

Fast Trajectory Planning for Multiple Quadrotors using Relative Safe Flight Corridor

Jungwon Park¹ and H. Jin Kim¹

Abstract—This paper presents a new trajectory planning method for multiple quadrotors in obstacle-dense environments. We suggest a relative safe flight corridor (RSFC) to model safe region between a pair of agents, and it is used to generate linear constraints for inter-collision avoidance by utilizing the convex hull property of relative Bernstein polynomial. Our approach employs a graph-based multi-agent pathfinding algorithm to generate an initial trajectory, which is used to construct a safe flight corridor (SFC) and RSFC. We express the trajectory as a piecewise Bernstein polynomial and formulate the trajectory planning problem into one quadratic programming problem using linear constraints from SFC and RSFC. The proposed method can compute collision-free trajectory for 16 agents within a second and for 64 agents less than a minute, and it is validated both through simulation and indoor flight test.

I. INTRODUCTION

Multi-agent systems consisting of micro aerial vehicles (MAVs) are receiving attention from many industrial domains due to their agility, mobility, and applicability. To maximize their capabilities for various missions such as cooperative surveillance [1] and transportation [2], it requires to generate safe trajectories for multiple quadrotors in a complex environment within a short time. However, it has been challenging to efficiently formulate constraints to avoid obstacles and other agents. Furthermore, deadlock may happen if agents are packed in a narrow space. In this paper, we focus on an efficient planning method in terms of both cost and computation time which generates safe, dynamically feasible trajectories in an obstacle-dense environment without deadlock.

One popular approach to generate multi-agent trajectories is a centralized optimization method. In [3], constraints for collision avoidance are reformulated in integer constraints for mixed-integer quadratic programming (MIQP). However, it requires over 500–1000 seconds to optimize the trajectory of 2–4 agents due to the computational complexity of the MIQP. In [4], sequential convex programming (SCP) is proposed to replace non-convex constraints with convex ones. SCP shows good performance when planning a small number of quadrotors, but it is intractable for a large team and complex environment. The authors of [5] suggest nonlinear programming (NLP) which combines sequential planning to deal with nonlinear constraint directly. This use of sequential planning method allows to achieve better scalability, but it

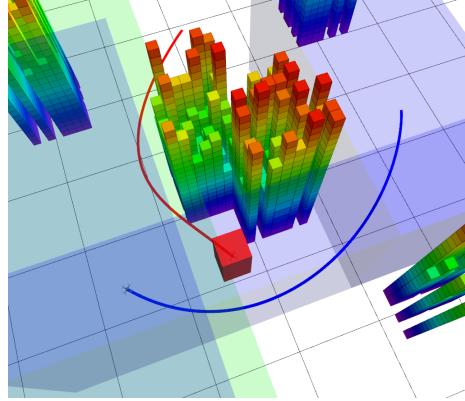


Fig. 1: SFC (semi-transparent blue box) and RSFC (semi-transparent green box) used for planning trajectories for two agents (red and blue lines). The trajectory of agent 2 (blue line) is generated within the intersection between SFC and RSFC to avoid obstacles and inter-collision model between two agents (red box)

has a limitation that no feasible solution can be found for a crowded situation.

A decentralized method also has been considered to reduce total planning time by distributing the computational load. Approaches based on LQR-obstacle [6] and buffered Voronoi cells [7] show that they can generate a collision-free path in real time. However, such distributed methods are not able to guarantee the completeness, no deadlock. In [8] and [9], distributed model predictive control (DMPC) is proposed to optimize the control input instead of a piecewise polynomial path, but they do not consider obstacles.

In single quadrotor path planning, many researchers have adopted a safe flight corridor (SFC) to model free space in a map. SFC is composed of connected convex sets, and it can be represented as linear inequality constraints for obstacle avoidance in quadratic programming (QP) [10, 11, 12]. In [13], SFC is used to separate a safe region of each agent in quadrotor swarm. By resizing SFC iteratively, trajectories of quadrotor swarm can be refined separately without inter-collision. This decoupled iterative optimization method shows good scalability in a maze-like environment, but it requires many iteration steps for the convergence of the overall cost.

In this paper, we propose a new centralized multi-agent path planning method that uses a relative safe flight corridor (RSFC) to find a feasible trajectory without any sequential or iterative process. Similar to SFC, RSFC utilizes a property

¹Jungwon Park is with the Department of Mechanical and Aerospace Engineering, Seoul National University, Seoul, South Korea
qwerty35@snu.ac.kr

H. Jin Kim is with the Department of Mechanical and Aerospace Engineering, Seoul National University, Seoul, South Korea
hjinkim@snu.ac.kr

of Bernstein polynomial to convert non-convex inter-collision avoidance constraints into linear ones as illustrated in Fig. 1, and it does not need an additional resizing process to find the feasible trajectory. Thus, our proposed method can optimize a piecewise polynomial trajectory by using QP only once, and it guarantees a feasible solution of QP does not cause a collision and deadlock. Recently, distributed planning is receiving much attention due to scalability, yet centralized methods can still provide the quality solution by the efficient formulation using RSFC.

Our main contributions can be summarized as follows.

- A collision avoidance constraint formulation method using relative safe flight corridor is proposed, which does not require sequential or iterative process.
- A fast trajectory optimization framework is presented in an obstacle-dense environment, which guarantees collision- and deadlock-free.

This paper is structured as follows. The problem statement is presented in section II. In section III, we describe the method of multi-agent trajectory planning using relative safe flight corridor. Experimental results are presented in section IV. Finally, section V contains conclusions.

II. PROBLEM STATEMENT

Consider a multi-agent robot system that consists of N_q quadrotors. Each quadrotor is assumed to have different size with radius r^1, \dots, r^{N_q} but has the same dynamic limit. We assume that prior knowledge of the free space \mathcal{F} and obstacle \mathcal{O} is given in 3D occupancy map and start, goal point of the i^{th} quadrotor is assigned as s^i, g^i .

It has been shown that quadrotor dynamics is differentially flat and trajectory can be represented in piecewise polynomials with flat outputs in time t [14]. Thus, trajectory of i^{th} quadrotor, $p^i(t)$, can be represented in M -segment piecewise polynomials.

In this paper, we aim to generate continuous, smooth trajectory $p^i(t)$ for all $i = 1, \dots, N_q$ which minimizes the integral of the square of the n^{th} derivative and does not collide with any obstacle and other agents.

III. METHOD

The overall structure of our proposed method is depicted in Fig. 2. Our method consists of three steps. First, the discrete path planner plans the initial trajectory using the multi-agent pathfinding (MAPF) algorithm. Then safe flight corridor (SFC) and relative safe flight corridor (RSFC) are constructed based on the initial trajectory. Finally, SFC and RSFC are converted into inequality constraints of quadratic programming (QP) and we obtain the desired trajectory by utilizing the convex hull property of Bernstein polynomial. The detail of each part is described in the following subsections.

A. Initial Trajectory Planning

When planning the trajectory of a single quadrotor, many researchers have divided the planning process into initial

trajectory planning and trajectory refinement, and such two-step method is now being adopted in the multi-agent case [15, 13]. Inspired by that, we first plan the discrete initial trajectory by using a graph-based MAPF algorithm.

The initial trajectory of the i^{th} quadrotor, p_{init}^i , is defined as an array of waypoints that connect start and goal position in a graph. In the MAPF, the cost function is defined as the sum of each trajectory's length.

There have been many researches about MAPF algorithm such as HCA* [16], M* [17], conflict-based search (CBS) [18]. Among them, we choose enhanced CBS (ECBS) [19] as a discrete initial trajectory planner because it can find a suboptimal solution in a short time and we can specify the bound of the solution cost. In other words, it is guaranteed that the cost of the trajectory is lower than c_w : optimal cost, where c_w is a user-specified bounding factor.

To utilize the graph-based ECBS in our problem, discrete planner translates 3D occupancy map into a 3D grid map. After translation, ECBS computes a discrete initial trajectory that connects start and goal points. If start and goal points are not located on the 3D grid map, then we use the nearest grid points to obtain trajectory and append the start/goal points to both ends. Finally, to calculate relative trajectory between two agents, we match the initial trajectory of all agents as the same length l_{max} by appending each goal point at the end of the trajectory, where l_{max} is the length of the longest initial trajectory.

B. Safe Flight Corridor Construction

SFCs of the i^{th} quadrotor, $SFC_1^i, \dots, SFC_{M_s}^i$, are defined as convex sets that do not collide with obstacle and are sequentially connected: for $m = 1, \dots, M_s$

$$SFC_m^i \oplus \mathcal{C}_{obs}^i \in \mathcal{F} \quad (1a)$$

and for $m = 1, \dots, M_s - 1$

$$SFC_m^i \cap SFC_{m+1}^i \neq \emptyset \quad (1b)$$

where \oplus is the Minkowski sum and \mathcal{C}_{obs}^i is the obstacle-collision model for the i^{th} quadrotor, which is defined as a sphere with radius r^i representing safety clearance between an obstacle and a quadrotor.

The trajectory of i^{th} quadrotor is collision-free from obstacle if for arbitrary $t \in [0, T]$, there exists $m \in \{1, \dots, M_s\}$ such that $p^i(t) \in SFC_m^i$, where T is the total flight time.

We construct SFC by the axis-search method. We initialize corridors with a predefined size at each waypoint of the initial trajectory. Then, except the corridors from the start and goal points, we expand them in the previous waypoint direction to connect two sequential convex sets. All waypoints except start and goal points are aligned on the 3D grid map, thus we can achieve the condition (1b) by this method. After that, we expand each corridor in all the other axis-aligned directions until it has a maximum possible free space. Finally, we delete duplicated corridors.

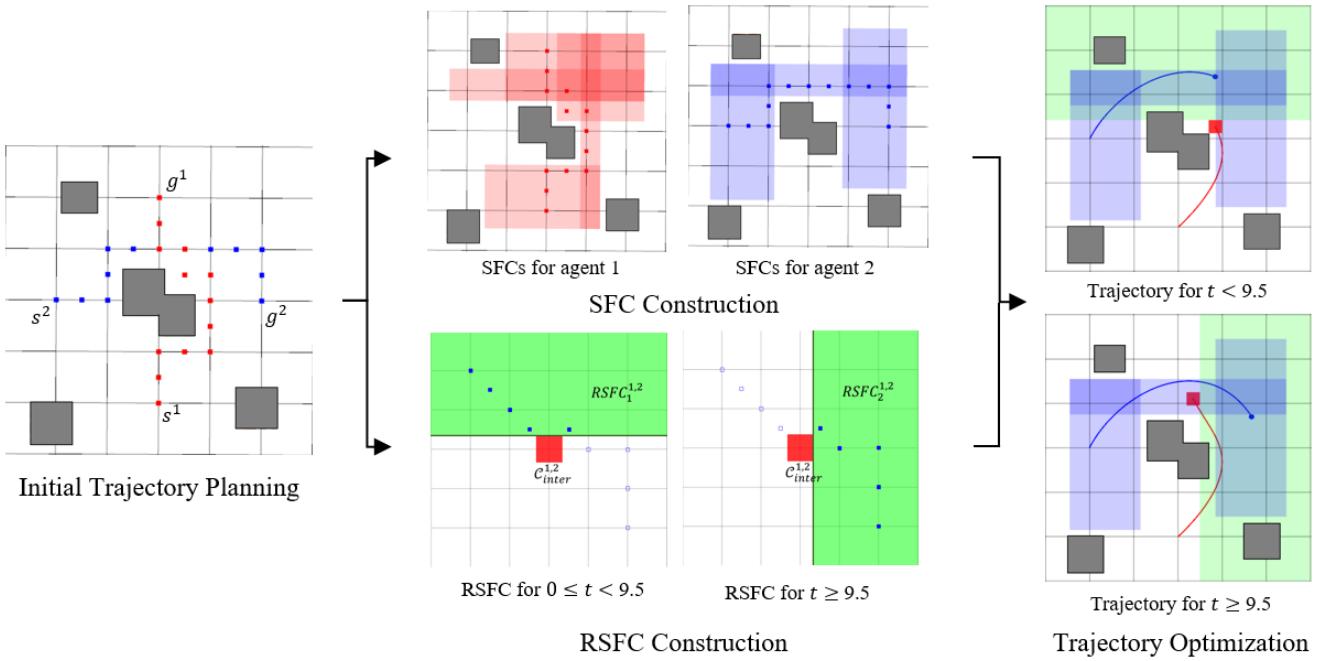


Fig. 2: Overview of the proposed method. The proposed method can plan the trajectory in the 3D space, but in this example, we plan the trajectory in 2D space for the convenience of explanation. We assigned the start positions for agent 1 (red) and agent 2 (blue) at $(0, -2)$, $(0, -2)$, and goal positions at $(0, 2)$, $(2, 0)$. We convert 3D occupancy map into the 3D grid map and compute discrete initial trajectory by using MAPF algorithm. SFC of each agent (red and blue boxes in SFC construction) is constructed along waypoints of their initial trajectories to prevent obstacle collision. RSFC (green boxes in RSFC construction) is used to confine a relative trajectory between two agents. Finally, SFC and RSFC are translated into linear inequality constraints and QP solver generates the continuous smooth trajectory that satisfies SFC and RSFC constraints.

C. Relative Safe Flight Corridor Construction

As SFC models a obstacle-free space as a convex set, RSFC models a free space for evasive maneuver between two agents. We define RSFCs between the i^{th} and j^{th} agents, $RSFC_1^{i,j}, \dots, RSFC_{M_r}^{i,j}$, as convex sets that do not invade the collision region between the i^{th} and j^{th} agents and are sequentially connected: for $m = 1, \dots, M_r$

$$RSFC_m^{i,j} \cap \mathcal{C}_{inter}^{i,j} = \emptyset \quad (2a)$$

and for $m = 1, \dots, M_r - 1$

$$RSFC_m^{i,j} \cap RSFC_{m+1}^{i,j} \neq \emptyset \quad (2b)$$

where $\mathcal{C}_{inter}^{i,j}$ is a inter-collision model between i^{th} and j^{th} quadrotors which has a rectangular parallelepiped oriented with the body frame of i^{th} quadrotor. Note that $\mathcal{C}_{inter}^{i,j}$ can vary for each pair of agents, which mean that it can handle different size of quadrotors. In our implementation, we assign length and width of $\mathcal{C}_{inter}^{i,j}$ as $2(r^i + r^j)$ and a height as $2c_{dw}(r^i + r^j)$ to consider downwash effect, where c_{dw} is downwash coefficient. The trajectory of j^{th} quadrotor does not collide with i^{th} quadrotor if for arbitrary $t \in [0, T]$, there exists $m \in \{1, \dots, M_s\}$ such that $p^j(t) - p^i(t) \in RSFC_m^{i,j}$.

Construction of RSFC is described in Fig. 3. First, we convert the initial trajectory into the relative initial trajectory for each pair of agents (Fig. 3b). The relative initial trajectory of

i^{th} and j^{th} quadrotors, $p_{init}^{i,j}$, can be obtained by subtracting corresponding waypoints of two initial trajectories. Next, for each waypoint in the relative path, we choose proper RSFC from RSFC candidates. Using the quadrotors' differential flatness property, we design RSFC candidates to reduce the number of decision variable at the optimization step. There are six RSFC candidates in direction $\pm x, \pm y, \pm z$, and each candidate $RSFC_\mu$ is defined as follows:

$$RSFC_\mu = \begin{cases} \{p | p \cdot n_\mu > r^i + r^j\} & \text{if } \mu = \pm x, \pm y \\ \{p | p \cdot n_\mu > c_{dw}(r^i + r^j)\} & \text{if } \mu = \pm z \end{cases} \quad (3)$$

where n_μ is a unit vector in the direction $\mu \in \{\pm x, \pm y, \pm z\}$. For each waypoint $p_{init}^{i,j}[k]$ in $p_{init}^{i,j}$, any $RSFC_\mu$ can be selected if μ satisfies the following condition:

$$p_{init}^{i,j}[k] \cdot n_\mu > 0 \quad (4)$$

However, redundant RSFC transitions along the waypoints may increase the number of polynomial segments and the computation time. Figs. 3c and 3d show the example. When we generate a smooth relative trajectory in RSFC, we need to plan two segment polynomials to represent the relative trajectory if there is one transition of RSFC (i.e. $RSFC_1^{i,j} \rightarrow RSFC_2^{i,j}$) along the waypoints as shown in Fig. 3c. However, if three transitions are involved (i.e.

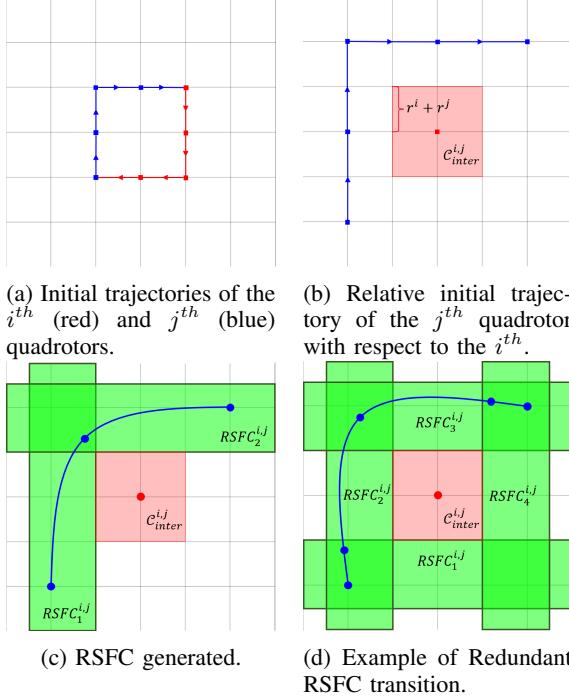


Fig. 3: Construction process of RSFC. For the convenience of explanation, it is depicted in 2D space. The initial trajectory and the relative initial trajectory of two multirotors are shown in Figs. 3a and 3b. The inter-collision model between two agents are depicted as a red box and RSFC is constructed along the waypoints of relative path avoiding the inter-collision model. Redundant RSFC transitions may occur as shown in Fig. 3d, so the greedy algorithm is used to minimize the number of RSFC transitions and the result is shown in Fig. 3c.

$RSFC_1^{i,j} \rightarrow RSFC_2^{i,j} \rightarrow RSFC_3^{i,j} \rightarrow RSFC_4^{i,j}$ as shown in Fig. 3d, we have to plan two more polynomial segments compared to the previous one. Thus, we use the greedy algorithm (Alg. 1) to minimize the number of RSFC transitions.

The algorithm receives p_{init}^i and p_{init}^j as input and returns $RSFC^{i,j}$. It initializes $RSFC^{i,j}$ (line 3) as an empty array and s_μ as an array of all zero with length l_{max} (line 4-5). After initialization, the algorithm verifies RSFC candidates using (4) and saves the result in s_μ (line 6-12). At the end of the relative path, it finds an RSFC candidate that includes the maximum number of waypoints and appends the candidate to $RSFC^{i,j}$ (line 14-15). After that, it goes to the last waypoint among the included ones (line 16). Then again it finds the maximum including candidate until it reaches the start point of relative path (line 17-20). Note that the new candidate must not be located at the opposite side of the previous candidate because quadrotors cannot jump through an empty space between two opposite candidates (line 18).

Algorithm 1 RSFC construction

```

1: Input:  $p_{init}^i, p_{init}^j$ 
2:  $l_{max} \leftarrow \max(\text{size}(p_{init}^i), \text{size}(p_{init}^j))$ 
3:  $RSFC^{i,j} \leftarrow \emptyset$ 
4: for all  $\mu \in \{+x, +y, +z, -x, -y, -z\}$  do
5:   initialize  $s_\mu[l_{max}]$  to 0
6: for  $n \leftarrow 1$  to  $l_{max}$  do
7:   for all  $\mu \in \{+x, +y, +z, -x, -y, -z\}$  do
8:     if  $(p_{init}^j[n] - p_{init}^i[n]) \cdot n_\mu > 0$  then
9:       if  $n = 1$  then
10:          $s_\mu[n] \leftarrow 1$ 
11:       else
12:          $s_\mu[n] \leftarrow s_\mu[n - 1] + 1$ 
13:    $n \leftarrow l_{max}$ 
14:    $\mu_M \leftarrow \arg \max_\mu (s_\mu[n])$ 
15:    $RSFC^{i,j}.\text{push\_front}(RSFC_{\mu_M})$ 
16:    $n \leftarrow n - s_{\mu_M}[n]$ 
17: while  $n > 0$  do
18:    $\mu_M \leftarrow \arg \max_{\mu \neq -\mu_M} (s_\mu[n])$ 
19:    $RSFC^{i,j}.\text{push\_front}(RSFC_{\mu_M})$ 
20:    $n \leftarrow n - s_{\mu_M}[n]$ 
21: return  $RSFC^{i,j}$ 

```

D. Time Segment Allocation

To formulate an optimization problem, we determine the *time segment* of the piecewise polynomial trajectory and allocate each SFC and RSFC to each polynomial segment. Let $p_m^i(t)$ be the m^{th} segment of $p^i(t)$ which is defined at $t \in [t_{m-1}^i, t_m^i]$. The time segment of the i^{th} quadrotor t_s^i is defined as:

$$t_s^i = [t_0^i, \dots, t_M^i] \quad (5)$$

In this paper, we set the trajectory of all agents to have the same time segment t_s because it is necessary for our algorithm to utilize the convex hull property of Bernstein basis polynomial. However, it can result in too many decision variables, which can increase the computation time. Thus, the following method is used to decrease the number of decision variables.

To construct the time segment for all agents, we generate partial time segments t_{sp} from SFC and RSFC. Alg. 2 shows the process of finding partial time segment. The algorithm receives SFC or RSFC and initial or relative initial trajectory as its input, and searches for the middle waypoint among the intersection of two sequential convex sets (line 11-13). After that, the algorithm records the index of this middle waypoint to assign as the location at which the SFC or RSFC transition occurs (line 14). In other words, m^{th} SFC or RSFC is allocated before the time $(n + \lfloor count/2 \rfloor) * t_{step}$ and $m + 1^{th}$ SFC or RSFC is allocated after time $(n + \lfloor count/2 \rfloor) * t_{step}$, where $n + \lfloor count/2 \rfloor$ is the index of middle waypoint among the intersection of m^{th} and $m + 1^{th}$ convex sets. In the SFC case, it is guaranteed that there exists a waypoint in two sequential SFCs because we connect the waypoint

Algorithm 2 Finding partial time segment

```

1: Input: Initial or relative initial trajectory  $p_{init}$ ,
2:           Array of sequential convex sets  $C$ ,
3:           Time step  $t_{step}$ 
4:  $t_{sp} \leftarrow \emptyset$ 
5:  $m \leftarrow 1$ 
6: for  $n \leftarrow 1$  to  $l_{max}$  do
7:   if  $m \geq \text{size}(C)$  then
8:     break
9:   if  $p_{init}[n] \in (C[m] \cap C[m + 1])$  then
10:    count  $\leftarrow 1$ 
11:    while  $p_{init}[n + count] \in (C[m] \cap C[m + 1])$ 
12:      and  $n + count \leq l_{max}$  do
13:        count  $\leftarrow count + 1$ 
14:         $t_{sp}.\text{push\_back}(n + \lfloor count/2 \rfloor * t_{step})$ 
15:         $n \leftarrow n + \lfloor count/2 \rfloor$ 
16:         $m \leftarrow m + 1$ 
17:   else if  $p_{init}[n] \in C[m + 1]$  then
18:      $t_{sp}.\text{push\_back}((n + 0.5) * t_{step})$ 
19:      $m \leftarrow m + 1$ 
20: return  $t_{sp}$ 

```

with SFC by the axis-search method. However, in the RSFC case, there may be no waypoint in an intersection between two sequential RSFCs (line 17). In this case, we do not use the integer index because it can make an infeasible constraint when SFC and RSFC are changed simultaneously at the same time. Instead, we use a heuristic method that gives a time delay to RSFC transition to avoid the simultaneous change of SFC and RSFC (line 18). It may increase the number of the decision variables, but we can increase the success rate of finding a feasible trajectory. This algorithm always returns an array with a maximum size of $2l_{max}$, so it is guaranteed that the piecewise trajectory has a maximum of $2l_{max} - 1$ segments.

After generating the partial time segments, we combine all of them into one and sort them. We delete duplicated elements and we generate the total time segment by appending the start time and total flight time at each end of the combined array. This method can reduce the size of the total time segment by overlapping the elements of partial time segment as much as possible.

We allocate SFC and RSFC to time segment by comparing t_{sp} with t_s . For example, assume that t_s is determined as $[0, 1, 2, 3]$ and t_{sp} of SFC is [2]. Then we can guess that SFC_1^i is assigned for the first and second segments of piecewise polynomials and SFC_2^i is assigned for the third segment. Let $SFC_{(m)}^i$ and $RSFC_{(m)}^i$ be convex sets that are allocated to the m^{th} polynomial segment. In this example, SFC is allocated as $SFC_{(1)}^i = SFC_{(2)}^i = SFC_1^i$ and $SFC_{(3)}^i = SFC_2^i$.

E. Trajectory Optimization

In the optimization step, we plan the smooth polynomial trajectory using SFC, RSFC and time segment t_s , but it is

difficult to handle SFC and RSFC with standard polynomial basis. Thus, we formulate the piecewise polynomial $p^i(t)$ of all agents as a piecewise Bernstein polynomial.

Bernstein basis polynomials of degree N are defined as:

$$B_{k,N}(t) = \binom{N}{k} t^k (1-t)^{N-k} \quad (6)$$

for $t \in [0, 1]$ and $k = 0, 1, \dots, N$, and Bernstein polynomial is the linear combination of Bernstein basis polynomials.

The m^{th} segment of $p^i(t)$ can be represented in Bernstein polynomial as:

$$p_m^i(t) = c_{m,0}^i B_{0,n}(\tau_m) + \dots + c_{m,N}^i B_{N,n}(\tau_m) \quad (7)$$

where $\tau_m = \frac{t - t_{m-1}}{t_m - t_{m-1}}$ and $c_m^i = [c_{m,0}^i, \dots, c_{m,N}^i]$ is the vector consisting of all control points of $p_m^i(t)$.

It is shown that a Bernstein polynomial has a convex hull property [20], in other words, a Bernstein polynomial $p^i(t)$ is confined within the convex hull of its control points c_m^i . In [10, 11, 12, 13], it has been used to confine $p_m^i(t)$ within $SFC_{(m)}^i$ by limiting control point c_m^i within $SFC_{(m)}^i$.

Here, this convex hull property can be used to confine the relative polynomial trajectory. Assume that $p^i(t)$ and $p^j(t)$ have the same time segment, then the m^{th} segment of $p_m^{i,j}$ can be written as:

$$\begin{aligned} p_m^{i,j}(t) &= \sum_{k=0}^N (c_{m,k}^j - c_{m,k}^i) B_{0,n}(\tau_m) \\ &= \sum_{k=0}^N c_{m,k}^{i,j} B_{0,n}(\tau_m) \end{aligned} \quad (8)$$

where $c_{m,k}^{i,j} = c_{m,k}^j - c_{m,k}^i$ for $k = 0, \dots, N$ is the control point of $p_m^{i,j}(t)$. We can observe that the relative Bernstein polynomial is also a Bernstein polynomial. Therefore, by the convex hull property, we can enforce quadrotors i, j not to collide with each other by limiting all control points $c_{m,k}^{i,j}$ within $RSFC_m^i$. In this way, we can generate the safe trajectory by adjusting RSFC for each pair of agents.

Our decision vector c consists of all control points of $p_m^i(t)$ for $m = 1, \dots, M$ and $i = 1, \dots, N_q$:

$$c = [c_1^1, \dots, c_M^1, \dots, c_1^{N_q}, \dots, c_M^{N_q}]^T \quad (9)$$

where M is the total number of polynomial segments that all agents share, and it is up to $2l_{max}$ as explained in section III-D.

The cost function of polynomials is defined as follows:

$$J = \sum_{i=1}^{N_q} \sum_{\mu \in \{x,y,z\}} \int_0^T \left(\frac{d^n p_\mu^i(t)}{dt^n} \right)^2 dt \quad (10)$$

where T is total flight time, and this can be represented into a quadratic form. In this paper, we set $n = 3$, so that it minimizes the integral of the square jerk of total trajectory. It is a reasonable choice because we can minimize the input aggressiveness of quadrotor [21].

The waypoint constraints for start, goal positions and continuity constraints for smooth trajectory can be reformulated

in linear equality constraints ($A_{eq}c = b_{eq}$). Therefore, our trajectory generation problem is reformulated as quadratic programming (QP) problem:

$$\begin{aligned} & \text{minimize} && c^T Q c \\ & \text{subject to} && A_{eq}c = b_{eq} \\ & && c_{m,k}^i \in SFC_{(m)}^i, \forall i, k \\ & && c_{m,k}^j - c_{m,k}^i \in RSFC_{(m)}^{i,j}, \forall i, j > i, k \end{aligned}$$

where Q is the Hessian cost matrix derived by concatenating all Hessian cost function of individual agents with a block-diagonal matrix form. The detailed formulation of equality constraints can be found in [12]. Note that we need only one QP to generate a smooth trajectory for all agents.

During optimization, we do not consider dynamic limits because they can be infeasible constraints for QP. Instead, we scale the time segment for all agents uniformly after optimization, similar to [13].

IV. EXPERIMENTS

A. Implementation Details

We implement our proposed method in C++14. We use the Octomap library [22] to represent the 3D occupancy map and use the dynamicEDT3D library [23] to compute distance between corridor and obstacle for SFC construction. For trajectory optimization, CPLEX QP solver [24] is used to solve (III-E).

We model the collision models of quadrotors with radius $r = 0.15\text{m}$ and downwash coefficient $c_{dw} = 2$ based on the specification of Crazyflie 2.0 in [13]. For initial trajectory planning, the grid size of the 3D grid map is determined to 0.5 m in x, y-axis directions and 1 m in z-axis direction. We set the degree of polynomials to $N = 5$ and give constraints to be continuous up to acceleration. Fig. 4 shows the planning result of 16 agents.

B. Computation Time Evaluation

We evaluate the computation time on a PC running Ubuntu 16.04. with Intel Core i7-7700 @ 3.60GHz CPU and 16G RAM. Our experiment is conducted in $10\text{ m} \times 10\text{ m} \times 2.5\text{ m}$ space. We randomly deploy 30 trees of size $0.3\text{ m} \times 0.3\text{ m} \times 1\text{--}2.5\text{ m}$. Start positions of quadrotors are uniformly distributed in a boundary of the xy-plane in 1 m height, and we assigned the goal points at the opposite to their start position as shown in Fig. 4.

We conduct the experiments by randomly changing the location of obstacles and measure the computation time of each step. Table I shows the average computation time of 30 experiments. The proposed method takes about a second for 16 quadrotors and a minute for 64 quadrotors. Although it uses the solver with $O(n^3)$ time complexity, the actual total computation time is short enough.

C. Success Rate Analysis

In section III-D, we give the time delay at the RSFC partial time segment to avoid infeasible constraints. To verify that, we compared the two time allocation method, one is

time allocation with RSFC time delay and the other is time allocation without the time delay by changing the line 18 of Alg. 2 to $n * t_{step}$. We plan the trajectory of 16 quadrotors 50 times to measure the success rate. We use the same environment setting in section IV-B except quadrotor size.

Fig. 5 shows that the time allocation with RSFC time delay has a higher probability to find a feasible solution. It also shows a 100 percent success rate when quadrotor size is 0.15 m and 0.2 m . As expected, the success rate of both methods decrease as the quadrotor size increase.

D. Flight Test

We demonstrate our algorithm with 6 Crazyflie 2.0 quadrotors in a $5\text{ m} \times 7\text{ m} \times 2.5\text{ m}$ space. Crazyswarm [25] is used to follow the pre-computed trajectory, and Vicon motion capture system is used to estimate the position of each agent at 100 Hz. It takes 0.138 seconds to plan the trajectory for all agents. Fig. 6 shows the snapshot of flight test and the pre-computed trajectory. Full flight is presented in the supplemental video.

V. CONCLUSIONS

In this paper, we propose a trajectory planning method using RSFC to deal with the inter-collision problem in a multi-MAV system. RSFC models the free space for inter-collision avoidance into a convex set, and we show that it can be converted into linear constraints by utilizing the convex hull property of Bernstein polynomial. To generate trajectory for multiple quadrotors, we adopt the ECBS algorithm to obtain the initial trajectory in a 3D grid map. Then we construct SFC and RSFC based on the initial trajectory, and allocate them to each segment considering infeasible simultaneous transition. Finally, an optimization solver generates a smooth trajectory that is collision-free and deadlock-free. The proposed method can generate a safe trajectory for 64 agents in a minute, and flight test is executed to validate our solution.

In future work, we plan to reduce the computational effort of our work for online trajectory generation and we plan to develop more precise time allocation method that guarantees a feasible solution.

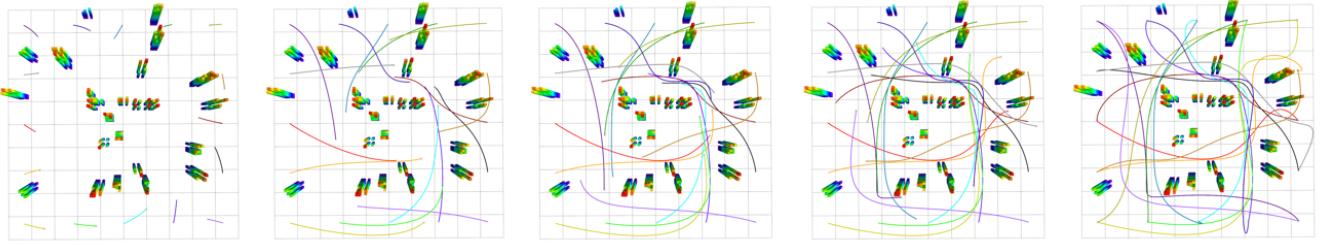


Fig. 4: Top-down view of 16 quadrotors in a $10 \text{ m} \times 10 \text{ m} \times 2.5 \text{ m}$ random forest map. Goal points are opposite to start positions.

TABLE I: Computation time by the number of quadrotors

Agents	ECBS($c_w=1.3$) (s)	SFC Construction (s)	RSFC Construction (s)	Traj. Optimization (s)	Total Comp. Time (s)
4	0.034	0.039	1.68E-5	0.034	0.11
8	0.037	0.053	4.84E-5	0.139	0.23
16	0.048	0.081	1.70E-4	0.800	0.93
32	0.059	0.137	5.77E-4	6.65	6.86
64	0.167	0.256	2.14E-3	50.7	51.2

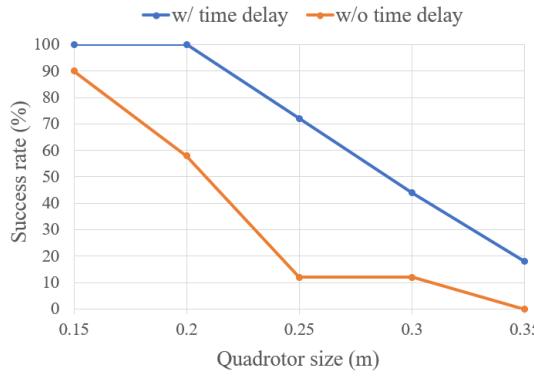
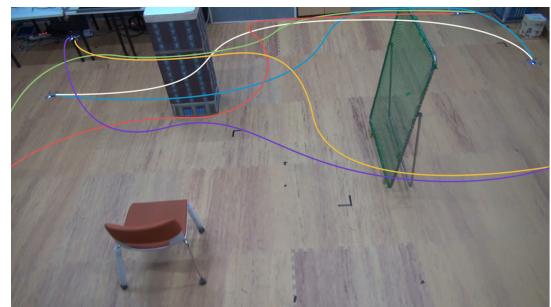
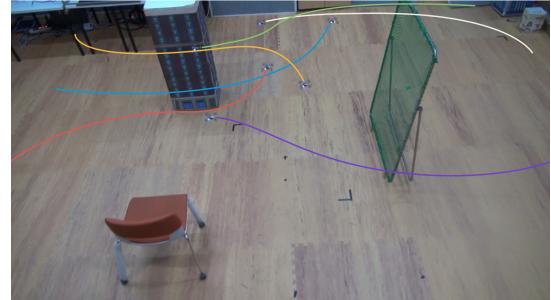


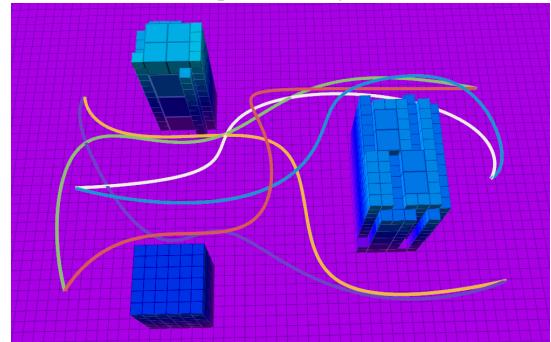
Fig. 5: Success rate of trajectory planning for 16 agents by the time allocation method.

REFERENCES

- [1] Martin Saska et al. “Swarm distribution and deployment for cooperative surveillance by micro-aerial vehicles”. In: *Journal of Intelligent & Robotic Systems* 84.1-4 (2016), pp. 469–492.
- [2] Hyooin Kim et al. “Motion planning with movement primitives for cooperative aerial transportation in obstacle environment”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 2328–2334.
- [3] Daniel Mellinger, Alex Kushleyev, and Vijay Kumar. “Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams”. In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 477–483.
- [4] Federico Augugliaro, Angela P Schoellig, and Raffaello D’Andrea. “Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach”. In: *Intelligent Robots and*



(a) Snapshots of flight test.



(b) Trajectory plan for flight test.

Fig. 6: Flight test with 6 quadrotors. Full flight is presented in the supplemental video.

- Systems (IROS), 2012 IEEE/RSJ International Conference on.* IEEE. 2012, pp. 1917–1922.
- [5] D Reed Robinson et al. “An Efficient Algorithm for Optimal Trajectory Generation for Heterogeneous Multi-Agent Systems in Non-Convex Environments”. In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 1215–1222.
- [6] Daman Bareiss and Jur Van den Berg. “Reciprocal collision avoidance for robots with linear dynamics using lqr-obstacles”. In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on.* IEEE. 2013, pp. 3847–3853.
- [7] Dingjiang Zhou et al. “Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells”. In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 1047–1054.
- [8] Li Dai et al. “Distributed MPC for formation of multi-agent systems with collision avoidance and obstacle avoidance”. In: *Journal of the Franklin Institute* 354.4 (2017), pp. 2068–2085.
- [9] Peng Wang and Baocang Ding. “A synthesis approach of distributed model predictive control for homogeneous multi-agent system with collision avoidance”. In: *International Journal of Control* 87.1 (2014), pp. 52–63.
- [10] Sarah Tang and Vijay Kumar. “Safe and complete trajectory generation for robot teams with higher-order dynamics”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE. 2016, pp. 1894–1901.
- [11] Sikang Liu et al. “Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments”. In: *IEEE Robotics and Automation Letters* 2.3 (2017), pp. 1688–1695.
- [12] Fei Gao et al. “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.
- [13] Wolfgang Höning et al. “Trajectory planning for quadrotor swarms”. In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 856–869.
- [14] Daniel Mellinger and Vijay Kumar. “Minimum snap trajectory generation and control for quadrotors”. In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on.* IEEE. 2011, pp. 2520–2525.
- [15] Yang Xu et al. “Concurrent Optimal Trajectory Planning for Indoor Quadrotor Formation Switching”. In: *Journal of Intelligent & Robotic Systems* (2018), pp. 1–18.
- [16] David Silver. “Cooperative Pathfinding.” In: *AIIDE* (2005), pp. 117–122.
- [17] Glenn Wagner and Howie Choset. “M*: A complete multirobot path planning algorithm with performance bounds”. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems.* IEEE. 2011, pp. 3260–3267.
- [18] Gunil Sharon et al. “Conflict-based search for optimal multi-agent pathfinding”. In: *Artificial Intelligence* 219 (2015), pp. 40–66.
- [19] Max Barer et al. “Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem”. In: *Seventh Annual Symposium on Combinatorial Search.* 2014.
- [20] Michael Zettler and Jürgen Garloff. “Robustness analysis of polynomials with polynomial parameter dependency using Bernstein expansion”. In: *IEEE Transactions on Automatic Control* 43.3 (1998), pp. 425–431.
- [21] Mark W Mueller, Markus Hehn, and Raffaello D’Andrea. “A computationally efficient motion primitive for quadrocopter trajectory generation”. In: *IEEE Transactions on Robotics* 31.6 (2015), pp. 1294–1310.
- [22] Armin Hornung et al. “OctoMap: An efficient probabilistic 3D mapping framework based on octrees”. In: *Autonomous robots* 34.3 (2013), pp. 189–206.
- [23] Boris Lau, Christoph Sprunk, and Wolfram Burgard. “Efficient grid-based spatial representations for robot navigation in dynamic environments”. In: *Robotics and Autonomous Systems* 61.10 (2013), pp. 1116–1130.
- [24] ILOG CPLEX. *12.7. 0 User’s Manual.* 2016.
- [25] James A Preiss et al. “Crazyswarm: A large nano-quadcopter swarm”. In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on.* IEEE. 2017, pp. 3299–3304.

ACKNOWLEDGMENT

This material is based upon work supported by the Ministry of Trade, Industry & Energy(MOTIE, Korea) under Industrial Technology Innovation Program. No.10067206, ‘Development of Disaster Response Robot System for Life-saving and Supporting Fire Fighters at Complex Disaster Environment’

This work was supported by the Robotics Core Technology Development Project (10080301) funded by the Ministry of Trade, Industry and Energy (MoTIE, Korea)