공학석사학위논문

상대적 안전비행영역과 상대적 번스타인 다항식을 이용한 다수 쿼드로터의 경로 계획

Trajectory Planning for Multiple Quadrotors using Relative Safe Flight Corridor and Relative Bernstein Polynomial

2020 년 8 월

서울대학교 대학원

기계항공공학부

박 정 원

Trajectory Planning for Multiple Quadrotors using Relative Safe Flight Corridor and Relative Bernstein Polynomial

A Thesis

by

Jungwon Park

Presented to the Faculty of the Graduate School of
Seoul National University
in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE

Department of Mechanical & Aerospace Engineering

Seoul National University

Supervisor : Professor H. Jin Kim

August 2020

to my

FAMILY

with love

Abstract

Trajectory Planning for Multiple Quadrotors using Relative Safe Flight Corridor and Relative Bernstein Polynomial

Jungwon Park
Department of Mechanical & Aerospace Engineering
The Graduate School
Seoul National University

Multi-agent systems consisting of unmanned aerial vehicles (UAVs) are receiving attention from many industrial domains due to their mobility, and applicability. To safely operate these multi-agent systems, path planning algorithm that can generate safe, dynamically feasible trajectory is required. However, existing multi-agent trajectory planning methods may fail to generate multi-agent trajectory in obstacle-dense environment due to deadlock or optimization failure caused by infeasible collision constraints. In this paper, we presents a new efficient algorithm which guarantees a solution for a class of multi-agent trajectory planning problems in obstacle-dense environments. Our algorithm combines the advantages of both grid-based and optimization-based approaches, and generates safe, dynamically feasible trajectories without suffering from an erroneous optimization setup such as imposing infeasible collision constraints. We adopt a sequential optimization method with *dummy agents* to improve the scalability of the algorithm, and utilize the convex hull property of Bernstein polynomial to replace non-convex collision avoidance constraints to convex ones. We validate the proposed algorithm through the comparison with our previous work and SCP-based method. The proposed method reduces more than 50% of the objective cost compared to our previous work, and reduces more than 75% of the computation time compared to SCP-based method. Furthermore, the proposed method can compute the trajectory for 64 agents on average 6.36 seconds with Intel Core i7-7700 @ 3.60GHz CPU and 16G RAM.

Keyword : Multi-agent path planning, Collision avoidance, Quadrotor.

Student Number : 2018-28113

Table of Contents

List of Figures

List of Tables

1 Introduction

Multi-agent systems with many unmanned aerial vehicles (UAVs) broaden the range of achievable missions to complex environments unsafe or hard to reach for humans or a single agent. For successful operation of these multi-agent systems, path planning algorithm is required to generate a collision-free trajectory in any obstacle environment. However, many works have a risk to fail in dense cluttered environments due to deadlock [1, 2] or failure caused by enforcing infeasible collision constraints in the formulation [3, 4].

In this paper, we present an efficient multi-agent trajectory planning algorithm which generates safe, dynamically feasible trajectories in obstacle-dense environments by extending our previous work [5]. The proposed algorithm is designed to have the advantages of both grid-based and optimization-based approaches. First, it guarantees the feasibility of optimization problem formulation by utilizing an initial trajectory computed from grid-based multi-agent path finding algorithm. Second, it generates a dynamically feasible continuous trajectory by optimizing the initial trajectory with consideration of quadrotor dynamics. When we formulate the optimization problem, we utilize the convex hull property of relative Bernstein polynomial to translate non-convex collision avoidance constraints to convex ones. Compared to the previous work [5], we modify the method for constructing constraints not to occur infeasible constraints between collision avoidance constraints, and we introduce a sequential optimization method. This sequential

method can deal with a large scale of agents with improved computational efficiency, and does not cause deadlock or optimization failure by employing *dummy agents*.

1.1 Literature review

There have been discussions in literature closely related to our work on multi-agent trajectory planning. In [3, 6, 7], the trajectory generation problems are reformulated as mixed-integer quadratic programming (MIQP) or sequential convex programming (SCP) problems, that apply collision constraints at each discrete time step. These methods suit well systems with a small number of agents, but they are intractable for large teams and complex environments because an additional adaptation process is required to find proper discretization time step depending on the size of agents and obstacles. On the other hand our method does not require this process because we do not use time discretization.

Sequential planning proposed in [8] for better scalability is similar to our work. However, it may not be able to find a feasible solution for a crowded situation. To solve this, we adopt dummy agents which move along the initial trajectory computed by a grid-based planner to prevent deadlock.

The most relevant work can be found in [9, 10]. They plan an initial trajectory with a grid-based planner and then construct a safe flight corridor (SFC), which indicates a safe region of each agent. However, they need to resize SFC iteratively until the overall cost converges, while our proposed method does not need an additional resizing process by using relative Bernstein polynomial.

Recently, distributed planning is receiving much attention due to scalability [1, 2, 4]. However, such distributed methods are not able to guarantee a safe solution in obstacle-dense environments due to deadlock.

1.2 Thesis contribution

In this paper, we propose an efficient multi-agent trajectory optimization method in both term of optimality of trajectory and computation time. Our main contributions can be summarized as follows:

1. A multi-agent trajectory planning algorithm is presented for obstacle-dense environments, which generates collision-free and dynamically feasible trajectories without a potential optimization failure by using relative Bernstein polynomial.

2. A sequential trajectory optimization method is proposed with dummy agents, which reduces computational load.

1.3 Thesis outline

The remainder of this paper is organized as follows. In Section 2, we introduce the background knowledge for Bernstein polynomial. Section 3 propose a multi-agent trajectory optimization method. In Section 4, we introduce sequential optimization method using dummy agents. Experimental results are presented in Section 5. Finally, Section 6 contains conclusions.

2

Bernstein polynomial

Due to the differential flatness of quadrotor dynamics, it is known that the trajectory of quadrotor can be represented in a polynomial function with flat outputs (x, y, z, ψ) in time t, where x, y, z is the quadrotor's position and ψ is the quadrotor's yaw angle [11]. However, it is difficult to handle collision avoidance constraints with standard polynomial basis because standard polynomial basis does not provide spatial information of polynomial. For this reason, we will utilize Bernstein polynomial to represent the trajectory of quadrotors. Bernstein polynomial is one of the special form of Bézier curve, and has various useful properties compared to standard polynomial.

2.1 Definition

The Bernstein basis polynomial of degree n is defined as follows:

$$B_{k,n}(\tau) = \binom{n}{k} \tau^k (1-\tau)^{n-k} \tag{2.1}$$

where $\tau \in [0, 1]$ and $k = 0, 1, ..., n$.

The Bernstein polynomial $p(\tau) \in \mathbb{R}^3$ is defined as the linear combination of Bernstein basis polynomials:

$$p(\tau) = \sum_{k=0}^{n} c_k B_{k,n}(\tau) \tag{2.2}$$

The coefficients $c_k \in \mathbb{R}^3$ are called control points of the Bernstein polynomial.

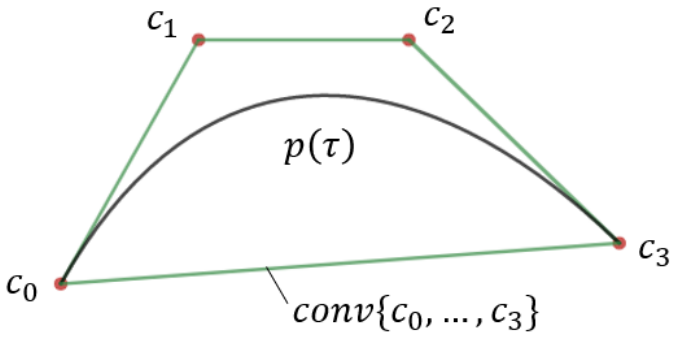


Figure 2.1: Convex hull property of Bernstein polynomial.

2.2 Properties

In this section, we introduce useful properties of Bernstein polynomial which will be utilized in trajectory optimization.

2.2.1 Convex hull property

One of the important properties of Bernstein polynomial is convex hull property. Convex hull is a convex envelop of a set of points, which is defined as follows:

$$\text{conv}\{c_0, ..., c_n\} = \left\{ \sum_{k=0}^{n} \lambda_k c_k \middle| \lambda_k \geq 0 \text{ for all } k \text{ and } \sum_{k=0}^{n} \lambda_k = 1 \right\} \tag{2.3}$$

As shown in Fig. 2.1, Bernstein polynomial $p(\tau)$ is always confined within the convex hull of control points:

$$p(\tau) \in \text{conv}\{c_0, ..., c_n\} \text{ for all } \tau \in [0, 1] \tag{2.4}$$

This property can be used to confine the polynomial trajectory within the desired region.

2.2.2 Endpoint interpolation property

For given n^{th} order Bernstein polynomial $p(\tau)$ with control points $c_0, ..., c_n$, $p(\tau)$ always start at the first control point c_0 and end at the last control point c_n:

$$p(0) = c_0, p(1) = c_n \tag{2.5}$$

Using this property, we can assign start and goal points of quadrotors by placing the first and last control points at the proper position.

2.2.3 Arithmetic operations and derivatives

We can show that the sum and difference of two Bernstein polynomials are still Bernstein polynomials if two polynomial have the same degree. Assume that two Bernstein polynomial $p^i(\tau), p^j(\tau)$, have control points $c^i_{k=0,...,n}, c^j_{k=0,...,n}$ respectively. Then the sum or difference of two Bernstein polynomial can be written as follows:

$$\begin{aligned} p^j(\tau) \pm p^i(\tau) &= \sum_{k=0}^{n} c_k^j B_{k,n}(\tau) \pm \sum_{k=0}^{n} c_k^i B_{k,n}(\tau) \\ &= \sum_{k=0}^{n} (c_k^j \pm c_k^i) B_{k,n}(\tau) \end{aligned} \tag{2.6}$$

Similar to arithmetic operation case, derivatives of Bernstein polynomial is also Bernstein polynomial. Assume that with derivative of n^{th} order Bernstein polynomial has control points $c'_0, ..., c'_{n-1}$. Then, control points of derivative can be derived from control points of original Bernstein polynomial:

$$c'_k = n(c_{k+1} - c_k) \text{ for all } k = 0, ..., n-1 \tag{2.7}$$

3

Multi-agent trajectory optimization

In this chapter, we formulate multi-agent trajectory planning problem as an optimization problem. Our objective is to generate dynamically feasible, point to point trajectory for multiple quadrotors without any collision and deadlock. We design objective function considering quadrotor dynamics, and we introduce a constraint construction method which guarantees that the optimization problem consists of feasible constraints. The detail of each part will be described in the following sections.

3.1 Problem formulation

In this section, we formulate an optimization problem to generate point to point trajectories for a multi-UAV system consisting of N quadrotors. For the i^{th} quadrotor, start point is given as s^i and goal point is assigned as g^i. The quadrotors may have a different size with the radius $r^1, ..., r^N$, and may have a different dynamic limit with the maximum velocity $v^1_{max}, ..., v^N_{max}$ and the maximum acceleration $a^1_{max}, ..., a^N_{max}$.

3.1.1 Assumption

To generate multi-agent trajectory with feasibility guarantee, we need several assumptions:

1. The prior knowledge of an obstacle space $\mathcal{O}$ and a free space $\mathcal{F}$ of the environment $\mathcal{E}$ is given as a 3D occupancy map.

2. There is no dynamic obstacle in the environment $\mathcal{E}$ during the flight.

3. When we translate the environment to the grid space with grid size d, there exists a collision-, deadlock-free solution for N quadrotors which connects given start and goal points.

While the first and second assumptions are widely used assumption in multi-agent path planning area, our proposed method requires the third assumption unlike other path planning algorithms. It is because our algorithm utilizes the solution from grid-based path planner to construct collision avoidance constraints. If there exists collision-, deadlock-free trajectories in the environment $\mathcal{E}$, then the third assumption can be satisfied by decreasing the grid size d until finding the solution.

3.1.2 Trajectory Representation

As we mentioned in the previous chapter, we can represent the trajectory of quadrotor in a polynomial with flat outputs (x, y, z, ψ). However, yaw angle is not important when we plan the point to point trajectory, so we hold yaw angle to zero.

In this paper, we formulate the trajectory of the i^{th} quadrotor, $p^i(t) \in \mathbb{R}^3$, as M-segment piecewise Bernstein polynomials:

$$p^i(t) = \begin{cases} p_1^i(t) = \sum_{k=0}^{n} c_{1,k}^i B_{k,n}(\tau_1) & t \in [T_0, T_1] \\ p_2^i(t) = \sum_{k=0}^{n} c_{2,k}^i B_{k,n}(\tau_2) & t \in [T_1, T_2] \\ \vdots & \vdots \\ p_M^i(t) = \sum_{k=0}^{n} c_{M,k}^i B_{k,n}(\tau_M) & t \in [T_{M-1}, T_M] \end{cases} \tag{3.1}$$

where $\tau_m = \frac{t - T_{m-1}}{T_m - T_{m-1}}$, and $p_m^i(t)$ is the the m^{th} segment of the i^{th} quadrotor's trajectory. $c_{m,k}^i \in \mathbb{R}^3$ is the k^{th} control point of $p_m^i(t)$, and T_{m-1}, T_m are the start and end time of the m^{th} segment, respectively. We note that all quadrotors have the same number of segments M and they share the segment time $T_0, ..., T_M$.

For the multi-agent trajectory planning, we need to determine all the control points and segment times in Eq. (3.1). Here, we set the decision vector of optimization problem to include all the control points of $p^i(t)$ for $i = 1, ..., N$, as the following equation:

$$c = [c^1; ...; c^N] \tag{3.2}$$

where $c^i = [c^i_1; ...; c^i_M]$ and $c^i_m = [c^i_{m,0}; ...; c^i_{m,n}]$. The symbol ; denotes appending of two vectors in the direction of the column.

3.1.3 Objective function

We design the objective function to minimize the integral of the square of the desired derivative ϕ:

$$J = \sum_{i=1}^{N} \int_{T_0}^{T_M} \left\| \frac{d^\phi}{dt^\phi} p^i(t) \right\|_2^2 dt \tag{3.3}$$

We can show that this objective function can be reformulated to quadratic form:

$$\begin{aligned} J &= \sum_{i=1}^{N} \sum_{m=1}^{M} \int_{T_{m-1}}^{T_m} \left\| \frac{d^\phi}{dt^\phi} p^i_m(t) \right\|_2^2 dt \\ &= \sum_{i=1}^{N} \sum_{m=1}^{M} \int_{T_{m-1}}^{T_m} {c^i_m}^T (q^T q) c^i_m dt \\ &= \sum_{i=1}^{N} \sum_{m=1}^{M} {c^i_m}^T (\int_{T_{m-1}}^{T_m} q^T q dt) c^i_m \\ &= c^T Q c \end{aligned} \tag{3.4}$$

where $q = \frac{d^\phi}{dt^\phi}[B_{0,n}(\tau_m), ..., B_{n,n}(\tau_m)]^T$, and Q is the Hessian matrix of the objective function. For the desired derivative, we assign $\phi = 3$ to minimize the jerk of the trajectory. so that the input to the quadrotor becomes less aggressive [12].

3.1.4 Convex constraints

Assume that the control points of l^{th} derivative of $p^i_m(t)$ is $c^{l,i}_{m,0}, ..., c^{l,i}_{m,n}$. Then we can derive control point of l^{th} derivative using derivative property of Bernstein polynomial.

$$c^{0,i}_{m,k} = c^i_{m,k}, c^{l,i}_{m,k} = \frac{n!}{(n-l)!} \frac{c^{l-1,i}_{m,k+1} - c^{l-1,i}_{m,k}}{T_m - T_{m-1}} \tag{3.5}$$

Using these control points, we can formulate convex constraints such as waypoint, continuity and dynamical feasibility constraints.

The waypoint constraint is a constraint that enforces the trajectory to pass start and goal points. Furthermore, quadrotor's velocity and acceleration should be zero at the start and goal points. To construct waypoint constraints, we utilize endpoint interpolation property of Bernstein polynomial for all $i = 1, ..., N$ and $l = 1, ..., \phi - 1$:

$$\begin{aligned} c_{1,0}^{0,i} &= s^i, c_{M,n}^{0,i} = g^i \\ c_{1,0}^{l,i} &= 0, c_{M,n-l}^{l,i} = 0 \end{aligned} \tag{3.6}$$

The continuity constraint is a constraint that the trajectory should be continuous up to the $\phi - 1^{th}$ derivatives. Similar to waypoint constraints, we can construct continuity constraints using endpoint interpolation property for all $l = 1, ..., \phi - 1$, $m = 1, ..., M - 1$:

$$c_{m,n}^{l,i} = c_{m+1,0}^{l,i} \tag{3.7}$$

For the dynamic feasibility of the trajectory, the quadrotor must not exceed maximum velocity and acceleration. The authors of [13] utilize convex hull property of Bernstein polynomial to construct dynamical feasibility constraint. However, this approach may cause infeasible constraints in optimization problem. Thus, we will solve this problem by uniform time scaling after planning the trajectory instead of imposing hard constraints.

To summarize, convex constraints can be written in affine equality and inequality constraints:

$$A_{eq}c = b_{eq} \tag{3.8}$$

3.1.5 Non-convex collision avoidance constraints

We define an obstacle collision model of the i^{th} quadrotor, which models a collision region between a quadrotor and obstacles (See Fig. 3.1):

$$\mathcal{C}_{obs}^i = \{p \in \mathbb{R}^3 \mid \|p\|_2^2 \leq (r^i)^2\} \tag{3.9}$$

The i^{th} quadrotor must satisfy the condition below not to collide with obstacles:

$$p^i(t) \oplus \mathcal{C}_{obs}^i \subset \mathcal{F}, \quad t \in [T_0, T_M] \tag{3.10}$$

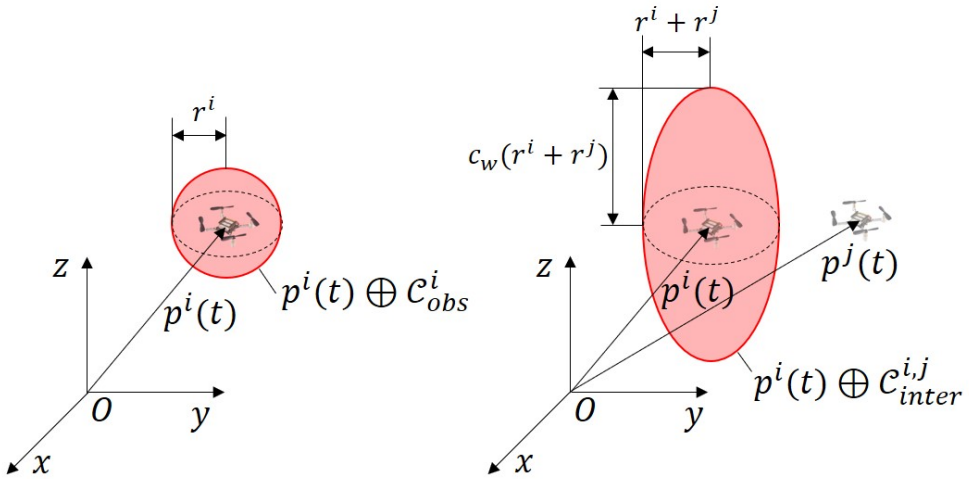


Figure 3.1: Obstacle collision model (Left), and Inter-collision model (Right).

where $\oplus$ is the Minkovski sum.

A collision region between i^{th} and j^{th} agents can be expressed with an inter-collision model $\mathcal{C}^{i,j}_{inter}$:

$$\mathcal{C}^{i,j}_{inter} = \{p \in \mathbb{R}^3 \mid p^T E p \leq (r^i + r^j)^2\} \tag{3.11}$$

where E is $diag([1, 1, 1/(c_{dw})^2])$, and c_{dw} is a coefficient to consider a downwash effect. The i^{th} agent does not collide with the j^{th} agent if the relative trajectory of the j^{th} agent respect to the i^{th} agent, $p^{i,j}(t) = p^j(t) - p^i(t)$, satisfies the following condition:

$$p^{i,j}(t) \cap \mathcal{C}^{i,j}_{inter} = \emptyset, \quad t \in [T_0, T_M] \tag{3.12}$$

Non-convexity of (3.10) and (3.12) makes it difficult to directly employ them. In the next section, we will show the method that relaxes those non-convex constraints to convex ones using Bernstein polynomial.

3.2 Collision constraints construction

One of the useful properties of the Bernstein polynomial is a convex hull property that the Bernstein polynomial is confined within the convex hull of its control points. This property has been used to confine the trajectory to a convex set called safe flight corridor (SFC) for obstacle avoidance [13, 14].

Similar approach can be used to confine the relative polynomial trajectory to inter-collision-free region. Let $p_m^{i,j}(t) = p_m^j(t) - p_m^i(t)$ is their relative trajectory. As we noted in previous section, all quadrotors have the same number of segments and they share the same segement time. Thus, we can use the arithmetic property of Bernstein polynomial when we compute relative trajectory between two agents:

$$\begin{aligned} p_m^{i,j}(t) &= \sum_{k=0}^{n} (c_{m,k}^j - c_{m,k}^i) B_{k,n}(\tau_m) \\ &= \sum_{k=0}^{n} c_{m,k}^{i,j} B_{k,n}(\tau_m) \end{aligned} \tag{3.13}$$

where $c_{m,k}^{i,j} = c_{m,k}^j - c_{m,k}^i$ is the control point of $p_m^{i,j}(t)$. Thus, by the convex hull property, we can enforce the i^{th} and j^{th} quadrotors not to collide with each other by limiting all control points $c_{m,k}^{i,j}$ within a convex, inter-collision free region. We call this region a relative safe flight corridor (RSFC). In this way, we can generate the safe trajectory by adjusting SFC, RSFC in our problem.

In the previous work [5], we determined RSFC by choosing a proper one among pre-defined RSFC candidates. RSFC candidates were designed to be able to utilize the differential flatness of quadrotor, and so as to achieve fast planning speed. However, it may fail to find a trajectory because a feasible region that satisfies both RSFC and SFC constraints may not exist.

Fig. 3.2 shows the example of feasible region between RSFC and SFC constraints. The region surrounded by the blue dashed line is safe flight corridor (SFC) for the blue agent, and the region surrounded by green dashed line is the intersection of relative safe flight corridor (RSFC) for the blue agent. To generate a safe trajectory, there must exist intersection between SFC and RSFC (gray shaded area).

To guarantee the existence of such feasible region, we first define three key terms in this paper: initial trajectory, SFC, and RSFC. Then, we introduce construction method of these collision constraints.

3.2.1 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

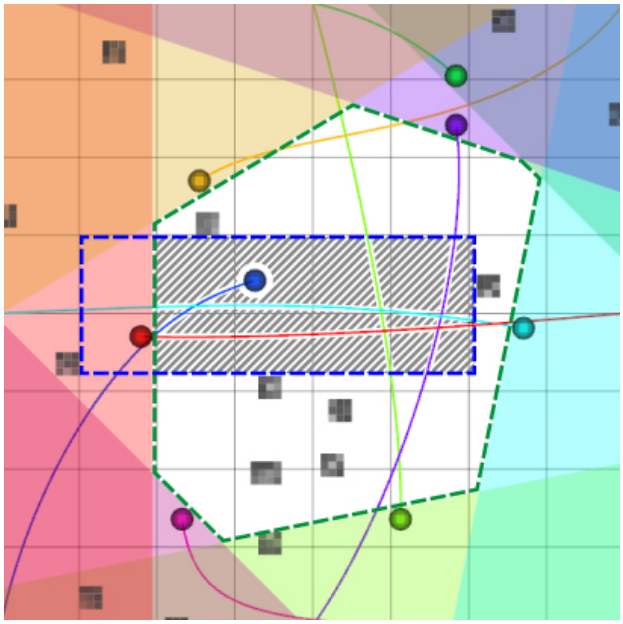


Figure 3.2: Feasible region between relative safe flight corridor (RSFC) and safe flight corridor (SFC).

now being adopted in the multi-agent case [9,15]. Inspired by that, we first plan the discrete initial trajectory by using a graph-based MAPF algorithm and utilize it to construct SFC and RSFC.

An initial trajectory of the i^{th} quadrotor, $\pi^i = \{\pi_0^i, ..., \pi_M^i\}$, is defined as a path that satisfies the following conditions for all $m = 0, ..., M$ and $i \neq j$:

$$\pi_0^i = s^i, \pi_M^i = g^i \tag{3.14}$$

$$\langle \pi_{m-1}^i, \pi_m^i \rangle \oplus \mathcal{C}_{obs}^i \subset \mathcal{F} \tag{3.15}$$

$$\langle \pi_{m-1}^{i,j}, \pi_m^{i,j} \rangle \cap \mathcal{C}_{inter}^{i,j} = \emptyset \tag{3.16}$$

where $\langle \pi_{m-1}^i, \pi_m^i \rangle = \{\alpha \pi_{m-1}^i + (1-\alpha)\pi_m^i \mid 0 \leq \alpha \leq 1\}$ is a line segment between waypoints π_{m-1}^i and π_m^i, and $\pi_m^{i,j} = \pi_m^j - \pi_m^i$ is a relative waypoint between two agents. (3.15) shows that the initial trajectory does not collide with obstacles, and (3.16) means that the agents do not collide with other agents when all the agents move along their initial trajectory at constant velocity.

To plan an initial trajectory, we use a graph-based multi-agent pathfinding (MAPF) algorithm. There have been many researches about MAPF algorithm such as HCA* [16], M* [17], conflict-based search (CBS) [18]. Among them, we choose enhanced conflict-based search (ECBS) for the following two reasons: (i) ECBS can find a suboptimal solution in a short time. Because the

optimal MAPF algorithm is NP-complete [19], it could be better to use a suboptimal MAPF solver with respect to computation time. (ii) The ECBS algorithm is complete. To guarantee the completeness of trajectory optimization algorithm, individual submodules in the algorithm must be complete.

Initial trajectory planning process is as follows. First, we translate the given 3D occupancy map into a 3D grid map with grid size d. Next, we set constraints which determine conflict in the ECBS algorithm to satisfy the condition (3.16). After that, we give start and goal points as the input and compute the initial trajectory. If start and goal points are not located on the 3D grid map, then we use the nearest grid points instead and append the start/goal points to both ends respectively. We note that this algorithm always return a solution due to the third assumption in Section 3.1.1.

3.2.2 Safe flight corridor

The m^{th} safe flight corridor (SFC) of the i^{th} quadrotor, $\mathcal{S}_m^i$, is defined as a convex set satisfies following conditions:

$$\mathcal{S}_m^i \oplus \mathcal{C}_{obs}^i \subset \mathcal{F} \tag{3.17}$$

$$\langle \pi_{m-1}^i, \pi_m^i \rangle \subset \mathcal{S}_m^i \tag{3.18}$$

The condition (3.17) shows that an agent in SFC does not collide with obstacles, so SFC can be used for obstacle collision avoidance. We note that there always exists a convex set that satisfies Eq. (3.17)) and Eq. (3.18), for given initial trajectory π^i (e.g. $\langle \pi_{m-1}^i, \pi_m^i \rangle$).

Theorem 1 shows that obstacle avoidance constraint can be represented in convex constraint.

Theorem 1 *The i^{th} quadrotor does not collide with obstacle if control point of $p^i(t)$ satisfies below equation for all $m = 1, ..., M$, and $k = 0, ..., n$.*

$$c_{m,k}^i \in \mathcal{S}_m^i \tag{3.19}$$

Proof 1 *SFC is a convex set, so we can apply the convex hull property of Bernstein polynomial for the each segment of the trajectory:*

$$p_m^i(t) \in conv\{c_{m,0}^i, ..., c_{m,n}^i\} \subset \mathcal{S}_m^i, t \in [T_{m-1}, T_m] \tag{3.20}$$

By the Eq. (3.20) *and Eq.* (3.17)*:*

$$p_m^i(t) \oplus \mathcal{C}_{obs}^i \subset \mathcal{F}, t \in [T_{m-1}, T_m] \tag{3.21}$$

which is equal to obstacle collision avoidance condition Eq. (3.10)

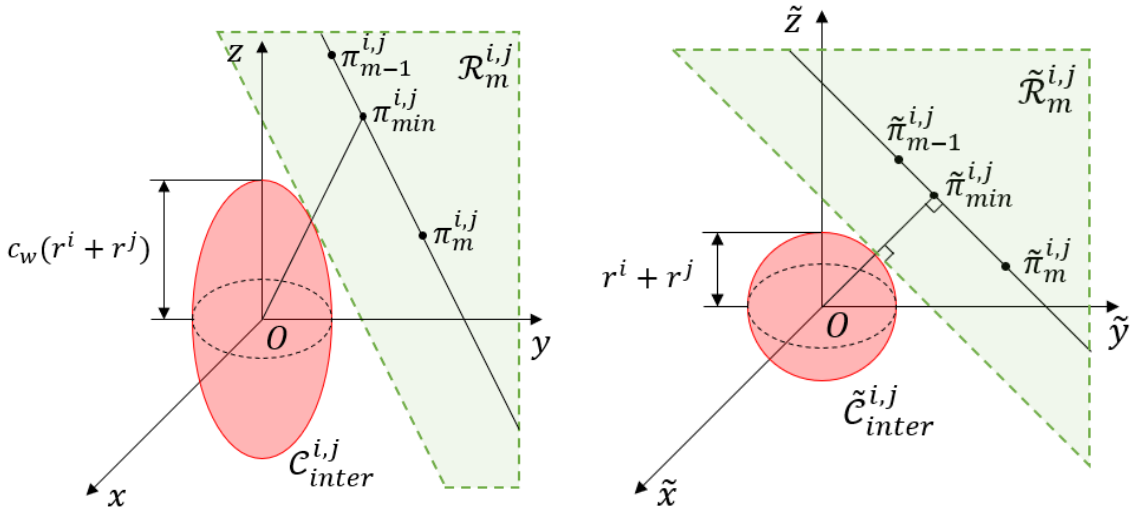
Alg. 1 shows the construction process of safe flight corridor (SFC). We initialize SFC to $\langle \pi_{m-1}^i, \pi_m^i \rangle$ to fulfill the condition (3.18) (line 3). For all direction, we check whether SFC is expandable (line 5-9), and we expand SFC by a pre-specified length (line 10). This algorithm guarantees to return convex sets that satisfy the definition of SFC. In practice, we skip the redundant expansion process to reduce computation time. In other words, we use previously generated SFC $\mathcal{S}_{m-1}^i$ as $\mathcal{S}_m^i$ when $\mathcal{S}_{m-1}^i$ satisfies the definition of $\mathcal{S}_m^i$.

Algorithm 1: buildSFC

Input: initial trajectory π^i, 3D occupancy map $\mathcal{E}$

Output: safe flight corridor $\mathcal{S}^i = (\mathcal{S}_0^i, ..., \mathcal{S}_M^i)$

```
1  D ← {±x, ±y, ±z};
2  for m ← 1 to M do
3      S_m^i ← ⟨π_{m-1}^i, π_m^i⟩;
4      while D is not empty do
5          for μ in D do
6              if S_m^i cannot expand to direction μ then
7                  D ← D\μ
8              end
9          end
10         expand S_m^i to all direction in D;
11     end
12 end
```



(a) Before coordinate transformation.

(b) After coordinate transformation.

Figure 3.3: Construction of relative safe flight corridor.

3.2.3 Relative safe flight corridor

The m^{th} relative safe flight corridor (RSFC) between i^{th} and the j^{th} quadrotor, $\mathcal{R}^{i,j}_m$, is defined as a convex set that satisfies the following conditions:

$$\mathcal{R}^{i,j}_m \cap \mathcal{C}^{i,j}_{inter} = \emptyset \tag{3.22}$$

$$\langle \pi^{i,j}_{m-1}, \pi^{i,j}_m \rangle \subset \mathcal{R}^{i,j}_m \tag{3.23}$$

If $\mathcal{R}^{i,j}_m$ includes $p^{i,j}(t)$ for $t \in [T_{m-1}, T_m]$, then there is no collision between the i^{th} and j^{th} agents for $t \in [T_{m-1}, T_m]$ due to (3.12) and (3.22). For this reason, we can use RSFC to avoid collision between agents. We note that there always exists a convex set that satisfies above conditions, for given initial trajectories π^i, π^j (e.g. $\langle \pi^{i,j}_{m-1}, \pi^{i,j}_m \rangle$).

Similar to SFC, we can prove that inter-collision avoidance constraint can be represented in convex constraint.

Theorem 2 *The i^{th} quadrotor does not collide with the j^{th} quadrotor if control point of $p^i(t)$ and $p^j(t)$ satisfies below equation for all $m = 1, ..., M$, and $k = 0, ..., n$.*

$$c^j_{m,k} - c^i_{m,k} \in \mathcal{R}^{i,j}_m \tag{3.24}$$

Proof 2 *RSFC is a convex set, so we can apply the convex hull property of Bernstein polynomial for the each segment of the trajectory:*

$$p_m^{i,j}(t) \in conv\{c_{m,0}^j - c_{m,0}^i, ..., c_{m,0}^j - c_{m,n}^i\} \subset \mathcal{R}_m^{i,j}, t \in [T_{m-1}, T_m] \tag{3.25}$$

By the Eq. (3.25) *and Eq.* (3.22)*:*

$$p_m^{i,j}(t) \cap \mathcal{C}_{inter}^{i,j} = \emptyset, t \in [T_{m-1}, T_m] \tag{3.26}$$

which is equal to inter-collision avoidance condition Eq. (3.12)

To build RSFC, we first perform affine coordinate transformation $\widetilde{x} = E^{\frac{1}{2}}x$, where $E^{\frac{1}{2}}$ is $diag([1, 1, 1/c_{dw}])$. Then, the inter-collision model $\mathcal{C}_{inter}^{i,j}$ and initial trajectory $\pi^{i,j}$ are transformed to $\widetilde{\mathcal{C}}_{inter}^{i,j}$ and $\widetilde{\pi}^{i,j}$ as shown in Fig. 3.3b. In Fig. 3.3b, the red ellipsoid is an inter-collision model between the quadrotors i, j, and the green-shaded region is the relative safe flight corridor (RSFC). Let $\widetilde{\pi}_{min}^{i,j}$ be the nearest point of $\langle \widetilde{\pi}_{m-1}^{i,j}, \widetilde{\pi}_m^{i,j} \rangle$ to the origin. We construct RSFC as follows:

$$\mathcal{R}_m^{i,j} = \{x = E^{-\frac{1}{2}}\widetilde{x} \mid \widetilde{x} \cdot \widetilde{n}_{min} - (r^i + r^j) > 0\} \tag{3.27}$$

where $\widetilde{n}_{min} = \widetilde{\pi}_{min}^{i,j} / \|\widetilde{\pi}_{min}^{i,j}\|$. As depicted in Fig. 3.3a, our RSFC is a half-space divided by the plane, which is tangent to the inter-collision model at the $\pi_{min}^{i,j} = E^{-\frac{1}{2}}\widetilde{\pi}_{min}^{i,j}$. We note that the convex set in (3.27) satisfies the definition of RSFC.

Lemma 1 *Assume that* $\widetilde{x} \in \langle \widetilde{\pi}_{m-1}^{i,j}, \widetilde{\pi}_m^{i,j} \rangle$*, and* $\widetilde{x} \cdot \widetilde{n}_{min} - (r^i + r^j) > 0$*. Then* $\|x\| > (r^i + r^j)$*.*

Proof of Lemma 1 *Due to the assumption of the lemma:*

$$\widetilde{x} \cdot \widetilde{n}_{min} = \|\widetilde{x}\|(\frac{\widetilde{x}}{\|\widetilde{x}\|} \cdot \widetilde{n}_{min}) > r^i + r^j \tag{3.28}$$

$$\|\widetilde{x}\| > \frac{r^i + r^j}{(\widetilde{x}/\|\widetilde{x}\|) \cdot \widetilde{n}_{min}} \geq r^i + r^j \tag{3.29}$$

Thus, $\|x\| > (r^i + r^j)$*.*

Lemma 2 *Assume that* $\widetilde{x} \in \langle \widetilde{\pi}_{m-1}^{i,j}, \widetilde{\pi}_m^{i,j} \rangle$ *satisfies* $\|\widetilde{x}\| > (r^i + r^j)$*. Then* $\widetilde{x} \cdot \widetilde{n}_{min} - (r^i + r^j) > 0$

Proof of Lemma 2 *Since* $\widetilde{\pi}^{i,j}_{min}$ *is the nearest point of* $\langle \widetilde{\pi}^{i,j}_{m-1}, \widetilde{\pi}^{i,j}_{m} \rangle$ *to the origin, the following equation is satisfied.*

$$\widetilde{\pi}^{i,j}_{min} \cdot \frac{(\widetilde{x} - \widetilde{\pi}^{i,j}_{min})}{\|\widetilde{x} - \widetilde{\pi}^{i,j}_{min}\|} \geq 0 \tag{3.30}$$

Using Eq. (3.30)*, we can obtain the below result.*

$$\widetilde{x} \cdot \widetilde{n}_{min} - (r^i + r^j) > \widetilde{x} \cdot \widetilde{n}_{min} - \|\widetilde{\pi}^{i,j}_{min}\| \geq 0 \tag{3.31}$$

Theorem 3 *A convex set in Eq.* (3.27) *satisfies the definition of RSFC, Eq.* (3.22) *and Eq.* (3.23)

Proof 3 *Due to the Lemma 1, the vector* $x \in \mathcal{R}^{i,j}_{m}$ *satisfies the following equation:*

$$\|E^{\frac{1}{2}}x\| > r^i + r^j \tag{3.32}$$

$E^{\frac{1}{2}}$ *is a diagonal matrix, so we can show that* $\mathcal{R}^{i,j}_{m}$ *in Eq.* (3.27) *satisfies Eq.* (3.22)*:*

$$x^T E x > (r^i + r^j)^2 \tag{3.33}$$

Furthermore, the vector $x \in \langle \pi^{i,j}_{m-1}, \pi^{i,j}_{m} \rangle$ *satisfies the below equation due to the Lemma 2.*

$$E^{\frac{1}{2}}x \cdot \widetilde{n}_{min} - (r^i + r^j) > 0 \tag{3.34}$$

, which implies that $\mathcal{R}^{i,j}_{m}$ *satisfies the condition Eq.* (3.23)*. Thus,* $\mathcal{R}^{i,j}_{m}$ *satisfies the definition of RSFC.*

3.3 Trajectory optimization

In this section, we introduce the trajectory optimization algorithm using convex safe corridors. Alg. 2 shows the process of trajectory optimization. First, we plan initial trajectories (line 1), and we use them to determine safe flight corridor (SFC) (line 3) and relative safe flight corridor (RSFC) (line 5). After that, we compose the quadratic programming (QP) problem using initial trajectories and safe corridors (line 8):

$$\begin{aligned} \text{minimize} \quad & c^T Q c \\ \text{subject to} \quad & A_{eq} c = b_{eq} \\ & c^i_{m,k} \in \mathcal{S}^i_m, \qquad\qquad \forall i, m, k \\ & c^j_{m,k} - c^i_{m,k} \in \mathcal{R}^{i,j}_m \quad \forall i, j > i, m, k \end{aligned} \tag{3.35}$$

As you can see in Eq. (4.2), we do not consider dynamic limits in the QP problem because they can be infeasible constraints for QP. Instead, similar to [9], we uniformly scale the total flight time to satisfy dynamic feasibility constraints after optimization (line 9). This uniform time scaling does not affect spatial path of the trajectory due to the property of hover to hover polynomial trajectory.

Algorithm 2: Trajectory Planning Algorithm

Input: start point s^i, goal point g^i for agents $i \in \{1, ..., N\}$, 3D occupancy map $\mathcal{E}$

Output: total flight time T, trajectory $p^i(t)$ for agents $i \in \{1, ..., N\}$, $t \in [0, T]$

1 $\pi = (\pi^1, ..., \pi^N) \leftarrow$ planInitialTraj($s^{\forall i}$, $g^{\forall i}, \mathcal{E}$);
2 **for** $i \leftarrow 1$ **to** N **do**
3 $\mathcal{S}^i = (\mathcal{S}^i_0, ..., \mathcal{S}^i_M) \leftarrow$ buildSFC($\pi^i, \mathcal{E}$);
4 **for** $j \leftarrow i + 1$ **to** N **do**
5 $\mathcal{R}^{i,j} = (\mathcal{R}^{i,j}_0, ..., \mathcal{R}^{i,j}_M) \leftarrow$ buildRSFC(π^i, π^j);
6 **end**
7 **end**
8 $p^0(t), ..., p^N(t) \leftarrow$ trajOpt($\pi, \mathcal{S}^{\forall i}, \mathcal{R}^{\forall i, j>i}$);
9 $T, p^0(t), ..., p^N(t) \leftarrow$ timeScale($p^0(t), ..., p^N(t)$);
10 **return** $T, p^0(t), ..., p^N(t)$

Algorithm 2 can compute safe, continuous trajectory for multiple quadrotors, however, optimizing all control points of polynomials at once can cause the scalability problem. It is because the time complexity of the QP solver is about $O(N^3)$. In the next chapter, we will introduce an efficient sequential optimization method that can reduce the time complexity of the optimization process.

4

Sequential optimization with dummy agents

In this chapter, we propose an efficient sequential optimization method using *dummy agents*. This method can improve scalability by dividing the big problem into several small ones, and prevent deadlock by utilizing dummy agents.

Algorithm 3: trajOpt

Input: initial trajectory π, safe flight corridor $\mathcal{S}^{\forall i}$, relative safe flight corridor $\mathcal{R}^{\forall i,j>i}$

Output: trajectory $p^i(t)$ for agents $i \in b$, $t \in [0, T]$

$p_{dmy}(t) = (p^0_{dmy}(t), ..., p^N_{dmy}(t)) \leftarrow$ planDummy(π);

for $l \leftarrow 1$ **to** N_b **do**

  $b \leftarrow$ agents in l^{th} batch;

  $p^b(t) \leftarrow$ solveQP($\pi^b, \mathcal{S}^b, \mathcal{R}^{\forall i,j>i}, p^{\forall i \notin b}_{dmy}(t)$);

  $p_{dmy}(t) \leftarrow p(t)$;

end

return $p^0(t), ..., p^N(t)$

Alg. 3 shows the process of the sequential optimization. First, we generate trajectories for

dummy agents $p_{dmy}(t)$ using the following control points (line 1):

$$c^i_{m,k} = \begin{cases} \pi^i_{m-1} & k = 0, ..., \phi - 1 \\ \pi^i_m & k = n - (\phi - 1), ..., n \\ x \in \langle \pi^i_{m-1}, \pi^i_m \rangle & else \end{cases} \tag{4.1}$$

These dummy agents are used to prevent the previously planned trajectory from blocking the space for the other agents. Next, we divide the agents into N_b batches, and we optimize the trajectory by solving the below QP problem for the batch b (line 3-4)

$$\begin{aligned} \text{minimize} \quad & c^T Q c \\ \text{subject to} \quad & A_{eq} c = b_{eq} \\ & c^i_{m,k} = \text{control points of } p^i_{dmy}(t), \quad \forall i \notin b, m, k \\ & c^i_{m,k} \in \mathcal{S}^i_m, \quad \forall i \in b, m, k \\ & c^j_{m,k} - c^i_{m,k} \in \mathcal{R}^{i,j}_m \quad \forall i, j > i, m, k \end{aligned} \tag{4.2}$$

where $c \in \mathbb{R}^{\frac{N}{N_b} M(n+1)}$, $A_{eq} \in \mathbb{R}^{\frac{N}{N_b}(M+1)\phi \times \frac{N}{N_b} M(n+1)}$, and the number of inequality constraints is $(N - \frac{1}{2}(\frac{N}{N_b} + 1))\frac{N}{N_b} M(n+1)$. $p^i_{dmy}(t)$ is the trajectory for i^{th} dummy agent. At last, we replace the trajectory of dummy agents to the previously planned one (line 5), and plan the trajectory for the next batch sequentially.

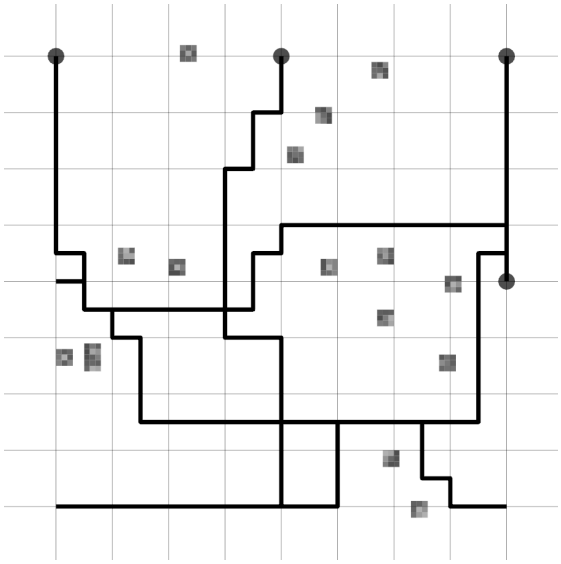
Fig. 4.1 visualizes the overall process. In Fig. 4.1, dummy agent is depicted as a black circle, and agent in the current batch is depicted as a colored circle. For each iteration, we plan a trajectory for the current batch is described as a color line and the trajectory of dummy agents is depicted in a black line. For each iteration, we deploy dummy agents except the agents in the current batch (Fig. 4.1a). Then, we plan the trajectory for the current batch to avoid dummy agents (Fig. 4.1b). After that, agents in the current batch are used as dummy agents at the next iteration (Fig. 4.1c). At the end of the iteration, collision-free trajectories are found without deadlock because all the agents are planned to avoid the previous batch (Fig. 4.1d).

This sequential method can achieve better scalability because we can avoid the high time complexity of QP solver by increasing the number of the batches as the number of agents increases while keeping the same number of decision variables of QP. Furthermore, we can prove that (4.2)

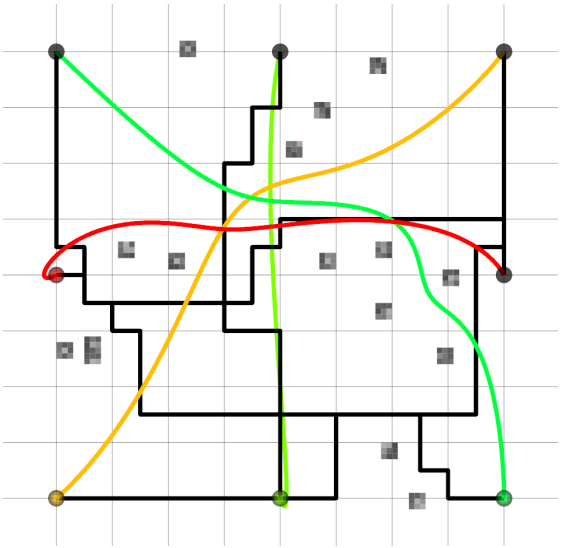
consists of feasible constraints, which means that our method does not cause optimization failure due to infeasible constraints.

Theorem 4 *If $n \geq 2\phi - 1$, then there exists decision vector c that satisfies the constraints of eq. (4.2).*

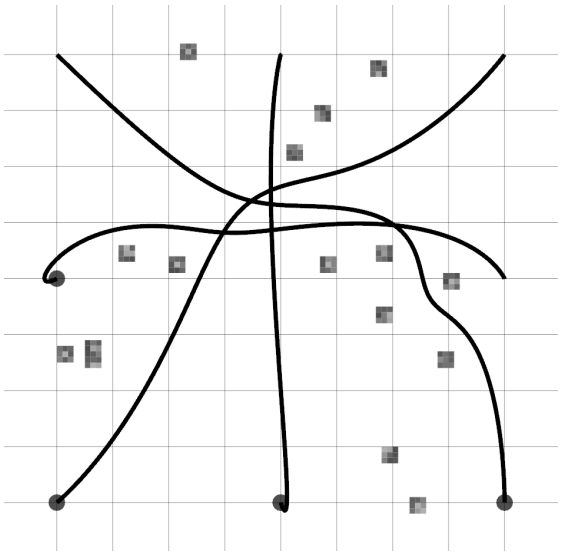
Proof 4 *Let us assign the decision vector c as* (4.1) *for $i = 1, ..., N$ and $m = 1, ..., M$. Then c satisfies the waypoint constraints due to* (3.14)*. $p^i(t)$ is continuous up to $\phi - 1$ derivatives at $t = T_m$ for $m = 1, ..., M - 1$ because $c^i_{m,n-(\phi-1),...,n} = c^i_{m+1,0,...,\phi-1} = \pi^i_m$. c also fulfills safe corridor constraints due to* (3.18) *and* (3.23)*. Thus, c is the decision vector that satisfies the constraints of* (4.2)*.*



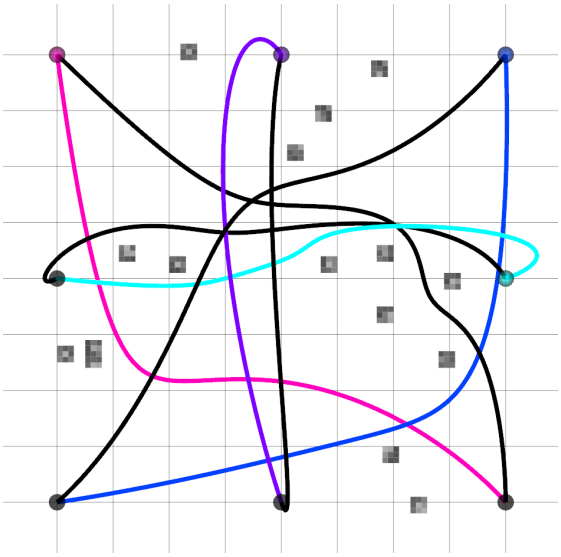
(a) Deploy dummy agents.



(b) Plan for one batch.



(c) Replace dummy agents with the previous batch.



(d) Plan for next batch.

Figure 4.1: Sequential planning with dummy agents when $N_b = 2$.

5

Experimental results

The proposed algorithm is run in C++ and executed the proposed algorithm on a PC running Ubuntu 18.04. with Intel Core i7-7700 @ 3.60GHz CPU and 16G RAM. We model the quadrotor with radius $r^{i=1,\dots,N} = 0.15$m, maximum velocity $v_{max}^{i=1,\dots,N} = 1.7m/s$, maximum acceleration $a_{max}^{i=1,\dots,N} = 6.2m/s^2$ and downwash coefficient $c_{dw} = 2$ based on the specification of Crazyflie 2.0 in [10]. We use the Octomap library [20] to represent the 3D occupancy map, and we use the dynamicEDT3D library [21] to convert occupancy map to 3D grid map. CPLEX QP solver [22] is used to solve QP problem in trajectory optimization. The degree of polynomials is determined to $n = 5$ to satisfy the assumption ($n \geq 2\phi - 1$) in the Theorem 4. We plan the initial trajectory in 3D grid map with grid size $d = 0.5$ m, and set suboptimal bound of ECBS to 1.3.

5.1 Comparison with the previous work

To validate the performance of our proposed algorithm, we compared the result with previous work [5]. [5] also utilizes RSFC for inter-collision avoidance, but it uses the RSFC candidate method, which constructs RSFC by choosing the proper one of pre-defined RSFC candidates. We conducted the simulations in 50 random forests. Each forest has a size of 10 m $\times$ 10 m $\times$ 2.5 m and contains randomly deployed 20 trees of size 0.3 m $\times$ 0.3 m $\times$ 1–2.5 m. We assigned start point of quadrotors in a boundary of the xy-plane in 1 m height, and the goal points at the opposite to

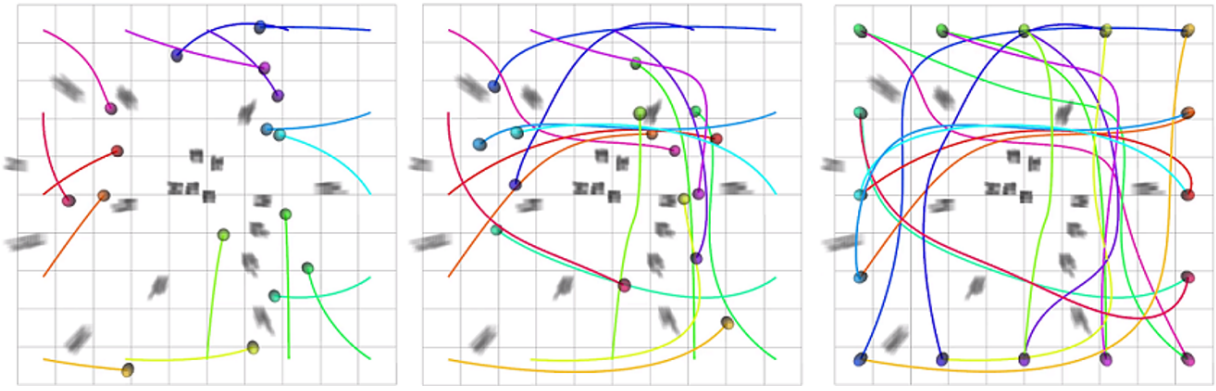


Figure 5.1: Trajectory generated for 16 agents in a 10 m × 10 m × 2.5 m random forest.

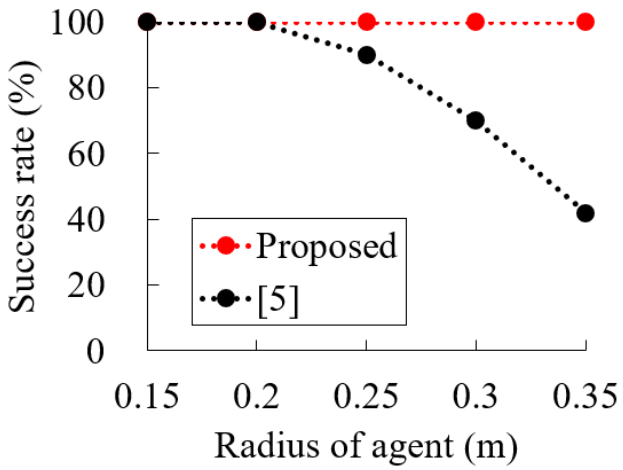


Figure 5.2: Success rate of two trajectory generation methods for 16 agents.

their start position. Fig. 5.1 shows the simulation result of the proposed method with 16 agents. Agents are marked with colored circles at the goal (assigned to the opposite of the start points), along with their trajectories.

5.1.1 Success rate

We executed the simulation with 16 agents, and measured the success rate by the size of agents. As shown in the Fig. 5.2, both methods show a 100% success rate in 50 random forest when the radius of agents is small, but the success rate of [5] decreases as the size of agents increases. It is because the larger the agent size, the smaller the space for agents can exist, which lead to

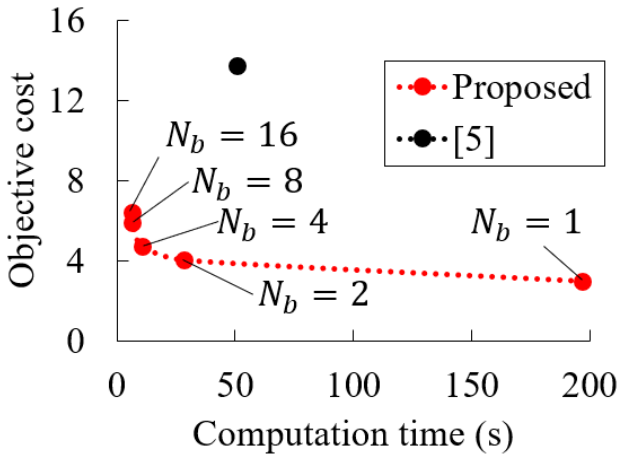


Figure 5.3: Objective cost vs computation time for 64 agents.

the higher probability that the constraints for SFC and RSFC are infeasible each other. On the contrary, the proposed method shows a perfect success rate for all the case because we design SFC and RSFC to feasible each other.

5.1.2 Solution quality

As described in the Fig. 5.3 and Table 5.1, the proposed algorithm shows better performance with respect to both objective cost and computation time compare to previous work when the number of the batch N_b is more than one. It can generate a trajectory for 64 agents in 6.36 s ($N_b = 16$), and it has 78% ($N_b = 1$), 53% ($N_b = 16$) less objective cost. Note that we can adjust N_b depending on the desired objective cost and computation time.

5.1.3 Scalability analysis

The computation time increment by the number of agents is shown in Table 5.2. The numbers in parentheses represent the computation time increment when the number of agents is doubled. When the number of agents is small, the computation time increases linearly, regardless of the trajectory optimization method, but it follows the time complexity of QP solver as the number of agents increases if we do not adopt the sequential optimization method. On the other hand, if we maintain the size of the batch (N/N_b), it still shows good scalability with the high number of agents.

Table 5.1: Performance comparison with previous work [5]

	Computation Time (s)					Objective Cost
Agents	4	8	16	32	64	64
[5]	0.093	0.19	0.81	5.30	51.1	13.7
Proposed ($N_b = 1$)	0.11	0.29	1.15	11.1	197.0	2.98
Proposed ($N_b = 2$)	0.091	0.23	0.68	2.97	28.7	4.02
Proposed ($N_b = 4$)	0.089	0.19	0.59	1.71	10.8	4.68
Proposed ($N_b = 8$)	-	0.19	0.45	1.55	6.38	5.86
Proposed ($N_b = 16$)	-	-	0.45	1.17	6.36	6.39

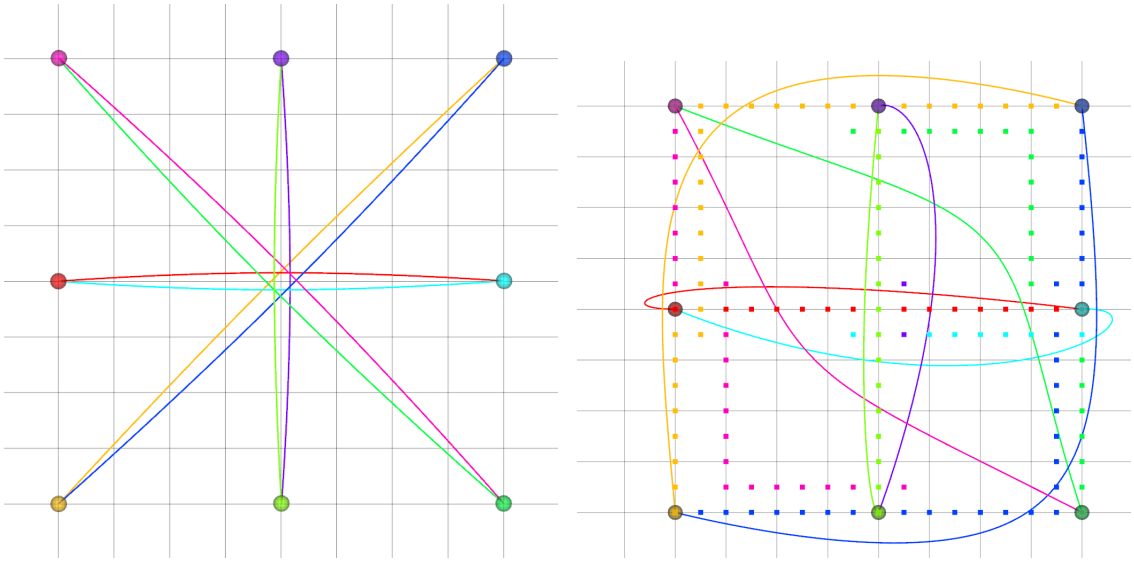
Table 5.2: Scalability with previous work [5].

	Computation Time (s)				
Agents	4	8	16	32	64
[5]	0.093	0.19 (×2.0)	0.81 (×4.3)	5.30 (×6.5)	51.1 (×9.6)
Proposed ($N_b = 1$)	0.11	0.29 (×2.7)	1.15 (×3.9)	11.1 (×9.6)	197 (×17.8)
Proposed ($N/N_b = 4$)	0.11	0.23 (×2.2)	0.59 (×2.5)	1.55 (×2.6)	6.36 (×4.1)

5.2 Comparison with SCP-based method

We compared the proposed algorithm with SCP-based method [7]. Experiments are done in 10 m × 10 m × 2.5 m empty space with 8 agents. Start position and goal points are same as the previous experiment, and we assigned the same total flight time to both algorithm. When we run the SCP method, we did not consider the case when the time step h is over 1 because it ignores collision avoidance constraint at all. Fig. 5.4 shows the trajectory planning result of two methods. In Fig. 5.4, the dots in (b) are the initial trajectory of corresponding agents.

Table 5.3 shows that the proposed algorithm requires less computation time for all the cases,



(a) SCP-based method [7]

(b) Proposed method

Figure 5.4: Trajectory planning result of the propose algorithm and SCP-based method in empty space.

and this result does not change when we stop the SCP at the first iteration with collision avoidance constraints. The third column shows the safety margin ratio of each method. Safety margin ratio γ is calculated as follows:

$$\gamma = \mathrm{argmin}_{i,j} \frac{d_{min}^{i,j}}{r^i + r^j} \tag{5.1}$$

where $d_{min}^{i,j}$ is a minimum distance between two agents i, j. Safety margin ratio must be over 100% to guarantee the collision avoidance, however, SCP-based method does not satisfy this because SCP checks only collision avoidance between discrete points on each trajectory. On the contrary, the proposed method satisfies the safety condition completely.

Although the proposed method perform better in computation time and safety margin, it has longer total flight distance compared to the SCP method. It is because our initial trajectory is not optimal respect to total flight distance in non-grid space. Thus, we need to plan initial trajectory considering total flight distance, and leave it as future work.

Table 5.3: Comparison of proposed algorithm and SCP-based method.

	Comp. Time per Iter. (s)	Total Comp. Time (s)	Safety Margin Ratio	Total Flight Dist. (m)
SCP ($h = 1.0$ s)	0.78	2.80	12%	77.29
SCP ($h = 0.5$ s)	5.5	20.5	81%	77.36
SCP ($h = 0.34$ s)	16.2	60.4	92%	77.38
SCP ($h = 0.25$ s)	42.1	156.6	96%	77.40
Proposed ($N_b = 1$)	-	0.65	101%	90.74

5.3 Flight test

We conducted real flight test with 6 Crazyflie 2.0 quadrotors in a 5 m x 7 m x 2.5 m space. We used Crazyswarm [23] to follow the pre-computed trajectory, and we used Vicon motion capture system to obtain the position information at 100 Hz. The desired trajectory of the flight test is depicted in Fig. 5.5, and the snapshots of the flight test are shown in Fig. 5.6.

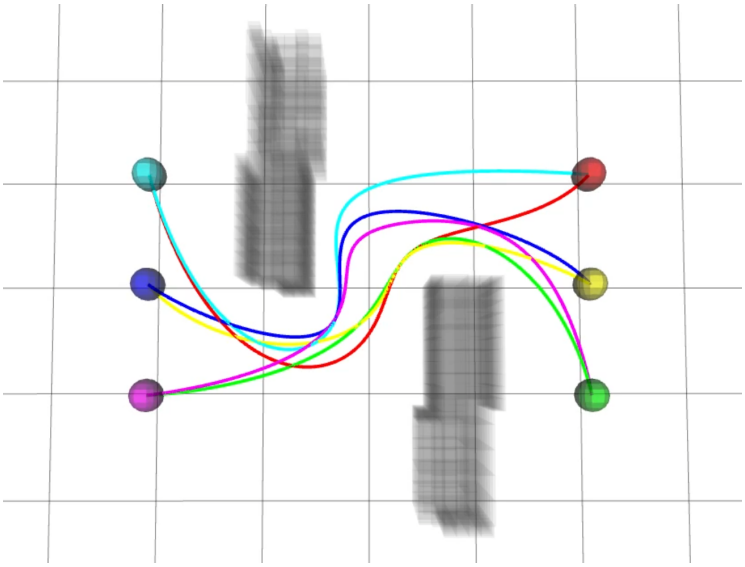


Figure 5.5: The desired trajectory of the flight test with 6 quadrotors.

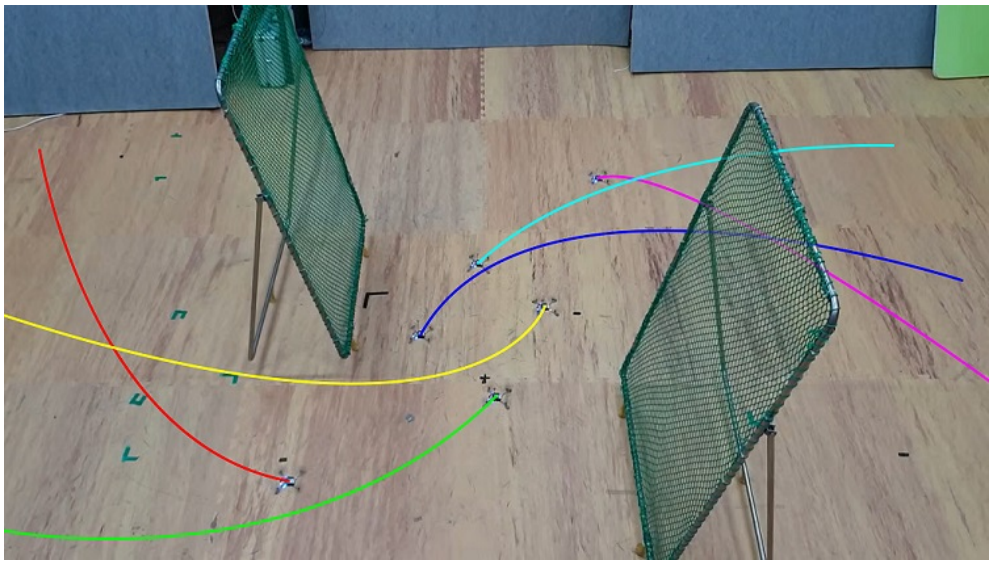


Figure 5.6: Flight in an obstacle environment with 6 quadrotors.

6

Conclusion

We presented an efficient trajectory planning algorithm for multiple quadrotors in obstacle environments, combining the advantages of grid-based and optimization-based planning algorithm. Using relative Bernstein polynomial, we reformulated trajectory generation problem to convex optimization problem, which guarantees to generate continuous, collision-free, and dynamically feasible trajectory. We improved the scalability of the algorithm by using sequential optimization method, and we proved overall process does not cause the failure of optimization if there exist initial trajectory. The proposed algorithm shows considerable reduction in computation time and objective cost compared to our previous work, and it shows better performance in computation time and safety, compared to SCP-based method.

In future work, we plan to develop initial trajectory planner that optimizes total flight distance in non-grid space, and we will extend our work to dynamic obstacle environment.

References

[1] D. Bareiss and J. 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.

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

[3] Y. Chen, M. Cutler, and J. P. How, "Decoupled multiagent path planning via incremental sequential convex programming," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on.* IEEE, 2015, pp. 5954–5961.

[4] 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.

[5] J. Park and H. J. Kim, "Fast trajectory planning for multiple quadrotors using relative safe flight corridor," 2019.

[6] D. Mellinger, A. Kushleyev, and V. 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.

[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 *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on.* IEEE, 2012, pp. 1917–1922.

[8] D. R. Robinson, R. T. Mar, K. Estabridis, and G. Hewer, "An efficient algorithm for optimal trajectory generation for heterogeneous multi-agent systems in non-convex environments," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1215–1222, 2018.

[9] 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.

[10] M. Debord, W. Hönig, and N. Ayanian, "Trajectory planning for heterogeneous robot teams," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 7924–7931.

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

[12] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1294–1310, 2015.

[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] S. Tang and V. 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.

[15] Y. Xu, S. Lai, J. Li, D. Luo, and Y. You, "Concurrent optimal trajectory planning for indoor quadrotor formation switching," *Journal of Intelligent & Robotic Systems*, pp. 1–18, 2018.

[16] D. Silver, "Cooperative pathfinding." *AIIDE*, pp. 117–122, 2005.

[17] G. Wagner and H. 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] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[19] J. Yu and S. M. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[20] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, no. 3, pp. 189–206, 2013.

[21] B. Lau, C. Sprunk, and W. Burgard, "Efficient grid-based spatial representations for robot navigation in dynamic environments," *Robotics and Autonomous Systems*, vol. 61, no. 10, pp. 1116–1130, 2013.

[22] I. CPLEX, "12.7. 0 user's manual," 2016.

[23] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 3299–3304.

국 문 초 록

무인비행체(UAV)로 구성된 다중 에이전트 시스템은 높은 기동성 및 응용 가능성으로 많은 산업 분야에서 관심을 받고 있다. 이러한 다중 에이전트 시스템을 안전하게 운용하려면 안전하고 동적으로 실현 가능 경로를 생성할 수 있는 경로 계획 알고리즘이 필요하다. 그러나 기존의 다중 에이전트 경로 계획 방법은 장애물 환경에서 교착 상태나 부적절한 충돌 회피 조건으로 인한 최적화 실패가 일어날 수 있다는 한계가 있다. 본 논문에서는 장애물 환경에서 해의 존재를 보장하도록 다중 에이전트 경로 계획 문제를 변환한 뒤 이를 효율적으로 풀어낼 수 있는 새로운 경로 계획 알고리즘을 제시한다. 이 알고리즘은 그리드 기반 접근법과 최적화 기반 접근법의 장점을 모두 가지도록 설계되었으며, 불가능한 충돌 구속조건을 부과하지 않고 안전하고 동적으로 실현 가능한 궤적을 생성할 수 있다. 이 알고리즘은 더미 에이전트(dummy agents)을 이용한 순차 최적화 방법을 사용하여 알고리즘의 확장성(scalability)을 높였으며, 번스타인(Bernstein) 다항식의 볼록 껍질(convex hull) 성질을 활용하여 볼록하지 않은 충돌 회피 제약 조건을 볼록화하였다. 제안된 알고리즘의 성능은 선행 연구와 SCP 기반 방법과의 비교를 통해 검증되었다. 제안된 방법은 선행 연구에 비해 목표 비용의 50% 이상 절감하였으며, SCP 기반 방법에 비해 계산 시간의 75% 이상 감소하였다. 또한 제안된 방법은 인텔 코어 i7-7700 @ 3.60GHz CPU 및 16G RAM 환경에서 64개 에이전트의 궤적을 계산하는데 평균 6.36초가 소요된다.

주요어 : 다중 에이전트 경로 계획, 충돌 회피, 쿼드로터.

학번 : 2018-28113