2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

AMSwarmX: Safe Swarm Coordination in CompleX Environments via Implicit Non-Convex Decomposition of the Obstacle-Free Space

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

***Abstract*— Quadrotor motion planning in complex environments leverage the concept of safe flight corridor (SFC) to facilitate static obstacle avoidance. Typically, SFCs are constructed through convex decomposition of the environment's free space into cuboids, convex polyhedra, or spheres. However, such SFCs can be overly conservative when dealing with a quadrotor swarm, substantially limiting the available free space for quadrotors to coordinate. This paper presents an Alternating Minimization-based approach that does not require building a conservative free-space approximation. Instead, both static and dynamic collision constraints are treated in a unified manner. Dynamic collisions are handled based on shared position trajectories of the quadrotors. Static obstacle avoidance is coupled with distance queries from the Octomap, providing an implicit non-convex decomposition of free space. As a result, our approach is scalable to arbitrary complex environments. Through extensive comparisons in simulation, we demonstrate a 60% improvement in success rate, an average $1.8\times$ reduction in mission completion time, and an average $23\times$ reduction in per-agent computation time compared to SFC-based approaches. We also experimentally validated our approach using a Crazyflie quadrotor swarm of up to 12 quadrotors in obstacle-rich environments. The code, supplementary materials, and videos are released for reference.**

I. Introduction

The excellent maneuverability and agility of quadrotors make them very popular for applications such as search and rescue missions [1], environmental mapping and monitoring [2], and payload transport [3]. While single quadrotors are impressive, quadrotor swarms offer even greater advantages, including increased flexibility, efficiency, and robustness [4]. As the quadrotors operate in a shared space, they must *coordinate* among themselves to resolve conflicts while also *avoiding static obstacles* present in the environment.

Trajectory optimization approaches for coordinating quadrotor swarms use the ellipsoidal collision avoidance constraints [5] for inter-agent collision avoidance. These approaches can be divided into two categories: centralized and distributed. Centralized approaches [6]–[8] solve a joint trajectory optimization problem for all the quadrotors. Despite offering an extended solution space, these approaches become computationally intractable when dealing with a larger number of quadrotors.

Distributed approaches such as [9]–[11] provide a scalable alternative. In these approaches, each quadrotor independently solves an optimization problem. The collision avoidance constraints are formulated based on the trajectories shared by neighbouring quadrotors. As shown in [11], the independent optimization problem is a non-convex Quadratically Constrained Quadratic Program (QCQP), arising from non-convex quadratic ellipsoidal collision avoidance and kinematic feasibility constraints. Existing distributed approaches [9], [10], [12] rely on affine approximations: linearizing the collision avoidance constraints and axis-wise decoupling of kinematic constraints. These approximations result in a QP but with small feasible sets. Our previous work [11] showed how to avoid these approximations and still obtained a QP, achieving superior inter-agent collision avoidance performance.

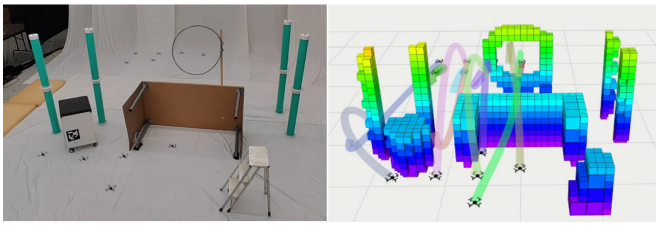

Fig. 1. Twelve quadrotors performing a position exchange in a complex environment. Link to video: `http://tiny.cc/AMSwarmXVideo`. Link to code and supplementary material: `https://github.com/utiasDSL/AMSwarmX`.

To navigate a single quadrotor in complex 3D environments, many works have extensively employed the concept of Safe Flight Corridor (SFC) for static obstacle avoidance. Existing works perform convex decomposition of the free space to obtain SFCs, which serve as additional constraint sets in trajectory optimization. Several examples of such convex constraint sets include cuboid [13], spheres [14], and convex polyhedra [15].

SFC-based approaches also utilize high-level path planners such as A* and RRT* to generate a guiding path for trajectory optimization. Some methods [13]–[15] construct a safe corridor around this guiding pah, while others [10] rely on trajectories from previous planning steps. There exists Gradient Descent (GD) based approaches [16], [17] that directly incorporate the distance to obstacles, but as a cost, which limits their generalization to different environments.

SFC-based approaches are also popular in both centralized and distributed swarm setups. In [18], [19], convex polyhedra are employed for each quadrotor, and authors in [10], [20] use cuboids to avoid the static obstacles, while inter-agent collision avoidance is formulated via ellipsoidal collision avoidance as previously mentioned. The convex polyhedra and cuboid corridors are represented as affine inequalities

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. Emails: {vivek.adajania, siqi.zhou}@robotics.utias.utoronto.ca, arun.singh@ut.ee, and angela.schoellig@tum.de.

and can be incorporated into the QP trajectory generation framework. However, decomposing the free space into these convex regions can be conservative in tight environments, resulting in less free space available for the quadrotors to negotiate around each other.

Contribution: In this work, we propose an Alternating Minimization (AM) approach that treats static and dynamic collision avoidance constraints in a unified manner yet scales to arbitrary environments. The dynamic inter-agent collisions are handled based on the shared trajectories of the neighbours while the static obstacle constraints are coupled to the distance queries from the Octomap. We show that our approach implicitly constructs a non-convex decomposition of the free space that is much larger than that of explicit convex decomposition approaches. We compare our approach with state-of-the-art distributed SFC-based approaches from [11], [15], [20]. Our simulation comparison shows a 60% improvement in success rate, on average a 1.8× reduction in mission completion time, and on average a 23× reduction in per-agent computation time.

II. Distributed Motion Planning Problem

Our objective is to generate smooth, collision-free, and kinematically feasible trajectories that guide N quadrotors from their initial positions $\mathbf{p}_{i,o}$ to their desired goal positions $\mathbf{p}_{i,g}$ within a cluttered and complex environment. The vector $\mathbf{p} = [x, y, z]^T$ represents the three-dimensional position of a quadrotor, with the subscript i denoting the quadrotor index and the subscripts o and g indicating initial and goal variables. Similar to our previous work [11], we formulate the motion planning for the quadrotor swarm as a distributed trajectory optimization problem. We assume that each quadrotor can communicate with its neighbour without any communication loss or delay. We also assume that each quadrotor has access to a prior map of the environment.

A. Problem Formulation

We describe the optimization problem that needs to be solved by quadrotor i at each planning step:

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

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

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

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

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

$$\mathbf{p}_i[k] \in \mathcal{C}_{free}, \; \forall k, \quad (1f)$$

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 error-to-goal cost applied 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 of respective 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)-(1d) enforce bounds on the velocity $(0, \overline{v})$, and the acceleration $(\underline{f}, \overline{f})$. The inequalities (1e) enforce the collision avoidance with the j-th neighbouring quadrotor with position $\boldsymbol{\xi}_j[k]$. Note that $\boldsymbol{\xi}_j[k]$ is known since the quadrotor communicates the trajectory they computed at the previous planning step to their neighbours. The constant s is the radius of the sphere modelling the quadrotor. $\mathbf{\Theta}_{ij}$ is a diagonal matrix with $(1, 1, 2)$ characterizing an ellipsoidal envelope in the inter-agent collision avoidance. The vector $\mathbf{g} = [0,\ 0,\ g]^T$ is the gravitational acceleration vector, where g is the acceleration due to gravity. Constraint (1f) enforces the quadrotor to remain in the space not occupied by the static obstacles in the environment.

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

$$\begin{bmatrix} x_i[0] & x_i[1] & \dots & x_i[K-1] \end{bmatrix}^T = \mathbf{W}\mathbf{c}_{i,x}, \quad (2)$$

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 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.

III. Main Algorithmic Results

This section presents our main algorithmic results. We first describe our novel static obstacle avoidance model and its integration into the AM approach. We then discuss how a discrete path planner can be leveraged to obtain some of the hyperparameters of our trajectory optimizer.

A. Static Obstacle Avoidance Constraints

Referring to Fig. 2, let $\mathbf{p}_r$ denote a known position in obstacle-free space, and $\mathbf{p}$ represent an arbitrary position in 2D space. We will discuss the possible choices for these positions later in this section. Nevertheless, the condition for $\mathbf{p}$ to be obstacle-free can be expressed as follows:

$$\|\mathbf{p} - \mathbf{p}_r\|^2 \leq (d_r^*(\alpha_r(\mathbf{p})))^2, \quad (3)$$

where $d_r^*(\alpha_r(\mathbf{p}))$ represents the obstacle clearance from $\mathbf{p}_r$ in the direction of α_r. We refer to d_r^* as directional clearance and $\mathbf{p}_r$ as the attractor position. Notably, the directional clearance depends on α_r, which, in turn, relies on $\mathbf{p}$. Moreover, it can be easily obtained through ray-casting on an Octomap while accounting for the dimension of the quadrotor.

As shown in Fig. 2, the constraint (3) implicitly characterizes a non-convex obstacle-free space (shown in purple) in the vicinity of $\mathbf{p}_r$. Existing works often involve computing an explicit convex decomposition of this space, referred to as SFC. For example, two such decompositions in the form of an axis-aligned rectangle and convex polygon are shown in

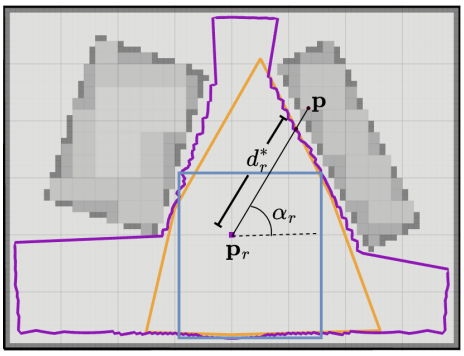

Fig. 2. The figure shows the non-convex obstacle-free region (purple) obtained by Octomap distance queries (ray-casting) from a known obstacle-free position $\mathbf{p}_r$. Our AM-based approach provides a tractable way of incorporating the purple region into the trajectory optimization. Existing SFC-based approaches decompose the free space into conservative convex sets such as the rectangle [10] (blue) and convex polygon [15] (orange).

Fig. 2. Clearly, such decompositions are overly conservative, covering only a fraction of the actual obstacle-free space. This severely limits the range of feasible motions for the quadrotor swarms.

We can extend (3) to 3D by formulating it for quadrotor i at time step k as follows:

$$\|\mathbf{p}_i[k] - \mathbf{p}_{i,r}\|^2 \leq (d^*_{i,r}(\alpha_{i,r}(\mathbf{p}_i), \beta_{i,r}(\mathbf{p}_i))[k])^2, \ \forall k. \quad (4)$$

Here, $\mathbf{p}_{i,r}$ represents the attractor position associated with quadrotor i. The scalar $d^*_{i,r}(\alpha_{i,r}(\mathbf{p}_i), \beta_{i,r}(\mathbf{p}_i))[k]$ denotes the directional clearance from the attractor position in directions $(\alpha_{i,r}[k], \beta_{i,r}[k])$ at each prediction step k. We drop the parenthesis and refer to directional clearance as $d^*_{i,r}[k]$.

Incorporating constraints of the form (4) into the optimization problem poses a challenge due to the absence of an analytical, functional form for the directional clearance $d^*_{i,r}$. The subsequent subsections elaborate on how our AM-based trajectory optimizer provides an effective workaround. The key intuition has two core components. First, we initially treat $\alpha_{i,r}[k], \beta_{i,r}[k]$ as independent of $\mathbf{p}_i[k]$ and then gradually enforce their dependency as the optimizer iterations progress (see (8)). Second, at every step of our AM, we fix $d^*_{i,r}[k]$ based on some guess of $\alpha_{i,r}[k], \beta_{i,r}[k]$ and we gradually update these guess across iterations. Moreover, we leverage the fact that (4) becomes a convex quadratic constraint if we fix $d^*_{i,r}[k]$ on the right-hand side.

B. Constraint Reformulation

The static obstacle avoidance constraints (4), inter-agent constraints (1e), acceleration constraints (1d), and velocity constraints (1c) are inherently quadratic in nature. Solving trajectory optimization with these constraints necessitates tackling expensive QCQPs. In this subsection, we undertake the task of reformulating these constraints into a polar form [11], ultimately enabling us to achieve a QP structure without the need for linearization. We express all these constraints as distinct sets:

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

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

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

$$\mathcal{C}_{i,r}[k] = \{\mathbf{p}_i[k] \in \mathbb{R}^3 \mid \mathbf{f}_{i,r}[k] = 0, d_{i,r}[k] \leq d^*_{i,r}[k]\}, \ \forall k. \quad (8)$$

In these sets, we introduce functions $\mathbf{f}_{i,v}$, $\mathbf{f}_{i,a}$, $\mathbf{f}_{ij,c}$, and $\mathbf{f}_{i,r}$ defined as follows:

$$\begin{aligned}
\mathbf{f}_{i,v}[k] &= \dot{\mathbf{p}}_i[k] - d_{i,v}[k] \cdot \boldsymbol{\omega}(\alpha_{i,v}[k], \beta_{i,v}[k]),\\
\mathbf{f}_{i,a}[k] &= \ddot{\mathbf{p}}_i[k] + \mathbf{g} - d_{i,a}[k] \cdot \boldsymbol{\omega}(\alpha_{i,a}[k], \beta_{i,a}[k]),\\
\mathbf{f}_{ij,c}[k] &= (2s\boldsymbol{\Theta}_{ij})^{-1}(\mathbf{p}_i[k] - \boldsymbol{\xi}_j[k])\\
&\quad - d_{ij,c}[k] \cdot \boldsymbol{\omega}(\alpha_{ij,c}[k], \beta_{ij,c}[k]),\\
\mathbf{f}_{i,r}[k] &= \mathbf{p}_i[k] - \mathbf{p}_{i,r} - d_{i,r}[k] \cdot \boldsymbol{\omega}(\alpha_{i,r}[k], \beta_{i,r}[k]),\\
\boldsymbol{\omega}(\alpha_{(.)}, \beta_{(.)}) &= [\cos\alpha_{(.)} \sin\beta_{(.)}, \sin\alpha_{(.)} \sin\beta_{(.)}, \cos\beta_{(.)}]^T.
\end{aligned}$$

Notably, the parameters $(\alpha_{(.)}, \beta_{(.)}, d_{(.)})$ represent the polar form representations of the constraints and will be computed by our optimizer concurrently with the trajectory [11].

It is worth pointing out that our reformulation of inter-agent collision and static obstacle avoidance given by $\mathbf{f}_{ij,c}$ and $\mathbf{f}_{i,r}$ respectively have the same structure. The only difference stems from the fact that for the former, the feasible space of $d_{ij,c}[k]$ is completely defined analytically. In contrast, evaluating the feasibility of $d_{i,r}[k]$ requires distance queries from the Octomap.

C. Reformulated Problem

We can express the cost function (1a), initial conditions (1b), and the polar constraints (5)-(8) as a concise optimization problem:

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

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

$$\boldsymbol{\zeta}_{i,1} \in \mathcal{C}_{\boldsymbol{\zeta}_{i,1}}, \quad \boldsymbol{\zeta}_{i,3} \in \mathcal{C}_{\boldsymbol{\zeta}_{i,3}} \quad (9c)$$

Here, $\boldsymbol{\zeta}_{i,1} = [\mathbf{c}_{i,x}^T, \mathbf{c}_{i,y}^T, \mathbf{c}_{i,z}^T]^T$, $\boldsymbol{\zeta}_{i,2} = [\boldsymbol{\alpha}_{i,c}^T, \boldsymbol{\alpha}_{i,a}^T, \boldsymbol{\alpha}_{i,v}^T, \boldsymbol{\alpha}_{i,r}^T, \boldsymbol{\beta}_{i,c}^T, \boldsymbol{\beta}_{i,a}^T, \boldsymbol{\beta}_{i,v}^T, \boldsymbol{\beta}_{i,r}^T]^T$, and $\boldsymbol{\zeta}_{i,3} = [\mathbf{d}_{i,c}, \mathbf{d}_{i,a}, \mathbf{d}_{i,v}, \mathbf{d}_{i,r}]^T$ represent the optimization variables. Note that the different $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\mathbf{d}$ are formed by stacking the respective variables at different prediction steps. For, example $\boldsymbol{\alpha}_{i,r}$ is formed by stacking $\alpha_{i,r}[k]$ for different k. The matrix $\mathbf{A}$ and vector $\mathbf{b}$ arise from the equality constraints in the polar form. The matrix $\mathbf{Q}$ and vector $\mathbf{q}$ pertain to the cost function. The sets $\mathcal{C}_{\boldsymbol{\zeta}_{i,1}}$ and $\mathcal{C}_{\boldsymbol{\zeta}_{i,3}}$ correspond to the initial boundary conditions and the inequality constraints from the polar form, respectively. Please refer to Section III-B of [11] for details.

D. Relaxation and Solution using AM

Our solution process consists of two core steps. First, we relax the equality constraints in the reformulated problem by incorporating them as penalties into the cost function as follows:

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

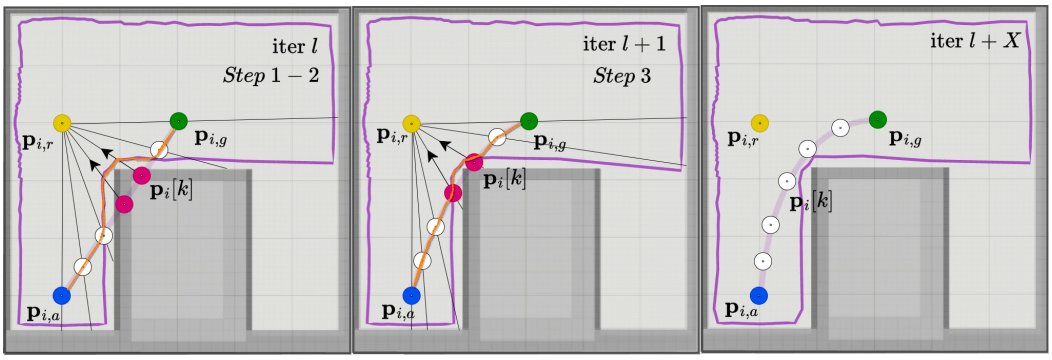

Fig. 3. Graphical description of the different steps of our AM-based optimizer. The purple line shows the obstacle-free space estimated by a 360-degree Octomap distance query from the attractor position $\mathbf{p}_{i,r}$. Steps 1 and 2 (left) jointly define an approximation (orange line) of the boundary of the obstacle-free boundary. In step 3, the colliding positions (red) are pushed towards the free space using the boundary estimated in steps 1-2. The updated trajectory of step 3 is used to further refine the free-space boundary estimate in the subsequent iterations. The rightmost figure shows the final output from our optimizer.

Here, the penalty parameter ρ and the Lagrange multiplier $\boldsymbol{\lambda}_i$ control the residual of the equality constraints. Next, we apply the AM technique to the relaxed problem. In the following, left superscript l is used to track the values of the variable across iterations. That is, ${}^{l}(.)$ represents the value of $(.)$ at iteration l.

Step 1: At iteration $l+1$, we fix ${}^{l}\boldsymbol{\zeta}_{i,1}$ and ${}^{l}\boldsymbol{\zeta}_{i,3}$ to the values obtained at iteration l and only optimize over $\boldsymbol{\zeta}_{i,2}$. Individual optimizations over different $\boldsymbol{\alpha}, \boldsymbol{\beta}$ in $\boldsymbol{\zeta}_{i,2}$ can be decoupled into parallel problems with similar structures. For example, the optimization problem for $(\boldsymbol{\alpha}_{i,r}, \boldsymbol{\beta}_{i,r})$ can be reduced to (while ignoring $d^*_{i,r}$ dependency for now):

$$
\begin{aligned}
{}^{l+1}\boldsymbol{\alpha}_{i,r},\, {}^{l+1}\boldsymbol{\beta}_{i,r} = \arg \min_{\boldsymbol{\alpha}_{i,r}, \boldsymbol{\beta}_{i,r}} \\
\left\| \mathbf{W}^{l}\mathbf{c}_{i,x} - \mathbf{x}_{i,r} - {}^{l}\mathbf{d}_{i,r} \cos\boldsymbol{\alpha}_{i,r} \sin\boldsymbol{\beta}_{i,r} \right\|^2 \\
+ \left\| \mathbf{W}^{l}\mathbf{c}_{i,y} - \mathbf{y}_{i,r} - {}^{l}\mathbf{d}_{i,r} \sin\boldsymbol{\alpha}_{i,r} \sin\boldsymbol{\beta}_{i,r} \right\|^2 \\
+ \left\| \mathbf{W}^{l}\mathbf{c}_{i,z} - \mathbf{z}_{i,r} - {}^{l}\mathbf{d}_{i,r} \cos\boldsymbol{\beta}_{i,r} \right\|^2 . \qquad (11)
\end{aligned}
$$

The solution to (11) can be derived in closed form based on purely geometric considerations [11]. As shown in Fig. 3 (left), this step can be geometrically interpreted as obtaining the Octomap distance query directions (black lines) from the attractor position.

Step 2: In this step, we solve for ${}^{l+1}\boldsymbol{\zeta}_{i,3}$ while using the known values for ${}^{l}\boldsymbol{\zeta}_{i,1}, {}^{l+1}\boldsymbol{\zeta}_{i,2}$. Thus, this step reduces to decoupled problems over $\mathbf{d}_{i,c}, \mathbf{d}_{i,r}, \mathbf{d}_{i,v}, \mathbf{d}_{i,a}$. For example, the optimization over $\mathbf{d}_{i,r}$ can be expressed as:

$$
\begin{aligned}
{}^{l+1}\mathbf{d}_{i,r} = \arg \min_{\mathbf{d}_{i,r} \le \mathbf{d}^*_{i,r}} \\
\left\| \mathbf{W}^{l+1}\mathbf{c}_{i,x} - \mathbf{x}_{i,r} - \mathbf{d}_{i,r} \cos{}^{l+1}\boldsymbol{\alpha}_{i,r} \sin{}^{l+1}\boldsymbol{\beta}_{i,r} \right\|^2 \\
+ \left\| \mathbf{W}^{l+1}\mathbf{c}_{i,y} - \mathbf{y}_{i,r} - \mathbf{d}_{i,r} \sin{}^{l+1}\boldsymbol{\alpha}_{i,r} \sin{}^{l+1}\boldsymbol{\beta}_{i,r} \right\|^2 \\
+ \left\| \mathbf{W}^{l+1}\mathbf{c}_{i,z} - \mathbf{z}_{i,r} - \mathbf{d}_{i,r} \cos{}^{l+1}\boldsymbol{\beta}_{i,r} \right\|^2 . \qquad (12)
\end{aligned}
$$

Since ${}^{l+1}\boldsymbol{\alpha}_{i,r}, {}^{l+1}\boldsymbol{\beta}_{i,r}$ have already been obtained in the previous step, we can use them to determine the directional clearance $\mathbf{d}^*_{i,r}$ from the Octomap queries for each prediction step k. Thus, (12) reduces to a QP with a closed form solution [11].

Step 3: In this step, we use the known values of ${}^{l+1}\boldsymbol{\zeta}_{i,2}, {}^{l+1}\boldsymbol{\zeta}_{i,3}$ to optimize over just $\boldsymbol{\zeta}_{i,1}$. This reduction has two important implications. First, fixing $\boldsymbol{\alpha}_{i,r}$ (from $\boldsymbol{\zeta}_{i,2}$) and $\mathbf{d}_{i,r}$ (from $\boldsymbol{\zeta}_{i,3}$) allows us to construct an estimate of the boundary of the obstacle-free space (orange strip in Fig. 3 (left)) as $\mathbf{p}_{i,r}[k] + d_{i,r}[k] \cdot \boldsymbol{\omega}[k]$ (recall (8)). Second, (10) is transformed into an equality-constrained QP with a closed-form solution. Moreover, geometrically, the effect of this QP is to push the trajectory positions in collision with the obstacle towards the boundary of the obstacle-free space (Fig. 3 (left)). The solution of this step will be fed to the next iteration. It will lead to further refinement of $\boldsymbol{\zeta}_{i,2}, \boldsymbol{\zeta}_{i,3}$ and consequently an updated definition of the boundary of the obstacle-free space (Fig.3 (middle)).

A few additional points about our AM-based approach are worth pointing out. As shown in Fig.3, the description of obstacle-free space (orange strip) as used by our optimizer is slightly conservative when compared to the true boundary (shown in purple). But importantly, by construction the estimated boundary overlaps with the true boundary for the colliding trajectory segment.

Step 4: The Lagrange multiplier $\boldsymbol{\lambda}_i$ is updated using the gradient of the penalty term [21]. We increment the penalty parameter ρ by $\Delta\rho$ and repeat Steps 1 to 4 until the residuals of the penalty term fall below a predefined threshold. A typical final output is presented in Fig. 3 (right). We recommend watching the video at the following link: `http://tiny.cc/AMIterViz`.

E. Visibility Condition

The performance of our AM optimizer depends on the position of the attractor $\mathbf{p}_{i,r}$. Empirically, we have observed the best performance when $\mathbf{p}_{i,r}$ is visible from both the current position $(\mathbf{p}_{i,a})$ and the goal position $(\mathbf{p}_{i,g})$, and vice versa. The intuition behind this condition is visualized in Fig. 4. In the left figure of Fig. 4, the visibility condition is met, ensuring that $\mathbf{p}_{i,r}$, $\mathbf{p}_{i,a}$, and $\mathbf{p}_{i,g}$ all lie within the obstacle-free space constructed around $\mathbf{p}_{i,r}$ using Octomap distance queries. In contrast, when the visibility condition is not met (for example, when the current position is outside the constructed obstacle-free space, as shown in the right

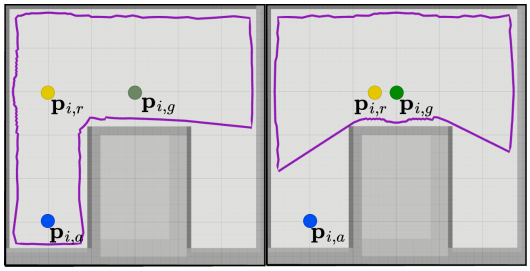

Fig. 4. The visibility condition says that the attractor position $\mathbf{p}_{i,r}$ should be visible from the current $\mathbf{p}_{i,a}$ and the goal position $\mathbf{p}_{i,g}$, and vice-versa. In the left figure, the condition is met indicating all three of them lie in the same feasible space of the static obstacle avoidance constraints. While in the right figure, the condition is not met and the AM algorithm may not find an obstacle-free solution.

figure), using the AM optimizer may lead to infeasible solutions. Similarly, if the goal position lies outside the feasible obstacle-free space, the quadrotor will not make any progress toward it, and the generated trajectory will remain confined to the feasible obstacle-free space (see accompanying video). We ensure the visibility condition by carefully selecting $\mathbf{p}_{i,r}$ and incorporating an intermediate goal selection routine within our pipeline, which is described in the following section.

F. Discrete Path Planning

We employ an off-the-shelf discrete path planner, A*, on the prior map and use a simple heuristic for meeting the visibility condition. This path planner generates an array $\mathbf{q}_{i,\text{gp}}$ consisting of obstacle-free positions that connect the current position to the final goal position. From this array, we follow a two-step process. We first select the last visible position from the current position within the array. This selected position becomes our attractor. Next, starting from the obtained attractor position, we choose the last visible position as our intermediate goal. As the quadrotor moves, we select a new attractor and intermediate goal at each planning step. Eventually, the intermediate goal would converge to the final goal position. Note that we account for the dimension of the quadrotor when checking for visibility.

G. Summary of Proposed Approach

Algorithm 1 describes all the components i^{th} quadrotor would use to navigate in a complex 3D environment. The input to the algorithm is the current state, desired goal position and map information. First, the quadrotor i receives planned trajectories from neighbouring quadrotors, allowing inter-agent collision avoidance constraints to be formulated. Second, it runs A* on the prior map to obtain a path. Third, from this path, an attractor and an intermediate goal position satisfying visibility conditions are selected for the formulation of static obstacle avoidance constraints. Finally, the quadrotor builds the reformulated problem and applies the AM optimizer to generate the trajectory. These steps are repeated in the next planning step.

IV. Validation and Benchmarking

This section presents a comprehensive comparison in a simulation of our proposed approach with state-of-the-art

Algorithm 1 `generateTrajectory`

Input Current state $\mathbf{p}_{i,a}^{(q)}$, Final goal $\mathbf{p}_{i,g}$, Octomap $\mathcal{W}$
Output Trajectory coefficients $\boldsymbol{\zeta}_{i,1}$

$\boldsymbol{\xi}_j \leftarrow$ `NeighbouringQuadrotorsTrajectories`
$\mathbf{q}_{i,gp} \leftarrow$ `runGridPlanner`$(\mathbf{p}_{i,a}, \mathbf{p}_{i,g}, \mathcal{W})$
$\mathbf{p}_{i,r} \leftarrow$ `selectAttractorPosition`$(\mathbf{q}_{i,gp}, \mathbf{p}_{i,a})$
$\mathbf{p}_{i,w} \leftarrow$ `selectGoalPosition`$(\mathbf{q}_{i,gp}, \mathbf{p}_{i,r})$
`buildReformulatedProblem`$(\mathbf{p}_{i,a}, \mathbf{p}_{i,w}, \mathbf{p}_{i,r}, \boldsymbol{\xi}_j)$
$\boldsymbol{\zeta}_{i,1} \leftarrow$ `alternatingMinimization`$(\mathcal{W})$
return $\boldsymbol{\zeta}_{i,1}$

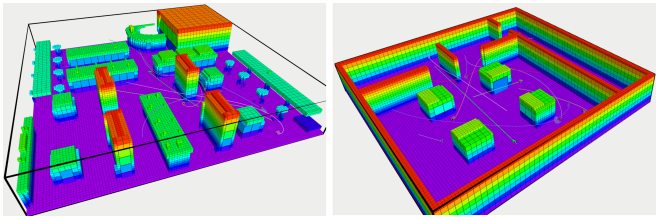

Fig. 5. We conduct a total of 60 simulation runs of different swarm planning approaches in a bookstore (left) and random room (right) environment. The dimension of the environment is $12m \times 12m \times 2.5m$ for the bookstore and $10m \times 10m \times 2m$ for the random room.

baselines [10], [11], [15] and experimental validation. All simulations and experiments were executed on a PC with Intel Xeon CPU with 8 cores and 16 GB of RAM, running at 3 GHz. Our simulation evaluation includes two complex environments: the "bookstore" and the "random room" see Fig. 5. We conducted a total of 60 trials, 30 in each environment, using swarm sizes from 10 to 50, and all with randomized start-goal positions. A trial is successful if all quadrotors reach their designated goal positions within a time limit of $60s$ while avoiding collisions. The C++ implementation, baselines, the parameters used, and additional environments can be found at [22]. We refer to our proposed approach as "AMSwarmX."

A. Distributed Swarm Baselines

We compare AMSwarmX with the following two distributed swarm baselines:

LSC-Planner [10]: This approach uses Octomap for environment representation and subsequent construction of SFC in the form of axis-aligned cuboids [20] for static obstacle avoidance. Inter-agent collision avoidance is achieved by incorporating ellipsoidal collision avoidance constraints, which are linearized using the convex hull property of the Bernstein polynomial. It also employs A* as a high-level path planner and chooses a visible position from current position on the planned path as the intermediate goal. The original approach includes deadlock resolution, but we exclude it in our comparison to focus on collision avoidance capabilities. Note that a deadlock resolution strategy can potentially enhance the performance of any approach. With this baseline, we showcase the advantages provided by a unified linearization-free treatment of static and dynamic collision avoidance.

AMSwarmED (combination of [11] and [15]): This baseline is a combination of our prior work [11] augmented with the ellipsoidal free space decomposition method to compute a convex polyhedron [15]. The use of high-level

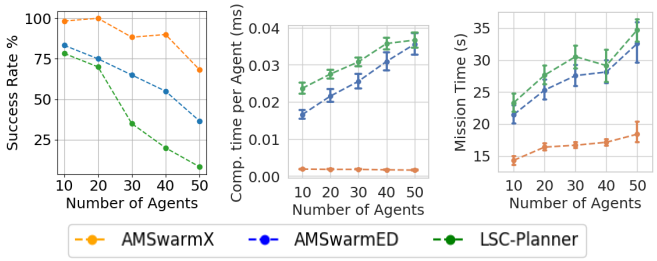

Fig. 6. Performance comparison of the different approaches in a point-to-point transition setting with an increasing swarm size. Sixty random trials were run for each swarm size, and the averages are plotted.

path planner and intermediate goals is the same as the previous baseline. Since the dynamic inter-agent collision avoidance part of [11] is the same as our current work, this baseline essentially validates the efficacy of our implicit non-convex free space decomposition.

B. Comparative Analysis

Success Rate: Fig. 6 (left) shows the improvement achieved by the AMSwarmX approach over AMSwarmED and LSC-Planner. For swarm sizes up to 20, the success rate of LSC-Planner is similar to that of AMSwarmED, while AMSwarmX outperforms both approaches. However, as the swarm size increases, the performance of LSC-Planner degrades significantly compared to AMSwarmED and AMSwarmX. As mentioned earlier, LSC-Planner linearizes the inter-agent collision avoidance constraints, leading to hyperplane constraints that are known to be conservative [11]. Also, the cuboid decomposition of obstacle-free space, especially in tight space, is conservative. Consequently, the quadrotors often end up in deadlocks. In contrast, both AMSwarmED and AMSwarmX can directly handle the quadratic form of inter-agent collision avoidance constraints in the optimizer. AMSwarmED performs better than LSC-Planner, as the convex polyhedron corridor provides larger free space for the quadrotors to maneuver. However, AMSwarmX consistently outperforms both approaches, showing an improvement of 15%-60%, validating the benefits of our proposed static obstacle avoidance strategy.

Computation Time: The middle plot in Fig. 6 shows the computation time per agent for all approaches. There is only a small (max. $20ms$) difference between the mean values for swarm sizes 10 to 50. This can be attributed to the distributed nature of the approaches, with each quadrotor solving its own optimization problem and considering only neighbouring quadrotors during optimization. AMSwarmX has the lowest average computation time per quadrotor, showing a $21\times$ and $23\times$ reduction compared to AMSwarmED and LSC-Planner, respectively. This is because AMSwarmX adds only one static obstacle avoidance constraint at each planning step. In contrast, AMSwarmED adds numerous hyperplane constraints stemming from the convex polyhedron, and similarly, LSC-Planner adds numerous hyperplane constraints stemming from cuboid corridors. Additionally, LSC-Planner employs numerous polynomial pieces, which increases the number of decision variables. It is important to note that the plot only shows the time required to solve the optimization problem, while the time taken to generate a discrete path and a SFC is in the sub-millisecond range.

Mission Time: The rightmost plot in Fig. 6 shows mission completion times. LSC-Planner performs the worst due to non-smooth transitions around corners caused by the cuboid corridors. AMSwarmED, benefiting from a better decomposition, exhibits smoother transitions compared to LSC-Planner. AMSwarmX achieves the smoothest transitions as it better captures the local shape of the free space (see accompanying video). Cuboid-shaped corridors and convex polyhedron are overly conservative in tight spaces, making it difficult for quadrotors to maneuver around each other, resulting in increased mission completion times. Overall, AMSwarmX demonstrates a time reduction of $1.75\times$ and $1.87\times$ over AMSwarmED and LSC-Planner, respectively.

C. Experimental Validation

We conducted the experimental validation of AMSwarmX using our Crazyflie 2.0 swarm testbed in two complex environments. Trajectories were computed on a single computer, with a CPU thread assigned to each quadrotor. These computed trajectories were transmitted to the lower-level controller at each planning step. We provided AMSwarmX with the Octomap representation. In both scenarios, the quadrotors perform three transitions: first, the quadrotors execute a position exchange; next, a random transition; and finally, they return to their original take-off positions. The average per-agent computation time, inter-agent distance, and distance to obstacles were found to be $4.4ms$, $0.43m$, and $0.37m$, respectively. The smallest inter-agent distance and distance to obstacles were $0.26m$ and $0.1m$, respectively. The quadrotors successfully complete the task without collisions. The demonstration video can be found here: `http://tiny.cc/AMSwarmXVideo` and also in the submitted supplementary media file.

V. Conclusions and Future Work

In this work, we have taken a step towards deploying quadrotor swarms in complex 3D environments. We have introduced a novel approach that addresses dynamic collisions through shared trajectories without linearization. Meanwhile, static collisions are handled by exploiting Octomap distance queries to build an implicit non-convex decomposition of free space. Our approach allows quadrotors to utilize available free space more efficiently, and thus, we outperform SFC-based methods in terms of mission completion time, success rate, and per-agent computation time. Furthermore, we have conducted experimental validations of our approach using a Crazyflie swarm testbed. Our work assumes prior map information but this requirement can be relaxed. For instance, our pipeline can be integrated with a high-level exploration planner [23] that generates paths based on local map information. Our future research efforts are directed towards the development of high-level path planners tailored to specific applications such as warehouse inventory management, surveillance, and exploration.

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 *2012 IEEE international symposium on safety, security, and rescue robotics (SSRR)*. IEEE, 2012, pp. 1–4.

[2] P. Schmuck and M. Chli, "Multi-uav collaborative monocular slam," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 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 *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 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] J. A. Preiss, W. Hönig, N. Ayanian, and G. S. Sukhatme, "Downwash-aware trajectory planning for large quadrotor teams," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 250–257.

[6] 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.

[7] 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.

[8] F. Rastgar, H. Masnavi, J. Shrestha, K. Kruusamäe, A. Aabloo, and A. K. Singh, "Gpu accelerated convex approximations for fast multi-agent trajectory optimization," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3303–3310, 2021.

[9] 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.

[10] 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.

[11] V. K. Adajania, S. Zhou, A. K. Singh, and A. P. Schoellig, "Amswarm: An alternating minimization approach for safe motion planning of quadrotor swarms in cluttered environments," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 1421–1427.

[12] 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.

[13] F. Gao, W. Wu, Y. Lin, and S. Shen, "Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 344–351.

[14] J. Ji, Z. Wang, Y. Wang, C. Xu, and F. Gao, "Mapless-planner: A robust and fast planning framework for aggressive autonomous flight without map fusion," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6315–6321.

[15] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.

[16] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "Ego-planner: An esdf-free gradient-based local planner for quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 478–485, 2020.

[17] 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 *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2021, pp. 4101–4107.

[18] 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.

[19] B. Şenbaşlar, W. Hönig, and N. Ayanian, "Rlss: real-time, decentralized, cooperative, networkless multi-robot trajectory planning using linear spatial separations," *Autonomous Robots*, pp. 1–26, 2023.

[20] J. Park, J. Kim, I. Jang, and H. J. Kim, "Efficient multi-agent trajectory planning with feasibility guarantee using relative bernstein polynomial," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 434–440.

[21] 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.

[22] V. K. Adajania, S. Zhou, A. K. Singh, and A. P. Schoellig, "AMSwarmX," https://github.com/utiasDSL/AMSwarmX, 2023.

[23] T. Dang, M. Tranzatto, S. Khattak, F. Mascarich, K. Alexis, and M. Hutter, "Graph-based subterranean exploration path planning using aerial and legged robots," *Journal of Field Robotics*, vol. 37, no. 8, pp. 1363–1388, 2020, wiley Online Library.