible initializations.

Remark 4: For a given $\boldsymbol{\xi}_2, \boldsymbol{\xi}_3$, the equality constraints (22b) is affine in $\boldsymbol{\xi}_1$ and thus the optimization (25) is a convex QP with respect to the same variable. Similarly, for a given $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2$, the optimization is convex QP in the $\boldsymbol{\xi}_3$.

Remark 5: For a given $\boldsymbol{\xi}_1, \boldsymbol{\xi}_3$, optimization (25) is solvable in closed form for $\boldsymbol{\xi}_2$.

Remark 4 is precisely the multi-convex structure foreshadowed in the earlier sections. We validate Remark 5 later in this section.

1) SOLUTION PROCESS
SB employs AM approach for minimizing (25) that reduces to the following iterates, where left superscript k again specifies

the iteration index.

$$^{k+1}\boldsymbol{\xi}_1 = \arg\min_{\boldsymbol{\xi}_1 \in \mathcal{C}_{\xi_1}} \frac{1}{2}\boldsymbol{\xi}_1^T\mathbf{Q}\boldsymbol{\xi}_1 - \langle{}^k\boldsymbol{\lambda}, \boldsymbol{\xi}_1\rangle$$
$$+ \frac{\rho}{2}\left\|\mathbf{A}\boldsymbol{\xi}_1 - \mathbf{b}({}^k\boldsymbol{\xi}_2, {}^k\boldsymbol{\xi}_3)\right\|_2^2, \tag{26}$$

$$\boldsymbol{\xi}_2 = \arg\min_{\boldsymbol{\xi}_2} \frac{\rho}{2}\left\|\mathbf{A}^{k+1}\boldsymbol{\xi}_1 - \mathbf{b}(\boldsymbol{\xi}_2, {}^k\boldsymbol{\xi}_3)\right\|_2^2, \tag{27}$$

$$\boldsymbol{\xi}_3 = \arg\min_{\boldsymbol{\xi}_3 \in \mathcal{C}_{\xi_3}} \frac{\rho}{2}\left\|\mathbf{A}^{k+1}\boldsymbol{\xi}_1 - \mathbf{b}({}^{k+1}\boldsymbol{\xi}_2, \boldsymbol{\xi}_3)\right\|_2^2. \tag{28}$$

The Lagrange multiplier is updated based on the following rule [8], [9]:

$$^{k+1}\boldsymbol{\lambda} = {}^k\boldsymbol{\lambda} - \rho\nabla_{\xi_1}\left\|\mathbf{A}^{k+1}\boldsymbol{\xi}_1 - \mathbf{b}({}^{k+1}\boldsymbol{\xi}_2, {}^{k+1}\boldsymbol{\xi}_3)\right\|_2^2. \tag{29}$$

Note, how the gradient in the second term of (29) is taken with respect to only $\boldsymbol{\xi}_1$. This is because our primary cost function $g(.)$ is independent of $\boldsymbol{\xi}_2, \boldsymbol{\xi}_3$ [8].

2) CONNECTIONS TO ADMM

The SB technique is closely related to the Alternating Direction Method of Multipliers (ADMM) [28], where a Lagrange multiplier will be associated with each of constraints (22b). In other words, the dimension of $\boldsymbol{\lambda}$ will be equal to the rows of $\mathbf{A}$. In contrast, in our use of SB technique, $\boldsymbol{\lambda}$ will have a much lower dimension equal to that of $\boldsymbol{\xi}_1$. However, the main reason for choosing SB over ADMM is because it simplifies the iterates (27)-(28) to parallel least-squares problem. We discuss this further next.

a: ANALYSIS OF STEP (26)

This optimization is a standard convex constrained QP. Since the obstacles are axis-alligned ellipsoids (or cylinders), this QP can be split into three decoupled problems for each of $\mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_z$. For rotated ellipsoids, all three coefficients would need to be computed simultaneously as the matrix $\mathbf{A}$ will no longer be block-diagonal. The state-of-the-art interior-point solvers for QP show a cubic scaling with respect to the total number of equality and inequality constraints. In (26), the number of hard inequality and equality constraints are fixed as these just stem from simple kinematic bounds and boundary conditions, respectively (set $\mathcal{C}_{\xi_1}$). The occlusion/collision avoidance enters just as a quadratic cost. Thus, the computation complexity of solving QP (26) is independent of the number of obstacles. As the number of obstacles increase, only the computation cost associated with constructing the QP, or in other words, obtaining $\mathbf{A}^T\mathbf{b}$ will change. Moreover, the growth in computation complexity can be made approximately linear with parallelization of matrix-vector product. See Section VI-E for validation.

b: ANALYSIS OF STEP (27)

In the previous step, we obtained ${}^{k+1}\boldsymbol{\xi}_1$ or in other words, $({}^{k+1}\mathbf{c}_x, {}^{k+1}\mathbf{c}_y, {}^{k+1}\mathbf{c}_z)$. Using this solution, we can compute $({}^{k+1}x(t), {}^{k+1}y(t), {}^{k+1}z(t))$ and stack them up together to

obtain the position trajectory $({}^{k+1}\mathbf{x}, {}^{k+1}\mathbf{y}, {}^{k+1}\mathbf{z})$. Now, for a given position trajectory and ${}^k\boldsymbol{\xi}_3$, the variable blocks $(\boldsymbol{\alpha}_r, \boldsymbol{\beta}_r)$ and $(\boldsymbol{\alpha}_o, \boldsymbol{\beta}_o)$ constituting $\boldsymbol{\xi}_2$ are decoupled from each other. Thus, optimization (27) decomposes into following two parallel problems:

$$\boldsymbol{\alpha}_r, \boldsymbol{\beta}_r = \arg\min_{\boldsymbol{\alpha}_r, \boldsymbol{\beta}_r} \left\|{}^{k+1}\mathbf{x} - \mathbf{x}_r - {}^k\mathbf{d}_r\cos\boldsymbol{\alpha}_r\sin\boldsymbol{\beta}_r\right\|_2^2$$
$$+ \left\|{}^{k+1}\mathbf{y} - \mathbf{y}_r - {}^k\mathbf{d}_r\sin\boldsymbol{\alpha}_r\sin\boldsymbol{\beta}_r\right\|_2^2$$
$$+ \left\|{}^{k+1}\mathbf{z} - \mathbf{z}_r - {}^k\mathbf{d}_r\cos\boldsymbol{\beta}_r\right\|_2^2. \tag{30}$$

$$\boldsymbol{\alpha}_o, \boldsymbol{\beta}_o = \arg\min_{\boldsymbol{\alpha}_o, \boldsymbol{\beta}_o} \left\|{}^{k+1}\widetilde{\mathbf{x}} - \widetilde{\mathbf{x}}_o - {}^k\mathbf{d}_o\cos\boldsymbol{\alpha}_o\sin\boldsymbol{\beta}_o\right\|_2^2$$
$$+ \left\|{}^{k+1}\widetilde{\mathbf{y}} - \widetilde{\mathbf{y}}_o - {}^k\mathbf{d}_o\sin\boldsymbol{\alpha}_o\sin\boldsymbol{\beta}_o\right\|_2^2$$
$$+ \left\|{}^{k+1}\widetilde{\mathbf{z}} - \widetilde{\mathbf{z}}_o - {}^k\mathbf{d}_o\cos\boldsymbol{\beta}_o\right\|_2^2. \tag{31}$$

where,

$$^{k+1}\widetilde{\mathbf{x}} = \mathbf{A}_{occ}{}^{k+1}\mathbf{c}_x, \quad {}^{k+1}\widetilde{\mathbf{y}} = \mathbf{A}_{occ}{}^{k+1}\mathbf{c}_y, \quad {}^{k+1}\widetilde{\mathbf{z}} = \mathbf{A}_{occ}{}^{k+1}\mathbf{c}_z. \tag{32}$$

Optimizations (30) can be further simplified by noting that $(\alpha_r(t), \beta_r(t))$ at different time instants are decoupled from each other for a given position trajectory. In other words, all the elements of $\boldsymbol{\alpha}_r, \boldsymbol{\beta}_r$ are independent of each other. A similar reasoning can be used to deduce that each of $(\alpha_{oi}(t, u_j), \beta_{oi}(t, u_j))$ are decoupled from each other and the decoupling here happens across time t, obstacle index i, and u_j. As a result, the elements of $\boldsymbol{\alpha}_o, \boldsymbol{\beta}_o$ also have no inter-dependency.

The decoupled nature of the variables highlights one more computational structure. The l_2 penalties in (30) are just several parallel projections of ${}^{k+1}\mathbf{x} - \mathbf{x}_r$, ${}^{k+1}\mathbf{y} - \mathbf{y}_r$ and ${}^{k+1}\mathbf{z} - \mathbf{z}_r$ onto a sphere centered at origin with radius ${}^k\mathbf{d}_r(t)$. Thus, we can represent the solution as the following closed-form expression.

$$^{k+1}\boldsymbol{\alpha}_r = \arctan 2({}^{k+1}\mathbf{y} - {}^{k+1}\mathbf{y}_r, {}^{k+1}\mathbf{x} - {}^{k+1}\mathbf{x}_r),$$
$$^{k+1}\boldsymbol{\beta}_r(t) = \arctan 2(\frac{{}^{k+1}\mathbf{x}(t) - {}^{k+1}\mathbf{x}_r}{\cos{}^{k+1}\boldsymbol{\alpha}_r}, {}^{k+1}\mathbf{z} - {}^{k+1}\mathbf{z}_r). \tag{33}$$

Following a similar process, we can derive the solution of (31) in the following form.

$$^{k+1}\boldsymbol{\alpha}_o = \arctan 2({}^{k+1}\widetilde{\mathbf{y}} - {}^{k+1}\mathbf{y}_o, {}^{k+1}\widetilde{\mathbf{x}} - \widetilde{\mathbf{x}}_o),$$
$$^{k+1}\boldsymbol{\beta}_o(t) = \arctan 2(\frac{{}^{k+1}\widetilde{\mathbf{x}} - \widetilde{\mathbf{x}}_o}{\cos{}^{k+1}\boldsymbol{\alpha}_o}, {}^{k+1}\widetilde{\mathbf{z}} - \widetilde{\mathbf{z}}_o). \tag{34}$$

c: ANALYSIS OF STEP (28)

For a given position trajectory at the $k+1$ iteration, $\mathbf{d}_r$ and $\mathbf{d}_o$ constituting $\boldsymbol{\xi}_3$ are decoupled from each other and thus (28) decomposes into following parallel sub-problems:

$$^{k+1}\mathbf{d}_r$$
$$= \arg\min_{s_{min} \leq \mathbf{d}_r \leq s_{max}} \left\|{}^{k+1}\mathbf{x} - \mathbf{x}_r - \mathbf{d}_r\cos{}^{k+1}\boldsymbol{\alpha}_r\sin{}^{k+1}\boldsymbol{\beta}_r\right\|_2^2$$

$$+ \left\| {}^{k+1}\mathbf{y} - \mathbf{y}_r - \mathbf{d}_r \sin {}^{k+1}\boldsymbol{\alpha}_r \sin {}^{k+1}\boldsymbol{\beta}_r \right\|_2^2$$

$$+ \left\| {}^{k+1}\mathbf{z} - \mathbf{z}_r - \mathbf{d}_r \cos {}^{k+1}\boldsymbol{\beta}_r \right\|_2^2. \tag{35}$$

$${}^{k+1}\mathbf{d}_o$$

$$= \arg\min_{\mathbf{d}_o \geq 1} \left\| {}^{k+1}\widetilde{\mathbf{x}} - \widetilde{\mathbf{x}}_o - \mathbf{d}_o \cos {}^{k+1}\boldsymbol{\alpha}_o \sin {}^{k+1}\boldsymbol{\beta}_o \right\|_2^2$$

$$+ \left\| {}^{k+1}\widetilde{\mathbf{y}} - \widetilde{\mathbf{y}}_o - \mathbf{d}_o \sin {}^{k+1}\boldsymbol{\alpha}_o \sin {}^{k+1}\boldsymbol{\beta}_o \right\|_2^2$$

$$+ \left\| {}^{k+1}\widetilde{\mathbf{z}} - \widetilde{\mathbf{z}}_o - \mathbf{d}_o \cos {}^{k+1}\boldsymbol{\beta}_o \right\|_2^2. \tag{36}$$

Both of the optimizations (35) and (36) are convex QPs with simple box constraints. We note that different time instant values of $d_r(t)$ are independent of each other when the position trajectory is fixed. In other words, the elements of $\mathbf{d}_r$ do not have any inter-dependency. A similar decoupling exists across the elements of $\mathbf{d}_o$. Thus, optimization (35) gets simplified to q parallel single-variable QPs, where we again recall q to be the number of planning steps. Similarly, (36) reduces to $m * n * q$ parallel single-variable QPs. Each of these single variables QPs is solvable in symbolic form. More precisely, we can compute the unconstrained solution and then ensure the constraints by simply clipping the solution to the min/max values.

Remark 6: The evaluation of closed form symbolic solutions of (30)-(31) and (35)-(36) are linear complexity operations that do not require any matrix-factorization/inverses or matrix-vector products.

F. INTERPRETATION OF THE OCCLUSION COST IN OUR FORMULATION

Consider the following residual extracted from the norm on the r.h.s of (29).

$$\begin{aligned} {}^{k+1}r_{occ} &= \left\| \mathbf{A}_{occ} {}^{k+1}\boldsymbol{\xi}_1 - \mathbf{b}_{occ}({}^{k+1}\boldsymbol{\xi}_2, {}^{k+1}\boldsymbol{\xi}_3) \right\|_2^2 \\ &= \left\| {}^{k+1}\widetilde{\mathbf{x}} - \widetilde{\mathbf{x}}_o - {}^{k+1}\mathbf{d}_o \cos {}^{k+1}\boldsymbol{\alpha}_o \sin {}^{k+1}\boldsymbol{\beta}_o \right\|_2^2 \\ &\quad + \left\| {}^{k+1}\widetilde{\mathbf{y}} - \widetilde{\mathbf{y}}_o - {}^{k+1}\mathbf{d}_o \sin {}^{k+1}\boldsymbol{\alpha}_o \sin {}^{k+1}\boldsymbol{\beta}_o \right\|_2^2 \\ &\quad + \left\| {}^{k+1}\widetilde{\mathbf{z}} - \widetilde{\mathbf{z}}_o - {}^{k+1}\mathbf{d}_o \cos {}^{k+1}\boldsymbol{\beta}_o \right\|_2^2. \end{aligned} \tag{37}$$

The variable ${}^{k+1}r_{occ}$ shows how much of the occlusion-constraint is violated after the $k + 1$ iteration of our optimizer. Alternately, it can also be viewed as the analogy of occlusion cost in our formulation. A visualization of this cost surface at any particular time instant is shown in Fig. 4(a) and its heat map representation is shown in Fig. 4(b). For the construction of the cost surface, we sampled many different $({}^{k+1}\mathbf{x}, {}^{k+1}\mathbf{y})$ and computed the corresponding optimal ${}^{k+1}\boldsymbol{\alpha}_o, {}^{k+1}\boldsymbol{\beta}_o, {}^{k+1}\mathbf{d}_o$ from (34) and (36), respectively.

As can be seen, our occlusion cost perfectly captures the effect of obstacle geometry. The cost value peaks at the obstacle center and then smoothly drops off to zero as we move away from it. Our occlusion cost also correctly assigns zero cost to any point from where the line-of-sight to the

target is unobstructed from the obstacle. Both the smooth cost surface and accurate modeling of occlusion are in sharp contrast to the cost surface shown in Fig. 1(a).

G. MPC THROUGH REAL-TIME ITERATION

Our MPC implementation involves solving the trajectory optimization (22a)-(22c) in a receding horizon manner. That is, at each step, we obtain the full trajectory but only execute a small portion of it. In practice, we average a small portion of the velocity trajectory to obtain a piece-wise constant approximation of the time-varying profile which is then commanded to the quadrotor. Similar process has been followed in many works such as [29].

A fundamental challenge in MPC is to solve the underlying trajectory optimization in real-time. In practice, it is not possible to solve the optimization till convergence. As a result, the so-called Real-Time Iterations scheme [10] is adopted where only an approximate solution is obtained by running the optimization for a few iterations. For example, in non-linear MPC community, it is common to perform only one iteration of the sequential quadratic programming or Gauss-Newton method [10], [30]. The obtained solution is given to the robot and for the next MPC step, the trajectory optimization is warm-started from the previous solution. The process of warm-starting is the key and simulates an online approach towards solving an optimization problem.

V. VALIDATION

The objective of this section is two-fold. First, we empirically validate the convergence of our optimizer. Second, we compare it with an alternate approach where we solve the original formulation (1a)-(1c) using the state-of-the-art convex-concave procedure (CCP) [4]. The occlusion constraints were modeled through the standard quadratic inequality form presented in (14). Our comparison with CCP baseline is meant to establish the effectiveness of our several layers of reformulations and the resulting AM optimizer presented in the previous section.

Running Example: To empirically validate the convergence of our optimizer, we consider a simple problem set-up, where a quadrotor needs to move between a start and a goal position while keeping the LOS to a static target occlusion-free. Thus, in this setting, we do not have the tracking constraints. We also ignore the velocity and acceleration bounds.

A. CONVERGENCE VALIDATION

Fig. 5(a)-5(f) summarizes the results for different initial guesses. In Fig. 5(a), 5(c), the initial-guess trajectory violates both collision and occlusion avoidance constraints. In Fig. 5(e), a trivial straight line from the start position to the target was used as the initial guess. Following core observations are worth pointing out. First, our optimizer is robust to initial guess and we have observed that on an average 40-50 iterations are enough to obtain occlusion residual in the order of 10^{-3}. Second, if we ignore the occlusion constraints, the trivial straight line trajectory from start to

(a)

(b)
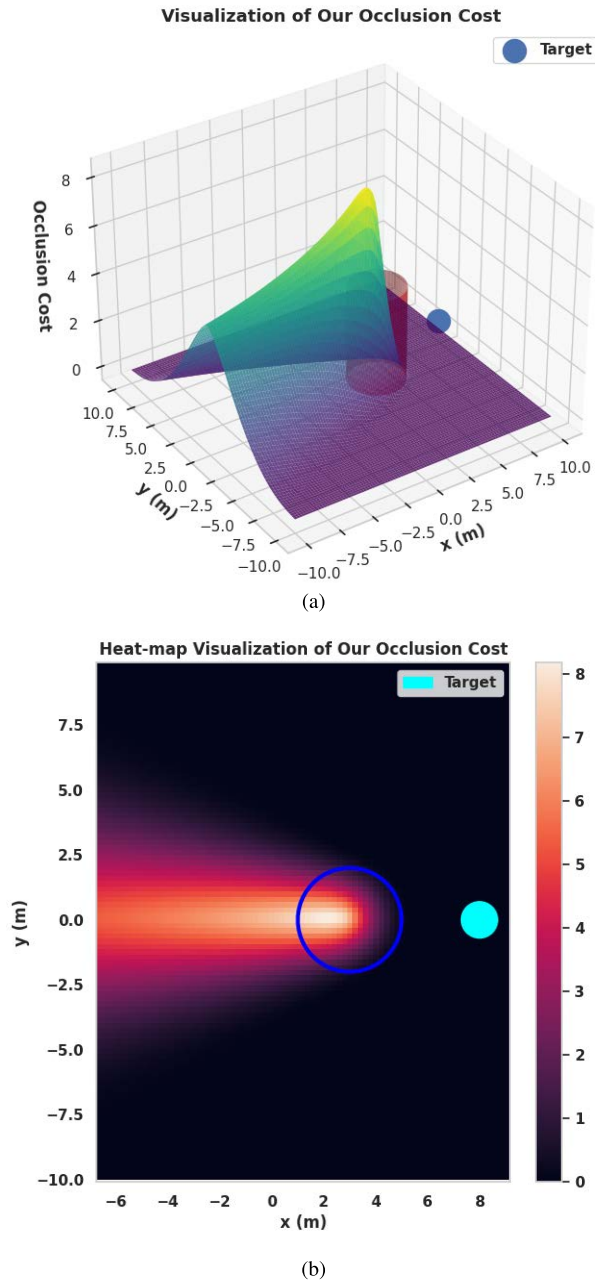
FIGURE 4. Fig. (a) shows the visualization of the our occlusion cost given by (37) while Fig. (b) shows its heat map representation. As can be seen, the cost-surface peaks at the center of the obstacle and gradually reduces to zero as we move away from it. Importantly, the exact geometry of the obstacle is reflected in our occlusion cost. This is in sharp contrast to the occlusion model of [5] shown in Fig. 1(a)-1(b).

goal satisfies the collision avoidance constraints. Thus, even this simple problem set-up shows how occlusion avoidance in tracking applications often conflicts with the collision avoidance requirement.

B. COMPARISONS WITH CCP
A CCP [4] approach for solving the point to point navigation discussed in the previous sub-section will amount to solving

the following optimization problem:

$$\text{minimize } (1a), \quad \text{subject to } \mathbf{F}\boldsymbol{\xi}_1 \leq \mathbf{g} \qquad (38)$$

where $\mathbf{F}$, $\mathbf{g}$ are obtained by linearizing the quadratic occlusion constraints (14). Fig. 6(a)-6(b) show one qualitative result obtained with CCP. For comparison, we also show the final trajectory obtained with our optimizer. From qualitative standpoint, the trajectories resulting from both the optimizers seem similar. The CCP optimizer usually obtains a feasible solution within 2 to 3 iterations and a couple of more iterations are required to decrease the cost value. Thus the number of iterations required by CCP is substantially less than our optimizer (40-50). However, each iteration of CCP is more expensive and thus, as shown in Fig. 6(c), our optimizer outperforms CCP's computation time by more than two orders of magnitude. The timings presented in Fig. 6(c) are averaged values over 10 different problem instances and correspond to the Python implementations of both of the optimizers on an i7-8600 laptop with 32 GB RAM.

The difference in computation time can be co-related to the structural contrast between our optimizer and CCP. The number of constraints stemming from occlusion avoidance in the latter is equal to the number of rows in $\mathbf{F}$ in (38), which in turn is equal to $n * m * q$ (planning horizon times number of obstacles times discretization of u_j, recall Table 1). We ran the CCP optimizer with $n = 2, m = 20, q = 100$ resulting in a total of 2000 inequality constraints. There is also an additional overhead of obtaining the matrix $\mathbf{F}$ and vector $\mathbf{g}$ at each iteration of CCP based on the refined linearization of (14) around the current solution. We note that in some experiments, a denser discretization of u_j could be needed which will only further increase the computational burden of CCP.

In contrast, our optimizer used $n = 2, m = 100, q = 100$ and yet had a substantially lower computation time. This is because our optimizer handles occlusion constraints by augmenting it as an l_2 norm penalty in steps (26)-(27) and as a combination of both penalty and inequality constraints in step (28). In (26), occlusion constraints are reformulated as a quadratic cost (third term) and since there are no velocity/acceleration bounds in the running example of this sub-section, this step is simply an equality constrained QP with a closed form solution. The step (27) has a symbolic solution (34) whose evaluation do not require any computation of matrix-vector/matrix-matrix products or matrix factorization. The step (28) is a convex constrained QP with dimension equal to $n*m*q$. However, the massive distributive nature of this step allows us to again obtain a closed form symbolic solution with no requirement of any sort of matrix operation (Recall (36) and discussions around it).

The CCP optimizer marginally outperforms ours in terms of achieved optimal cost (Fig. 6(d)). This is because our choice of augmenting occlusion constraints in the cost function leads to a conflict between occlusion and the primary objective of minimizing the acceleration norm.
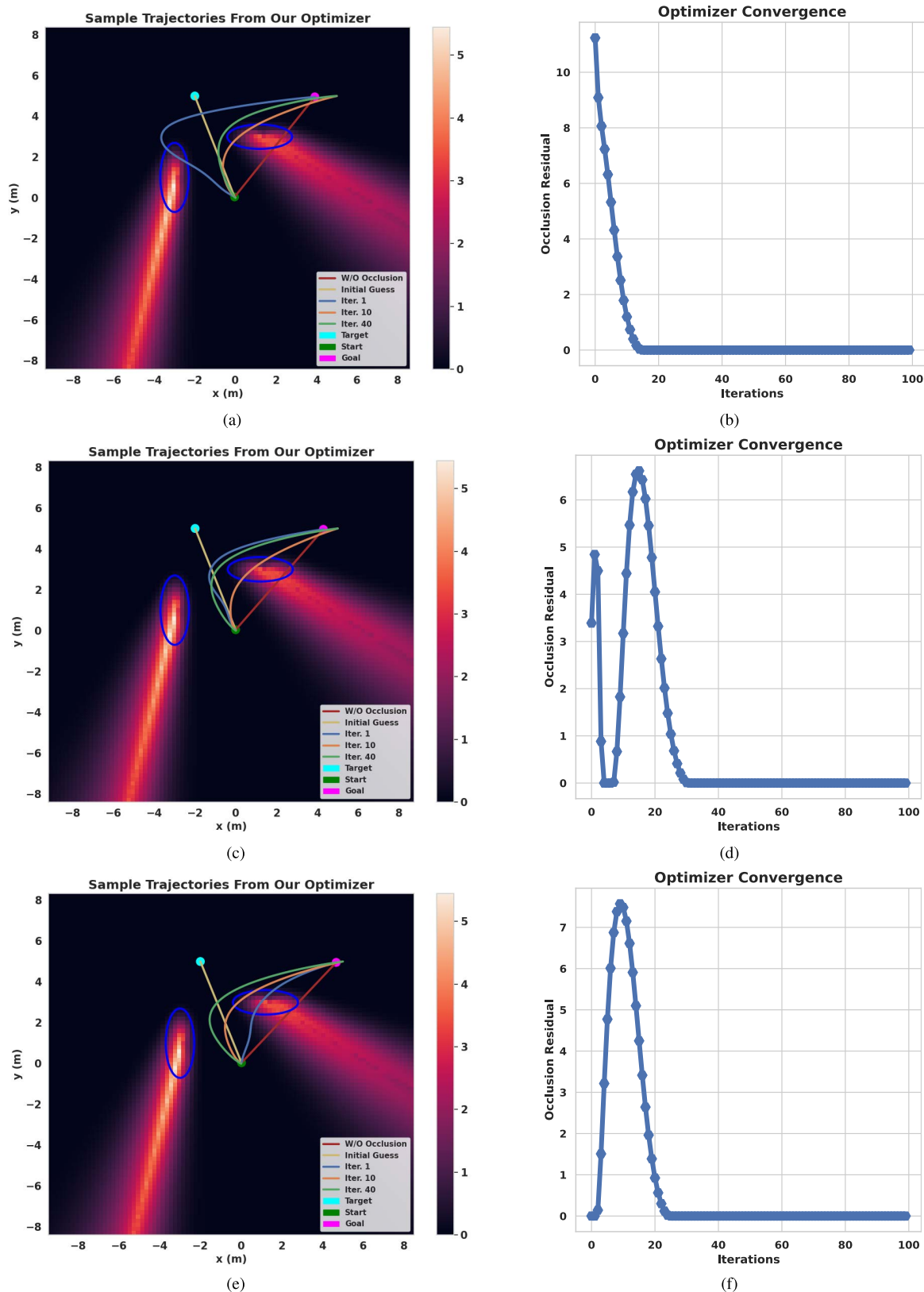
FIGURE 5. Figures show the convergence of our optimizer for different initial guesses for a problem set-up where the quadrotor needs to move between a start and a goal position while keeping the LOS to the target (cyan) occlusion-free at all time instants. The obstacles are shown in blue and have been inflated with the size of the quadrotor. The heat map shows the occlusion cost in different regions of the workspace. Note that if we ignore the occlusion constraints, the collision avoidance constraints are trivially satisfied by a straight line from start to goal (red line). On average, the occlusion residuals converge to zero in around 40-50 iterations.

(a)

(b)

(c)

(d)
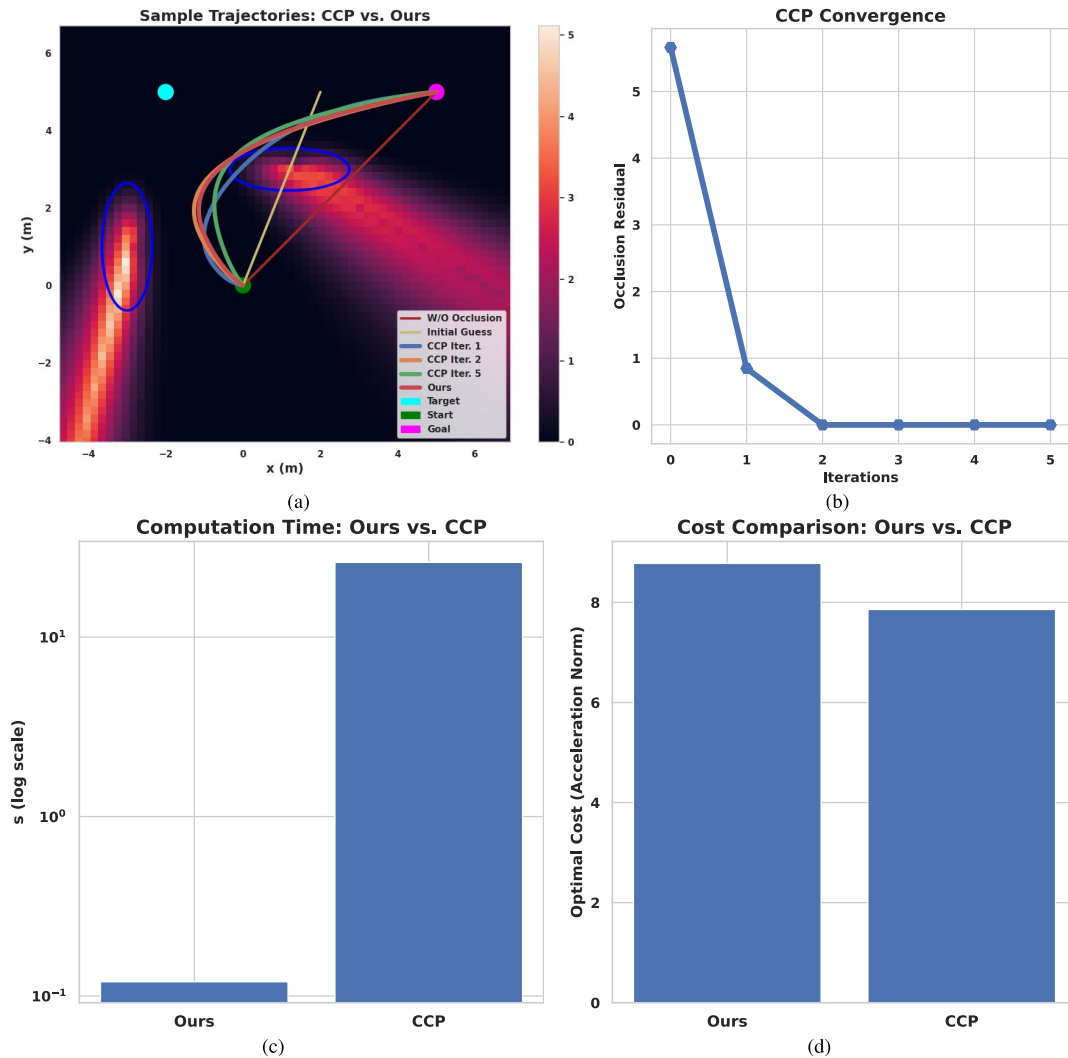
FIGURE 6. Fig. (a) shows the sample trajectories obtained from the CCP [4] optimizer for the point to point navigation task presented in Fig. 5(a). The trajectories obtained after convergence are qualitiatively very similar to that obtained with our optimizer. As shown in Fig. (b), the CCP optimizer is able to obtain a feasible solution in 3 iterations. However, each iteration of CCP is computationally very expensive.

C. TARGET-TRACKING EXAMPLE

We now present a simple example to demonstrate the convergence of our optimizer on target-tracking example. The main objective is to show that if the target trajectory is known completely, then the residual of our tracking constraints goes to zero. Fig. 7(a) shows the output of our optimizer. The initial guess for this example was taken to be the target trajectory itself. The minimum and maximum tracking distance was kept as $1m$ and $3m$, respectively. Fig. 7(b) shows the convergence of occlusion and tracking residuals. Once again, we see that around 50 iterations are enough to obtain a low residual solution. We would like to point out that if the target trajectory is not known, then the convergence of the tracking constraints is not guaranteed. In such a case, the aim should be to satisfy the tracking requirement as best as possible. As long as occlusion is avoided, the target can always be kept in the

field of view, albeit at a distance outside the minimum and maximum thresholds.

VI. BENCH-MARKING

The objective of this section is to benchmark the MPC built on top of our optimizer with existing state-of-the-art approaches. Our entire multi-convex MPC and simulation pipeline is publicly available at https://bit.ly/3fLI6zi. Our simulation pipeline is shown in Fig. 8. As presented in Section IV, our main focus is on designing the optimizer/high-level MPC that provides a feasible feed-forward trajectory to the lower level controller. We used [31] as our low-level controller that is guaranteed to track any smooth, differentiable trajectory respecting the kinematic bounds. The trajectories resulting from our optimizer are guaranteed to fulfill all these conditions. The smoothness stems from the polynomial nature

of the trajectories while the kinematic bounds are included as convex constraints in the optimizer and thus they are guaranteed to be satisfied at each control cycle.

A. IMPLEMENTATION DETAILS

For benchmarking, we implemented trajectory optimization (22a)-(22c) and the resulting MPC in C++ using Eigen [32] as our linear algebra back-end. We integrated our MPC with Gazebo physics engine in ROS [11] to perform real-time high fidelity simulations. The simulator provides state feedback of the quadrotor and the obstacles at 100 Hz. However, our MPC can run potentially at up to 250Hz depending upon the number of obstacles in the environment and number of iterations of the optimizer. Due to warm-starting of the MPC from the solution obtained in the previous control cycle, just one iteration of the optimizer proved sufficient in almost all the benchmarks. We designed the target trajectory by first teleoperating it over the workspace and then replaying the recorded trajectory during run-time. The teleoperation setup allowed us to generate complex trajectories for the target.

Our MPC implementation had terminal constraints that forced the final velocity and acceleration of the quadrotor to be zero. We also tried setting the terminal values to the predicted velocity and acceleration of the target. However, this led to poor performance. This is unsurprising as long-term trajectory of the target is not known in advance and the linear predictions used by our MPC are only rough estimates of the true values. The prediction horizon of our MPC was set to 10s.

B. STATE-OF-THE-ART AND COMPARISON METRICS

A core objective of this section is to establish the superiority of our multi-convex MPC over two state-of-the-art approaches: the AutoChaser algorithm proposed in [12] and MPC algorithm of [5]. The former is designed for tracking amongst only static obstacles, while the latter also considers dynamic obstacles. We use the publicly available author implementation of [12]. This implementation had a prediction horizon of 7s. We tried increasing this value but it increased the computation time and the resulting control delay led to poor tracking performance.

To the best of our knowledge, no open-source implementation of [5] has been released by the authors. Thus, we built our own implementation following closely the mathematical formulation presented in the cited work. We used the state-of-the-art library ACADO [18] to prototype the trajectory optimization and the resulting MPC of [5]. We performed extensive tuning between collision avoidance, occlusion, and acceleration cost to get the best performance. We also release our implementation of [5] for verification.

We use the following metrics to benchmark our MPC with state-of-the-art approaches:

1) VISIBILITY SCORE

Following [12], we define visibility score as the smallest distance between static/dynamic obstacles and the LOS trajectory. Clearly, a distance below zero will imply that the line of the sight is blocked by the obstacles and the target is occluded from the quadrotor's camera. The visibility score is a proxy for the satisfaction of occlusion constraints and thus, a value less than zero means the occlusion constraints are not satisfied. Note that exact value of visibility score is not important as long as it is above zero.

2) ACCELERATION NORM

This metric measures how rapidly the quadrotor needs to change its direction and speed to ensure collision/occlusion-free tracking. In other words, the acceleration norm quantifies trajectory smoothness and has been extensively used to benchmark trajectory optimization algorithms. Furthermore, having a smooth trajectory is also critical for applications like drone cinematography to ensure that the images taken of the target are of high-quality [15].

3) COMPUTATION TIME

This metric measures the time taken per MPC step and quantifies the real-time applicability of the algorithm.

C. STATIC ENVIRONMENT TESTS
1) BENCHMARK

Fig. 9(a) shows our test environment with static obstacles (grey rectangles), wherein the moving target is shown as a red colored cylinder. The magenta colored line shows the line of sight connecting the quadrotor and the target. We created different benchmarks in this environment by varying the trajectory of the target. Notably, the target's trajectories were designed in a way to ensure sharp turns near obstacles to thoroughly test the occlusion avoidance capabilities of our multi-convex MPC and the existing approach [12]. Fig. 9(a)-9(b) show the qualitative results obtained across two benchmarks. Note, Fig. 9(a) shows only the first few seconds of a long trajectory spread out over several minutes. As can be seen, our multi-convex MPC can leverage fast re-planning to counter the target's motion behind the obstacles and maintain occlusion-free tracking. In contrast, the AutoChaser [12] is not responsive enough and thus at times, the target is completely occluded from its field of view. Fig. 10(a)-10(b) further quantify the occlusion avoidance in terms of visibility score across eight different trajectories. The exact visibility score is not important and Fig. 10(a)-10(b) aims to highlight the parts where the visibility score goes to zero. This in turn signifies the violation of the occlusion avoidance constraints. Our MPC maintains perfect tracking while in sharp contrast, the visibility score for AutoChaser [12] goes below zero at several time instants.

Fig. 10(c) compares the acceleration statistics of our MPC with AutoChaser [12] across all benchmarks. The median value of linear acceleration employed by our MPC is $0.26 m/s^2$. This is marginally lower than that employed by AutoChaser algorithm [12] which stands at $0.30 m/s^2$. However, the difference is more stark if we compare the maximum acceleration values. The maximum acceleration employed
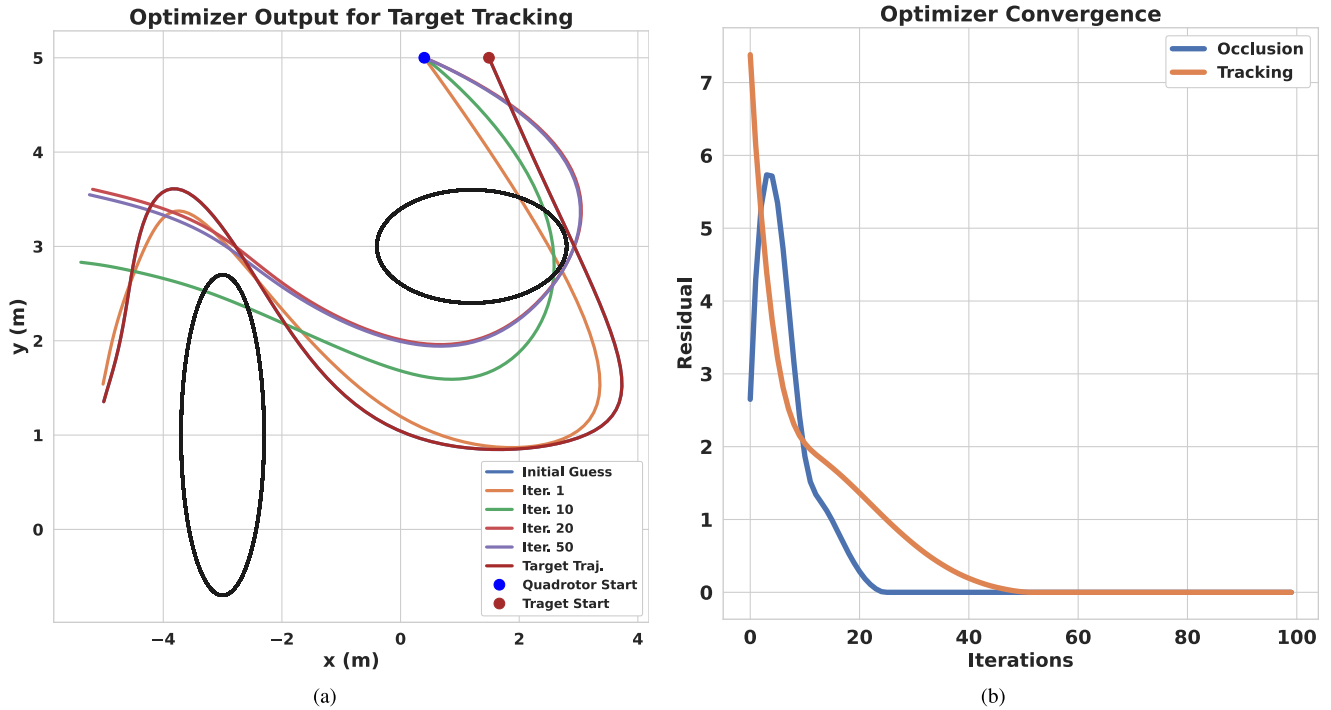
FIGURE 7. Fig. (a): Sample trajectories from our optimizer for target tracking application. The obstacles are shown as black ellipses. Since the target is moving, the occlusion cost surface would vary over time. Thus, for this example, we do not overlay the trajectories on the heat-map of the occlusion cost. Fig. (b) shows the convergence of the occlusion and tracking constraints. The quadrotor starts at the position shown in blue which is different from the start position of the target trajectory (shown in brown).
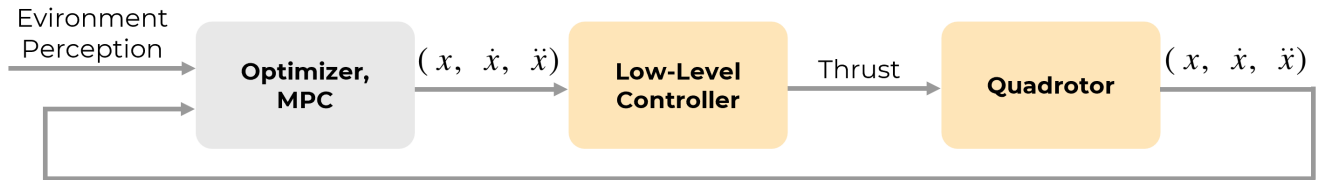
FIGURE 8. The simulation pipeline. Our main focus is on designing the high level optimizer/MPC that provides a feasible feed-forward trajectory to an off-the-shelf low-level controller [31]. Our optimizer is guaranteed to provide a feasible trajectory to the low-level controller. To maintain clarity, we only show the x component of state-feedback but the remaining y, z components are indeed integral parts of the state feedback.

by our MPC across all benchmarks is $0.95 m/s^2$. This is 30% lower than that used by the AutoChaser ($1.35 m/s^2$). We observed in the simulation that the higher acceleration of the latter stems from the fact that it has a significant control delay due to its higher computation time and thus it often needs to accelerate sharply to maintain the visibility of the target. We highlight this behavior in the accompanying simulation video as well.

The angular acceleration employed by our MPC is higher than that of AutoChaser [12].[2] Our MPC's median angular acceleration value is $0.09 rad/s^2$ which is 2 times higher than $0.047 rad/s^2$ observed for the AutoChaser [12]. The factor of difference between the maximum values is also approximately the same. This pattern in angular acceleration is due to the fact that our MPC aggressively tries to keep the target
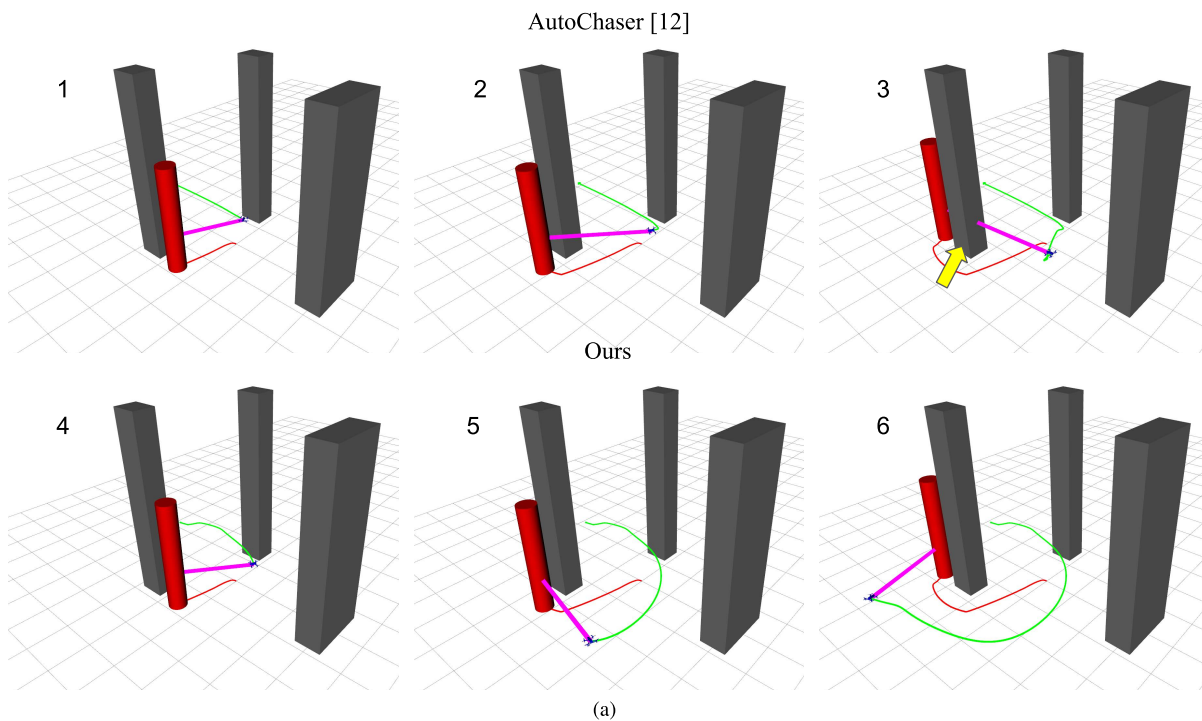
at the center of the field of view. In contrast, AutoChaser adopts a more relaxed approach in maintaining the orientation towards the target.
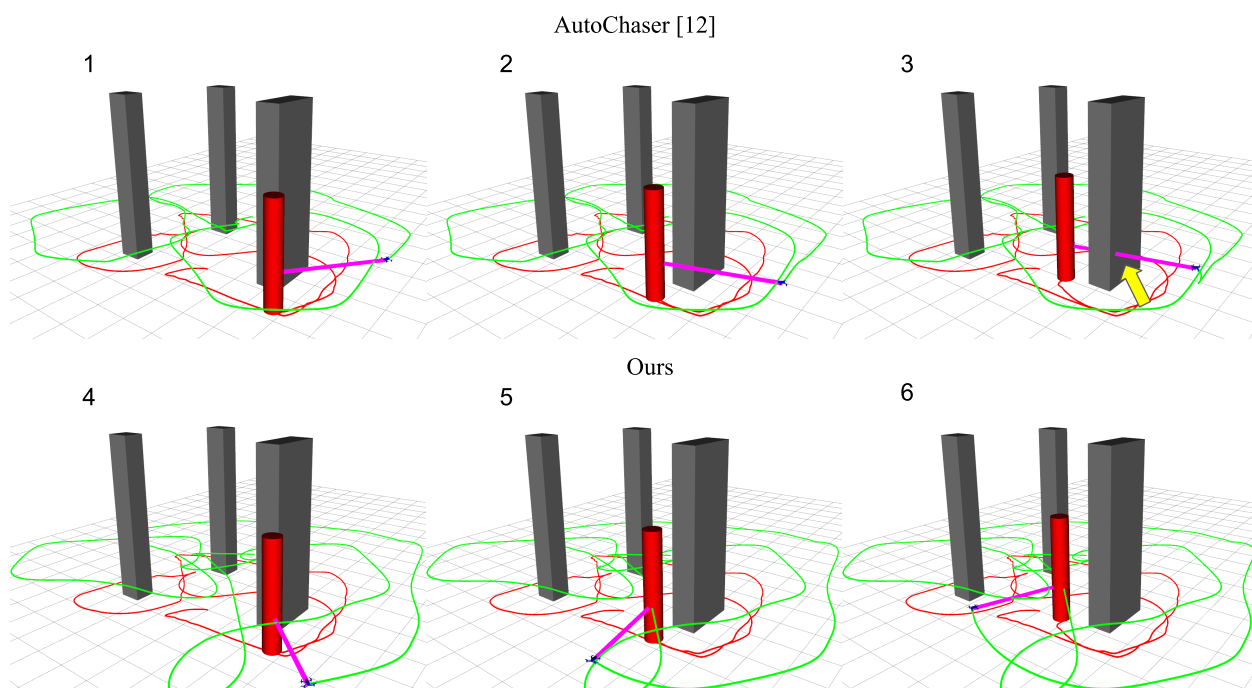
Fig. 10(d) compares the computation time statistics observed across all benchmarks for our MPC and AutoChaser algorithm [12]. The median computation time for our MPC was $0.006 s$ which was more than one order of magnitude smaller than $0.08 s$ observed for the AutoChaser. The worst-case timing of AutoChaser was around $0.11 s$ while our MPC showed minimal variation across all MPC iterations and always hovered around the median value.

2) INSIGHT INTO PERFORMANCE GAIN

The performance gain of our MPC over [12] can be understood in the following manner. As shown in the accompanying video, [12] observes few instants of target motion and then does a complex prediction of its future trajectory. This process has a significant overhead. While the prediction is

---

[2]As mentioned earlier, we independently control the yaw angle of the quadrotor to always align with the LOS vector. Thus accelerations are computed by second-order finite difference of the yaw angle at subsequent time instants.
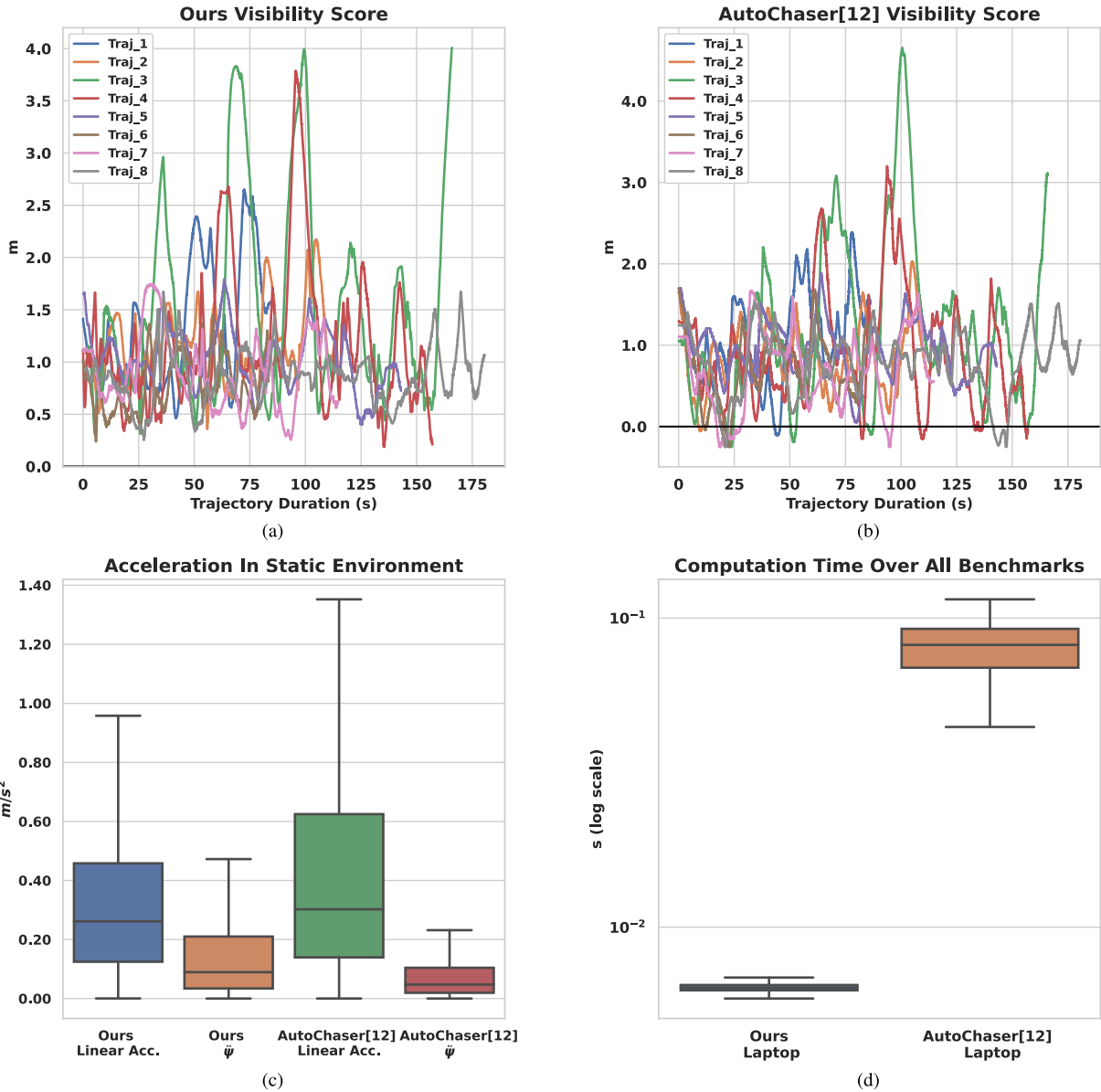
FIGURE 9. Fig. (a)-(b) show the qualitative comparison between the tracking performance obtained with our multi-convex MPC and the state-of-the-art algorithm AutoChaser [12]. While AutoChaser gets occluded at multiple instances (yellow arrow), our MPC can react fast to abrupt changes of the target's motion near the obstacles and avoid occlusions. It should be noted that AutoChaser has been provided with information about the global trajectory of the target in the form of intermediate waypoints. In contrast, our MPC only needs information about the instantaneous position and velocity of the target.

being computed, the quadrotor is almost static and once the prediction is completed, it tries to accelerate and catch up with the target. Thus, if the target is moving with moderately high velocity and making very sharp turns around the obstacle, by the time the prediction computation is done, the target has already moved to a difficult position from where the

FIGURE 10. Figures show the quantitative comparison between our multi-convex MPC and the state-of-the-art algorithm AutoChaser [12]. Our MPC outperforms with a higher visibility score across all benchmarks (a, b). A visibility score of below zero implies occlusion or alternately non-satisfaction of occlusion avoidance constraints. Note that the specific values of visibility score are not important as long as it is above zero. Thus, the main point to note from (a, b) is that unlike [12], our visibility score never goes below zero. Our MPC also achieves a smoother acceleration profile (c) and shorter computation times (d).

occlusion cannot be avoided. Finally, [12] replans at a much slower rate which proves detrimental to reliable tracking in cluttered environments. In contrast to [12], our MPC uses just a linear prediction but updates the motion plan almost ten times faster.

D. DYNAMIC ENVIRONMENT TESTS
1) BENCHMARK
Fig. 11(a)-11(b) show our test environments with dynamic obstacles (grey cylinders). For these test-cases, we assumed that the target is stationary and thus the quadrotor needs to change its position only to avoid occlusion from the dynamic

obstacles. However, as it moves, collision avoidance also comes into play. Note, that if we disregard the occlusion, then collision avoidance in these test-cases becomes trivial as the quadrotor can just hover at its initial position. We chose a stationary target because it was easier to benchmark our MPC with [5] in this setting. To elaborate further, the target trajectory also needs to avoid collisions with the dynamic obstacles for a meaningful comparison. Thus, it is extremely difficult to recreate the exact same target trajectory within a physics engine for a fair evaluation of our MPC and [5]. We created different benchamrks in this test case by varying the trajectories of the dynamic obstacles. Fig. 11(a)-11(b) show the

(a)

(b)
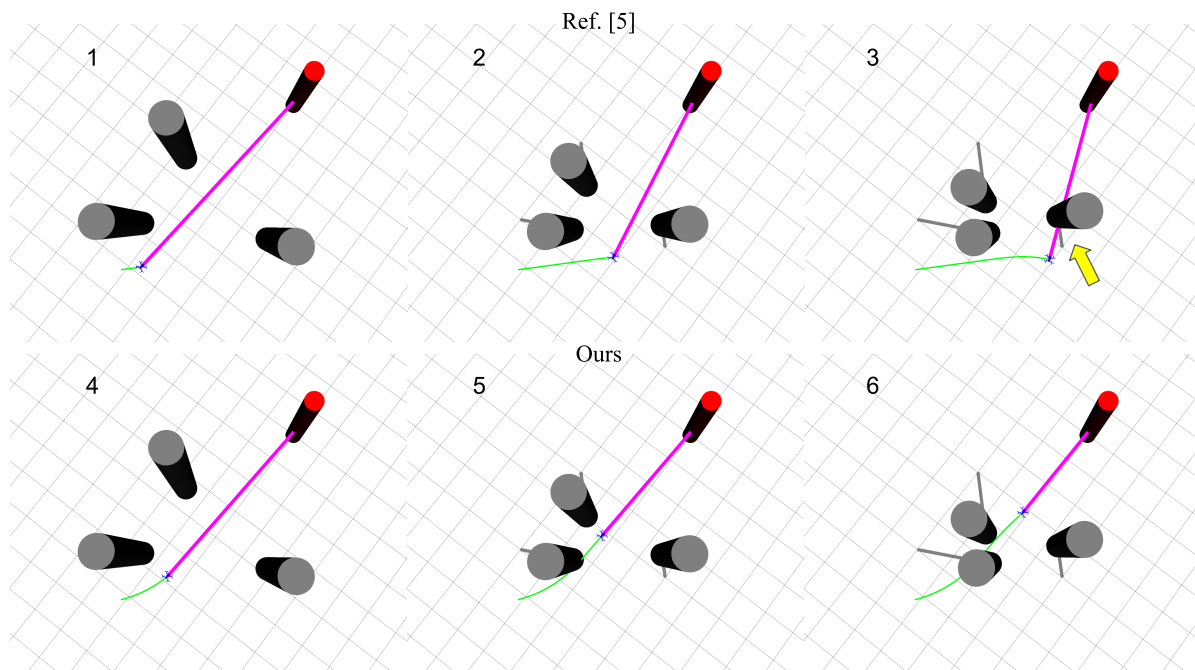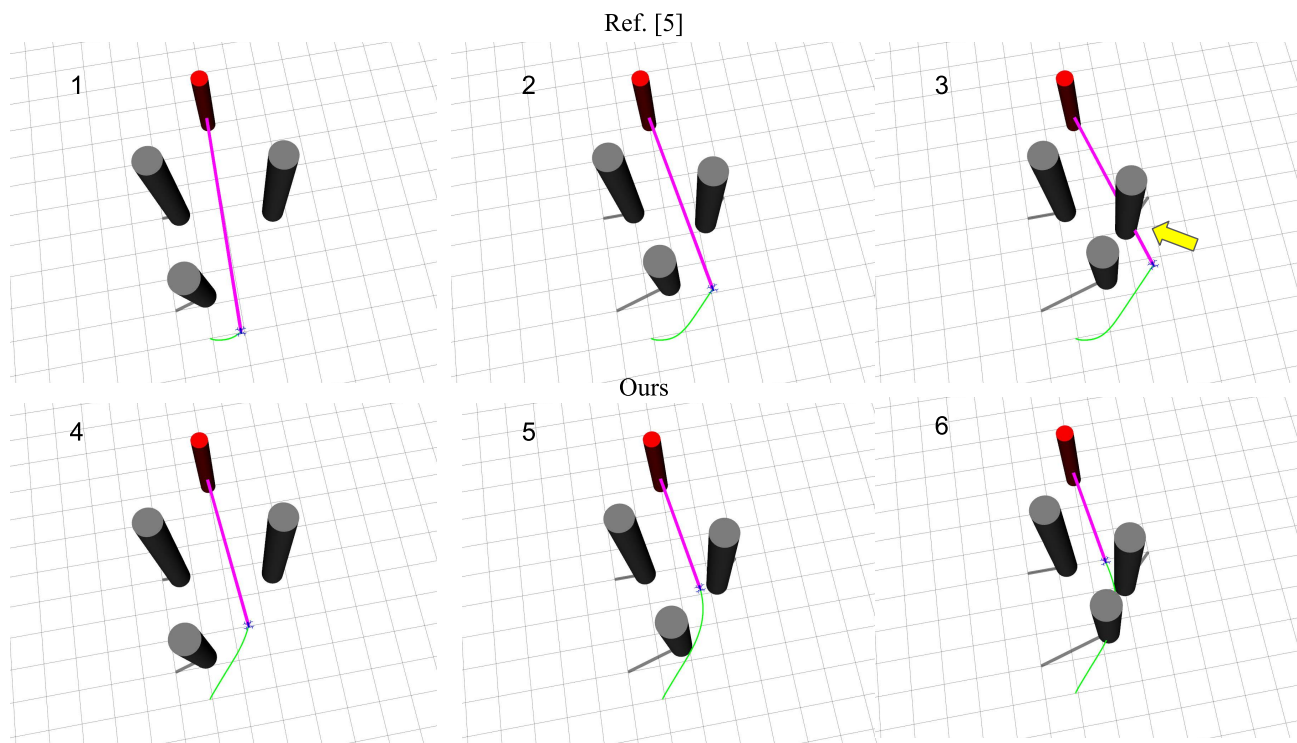
FIGURE 11. Figues show the qualitative comparison between our multi-convex MPC and [5]. We consider a static target amongst dynamic obstacles. The main task for the quadrotor is to continuously adapt its position to prevent occlusion from the dynamic obstacles. Our MPC maintains perfect visibility while the trajectories resulting from [5] get occluded at multiple instances.

qualitative results obtained across two benchmarks. In both these benchmarks, the quadrotor needs to find the narrow path among the obstacles and execute it before the obstacles enter its field of view. These represent a very challenging scenario and as shown, our MPC outperforms that proposed in [5] in avoiding occlusion. Fig. 12(a)-12(b) further quantify
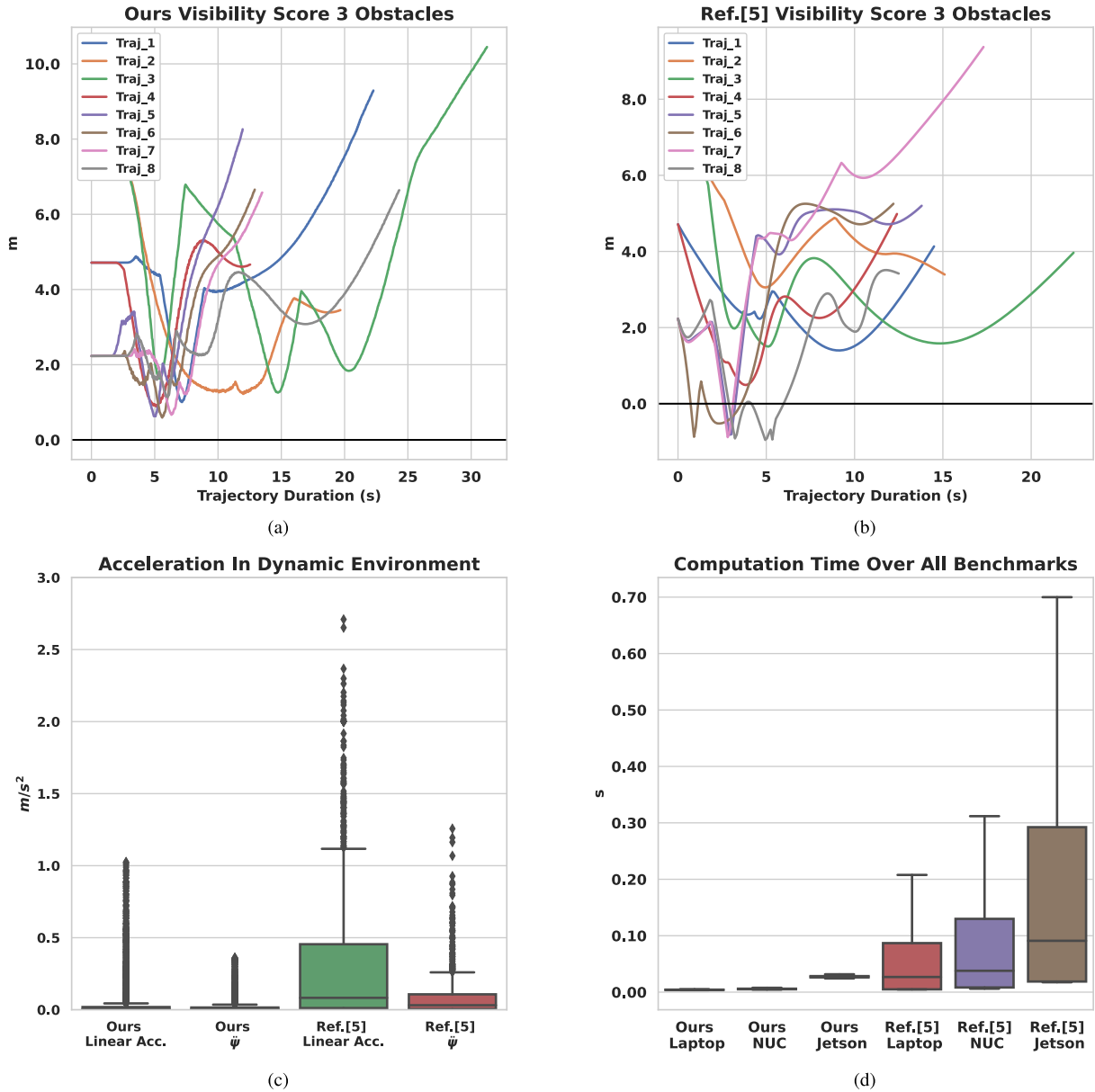
FIGURE 12. Quantitative comparison between our multi-convex MPC and [5]. We again outperform across all metrics: visibility score (a, b), trajectory smoothness (c), and computation time (d). We again reiterate that the visibility score is a proxy for satisfaction of occlusion avoidance constraints. Fig. (d) shows that our MPC can run in real-time on embedded hardware such as Intel NUC and Nvidia Jetson TX2. The latter is particularly crucial for quadrotors given its light-weight structure. Note that, in this particular experiment, the target is static and the simulation runs till the quadrotor reaches an occlusion-free region. Thus, the simulation times of ours and [5] are different.

the performance in terms of visibility score observed across all benchmarks. Visibility score along trajectories obtained with the MPC proposed in [5] go below zero at several time instants across different benchmarks.

Fig. 12(c) presents a comparison of the acceleration profiles observed across all the benchmarks in the considered dynamic environment tests. Due the very nature of this particular test set-up, we observed that accelerations generated by both our MPC and [5] follow a bang-bang like structure. That is, it increases sharply for a brief moment to provide the necessary velocity to the quadrotor followed by a large

portion of zero accelerations. A negative acceleration is then applied to bring the quadrotor to rest at the end. Due to this behavior, the median (or even mean) acceleration values will naturally be small and will not provide any meaningful information. Consequently, we compare the maximum acceleration values between our MPC and [5]. Our MPC uses a maximum acceleration of around $1.1 m/s$ which is 2.8 times lower than that generated by the [5]. The difference is similar for the angular acceleration with our MPC's maximum value ($0.4 rad/s^2$) being 2.5 times lower than that resulting from [5].
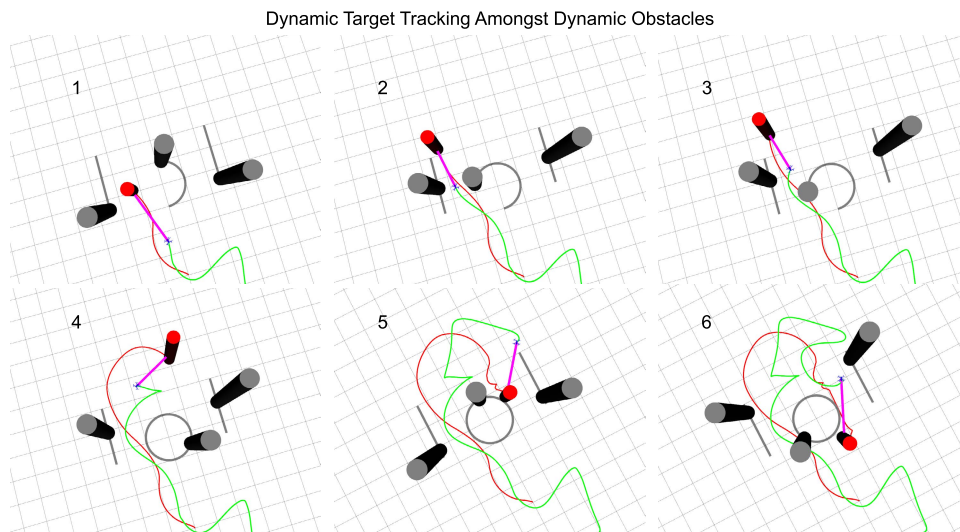
Dynamic Target Tracking Amongst Dynamic Obstacles

FIGURE 13. Demonstration of our MPC while tracking a dynamic target (red) amongst dynamic obstacles (grey).
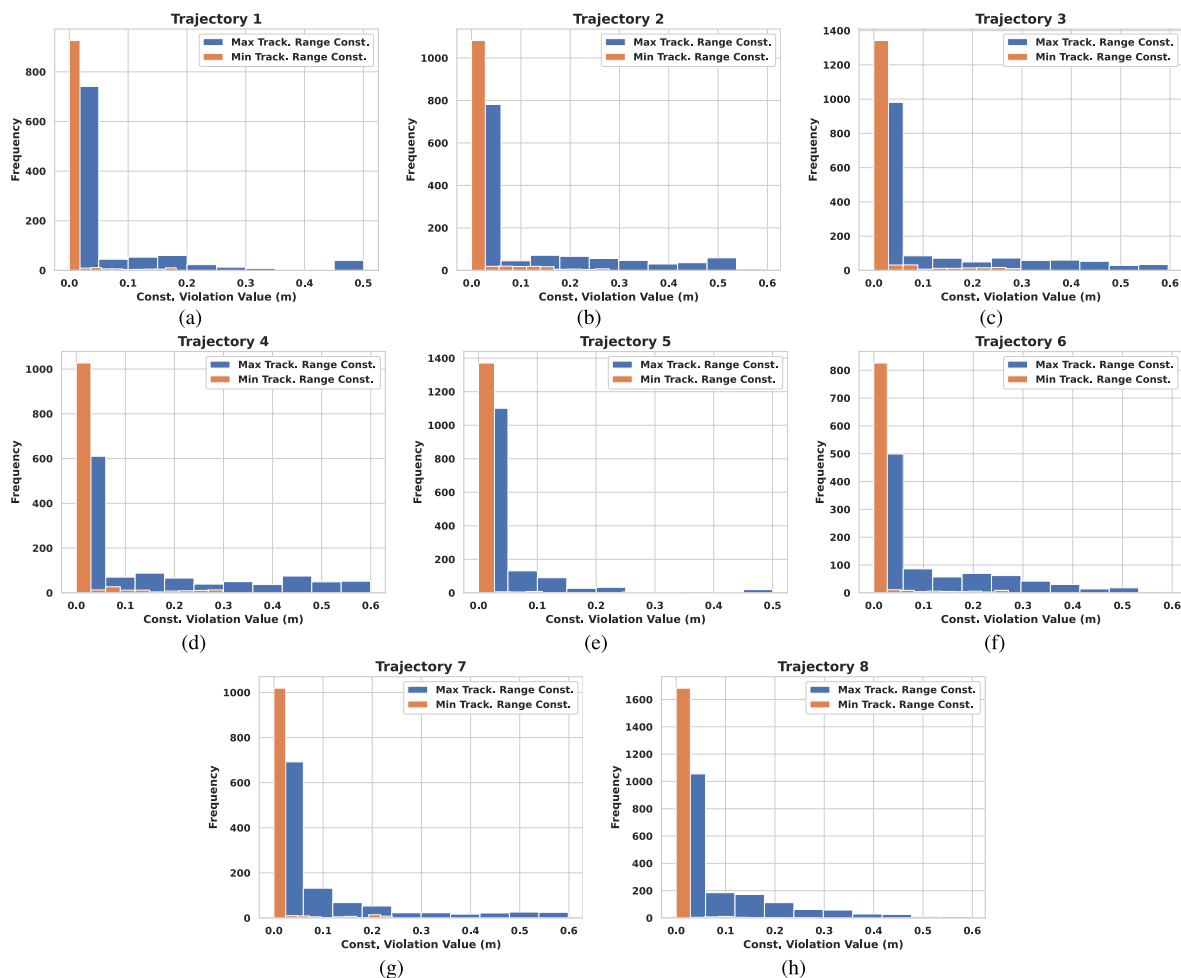

FIGURE 14. Fig. (a)-(h) verify how well our MPC maintains the range specified for tracking constraints (10) in the benchmarks considered in Section VI-C. Since we assume that the target trajectory is not known in advance, guaranteeing satisfaction of range thresholds at all time is intractable. Thus, we shift our attention to evaluating the extent to which our MPC can minimize the violation of minimum and maximum specified distance for target tracking. The above figures summarize the results in the form of histograms of the constraint violation observed in each simulation run. As can be seen, the violation is less than 10 cm for a large fraction of time in each simulation run.
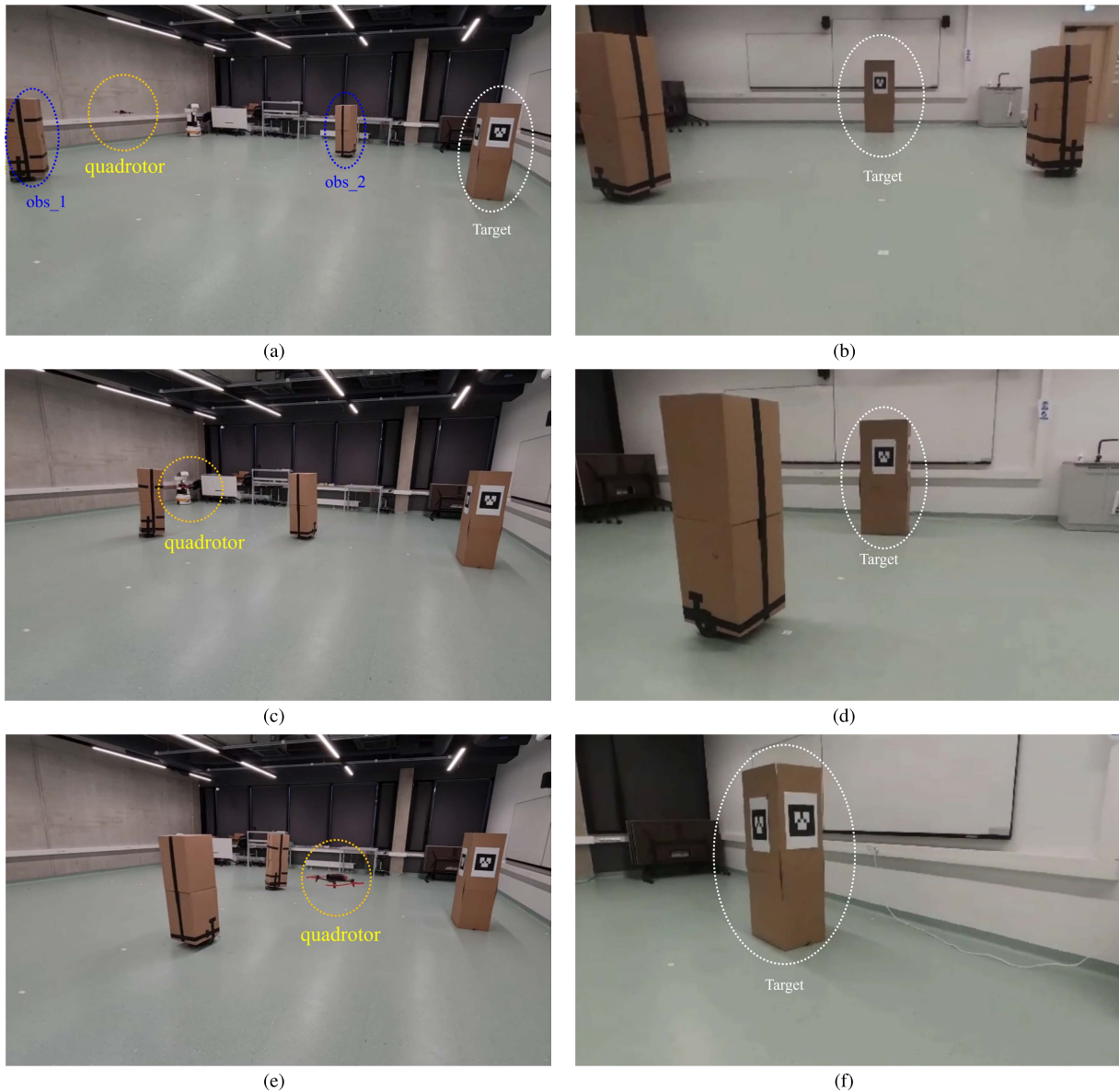
FIGURE 15. Snapshots from our hardware experiment. We consider the case of a static target amongst dynamic obstacles similar to the set-up discussed in Section VI-D. The quadrotor needs to constantly change its position to avoid both occlusion and collision from the dynamic obstacles. The first column shows the third-person view while the second column shows the view from the quadrotor's on-board camera.

Fig. 12(d) presents a comparison of the computation time statistics. In this particular test setup, we performed the comparisons on both laptops as well as embedded hardware devices such Intel i5 NUC and NVIDIA Jetson TX2. The laptop used had an i7-8750 processor with 16 GB RAM.

On the laptop, our MPC's median computation time of $0.004s$ is 5 times less than that required by [5]. However, a more meaningful difference can be obtained by looking at the maximum computation time required as this reflects the worst case performance. The computation time of our MPC has minimal variation and thus it hovers around the median value ($0.006s$). This turns out to be 15 times less than the maximum computation time observed for [5].

Our MPC maintains almost identical performance on both laptops and Intel NUC. But the worst-case computation time of [5] increases by 54%. Our MPC is also able to maintain a real-time performance on Jetson TX2 with worst-case computation time of $0.06s$. However, the performance of [5] degrades significantly with the worst-case computation time reaching up to $0.70s$.

2) INSIGHT INTO PERFORMANCE GAIN
The performance difference of [5] and our MPC can be correlated to their occlusion cost. As shown in 1(a)-1(b), the model (2) proposed in [5] is highly conservative as it assigns high cost value to regions where the field of view

is not occluded. Thus, in a highly cluttered and dynamic environment, the SQP optimizer used to optimize this cost fails to find zero occlusion regions. In contrast, our occlusion cost shown in Fig.4(a)-4(b) precisely captures the effect of obstacle geometry on visibility. Finally, the performance difference between [5] and ours can also be attributed to our faster re-planning due to shorter computation times.

E. ADDITIONAL RESULTS

Fig. 13 presents snapshots of a quadrotor tracking a moving target (red cylinder) amongst dynamic obstacles (grey cylinders) using our MPC. The quadrotor is provided with information about only the instantaneous positions and velocities of the target and the obstacles. Thus, it can construct only a crude linear approximation of their future trajectory. Although the actual trajectories for the target and the obstacles are highly non-linear, the fast re-planning ensured by our MPC allows the quadrotor to cope with the prediction errors and ensure occlusion/collision free tracking.

1) RANGE-CONSTRAINTS IN TARGET-TRACKING

For the benchmark considered in Section VI-C, our optimizer worked under the assumption that the target trajectory is not known. As a result, it is not possible to guarantee satisfaction of the range constraints ((1b) or (9)) at all times in real-time MPC setting. Nevertheless, in Fig. 14(a)-14(h), we verify how well our optimizer is able to maintain the desired minimum and maximum distance from the target. The min and max values were chosen as $2m$ and $2.5m$, respectively. As can be seen, even in the absence of any long-term information about the target trajectory, our MPC maintains the range thresholds for most of the duration in each simulation run. For a large fraction of time, the violation in minimum and maximum distance thresholds is less than 10cm.

2) COMPUTATION TIME SCALING

Fig.16 shows how the computation time of our MPC scales with the number of obstacles. The linear trend observed validates the remarks made in Section IV-E2.a. To recall, with an increase in the number of obstacles, only the computation complexity of constructing the cost function in (26) changes. The number of variables of the QP ($\mathbf{c}_x$, $\mathbf{c}_y$, $\mathbf{c}_z$) remains the same as it depends only on the trajectory parametrization. Furthermore, even within the cost term, only the matrix-vector product $\mathbf{A}^T\mathbf{b}$ needs to computed at each MPC iteration. The matrix-matrix product $\mathbf{A}^T\mathbf{A}$ does not change between the MPC iterations and thus can be precomputed. Although a naive matrix-vector product scales quadratically, with appropriate parallelization this scaling can be made linear. Most linear algebra libraries like Eigen [32] automatically implement such parallelization at the back-end through multi-threading over CPUs.

3) HARDWARE IMPLEMENTATION

Fig. 15(a)-15(f) show the snapshots from our hardware implementation. We consider a set-up similar to that described
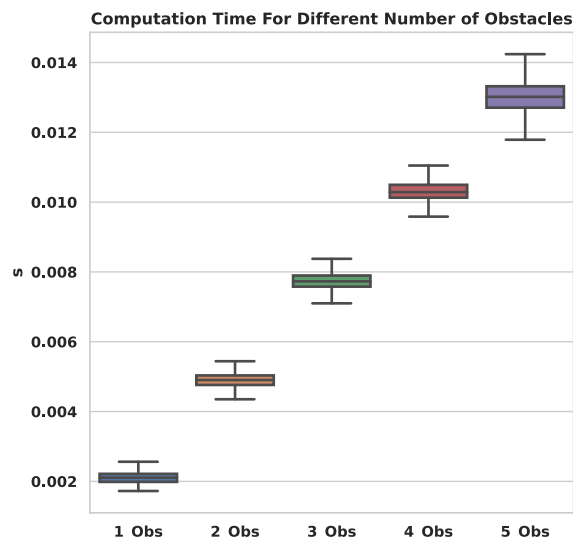

FIGURE 16. Linear computation-time scaling of our MPC with number of obstacles.

in Section VI-D: static target amongst dynamic obstacles. If we ignore the occlusion constraints, then the quadrotor can just hover in place. In contrast, to avoid occlusions from the dynamic obstacles and keep the target in the field of view, the quadrotor needs to constantly change its positions based on the current prediction of the obstacle trajectory.

VII. DISCUSSIONS

We proposed a real-time MPC for target tracking amongst static and dynamic obstacles. The efficacy of our MPC is derived from the custom optimizer employed to solve the underlying non-convex trajectory optimization problem. Our optimizer, in turn, is built on several layers of reformulation that induce multi-convexity into the problem structure. We show that our MPC substantially outperforms the state-of-the-art in several metrics: visibility score, control effort, and computation time. An attractive feature of our MPC is that it can cope with a very crude linear prediction of obstacle and target trajectory. In contrast, the state-of-the-art algorithm [12] requires the user to provide intermediate goal-points of the target, which is then used to perform sophisticated trajectory prediction for it. We note that such apriori knowledge of intermediate target goal-points is unlikely to be available in real-world settings. We observed in our experiments that in the absence of intermediate-goal points, the performance of [12] degraded substantially (see attached video).

As mentioned earlier, occlusion-free target tracking amongst dynamic obstacles is relatively less studied in the existing literature. Our simulation tests were significantly more complex than the current state-of-the-art [5] and our MPC showed substantial improvement over the cited work. In the accompanying video, we also show real-world hardware demonstration for dynamic environments.

Our simulations implicitly validates the ability of our multi-convex optimizer based MPC to withstand

uncertainties in dynamics and lower level control parameters. Our optimizer only considers kinematic model of the quadrotor. However, our ROS based simulator considers the full dynamics along with all the possible control delays. Our optimizer does not have access to the simulator parameters and thus in essence, these act as disturbances to our optimizer. It is worth pointing out that the disturbance rejection is purely a consequence of the fast milliseconds-level re-planning time of our multi-convex optimizer that allows for fast course correction of the quadrotor.

A. LIMITATIONS

The main limitation of a SB/ADMM approach for solving non-convex optimization problems such as ours is that the theoretical convergence properties are not well understood. That is, the AM iterates (26)-(28) may not converge. In spite of this, existing literature has an abundance of examples where SB/ADMM have been empirically found to work on even non-convex problems [33], [34]. Our own prior works [22], [23] have also presented similar empirical evidence. We have also empirically validated the convergence of the proposed optimizer in Section V. We conjecture that one of the reasons behind our AM's reliable convergence is that each sub-problem in every iteration is solved exactly to its minimum. A more rigorous analysis is part of our future work.

Another limitation of our optimizer is its sensitivity towards the parameter ρ that controls the violation of the occlusion and tracking constraints. We observed that although a default value of $\rho = 1$ always worked, it resulted in slow convergence. On the other hand, a larger value affected the smoothness of the resulting trajectory but converged to a low residual solution in lower number of iterations. In our implementation, we came up with an appropriate value by trial and error. One way to improve this aspect of our optimizer is to use a more sophisticated hyper-parameter tuning through Bayesian optimization.

B. FUTURE WORK

The applications of our optimizer and MPC go beyond target-tracking. It can also be used to keep important features in the field of view while navigating from a point to another point (Fig. 5(a)-5(e)). We are currently building on this foundation to improve localization in autonomous driving. In this set-up, neighboring cars often block the autonomous vehicle from observing features from either side of the road. Thus, the latter needs to constantly overtake or lane-change to maintain the visibility of important features. This is similar to the experiment shown in Fig. 11(a)-11(b).

In future, we also want to use our MPC as a teacher in a data-driven behavior-cloning framework to achieve occlusion-free target tracking based on just on-board depth images. We note that target tracking is also extensively performed with fixed-wing aerial vehicles that have more complicated kinematics than quadrotors. We aim to extend our current MPC to such class of vehicles as well while retaining the fundamental multi-convex computational structure.

REFERENCES

[1] C. Huang, F. Gao, J. Pan, Z. Yang, W. Qiu, P. Chen, X. Yang, S. Shen, and K.-T.-T. Cheng, "ACT: An autonomous drone cinematography system for action scenes," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 7039–7046.

[2] A. Ashtari, S. Stevšić, T. Nägeli, J.-C. Bazin, and O. Hilliges, "Capturing subjective first-person view shots with drones for automated cinematography," *ACM Trans. Graph.*, vol. 39, no. 5, pp. 1–14, Oct. 2020.

[3] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "PAMPC: Perception-aware model predictive control for quadrotors," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 1–8.

[4] T. Lipp and S. Boyd, "Variations and extension of the convex–concave procedure," *Optim. Eng.*, vol. 17, no. 2, pp. 263–287, 2016.

[5] T. Nägeli, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges, "Real-time motion planning for aerial videography with dynamic obstacle avoidance and viewpoint optimization," *IEEE Robot. Automat. Lett.*, vol. 2, no. 3, pp. 1696–1703, Jul. 2017.

[6] B. Penin, P. R. Giordano, and F. Chaumette, "Vision-based reactive planning for aggressive target tracking while avoiding collisions and occlusions," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3725–3732, Oct. 2018.

[7] P. Jain and P. Kar, "Non-convex optimization for machine learning," *Found. Trends Mach. Learn.*, vol. 10, nos. 3–4, pp. 142–336, 2017.

[8] T. Goldstein and S. Osher, "The split Bregman method for L1-regularized problems," *SIAM J. Imag. Sci.*, vol. 2, no. 2, pp. 323–343, Apr. 2009.

[9] G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein, "Training neural networks without gradients: A scalable Admm approach," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2722–2731.

[10] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: Bridging the gap via the real-time iteration," *Int. J. Control*, vol. 93, no. 1, pp. 62–80, 2016.

[11] M. Quigley, K. Conley, B. Gerkey, and J. Faust, "ROS: An open-source robot operating system," in *Proc. Workshop Open Source Softw. (ICRA)*, vol. 3, no. 3, Kobe, Japan, 2009, p. 5.

[12] B. Jeon, Y. Lee, and H. J. Kim, "Integrated motion planner for real-time aerial videography with a drone in a dense environment," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 1243–1249.

[13] Z. Han, R. Zhang, N. Pan, C. Xu, and F. Gao, "Fast-tracker: A robust aerial system for tracking agile target in cluttered environments," 2020, *arXiv:2011.03968*.

[14] P. Cozzi and F. Stoner, "GPU ray casting of virtual globes," in *Proc. ACM SIGGRAPH Posters (SIGGRAPH)*, 2010, p. 1.

[15] R. Bonatti, Y. Zhang, S. Choudhury, W. Wang, and S. A. Scherer, "Autonomous drone cinematographer: Using artistic principles to create smooth, safe, occlusion-free trajectories for aerial filming," in *Proc. ISER*, 2018.

[16] T. Bandyopadhyay, N. Rong, M. Ang, D. Hsu, and W. S. Lee, "Motion planning for people tracking in uncertain and dynamic environments," in *Proc. Workshop People Detection Tracking, IEEE Int. Conf. Robot. Automat.*, Apr. 2009, pp. 1–8.

[17] B. F. Jeon and H. J. Kim, "Online trajectory generation of a MAV for chasing a moving target in 3D dense environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 1115–1121.

[18] B. Houska, H. J. Ferreau, and M. Diehl, "ACADO toolkit–An open-source framework for automatic control and dynamic optimization," *Optim. Control Appl. Methods*, vol. 32, no. 3, pp. 298–312, May 2011.

[19] S. G. Johnson. (2017). *The NLopt Nonlinear-Optimization Package*. [Online]. Available: http://ab-initio.mit.edu/nlopt

[20] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 489–494.

[21] Q. Wang, Y. Gao, J. Ji, C. Xu, and F. Gao, "Visibility-aware trajectory optimization with application to aerial tracking," 2021, *arXiv:2103.06742*.

[22] F. Rastgar, A. K. Singh, H. Masnavi, K. Kruusamae, and A. Aabloo, "A novel trajectory optimization for affine systems: Beyond convex-concave procedure," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 1308–1315.

[23] F. Rastgar, H. Masnavi, J. Shrestha, K. Kruusamae, A. Aabloo, and A. K. Singh, "GPU accelerated convex approximations for fast multi-agent trajectory optimization," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 3303–3310, Apr. 2021.

[24] X. Shen, S. Diamond, M. Udell, Y. Gu, and S. Boyd, "Disciplined multi-convex programming," in *Proc. 29th Chin. Control Decis. Conf. (CCDC)*, May 2017, pp. 895–900.

[25] C. Gebhardt, B. Hepp, T. Nägeli, S. Stevšić, and O. Hilliges, "Airways: Optimization-based planning of quadrotor trajectories according to high-level user goals," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, 2016, pp. 2508–2519.

[26] C. Rösmann, F. Hoffmann, and T. Bertram, "Online trajectory planning in ROS under kinodynamic constraints with timed-elastic-bands," in *Robot Operating System (ROS): The Complete Reference*, vol. 2, A. Koubaa, Ed. Cham, Switzerland: Springer, 2017, pp. 231–261, doi: 10.1007/978-3-319-54927-9_7.

[27] Y. Liu, Q. Wang, H. Hu, and Y. He, "A novel real-time moving target tracking and path planning system for a quadrotor UAV in unknown unstructured outdoor scenes," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 11, pp. 2362–2372, Nov. 2019.

[28] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2010.

[29] M. W. Mueller and R. D'Andrea, "A model predictive controller for quadrocopter state interception," in *Proc. Eur. Control Conf. (ECC)*, Jul. 2013, pp. 1383–1389.

[30] P. Hespanhol and R. Quirynen, "A real-time iteration scheme with quasi-Newton Jacobian updates for nonlinear model predictive control," in *Proc. Eur. Control Conf. (ECC)*, Jun. 2018, pp. 1517–1522.

[31] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. von Stryk, "Comprehensive simulation of quadrotor UAVs using ROS and Gazebo," in *Simulation, Modeling, and Programming for Autonomous Robots*, I. Noda, N. Ando, D. Brugali, and J. J. Kuffner, Eds. Berlin, Germany: Springer, 2012, pp. 400–411.

[32] G. Guennebaud. (2010). *Eigen*. [Online]. Available: http://eigen.tuxfamily.org

[33] Q. Li, Z. Zhu, and G. Tang, "Alternating minimizations converge to second-order optimal solutions," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 3935–3943.

[34] J. Bento, N. Derbinsky, J. Alonso-Mora, and J. S. Yedidia, "A message-passing algorithm for multi-agent trajectory planning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 521–529.

VIVEK KANTILAL ADAJANIA is currently pursuing the master's degree with the University of Toronto Institute for Aerospace Studies (UTIAS), Canada. He is currently working as a Researcher (remotely) under Prof. Arun Kumar Singh at the University of Tartu, Estonia.

KARL KRUUSAMÄE received the M.S.E. degree in information technology and the Ph.D. degree in physics from the University of Tartu, in 2008 and 2012, respectively. He is currently an Associate Professor of robotics engineering at the University of Tartu, Estonia. Earlier, he has been a Postdoctoral Fellow at The University of Texas at Austin, Austin, TX, USA, and AIST, Japan. His research interests include human–robot and human–machine interaction and developing fault-tolerant robotic systems.

HOUMAN MASNAVI received the M.S.E. degree in robotics and computer engineering from the University of Tartu, Estonia, in 2020, where he is currently pursuing the Ph.D. degree. His research interests include motion planning, control, and ROS-based system integration for robotics.

ARUN KUMAR SINGH received the Ph.D. degree from IIIT-Hyderabad, India. He subsequently went to do his Postdoctoral Fellowships with Ben-Gurion University, Israel; Nanyang Technological University, Singapore; and Tampere University, Finland. He is currently an Associate Professor at the University of Tartu, Estonia. His research interests include motion planning, control, and learning as applied to control problems.

• • •