2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

AMSwarm: An Alternating Minimization Approach for Safe Motion Planning of Quadrotor Swarms in Cluttered Environments

Vivek K. Adajania, Siqi Zhou, Arun Kumar Singh, and Angela P. Schoellig

***Abstract*— This paper presents a scalable online algorithm to generate safe and kinematically feasible trajectories for quadrotor swarms. Existing approaches rely on linearizing Euclidean distance-based collision constraints and on axis-wise decoupling of kinematic constraints to reduce the trajectory optimization problem for each quadrotor to a quadratic program (QP). This conservative approximation often fails to find a solution in cluttered environments. We present a novel alternative that handles collision constraints without linearization and kinematic constraints in their quadratic form while still retaining the QP form. We achieve this by reformulating the constraints in a polar form and applying an Alternating Minimization algorithm to the resulting problem. Through extensive simulation results, we demonstrate that, as compared to Sequential Convex Programming (SCP) baselines, our approach achieves on average, a 72% improvement in success rate, a 36% reduction in mission time, and a 42 times faster per-agent computation time. We also show that collision constraints derived from discrete-time barrier functions (BF) can be incorporated, leading to different safety behaviours without significant computational overhead. Moreover, our optimizer outperforms the state-of-the-art optimal control solver ACADO in handling BF constraints with a 31 times faster per-agent computation time and a 44% reduction in mission time on average. We experimentally validated our approach on a Crazyflie quadrotor swarm of up to 12 quadrotors. The code with supplementary material and video are released for reference.**

I. Introduction

Quadrotor swarms have a great potential in applications such as search and rescue [1], mapping and environmental monitoring [2], and payload transport [3]. As compared to single quadrotors, quadrotors swarms offer increased flexibility, efficiency, and robustness [4].

In this paper, we consider the problem of motion planning for quadrotor swarms in cluttered environments and treat it as a trajectory optimization problem to be solved. In this context, the most straightforward approach is to formulate one joint optimization problem that computes trajectories for all quadrotors. Existing works have used both global mixed-integer linear programming [5] and local optimization-based Sequential Convex Programming (SCP) [6] approaches for solving the joint trajectory optimization problem. The solution space of these approaches is large but they quickly become intractable as the number of quadrotors grows.

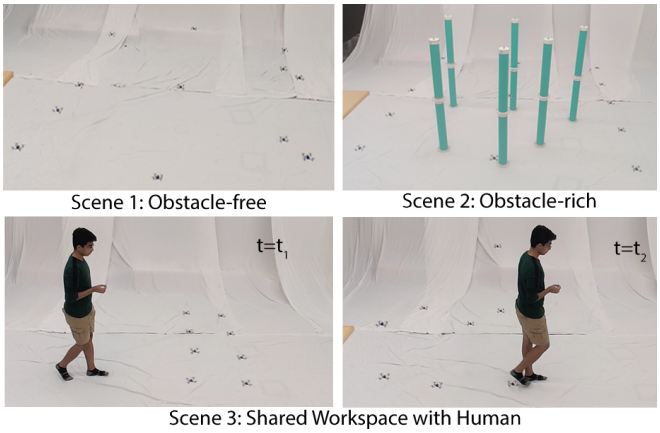

Fig. 1. Experimental demonstration of our distributed alternating minimization based approach for quadrotor swarm motion planning in challenging scenes. Link to video: `http://tiny.cc/AMSwarmVideo`. Link to code and supplementary material: `https://github.com/utiasDSL/AMSwarm`.

Distributed approaches provide a more scalable alternative, where each quadrotor solves an independent trajectory optimization problem taking into account the predicted trajectories of its neighbours [7]–[10]. The predicted trajectories are often assumed to be shared by the neighbouring quadrotors. These approaches have successfully demonstrated swarm motion planning for tens of quadrotors. However, we will show that their performance in terms of scalability, mission time, and computation time remains poor in highly cluttered environments. While there exist works that further incorporate a high-level discrete planner to improve the swarm performance in cluttered environments [11]–[13], in this work, we focus on the low-level trajectory optimization problem and propose an algorithm that addresses the limitations of existing distributed trajectory optimization baselines.

Some of the limitations of [7]–[9] and other related works such as [6] and [14] can be attributed to the underlying trajectory optimizer that relies on axis-wise decoupling of kinematic constraints and linearization of collision avoidance constraints. These affine approximations are made to obtain a quadratic program (QP) that can be solved efficiently. However, the computational benefits come at the expense of a reduced solution space. Moreover, while replanning in a receding horizon setting, the collision constraints are not active until the planned trajectories intersect with the neighbouring quadrotors or obstacles [15]. This reduces the responsiveness of the collision avoidance behaviour. One way to mitigate this issue is to use a longer prediction

Vivek K. Adajania, Siqi Zhou, and Angela P. Schoellig are with the Learning Systems and Robotics Lab (http://www.learnsyslab.org) at the University of Toronto Institute for Aerospace Studies, Canada, and the Technical University of Munich, Germany. They are also with the Vector Institute for Artificial Intelligence. Arun Kumar Singh is with the University of Tartu, Estonia. This research was in part supported by the European Social Fund via the ICT program measure and grant PSG753 from the Estonian Research Council. Emails: {vivek.adajania, siqi.zhou}@robotics.utias.utoronto.ca, arun.singh@ut.ee, and angela.schoellig@tum.de.

horizon; however, this increases the computation time.

Our proposed optimizer addresses both limitations discussed above: the conservativeness of existing approaches due to approximations and the late responsiveness to neighbouring quadrotors or obstacles. We show that by reformulating the quadratic kinematic constraints and collision constraints into polar form and applying an Alternating Minimization (AM) algorithm to the resulting optimization problem, we can retain a QP without requiring any linearization. As a result, we obtain more aggressive motions with improved metrics for swarm planning, such as mission time and success rate. Our formulation naturally extends to the case when collision avoidance is modelled by a discrete-time barrier function (BF) [15]. This dramatically improves safety metrics such as clearance to neighbouring quadrotors and obstacles while incurring no significant computational cost. To the best of our knowledge, only a few works, such as [16], [17], have incorporated BF constraints over the entire planning horizon; most works such as [18] and [19] consider a one-step reactive planning approach. Among the multi-step approaches, ours is the first to formulate trajectory planning with BF constraints as a QP (see Section II-C).

We compare our approach with the SCP baselines from [7]–[9] and show on average a 72% improvement in success rate, a 36% reduction in mission time, and a 42 times faster per-agent computation time. Additionally, we show that the proposed approach with BF constraints allows us to introduce different safety behaviours. We further show that our optimizer's handling of discrete-time BF constraints outperforms the state-of-the-art solver ACADO [20] with a 31 times faster per-agent computation time and a 44% reduction in mission time on average.

II. Distributed Motion Planning Problem

Our goal is to generate smooth, collision-free, and kinematically feasible trajectories that navigate N quadrotors from their initial positions $\mathbf{p}_{i,o}$ to their desired goal positions $\mathbf{p}_{i,g}$ in an obstacle-rich and possibly dynamic environment. The vector $\mathbf{p} = [x,\ y,\ z]^T$ is the three-dimensional position of the quadrotor, the subscript i is the quadrotor index, and the subscripts o and g denote initial and goal variables.

Similar to [7], [8], we formulate the quadrotor swarm motion planning as a distributed trajectory optimization problem, where the computation for each quadrotor is parallelized. At each time step, the quadrotors exchange the planned trajectories from the previous step and re-optimize the trajectories towards their goal positions subject to constraints. The distributed optimization problem is solved in a receding horizon fashion until each quadrotor reaches its goal position.

We note that, in this work, the quadrotors' trajectories are optimized online to account for possible dynamic obstacles. We assume that the obstacles' current positions and velocities are available to each quadrotor.

A. Optimization Problem Formulation

At each planning step, the optimization problem solved by quadrotor i is formulated as follows:

$$\min_{\mathbf{p}_i}\ w_g \sum_{k=K-\kappa}^{K-1} \|\mathbf{p}_i[k] - \mathbf{p}_{i,g}\|^2 + w_s \sum_{k=0}^{K-1} \left\|\mathbf{p}_i^{(q)}[k]\right\|^2 \quad \text{(1a)}$$

$$\text{s.t. } \mathbf{p}_i^{(q)}[0] = \mathbf{p}_{i,a}^{(q)},\ \forall q = \{0,1,2\} \quad \text{(1b)}$$

$$\underline{\mathbf{p}} \preceq \mathbf{p}_i[k] \preceq \overline{\mathbf{p}},\ \forall k \quad \text{(1c)}$$

$$\|\dot{\mathbf{p}}_i[k]\|^2 \leq \overline{v}^2,\ \forall k \quad \text{(1d)}$$

$$\underline{f}^2 \leq \|\ddot{\mathbf{p}}_i[k] + \mathbf{g}\|^2 \leq \overline{f}^2, \forall k \quad \text{(1e)}$$

$$h_{ij}[k] = \left\|\mathbf{\Theta}_{ij}^{-1}(\mathbf{p}_i[k] - \boldsymbol{\xi}_j[k])\right\|^2 - 1 \geq 0,\ \forall k, j, \quad \text{(1f)}$$

where k is the discrete-time index, K is the planning horizon length, $||\cdot||$ denotes the Euclidean norm, and the superscript (q) denotes the q-th time derivative of a variable.

The cost function consists of two terms. The first term is the goal cost that penalizes the deviation of the position of the quadrotor from the specified goal position over the last $\kappa < K$ steps in the prediction horizon; the second term is the smoothness cost that penalizes the q-th derivatives of the position trajectory. The constants w_g and w_s are weights trading off the respective cost terms.

The equality constraints (1b) set the initial position of the trajectory and the higher derivatives to be consistent with the current values of the quadrotor. The inequalities (1c)-(1e) enforce bounds on the position $(\underline{\mathbf{p}}, \overline{\mathbf{p}})$, bounds on the velocity $(-\overline{v}, \overline{v})$, and bounds on the acceleration $(\underline{f}, \overline{f})$. The inequalities (1f) enforce the collision avoidance requirement with either the j-th neighbouring quadrotor or obstacle with position $\boldsymbol{\xi}_j[k]$. The matrix $\mathbf{\Theta}_{ij}$ is a diagonal matrix with (a, b, c) being its element. These scalars (a, b, c) characterize the axis lengths of the ellipsoidal envelopes around the neighbouring quadrotors or the obstacles. The vector $\mathbf{g} = [0,\ 0,\ g]^T$ is the gravitational acceleration vector, where g is the acceleration due to gravity.

Alternative Collision Avoidance Constraint: The condition in (1f) is commonly found in works on quadrotor swarms (e.g., [7], [8], [13], [21]). A fundamental problem with the standard collision avoidance constraint (1f) is that these inequalities do not get activated until the planned trajectory intersects with the neighbouring quadrotors or obstacles [15]. Due to the receding horizon nature of the planning, a quadrotor only tries to avoid collisions with its neighbours or obstacles when it is sufficiently close to them. Increasing the planning horizon can mitigate this issue but at the cost of increased computation time. An alternative approach is to use BF constraints to induce a desired collision avoidance behaviour [15]:

$$h_{ij}[k] - h_{ij}[k-1] \geq -\gamma\, h_{ij}[k-1],\ \forall k, j, \qquad (2)$$

where $\gamma \in [0, 1]$ is a constant controlling how fast the quadrotor is allowed to approach the constraint boundary given by $h_{ij} = 0$. Smaller values of γ generally result in more gradual and conservative collision avoidance behaviours. With $\gamma = 1$, we recover the original collision-avoidance constraint in (1f).

B. Trajectory Parameterization

We parameterize the x-, y-, and z-position trajectories for each quadrotor as Bernstein polynomials of degree n. For instance, the x-position trajectory for the i-th quadrotor is

$$\begin{bmatrix} x_i[0] & x_i[1] & \dots & x_i[K-1] \end{bmatrix}^T = \mathbf{W}\mathbf{c}_{i,x}, \tag{3}$$

where $\mathbf{W} \in \mathbb{R}^{K\times(n+1)}$ is the Bernstein basis matrix and $\mathbf{c}_{i,x}$ are the coefficients associated with it. The k-th row and m-th column element of $\mathbf{W}$ is $[\mathbf{W}]_{km} = \binom{n}{m}(1 - t/(K-1)\delta t)^{n-m}(t/(K-1)\delta t)^m$, where δt is the discrete-time step size, and $t = k\delta t$ is the continuous-time variable. The higher derivatives of the position trajectory have the general form $\mathbf{W}^{(q)}\mathbf{c}_{i,x}$, where $\mathbf{W}^{(q)}$ is the q-th derivative of the Bernstein basis matrix. The position trajectories for the y- and z-directions are defined in a similar way.

C. Challenges in Solving the Optimization Problem

The optimization problem in (1a)-(1f) is a non-convex quadratically constrained quadratic program (QCQP). Existing works (e.g., [6]–[8]) achieve a more favourable QP form by deriving the convex approximation of (1d)-(1f): the velocity and acceleration bounds are split into axis-wise affine bounds, and the non-convex collision avoidance constraints are approximated as affine constraints through linearization along a trajectory. These approximations can lead to a substantial loss of the feasible space (see Fig. 3 in [22], Fig. 2 in [23]).

Achieving a QP structure with the BF constraints (2) is even more challenging. To see this, we rewrite (2) as

$$-h_{ij}[k] + (1-\gamma)h_{ij}[k-1] \leq 0,\ \forall k, j. \tag{4}$$

The constraint is non-convex, and the non-convexity comes from the first term $-h_{ij}[k]$. If we linearize the first term, we get an exact but a more conservative convex substitute for the BF constraint. However, the resulting constraint still remains quadratic in the decision variables due to the presence of $h_{ij}[k-1]$. Linearizing the complete right-hand side of (4) will lead to a QP form. However, satisfaction of a completely linearized version may not imply satisfaction of (4) [24].

III. The Alternating Minimization Algorithm

This section presents our main algorithmic results: an AM-based linearization-free trajectory optimizer for solving the motion planning problem introduced in Sec. II-A. We first present the solution for the case with standard collision constraints (1f). We then show how it naturally extends to the BF constraints (2).

A. Constraints Reformulation

In our proposed approach, one key ingredient that enables us to bring the QCQP problem to a QP form is a polar reformulation of the quadratic constraints (1d)-(1f). Here we present a general form of the polar representation presented in [25] for quadratic constraints.

Consider an inequality of the form $||\mathbf{M}(\mathbf{v}-\mathbf{v}_0)||^2 \leq \eta^2$ (or $||\mathbf{M}(\mathbf{v}-\mathbf{v}_0)||^2 \geq \eta^2$) with $\mathbf{M}$ being a diagonal matrix with positive entries. The inequality constraint can be equivalently written in a polar form as follows: $\mathbf{f} = \mathbf{M}(\mathbf{v}-\mathbf{v}_0) - d\,\boldsymbol{\omega}(\alpha,\beta) = 0$ with $d \leq \eta$ (or $d \geq \eta$). Here, $\boldsymbol{\omega}(\alpha,\beta) = [\cos\alpha\sin\beta,\ \sin\alpha\sin\beta,\ \cos\beta]^T$ is a unit direction vector pointing from $\mathbf{v}_0$ to $\mathbf{v}$ with α being the azimuthal angle and β being the polar angle, and the scalar d is the magnitude of the vector $\mathbf{M}(\mathbf{v}-\mathbf{v}_0)$.

Using the polar reparametrization, we can write the quadratic constraints (1d), (1e), and (1f) as the following constraint sets:

$$\mathcal{C}_{i,v}[k] = \{\dot{\mathbf{p}}_i[k] \in \mathbb{R}^3 \mid \mathbf{f}_{i,v}[k] = 0, d_{i,v}[k] \leq \overline{v}\},\ \forall k, \tag{5}$$

$$\mathcal{C}_{i,a}[k] = \{\ddot{\mathbf{p}}_i[k] \in \mathbb{R}^3 \mid \mathbf{f}_{i,a}[k] = 0, \underline{f} \leq d_{i,a}[k] \leq \overline{f}\},\ \forall k, \tag{6}$$

$$\mathcal{C}_{ij,c}[k] = \{\mathbf{p}_i[k] \in \mathbb{R}^3 \mid \mathbf{f}_{ij,c}[k] = 0, d_{ij,c}[k] \geq 1\},\ \forall k, j, \tag{7}$$

where the functions $\mathbf{f}_{ij,c}$, $\mathbf{f}_{i,v}$, and $\mathbf{f}_{i,a}$ are

$$\begin{aligned}
\mathbf{f}_{i,v}[k] &= \dot{\mathbf{p}}_i[k] - d_{i,v}[k]\,\boldsymbol{\omega}(\alpha_{i,v}[k], \beta_{i,v}[k]),\\
\mathbf{f}_{i,a}[k] &= \ddot{\mathbf{p}}_i[k] + \mathbf{g} - d_{i,a}[k]\,\boldsymbol{\omega}(\alpha_{i,a}[k], \beta_{i,a}[k]),\\
\mathbf{f}_{ij,c}[k] &= \boldsymbol{\Theta}_{ij}^{-1}(\mathbf{p}_i[k] - \boldsymbol{\xi}_j[k]) - d_{ij,c}[k]\boldsymbol{\omega}(\alpha_{ij,c}[k], \beta_{ij,c}[k]).
\end{aligned}$$

Note that $(\alpha_{\cdot,\cdot}, \beta_{\cdot,\cdot}, d_{\cdot,\cdot})$ are the parameters of the polar form representations of the constraints and will be computed by our optimizer together with the trajectory.

B. Reformulated Problem

Before deriving the final form of our reformulated problem, we rewrite the polar form constraints derived in the previous subsection in a compact matrix form. Given the parametrization in (3), the equality part of constraints (5), (6), (7) can be represented as

$$\overbrace{\begin{bmatrix} \widetilde{\mathbf{A}} & & \\ & \widetilde{\mathbf{A}} & \\ & & \widetilde{\mathbf{A}} \end{bmatrix}}^{\mathbf{A}} \overbrace{\begin{bmatrix} \mathbf{c}_{i,x} \\ \mathbf{c}_{i,y} \\ \mathbf{c}_{i,z} \end{bmatrix}}^{\zeta_{i,1}} = \overbrace{\left[\begin{array}{c} \mathbf{d}_{i,v}\cos\boldsymbol{\alpha}_{i,v}\sin\boldsymbol{\beta}_{i,v} \\ \mathbf{d}_{i,a}\cos\boldsymbol{\alpha}_{i,a}\sin\boldsymbol{\beta}_{i,a} \\ \boldsymbol{\xi}_x + a\mathbf{d}_{i,c}\cos\boldsymbol{\alpha}_{i,c}\sin\boldsymbol{\beta}_{i,c} \\ \hline \mathbf{d}_{i,v}\sin\boldsymbol{\alpha}_{i,v}\sin\boldsymbol{\beta}_{i,v} \\ \mathbf{d}_{i,a}\sin\boldsymbol{\alpha}_{i,a}\sin\boldsymbol{\beta}_{i,a} \\ \boldsymbol{\xi}_y + b\mathbf{d}_{i,c}\sin\boldsymbol{\alpha}_{i,c}\sin\boldsymbol{\beta}_{i,c} \\ \hline \mathbf{d}_{i,v}\cos\boldsymbol{\beta}_{i,v} \\ -\widetilde{\mathbf{g}} + \mathbf{d}_{i,a}\cos\boldsymbol{\beta}_{i,a} \\ \boldsymbol{\xi}_z + c\mathbf{d}_{i,c}\cos\boldsymbol{\beta}_{i,c} \end{array}\right]}^{\mathbf{b}}. \tag{8}$$

Here, $\widetilde{\mathbf{A}} = \begin{bmatrix} \dot{\mathbf{W}}^T & \ddot{\mathbf{W}}^T & \mathbf{F}_c^T \end{bmatrix}^T$, the matrix $\mathbf{F}_c$ is constructed by vertically stacking the matrix $\mathbf{W}$ as many times as the number of neighbouring quadrotors and obstacles present in the environment. The vectors $(\boldsymbol{\xi}_x, \boldsymbol{\xi}_y, \boldsymbol{\xi}_z)$ are formed by vertically stacking the corresponding variables $(\xi_{j,x}[k], \xi_{j,y}[k], \xi_{j,z}[k])$ at different time steps of the prediction horizon and for all the neighbouring quadrotors and obstacles. In a similar fashion, $(\boldsymbol{\alpha}_{\cdot,\cdot}, \boldsymbol{\beta}_{\cdot,\cdot}, \mathbf{d}_{\cdot,\cdot})$ are formed by vertically stacking $(\alpha_{\cdot,\cdot}[k], \beta_{\cdot,\cdot}[k], d_{\cdot,\cdot}[k])$. The vector $\widetilde{\mathbf{g}}$ is formed by vertically stacking g as many times as the length of prediction horizon.

Using the derivations above, we now write the reformulated optimization problem as

$$\min_{\zeta_{i,1}, \zeta_{i,2}, \zeta_{i,3}} \ \frac{1}{2}\boldsymbol{\zeta}_{i,1}^T\mathbf{Q}\boldsymbol{\zeta}_{i,1} + \mathbf{q}^T\boldsymbol{\zeta}_{i,1} \tag{9a}$$

$$\text{s.t. } \mathbf{A}\boldsymbol{\zeta}_{i,1} = \mathbf{b}(\boldsymbol{\zeta}_{i,2}, \boldsymbol{\zeta}_{i,3}) \tag{9b}$$

$$\mathbf{G}\boldsymbol{\zeta}_{i,1} \preceq \mathbf{h} \tag{9c}$$

$$\boldsymbol{\zeta}_{i,1} \in \mathcal{C}_{\zeta_{i,1}}, \boldsymbol{\zeta}_{i,3} \in \mathcal{C}_{\zeta_{i,3}}, \tag{9d}$$

where $\boldsymbol{\zeta}_{i,1} = [\mathbf{c}_{i,x}^T, \mathbf{c}_{i,y}^T, \mathbf{c}_{i,z}^T]^T$, $\boldsymbol{\zeta}_{i,2} = [\boldsymbol{\alpha}_{i,c}^T, \boldsymbol{\alpha}_{i,a}^T, \boldsymbol{\alpha}_{i,v}^T, \boldsymbol{\beta}_{i,c}^T, \boldsymbol{\beta}_{i,a}^T, \boldsymbol{\beta}_{i,v}^T]^T$, and $\boldsymbol{\zeta}_{i,3} = [\mathbf{d}_{i,c}, \mathbf{d}_{i,a}, \mathbf{d}_{i,v}]^T$ are the variables to be optimized. The matrix $\mathbf{Q}$ and vector $\mathbf{q}$ are formed from the objective function (1a). The matrix $\mathbf{G}$ and vector $\mathbf{h}$ in the inequality constraint (9c) stem from the positional bounds (1c).

The set $\mathcal{C}_{\boldsymbol{\zeta}_{i,1}}$=$\{\boldsymbol{\zeta}_{i,1} \in \mathbb{R}^{3n} | \mathbf{C}\boldsymbol{\zeta}_{i,1} = \mathbf{e}\}$ encodes the initial conditions (1b). Here, the matrix $\mathbf{C} = [\mathbf{W}_0^T, \dot{\mathbf{W}}_0^T, \ddot{\mathbf{W}}_0^T]^T$, and the subscript 0 represents the 0-th row of the respective matrices. The vector $\mathbf{e} = [\mathbf{p}_{i,a}^T, \dot{\mathbf{p}}_{i,a}^T, \ddot{\mathbf{p}}_{i,a}^T]^T$ contains the current position, velocity and acceleration values. The set $\mathcal{C}_{\boldsymbol{\zeta}_{i,3}}$ consists of the conditions on each of the variables $(\mathbf{d}_{ij,c}, \mathbf{d}_{i,v}, \mathbf{d}_{i,a})$ derived from the polar reformulation.

C. Relaxation and Solution by AM

The optimization (9a)-(9d) has some hidden convex structures which makes it suitable for AM-based approaches. To exploit these structures, we first relax the non-convex equality (9b) and affine (9c) constraints as penalties in the following form:

$$\min_{\boldsymbol{\zeta}_{i,1} \in \mathcal{C}_{\zeta_{i,1}}, \boldsymbol{\zeta}_{i,3} \in \mathcal{C}_{\zeta_{i,3}}} \frac{1}{2}\boldsymbol{\zeta}_{i,1}^T\mathbf{Q}\boldsymbol{\zeta}_{i,1} + \mathbf{q}^T\boldsymbol{\zeta}_{i,1} - \langle \boldsymbol{\lambda}_i, \boldsymbol{\zeta}_{i,1} \rangle + \frac{\rho}{2}\left\|\mathbf{A}\boldsymbol{\zeta}_{i,1} - \mathbf{b}(\boldsymbol{\zeta}_{i,2}, \boldsymbol{\zeta}_{i,3})\right\|^2 + \frac{\rho}{2}\left\|\mathbf{G}\boldsymbol{\zeta}_{i,1} - \mathbf{h} + \mathbf{s}_i\right\|^2. \tag{10}$$

The parameter ρ trades-off satisfaction of constraint residual with the minimization of primary cost function. The slack variable $\mathbf{s}_i \geq \mathbf{0}$ is unknown, and we discuss shortly how these are obtained within an AM setup. The vector $\boldsymbol{\lambda}_i$ is called the Lagrange multiplier and is crucial for driving the constraint residuals to zero [26].

Algorithm 1 summarizes the AM steps for minimizing (10), wherein the superscript l in ${}^l(.)$ represents the value of $(.)$ at l-th iteration of the algorithm. At each step of the AM, only one of $\boldsymbol{\zeta}_{i,1}, \boldsymbol{\zeta}_{i,2}, \boldsymbol{\zeta}_{i,3}$ are optimized while the rest are held fixed at values obtained in the previous update. Each step in Algorithm 1 is either a convex QP or has a closed-form solution. We discuss these observations below in detail.

Step (S1): We solve for $\boldsymbol{\zeta}_{i,1}$ while keeping the other variables constant. We see that the problem is an equality-constrained convex QP whose solution boils down to solving a set of linear equations:

$$\begin{bmatrix} \check{\mathbf{A}} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} {}^{l+1}\boldsymbol{\zeta}_{i,1} \\ \boldsymbol{\mu} \end{bmatrix} = \begin{bmatrix} {}^l\check{\mathbf{b}} \\ \mathbf{e} \end{bmatrix}, \tag{11}$$

where the matrix $\check{\mathbf{A}}$=$\mathbf{Q} + \rho\mathbf{G}^T\mathbf{G} + \rho\mathbf{A}^T\mathbf{A}$, the vector ${}^l\check{\mathbf{b}}$=$\mathbf{q} + \rho\mathbf{G}^T(\mathbf{h} - {}^l\mathbf{s}_i) + \rho\mathbf{A}^T{}^l\mathbf{b} - {}^l\boldsymbol{\lambda}_i$, and $\boldsymbol{\mu}$ are the dual variables associated with the equality constraints.

Step (S2): We now solve for $\boldsymbol{\zeta}_{i,2}$. As an example, the optimization problem for the variables $(\boldsymbol{\alpha}_{i,c}, \boldsymbol{\beta}_{i,c})$ is

$${}^{l+1}\boldsymbol{\alpha}_{i,c}, {}^{l+1}\boldsymbol{\beta}_{i,c} = \arg\min_{\boldsymbol{\alpha}_{i,c}, \boldsymbol{\beta}_{i,c}}$$

Algorithm 1 The AM algorithm used by quadrotor i at each planning step

1: Initialize ${}^l\boldsymbol{\zeta}_{i,2} = \mathbf{0}, {}^l\boldsymbol{\zeta}_{i,3} = \mathbf{0}, {}^l\boldsymbol{\lambda}_i = \mathbf{0}, {}^l\mathbf{s}_i = \mathbf{0}$ at $l = 0$
2: **while** $l \leq$ maxiter or residuals $\geq$ thresold **do**
3:

$${}^{l+1}\boldsymbol{\zeta}_{i,1} = \arg\min_{\boldsymbol{\zeta}_{i,1} \in \mathcal{C}_{\zeta_{i,1}}} \frac{1}{2}\boldsymbol{\zeta}_{i,1}^T\mathbf{Q}\boldsymbol{\zeta}_{i,1} + \mathbf{q}^T\boldsymbol{\zeta}_{i,1} - \langle {}^l\boldsymbol{\lambda}, \boldsymbol{\zeta}_{i,1} \rangle + \frac{\rho}{2}\left\|\mathbf{A}\boldsymbol{\zeta}_{i,1} - \mathbf{b}({}^l\boldsymbol{\zeta}_{i,2}, {}^l\boldsymbol{\zeta}_{i,3})\right\|^2 + \frac{\rho}{2}\left\|\mathbf{G}\boldsymbol{\zeta}_{i,1} - \mathbf{h} + {}^l\mathbf{s}_i\right\|^2 \tag{S1}$$

$${}^{l+1}\boldsymbol{\zeta}_{i,2} = \arg\min_{\boldsymbol{\zeta}_{i,2}} \frac{\rho}{2}\left\|\mathbf{A}^{l+1}\boldsymbol{\zeta}_{i,1} - \mathbf{b}(\boldsymbol{\zeta}_{i,2}, {}^l\boldsymbol{\zeta}_{i,3})\right\|^2 \tag{S2}$$

$${}^{l+1}\boldsymbol{\zeta}_{i,3} = \arg\min_{\boldsymbol{\zeta}_{i,3} \in \mathcal{C}_{\zeta_{i,3}}} \frac{\rho}{2}\left\|\mathbf{A}^{l+1}\boldsymbol{\zeta}_{i,1} - \mathbf{b}({}^{l+1}\boldsymbol{\zeta}_{i,2}, \boldsymbol{\zeta}_{i,3})\right\|^2 \tag{S3}$$

$${}^{l+1}\mathbf{s}_i = \max\left(0, -\mathbf{G}^{l+1}\boldsymbol{\zeta}_{i,1} - \mathbf{h}\right) \tag{S4}$$

$${}^{l+1}\boldsymbol{\lambda}_i = {}^l\boldsymbol{\lambda}_i - \frac{\rho}{2}\mathbf{A}^T\left(\mathbf{A}^{l+1}\boldsymbol{\zeta}_{i,1} - \mathbf{b}({}^{l+1}\boldsymbol{\zeta}_{i,2}, {}^{l+1}\boldsymbol{\zeta}_{i,3})\right) - \frac{\rho}{2}\mathbf{G}^T\left(\mathbf{G}^{l+1}\boldsymbol{\zeta}_{i,1} - \mathbf{h} + {}^{l+1}\mathbf{s}_i\right) \tag{S5}$$

4: **end while**

$$\left\|\mathbf{F}_c{}^{l+1}\mathbf{c}_{i,x} - \boldsymbol{\xi}_x - a^l\mathbf{d}_{i,c}\cos\boldsymbol{\alpha}_{i,c}\sin\boldsymbol{\beta}_{i,c}\right\|^2 + \left\|\mathbf{F}_c{}^{l+1}\mathbf{c}_{i,y} - \boldsymbol{\xi}_y - b^l\mathbf{d}_{i,c}\sin\boldsymbol{\alpha}_{i,c}\sin\boldsymbol{\beta}_{i,c}\right\|^2 + \left\|\mathbf{F}_c{}^{l+1}\mathbf{c}_{i,z} - \boldsymbol{\xi}_z - c^l\mathbf{d}_{i,c}\cos\boldsymbol{\beta}_{i,c}\right\|^2. \tag{12}$$

The minimization (12) is simply a projection of $({}^{l+1}\mathbf{c}_{i,x}, {}^{l+1}\mathbf{c}_{i,y}, {}^{l+1}\mathbf{c}_{i,z})$ onto ellipsoids centered at $(\boldsymbol{\xi}_x, \boldsymbol{\xi}_y, \boldsymbol{\xi}_z)$ and has a closed-form solution [25]. Similarly, we can obtain $({}^{l+1}\boldsymbol{\alpha}_{i,v}, {}^{l+1}\boldsymbol{\beta}_{i,v})$ and $({}^{l+1}\boldsymbol{\alpha}_{i,a}, {}^{l+1}\boldsymbol{\beta}_{i,a})$.

Step (S3): The optimization over $\mathbf{d}_{i,c}$ involves solving the following QP:

$${}^{l+1}\mathbf{d}_{i,c} = \arg\min_{\mathbf{d}_{i,c} \geq 1} \left\|\mathbf{F}_c{}^{l+1}\mathbf{c}_{i,x} - \boldsymbol{\xi}_x - a\mathbf{d}_{i,c}\cos{}^{l+1}\boldsymbol{\alpha}_{i,c}\sin{}^{l+1}\boldsymbol{\beta}_{i,c}\right\|^2 + \left\|\mathbf{F}_c{}^{l+1}\mathbf{c}_{i,y} - \boldsymbol{\xi}_y - b\mathbf{d}_{i,c}\sin{}^{l+1}\boldsymbol{\alpha}_{i,c}\sin{}^{l+1}\boldsymbol{\beta}_{i,c}\right\|^2 + \left\|\mathbf{F}_c{}^{l+1}\mathbf{c}_{i,z} - \boldsymbol{\xi}_z - c\mathbf{d}_{i,c}\cos{}^{l+1}\boldsymbol{\beta}_{i,c}\right\|^2. \tag{13}$$

Each element of $\mathbf{d}_{i,c}$ is decoupled from each other. Thus (13) reduces to parallel single variable QPs, each of which can be solved in closed form. We clip the resulting solution to $(1, \infty)$ to satisfy the lower bound on $\mathbf{d}_{i,c}$ (recall the definition of $\mathcal{C}_{\zeta_{i,3}}$). We obtain $\mathbf{d}_{i,v}$ and $\mathbf{d}_{i,a}$ in a similar fashion with their respective clipping bounds.

Step (S4): The slack variables $\mathbf{s}_i$ is updated based on [27].

Step (S5): The Lagrange multipliter $\boldsymbol{\lambda}_i$ is updated using the approach presented in [26].

D. Incorporating BF Constraints

In (7), $d_{i,c}[k]$=1 corresponds to the boundary of the feasible set of the collision avoidance constraints (1f). In other words, $d_{ij,c}[k]$=1 ensures $h_{ij}[k]$=0. Along similar lines, $d_{ij,c}[k] < 1$ would correspond to the interior of the set. With this insight, we can define the polar reformulation of BF constraints in the following form:

$$\mathcal{C}_{ij,bf}[k] = \{\mathbf{p}_i[k] \mid \mathbf{f}_{ij,c}[k] = 0, \\ d_{ij,c}[k] \geq 1 + (1-\gamma)(d_{ij,c}[k-1] - 1)\},\ \forall k, j, \quad (14)$$

By comparing (7) and (14), we can see that the constraints differ only in the feasible region definition of $d_{ij,c}[k]$. When γ=1, the constraints are equivalent.

We integrate (14) into Algorithm 1 through a minor modification in *Step* (S3), specifically QP (13). Let ${}^{l}d_{ij,c}[k]$ be the value of $d_{ij,c}[k]$ obtained at the l-th iteration of Algorithm 1. We can use this value to approximate the feasible region of $d_{ij,c}[k]$ for BF constraints at the $(l+1)$-th iteration as

$$d_{ij,c}[k] \geq 1 + (1-\gamma)({}^{l}d_{ij,c}[k-1] - 1). \quad (15)$$

The right-hand side of (15) is constant, and thus the feasible region of $d_{ij,c}[k]$ for BF constraints is approximated through a simple lower bound. We can now just solve the QP (13) and clip the obtained value to this lower bound to solve for the optimal $d_{ij,c}[k]$ at iteration $(l+1)$.

IV. Simulation Results

This section provides a simulation analysis and comparison of our approach against the state-of-the-art baselines [7], [9], [20]. We denote our approach as "Ours (Quadratic)", and the "Quadratic" in the parenthesis refers to quadratic kinematic constraints. The proposed approach and the baselines are implemented in C++. The codes are available here [28]. All the simulations were executed on a PC with Intel Xeon CPU with 8 cores and 16 GB of RAM, running at 3 GHz.

The prediction horizon length is set to K=30 with a discretization of $0.1s$. The Bernstein polynomials used to parameterize the position trajectories have a degree of n=10. In the trajectory optimization problem (1a)-(1f), we set w_g=7000, w_s=100, κ=5 and penalize the acceleration $(q$=2) trajectory in the cost. In the constraints, we set $\overline{v}$=1.73ms^{-1}, $\underline{f}$=0.3g and $\overline{f}$=1.5g. For collision avoidance with neighbouring quadrotors, we set $\mathbf{\Theta}_{ij}$=diag$(0.17m, 0.17m, 0.45m)$, but a collision is declared with $\mathbf{\Theta}_{coll}$=diag$(0.13m, 0.13m, 0.40m)$. A quadrotor j is considered to be a potential conflict for quadrotor i if at any prediction step of the horizon, $\left\|(\mathbf{\Theta}_{ij} + \mathbf{\Theta}_p)^{-1}(\mathbf{p}_i[k] - \boldsymbol{\xi}_j[k])\right\|^2 \geq$1 holds, where $\mathbf{\Theta}_p$=diag$(0.2m, 0.2m, 0.2m)$. Similarly, we choose appropriate parameters for the collision constraints with obstacles. In Algorithm 1, we set maxiter=2000, thresold=0.01, and the penalty parameter ρ=$min(1.3^l, 5e10^5)$, where l is the iteration count of the algorithm.

A. Distributed Swarm Baselines

We compare our proposed approach Ours (Quadratic) with γ=1 with three different distributed swarm baselines. 1) SCP (On-demand) [8]: This approach relies on linearization of collision avoidance constraints and axis-wise decoupling of the kinematic bounds. It uses a so-called On-demand strategy where it tries to resolve only the first predicted collision. 2) SCP (Continuous) [9]: This approach is similar to SCP (On-demand) but it adds collision constraints over the entire prediction horizon. 3) Ours (Axiswise): This approach is the same as Ours (Quadratic) in the sense that it does not rely on linearizing collision constraints but has axis-wise kinematic bounds. The maximum and minimum values for the axis-wise velocity bounds are $\pm\overline{v}/\sqrt{3}$, and the acceleration limits can be computed such that it satisfies extreme cases (see (13) and (14) of [6]).

We consider a cluttered environment with 16 cylindrical static obstacles in a volume of $4m \times 4m \times 2m$ and vary swarms from 10 to 50. We tested the approaches in 100 configurations for each swarm size with randomized start-goal and obstacle positions. A trial is successful if all the quadrotors reach their assigned goal positions without collisions and under a time limit of $20s$.

B. Comparative Analysis

Success Rate: The first plot in Fig. 2 summarizes the improvement achieved by Ours (Quadratic) and Ours (Axiswise) over SCP (On-demand), and SCP (Continuous) in the success-rate metric. For swarm size up to 30, our approaches provide around 11%-62% improvement over SCP (On-demand) and 1%-19% over SCP (Continuous). As swarm size increase to 50, the performance gap between our approaches and the SCP (On-demand) and SCP (Continuous) swell to 39%-72%. The explanation is that the SCP baselines replace the non-convex collision constraints as hyperplane constraints, a conservative approximation of free space. Furthermore, SCP (On-demand) adds only a handful of collision constraints that result in unsafe separation and, ultimately, collisions in highly cluttered settings. For swarm size 50, we also see that Ours (Quadratic) has 15% more success rate than Ours (Axiswise) as the former has larger access to acceleration and velocity bounds.

Computation Time per Agent: The middle plot in Fig. 2 shows the average computation time per agent for the different swarm sizes. Both of our approaches have similar computation time per agent and are substantially faster than SCP (On-demand) and SCP (Continuous). For the swarm size of 10 case, we see that our approaches are 1.7 times faster than SCP (On-demand), and 18.7 times faster than SCP (Continuous). With a swarm size of 50, our approaches are still 1.7 times faster than SCP (On-demand) but are 42 times faster than SCP (Continuous). This excellent performance of our approaches can be attributed to the fact that AM optimizer of Algorithm 1 requires us to solve only one equality-constrained QP per iteration. On the contrary, the SCP (Continuous) solves constrained QP with a large number of inequality constraints. Moreover, it has to incorporate one slack variable per collision constraint to prevent potential in-feasibility in the optimization problem. The strategy of incorporating only the most imminent collision in SCP (On-demand) offers improvement in computation time per agent but at the expense of success rate as described previously.

Mission Time: The rightmost plot in Fig. 2 presents the average time taken by the swarm to reach their desired goal

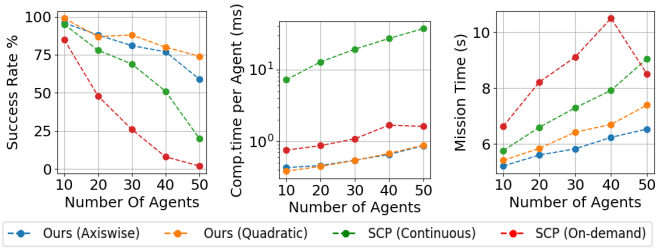

Fig. 2. Average performance comparison of approaches in point-to-point transition setting with an increasing number of swarm sizes in a fixed volume of $32m^3$ and with 16 static obstacles. 100 configurations were run for each swarm size.

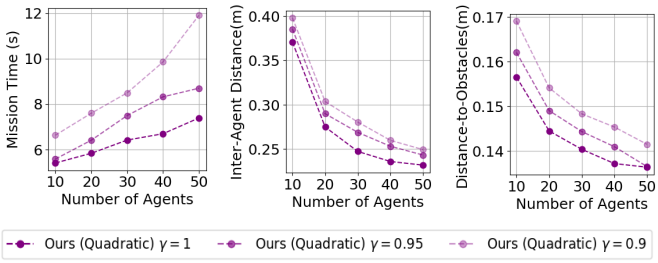

Fig. 3. Average performance comparison of our approach in point-to-point transition setting with different values of γ in the barrier function constraints. The environment is of $32m^3$ in volume and has 16 static obstacles. For each swarm size, 100 configurations were executed.

positions. We see that SCP (On-demand) performs the worst as the on-demand strategy leads to agents being closer to each other and obstacles, thus, slowing down the progress toward their goals. SCP (Continuous) performs relatively better than SCP (On-demand), but our approaches have the lowest mission times. For swarm size 40, we see that Ours (Quadratic) on average shows 36% and 15% reduction in mission time than SCP (On-demand) and SCP (Continuous), respectively. Interestingly, the trends show that Ours (Axis-wise), on average, completed the task $0.5s$-$0.9s$ faster than Ours (Quadratic).

C. Trade-off Between Performance and Safety

Figure 3 showcases the performance comparison of Ours (Quadratic) with three different values of γ. We ran the same 100 configurations and recorded the smallest inter-agent distance and distance-to-obstacles observed at each time step. With γ=0.95, we see 3.98%-8.69% improvement in inter-agent distances over γ=1. The improvements increase to 7.11%-12.35% with a more conservative γ=0.9. We see a similar trend in distance-to-obstacles. This improvement in clearance comes at the expense of increased mission time (see first plot in Fig. 3). We also observed that the computation time per agent increased with decreasing γ. For swarm size 50, the average computation time per agent is $1.61ms$, 82% increase over γ=1. Nevertheless, our approach is still real-time.

D. Off-the-shelf Solver Baseline

We now compare Ours (Quadratic) against the optimal control solver ACADO [20]. ACADO is provided with the original BF constraints (2), while Ours (Quadratic) uses the reformulation presented in (14). We ran an antipodal position exchange with a swarm size of 2, 4, 6, and 8 with no static obstacles. Figure 4 presents the observed metrics. We see that the computation time per agent scaling for Ours (Quadratic) is linear, while for ACADO, it increases quadratically with the number of agents. Furthermore, Ours (Quadratic) can complete the task faster than ACADO.

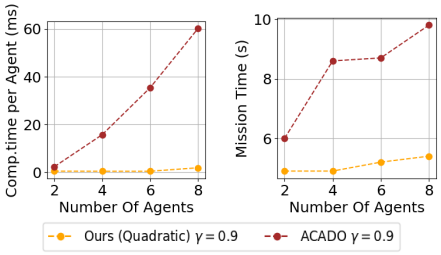

Fig. 4. Performance comparison of our approach and ACADO with BF constraints ($\gamma = 0.9$) in antipodal position exchange scenario with no static obstacles. ACADO could accomodate only a maximum of 8 agents.

V. Experimental Evaluation

We tested our approach on our Crazyflie 2.0 swarm testbed. The quadrotors' trajectories are computed on a single computer, and we send position and velocity trajectories to the underlying lower controller based on [29]. More details about the testbed can be found here [28].

A video summarizing the experimental results can be found here: `http://tiny.cc/AMSwarmVideo`. The algorithm is tested on various challenging scenarios. First, a 12 quadrotor swarm performs a head-on transition where a single quadrotor is in conflict with other 11 quadrotors at a time in an obstacle-free environment. Second, we repeat the same transition but with 6 static obstacles in the environment. We see that our approach navigates the quadrotors to their desired goals in an agile and smooth manner. Third, we qualitatively show a 12 drone random transition in an obstacle-free setting with two values of γ. We see quadrotors with conservative γ show a safe, evasive behaviour. Lastly, a formation of 8 quadrotors performs a transition in the presence of an unpredictable human. With the help of BF constraints, each quadrotor is able to navigate around the human safely.

VI. Conclusion

We presented a novel contribution toward making quadrotor swarm navigation more reliable and scalable. We showed how to formulate the original problem with quadratic collision and kinematic constraints as a QP without relying on conservative approximations. Furthermore, our approach can naturally handle more sophisticated collision avoidance constraints based on discrete-time BFs. In simulation, our optimizer significantly outperformed SCP-based approaches that have been common in recent works. Similarly, our approach also proved to be more computationally efficient than the state-of-the-art optimal control solver ACADO when considering BF constraints. In experiments, we showed the efficacy of the proposed algorithm in challenging scenarios in both obstacle-free and cluttered environments.

REFERENCES

[1] L. Marconi, C. Melchiorri, M. Beetz, D. Pangercic, R. Siegwart, S. Leutenegger, R. Carloni, S. Stramigioli, H. Bruyninckx, P. Doherty *et al.*, "The SHERPA project: Smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments," in *Proc. of the IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2012, pp. 1–4.

[2] P. Schmuck and M. Chli, "Multi-UAV collaborative monocular SLAM," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3863–3870.

[3] S. Tang and V. Kumar, "Mixed integer quadratic program trajectory generation for a quadrotor with a cable-suspended payload," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2216–2222.

[4] S.-J. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A survey on aerial swarm robotics," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 837–855, 2018.

[5] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *Proc. of the IEEE European Control Conference (ECC)*, 2001, pp. 2603–2608.

[6] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *Proc. of the IEEE/RSJ international conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 1917–1922.

[7] C. E. Luis and A. P. Schoellig, "Trajectory generation for multiagent point-to-point transitions via distributed model predictive control," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 375–382, 2019.

[8] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 604–611, 2020.

[9] E. Soria, F. Schiano, and D. Floreano, "Distributed predictive drone swarms in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 73–80, 2021.

[10] ——, "Predictive control of aerial swarms in cluttered environments," *Nature Machine Intelligence*, vol. 3, no. 6, pp. 545–554, 2021.

[11] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, online collision avoidance for dynamic vehicles using buffered voronoi cells," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1047–1054, 2017.

[12] X. Zhou, J. Zhu, H. Zhou, C. Xu, and F. Gao, "Ego-swarm: A fully autonomous and decentralized quadrotor swarm system in cluttered environments," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 4101–4107.

[13] J. Park, D. Kim, G. C. Kim, D. Oh, and H. J. Kim, "Online distributed trajectory planning for quadrotor swarm with feasibility guarantee using linear safe corridor," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4869–4876, 2022.

[14] D. Morgan, G. P. Subramanian, S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic guidance of distributed systems using sequential convex programming," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014, pp. 3850–3857.

[15] J. Zeng, B. Zhang, and K. Sreenath, "Safety-critical model predictive control with discrete-time control barrier function," in *Prof. of the IEEE American Control Conference (ACC)*, 2021, pp. 3882–3889.

[16] P. Mali, K. Harikumar, A. K. Singh, K. M. Krishna, and P. Sujit, "Incorporating prediction in control barrier function based distributive multi-robot collision avoidance," in *Proc. of the IEEE European Control Conference (ECC)*, 2021, pp. 2394–2399.

[17] V. Varun, A. P. Vinod, and S. Kolathaya, "Motion planning with dynamic obstacles using convexified control barrier functions," in *Proc. of the IEEE Indian Control Conference (ICC)*, 2021, pp. 81–86.

[18] Y. Chen, A. Singletary, and A. D. Ames, "Guaranteed obstacle avoidance for multi-robot operations with limited actuation: A control barrier function approach," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 127–132, 2020.

[19] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[20] B. Houska, H. J. Ferreau, and M. Diehl, "Acado toolkit—an open-source framework for automatic control and dynamic optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[21] W. Hönig, J. A. Preiss, T. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory planning for quadrotor swarms," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 856–869, 2018.

[22] M. W. Mueller and R. D'Andrea, "A model predictive controller for quadrocopter state interception," in *Proc. of the IEEE European Control Conference (ECC)*, 2013, pp. 1383–1389.

[23] D. Morgan, G. P. Subramanian, S.-J. Chung, and F. Y. Hadaegh, "Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1261–1285, 2016.

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

[25] V. K. Adajania, H. Masnavi, F. Rastgar, K. Kruusamae, and A. K. Singh, "Embedded hardware appropriate fast 3d trajectory optimization for fixed wing aerial vehicles by leveraging hidden convex structures," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 571–578.

[26] G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein, "Training neural networks without gradients: A scalable admm approach," in *Proc. of the International Conference on Machine Learning (ICML)*, 2016, pp. 2722–2731.

[27] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson, "Optimal parameter selection for the alternating direction method of multipliers (admm): quadratic problems," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 644–658, 2014.

[28] V. K. Adajania, S. Zhou, A. K. Singh, and A. P. Schoellig, "AMSwarm," https://github.com/utiasDSL/AMSwarm, 2022.

[29] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 2520–2525.