공학박사학위논문

Decentralized Trajectory Planning for Quadrotor Swarm in Cluttered Environments with Goal Convergence Guarantee

장애물 환경에서 목적지 도달을 보장하는 쿼드로터 군집 분산 경로 계획

2023년 8월

서울대학교 대학원

항공우주공학과

박 정 원

Decentralized Trajectory Planning for Quadrotor Swarm in Cluttered Environments with Goal Convergence Guarantee

장애물 환경에서 목적지 도달을 보장하는 쿼드로터 군집 분산 경로 계획

지도교수 김 현 진

이 논문을 공학박사 학위논문으로 제출함

2023년 5월

서울대학교 대학원

기계항공공학부

박 정 원

박정원의 공학박사 학위논문을 인준함

2023년 6월

위 원 장 : _____

부위원장 : _____

위 원 : _____

위 원 : _____

위 원 : _____

Decentralized Trajectory Planning for Quadrotor Swarm in Cluttered Environments with Goal Convergence Guarantee

A Dissertation

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

DOCTOR OF PHILOSOPHY

Department of Aerospace Engineering

Seoul National University

Supervisor : Professor H. Jin Kim

AUGUST 2023

Decentralized Trajectory Planning for Quadrotor Swarm in Cluttered Environments with Goal Convergence Guarantee

JUNGWON PARK

Department of Aerospace Engineering

Seoul National University

APPROVED:

Youdan Kim, Chair, Ph.D.

H. Jin Kim, Ph.D.

Chan Gook Park, Ph.D.

Jaehyun Yoo, Ph.D.

Hyeonbeom Lee, Ph.D.

to my

FAMILY

with love

Abstract

Decentralized Trajectory Planning for Quadrotor Swarm in Cluttered Environments with Goal Convergence Guarantee

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

This dissertation presents an online distributed multi-agent trajectory planning (MATP) algorithm for a quadrotor swarm in a maze-like dynamic environment. For deadlock resolution, the proposed algorithm utilizes the subgoal optimization method that ensures the agent converges to the subgoal without deadlock and uses a waypoint from a grid-based multi-agent path planning (MAPP) algorithm to guide the subgoal to the desired goal. In addition, the proposed algorithm adopts a safe flight corridor (SFC) and linear safe corridor (LSC) for static obstacle avoidance and inter-agent collision avoidance. As a result, the proposed algorithm guarantees collision avoidance, the feasibility of the constraints, and goal convergence in a static obstacle-rich environment under a limited communication range. For dynamic obstacle avoidance, a relative safe flight corridor (RSFC) is introduced to cover the reachable region of the dynamic obstacles. Moreover, priority-based MAPP is adopted to prevent other agents' interference when avoiding dynamic obstacles. To verify the proposed algorithm, the simulation was conducted in an empty space, random forest, and maze. In an obstacle-free space, the proposed method can compute the trajectories for 70 agents on average 9.69 ms per agent with an Intel i7 laptop and shows the perfect success rate. Also, our method shows 44.7% shorter flight time than buffered Voronoi cell (BVC) and 23% shorter than with our previous work. The proposed algorithm shows the highest success rate and shortest flight time compared to state-of-the-art baseline algorithms. In particular, the proposed algorithm shows over 90% success rate when the velocity of moving

obstacles is below the agent's maximum speed. The safety and robustness of the proposed algorithm were validated through a hardware demonstration with ten quadrotors and one pedestrian in a maze-like environment.

Keywords: Path Planning for Multiple Mobile Robots or Agents, Collision Avoidance, Deadlock Resolution, Distributed Robot Systems.

Student Number: 2020-39650

Table of Contents

List of Tables

List of Figures

1 Introduction

Multi-robot systems with a group of mobile robots or unmanned aerial vehicles (UAVs) have received considerable attention due to their application such as transportation, surveillance, search, and rescue. There may be a need to deploy them in a workspace involving dynamic obstacles or humans, such as a warehouse [3] or office [4], which raises a need for a reliable multi-agent trajectory planning (MATP) algorithm.

Among many MATP algorithms, decentralized approaches have received much attention due to their high scalability and low computation load, which enables online planning. However, it has been challenging for the decentralized MATP algorithm to generate a safe trajectory in a cluttered dynamic environment for the following reasons. First, the MATP algorithm must guarantee deadlock-free to ensure the agents reach their desired goal. However, many decentralized algorithms have a risk of causing a deadlock even in sparse environments [5, 6, 7]. Second, the trajectory planning algorithm must guarantee collision avoidance and dynamical feasibility for safety, and it should guarantee the feasibility of the collision constraints to prevent optimization failure during the flight. However, as the number of agents increases, it becomes more difficult to guarantee all of them at the same

time. Third, the trajectories of dynamic obstacles are unknown in a general situation, so we have to consider the uncertainty of the obstacle's maneuver during trajectory planning. It makes collision avoidance more difficult since there is not enough space to move in the maze-like environment.

This dissertation presents an online distributed trajectory planning algorithm that can generate a safe, deadlock-free trajectory in a cluttered environment with dynamic obstacles. The proposed method solves a deadlock through the following three steps. First, the proposed algorithm places a subgoal in a feasible region that satisfies the collision constraints and the communication range constraints. This subgoal allows the agent to reach the subgoal without a deadlock. Next, subgoal optimization is conducted to make the subgoal converges to the waypoint, which is on the grid vertex. Finally, a decentralized grid-based multi-agent path planning (MAPP) algorithm is utilized to guide the waypoint to the desired goal. As a result, the proposed algorithm can guide the agent to the desired goal. It should be noted that if there is no dynamic obstacle and the communication range is large enough, the proposed algorithm guarantees *goal convergence*, which means that the agent can reach their desired goal. The proposed algorithm adopts a linear safe corridor (LSC) [1] to guarantee the feasibility of the optimization problem and collision avoidance, and it utilizes a relative safe flight corridor (RSFC) [8] to avoid dynamic obstacles. In addition, the grid-based MAPP algorithm is revised to prioritize dynamic obstacle avoidance when the agents meet the obstacle in a narrow corridor. The proposed algorithm can be employed for robots with a limited communication range as long as they can configure a mobile ad-hoc network. To the best of our knowledge, this is the first decentralized MATP algorithm that guarantees the feasibility of the optimization problem, collision avoidance, and deadlock-free in a dense maze-like environment.

I compare the proposed algorithm with state-of-the-art methods, buffered Voronoi cell (BVC) [2], DMPC [9], EGO-Swarm [5], and MADER [6] in simulations with the various 2D and 3D environment. Also, I executed an experiment with 10 quadrotors and one pedestrian to verify the robustness of the proposed algorithm.

1.1 Literature Survey

1.1.1 Multi-Agent Collision Avoidance

There have been discussions in literature closely related to our work on multi-agent trajectory planning. In [10, 11, 12], the trajectory generation problems are reformulated as mixed-integer quadratic programming (MIQP) or sequential convex programming (SCP) problems to deal with non-convex collision constraints. 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 steps depending on the size of agents and obstacles. In [13, 14], linearized collision constraints are used to reduce the computation time. They plan an initial trajectory with a grid-based planner and then construct a safe flight corridor (SFC), which is a safe convex region for each agent. However, they require an iterative trajectory refinement process that costs much computation time.

To achieve high scalability, an artificial potential field (APF) [15, 16, 17] has been considered since it requires little computation load. However, these methods are not suitable for a maze-like environment because non-convex obstacles may cause local minima or oscillatory motion. Besides that, a velocity obstacle (VO)-based approach and its variants [18, 19, 20, 21, 22] or optimization-based methods such as on-demand collision avoidance [9, 23] and gradient-based local planning [5, 24] are presented to reduce the computation time, but they cannot guarantee collision avoidance between agents.

For the safety of the multi-robot system, many researchers have studied distributed algorithm that ensures safety with high scalability. The authors of [2] present buffered Voronoi cell (BVC) to separate the safe region for agents, and the author of [25] utilize control barrier function (CBF) and hybrid braking controllers to ensure provably collision-free behaviors. Recently, distributed model predictive control (DMPC) [26, 27, 28, 29, 30] has received much attention due to its versatility and theoretical guarantee of safety and

feasibility. However, most works do not consider non-convex obstacles or dynamic obstacles.

There are several works for collision avoidance in a maze-like environment. In [31], a token-based cooperative strategy is presented to prevent deadlock in a cluttered environment. However, it requires more time for replanning as the number of agents increases since the agents with the token can update their trajectory. The authors of [32] extend VO to handle a complex environment with dynamic obstacles, but it may cause an infeasible optimization problem when the obstacle does not follow the constant velocity assumption. On the other hand, the proposed method guarantees the feasibility of the optimization problem.

To deal with the uncertainty of dynamic obstacles, [33] suggests search-based motion planning that models the moving obstacle as a polyhedron that inflates over time. Similarly, [34, 35] models dynamic obstacles using the constant velocity assumption with Gaussian noise acceleration and finds a collision-free trajectory using nonlinear DMPC. In [36, 37], Hamilton-Jacobi reachability analysis is used to compute the reachable region of the obstacle. However, these methods show a lack of scalability due to their long computation time.

The authors of [6] propose an asynchronous planner that guarantees to generate a safe trajectory by executing a collision-free trajectory through communication between agents. This method can handle a dense environment with static and dynamic obstacles, but it often takes a few seconds to update the agent's trajectory in practice because the planner blocks the update while it receives other agents' trajectories. For this reason, this method may not respond to the unpredictable maneuver of dynamic obstacles unless the exact trajectories of the obstacles are given, which is not realistic in the actual implementation. On the other hand, the proposed method adopts the time-synchronized approach in which all agents plan and execute the trajectory at the same time. Although the synchronized approaches have limitations in fully utilizing the computational capabilities of individual agents, the proposed algorithm can respond to dynamic obstacles within constant time thanks to its short computation time. Also, our method constructs the collision constraints

considering the reachable region of dynamic obstacles, so it does not need exact knowledge of the obstacle's trajectory.

1.1.2 Decentralized Deadlock Resolution

The centralized methods like conflict-based search (CBS) [38] can guarantee no deadlock, but they are not appropriate for online planning due to their long computation time. For this reason, many decentralized algorithms adopt the right-hand rule [2, 25, 39, 40, 7], which moves the goal point to the right side after the deadlock is detected. This approach works well in an obstacle-free environment, but there is a risk of another deadlock even after changing the goal point.

Another deadlock resolution method is to replan each robot's trajectory sequentially. In [41], a local coordinator asks neighboring agents to plan different trajectories until the deadlock is resolved. The authors of [31] introduce a token-based cooperative strategy, that determines which robots to yield the path by bidding. However, under these methods, there are cases where deadlock cannot be resolved by replanning an alternative trajectory of individual agents. The authors of [32] introduce a centralized high-level coordinator for deadlock resolution. This method is suitable for deadlock resolution in a cluttered environment, but all agents must be connected to the centralized coordinator during the entire mission. The authors of [42] utilize the grid-based planner to avoid conflict between agents. However, it often fails to find the discrete path in a compact space because it treats all other agents as static obstacles.

Several works guarantee deadlock-free in obstacle-free or sparse environments. The authors of [43] introduce a warning band to prevent the agents from clustering. In [17], an artificial potential field (APF) is extended to solve the deadlock. The authors of [44] conduct deadlock analysis and resolution for 2 to 3 agents. However, these methods share the same limitation in that they cannot solve deadlock in a cluttered environment such as a maze. In [45, 46], the grid-based MAPP is utilized to solve deadlock, similar to the proposed method. The authors of [45] adopt a mode-switching strategy, which converts the planner

mode to follow the waypoint from MAPP when the deadlock is detected. The authors of [46] utilize the discrete path from MAPP as an initial trajectory. However, these methods do not provide a theoretical guarantee for deadlock resolution. Compared to the previous work [1], the proposed algorithm does not require a fully connected network for collision avoidance, and it guarantees deadlock-free for dense maze-like environments.

1.2 Contributions

The contributions of the dissertation are summarized as follows:

- Decentralized multi-agent trajectory planning algorithm that guarantees collision avoidance, the feasibility of the optimization problem, and goal convergence in a dense maze-like environment.

- Constraint generation method and multi-agent path planning (MAPP) for dynamic obstacle avoidance in narrow space: (i) Relative Safe Flight Corridor for avoiding dynamic obstacles with unknown trajectories, (ii) Priority-based MAPP to prevent other agent's interference when avoiding dynamic obstacles.

Table 1.1 shows the comparison with the state-of-the-art algorithms. As shown in the table, many algorithms including our previous works have a limitation on an applicable workspace due to a lack of consideration of dynamic obstacles or deadlock resolution in the narrow space. On the other hand, the proposed algorithm can deal with various environments, such as a random forest or even a maze for an extreme case. Furthermore, the proposed algorithm guarantees collision avoidance between agents, the feasibility of constraints, and goal convergence if there is no dynamic obstacle. To the best of our knowledge, the proposed approach is the first MATP algorithm that can deal with a maze-like dynamic environment while ensuring inter-collision avoidance, the feasibility of contraints, and goal convergence.

Table 1.1: Comparison with the state-of-the-art algorithms. ✓ means that the algorithm explicitly considers/provides the corresponding item. (CAA: collision avoidance between agents, FO: feasibility of the optimization problem, GC: goal convergence)

Method	Environment		Theoretical Guarantee		
	Maze	Dynamic obstacle	Inter-agent avoidance	Feasibility of constraints	Goal convergence
BVC-based [2]	✕	✕	✓	✕	✕
DMPC [9]	✕	✕	✕	✓	✕
NMPC [30]	✕	✕	✓	✕	✕
EGO-Swarm [5]	△*	✕	✕	✓	✕
MIQP-based [7]	△*	✕	✓	✕	✕
MADER [6]	✕	✓†	✓	✓	✕
VO-based [32]	✓	✓	✕	✕	✕
RSFC [8]	✕	✓	✕	✓	✕
LSC [1]	✓	✕	✓	✓	✕
Proposed	✔	✔	✔	✔	✔

* The result is not reported in the paper, but there is a possibility of extension to a sparse maze.
† Requires the future trajectory of dynamic obstacle.

1.3 Outline

The outline of the dissertation is as follows. Chapter 2 introduces the background knowledge for the Bernstein polynomial. Chapter 3 presents the trajectory optimization method for multiple UAVs in a dynamic cluttered environment. Chapter 4 demonstrates the theoretical guarantee of the proposed algorithm. For validation, Chapter 5 provides experimental results, and Chapter 6 ends the dissertation with concluding remarks.

2

Bernstein Polynomial

Due to the differential flatness of quadrotor dynamics, it is known that the trajectory of a 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 [47]. However, it is difficult to handle collision avoidance constraints with a standard polynomial basis because a standard polynomial basis does not provide spatial information on polynomials. For this reason, the Bernstein polynomial [48] is utilized to represent the trajectory of quadrotors. Bernstein polynomial is one of the special forms of the Bézier curve and has various useful properties compared to the 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 $\mathbf{p}(\tau) \in \mathbb{R}^3$ is defined as the linear combination of Bernstein basis polynomials:

$$\mathbf{p}(\tau) = \sum_{k=0}^{n} \mathbf{c}_k b_{k,n}(\tau) \tag{2.2}$$

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

2.2 Properties

This section introduces the properties of the Bernstein polynomial. First, the Bernstein polynomial has the convex hull property. A Convex hull is convex envelop of a set of points, which is defined as follows:

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

The convex hull property means that As shown in Fig. 2.1, Bernstein polynomial $\mathbf{p}(\tau)$ is always confined within the convex hull of control points:

$$\mathbf{p}(\tau) \in \text{Conv}(\{\mathbf{c}_0, ..., \mathbf{c}_n\}) \text{ for all } \tau \in [0, 1] \tag{2.4}$$

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

Second, the start and end points of the polynomial are equal to the first and last control points of the polynomial. More precisely, for given n^{th} order Bernstein polynomial $\mathbf{p}(\tau)$ with control points $\mathbf{c}_0, ..., \mathbf{c}_n$, $\mathbf{p}(\tau)$ always start at the first control point $\mathbf{c}_0$ and end at the last control point $\mathbf{c}_n$:

$$\mathbf{p}(0) = \mathbf{c}_0, \mathbf{p}(1) = \mathbf{c}_n \tag{2.5}$$

Using this property, the start and goal points of quadrotors can be assigned by placing the first and last control points in the proper position.

Third, the sum and difference of two Bernstein polynomials are still Bernstein polynomials if two polynomials have the same degree. Assume that two Bernstein polynomial

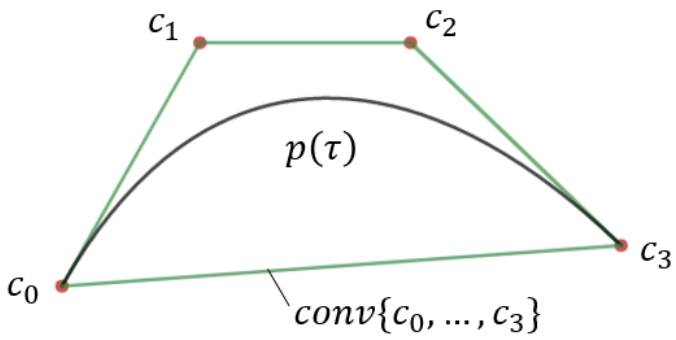


Figure 2.1: Convex hull property of Bernstein polynomial.

$\mathbf{p}_i(\tau), \mathbf{p}_j(\tau)$, have control points $\mathbf{c}_{i,k=0,\dots,n}, \mathbf{c}_{j,k=0,\dots,n}$ respectively. Then the sum or difference of two Bernstein polynomials can be written as follows:

$$\begin{aligned} \mathbf{p}_i(\tau) \pm \mathbf{p}_j(\tau) &= \sum_{k=0}^{n} \mathbf{c}_{i,k} b_{k,n}(\tau) \pm \sum_{k=0}^{n} \mathbf{c}_{j,k} b_{k,n}(\tau) \\ &= \sum_{k=0}^{n} (\mathbf{c}_{i,k} \pm \mathbf{c}_{j,k}) b_{k,n}(\tau) \end{aligned} \tag{2.6}$$

Fourth, a derivative of the Bernstein polynomial can be represented as a Bernstein polynomial. For example, assume that with $\dot{\mathbf{p}}(\tau)$ and $\ddot{\mathbf{p}}(\tau)$ have control points $\mathbf{v}_{0,\cdots,n-1}$ and $\mathbf{a}_{0,\cdots,n-2}$, respectively. Then, control points of derivatives can be derived from control points of the original Bernstein polynomial:

$$\mathbf{v}_k = n(\mathbf{c}_{k+1} - \mathbf{c}_k), \quad \forall k = 0, ..., n-1 \tag{2.7}$$

$$\mathbf{a}_k = n(n-1)(\mathbf{c}_{k+2} - 2\mathbf{c}_{k+1} + \mathbf{c}_k), \quad \forall k = 0, ..., n-2 \tag{2.8}$$

3

Multi-Agent Trajectory Planning

3.1 Problem Statement

Suppose that there are N_a agents with the radius r in an obstacle environment $\mathcal{O}$ with N_o dynamic obstacles. The goal is to generate a safe and dynamically feasible trajectory that allows the agent to reach the goal point.

3.1.1 Notation

This chapter will use the notation in Table 3.1. The calligraphic letter denotes a set, the bold letter indicates a vector, and the italic lowercase letter means a scalar value. The superscript with parenthesis such as $\mathbf{x}^{(h)}$ denotes that the symbol is generated at the replanning step h, and the superscript will be omitted when the symbol is planned at the current replanning step.

Table 3.1: Notation for Chapter 3

Symbol	Definition
$\mathbf{x}^{(h)}$	This superscript indicates that the symbol is planned at the replanning step h. It is omitted if the symbol is from the current replanning step.
$\mathcal{I}_a$, $\mathcal{I}_o$	Set of agents and dynamic obstacle, respectively.
$G = (\mathcal{V}, \mathcal{E})$, d	Grid space ($\mathcal{V}$: grid vertices, $\mathcal{E}$: grid edges), grid size $d > 2\sqrt{2}r$
r_c, $\mathcal{N}_i$	Communication range $r_c > 2d$, connected group that can communicate with the agent i.
M, n	The number of the trajectory segments, degree of the polynomial $n \geq 5$.
$\mathcal{I}_M, \mathcal{I}_n$	$\mathcal{I}_M = \{1, \cdots, M\}$, $\mathcal{I}_n = \{0, \cdots, n\}$.
T_h, Δt	Start time of the replanning step h, replanning period.
$\mathbf{p}_i(t), \hat{\mathbf{p}}_i(t)$	Trajectory, initial trajectory of the agent i.
$\mathbf{c}_{i,m,k}$, $\mathbf{v}_{i,m,k}$, $\mathbf{a}_{i,m,k}$, $\hat{\mathbf{c}}_{i,m,k}$	The k^{th} control point of the m^{th} segment of $\mathbf{p}_i(t)$, $\dot{\mathbf{p}}_i(t)$, $\ddot{\mathbf{p}}_i(t)$, $\hat{\mathbf{p}}_i(t)$, respectively.
$\mathcal{O}$	The space occupied by the static obstacles.
r, r_j	Radius of the agent and dynamic obstacle j.
$\mathbf{z}_i, \mathbf{w}_i, \mathbf{g}_i$	Desired goal, waypoint, subgoal of the agent i.
$\mathcal{S}_m$, $\mathcal{L}_{m,k}^{i,j}$	Safe flight corridor (SFC), Linear safe corridor (LSC) between the agent i and the object j.
$\|\mathbf{x}\|$, $\|\mathbf{x}\|_\infty$, $\|\boldsymbol{\pi}_i\|$, $\mathbf{a} \preceq \mathbf{b}$, $\oplus$, $\text{Conv}(\mathcal{X})$, $[\mathbf{a}, \mathbf{b}]$	Euclidean norm, L-infinity norm, makespan of the discrete path $\boldsymbol{\pi}_i$, element-wise inequality, Minkowski sum, convex hull operator, and line segment between two points $\mathbf{a}$, $\mathbf{b}$.

3.1.2 Assumption

This dissertation supposes the following assumptions:

- (Obstacle) The static obstacle space $\mathcal{O}$ and the dynamical limit of dynamic obstacles are given as prior knowledge. The trajectory of the dynamic obstacle is unknown to the agents, but each agent can observe the current position and velocity of the obstacles.

- (Grid-based planner) All agents share the same grid space $G = (\mathcal{V}, \mathcal{E})$, where the grid size d is larger than $2\sqrt{2}r$. If the agent is on the grid, There is no collision between the agent and static obstacles if the agent is on the grid.

- (Mission) All agents share the same grid space $G = (\mathcal{V}, \mathcal{E})$, where the grid size d is larger than $2\sqrt{2}r$. If the agent is on the grid, There is no collision between the agent and static obstacles if the agent is on the grid.

- (Communication) This work assumes that the agents can establish an ideal mobile ad-hoc network (MANET) to relay messages between them. In other words, the agents i and j can communicate with each other if they satisfy the following:

$$\|\mathbf{p}_i(t) - \mathbf{p}_j(t)\|_\infty \leq r_c \tag{3.1}$$

  where $\mathbf{p}_i(t)$ is the position of the agent i, $\|\cdot\|_\infty$ is the L-infinity norm, and $r_c > 2d$ is the communication range. This work assumes that the transmission time per hop is negligible, and the agents within the communication range can share the information with each other without a communication loss or delay. Fig. 3.1 shows an example. In this example, the blue and red agents are too far apart to communicate directly, but they are both within the communication range of the green agent. In this case, the green agent can help them communicate by passing messages between them like a router. In the same way, the agents that are far apart can communicate with each

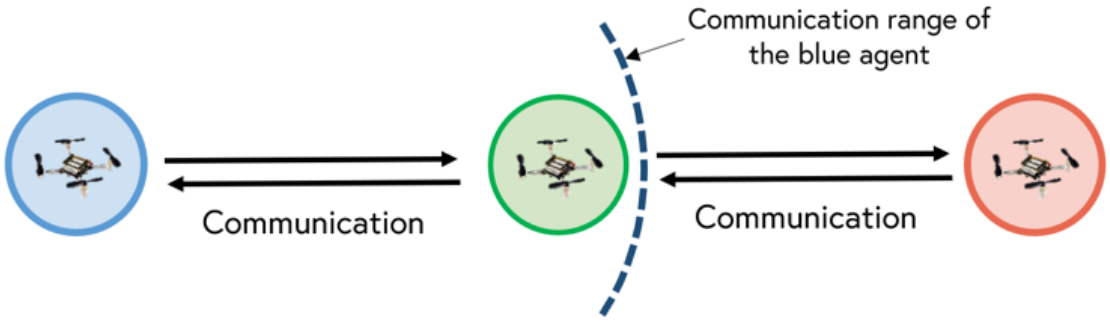


Figure 3.1: Ad-hoc network example.

other if they have one or more router agents to relay messages. A set of agents that can communicate with the agent i through the ad-hoc network will be denoted as a *connected group* $\mathcal{N}_i$.

3.1.3 Trajectory Representation

As discussed in the previous chapter, the trajectory of the quadrotor can be represented as a polynomial with flat outputs (x, y, z, ψ). Therefore, the agent's trajectory is represented to a piecewise Bernstein polynomial [48], thanks to the differential flatness of quadrotor dynamics [10]. The trajectory consists of M segments, and each segment of the trajectory of the agent i is formulated as follows:

$$\mathbf{p}_i(t) = \begin{cases} \sum_{k=0}^{n} \mathbf{c}_{i,1,k} b_{k,n}(t) & t \in [T_h, T_{h+1}] \\ \sum_{k=0}^{n} \mathbf{c}_{i,2,k} b_{k,n}(t) & t \in [T_{h+1}, T_{h+2}] \\ \vdots & \vdots \\ \sum_{k=0}^{n} \mathbf{c}_{i,M,k} b_{k,n}(t) & t \in [T_{h+M-1}, T_{h+M}] \end{cases} \tag{3.2}$$

where h is the current replanning step, $\mathbf{p}_i(t)$ is the trajectory of the agent i, $\mathbf{c}_{i,m,k} \in \mathbb{R}^3$ is the control point, $n \geq 5$ is the degree of the polynomial, $b_{k,n}(t)$ is Bernstein basis polynomial, T_0 is the mission start time, $T_h = T_0 + h\Delta t$, and Δt is the replanning period. Note that

the segment duration is equal to the replanning period Δt. In this work, the agent's yaw angle is fixed as a constant. The decision vector that include all the control points of $\mathbf{p}_i(t)$ will be denoted as follows:

$$\mathbf{c}_i = [\mathbf{c}_{i,1,0}^T, \cdots, \mathbf{c}_{i,M,n}^T]^T \tag{3.3}$$

3.1.4 Collision avoidance

Inter-agent collision avoidance

The collision avoidance constraint between the agents i and j can be represented as follows:

$$\mathbf{p}_i(t) - \mathbf{p}_j(t) \notin \mathcal{C}_{i,j}, \quad \forall t \tag{3.4}$$

where $\mathcal{C}_{i,j}$ is the inter-collision model, which is a compact convex set that satisfies $\mathcal{C}_{j,i} = -\mathcal{C}_{i,j} = \{-\mathbf{x} \mid \mathbf{x} \in \mathcal{C}_{i,j}\}$ to maintain symmetry between agents. This work adopts the ellipsoidal collision model as follows:

$$\mathcal{C}_{i,j} = \{\mathbf{x} \in \mathbb{R}^3 \mid \|D_{i,j}\mathbf{x}\| < 2r\} \tag{3.5}$$

where $\|\cdot\|$ is the Euclidean norm and r is the radius of the agents, $D_{i,j} = \text{diag}([1, 1, 1/\gamma_{i,j}])$ is the scaling matrix, and $\gamma_{i,j} \geq 1$ is the scaling coefficient.

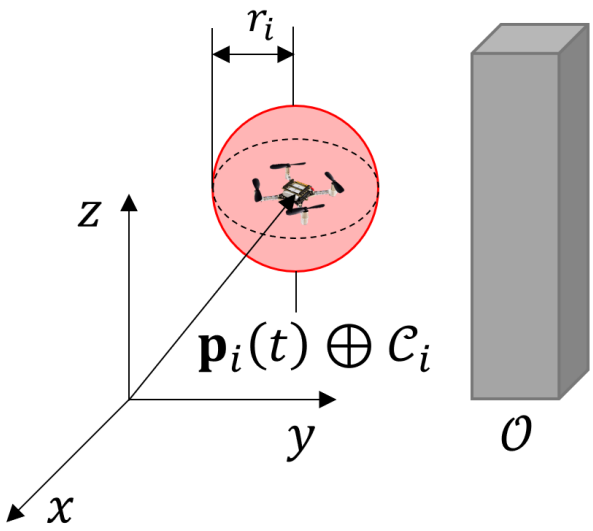
Static obstacle avoidance

The agent i does not collide with static obstacles if the following condition holds:

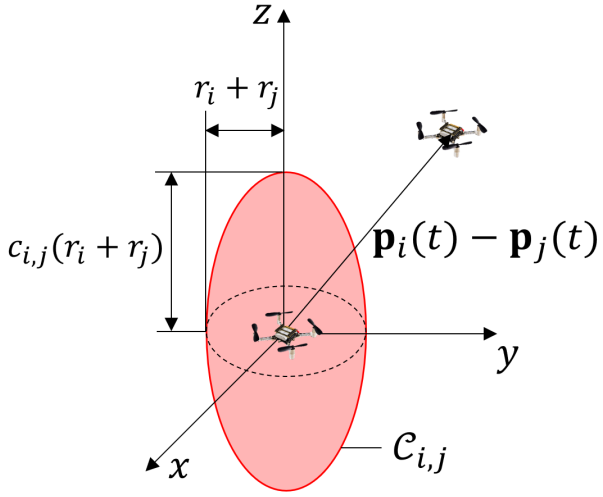
$$(\mathbf{p}_i(t) \oplus \mathcal{C}_i) \cap \mathcal{O} = \emptyset, \forall t \tag{3.6}$$

$$\mathcal{C}_i = \{\mathbf{x} \in \mathbb{R}^3 \mid \|\mathbf{x}\| < r\} \tag{3.7}$$

where $\oplus$ is the Minkowski sum, $\mathcal{O}$ is the space occupied by the obstacles, and $\mathcal{C}_i$ is the obstacle collision model that has a sphere shape as Fig. 3.2a.



(a) Collision model between the agent and static obstacles.



(b) Collision model for dynamic obstacle avoidance.

Figure 3.2: Collision models for collision avoidance.

Dynamic obstacle avoidance

The agent i does not collide with the dynamic obstacle $j \in \mathcal{I}_o$ if the following condition is satisfied:

$$\mathbf{p}_i(t) - \mathbf{p}_j(t) \notin \mathcal{C}_{i,j}, \quad \forall t \tag{3.8}$$

$$\mathcal{C}_{i,j} = \{\mathbf{x} \in \mathbb{R}^3 \mid \|D_{i,j}\mathbf{x}\| \leq r + r_j\} \tag{3.9}$$

where r_j is the radius of the dynamic obstacle j, $\mathcal{C}_{i,j}$ is the collision model between the agent i and dynamic obstacle j that has an ellipsoidal shape as Fig. 3.2b, $D_{i,j} = \text{diag}([1, 1, 1/\gamma_{i,j}])$ is the scaling matrix, and $\gamma_{i,j} \geq 1$ is the scaling coefficient.

3.1.5 Dynamical limit

The dynamical limit of the agent is given as follows:

$$\|\mathbf{v}_i(t)\|_\infty \leq v_{max}, \forall t \tag{3.10}$$

$$\|\mathbf{a}_i(t)\|_\infty \leq a_{max}, \forall t \tag{3.11}$$

where $\mathbf{v}_i(t)$ and $\mathbf{a}_i(t)$ are the velocity and acceleration of the agent i, respectively, and v_{max} and a_{max} are the agent's maximum velocity and acceleration, respectively.

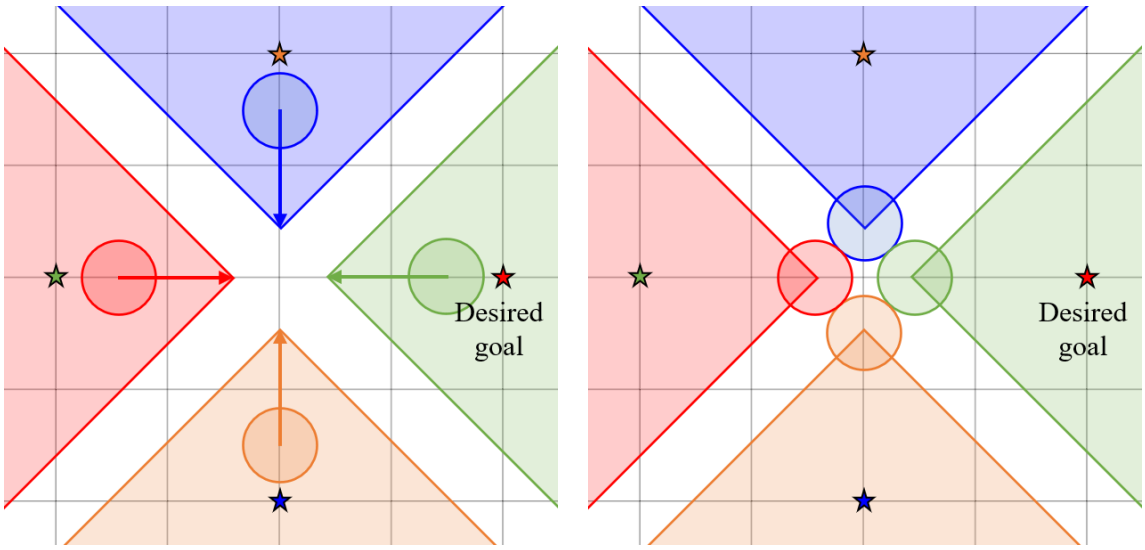
3.2 Overview

3.2.1 Goal Convergence Strategy

The root cause of deadlock in many online multi-agent trajectory planning (MATP) algorithms is that they do not consider collision avoidance constraints when determining the current subgoal. Suppose that all agents try to reach the desired goal directly as shown in Fig. 3.3a. Then, as shown in Fig. 3.3b, the deadlock will occur since the direct path

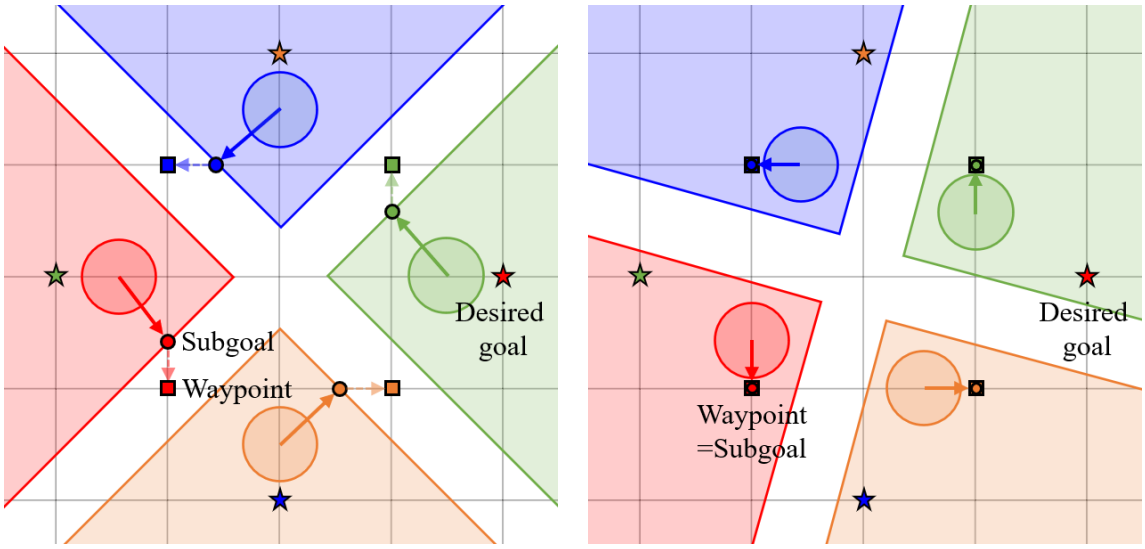
to the goal is blocked by collision constraints. For this reason, many MATP algorithms have used various heuristics for deadlock detection and resolution. However, to predict whether a deadlock occurs, the other agent's current position and desired goal point must be considered. As a result, as the number of agents increases, it becomes more difficult to predict where each agent will converge, which makes deadlock resolution much more challenging.

To solve this problem, the proposed algorithm is designed to guide the agent toward the desired goal through three steps. First, as shown in Fig. 3.4, the proposed algorithm places the subgoal in the feasible region that satisfies the collision constraints. It ensures that each agent reaches its subgoal. Next, the subgoal optimization is conducted to make the subgoal converges to a waypoint, which is on the vertex of grid space G. Finally, a grid-based multi-agent path planning (MAPP) algorithm is used to guide the waypoint to the desired goal. Using this process, the proposed algorithm can prevent deadlock preemptively, as shown in Fig. 3.4b. Furthermore, if the MAPP algorithm guarantees completeness, then the proposed algorithm also guarantees goal convergence. In other words, the theoretical property of grid-based MAPP can be applied to MATP in the continuous space using the proposed approach.



(a) (b)

Figure 3.3: Trajectory planning result when all agents try to reach the desired goal directly.



(a) (b)

Figure 3.4: Trajectory planning result of the proposed algorithm.

3.2.2 Algorithm Overview

Fig. 3.5 and Alg. 1 describe the proposed algorithm. The proposed algorithm consists of the communication phase (lines 3-4) and trajectory generation phase (lines 5-17). During the communication phase, each agent configures an ad-hoc network between the agents within the communication range. After network configuration, a decentralized grid-based MAPP algorithm is executed to determine the waypoint of the agent (line 3, Sec. 3.3). Then, the agent shares the previously planned trajectory and subgoal with the connected group (line 4). In the trajectory generation phase, initial trajectories are generated using the previously planned trajectories and the obstacles' position and velocity. (lines 5-12, Sec. 3.4). Next, the initial trajectories are utilized to construct feasible collision constraints (lines 11 and 13, Sec. 3.5). In this work, the collision constraints are designed to allow the subgoal to converge to the waypoint. After that, the subgoal is determined in the feasible region that satisfies all collision constraints to prevent deadlock (line 14, Sec. 3.6). Finally, the trajectory optimization problem is formulated using the collision constraints and subgoal and solved using the convex solver (lines 15, Sec. 3.7). The above process is repeated until all agents reach the desired goal. The detail of each module will be described in the following sections.

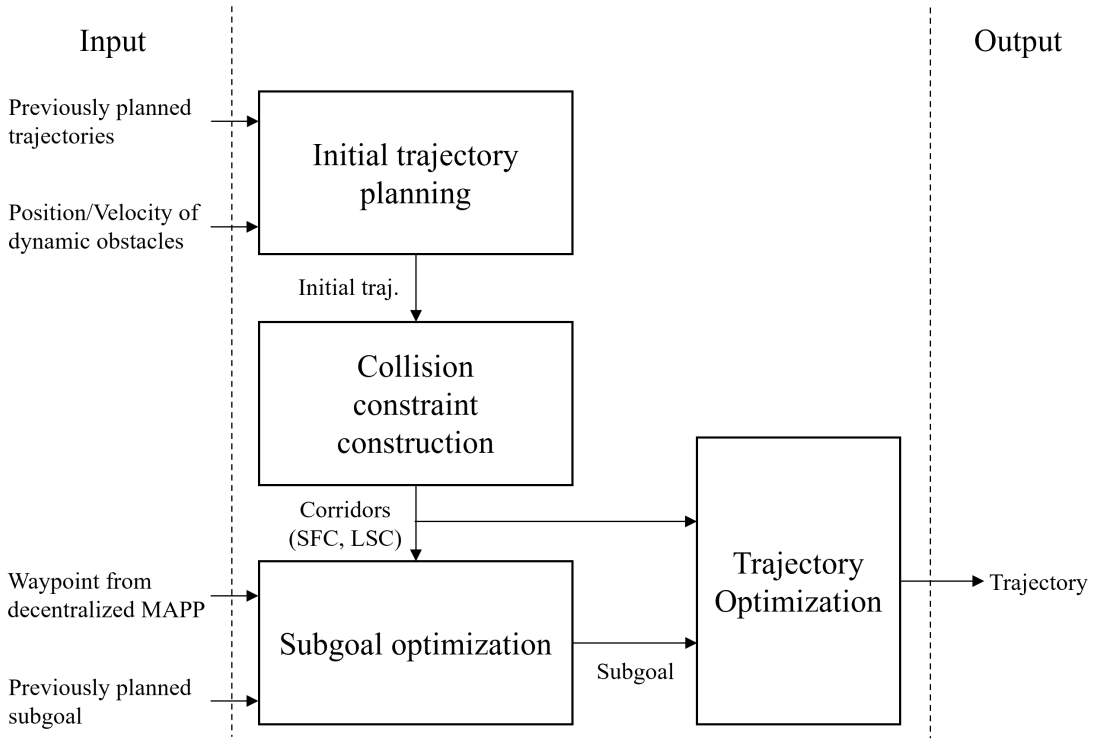


Figure 3.5: Flowchart of the proposed distributed algorithm run by each agent. First, the agent receives other agents' previously planned trajectories through communication and observes obstacles' position and velocity. Then, the agent plans the initial trajectory from the inputs and constructs collision constraints. After that, the subgoal is determined within a feasible region that satisfies the collision constraints. Finally, trajectory optimization is conducted to generate a safe trajectory.

Algorithm 1: Trajectory planning for the agent i

Input: Start point $\mathbf{s}_i$, desired goal $\mathbf{z}_i$, obstacle space $\mathcal{O}$, observed position/velocity of dynamic obstacles $\bar{\mathbf{p}}_{k\in\mathcal{I}_o}$, $\bar{\mathbf{v}}_{k\in\mathcal{I}_o}$

Output: Trajectory of the agent i, $\mathbf{p}_i(t)$

$h \leftarrow 0$;

while *not all agents at desired goal* **do**

`// Communication phase`

$\mathbf{w}_{j\in\mathcal{N}_i} \leftarrow \text{decentralizedMAPP}(\mathbf{p}_i^{(h-1)}, \mathbf{g}_i^{(h-1)}, \mathbf{w}_i^{(h-1)}, \mathbf{z}_i, \bar{\mathbf{p}}_{k\in\mathcal{I}_o}, \bar{\mathbf{v}}_{k\in\mathcal{I}_o})$;

$\mathbf{p}_{j\in\mathcal{N}_i}^{(h-1)}, \mathbf{g}_{j\in\mathcal{N}_i}^{(h-1)} \leftarrow \text{communicate}(\mathbf{p}_i^{(h-1)}, \mathbf{g}_i^{(h-1)})$;

`// Trajectory generation phase`

for $\forall j \in \mathcal{N}_i \cup \mathcal{I}_o$ **do**

if $j \in \mathcal{N}_i$ **then**

$\hat{\mathbf{p}}_j(t) \leftarrow \text{planInitialTraj}(\mathbf{p}_j^{(h-1)})$;

else

$\hat{\mathbf{p}}_j(t) \leftarrow \text{planInitialTraj}(\bar{\mathbf{p}}_j, \bar{\mathbf{v}}_j)$;

end

$\mathcal{L}_{m,k}^{i,j} \leftarrow \text{buildLSC}(\hat{\mathbf{p}}_i(t), \hat{\mathbf{p}}_j(t))$;

end

$\mathcal{S}_m \leftarrow \text{buildSFC}(\hat{\mathbf{p}}_i(t), \mathcal{O})$;

$\mathbf{g}_i \leftarrow \text{subgoalOpt}(\mathbf{g}_i^{(h-1)}, \mathbf{w}_i, \mathcal{S}_m, \mathcal{L}_{m,k}^{i,j})$;

$\mathbf{p}_i(t) \leftarrow \text{trajOpt}(\mathcal{S}_m, \mathcal{L}_{m,k}^{i,j}, \mathbf{g}_i)$;

executeTrajectory($\mathbf{p}_i(t)$);

$h \leftarrow h + 1$;

end

Algorithm 2: decentralizedMAPP

Input: Prev. traj $\mathbf{p}_{j\in\mathcal{N}_i}^{(h-1)}$, prev. subgoals $\mathbf{g}_{j\in\mathcal{N}_i}^{(h-1)}$, prev. waypoints $\mathbf{w}_{j\in\mathcal{N}_i}^{(h-1)}$, desired goals $\mathbf{z}_{j\in\mathcal{N}_i}$, observed position and velocity of dynamic obstacles $\bar{\mathbf{p}}_{k\in\mathcal{I}_o}$, $\bar{\mathbf{v}}_{k\in\mathcal{I}_o}$

Output: Current waypoints $\mathbf{w}_{j\in\mathcal{N}_i}$

`// DOI Detection`

$\mathcal{D}_{j\in\mathcal{N}_i} \leftarrow \text{findDOI}(\mathbf{p}_{j\in\mathcal{N}_i}^{(h-1)}, \mathbf{w}_{j\in\mathcal{N}_i}^{(h-1)}, \bar{\mathbf{p}}_{k\in\mathcal{I}_o}, \bar{\mathbf{v}}_{k\in\mathcal{I}_o})$;

$\mathbf{z}_{j\in\mathcal{N}_i} \leftarrow \text{updateGoal}(\mathbf{z}_{j\in\mathcal{N}_i}, \mathcal{D}_{j\in\mathcal{N}_i})$;

`// Grid-based MAPP`

$\boldsymbol{\pi}_{j\in\mathcal{N}_i} \leftarrow \text{runMAPP}(\mathbf{w}_{j\in\mathcal{N}_i}^{(h-1)}, \mathbf{z}_{j\in\mathcal{N}_i}, \mathcal{D}_{j\in\mathcal{N}_i})$;

$\hat{\boldsymbol{\pi}}_{j\in\mathcal{N}_i} \leftarrow \text{modifyPreviousPath}(\boldsymbol{\pi}_{j\in\mathcal{N}_i}^{(h-1)}, \mathbf{w}_{j\in\mathcal{N}_i}^{(h-1)})$;

if $\mathcal{D}_{j\in\mathcal{N}_i} = \emptyset$, $\mathcal{N}_i = \mathcal{N}_i^{(h-1)}$, $|\hat{\boldsymbol{\pi}}_{j\in\mathcal{N}_i}| \leq |\boldsymbol{\pi}_{j\in\mathcal{N}_i}|$ **then**

  $\boldsymbol{\pi}_{j\in\mathcal{N}_i} \leftarrow \hat{\boldsymbol{\pi}}_{j\in\mathcal{N}_i}$

end

`// Waypoint update`

$\mathcal{Q} \leftarrow \emptyset$;

for $\forall j \in \mathcal{N}_i$ **do**

  if $h = 0$ *or the agent* j *satisfies (3.17), (3.18))* **then**

    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{j\}$;

    $\mathbf{w}_j \leftarrow$ second waypoint of $\boldsymbol{\pi}_j$;

  else

    $\mathbf{w}_j \leftarrow \mathbf{w}_j^{(h-1)}$;

  end

end

`// Conflict resolution`

for $\forall j \in \mathcal{Q}$ **do**

  if $\boldsymbol{w}_j = \boldsymbol{w}_q, \exists q \in \mathcal{N}_i \backslash \{j\}$ **then**

    $\mathbf{w}_j \leftarrow \mathbf{w}_j^{(h-1)}$;

  end

end

return $\boldsymbol{w}_{j\in\mathcal{N}_i}$

3.3 Decentralized Multi-agent Path Planning

The proposed algorithm introduces the decentralized multi-agent path planning (MAPP) to update the waypoint, which provides guidance to the desired goal. Alg. 2 describes the proposed waypoint update method. For every replanning step, each agent configures the ad-hoc network between agents within the communication range, and one agent among the connected group is selected as a local coordinator. The local coordinator collects the previously planned trajectory, subgoals, waypoints, and desired goals of the agents in the connected group. It also receives the observed position and velocity of the dynamic obstacles as input.

3.3.1 Dynamic Obstacles of Interest Detection

For dynamic obstacle avoidance, the obstacle that may collide with the agent within the planning horizon is identified as *dynamic obstacles of interest* (DOI) (line 1). In this work, the obstacle k is considered as one of the DOI of the agent j ($\mathcal{D}_j$) if the agent's previous trajectory passes through the obstacle's reachable region or the agent's previous waypoint is in the obstacle's reachable region:

$$\mathbf{p}_j^{(h-1)}(t) \in \mathcal{R}_{j,k}, \exists t \in [T_h, T_h + M\Delta t] \tag{3.12}$$

$$\mathbf{w}_j^{(h-1)} \in \mathcal{R}_{j,k} \tag{3.13}$$

where $\mathbf{p}_j^{(h-1)}(t)$ is the previous trajectory of the agent j, $\mathbf{w}_j^{(h-1)}$ is the previous waypoint of the agent j, $\mathcal{R}_{j,k}(t)$ is the reachable region of the dynamic obstacle k for the agent j, which is estimated based on the obstacle's dynamical limit:

$$\mathcal{R}_{j,k}(t) = \{\mathbf{x} \in \mathbb{R}^3 \mid \|D_{j,k}(\bar{\mathbf{p}}_k + \bar{\mathbf{v}}_k(t - T_h) - \mathbf{x}))\| < \hat{r}_k(t)\} \tag{3.14}$$

$$\hat{r}_k(t) = r + r_k + \frac{1}{2}a_{k,max}\min(t - T_h, M_e\Delta t)^2\} \tag{3.15}$$

where $\bar{\mathbf{p}}_k$ and $\bar{\mathbf{v}}_k$ are the observed position and velocity of the obstacle k, $D_{j,k}$ is the scaling matrix, $a_{k,max}$ is the maximum acceleration of the obstacle k and $M_e \geq 1$ is the number of the error prediction segments to prevent overly conservative reachable region estimation.

When the agent detects the DOI, the agent moves the goal as far as possible from the obstacle to reduce the risk of collision (line 2). To determine the new goal point, the breadth-first search is used to find candidate grid points where the distance to the agent's previous waypoint is smaller than the distance to the DOI. The farthest point from the DOI among these candidates is then selected as the new goal point for the agent.

3.3.2 Grid-based MAPP

After DOI detection, the local coordinator plans discrete paths using the MAPP algorithm on the grid space G (line 3). The start points of the MAPP are the previous waypoints $\mathbf{w}_{i\in\mathcal{N}_i}^{(h-1)}$, and the goal points are the desired goals updated after DOI detection. If it is the first time to run MAPP, the start points of MAPP are the agent's current position. To consider the downwash between the agents, the z-axis of the grid map is extended by the downwash coefficient. For example, if the grid size in the x and y axis is d, then the grid size in the z-axis direction is $\gamma_{i,j}d$ where $\gamma_{i,j}$ is downwash coefficient.

In this work, Priority Inheritance with Backtracking (PIBT) is adopted [49] for MAPP because it is a fast and scalable algorithm that guarantees *goal reachability*, which ensures that each agent reaches the desired goal. Furthermore, I modified the algorithm to allow the path to circumvent the obstacles since the original PIBT does not consider dynamic obstacles. First, the proposed algorithm assigns a higher priority to the agent that has a shorter distance to the DOI than other agents. Fig. 3.6 shows the example. The red agent has the highest priority since it is the closest agent to the obstacle. Therefore, it can push other agents to keep a safe distance against the obstacle. Second, when generating a grid map for MAPP, I block the path entering the reachable region of the dynamic obstacles from the outside as shown in Fig. 3.7. This directed grid map can prevent the discrete paths from passing through the obstacle's reachable region.

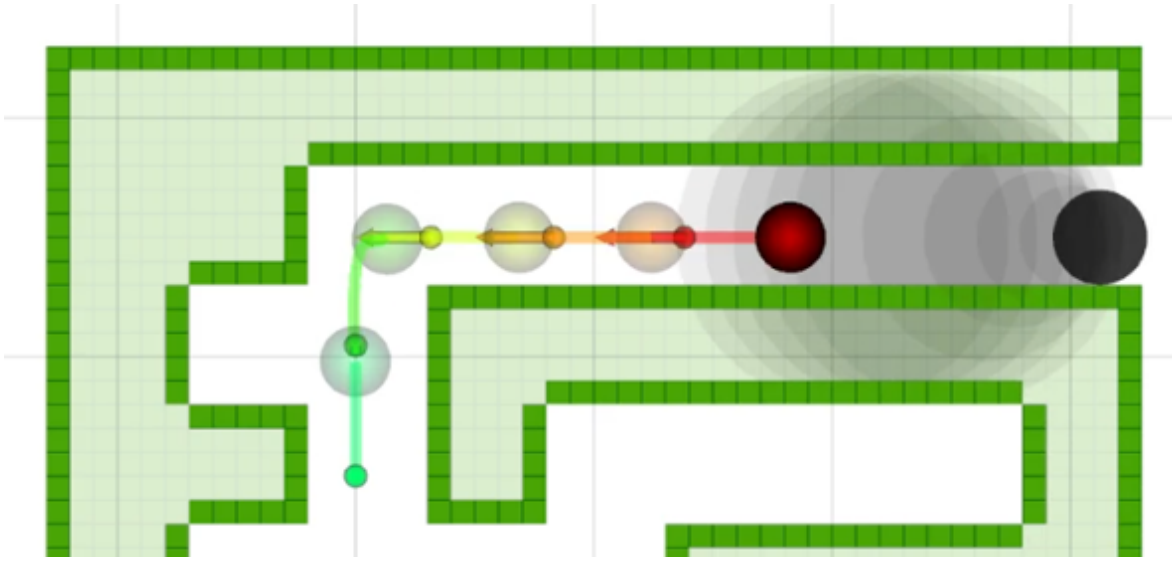


Figure 3.6: Priority assignment for dynamic obstacle avoidance. The black circle is a dynamic obstacle and the other circles are the agents. The gray region is the reachable region of the dynamic obstacle, and the green region is the static obstacle. The red agent has the highest priority because it is the closest agent to the obstacle.

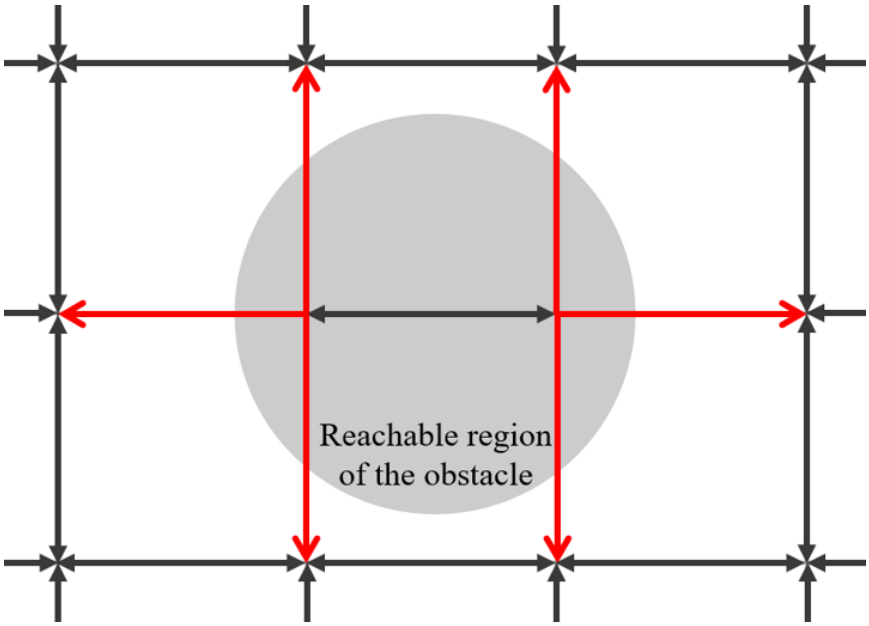


Figure 3.7: The grid map for dynamic obstacle avoidance. The gray region is the reachable region of the dynamic obstacle, the gray arrow denotes that the edge is bidirected, and the red arrow denotes that the edge is not bidirected.

Since PIBT is not an optimal planner, the path can be changed inconsistently for each replanning step, which can cause a livelock. Therefore, the proposed algorithm compares the current path with the previous one and chooses the better one. First, the previous path is modified to match the start point as follows:

$$\hat{\boldsymbol{\pi}}_j = \begin{cases} [\boldsymbol{\pi}_{j,2}^{(h-1)}, \cdots, \boldsymbol{\pi}_{j,L}^{(h-1)}] & \mathbf{w}_j^{(h-1)} = \boldsymbol{\pi}_{j,2}^{(h-1)} \text{for} \\ & \forall j \in \mathcal{N}_i \\ [\mathbf{w}_j^{(h-1)}, \boldsymbol{\pi}_{j,2}^{(h-1)}, \cdots, \boldsymbol{\pi}_{j,L}^{(h-1)}] & \text{else} \end{cases} \tag{3.16}$$

where $\hat{\boldsymbol{\pi}}_j$ is the modified previous path, $\boldsymbol{\pi}_{j,l}^{(h-1)}$ is the l^{th} waypoint of the path $\boldsymbol{\pi}_j^{(h-1)}$, $\boldsymbol{\pi}_j^{(h-1)}$ is the previous path of the agent j, and L is the length of the path. After that, the current path is replaced with the modified one if all agents do not detect the DOI ($\mathcal{D}_j = \emptyset$ for $\forall j \in \mathcal{N}_i$), the member of the connected group is the same as the previous replanning step ($\mathcal{N}_i = \mathcal{N}_i^{(h-1)}$), and the makespan of the current path is not shorter than the makespan of the modified path ($|\hat{\boldsymbol{\pi}}_{j\in\mathcal{N}_i}| \leq |\boldsymbol{\pi}_{j\in\mathcal{N}_i}|$). Note that this process guarantees that the makespan of the discrete path will not increase unless the DOI is detected or the member of the connected group is changed. It is because the modified previous path has a makespan that is equal to or less than the original path.

3.3.3 Waypoint update

After planning the discrete path, the local coordinator updates the agent's waypoint $\mathbf{w}_i$ to the second waypoint of the discrete path (the point one step after the start point) if the following two conditions are satisfied (lines 10-13). First, the subgoal and waypoint at the previous step must be equal (3.17). Second, the distance between the updated waypoint and the endpoints of the previous trajectory's segments must be shorter than $r_c/2$ (3.18):

$$\mathbf{g}_i^{(h-1)} = \mathbf{w}_i^{(h-1)} \tag{3.17}$$

$$\|\mathbf{w}_i - \mathbf{p}_i^{(h-1)}(T_{h+m-2})\|_\infty < \frac{r_c}{2}, \forall m \tag{3.18}$$

where $\mathbf{g}_i$ and $\mathbf{w}_i$ are the subgoal and waypoint, respectively. Otherwise, the previous waypoint is reused as the current waypoint (lines 13-15). Lastly, the proposed algorithm checks whether the waypoints are duplicated in the connected group. If the duplicated waypoints exist, one of them is restored to the previous waypoint. This process is repeated until there is no duplicated waypoint (lines 17-21). Lemma 3.1 shows that the proposed waypoint update rule prevents duplicated waypoints.

Lemma 3.1. *For any pair of the agents $i \in \mathcal{I}_a$ and $j \in \mathcal{I}_a \backslash \{i\}$, $\boldsymbol{w}_i \neq \boldsymbol{w}_j$ holds for every replanning step.*

Proof. If $j \in \mathcal{N}_i$, then the waypoints of the agent i and j cannot be duplicated because the duplicated waypoints are eliminated at the lines 17-21 in Alg. 2. Assume that $j \notin \mathcal{N}_i$ and the agents i and j have the same waypoints $\mathbf{w}_i = \mathbf{w}_j$. Then, $\mathbf{p}_i^{(h-1)}(T_{h+1}) = \mathbf{p}_i(T_h)$ by the initial condition of the trajectory. Hence the following inequality holds due to (3.18):

$$\|\mathbf{w}_i - \mathbf{p}_i(T_h)\|_\infty = \|\mathbf{w}_i - \mathbf{p}_i^{(h-1)}(T_{h+1})\|_\infty < \frac{r_c}{2} \tag{3.19}$$

$$\|\mathbf{w}_i - \mathbf{p}_j(T_h)\|_\infty = \|\mathbf{w}_j - \mathbf{p}_j(T_h)\|_\infty < \frac{r_c}{2} \tag{3.20}$$

Therefore, the distance between two agents is smaller than the communication range by triangle inequality:

$$\|\mathbf{p}_i(T_h) - \mathbf{p}_j(T_h)\|_\infty < r_c \tag{3.21}$$

However, it contradicts the assumption that $j \notin \mathcal{N}_i$. Thus, there are no duplicated waypoints. □

3.4 Initial Trajectory Planning

An initial trajectory is a nominal trajectory to generate feasible collision constraints. In other words, the collision constraints will be constructed that the initial trajectory satisfies.

3.4.1 Agents

The proposed algorithm generates the initial trajectory for the agent using the previously planned trajectories, as shown in Fig. 3.8.

$$\hat{\mathbf{p}}_i(t) = \begin{cases} \mathbf{s}_i & h = 0, t \in [T_0, T_M] \\ \mathbf{p}_i^{(h-1)}(t) & h > 0, t \in [T_h, T_{h+M-1}] \\ \mathbf{p}_i^{(h-1)}(T_{h+M-1}) & h > 0, t \in [T_{h+M-1}, T_{h+M}] \end{cases} \tag{3.22}$$

where $\hat{\mathbf{p}}_i(t)$ is the initial trajectory, and $\mathbf{s}_i$ is the start point of the agent i. The control point of the initial trajectory is represented as follows:

$$\hat{\mathbf{c}}_{i,m,k} = \begin{cases} \mathbf{s}_i & h = 0 \\ \mathbf{c}_{i,m+1,k}^{(h-1)} & h > 0, m < M \\ \mathbf{c}_{i,M,n}^{(h-1)} & h > 0, m = M \end{cases} \tag{3.23}$$

where $\hat{\mathbf{c}}_{i,m,k}$ is the control point of the initial trajectory.

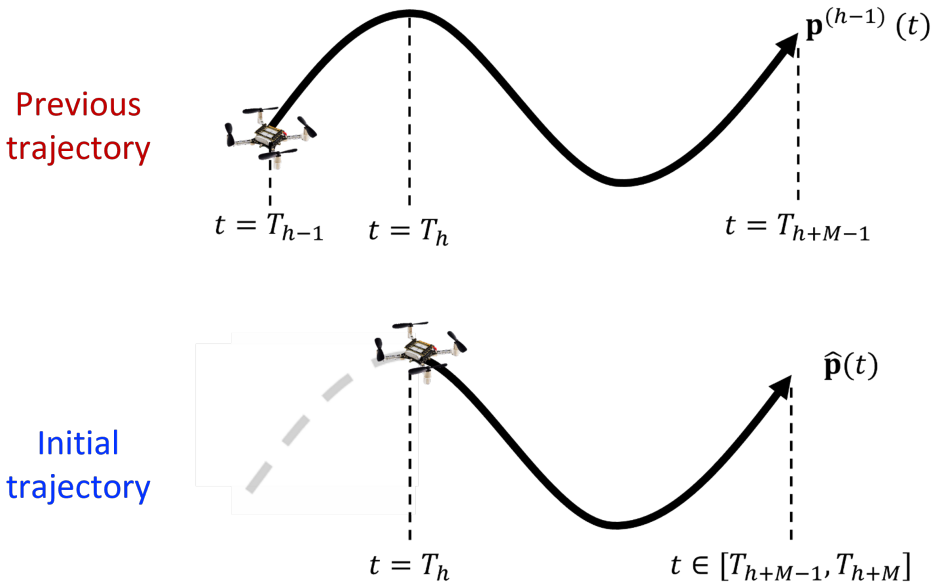


Figure 3.8: Initial trajectory planning for agents.

3.4.2 Dynamic obstacles

For the dynamic obstacle $j \in \mathcal{I}_o$, the predicted obstacle's trajectory is used as the initial trajectory, i.e.:

$$\hat{\mathbf{p}}_j(t) = \bar{\mathbf{p}}_j + \bar{\mathbf{v}}_j(t - T_h) \tag{3.24}$$

where $\hat{\mathbf{p}}_j(t)$ is the initial trajectory of the obstacle j, and $\bar{\mathbf{p}}_j$ and $\bar{\mathbf{v}}_j$ are the observed position and velocity of the obstacle, respectively.

3.5 Collision Constraints Construction

The collision constraints should satisfy the following conditions to achieve collision avoidance and goal convergence. First, they should ensure maximum safety even in environments with dynamic obstacles. Second, they must ensure the feasibility of the trajectory optimization problem until all agents reach their desired goal. Third, they must not block the agent while it converges to the subgoal. This section presents a collision constraint construction method that satisfies the above conditions.

3.5.1 Static obstacle avoidance

A safe flight corridor (SFC) [50] is utilized to prevent collision with static obstacles. In general, SFC is defined as a convex set that prevents the agent from a collision with static obstacles, i.e.:

$$(\mathcal{S} \oplus \mathcal{C}_i) \cap \mathcal{O} \neq \emptyset \tag{3.25}$$

where $\mathcal{S}$ is a SFC, and $\mathcal{C}_i$ is an obstacle collision model. Assume that the control points of the agent i are confined in the corresponding SFC:

$$\mathbf{c}_{i,m,k} \in \mathcal{S}_m, \forall m, k \tag{3.26}$$

where $\mathcal{S}_m$ is a SFC for the m^{th} trajectory segment. Then, the agent i does not collide with static obstacles due to the convex hull property (2.4).

The SFC is constructed as follows:

$$\mathcal{S}_m = \begin{cases} \mathcal{S}(\{\mathbf{s}_i, \mathbf{w}_i\}) & h = 0 \\ \mathcal{S}_{m+2}^{(h-1)} & h > 0, m < M, (3.28) \\ \mathcal{S}_{m+1}^{(h-1)} & h > 0, m < M, \text{else} \\ \mathcal{S}(\{\hat{\mathbf{c}}_{i,M,n}, \mathbf{g}_i^{(h-1)}, \mathbf{w}_i\})) & h > 0, m = M, (3.29) \\ \mathcal{S}(\{\hat{\mathbf{c}}_{i,M,n}, \mathbf{g}_i^{(h-1)}\}) & h > 0, m = M, \text{else} \end{cases} \tag{3.27}$$

$$\text{Conv}(\{\hat{\mathbf{c}}_{i,m,0}, \cdots, \hat{\mathbf{c}}_{i,m,n}\}) \subset \mathcal{S}_{m+2}^{(h-1)}, \quad m < M - 1 \tag{3.28}$$

$$(\text{Conv}(\{\hat{\mathbf{c}}_{i,M,n}, \mathbf{g}_i^{(h-1)}, \mathbf{w}_i\}) \oplus \mathcal{C}_i) \cap \mathcal{O} = \emptyset \tag{3.29}$$

where $\mathcal{S}_m$ is the SFC for m^{th} trajectory segment, $\mathcal{S}(\mathcal{P})$ is a convex polyhedron that includes the point set $\mathcal{P}$ and satisfies $(\mathcal{S}(\mathcal{P}) \oplus \mathcal{C}_i) \cap \mathcal{O} = \emptyset$, and $\text{Conv}(\cdot)$ is the convex hull operator that returns a convex hull of the input set. In this work, $\mathcal{S}(\mathcal{P})$ is constructed using the axis-search method [51]. Alg. 3 describes the axis-search method. First, the SFC is initialized

Algorithm 3: buildSFC

Input: point set $\mathcal{P}$, obstacle space $\mathcal{O}$
Output: safe flight corridor $\mathcal{S}(\mathcal{P})$
`// Initialization`
1 $\mathcal{S}_0 \leftarrow$ axis-aligned bounding box that contains $\mathcal{P}$;
2 $\mathcal{S}(\mathcal{P}) \leftarrow \mathcal{S}_0$;
3 $\mathcal{M} \leftarrow \{\pm x, \pm y, \pm z\}$;
`// Inflate the SFC along the axis direction`
4 **while** $\mathcal{M}$ *is not empty* **do**
5 **for** μ *in* $\mathcal{M}$ **do**
6 **if** $\mathcal{S}(\mathcal{P})$ *can expand to direction* μ **then**
7 expand $\mathcal{S}(\mathcal{P})$ to direction μ;
8 **else**
9 $\mathcal{M} \leftarrow \mathcal{M} \backslash \mu$;
10 **end**
11 **end**
12 **end**

using an input point set (line 1). After that, for all directions, the algorithm verifies whether the SFC is expandable (lines 5). If it is expandable, the SFC is expanded by a pre-specified length (line 6). This algorithm guarantees to return a convex polyhedron that satisfies the definition of SFC.

Compared to [52], $\mathcal{S}_{m+2}^{(h-1)}$ is used instead of $\mathcal{S}_{m+1}^{(h-1)}$ if (3.28) is satisfied. It allows the agent converges faster to the subgoal by discarding the inefficient SFC. In addition, it should be noted that the SFC for the last trajectory segment always includes the line segment between the points $\hat{\mathbf{c}}_{i,M,n}$ and $\mathbf{g}_i^{(h-1)}$. Therefore, the SFC does not block the agent while it converges to the subgoal. Lemma 3.2 shows that the proposed SFC always exists and guarantees static obstacle avoidance for every replanning step.

Lemma 3.2. *(Existence of feasible SFC) Assume that* $\boldsymbol{c}_{i,m,k}^{(h-1)} \in \mathcal{S}_m^{(h-1)}$ *for* $\forall m, k$ *at the replanning step* $h > 0$*. Then, a non-empty convex set* $\mathcal{S}_m$ *that satisfies (3.27) and* $(\mathcal{S}_m \oplus \mathcal{C}_i) \cap \mathcal{O} = \emptyset$ *always exists for* $\forall h, m$*.*

Proof. If $h = 0$, $\mathcal{S}_m^{(0)} = [\mathbf{s}_i, \mathbf{w}_i^{(0)}]$ satisfies (3.27) and $(\mathcal{S}_m \oplus \mathcal{C}_i) \cap \mathcal{O} = \emptyset$ by the assumption that collision does not occur when the agent is on the grid.

Assume that $\mathcal{S}_m^{(h-1)}$ exists and satisfies $(\mathcal{S}_m^{(h-1)} \oplus \mathcal{C}_i) \cap \mathcal{O} = \emptyset$ for $\forall m$. If $m < M$, then $\mathcal{S}_m$ in (3.27) satisfies $(\mathcal{S}_m \oplus \mathcal{C}_i) \cap \mathcal{O} = \emptyset$ by the assumption. If $m = M$ and (3.29) is satisfied, then $\mathcal{S}_M = \text{Conv}(\{\hat{\mathbf{c}}_{i,M,n}, \mathbf{g}_i^{(h-1)}, \mathbf{w}_i\})$ satisfies (3.27) and $(\mathcal{S}_M \oplus \mathcal{C}_i) \cap \mathcal{O} = \emptyset$ by (3.29). If $m = M$ and (3.29) is not satisfied, the SFC can be constructed as $\mathcal{S}_M = \text{Conv}(\{\hat{\mathbf{c}}_{i,M,n}, \mathbf{g}_i^{(h-1)}\})$. It fulfills (3.27) and $\mathcal{S}_m = \text{Conv}(\{\hat{\mathbf{c}}_{i,M,n}, \mathbf{g}_i^{(h-1)}\}) = \text{Conv}(\{\mathbf{c}_{i,M,n}^{(h-1)}, \mathbf{g}_i^{(h-1)}\}) \subset \mathcal{S}_M^{(h-1)}$, which implies $(\mathcal{S}_m \oplus \mathcal{C}_i) \cap \mathcal{O} = \emptyset$ by the assumption. Thus, there exists $\mathcal{S}_m$ that satisfies (3.27) and $(\mathcal{S}_m \oplus \mathcal{C}_i) \cap \mathcal{O} = \emptyset$ for every replanning step by mathematical induction. □

3.5.2 Inter-agent collision avoidance

A linear safe corridor (LSC) [1] is utilized for inter-agent collision avoidance. LSC between the agents i and $j \in \mathcal{I}_a$ is defined as a linear constraint that satisfies the following conditions:

$$\mathcal{L}_{m,k}^{i,j} = \{\mathbf{x} \in \mathbb{R}^3 \mid (\mathbf{x} - \hat{\mathbf{c}}_{j,m,k}) \cdot \mathbf{n}_m^{i,j} - d_{m,k}^{i,j} \geq 0\} \tag{3.30}$$

$$\mathbf{n}_m^{i,j} = -\mathbf{n}_m^{j,i} \tag{3.31}$$

$$d_{m,k}^{i,j} = \frac{1}{2}(\max\langle \mathcal{C}_{i,j}, \mathbf{n}_m^{i,j} \rangle + (\hat{\mathbf{c}}_{i,m,k} - \hat{\mathbf{c}}_{j,m,k}) \cdot \mathbf{n}_m^{i,j}) \tag{3.32}$$

where $d_{m,k}^{i,j}$ is the safety margin and $\mathbf{n}_m^{i,j} \in \mathbb{R}^3$ is the normal vector of the plane that separates the inter-agent collision model $\mathcal{C}_{i,j}$ and the convex hull $\hat{\mathcal{H}}_{i,j,m}$, $\max\langle \mathcal{C}_{i,j}, \mathbf{n}_m^{i,j} \rangle = \max_{\mathbf{x} \in \mathcal{C}_{i,j}} \mathbf{x} \cdot \mathbf{n}_m^{i,j}$. Let $\mathcal{H}_{i,j,m}$ be the convex hull of the control points of relative trajectory and $\hat{\mathcal{H}}_{i,j,m}$ be the convex hull of the control points of relative initial trajectory, i.e. $\mathcal{H}_{i,j,m} = \text{Conv}(\{\mathbf{c}_{i,m,k} - \hat{\mathbf{c}}_{j,m,k} \mid k = 0, \cdots, n\})$, $\hat{\mathcal{H}}_{i,j,m} = \text{Conv}(\{\hat{\mathbf{c}}_{i,m,k} - \hat{\mathbf{c}}_{j,m,k} \mid k = 0, \cdots, n\})$. Lemmas 3.4 and 3.5 present the main properties of LSC.

Lemma 3.3. *If the trajectories of all agents satisfy $\mathcal{H}_{i,j,m} \cap \mathcal{C}_{i,j} = \emptyset, \forall j \in \mathcal{I}_a \backslash i, m = 1, \cdots, M$, then there is no collision between agents.*

Proof. For any pair of the agents i and j, the m^{th} segment of the relative trajectory between

two agents is a Bernstein polynomial:

$$\mathbf{p}_i(t) - \mathbf{p}_j(t) = \sum_{k=0}^{n} (\mathbf{c}_{i,m,k} - \mathbf{c}_{j,m,k}) b_{k,n}(t) \tag{3.33}$$

Due to the convex hull property of the Bernstein polynomial, the following condition holds for $\forall m, t \in [T_{h+m-1}, T_{h+m}]$.

$$\mathbf{p}_i(t) - \mathbf{p}_j(t) \in \mathcal{H}_{i,j,m} \tag{3.34}$$

$$\mathbf{p}_i(t) - \mathbf{p}_j(t) \notin \mathcal{C}_{i,j} \tag{3.35}$$

Thus, there is no collision between agents because they satisfy the collision constraint (3.4) for any segment m. □

Lemma 3.4. *(Safety of LSC) If $\boldsymbol{c}_{i,m,k} \in \mathcal{L}^{i,j}_{m,k}$, $\boldsymbol{c}_{j,m,k} \in \mathcal{L}^{j,i}_{m,k}$ for $\forall m, k$ then $\mathcal{H}_{i,j,m} \cap \mathcal{C}_{i,j} = \emptyset$ which implies that the agent i does not collide with the agent j.*

Proof. Since $\mathbf{c}_{i,m,k} \in \mathcal{L}^{i,j}_{m,k}$, $\mathbf{c}_{j,m,k} \in \mathcal{L}^{j,i}_{m,k}$, the following inequality can be derived using (3.30) for each agent i and j:

$$(\mathbf{c}_{i,m,k} - \hat{\mathbf{c}}_{j,m,k}) \cdot \mathbf{n}^{i,j}_m + (\mathbf{c}_{j,m,k} - \hat{\mathbf{c}}_{i,m,k}) \cdot \mathbf{n}^{j,i}_m - (d^{i,j}_{m,k} + d^{j,i}_{m,k}) > 0 \tag{3.36}$$

This can be simplified as follows:

$$(\mathbf{c}_{i,m,k} - \mathbf{c}_{j,m,k}) \cdot \mathbf{n}^{i,j}_m + (\hat{\mathbf{c}}_{i,m,k} - \hat{\mathbf{c}}_{j,m,k}) \cdot \mathbf{n}^{i,j}_m - (d^{i,j}_{m,k} + d^{j,i}_{m,k}) > 0 \tag{3.37}$$

$$(\mathbf{c}_{i,m,k} - \mathbf{c}_{j,m,k}) \cdot \mathbf{n}^{i,j}_m > \max\langle \mathcal{C}_{i,j}, \mathbf{n}^{i,j}_m \rangle \tag{3.38}$$

The above inequality satisfies for $\forall k$, thus for any $\lambda_k \geq 0$ s.t. $\sum_{k=0}^{n} \lambda_k = 1$:

$$\sum_{k=0}^{n} \lambda_k (\mathbf{c}_{i,m,k} - \mathbf{c}_{j,m,k}) \cdot \mathbf{n}^{i,j}_m > \max\langle \mathcal{C}_{i,j}, \mathbf{n}^{i,j}_m \rangle \tag{3.39}$$

$$\mathcal{H}_{i,j,m} \cap \mathcal{C}_{i,j} = \emptyset \tag{3.40}$$

Thus, there is no collision between the agents by Lemma 3.3. □

Lemma 3.5. *(Existence of feasible LSC) If $\hat{\mathcal{H}}_{i,j,m} \cap \mathcal{C}_{i,j} = \emptyset$ for $\forall i, j \in \mathcal{I}_a, m, k$, then there exists $\mathcal{L}^{i,j}_{m,k}$ that satisfies the definition of LSC and $\hat{\mathbf{c}}_{i,m,k} \in \mathcal{L}^{i,j}_{m,k}$ for $\forall i, j \in \mathcal{I}_a, m, k$.*

Proof. $\hat{\mathcal{H}}_{i,j,m}$ and $\mathcal{C}_{i,j}$ are disjoint compact convex sets. Thus, by the hyperplane separation theorem [53], there exists $\mathbf{n}_s$ such that:

$$\min\langle \hat{\mathcal{H}}_{i,j,m}, \mathbf{n}_s \rangle > \max\langle \mathcal{C}_{i,j}, \mathbf{n}_s \rangle \tag{3.41}$$

where $\min\langle \hat{\mathcal{H}}_{i,j,m}, \mathbf{n}_s \rangle = \min_{\mathbf{x} \in \hat{\mathcal{H}}_{i,j,m}} \mathbf{x} \cdot \mathbf{n}_s$. Suppose that the normal vector and safety margin of $\mathcal{L}^{i,j}_{m,k}$ is given as follows:

$$\mathbf{n}^{i,j}_m = -\mathbf{n}^{j,i}_m = \mathbf{n}_s \tag{3.42}$$

$$d^{i,j}_{m,k} = d^{j,i}_{m,k} = \frac{1}{2}(\max\langle \mathcal{C}_{i,j}, \mathbf{n}^{i,j}_m \rangle + (\hat{\mathbf{c}}_{i,m,k} - \hat{\mathbf{c}}_{j,m,k}) \cdot \mathbf{n}^{i,j}_m) \tag{3.43}$$

They fulfill the definition of LSC, and they satisfy $(\hat{\mathbf{c}}_{i,m,k} - \hat{\mathbf{c}}_{j,m,k}) \cdot \mathbf{n}^{i,j}_m - d^{i,j}_{m,k} = \frac{1}{2}((\hat{\mathbf{c}}_{i,m,k} - \hat{\mathbf{c}}_{j,m,k}) \cdot \mathbf{n}^{i,j}_m - \max\langle \mathcal{C}_{i,j}, \mathbf{n}^{i,j}_m \rangle) > 0$ due to (3.41). It indicates that $\hat{\mathbf{c}}_{i,m,k} \in \mathcal{L}^{i,j}_{m,k}$ for $\forall i, j \in \mathcal{I}_a, m, k$. □

In this work, the LSC between the agents i and $j \in \mathcal{I}_a$ is generated as follows if it is not the LSC for the last trajectory segment ($m < M$):

$$\mathcal{L}^{i,j}_{m,k} = \{\mathbf{x} \in \mathbb{R}^3 \mid (\mathbf{x} - \hat{\mathbf{c}}_{j,m,k}) \cdot \mathbf{n}^{i,j}_m - d^{i,j}_{m,k} \geq 0\} \tag{3.44a}$$

$$d^{i,j}_{m,k} = r + \frac{1}{2}(\hat{\mathbf{c}}_{i,m,k} - \hat{\mathbf{c}}_{j,m,k}) \cdot \mathbf{n}^{i,j}_m \tag{3.44b}$$

where $\mathbf{n}^{i,j}_m \in \mathbb{R}^3$ is the normalized vector that points toward the closest point on the convex hull $\hat{\mathcal{H}}_{i,j,m}$ from the origin, and $\hat{\mathcal{H}}_{i,j,m} = \text{Conv}(\{\hat{\mathbf{c}}_{i,m,k} - \hat{\mathbf{c}}_{j,m,k} \mid k \in \mathcal{I}_n\}$. Fig. 3.9 describes the LSC construction process. First, the coordinate transformation is performed to convert the inter-collision model $\mathcal{C}_{i,j}$ to the sphere as shown in the middle figure of Fig. 3.9. Next, the Gilbert–Johnson–Keerthi (GJK) algorithm is conducted [54] to find the closest points between the convex hull $\hat{\mathcal{H}}_{i,j,m}$ and the collision model. The normal vector of the LSC is

the normal vector from the origin to the closest point on the convex hull, and the safety margin can be computed from the (3.44) using the normal vector. Finally, the desired LSC is generated by reversing the coordinate transform, as shown in the green shaded area in the right figure of Fig. 3.9.

The LSC for the last trajectory segment is constructed as follows so that the agent can converge to the subgoal:

$$\mathcal{L}_{M,k}^{i,j} = \{\mathbf{x} \in \mathbb{R}^3 \mid (\mathbf{x} - \mathbf{p}_{cls}^{j,i}) \cdot \mathbf{n}_M^{i,j} - d_{M,k}^{i,j} \geq 0\} \tag{3.45a}$$

$$\mathbf{n}_M^{i,j} = \frac{\mathbf{p}_{cls}^{i,j} - \mathbf{p}_{cls}^{j,i}}{\|\mathbf{p}_{cls}^{i,j} - \mathbf{p}_{cls}^{j,i}\|} \tag{3.45b}$$

$$d_{M,k}^{i,j} = r + \frac{1}{2}\|\mathbf{p}_{cls}^{i,j} - \mathbf{p}_{cls}^{j,i}\| \tag{3.45c}$$

where $\mathbf{p}_{cls}^{i,j} \in [\hat{\mathbf{c}}_{i,M,n}, \mathbf{g}_i^{(h-1)}]$ and $\mathbf{p}_{cls}^{j,i} \in [\hat{\mathbf{c}}_{j,M,n}, \mathbf{g}_j^{(h-1)}]$ are the closest points between $[\hat{\mathbf{c}}_{i,M,n}, \mathbf{g}_i^{(h-1)}]$ and $[\hat{\mathbf{c}}_{j,M,n}, \mathbf{g}_j^{(h-1)}]$, respectively, and $[a, b]$ is the line segment between two points, a and b. Similar to SFC, the LSC for the last trajectory segment also contains the line segment $[\hat{\mathbf{c}}_{i,M,n}, \mathbf{g}_i^{(h-1)}]$. Therefore, the collision constraints for the last trajectory segment always include the line segment $[\hat{\mathbf{c}}_{i,M,n}, \mathbf{g}_i^{(h-1)}]$ as shown in Fig. 3.10. As a result, each agent can secure the free space to proceed to the subgoal $\mathbf{g}_i^{(h-1)}$, and the subgoal will converge to the waypoint $\mathbf{w}_i$.

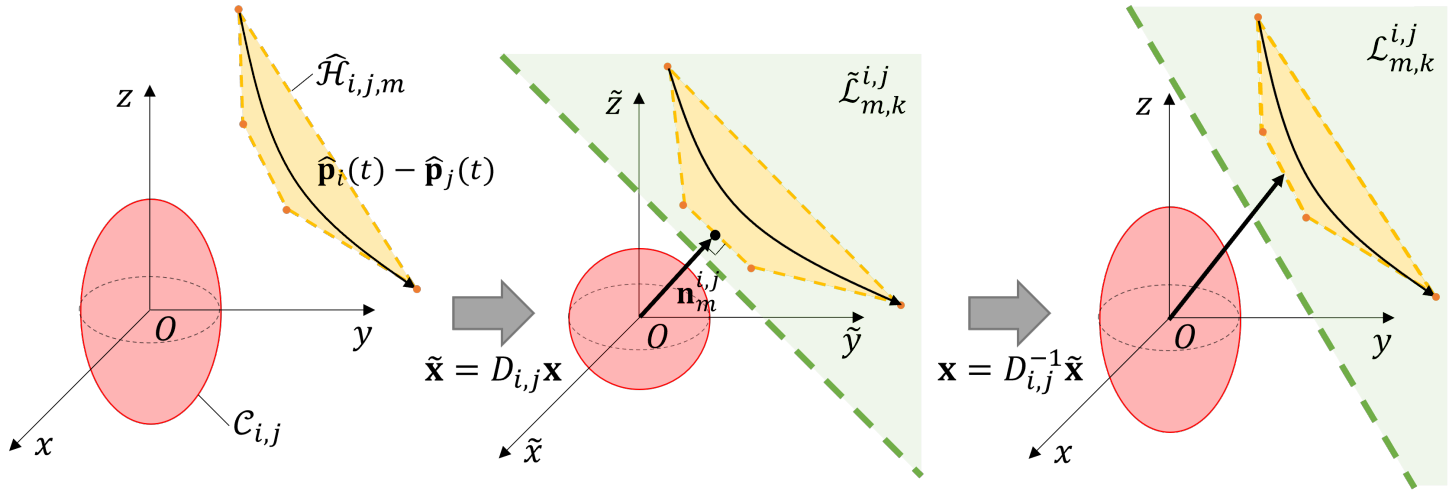


Figure 3.9: LSC for inter-agent collision avoidance. The red ellipsoid is the collision model between two agents, and the yellow-shaded region is the convex hull that consists of the control points of the relative initial trajectory. The green-shaded region is the LSC between two agents. The symbol with a tilde denotes the coordinate-transformed value.

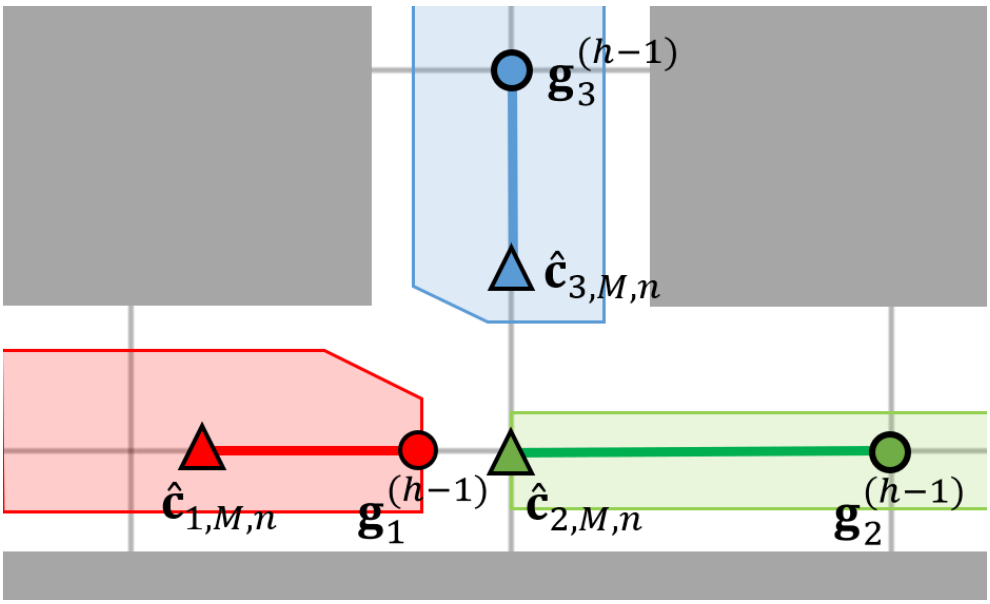


Figure 3.10: Collision constraints for the last trajectory segment. The triangles are the final points of the initial trajectories, and the circles are the previously planned subgoals. The gray box is the static obstacle, and the color-shaded region is the feasible region that satisfies the collision constraints. The collision constraint for the last segment always includes the line segment between the final point and the subgoal, which is depicted as the thick line.

Lemma 3.6 presents that the proposed LSC guarantees inter-collision avoidance.

Lemma 3.6. *(Safety of LSC) If* $\boldsymbol{c}_{i,m,k} \in \mathcal{L}_{m,k}^{i,j}$, $\boldsymbol{c}_{j,m,k} \in \mathcal{L}_{m,k}^{j,i}$ *for* $\forall m, k$ *then the agents* i *and* j *do not collide with each other.*

Proof. If $m < M$, then the LSC guarantees collision avoidance due to Lemma 3.4. If $m = M$:

$$(\mathbf{c}_{i,m,k} - \mathbf{p}_{cls}^{j,i}) \cdot \mathbf{n}_M^{i,j} + (\mathbf{c}_{j,m,k} - \mathbf{p}_{cls}^{i,j}) \cdot \mathbf{n}_M^{j,i} - (d_{M,k}^{i,j} + d_{M,k}^{j,i}) > 0 \tag{3.46}$$

This can be simplified as follows:

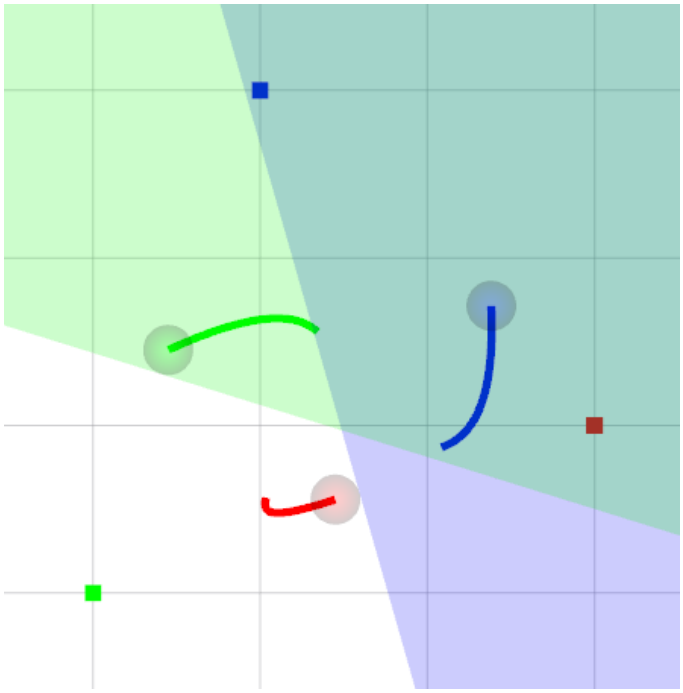
$$(\mathbf{c}_{i,m,k} - \mathbf{c}_{j,m,k}) \cdot \mathbf{n}_M^{i,j} + (\mathbf{p}_{cls}^{i,j} - \mathbf{p}_{cls}^{j,i}) \cdot \mathbf{n}_M^{i,j} - (d_{M,k}^{i,j} + d_{M,k}^{j,i}) > 0 \tag{3.47}$$

$$(\mathbf{c}_{i,m,k} - \mathbf{c}_{j,m,k}) \cdot \mathbf{n}_M^{i,j} + \|\mathbf{p}_{cls}^{i,j} - \mathbf{p}_{cls}^{j,i}\| - 2r + \|\mathbf{p}_{cls}^{i,j} - \mathbf{p}_{cls}^{j,i}\| > 0 \tag{3.48}$$

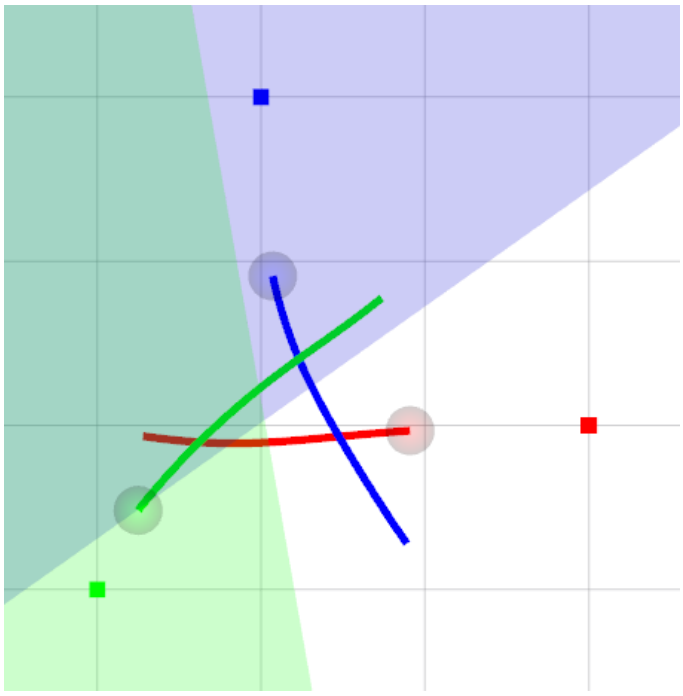
$$(\mathbf{c}_{i,m,k} - \mathbf{c}_{j,m,k}) \cdot \mathbf{n}_M^{i,j} > 2r \tag{3.49}$$

The above inequality holds for $\forall m$, thus the agents do not cause a collision by Lemma 3.3 In conclusion, there is no collision between the agents i and j. □

Fig. 3.11 shows the difference between LSC and BVC [2]. Since the BVC is generated using the agent's current position only, the desired trajectories from the BVC remain within each static cell, which leads to a conservative maneuver as shown in Fig. 3.11a. On the other hand, the LSC can utilize the full trajectory at the previous step. Thus, the agent can show more aggressive maneuvers while ensuring collision avoidance, as depicted in Fig. 3.11b.



(a) BVC [2]



(b) LSC

Figure 3.11: Trajectory planning comparison between BVC [2] and LSC. The colored lines, small squares, and circles are desired trajectories, goal points, and desired position at the end of the planning horizon (i.e. $\mathbf{p}_i(T_{h+M})$) respectively. The color-shaded regions denote the collision constraints for the red agent at the end of the planning horizon.

3.5.3 Dynamic obstacle avoidance

Since the exact trajectory of the dynamic obstacle is unknown, the reachable region of the obstacles should be considered when generating the constraints. To solve this problem, the proposed algorithm reformulates the predicted obstacle's trajectory as piecewise Bernstein polynomials:

$$\hat{\mathbf{p}}_j(t) = \begin{cases} \sum_{k=0}^{n} \hat{\mathbf{c}}_{j,1,k} b_{k,n}(t) & t \in [T_h, T_{h+1}] \\ \vdots & \vdots \\ \sum_{k=0}^{n} \hat{\mathbf{c}}_{j,M,k} b_{k,n}(t) & t \in [T_{h+M-1}, T_{h+M}] \end{cases} \tag{3.50}$$

where $\hat{\mathbf{c}}_{j,m,k}$ is the control point of the predicted trajectory. Then, the reachable area of the object is estimated through the dynamical limit of the object. As depicted in Fig. 3.12, the reachable area can be expressed using the predicted trajectory and the error bound $\mathcal{B}_j(t) \subset \mathbb{R}^3$ between the predicted trajectory and the actual trajectory of the object:

$$\mathbf{p}_j(t) \in \hat{\mathbf{p}}_j(t) \oplus \mathcal{B}_j(t) \tag{3.51}$$

where $\hat{\mathbf{p}}_j(t)$ is the predicted trajectory of the j^{th} object and the $\oplus$ is the Minkowski sum. Finally, as illustrated in Fig. 3.13, the sufficient condition of dynamic obstacle avoidance can be derived as follows:

$$(\mathbf{p}_i(t) - \hat{\mathbf{p}}_j(t)) \cap (\mathcal{B}_j(t) \oplus \mathcal{C}_{i,j}) = \emptyset, \quad t \in [T_h, T_{h+M}] \tag{3.52}$$

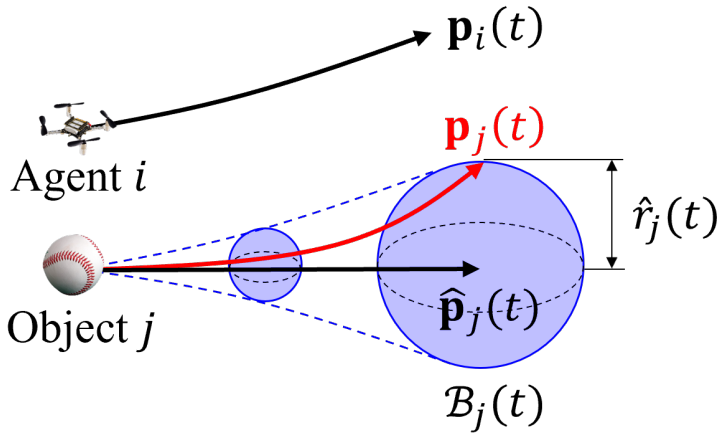


Figure 3.12: Trajectory prediction model with error bound (blue ellipsoid).

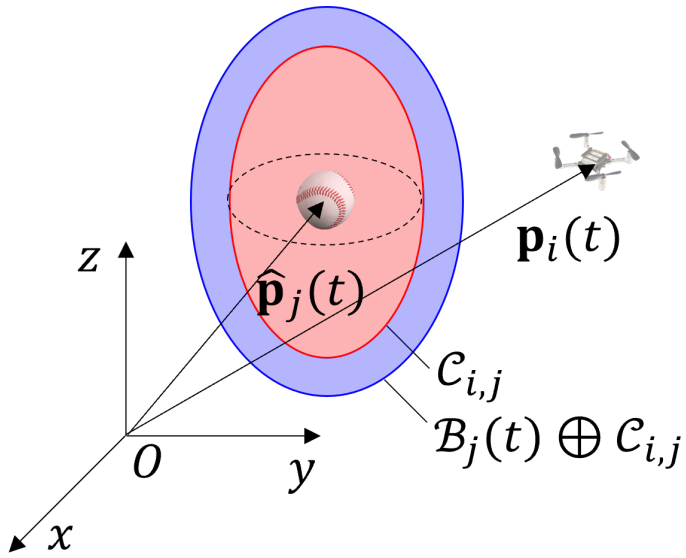


Figure 3.13: Collision model between agent i and object j (red ellipsoid) and inflated collision model to consider the error bound (blue ellipsoid).

This work utilizes a relative safe flight corridor (RSFC) [8] for dynamic obstacle avoidance. RSFC covers the obstacle's reachable region by inflating the collision model over time as follows:

$$\hat{\mathcal{C}}_{i,j}(t) = \{\mathbf{x} \in \mathbb{R}^3 \mid \|D_{i,j}\mathbf{x}\| \leq \hat{r}_j(t)\} \tag{3.53}$$

$$\hat{r}_j(t) = r + r_j + \frac{1}{2}a_{j,max}\min(t - T_h, M_e\Delta t)^2 \tag{3.54}$$

where $\hat{\mathcal{C}}_{i,j}(t)$ is the inflated collision model, $\hat{r}_j(t)$ is the radius of the inflated collision model, $a_{j,max}$ is the maximum acceleration of the obstacle j, and $M_e \geq 1$ is the number of error prediction segments. Since the inflated collision model includes the obstacle's reachable region, the sufficient condition of dynamic obstacle avoidance can be represented as follows:

$$\mathbf{p}_i(t) - \hat{\mathbf{p}}_j(t) \notin \hat{\mathcal{C}}_{i,j}(t), \forall t \in [T_h, T_{h+1}], h \tag{3.55}$$

where $\hat{\mathbf{p}}_j(t)$ is the initial trajectory of the obstacle j.

Lemma 3.7. *(Sufficient condition for dynamic obstacle avoidance) If the agent i satisfies (3.55) at the replanning steps h, then there is no collision between the agent i and the dynamic obstacle $j \in \mathcal{I}_o$ at the replanning steps h.*

Proof. Assume that the bound of the obstacle's trajectory is given as follows:

$$\mathbf{p}_j(t) - \hat{\mathbf{p}}_j(t) \in \mathcal{B}_j(t) \tag{3.56}$$

where $\hat{\mathbf{p}}_j(t)$ is the initial trajectory of the obstacle j, and $\mathcal{B}_j(t) = \{\mathbf{x} \in \mathbb{R}^3 \mid \|\mathbf{x}\| \leq \frac{1}{2}a_{j,max}(t - T_h)^2\}$ is a error bound. Let $\mathbf{a} \in \mathcal{C}_{i,j}$ and $\mathbf{b} \in \mathcal{B}_j(t)$. Then, $\|D_{i,j}(\mathbf{a} + \mathbf{b})\| \leq \|D_{i,j}\mathbf{a}\| + \|D_{i,j}\mathbf{b}\| \leq r + r_j + \frac{1}{2}\|a_{j,max}\|(t - T_h)^2$ since $\gamma_{i,j} \geq 1$. It implies that:

$$\mathcal{C}_{i,j} \oplus \mathcal{B}_j(t) \subset \hat{\mathcal{C}}_{i,j}(t), \forall t \in [T_h, T_h + M_e\Delta t] \tag{3.57}$$

Thus, the following equations hold for $\forall h, t \in [T_h, T_{h+1}]$:

$$\begin{aligned}
&\mathbf{p}_i(t) - \hat{\mathbf{p}}_j(t) \notin \hat{\mathcal{C}}_{i,j}(t) \\
&\Rightarrow \mathbf{p}_i(t) - \hat{\mathbf{p}}_j(t) \notin \mathcal{C}_{i,j} \oplus \mathcal{B}_j(t) \qquad (\because (3.57)) \\
&\Rightarrow \mathbf{p}_i(t) \notin \mathcal{C}_{i,j} \oplus \mathcal{B}_j(t) \oplus \{\hat{\mathbf{p}}_j(t)\} \\
&\Rightarrow \mathbf{p}_i(t) \notin \mathcal{C}_{i,j} \oplus \{\mathbf{p}_j(t)\} \qquad (\because (3.56)) \\
&\Rightarrow \mathbf{p}_i(t) - \mathbf{p}_j(t) \notin \mathcal{C}_{i,j}
\end{aligned} \tag{3.58}$$

In (3.58), the property of Minkowski sum that $\mathbf{a} \in \mathcal{A} \Leftrightarrow \mathbf{a} + \mathbf{b} \in \mathcal{A} \oplus \{\mathbf{b}\}$ is used. The last equation is equal to the original collision constraint (3.9). Therefore, if the agents satisfy (3.55) for all replanning step h, the agent does not collide with the dynamic obstacles. □

Based on the sufficient condition, the RSFC between the agent i and the obstacle j is defined as a linear constraint tangent to the inflated collision model:

$$\mathcal{L}_{m,k}^{i,j} = \{\mathbf{x} \in \mathbb{R}^3 \mid D_{i,j}(\mathbf{x} - \hat{\mathbf{c}}_{j,m,k}) \cdot \mathbf{n}_m^{i,j} - d_{m,k}^{i,j} > 0\} \tag{3.59}$$

$$d_{m,k}^{i,j} = \hat{r}_{j,m,k} + e_{i,j,m} \tag{3.60}$$

where $\mathbf{n}_m^{i,j}$ is the normal vector, $d_{m,k}^{i,j}$ is the safety margin, $\hat{r}_{j,m,k}$ is the control point of the Bernstein polynomial $\hat{r}_j(t)$, and $e_{i,j,m} \leq 0$ is the slack variable to prevent the constraint from becoming infeasible.

Fig. 3.14 describes the RSFC construction process. First, the coordinate transformation is performed to convert the inflated collision model to the sphere. Next, the proposed algorithm finds the closest point between the collision model and the segment of the relative initial trajectory, $\hat{\mathbf{p}}_i(t) - \hat{\mathbf{p}}_j(t)$. The RSFC is the tangent plane to the inflated collision model at the closest point (See the green plane of the middle figure of Fig. 3.14). Finally, the RSFC is generated by reversing the coordinate transform, as illustrated in the right figure of Fig. 3.14.

3.6 Subgoal Optimization

A subgoal is an intermediate goal point for the agents to reach the waypoint. As discussed in section 3.2.1, the subgoal must be placed in a feasible region that satisfies the collision constraints to avoid deadlock. In addition, it must converge to the waypoint to ensure goal convergence. For this reason, the proposed algorithm places the subgoal as the closest point to the waypoint within the feasible region. The subgoal is determined by

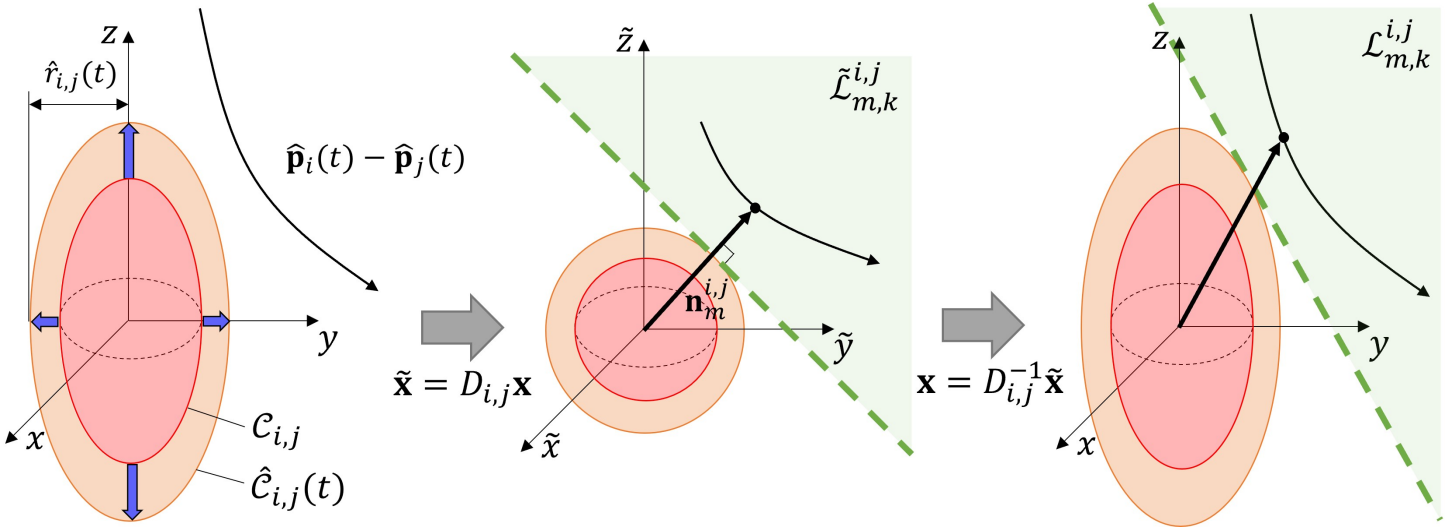


Figure 3.14: RSFC for dynamic obstacle avoidance. The red ellipsoid is the collision model between the agent and the moving obstacle, and the orange-shaded region is the inflated collision model that covers the reachable region of the dynamic obstacle. The green-shaded region is the LSC between the agent and the moving obstacle. The symbol with a tilde denotes the coordinate-transformed value.

solving the following optimization problem:

$$\begin{aligned} \underset{\mathbf{g}_i}{\text{minimize}} \quad & \|\mathbf{g}_i - \mathbf{w}_i\| \\ \text{subject to} \quad & \mathbf{g}_i \in [\mathbf{s}_i, \mathbf{w}_i] && \text{if } h = 0 \\ & \mathbf{g}_i \in [\mathbf{g}_i^{(h-1)}, \mathbf{w}_i] && \text{if } h > 0 \\ & \mathbf{g}_i \in \mathcal{S}_M \\ & \mathbf{g}_i \in \mathcal{L}_{M,n}^{i,j} && \forall j \in \mathcal{N}_i \end{aligned} \tag{3.61}$$

where $\mathbf{g}_i$ is the subgoal at the replanning step h. Note that the subgoal optimization problem can be converted to linear programming (LP) problem (See (4.31)). Lemma 3.8 shows the properties of the subgoal.

Lemma 3.8. *For the agents $i \in \mathcal{I}_a$, $j \in \mathcal{I}_a \backslash \{i\}$, the subgoal satisfies the following:*

(i) There exists a grid edge $e \in \mathcal{E}$ such that $[\boldsymbol{g}_i, \boldsymbol{w}_i] \subset [\boldsymbol{g}_i^{(h-1)}, \boldsymbol{w}_i] \subset e$ for $\forall h > 0$.

(ii) $\boldsymbol{g}_i \neq \boldsymbol{g}_j$.

(iii) $[\boldsymbol{g}_i^{(h-1)}, \boldsymbol{w}_i] \cap [\boldsymbol{g}_j^{(h-1)}, \boldsymbol{w}_j] \subset \mathcal{V}$ for $\forall h > 0$.

(iv) If there exists an edge $e \in \mathcal{E}$ such that $\boldsymbol{g}_i \in e$ and $\boldsymbol{g}_j \in e$, then $\boldsymbol{g}_i$ or $\boldsymbol{g}_j$ is on the grid vertex of the grid $G = (\mathcal{V}, \mathcal{E})$.

Proof. (i) If $h = 0$, there exists a grid edge $e \in \mathcal{E}$ such that $[\mathbf{s}_i, \mathbf{w}_i^{(0)}] \subset e$ because $\mathbf{s}_i$ is on the grid vertex by the assumption and $\mathbf{w}_i^{(0)}$ is the waypoint of discrete path from MAPP. Also, $\mathbf{g}_i^{(0)} \in [\mathbf{s}_i, \mathbf{w}_i^{(0)}]$ due to (3.61). Therefore, $[\mathbf{g}_i^{(0)}, \mathbf{w}_i^{(0)}] \subset [\mathbf{s}_i, \mathbf{w}_i^{(0)}] \subset e$.

Assume that there exists a grid edge $e^{(h-1)} \in \mathcal{E}$ such that $[\mathbf{g}_i^{(h-1)}, \mathbf{w}_i^{(h-1)}] \subset e^{(h-1)}$. If $\mathbf{g}_i^{(h-1)} \neq \mathbf{w}_i^{(h-1)}$, then $\mathbf{w}_i = \mathbf{w}_i^{(h-1)}$ by the waypoint update rule (3.17). Hence $[\mathbf{g}_i, \mathbf{w}_i] \subset [\mathbf{g}_i^{(h-1)}, \mathbf{w}_i] \subset e^{(h-1)}$ since $\mathbf{g}_i \in [\mathbf{g}_i^{(h-1)}, \mathbf{w}_i]$. If $\mathbf{g}_i^{(h-1)} = \mathbf{w}_i^{(h-1)}$, then there exists a grid edge e such that $[\mathbf{g}_i, \mathbf{w}_i] \subset [\mathbf{g}_i^{(h-1)}, \mathbf{w}_i] = [\mathbf{w}_i^{(h-1)}, \mathbf{w}_i] \subset e$ because $\mathbf{w}_i^{(h-1)}, \mathbf{w}_i$ are the consecutive waypoints of the discrete path from grid-based MAPP. Thus, there exists a grid edge $e \in \mathcal{E}$ such that $[\mathbf{g}_i, \mathbf{w}_i] \subset [\mathbf{g}_i^{(h-1)}, \mathbf{w}_i] \subset e$ for every replanning step by mathematical induction.

(ii) $\|\mathbf{g}_i - \mathbf{g}_j\| \geq 2r$ because $\mathbf{g}_i \in \mathcal{L}_{M,n}^{i,j}$ and $\mathbf{g}_j \in \mathcal{L}_{M,n}^{j,i}$. It implies that $\mathbf{g}_i \neq \mathbf{g}_j$.

(iii) If $h = 0$, $[\mathbf{s}_i, \mathbf{w}_i^{(0)}] \cap [\mathbf{s}_j, \mathbf{w}_j^{(0)}] \subset \mathcal{V}$ because the waypoints are planned using grid-based MAPP that ensures collision avoidance. Thus, $[\mathbf{g}_i^{(0)}, \mathbf{w}_i^{(0)}] \cap [\mathbf{g}_j^{(0)}, \mathbf{w}_j^{(0)}] \subset \mathcal{V}$ since $[\mathbf{g}_i^{(0)}, \mathbf{w}_i^{(0)}] \subset [\mathbf{s}_i, \mathbf{w}_i^{(0)}]$ by (3.61).

Suppose that $[\mathbf{g}_i^{(h-1)}, \mathbf{w}_i^{(h-1)}] \cap [\mathbf{g}_j^{(h-1)}, \mathbf{w}_j^{(h-1)}] \subset \mathcal{V}$.

(Case 1) If $\mathbf{g}_i^{(h-1)} \neq \mathbf{w}_i^{(h-1)}$ and $\mathbf{g}_j^{(h-1)} \neq \mathbf{w}_j^{(h-1)}$, $[\mathbf{g}_i^{(h-1)}, \mathbf{w}_i] \cap [\mathbf{g}_j^{(h-1)}, \mathbf{w}_j] \subset \mathcal{V}$ since $\mathbf{w}_i^{(h-1)} = \mathbf{w}_i$ and $\mathbf{w}_j^{(h-1)} = \mathbf{w}_j$ by the waypoint update rule (3.17). Therefore, $[\mathbf{g}_i, \mathbf{w}_i] \cap [\mathbf{g}_j, \mathbf{w}_j] \subset [\mathbf{g}_i^{(h-1)}, \mathbf{w}_i] \cap [\mathbf{g}_j^{(h-1)}, \mathbf{w}_j] \subset \mathcal{V}$ by (3.61).

(Case 2) If $\mathbf{g}_i^{(h-1)} = \mathbf{w}_i^{(h-1)}$ and $\mathbf{g}_j^{(h-1)} = \mathbf{w}_j^{(h-1)}$, then $[\mathbf{g}_i^{(h-1)}, \mathbf{w}_i] \cap [\mathbf{g}_j^{(h-1)}, \mathbf{w}_j] = [\mathbf{w}_i^{(h-1)}, \mathbf{w}_i] \cap [\mathbf{w}_j^{(h-1)}, \mathbf{w}_j] \subset \mathcal{V}$ since Alg. 2 updates the waypoints without any conflict. Therefore, $[\mathbf{g}_i, \mathbf{w}_i] \cap [\mathbf{g}_j, \mathbf{w}_j] \subset [\mathbf{g}_i^{(h-1)}, \mathbf{w}_i] \cap [\mathbf{g}_j^{(h-1)}, \mathbf{w}_j] \subset \mathcal{V}$.

(Case 3) Assume that only one of the subgoals is equal to the waypoint. In other words, $\mathbf{g}_i^{(h-1)} \neq \mathbf{w}_i^{(h-1)}$ and $\mathbf{g}_j^{(h-1)} = \mathbf{w}_j^{(h-1)}$ without loss of generality. Then, $[\mathbf{g}_i^{(h-1)}, \mathbf{w}_i] \cap [\mathbf{g}_j^{(h-1)}, \mathbf{w}_j] = [\mathbf{g}_i^{(h-1)}, \mathbf{w}_i^{(h-1)}] \cap [\mathbf{w}_j^{(h-1)}, \mathbf{w}_j]$ by the waypoint update rule (3.17). Here, a grid edge $e^{(h-1)} \in \mathcal{E}$ such that $[\mathbf{g}_i^{(h-1)}, \mathbf{w}_i^{(h-1)}] \subset e^{(h-1)}$ can be found due to (i) of Lemma 3.8. Also, it satisfies $[\mathbf{w}_j^{(h-1)}, \mathbf{w}_j] \cap e^{(h-1)} \subset \mathcal{V}$ because $\mathbf{w}_j^{(h-1)} \neq \mathbf{w}_i^{(h-1)}$ and $\mathbf{w}_j \neq \mathbf{w}_i = \mathbf{w}_i^{(h-1)}$ by Lemma 3.1. Therefore, $[\mathbf{g}_i, \mathbf{w}_i] \cap [\mathbf{g}_j, \mathbf{w}_j] \subset [\mathbf{g}_i^{(h-1)}, \mathbf{w}_i] \cap [\mathbf{g}_j^{(h-1)}, \mathbf{w}_j] \subset \mathcal{V}$. Thus, (iii) of Lemma 3.8 holds for every replanning step by mathematical induction.

(iv) Assume that there exists an edge e such that $\mathbf{g}_i^{(0)} \in e$, $\mathbf{g}_j^{(0)} \in e$. Since $\mathbf{s}_i$, $\mathbf{s}_j$, $\mathbf{w}_i^{(0)}$, and $\mathbf{w}_j^{(0)}$ are on the vertex of the grid, the necessary condition of $\mathbf{g}_i^{(0)} \notin \mathcal{V}$ and $\mathbf{g}_j^{(0)} \notin \mathcal{V}$ is:

$$[\mathbf{s}_i, \mathbf{w}_i^{(0)}] \cap [\mathbf{s}_j, \mathbf{w}_j^{(0)}] \neq \emptyset \tag{3.62}$$

However, it implies that there exists a collision between discrete paths, which is impossible by the MAPP algorithm. Thus, $\mathbf{g}_i^{(0)} = \mathbf{w}_i^{(0)} \in \mathcal{V}$ or $\mathbf{g}_j^{(0)} = \mathbf{w}_j^{(0)} \in \mathcal{V}$.

Assume that (iv) of Lemma 3.8 holds at the replanning step $h-1$ and the edge e satisfies $\mathbf{g}_i \in e$ and $\mathbf{g}_j \in e$. If $\mathbf{g}_i^{(h-1)} \notin e$ or $\mathbf{g}_j^{(h-1)} \notin e$, then $\mathbf{g}_i = \mathbf{w}_i \in \mathcal{V}$ or $\mathbf{g}_j = \mathbf{w}_j \in \mathcal{V}$ by the constraint $\mathbf{g}_i \in [\mathbf{g}_i^{(h-1)}, \mathbf{w}_i]$. If $\mathbf{g}_i^{(h-1)} \in e$ and $\mathbf{g}_j^{(h-1)} \in e$, then $\mathbf{g}_i^{(h-1)} \in \mathcal{V}$ or $\mathbf{g}_j^{(h-1)} \in \mathcal{V}$ by the

assumption. Suppose that $\mathbf{g}_i^{(h-1)} \in \mathcal{V}$ without loss of generality. If $\mathbf{w}_j = \mathbf{g}_i^{(h-1)}$, then $\mathbf{w}_i \notin e$ due to (iii) of Lemma 3.8. If $\mathbf{w}_j \neq \mathbf{g}_i^{(h-1)}$, then the waypoint $\mathbf{w}_i$ satisfies $\mathbf{w}_i = \mathbf{g}_i^{(h-1)}$ or $\mathbf{w}_i \notin e$ since the waypoints cannot be duplicated by Lemma 3.1. Therefore, $\mathbf{g}_i = \mathbf{g}_i^{(h-1)} \in \mathcal{V}$ for every case because of the constraint $\mathbf{g}_i \in [\mathbf{g}_i^{(h-1)}, \mathbf{w}_i]$ and the assumption that $\mathbf{g}_i \in e$. To summarize, if there exists an edge $e \in \mathcal{E}$ such that $\mathbf{g}_i \in e$ or $\mathbf{g}_j \in e$, then $\mathbf{g}_i \in \mathcal{V}$ or $\mathbf{g}_j \in \mathcal{V}$ by the mathematical induction. □

3.7 Trajectory Optimization

3.7.1 Cost function

The cost functions are formulated to minimize the distance to the subgoal, the jerk of the trajectory, and the size of the slack variables for dynamic obstacle avoidance:

$$J_{err}(\mathbf{p}_i(t)) = w_{err} \sum_{m=1}^{M} \delta_m \|\mathbf{c}_{i,m,n} - \mathbf{g}_i\|^2 \tag{3.63}$$

$$J_{der}(\mathbf{p}_i(t)) = w_{der} \int_{T_h}^{T_{h+M}} \left\| \frac{d^3}{dt^3} \mathbf{p}_i(t) \right\|^2 dt \tag{3.64}$$

$$J_{slk} = w_{slk} \sum_{j \in \mathcal{I}_o} \sum_{m=1}^{M} e_{i,j,m}^2 \tag{3.65}$$

where $w_m, w_{der}, w_{slk} > 0$ are the weight coefficients, and δ_m is discrete delta function defined as follows:

$$\delta_m = \begin{cases} 1 & m = M \text{ or } \hat{\mathbf{c}}_{i,m+1,n} = \mathbf{g}_i \\ 0 & \text{else} \end{cases} \tag{3.66}$$

3.7.2 Communication range

If the communication range is not considered when generating the trajectory, the agent may collide with an agent outside the range. Moreover, if the distance between the agent

and its waypoint is longer than half the communication range, an agent outside the range can assign the same waypoint. Therefore, the following constraints are added to prevent the collision and duplicated waypoints between agents outside the range:

$$\|\mathbf{c}_{i,m+l,k} - \mathbf{c}_{i,m,0}\|_\infty \leq \frac{r_c}{2} - r, \forall l > 0, m, k, \tag{3.67}$$

$$\|\mathbf{c}_{i,m,n} - \mathbf{w}_i\|_\infty \leq \frac{r_c}{2}, \forall m \tag{3.68}$$

where r_c is the communication range.

3.7.3 Other constraints

The properties of Bernstein polynomial (2.5), (2.7), and (2.8) are utilized to formulate the initial condition to match the agent's current state. Similarly, the continuity constraints are imposed to make the trajectory continuous up to the acceleration. The dynamical limit (3.10), (3.11) can be represented to affine inequality using the convex hull property of the Bernstein polynomial (2.4). Lastly, the final stop condition is added for the feasibility of the optimization problem (i.e., $\mathbf{c}_{i,M,n} = \mathbf{c}_{i,M,n-1} = \mathbf{c}_{i,M,n-2}$). These constraints can be represented as the affine constraints:

$$A_{eq}\mathbf{c}_i = \mathbf{b}_{eq} \tag{3.69}$$

$$A_{dyn}\mathbf{c}_i \preceq \mathbf{b}_{dyn} \tag{3.70}$$

where $\mathbf{c}_i$ is the vector that concatenates the control points of the trajectory $\mathbf{p}_i(t)$.

3.7.4 Optimization problem

The trajectory optimization is conducted by solving the following quadratic programming (QP) problem:

$$\begin{aligned}
\underset{\mathbf{c}_i}{\text{minimize}} \quad & J_{err} + J_{der} + J_{slk} \\
\text{subject to} \quad & \mathbf{c}_{i,m,k} \in \mathcal{S}_m && \forall m, k \\
& \mathbf{c}_{i,m,k} \in \mathcal{L}_{m,k}^{i,j} && \forall j \in \mathcal{N}_i \cup \mathcal{I}_o, m, k \\
& e_{i,j,m} \leq 0 && \forall j \in \mathcal{I}_o, m \\
& (3.67), (3.68), (3.69), (3.70)
\end{aligned} \tag{3.71}$$

The trajectory optimization problem can be solved by using a conventional convex solver. The time complexity of the convex QP solver is known to $O(N^3 L)$ [55] where N is the number of decision variables, and L is the number of bits in the input, which is proportional to the number of inequality constraints. Since the number of inequality constraints is proportional to the number of robots and the computation time of other modules is negligible when the number of agents is large enough, the computation time of the proposed algorithm increases approximately linearly with the number of robots.

4

Theoretical Guarantee

This chapter addresses the theoretical properties of the proposed algorithm. Section 4.1 describes proof that the proposed algorithm guarantees collision avoidance. Section 4.2 presents proof that the proposed algorithm ensures the feasibility of the optimization problem so that the planner never fails to return the trajectory during the mission. Section 4.3 proves that the proposed algorithm guarantees goal convergence if the communication range is large enough.

4.1 Collision Avoidance

Since the proposed algorithm utilizes a safe flight corridor (SFC) and linear safe corridor (LSC), it guarantees inter-collision avoidance and static obstacle avoidance during the mission. Moreover, Theorem 4.1 presents that the proposed algorithm prevents the collision between agents out of the communication range.

Theorem 4.1. *(Collision avoidance) The trajectory from (3.71) does not cause inter-agent collision or collision between the agent and static obstacles.*

Proof. By Lemma 3.2, there exists $\mathcal{S}_m$ for $\forall h, m$ that satisfies $(\mathcal{S}_m \oplus \mathcal{C}_i) \cap \mathcal{O} = \emptyset$. Therefore, the constraint $\mathbf{c}_{i,m,k} \in \mathcal{S}_m$ ensures static obstacle avoidance due to the convex hull property of Bernstein polynomial (2.4).

For the agent $j \in \mathcal{N}_i$, there is no collision between the agents i and j due to Lemma 3.4. For the agent $j \notin \mathcal{N}_i$, the following inequality holds for all agents due to (3.67):

$$\|\mathbf{c}_{i,m,k} - \mathbf{c}_{i,1,0}\|_\infty \leq \frac{r_c}{2} - r, \forall i \in \mathcal{I}_a, m, k \tag{4.1}$$

Due to the convex hull property (2.4) and end-point property (2.5) of Bernstein polynomial, the following holds for all agent $i \in \mathcal{I}$:

$$\|\mathbf{p}_i(t) - \mathbf{c}_{i,1,0}\|_\infty \leq \frac{r_c}{2} - r, \forall i \in \mathcal{I}_a, t \in [T_h, T_{h+M}] \tag{4.2}$$

$$\|\mathbf{p}_i(t) - \mathbf{p}_i(T_h)\|_\infty \leq \frac{r_c}{2} - r, \forall i \in \mathcal{I}_a, t \in [T_h, T_{h+M}] \tag{4.3}$$

Due to the assumption that $j \notin \mathcal{N}_i$:

$$\|\mathbf{p}_i(T_h) - \mathbf{p}_j(T_h)\|_\infty \geq r_c \tag{4.4}$$

$$\|\mathbf{p}_i(T_h) - \mathbf{p}_i(t) + \mathbf{p}_j(t) - \mathbf{p}_j(T_h) + \mathbf{p}_i(t) - \mathbf{p}_j(t)\|_\infty \geq r_c \tag{4.5}$$

According to (4.3) and triangle inequality:

$$\begin{aligned} &\|\mathbf{p}_i(t) - \mathbf{p}_i(T_h)\| + \|\mathbf{p}_j(t) - \mathbf{p}_j(T_h)\| \\ &+ \|\mathbf{p}_i(t) - \mathbf{p}_j(t)\|_\infty \geq r_c \end{aligned} \tag{4.6}$$

$$\|\mathbf{p}_i(t) - \mathbf{p}_j(t)\| \geq \|\mathbf{p}_i(t) - \mathbf{p}_j(t)\|_\infty \geq 2r \tag{4.7}$$

Therefore, there is no collision between the agents i and j. In conclusion, the trajectory from (3.71) guarantees inter-agent collision avoidance and static obstacle avoidance. □

Also, it should be noted the proposed algorithm guarantees dynamic obstacle avoidance

if the slack variables in the RSFC are all zero.

Theorem 4.2. *(Dynamic obstacle avoidance) If* $c_{i,m,k} \in \mathcal{L}_{m,k}^{i,j}$ *and* $e_{i,j,m} = 0$ *for* $\forall m, k$*, then the agent* i *does not collide with the dynamic obstacle* $j \in \mathcal{I}_o$ *at the replanning steps* h*.*

Proof. Since $e_{i,j,m} = 0$ for $\forall m$, $d_{m,k}^{i,j} = \hat{r}_{j,m,k}$. The following inequalities holds by (3.59) for $\forall j \in \mathcal{I}_o, m, k$:

$$D_{i,j}(\mathbf{c}_{i,m,k} - \hat{\mathbf{c}}_{j,m,k}) \cdot \mathbf{n}_m^{i,j} - d_{m,k}^{i,j} > 0 \tag{4.8}$$

$$D_{i,j}(\mathbf{c}_{i,m,k} - \hat{\mathbf{c}}_{j,m,k}) \cdot \mathbf{n}_m^{i,j} - \hat{r}_{j,m,k} > 0, \tag{4.9}$$

Here, the Bernstein basis is multiplied as follows:

$$\sum_{k=0}^{n} D_{i,j}(\mathbf{c}_{i,m,k} - \hat{\mathbf{c}}_{j,m,k}) b_{k,n}(\tau_{h,m}) \cdot \mathbf{n}_m^{i,j} - \hat{r}_{j,m,k} b_{k,n}(\tau_{h,m}) > 0 \tag{4.10}$$

$$D_{i,j}(\mathbf{p}_i(t) - \hat{\mathbf{p}}_j(t)) \cdot \mathbf{n}_m^{i,j} - \hat{r}_j(t) > 0, t \in [T_{h+m-1}, T_{h+m}] \tag{4.11}$$

where $\tau_{h,m} = (t - T_{h+m-1})/\Delta t$. It implies that $\mathbf{p}_i(t) - \hat{\mathbf{p}}_j(t) \notin \hat{\mathcal{C}}_{i,j}(t)$ for $\forall m, t \in [T_{h+m-1}, T_{h+m}]$. Thus, there is no collision between the agent i and j at the replanning step h by Lemma 3.7. □

4.2 Feasibility of Contraints

This section presents the proof that Alg. 1 ensures it returns a feasible trajectory for any arbitrary input. This can be proved by showing that the initial trajectory $\mathbf{p}_i(t)$ is one of the solutions to the trajectory optimization problem. Lemma 4.1 shows that the feasible SFC exists for every replanning step.

Lemma 4.1. *(Existence of the feasible SFC) Assume that* $\boldsymbol{c}_{i,m,k}^{(h-1)} \in \mathcal{S}_m^{(h-1)}$ *for* $\forall m, k$ *at the replanning step* $h > 0$*. Then,* $\hat{\boldsymbol{c}}_{i,m,k} \in \mathcal{S}_m$ *for* $\forall h, m, k$*.*

Proof. If $h = 0$, then $\hat{\mathbf{c}}_{i,m,k} = \mathbf{s}_i \in \mathcal{S}_m^{(0)}$ for $\forall m, k$ due to (3.23). If $h > 0$, $m < M$, and (3.28) is satisfied, $\hat{\mathbf{c}}_{i,m,k} \subset \text{Conv}(\{\hat{\mathbf{c}}_{i,m,0}, \cdots, \hat{\mathbf{c}}_{i,m,n}\}) \subset \mathcal{S}_{m+2}^{(h-1)} = \mathcal{S}_m$. If $h > 0$, $m < M$, and (3.28) is not satisfied, $\hat{\mathbf{c}}_{i,m,k} = \mathbf{c}_{i,m+1,k}^{(h-1)} \in \mathcal{S}_{m+1}^{(h-1)} = \mathcal{S}_m$ for $\forall k$ due to (3.23) and (3.27). If $h > 0$ and $m = M$, then $\hat{\mathbf{c}}_{i,M,k} = \hat{\mathbf{c}}_{i,M,n} \in \mathcal{S}(\{\hat{\mathbf{c}}_{i,M,n}, \mathbf{g}_i^{(h-1)}\}) \subset \mathcal{S}_M$ for $\forall k$ by (3.23) and (3.27). Thus, $\hat{\mathbf{c}}_{i,m,k} \in \mathcal{S}_m$ for $\forall h, m, k$. □

Suppose that $\hat{\mathcal{H}}_{i,j,m}$ is the convex hull of the control points of relative initial trajectory between the agents i and j:

$$\hat{\mathcal{H}}_{i,j,m} = \text{Conv}(\{\hat{\mathbf{c}}_{i,m,k} - \hat{\mathbf{c}}_{j,m,k} \mid k \in \mathcal{I}_n\}) \tag{4.12}$$

Lemma 4.2 presents that the feasible LSC always exists for every replanning step.

Lemma 4.2. *(Existence of the feasible LSC) Assume that $\mathcal{H}_{i,j,m}^{(h-1)} \cap \mathcal{C}_{i,j} = \emptyset$, for $\forall j \in \mathcal{N}_i, h > 0, m, k$. Then, there exists $\mathcal{L}_{m,k}^{i,j}$ that satisfies (3.44), (3.45), and $\hat{\mathbf{c}}_{i,m,k} \in \mathcal{L}_{m,k}^{i,j}$ for $\forall j \in \mathcal{N}_i, h, m, k$.*

Proof. If $h = 0$, the normal vector of LSC can be given as follows:

$$\mathbf{n}_m^{i,j} = \frac{\mathbf{s}_i - \mathbf{s}_j}{\|\mathbf{s}_i - \mathbf{s}_j\|} \tag{4.13}$$

Therefore, $\hat{\mathbf{c}}_{i,m,k} = \mathbf{s}_i \in \mathcal{L}_{m,k}^{i,j}$ for $\forall j \in \mathcal{N}_i, m, k$:

$$\begin{aligned} \mathbf{s}_i \in \mathcal{L}_{m,k}^{i,j} &\Leftrightarrow (\mathbf{s}_i - \hat{\mathbf{c}}_{j,m,k}) \cdot \mathbf{n}_m^{i,j} - d_{m,k}^{i,j} \geq 0 \\ &\Leftrightarrow (\mathbf{s}_i - \mathbf{s}_j) \cdot \mathbf{n}_m^{i,j} - (r + \frac{1}{2}(\mathbf{s}_i - \mathbf{s}_j) \cdot \mathbf{n}_m^{i,j}) \geq 0 \\ &\Leftrightarrow \frac{1}{2}(\mathbf{s}_i - \mathbf{s}_j) \cdot \mathbf{n}_m^{i,j} - r \geq 0 \Rightarrow \|\mathbf{s}_i - \mathbf{s}_j\| \geq 2r \end{aligned} \tag{4.14}$$

If $h > 0$ and $m < M$, the following equation holds by (3.23):

$$\hat{\mathcal{H}}_{i,j,m} = \mathcal{H}_{i,j,m+1}^{(h-1)} \tag{4.15}$$

Therefore, $\hat{\mathcal{H}}_{i,j,m}$ and $\mathcal{C}_{i,j}$ are disjoint non-empty convex sets due to Lemma 4.2:

$$\hat{\mathcal{H}}_{i,j,m} \cap \mathcal{C}_{i,j} = \emptyset \tag{4.16}$$

By the hyperplane separation theorem [53], there exists $\mathbf{n}_s$ such that:

$$\min\langle \hat{\mathcal{H}}_{i,j,m}, \mathbf{n}_s \rangle \geq 2r \tag{4.17}$$

where $\min\langle \hat{\mathcal{H}}_{i,j,m}, \mathbf{n}_s \rangle = \min_{\mathbf{x} \in \hat{\mathcal{H}}_{i,j,m}} \mathbf{x} \cdot \mathbf{n}_s$. Here, the normal vector of the LSC can be given as follows:

$$\mathbf{n}_m^{i,j} = -\mathbf{n}_m^{j,i} = \mathbf{n}_s \tag{4.18}$$

Then, the following equation holds:

$$\begin{aligned} &(\hat{\mathbf{c}}_{i,m,k} - \hat{\mathbf{c}}_{j,m,k}) \cdot \mathbf{n}_m^{i,j} - d_{m,k}^{i,j} \\ &\quad = \frac{1}{2}(\hat{\mathbf{c}}_{i,m,k} - \hat{\mathbf{c}}_{j,m,k}) \cdot \mathbf{n}_m^{i,j} - r \geq 0 \end{aligned} \tag{4.19}$$

Therefore, $\hat{\mathbf{c}}_{i,m,k} \in \mathcal{L}_{m,k}^{i,j}$ for $\forall j \in \mathcal{N}_i, m < M, k$.

If $h > 0$ and $m = M$, then $\hat{\mathbf{c}}_{i,M,k} = \hat{\mathbf{c}}_{i,M,n}$ for $\forall k$ by (3.23). Also, the LSC when $m = M$ satisfies $\hat{\mathbf{c}}_{i,M,n} \in \mathcal{L}_{M,k}^{i,j}$ for $\forall k$ because $[\hat{\mathbf{c}}_{i,M,n}, \mathbf{g}_i^{(h-1)}] \subset \mathcal{L}_{M,k}^{i,j}$ for $\forall k$. Therefore, $\hat{\mathbf{c}}_{i,M,k} \in \mathcal{L}_{M,k}^{i,j}$ for $\forall j \in \mathcal{N}_i, k$. This concludes the proof. □

Based on the above Lemmas, Theorem 4.3 shows that there exists at least one solution that satisfies all constraints of the trajectory optimization problem.

Theorem 4.3. *(Feasibility of constraints) If the segment duration is equal to the replanning period Δt, $\hat{\boldsymbol{p}}_i(t)$ is one of the solutions of (3.71) for every replanning step.*

Proof. If $h = 0$, $\hat{\mathbf{p}}_i(t)$ satisfies SFC and LSC constraints due to Lemmas 4.1, 4.2. The constraints for dynamic obstacle avoidance are negligible due to slack variables. $\hat{\mathbf{p}}_i(t)$ fulfills the initial condition, continuity constraint, final stop condition, dynamical limit constraints,

and (3.67) because $\hat{\mathbf{c}}_{i,m,k} = \mathbf{s}_i$ for $\forall m, k$. $\hat{\mathbf{p}}_i(t)$ also satisfies (3.68) due to the assumption that $d > 2\sqrt{2}r$. Therefore, $\hat{\mathbf{p}}_i(t)$ is one of the solutions that satisfy all constraints in (3.71).

Assume that there exists a solution at the previous replanning step $h-1$. Then, $\hat{\mathbf{p}}_i(t)$ satisfies SFC and LSC constraints due to Lemmas 4.1, 4.2. The constraints for dynamic obstacle avoidance are negligible due to slack variables. $\hat{\mathbf{p}}_i(t)$ fulfills the initial condition, continuity constraint, final stop condition, and dynamical limit constraints, (3.67), and (3.68) due to (3.23). Therefore, the solution of (3.71) always exists for every replanning step by mathematical induction. □

Moreover, Theorem 4.4 presents that the solution to the subgoal optimization problem always exists for every replanning step.

Theorem 4.4. *(Feasibility of subgoal optimization) The solution of (3.61) always exists for every replanning step.*

Proof. If $h = 0$, the start point $\mathbf{s}_i$ satisfies all constraints of the subgoal optimization problem (3.61). If $h > 0$, $\mathbf{g}_i^{(h-1)}$ satisfies the constraints of the problem because $\mathbf{g}_i^{(h-1)} \in \mathcal{S}_M$ by (3.27) and $\mathbf{g}_i^{(h-1)} \in \mathcal{L}_{M,n}^{i,j}$ by (3.45). This concludes the proof. □

Theorems 4.3 and 4.4 ensure that Alg. 1 always returns the feasible trajectory for any arbitrary inputs if the grid-based MAPP is complete.

4.3 Goal Convergence

This section demonstrates that the proposed algorithm guarantees goal convergence under the assumption that all agents are connected by the network and there is no dynamic obstacle. The proof consists of three steps. First, I demonstrate the convergence of the agent towards the subgoal. Next, I prove that the subgoal converges to the waypoint. Finally, I complete the proof by demonstrating that the proposed decentralized MAPP allows the waypoint to reach the desired goal.

Lemma 4.3 shows that the cost function of the trajectory optimization problem monotonically decreases for every replanning step. For the proof, the sum of the cost functions of the trajectory $\mathbf{p}_i(t)$ will be denoted as $J(\mathbf{p}_i(t))$.

Lemma 4.3. *Assume that there is no dynamic obstacle and the subgoal for the agent i is fixed. Then, the cost function of the agent's trajectory monotonically decreases for each replanning step as follows:*

$$J(\mathbf{p}_i^{(h-1)}(t)) \geq J(\hat{\mathbf{p}}_i(t)) \geq J(\mathbf{p}_i(t)) \tag{4.20}$$

Proof. Due to the definition of the initial trajectory (3.22), $J_{err}(\mathbf{p}_i^{(h-1)}(t)) \geq J_{err}(\hat{\mathbf{p}}_i(t))$. Moreover, the following inequality holds for every replanning step:

$$\begin{aligned} &J_{der}(\mathbf{p}_i^{(h-1)}(t)) - J_{der}(\hat{\mathbf{p}}_i(t)) \\ &\quad = w_{der} \int_{T_{h-1}}^{T_h} \left\| \frac{d^3}{dt^3} \mathbf{p}_i^{(h-1)}(t) \right\|^2 dt \geq 0 \end{aligned} \tag{4.21}$$

Therefore, $J(\mathbf{p}_i^{(h-1)}(t)) \geq J(\hat{\mathbf{p}}_i(t))$.

By Theorem 4.3, $\hat{\mathbf{p}}_i(t)$ is one of the solutions to the trajectory optimization problem. Since $\mathbf{p}_i(t)$ is the optimal solution to the optimization problem, $J(\hat{\mathbf{p}}_i(t)) \geq J(\mathbf{p}_i(t))$. This concludes the proof. □

Based on Lemma 4.3, Lemma 4.4 presents that the proposed algorithm allows the agent to reach the subgoal.

Lemma 4.4. *(Convergence to the subgoal) Assume that there is no dynamic obstacle and the subgoal for the agent i is fixed. Then, the agent i converges to the subgoal.*

Proof. Assume that $\hat{\mathbf{c}}_{i,M,n} \neq \mathbf{g}_i$. Then, the trajectory $\tilde{\mathbf{p}}_i(t)$ with the following control points

can be generated:

$$\tilde{\mathbf{c}}_{i,m,k} = \begin{cases} \hat{\mathbf{c}}_{i,M,n} + \lambda(\mathbf{g}_i - \hat{\mathbf{c}}_{i,M,n}) & m = M, k \geq n-2 \\ \hat{\mathbf{c}}_{i,m,k} & else \end{cases} \tag{4.22}$$

where $\tilde{\mathbf{c}}_{i,m,k}$ is the control point of $\tilde{\mathbf{p}}_i(t)$, and $\lambda \in [0,1]$. Here, $\tilde{\mathbf{p}}_i(t)$ is one of the solutions of the trajectory optimization problem (3.71) if λ is small enough. It satisfies the initial condition, continuity constraint, and final stop condition. If $m < M$, $\tilde{\mathbf{p}}_i(t)$ satisfies the collision constraints due to Theorem 4.1. If $m = M$, $\tilde{\mathbf{c}}_{i,M,k} \in [\hat{\mathbf{c}}_{i,M,n}, \mathbf{g}_i] \subset \mathcal{S}_M \cap \mathcal{L}_{i,M,k}$ since the subgoal is fixed as $\mathbf{g}_i^{(h-1)} = \mathbf{g}_i$. It indicates that $\tilde{\mathbf{p}}_i(t)$ fulfills the collision constraints. Therefore, $\tilde{\mathbf{p}}_i(t)$ is a feasible solution of the trajectory optimization problem (3.71) if λ satisfies the following:

$$\|\lambda(\mathbf{g}_i - \tilde{\mathbf{c}}_{i,M,n})\|_\infty \leq \frac{r_c}{2} - r \tag{4.23}$$

$$\left\| \frac{n\lambda(\mathbf{g}_i - \tilde{\mathbf{c}}_{i,M,n})}{\Delta t} \right\|_\infty \leq v_{max} \tag{4.24}$$

$$\| \frac{n(n-1)\lambda(\mathbf{g}_i - \tilde{\mathbf{c}}_{i,M,n})}{\Delta t^2} \|_\infty \leq a_{max} \tag{4.25}$$

where (4.23) is the sufficient condition of the communication range constraint (3.67), and (4.24), (4.25) are the sufficient conditions of dynamical limit constraints, which can be derived using the convex hull property of Bernstein polynomial. Note that if λ is small enough, there exists non-zero λ that satisfies the above constraints.

The difference between $J_{err}(\hat{\mathbf{p}}_i(t))$ and $J_{err}(\tilde{\mathbf{p}}_i(t))$ is:

$$\begin{aligned} & J_{err}(\hat{\mathbf{p}}_i(t)) - J_{err}(\tilde{\mathbf{p}}_i(t)) \\ & = w_{err}\|\mathbf{g}_i - \hat{\mathbf{c}}_{i,M,n}\|^2(1-(1-\lambda)^2) \\ & = A(1-(1-\lambda)^2) \end{aligned} \tag{4.26}$$

where $A = w_{err}\|\mathbf{g}_i - \hat{\mathbf{c}}_{i,M,n}\|^2 > 0$. The difference between $J_{der}(\hat{\mathbf{p}}_i(t))$ and $J_{der}(\tilde{\mathbf{p}}_i(t))$ is:

$$\begin{aligned}
&J_{err}(\hat{\mathbf{p}}_i(t)) - J_{err}(\tilde{\mathbf{p}}_i(t)) \\
&= -\int_{T_{h+M-1}}^{T_{h+M}} \left\| \frac{d^3}{dt^3}\tilde{\mathbf{p}}_i(t) \right\|^2 dt \\
&= -\int_{T_{h+M-1}}^{T_{h+M}} \left\| \frac{d^3}{dt^3}(\tilde{\mathbf{p}}_i(t) - \mathbf{c}_{i,M,n}) \right\|^2 dt \\
&= -\int_{T_{h+M-1}}^{T_{h+M}} \left\| \frac{d^3}{dt^3}\lambda\Delta\mathbf{p}_i(t) \right\|^2 dt \\
&= -\lambda^2 \int_{T_{h+M-1}}^{T_{h+M}} \left\| \frac{d^3}{dt^3}\Delta\mathbf{p}_i(t) \right\|^2 dt \\
&= -B\lambda^2
\end{aligned} \tag{4.27}$$

where $B > 0$ is a positive scalar value, and $\Delta\mathbf{p}_i(t)$ is the Bernstein polynomial with the control point $\Delta\mathbf{c}_{i,m,k}$, which is given as follows:

$$\Delta\mathbf{c}_{i,m,k} = \begin{cases} \mathbf{g}_i - \hat{\mathbf{c}}_{i,M,n} & m = M, k \geq n-2 \\ \mathbf{0} & else \end{cases} \tag{4.28}$$

To summarize, the cost difference between $\hat{\mathbf{p}}_i(t)$ and $\tilde{\mathbf{p}}_i(t)$ is represented as follows:

$$\begin{aligned}
\Delta J &= J(\hat{\mathbf{p}}_i(t)) - J(\tilde{\mathbf{p}}_i(t)) \\
&= A(1-(1-\lambda)^2) - B\lambda^2 \\
&= 2A\lambda - (A+B)\lambda^2
\end{aligned} \tag{4.29}$$

Note that $\Delta J > 0$ when $0 < \lambda < \frac{2A}{A+B}$. Thus, if λ satisfies $0 < \lambda < \frac{2A}{A+B}$, then $J(\mathbf{p}_i^{(h-1)}(t)) \geq J(\hat{\mathbf{p}}_i(t)) > J(\tilde{\mathbf{p}}_i(t)) \geq J(\mathbf{p}_i(t))$ due to Lemma 4.3 and the fact that $\mathbf{p}_i(t)$ is the optimal solution of the trajectory optimization problem. It implies that the proposed algorithm makes the cost function strictly decreasing until $\hat{\mathbf{c}}_{i,M,n} = \mathbf{g}_i$.

Assume that $\hat{\mathbf{c}}_{i,M,n} = \cdots = \hat{\mathbf{c}}_{i,M-l+1,n} = \mathbf{g}_i$ and $\mathbf{p}_i^{(h-1)}(t) \neq \mathbf{g}_i$. Then, the following

holds:

$$\begin{aligned}
&J(\mathbf{p}_i^{(h-1)}(t)) - J(\mathbf{p}_i(t)) \\
&\geq J(\mathbf{p}_i^{(h-1)}(t)) - J(\hat{\mathbf{p}}_i(t)) \qquad (\because \text{ Lemma 4.3}) \\
&\geq J_{err}(\mathbf{p}_i^{(h-1)}(t)) - J_{err}(\hat{\mathbf{p}}_i(t)) \qquad (\because \text{ (4.21)}) \\
&= w_{err}\|\mathbf{c}_{i,M-l,n}^{(h-1)} - \mathbf{g}_i\| > 0
\end{aligned} \tag{4.30}$$

Therefore, the cost of the trajectory is strictly decreasing until $\mathbf{c}_{i,1,n}^{(h-1)} = \mathbf{p}_i(T_h) = \mathbf{g}_i$. It implies that the agent converges to the subgoal. This concludes the proof. □

Lemma 4.5 shows that the subgoal converges to the waypoint if there is no dynamic obstacle and the grid size is larger than $2\sqrt{2}r$.

Lemma 4.5. *If there is no dynamic obstacle and $d > 2\sqrt{2}r$, then the subgoal converges to the waypoint.*

Proof. The subgoal optimization problem (3.61) can be reformulated as follows:

$$\begin{aligned}
\text{minimize} \quad & \delta \\
\text{subject to} \quad & \delta \in [0,1] \\
& \mathbf{w}_i + \delta(\mathbf{g}_i^{(h-1)} - \mathbf{w}_i) \in \mathcal{S}_M \\
& \mathbf{w}_i + \delta(\mathbf{g}_i^{(h-1)} - \mathbf{w}_i) \in \mathcal{L}_{M,n}^{i,j} \quad \forall j \in \mathcal{N}_i
\end{aligned} \tag{4.31}$$

where δ is the variable such that $\mathbf{g}_i = \mathbf{w}_i + \delta(\mathbf{g}_i^{(h-1)} - \mathbf{w}_i)$. Since $\mathcal{S}_M$ is a convex polyhedron, it can be represented as the intersection of linear constraints $a_s\delta - b_s \leq 0$. Hence the Lagrangian function of the subgoal optimization problem (4.31) of the agent i is given as:

$$\begin{aligned}
L = \delta &- \lambda_0\delta + \lambda_1(\delta - 1) \\
&+ \sum_s \lambda_s(a_s\delta - b_s) \\
&+ \sum_{j\in\mathcal{N}_i} \lambda_{i,j}(d_{M,k}^{i,j} - (\mathbf{w}_i + \delta(\mathbf{g}_i^{(h-1)} - \mathbf{w}_i) - \mathbf{p}_{cls}^{j,i}) \cdot \mathbf{n}_M^{i,j})
\end{aligned} \tag{4.32}$$

where λ_0, λ_1, λ_s, and $\lambda_{i,j}$ are the Lagrangian multipliers.

Assume that there is a non-empty set of the agents $\mathcal{D} \subset \mathcal{I}_a$ whose subgoals do not converge to the waypoint. By Lemma 4.4, all agents reach their subgoals after some replanning step, and this replanning step will be denoted as h_0. If $h > h_0$, $\delta^* = 1$ is the optimal solution of the subgoal optimization problem because the subgoal of the agent i does not converge to the waypoint. Therefore, $\lambda_0 = 0$ by the complementary slackness condition of KKT conditions [53]. Moreover, if $h > h_0$, there exists a grid edge $e \in \mathcal{E}$ such that $\text{Conv}(\{\hat{\mathbf{c}}_{i,M,n}, \mathbf{g}_i^{(h-1)}, \mathbf{w}_i\}) = [\mathbf{g}_i^{(h-1)}, \mathbf{w}_i] \in e$ for the agent $i \in \mathcal{D}$ due to (i) of Lemma 3.8. Since the grid edge does not collide with static obstacles, the condition (3.29) is satisfied, so $[\mathbf{g}_i^{(h-1)}, \mathbf{w}_i] \subset \mathcal{S}_M$ by (3.27). It implies that the agent i always satisfies the SFC constraint regardless of $\delta \in [0, 1]$. Thus, $\lambda_{\forall s} = 0$ by the complementary slackness condition of KKT conditions. As a result, the Lagrangian function for the agent $i \in \mathcal{D}$ when $h > h_0$ can be simplified as follows:

$$\begin{aligned} L = &\delta + \lambda_1(\delta - 1) \\ &+ \sum_{j \in \mathcal{N}_i} \lambda_{i,j}(d_{M,k}^{i,j} - (\mathbf{w}_i + \delta(\mathbf{g}_i^{(h-1)} - \mathbf{w}_i) - \mathbf{p}_{cls}^{j,i}) \cdot \mathbf{n}_M^{i,j}) \end{aligned} \tag{4.33}$$

By the stationary condition of KKT conditions:

$$\frac{\partial L}{\partial \delta} = 1 + \lambda_1 - \sum_{j \in \mathcal{N}_i} \lambda_{i,j}(\mathbf{g}_i^{(h-1)} - \mathbf{w}_i) \cdot \mathbf{n}_M^{i,j} = 0 \tag{4.34}$$

Since the agent converges to the subgoal after the replanning step h_0, $\mathbf{p}_{cls}^{j,i} = \mathbf{g}_j$, $\mathbf{p}_{cls}^{i,j} = \mathbf{g}_i$ and $\mathbf{g}_i^{(h-1)} = \mathbf{g}_i$. Hence the above equation can be simplified as follows:

$$\frac{\partial L}{\partial \delta} = 1 + \lambda_1 - \sum_{j \in \mathcal{N}_i} \lambda_{i,j} \frac{(\mathbf{g}_i - \mathbf{w}_i)^T(\mathbf{g}_i - \mathbf{g}_j)}{\|\mathbf{g}_i - \mathbf{g}_j\|} = 0 \tag{4.35}$$

$$\sum_{j \in \mathcal{N}_i} \lambda_{i,j} \frac{(\mathbf{g}_j - \mathbf{g}_i)^T(\mathbf{w}_i - \mathbf{g}_i)}{\|\mathbf{g}_i - \mathbf{g}_j\|} = 1 + \lambda_1 \tag{4.36}$$

To fulfill the above condition, there must exist an agent $j \in \mathcal{N}_i$ that satisfies $\lambda_{i,j} > 0$ and $(\mathbf{g}_j - \mathbf{g}_i)^T(\mathbf{w}_i - \mathbf{g}_i) > 0$ because $\lambda_1 \geq 0$ and $\lambda_{i,j} \geq 0$ by the dual feasibility of KKT conditions. Since $\lambda_{i,j} > 0$, the agent j must satisfy the following due to the complementary slackness of KKT conditions:

$$d_{M,k}^{i,j} - (\mathbf{w}_i + (\mathbf{g}_i^{(h-1)} - \mathbf{w}_i) - \mathbf{p}_{cls}^{j,i}) \cdot \mathbf{n}_M^{i,j} = 0 \tag{4.37}$$

$$r + \frac{1}{2}\|\mathbf{g}_i - \mathbf{g}_j\| - (\mathbf{g}_i^{(h-1)} - \mathbf{p}_{cls}^{j,i}) \cdot \frac{\mathbf{g}_i - \mathbf{g}_j}{\|\mathbf{g}_i - \mathbf{g}_j\|} = 0 \tag{4.38}$$

$$\|\mathbf{g}_i - \mathbf{g}_j\| = 2r \tag{4.39}$$

To summarize, if there is a non-empty agent set D whose subgoals do not converge to the waypoint, then there must exist an agent j that satisfies the following conditions for each agent $i \in \mathcal{D}$:

$$(\mathbf{g}_j - \mathbf{g}_i)^T(\mathbf{w}_i - \mathbf{g}_i) > 0 \tag{4.40}$$

$$\|\mathbf{g}_i - \mathbf{g}_j\| = 2r \tag{4.41}$$

Let us define the agent $B(i) \in \mathcal{I}$ that satisfies (4.40) and (4.41) to a *blocking agent* of the agent i, where $B(\cdot)$ indicates the blocking agent of the input. Suppose that $B(i) \notin \mathcal{D}$ when $i \in \mathcal{D}$. Then, the agents i and $B(i)$ must be on the same grid edge after the replanning step h_0, as shown in Fig. 4.1a. It is because $B(i)$ converges to its waypoint and the distance between two agents is $2r$ by (4.41). However, the waypoints of two agents must be different due to Lemma 3.1, so $B(i)$ cannot satisfy (4.40). Thus, $B(i) \in \mathcal{D}$ for $\forall i \in \mathcal{D}$.

Since the assumption that $\mathcal{D}$ is not an empty set, there exists an agent i in $\mathcal{D}$ that satisfies the following:

$$\Delta(i) \geq \Delta(j), \forall j \in \mathcal{D} \tag{4.42}$$

where $\Delta(i) = \|\mathbf{w}_i - \mathbf{g}_i\|$. As discussed earlier, the agent i has its blocking agent $B(i) \in \mathcal{D}$, and the agents i and $B(i)$ are on different grid edges due to (iv) of Lemma 3.8. Therefore,

$\Delta(B(i))$ is computed as follows:

$$\Delta(B(i)) = \begin{cases} d - 2r + \Delta(i), & \text{if } \mathbf{n}_{dir} > 0 \\ d - \sqrt{4r^2 - \Delta(i)^2}, & \text{else} \end{cases} \tag{4.43}$$

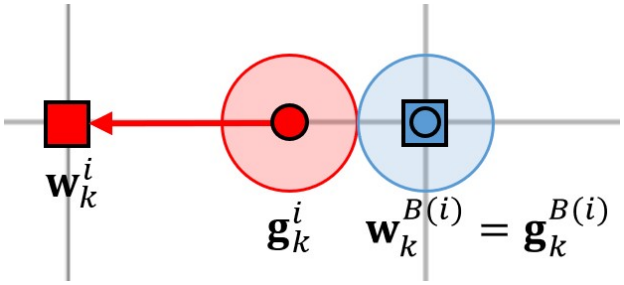
$$\mathbf{n}_{dir} = (\mathbf{w}_{B(i)} - \mathbf{g}_{B(i)})^T (\mathbf{w}_i - \mathbf{g}_i) \tag{4.44}$$

Fig. 4.1b and 4.1c illustrate the derivation process of the above equations. Note that $\Delta(B(i)) > \Delta(i)$ holds since the grid size is $d > 2\sqrt{2}r$. Therefore, this contradicts the assumption that the agent i satisfies (4.42). Fig. 4.2 shows the blocking agents by the grid size. If the grid size is $d = 2\sqrt{2}r$, the agents can be blocking agents of other agents, as shown in the left figure of Fig. 4.2. However, if the grid size holds $d > 2\sqrt{2}r$, at least one agent has no blocking agent as the blue agent in Fig. 4.2. Thus, there is no non-empty agent set $\mathcal{D}$ whose subgoals do not converge to the waypoint. This concludes the proof. □

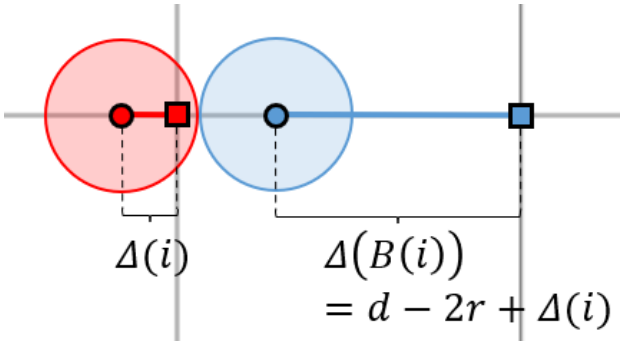
Using Lemmas 4.4 and 4.5, Theorem 4.5 demonstrates that the proposed algorithm guarantees goal convergence.

Theorem 4.5. *Assume that there is no dynamic obstacle, the mission is solvable for the grid-based MAPP, and the communication range r_c is large enough that all agents can communicate with each other. Then, the agent converges to the desired goal.*

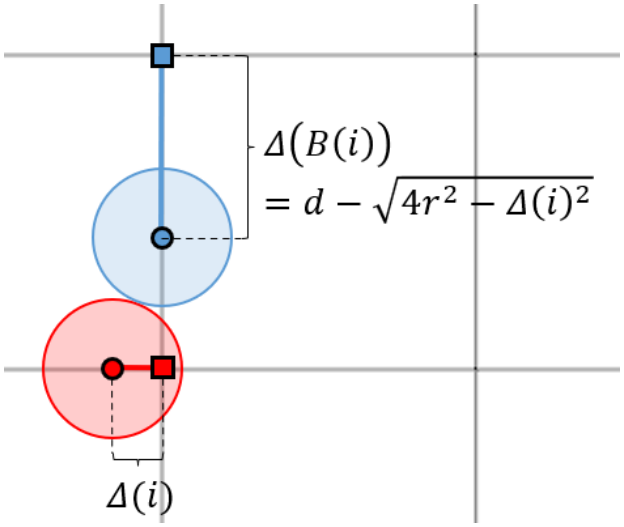
Proof. Due to Lemmas 4.4 and 4.5, the agent converges to the waypoint. Therefore, it is enough to show that the proposed algorithm allows the waypoint to reach the desired goal. By the definition of the modified previous path (3.16), the makespan of the modified previous path $\hat{\boldsymbol{\pi}}_j$ is equal to or less than the previous discrete path $\boldsymbol{\pi}_j^{(h-1)}$. Therefore, the makespan of the discrete path is monotonically decreasing since the discrete path from the MAPP is discarded if the makespan of the path is not smaller than the makespan of $\hat{\boldsymbol{\pi}}_j$ (See lines 5-7 of Alg. 2). Assume that the makespan of the discrete path does not decrease so that all agents cannot proceed further after the replanning step h_0. Then, there exists an agent i that satisfies $\mathbf{w}_i^{(h-1)} \neq \boldsymbol{\pi}_{i,2}^{(h-1)}$ for the replanning step $h > h_0$. However, the agent i



(a) The position of agents when $B(i) \notin \mathcal{D}$ and $i \in \mathcal{D}$



(b) $\mathbf{n}_{dir} > 0$



(c) $\mathbf{n}_{dir} \leq 0$

Figure 4.1: Illustrations for the proof of Lemma 4.5. The square dots are the waypoints and the circle dots are the subgoals. The circles denote the agent's current position.

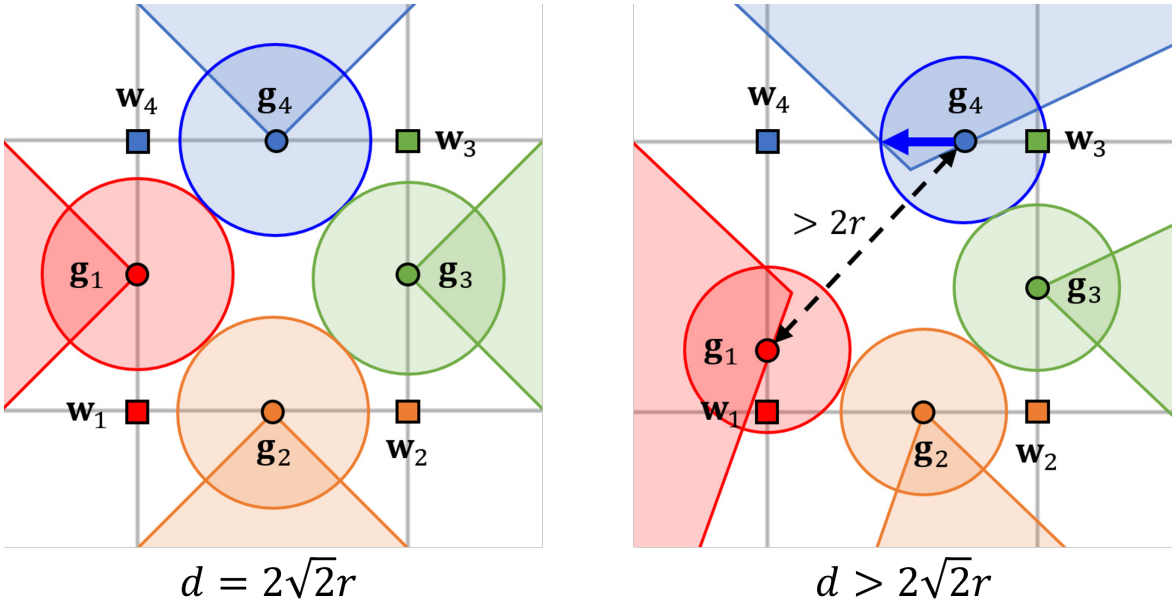


Figure 4.2: Blocking agents by the grid size. The color-shaded region denotes the feasible region of the agent.

has to update its waypoint to $\boldsymbol{\pi}_{i,2}^{(h-1)}$ because it will satisfy the waypoint update rules (3.17) and (3.18) due to Lemmas 4.4 and 4.5. Thus, the makespan of the discrete path decreases until all waypoints reach their desired goals. This concludes the proof. □

Remark 4.1. *If not all agents are connected to the same network, the makespan of the discrete path can increase when a new agent is connected to the network. Therefore, livelock may occur depending on the environment. However, if the grid-based MAPP algorithm is deadlock-free, the proposed algorithm also guarantees deadlock-free due to Lemmas 4.4 and 4.5.*

5

Experimental Validation

This section presents the simulation and experiment results. The agent is modeled with radius $r = 0.15$ m, maximum velocity $v_{max} = 1.0$ m/s, maximum acceleration $a_{max} = 2.0$ m/s^2 based on the experimental result with Crazyflie 2.1. The degree of polynomials of the trajectory is $n = 5$, the number of segments is $M = 10$, and the segment time is $\Delta t = 0.2$ s. Therefore, the total planning horizon is 2 s. The replanning period is assigned to be $\Delta t = 0.2$ s to satisfy the assumption in Thm. 4.3, so the trajectories are updated with the rate of 5 Hz at the same time. For decentralized MAPP, PIBT was implemented based on the source code of [56]. The grid size is $d = 0.5$ m to fulfill the assumption that $d > 2\sqrt{2}r$. The Octomap library [57] is used to represent the obstacles, and the Gilbert–Johnson–Keerthi (GJK) algorithm was implemented using the OpenGJK package [58]. The mazes used in the simulation are generated by randomized Prim's algorithm [59]. The parameters of the cost function are $w_m = 1$, $w_{der} = 0.01$, $w_{slk} = 100$, and the CPLEX solver [60] was used for subgoal and trajectory optimization. The simulation was executed on a laptop with Intel Core i7-9750H @ 2.60GHz CPU and 32G RAM.

5.1 Simulation in Obstacle-free Space

To verify the scalability of the proposed algorithm, I compare the following online trajectory planning algorithms in an obstacle-free space:

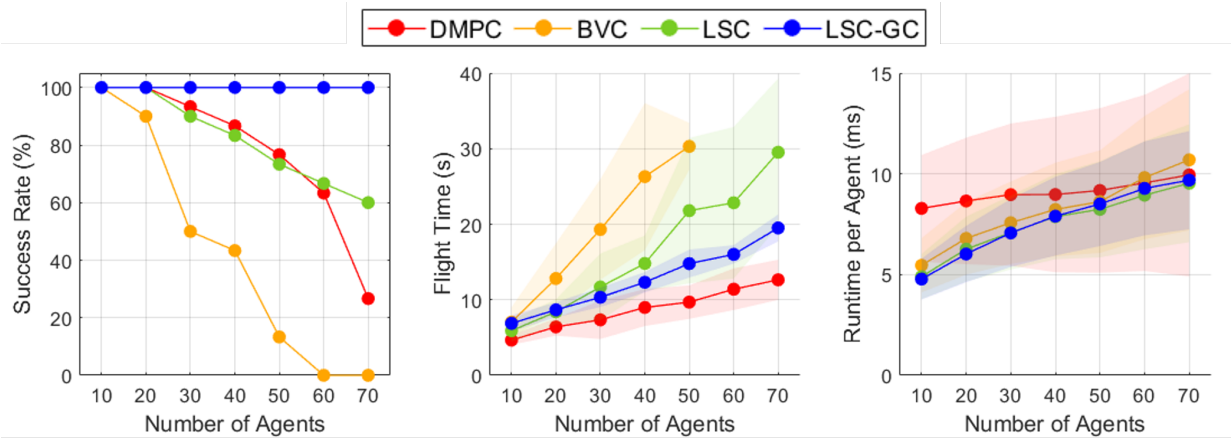
- Distributed Model Predictive Control [9] (DMPC)
- Buffered Voronoi Cell approach [2] (BVC)
- Linear Safe Corridor approach [1] (LSC)
- Linear Safe Corridor with Goal Convergence (LSC-GC, proposed algorithm)

The simulation is conducted with 10 to 70 agents in a 3 m $\times$ 3 m $\times$ 2 m space. For each number of agents, 30 trials were conducted to measure the success rate, and the start and goal points were randomly deployed for each mission. The test is judged to be a failure when the agents collided with each other or could not reach the goal point within 60 s. In this simulation, the degree of polynomials of the trajectory is $n = 5$, the number of segments is $M = 10$, and the segment time is $\Delta t = 0.2$ s to match the total planning horizon to 1 s. The communication range of the LSC-GC is 3 m. For DMPC, a smaller model with $r = 0.1$ m and $D_{i,j} = \mathrm{diag}([1, 1, 1/2.25])$ is used when judging the collision because DMPC uses soft constraints for collision avoidance. DMPC was tested in MATLAB R2020a, and the other methods were implemented in C++, Ubuntu 18.04.

Fig. 5.1 and Table 5.1 describe the simulation result in the obstacle-free space. DMPC shows good scalability with respect to computation time, but the collision occurs for all failure cases even though it uses a smaller collision model when judging the collision. BVC and LSC-based approaches do not cause the collision, but deadlock or livelock occurs as the number of agents increases. On the contrary, the proposed method shows the perfect success rate in the identical setting since it guarantees goal convergence. Moreover, The proposed method has a 44.7% shorter flight time than BVC and 23% shorter flight time than the LSC-based approach. It is because other algorithms trigger deadlock resolution

Table 5.1: Performance comparison in an obstacle-free space. The flight time is the averaged value from successful trials among 30 different simulations. The bold number indicates the best result.

Metric	Method	The number of agents						
		10	20	30	40	50	60	70
Success rate (%)	DMPC	100	100	93.3	86.7	76.7	63.3	26.7
	BVC	100	90	50	43.3	13.3	0	0
	LSC	100	100	90	83.3	73.3	66.7	60
	LSC-GC	**100**	**100**	**100**	**100**	**100**	**100**	**100**
Flight time (s)	DMPC	**4.66**	**6.41**	**7.35**	**8.98**	**9.69**	**11.4**	**12.6**
	BVC	7.05	12.8	19.3	26.3	30.3	-	-
	LSC	5.90	8.40	11.7	14.8	21.8	22.8	29.5
	LSC-GC	6.88	8.67	10.3	12.3	14.8	16.0	19.5
Runtime per agent (ms)	DMPC	8.28	8.65	8.97	8.97	9.18	9.55	9.95
	BVC	5.46	6.80	7.57	8.23	8.62	9.81	10.7
	LSC	4.90	6.26	7.08	**7.87**	**8.23**	**8.94**	**9.53**
	LSC-GC	**4.78**	**6.02**	**7.07**	7.90	8.50	9.28	9.69



(a) Success rate (b) Total flight time (c) Computation time

Figure 5.1: Simulation results in an obstacle-free space. We averaged the value from success cases among 30 different trials. The shaded region means the standard deviation interval (shown best in color).

when the deadlock is detected. As a result, the agents may be clustered until the deadlock is detected. On the other hand, the proposed algorithm prevents deadlock preemptively by placing the subgoal in a feasible region. Therefore, the agents can reach the desired goal without unnecessary movement. The proposed algorithm requires a similar computation time as the LSC-based approach. The proposed algorithm takes 9.69 ms per agent for 70 agents, which implies that it shows good scalability enough to achieve online replanning.

5.2 Simulation in 2D Obstacle Space

In this section, the LSC-based approach [1] and the proposed algorithm are compared in the following 2D obstacle environments:

(i) Random forest. 40 static obstacles are deployed in a random position and ten agents are deployed in a circle with a 4 m radius. The goal point of the agent is at the antipodal point of the start point, as shown in Fig. 5.2.

(ii) Sparse maze. It consists of 6 × 6 cells, and each cell size is 1.0 m × 1.0 m, thus three agents can pass the corridor simultaneously. The maze has two entrances, and there

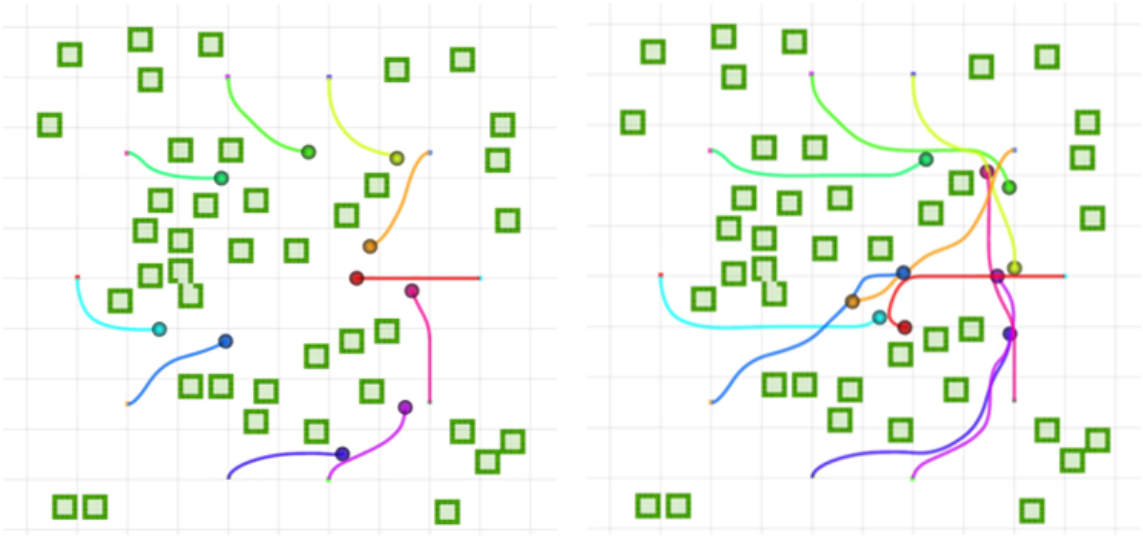
are five agents at each entrance. Each agent's goal point is assigned to the entrance on the other side of the maze, as depicted in Fig. 5.3.

(iii) Dense maze. It consists of 9 $\times$ 9 cells, and each cell size is 0.5 m $\times$ 0.5 m, thus only one agent can pass the corridor. Each agent's goal point is assigned to the entrance on the other side of the maze, as illustrated in Fig. 5.4.

The mission is judged to be a failure when a collision occurred or when the agent failed to reach the goal within 60 s. For each map, 30 trials were executed changing the obstacle's position. For LSC and LSC-GC, the degree of polynomials of the trajectory is $n = 5$, the number of segments is $M = 10$, and the segment time is $\Delta t = 0.2$ s to match the total planning horizon to 2 s.

Table 5.2 describes the simulation results in obstacle environments. The LSC-based approach shows the perfect success rate in sparse environments and does not cause a collision in all cases. However, it fails to reach the goal in the dense maze because it cannot solve a deadlock when there is no space to yield to a higher-priority agent. On the other hand, the proposed algorithm achieves the perfect success rate for all types of environments regardless of the communication range. It validates that the proposed algorithm can solve a deadlock even in a dense maze-like environment without a centralized coordinator.

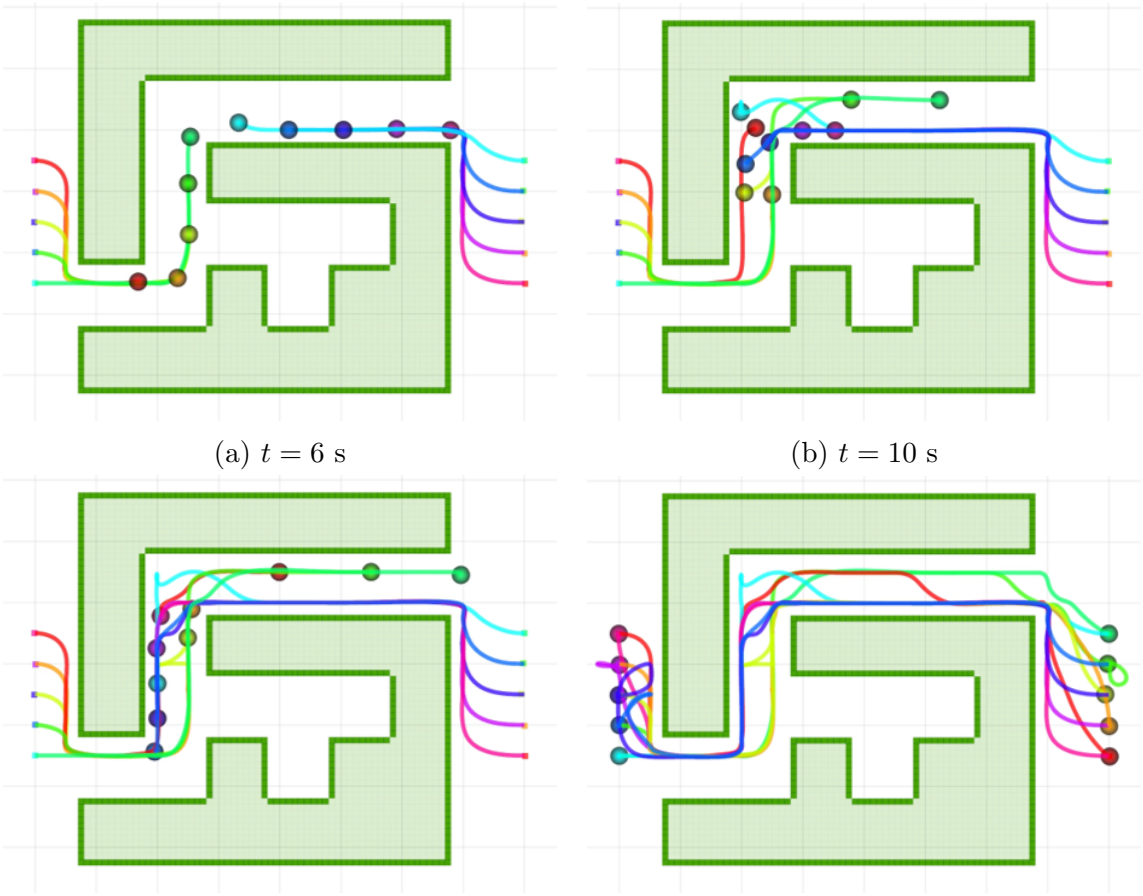
The proposed method shows a 27.6% shorter flight time and a 7.4% shorter flight distance compared to LSC-PB when the communication range is $r_c = \infty$. It is because the LSC-based approach performs a deadlock resolution only when the distance between the agents is close enough. On the contrary, the proposed algorithm utilizes the final trajectory point, not the current position, for deadlock resolution. Therefore, the agent does not need to wait until other agents clump together. In addition, the flight time and distance of the proposed algorithm reduce as the communication range increases since the agent can update the waypoint further away from the current position by the waypoint update rule (3.18).



(a) $t = 3$ s (b) $t = 7$ s

(c) $t = 11$ s (d) $t = 17$ s

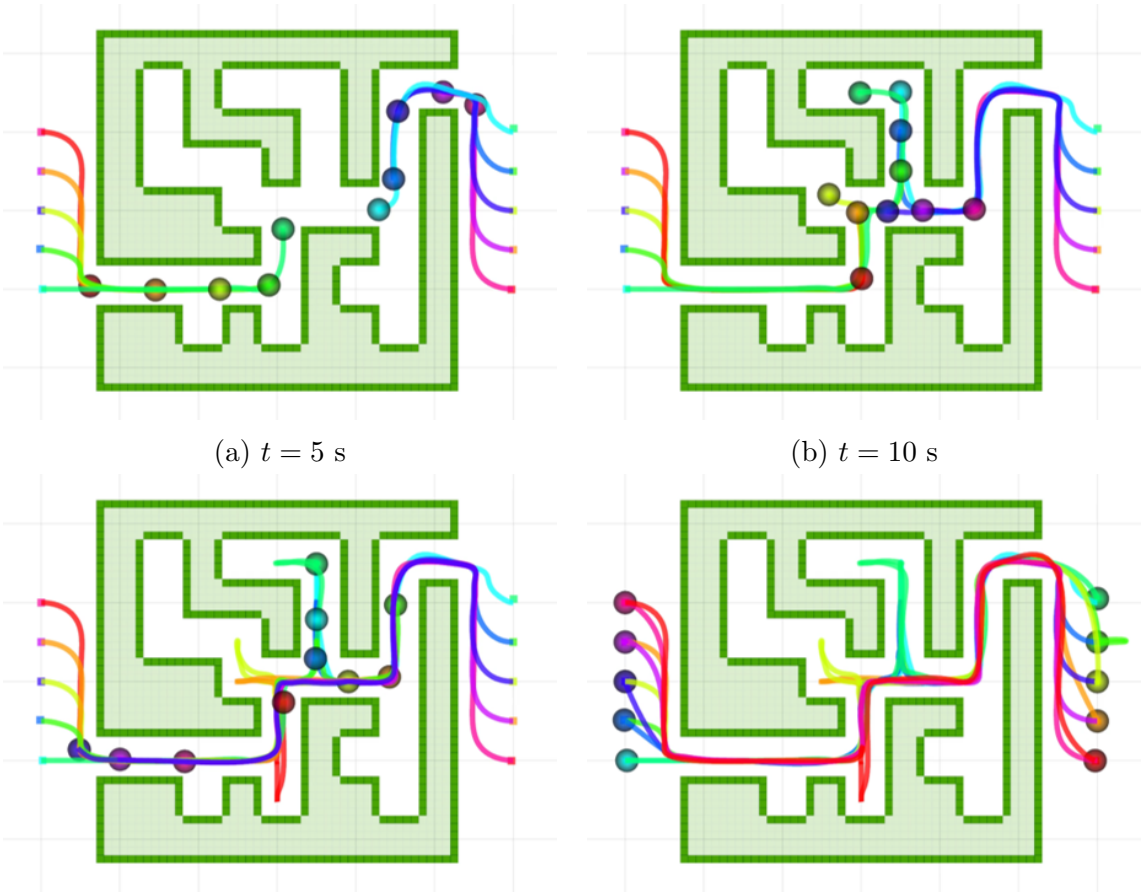
Figure 5.2: Trajectory generation result of the proposed method in the 2D random maze ($r_c = 3$ m). The circle and line are the agent at its final location and its trajectory respectively, and the green-shaded region is the static obstacle.



(a) $t = 6$ s (b) $t = 10$ s

(c) $t = 14$ s (d) $t = 24$ s

Figure 5.3: Trajectory generation result of the proposed method in the 2D sparse maze ($r_c = 3$ m). The circle and line are the agent at its final location and its trajectory respectively, and the green-shaded region is the static obstacle.



(a) $t = 5$ s (b) $t = 10$ s

(c) $t = 15$ s (d) $t = 25$ s

Figure 5.4: Trajectory generation result of the proposed method in the 2D dense maze ($r_c = 3$ m). The circle and line are the agent at its final location and its trajectory respectively, and the green-shaded region is the static obstacle.

Table 5.2: Comparison with previous work [1]. The bold number indicates the best result (sr: success rate (%), T_f: flight time (s), L: flight distance per agent (m), T_c: computation time (ms)).

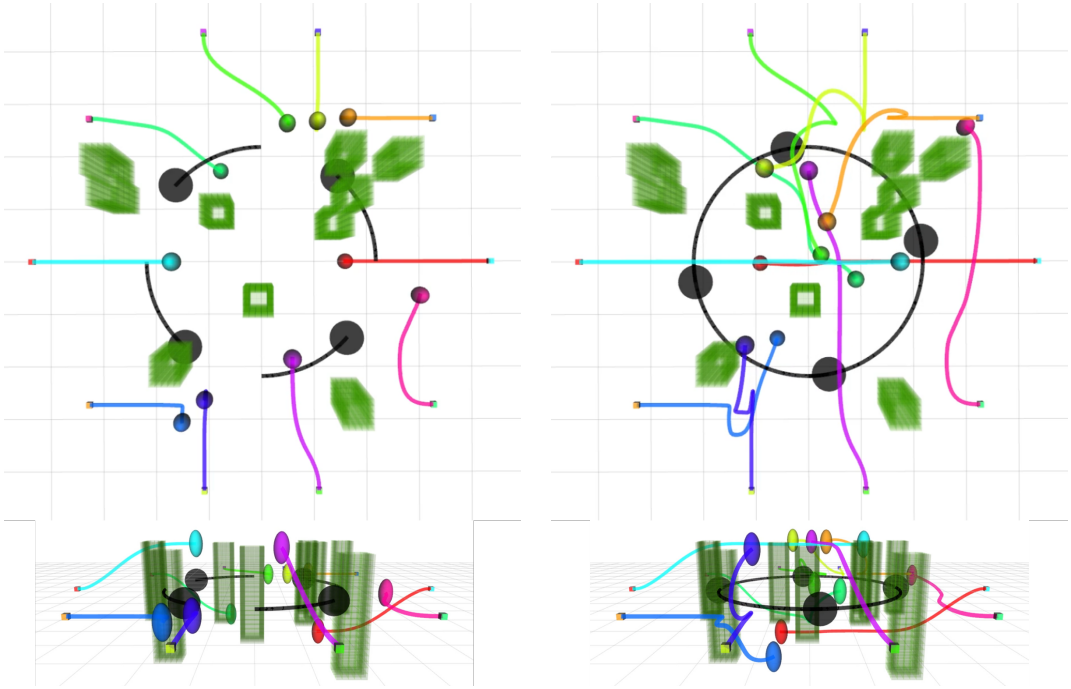
Env.	Method	sr	T_f	L	T_c
Random forest	LSC [1] ($r_c = \infty$)	100	25.7	11.8	8.15
	LSC-GC ($r_c = 2$ m)	100	28.8	11.7	**7.86**
	LSC-GC ($r_c = 3$ m)	100	20.7	11.3	8.01
	LSC-GC ($r_c = 4$ m)	100	19.9	11.3	8.23
	LSC-GC ($r_c = \infty$)	100	**19.1**	**11.1**	8.16
Sparse maze	LSC [1] ($r_c = \infty$)	100	33.7	13.9	9.05
	LSC-GC ($r_c = 2$ m)	100	34.4	13.5	**8.51**
	LSC-GC ($r_c = 3$ m)	100	27.1	13.1	8.62
	LSC-GC ($r_c = 4$ m)	100	**23.7**	**12.6**	8.66
	LSC-GC ($r_c = \infty$)	100	23.9	12.7	8.67
Dense maze	LSC [1] ($r_c = \infty$)	0	-	-	-
	LSC-GC ($r_c = 2$ m)	100	61.4	**16.5**	7.30
	LSC-GC ($r_c = 3$ m)	100	51.0	16.6	7.44
	LSC-GC ($r_c = 4$ m)	100	50.9	17.1	7.31
	LSC-GC ($r_c = \infty$)	100	**48.3**	16.7	**7.24**

5.3 Simulation in 3D Obstacle Space

In this section, the following state-of-the-art algorithms are compared in a 3D cluttered environment:

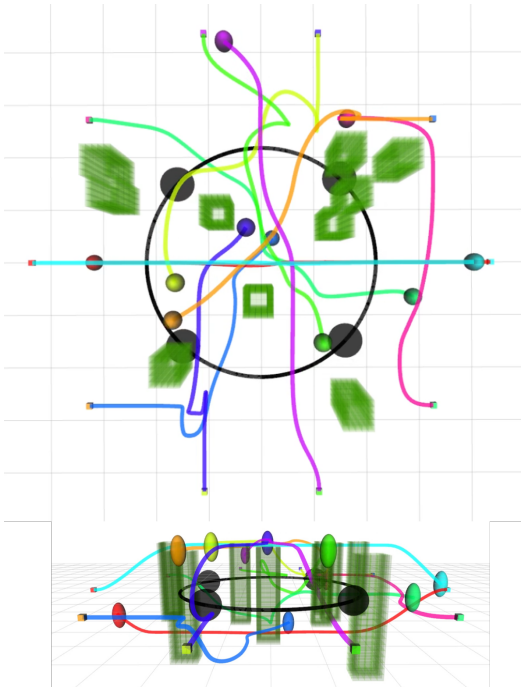
- EGO-Swarm [5]
- MADER [6]
- Linear Safe Corridor approach [1] (LSC)
- Linear Safe Corridor with Goal Convergence (LSC-GC, proposed algorithm)

The simulation was performed with ten agents in two types of environments: random forest and maze.

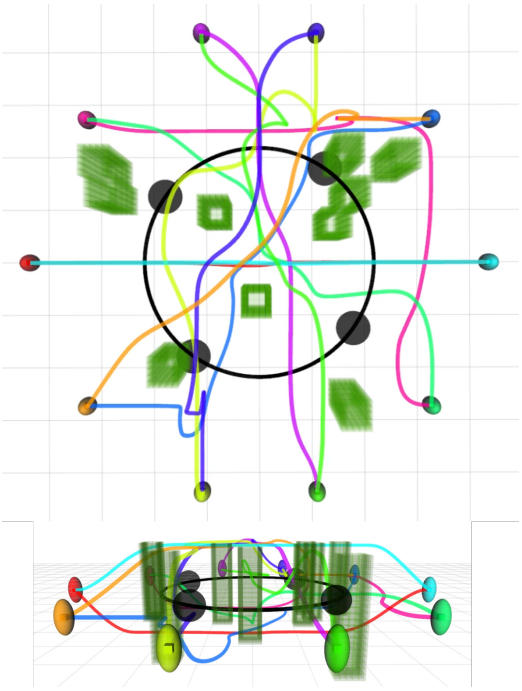


(a) $t = 3$ s

(b) $t = 8$ s

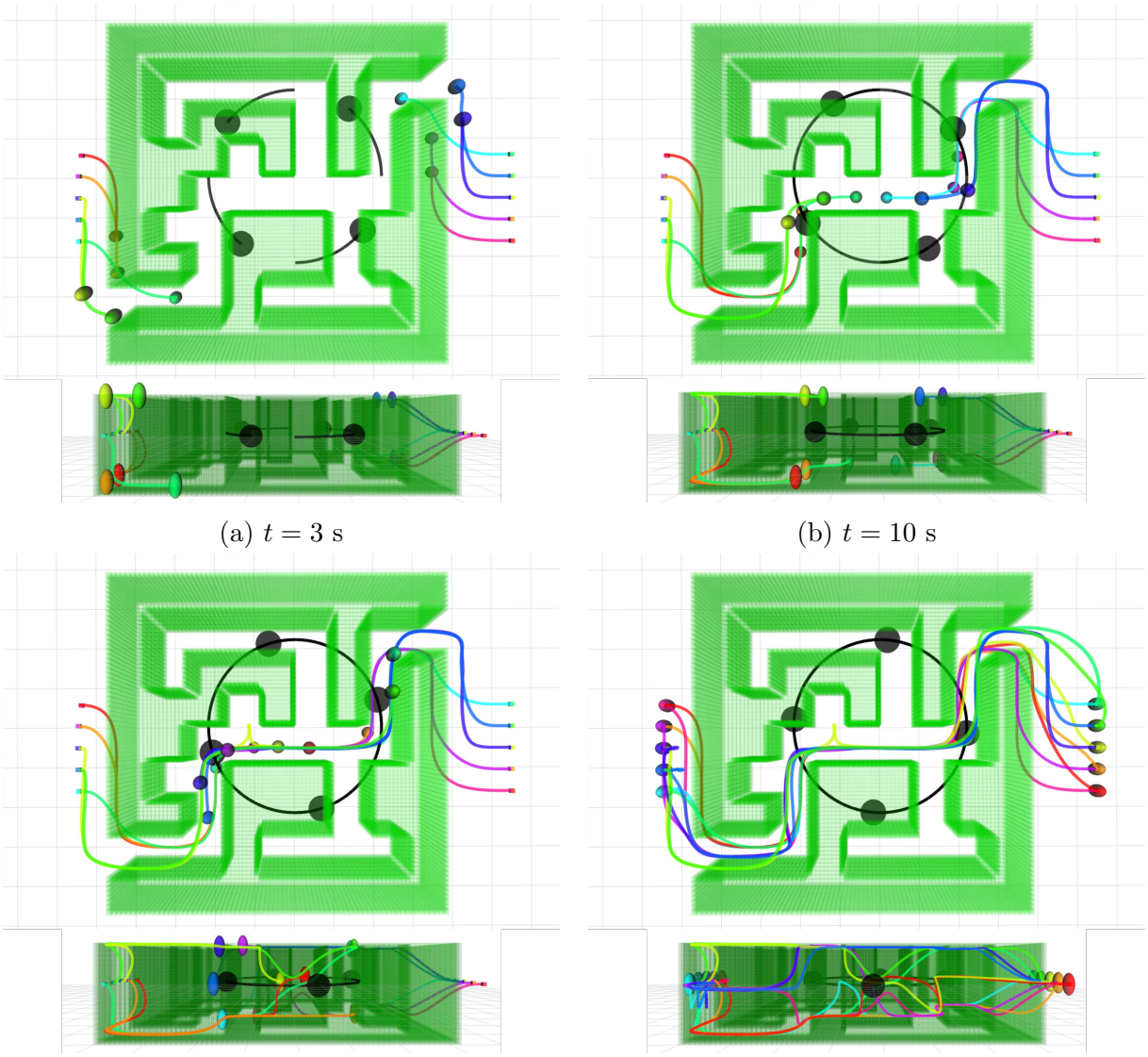


(c) $t = 11$ s



(d) $t = 18$ s

Figure 5.5: Trajectory planning result of the proposed method with 10 agents in the 3D random forest.



(a) $t = 3$ s

(b) $t = 10$ s

(c) $t = 16$ s

(d) $t = 24$ s

Figure 5.6: Trajectory planning result of the proposed method with 10 agents in the 3D maze.

Table 5.3: Simulation result with 10 agents in cluttered environments. The flight time is the averaged value from successful trials among 30 different simulations. The best result is highlighted in bold. Note that the exact obstacles' trajectories are provided to MADER, while only the obstacle's position and velocity are provided to the LSC-GC.

Environment	Metric	Method	Velocity of dynamic obstacles			
			No dyn. obs.	0.5 m/s	1.0 m/s	2.0 m/s
Forest	Success rate (%)	EGO-Swarm	90	-	-	-
		MADER	43.3	20	13.3	0
		LSC	100	-	-	-
		LSC-GC	**100**	**100**	**90**	**30**
	Flight time (s)	EGO-Swarm	17.3	-	-	-
		MADER	19.5	32.3	29.5	-
		LSC	16.1	-	-	-
		LSC-GC	**13.4**	**14.1**	**14.4**	**18.7**
	Runtime per agent (ms)	EGO-Swarm	**2.7**	-	-	-
		MADER	225.8	274.7	272.5	282.6
		LSC	7.3	-	-	-
		LSC-GC	12.9	**14.4**	**14.2**	**13.8**
Maze	Success rate (%)	EGO-Swarm	0	-	-	-
		MADER	0	0	0	0
		LSC	0	-	-	-
		LSC-GC	**100**	**100**	**93.3**	**66.7**
	Flight time (s)	EGO-Swarm	-	-	-	-
		MADER	-	-	-	-
		LSC	37.7	-	-	-
		LSC-GC	**21.9**	**28.7**	**29.6**	**31.8**
	Runtime per agent (ms)	EGO-Swarm	7.7	-	-	-
		MADER	279.0	296.9	287.2	283.6
		LSC	**4.7**	-	-	-
		LSC-GC	11.3	**12.3**	**12.2**	**12.7**

Table 5.4: Simulation result of LSC-GC with different maximum accelerations of dynamic obstacles. The flight time is the averaged value from successful trials among 30 different simulations. The best result is highlighted in bold.

Environment	Metric	Max. acc. of obs. (m/s^2)	Velocity of dynamic obstacles		
			0.5 m/s	1.0 m/s	2.0 m/s
Forest	Success rate (%)	0	70	3.3	13.3
		1	96.7	80	6.7
		2	**100**	**90**	**30**
		3	100	83.3	13.3
	Flight time (s)	0	12.9	**13.2**	**14.1**
		1	**12.8**	13.3	17.2
		2	14.1	14.4	18.7
		3	16.3	26.9	40.6
Maze	Success rate (%)	0	66.7	26.7	33.3
		1	96.7	76.7	43.3
		2	**100**	**93.3**	**66.7**
		3	0	0	0
	Flight time (s)	0	**26.3**	**26.1**	**26.6**
		1	26.9	27.2	28.1
		2	28.7	29.6	31.8
		3	-	-	-

(i) Random forest. The random forest consists of ten trees with the dimension 0.5 m $\times$ 0.5 m $\times$ 2.5 m. A circle swap mission was conducted in the random forest. The agents start at a circle with a 4 m radius and 1 m height, and the goal points are at the antipodes of the start points, as shown in Fig. 5.5.

(ii) Maze. The maze consists of 9 $\times$ 9 cells, and the dimension of each cell is 0.8 m $\times$ 0.8 m $\times$ 2.5 m. The maze was created from randomized Prim's algorithm [59]. The maze has two entrances, and five agents were deployed for each entrance. The mission of the agents is to reach the other side of the maze, as shown in Fig. 5.6.

For MADER and the proposed algorithm, the simulations with four dynamic obstacles were executed. The dynamic obstacles were modeled as a sphere with a radius of 0.15 m, and they rotated in a circle with the 2 m radius and 1 m height, with the maximum acceleration of 2 m/s^2. For a fair comparison, the sensor range of all methods was increased enough to recognize all obstacles. In addition, although it is unfair to the proposed algorithm, the exact trajectories of dynamic obstacles were provided to MADER, while only the obstacle's position and velocity were provided to the proposed algorithm. The communication range of the LSC-GC is 3 m. 30 trials were conducted for each environment, and the test was considered a failure when the collision occurred or the agent failed to reach the goal point within 60 s.

Table 5.3 describes the simulation result in cluttered environments with dynamic obstacles. EGO-Swarm shows the fastest computation speed in the random forest, but there were several cases where the agents collided with each other. It is because EGO-Swarm optimizes the trajectory without hard constraints, so it cannot guarantee inter-collision avoidance. MADER did not cause the collision since it utilizes the trajectories of all agents and obstacles during the planning. However, MADER shows a low success rate even in the random forest because it often fails to find the detour path when trees are placed densely. On the other hand, the proposed algorithm shows the highest success rate and the shortest flight time for all environment configurations. In particular, the proposed algorithm shows

over 90% success rate for both random forest and maze when the velocity of the dynamic obstacle is less than or equal to the agent's maximum velocity. It indicates that the proposed algorithm can avoid moving obstacles in a narrow space without exact knowledge of the obstacle's trajectory.

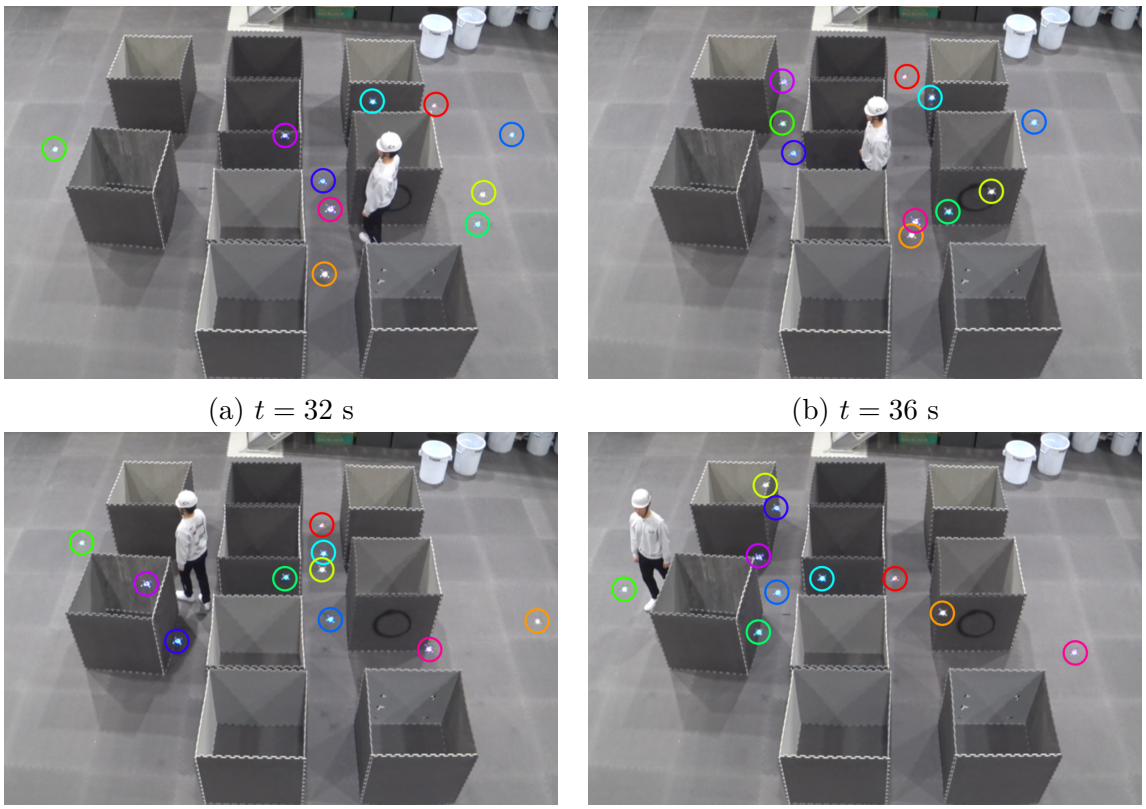
Table 5.4 shows the simulation result of the proposed method with different maximum accelerations of dynamic obstacles. Since the maximum acceleration determines the size of the obstacle's reachable region, it can control the conservatism of the collision constraints. As shown in the table, the success rate decreases as the maximum acceleration decreases since collisions occur more frequently. On the other hand, if the maximum acceleration is too big ($> 2\text{m/s}^2$), the agents can fail to reach the desired goal due to conservative constraints. As a result, the proposed algorithm shows the highest success rate when the obstacle's maximum acceleration is 2 m/s^2.

Figure 5.7: Crazyflie 2.1 quadrotors that were used in the hardware demonstration.

5.4 Hardware demonstration

The hardware demonstration was conducted with ten Crazyflie 2.1 quadrotors and one pedestrian acting as dynamic obstacles in the maze-like environment, as shown in Fig. 5.8. Fig. 5.7 shows the quadrotors used in the experiment. In the demonstration, the agents are patrolling the two waypoints while avoiding the pedestrians, and the maximum acceleration of the pedestrian is set to 2 m/s^2. The human was modeled as an ellipsoidal obstacle with the radius $r_{j\in\mathcal{I}_o} = 0.35$ m and the scaling matrix $D_{i,j} = \mathrm{diag}([1, 1, 1/4])$. The velocity of the pedestrian was estimated using the linear Kalman filter. The proposed algorithm was designed to run onboard each agent in a distributed manner as long as the agent has sufficient computation and communication capability. In the reported experiments, due to the hardware limit of the Crazyflie 2.1 quadrotor, the trajectory for each agent was computed on a single laptop. But it should be noted that the computation was done in a distributed manner. The Crazyswarm [61] was used to broadcast the trajectory to the agents, and the Optitrack motion capture system is utilized to measure the agent's position.

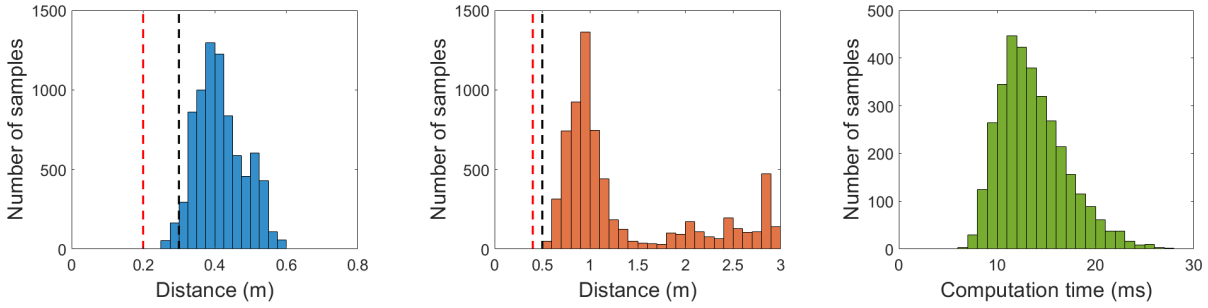
Fig. 5.9a and Fig. 5.9b describe the histogram of the minimum distance between agents and minimum distance to dynamic obstacles, respectively. As shown in Fig. 5.9a, the inter-agent distance invaded the desired safe distance for a short period due to a tracking error of the Crazyflie controller, but it did not lead to an actual collision between agents. Furthermore, the agents did not collide with the pedestrian during the hardware demonstration, as shown in Fig. 5.9b. Fig. 5.9c illustrates the computation time per agent during the experiment. The average computation time was 13.7 ms, and there was no timeout case during the flight.



(a) $t = 32$ s (b) $t = 36$ s

(c) $t = 39$ s (d) $t = 42$ s

Figure 5.8: Snapshots of the experiment with 10 quadrotors and one pedestrian in the maze-like environment. The colored circle denotes the position of the quadrotor.



(a) Minimum distance between agents (b) Minimum distance to pedestrian (c) Computation time per agent

Figure 5.9: Summary of the experiment in the maze-like environment. The red and black dashed lines denote the physical and desired safe distance, respectively.

6

Conclusion

This dissertation presented the online decentralized MATP algorithm that guarantees to generate collision avoidance, the feasibility of the constraints, and goal convergence in a cluttered environment. The decentralized MAPP is utilized for deadlock resolution, and the linear safe corridor (LSC) and relative safe flight corridor (RSFC) are utilized to construct feasible constraints considering the reachable region of dynamic obstacles. The dissertation demonstrates that the proposed algorithm guarantees the feasibility of the optimization problem, collision avoidance, and goal convergence for every replanning step.

The simulation result shows that the proposed method does not cause collision or deadlock in static environments, regardless of the density of the obstacles or communication range. In obstacle-free space, the proposed method can compute the trajectories for 70 agents on average 9.69 ms per agent with an Intel i7 laptop. The proposed algorithm has a 44.7% shorter flight time than the buffered Voronoi cell-based approach (BVC) and 23% shorter flight time than our previous work. Moreover, the proposed algorithm shows the highest success rate and shorter flight time in dynamic environments compared to state-of-the-art baselines: EGO-Swarm and MADER. In particular, the proposed algorithm shows

over 90% success rate when the velocity of moving obstacles is below the agent's maximum speed. The hardware demonstration was conducted with ten Crazyflie quadrotors and one pedestrian to validate the safety and operability of the proposed algorithm, and there was no collision during the flight.

The possible future works of this dissertation will be increasing the robustness. If the tracking or estimation error is larger than the collision model, the proposed algorithm can cause a collision. Also, the prediction error of the dynamic obstacle's trajectory highly affects the overall performance. Therefore, there exists room for the development of state estimation, reachability analysis, and trajectory prediction.

References

[1] 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.

[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] L. Wawrla, O. Maghazei, and T. Netland, "Applications of drones in warehouse operations," *Whitepaper. ETH Zurich, D-MTEC*, 2019.

[4] K. McGuire, C. De Wagter, K. Tuyls, H. Kappen, and G. C. de Croon, "Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment," *Science Robotics*, vol. 4, no. 35, p. eaaw9710, 2019.

[5] 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.

[6] J. Tordesillas and J. P. How, "Mader: Trajectory planner in multiagent and dynamic environments," *IEEE Transactions on Robotics*, 2021.

[7] C. Toumieh and A. Lambert, "Decentralized multi-agent planning using model predictive control and time-aware safe corridors," *IEEE Robotics and Automation Letters*, 2022.

[8] J. Park and H. J. Kim, "Online trajectory planning for multiple quadrotors in dynamic environments using relative safe flight corridor," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 659–666, 2020.

[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] 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.

[11] 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)*. IEEE, 2012, pp. 1917–1922.

[12] 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.

[13] 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.

[14] 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.

[15] A. Rahmani, K. Kosuge, T. Tsukamaki, and M. Mesbahi, "Multiple uav deconfliction via navigation functions," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008, p. 6626.

[16] A. K. Pamosoaji and K.-S. Hong, "A path-planning algorithm using vector potential functions in triangular regions," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 4, pp. 832–842, 2013.

[17] S. H. Semnani, A. H. de Ruiter, and H. H. Liu, "Force-based algorithm for motion planning of large agent," *IEEE Transactions on Cybernetics*, 2020.

[18] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.

[19] 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.

[20] J. Alonso-Mora, T. Naegeli, R. Siegwart, and P. Beardsley, "Collision avoidance for aerial vehicles in multi-agent scenarios," *Autonomous Robots*, vol. 39, no. 1, pp. 101–121, 2015.

[21] C. Y. Tan, S. Huang, K. K. Tan, R. S. H. Teo, W. Q. Liu, and F. Lin, "Collision avoidance design on unmanned aerial vehicle in 3d space," *Unmanned Systems*, vol. 6, no. 04, pp. 277–295, 2018.

[22] S. H. Arul and D. Manocha, "Dcad: Decentralized collision avoidance with dynamics constraints for agile quadrotor swarms," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1191–1198, 2020.

[23] S. Kandhasamy, V. B. Kuppusamy, and S. Krishnan, "Scalable decentralized multi-robot trajectory optimization in continuous-time," *IEEE Access*, vol. 8, pp. 173 308–173 322, 2020.

[24] X. Zhou, Z. Wang, X. Wen, J. Zhu, C. Xu, and F. Gao, "Decentralized spatial-temporal trajectory planning for multicopter swarms," *arXiv preprint arXiv:2106.12481*, 2021.

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

[26] A. Richards and J. P. How, "Model predictive control of vehicle maneuvers with guaranteed completion time and robust feasibility," in *Proceedings of the 2003 American Control Conference, 2003.*, vol. 5. IEEE, 2003, pp. 4034–4040.

[27] A. Richards and J. How, "Decentralized model predictive control of cooperating uavs," in *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601)*, vol. 4. IEEE, 2004, pp. 4286–4291.

[28] P. Wang and B. Ding, "A synthesis approach of distributed model predictive control for homogeneous multi-agent system with collision avoidance," *International Journal of Control*, vol. 87, no. 1, pp. 52–63, 2014.

[29] 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.

[30] E. Soria, F. Schiano, and D. Floreano, "Predictive control of aerial swarms in cluttered environments," *Nature Machine Intelligence*, vol. 3, no. 6, pp. 545–554, 2021.

[31] V. R. Desaraju and J. P. How, "Decentralized path planning for multi-agent teams with complex constraints," *Autonomous Robots*, vol. 32, no. 4, pp. 385–403, 2012.

[32] J. Alonso-Mora, J. A. DeCastro, V. Raman, D. Rus, and H. Kress-Gazit, "Reactive mission and motion planning with deadlock resolution avoiding dynamic obstacles," *Autonomous Robots*, vol. 42, no. 4, pp. 801–824, 2018.

[33] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Towards search-based motion planning for micro aerial vehicles," *arXiv preprint arXiv:1810.03071*, 2018.

[34] M. Kamel, J. Alonso-Mora, R. Siegwart, and J. Nieto, "Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 236–243.

[35] H. Zhu and J. Alonso-Mora, "Chance-constrained collision avoidance for mavs in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 776–783, 2019.

[36] M. Chen, J. C. Shih, and C. J. Tomlin, "Multi-vehicle collision avoidance via hamilton-jacobi reachability and mixed integer programming," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 1695–1700.

[37] M. Chen, S. Bansal, J. F. Fisac, and C. J. Tomlin, "Robust sequential trajectory planning under disturbances and adversarial intruder," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 4, pp. 1566–1582, 2018.

[38] 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.

[39] H. Zhu and J. Alonso-Mora, "B-uavc: Buffered uncertainty-aware voronoi cells for probabilistic multi-robot collision avoidance," in *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2019, pp. 162–168.

[40] M. Abdullhak and A. Vardy, "Deadlock prediction and recovery for distributed collision avoidance with buffered voronoi cells," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 429–436.

[41] M. Jager and B. Nebel, "Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems.*, vol. 3. IEEE, 2001, pp. 1213–1219.

[42] B. Şenbaşlar, W. Hönig, and N. Ayanian, "Robust trajectory execution for multi-robot teams using distributed real-time replanning," in *Distributed Autonomous Robotic Systems*. Springer, 2019, pp. 167–181.

[43] Y. Chen, M. Guo, and Z. Li, "Recursive feasibility and deadlock resolution in mpc-based multi-robot trajectory generation," *arXiv preprint arXiv:2202.06071*, 2022.

[44] J. S. Grover, C. Liu, and K. Sycara, "Deadlock analysis and resolution for multi-robot systems," in *International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2020, pp. 294–312.

[45] S. Dergachev and K. Yakovlev, "Distributed multi-agent navigation based on reciprocal collision avoidance and locally confined multi-agent path finding," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2021, pp. 1489–1494.

[46] J. Hou, X. Zhou, Z. Gan, and F. Gao, "Enhanced decentralized autonomous aerial robot teams with group planning," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9240–9247, 2022.

[47] 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.

[48] R. T. Farouki, "The bernstein polynomial basis: A centennial retrospective," *Computer Aided Geometric Design*, vol. 29, no. 6, pp. 379–419, 2012.

[49] K. Okumura, M. Machida, X. Défago, and Y. Tamura, "Priority inheritance with backtracking for iterative multi-agent path finding," *Artificial Intelligence*, vol. 310, p. 103752, 2022.

[50] M. E. Flores, "Real-time trajectory generation for constrained nonlinear dynamical systems using non-uniform rational b-spline basis functions," Ph.D. dissertation, California Institute of Technology, 2008.

[51] 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)*, 2020, pp. 434–440.

[52] J. Park, I. Jang, and H. J. Kim, "Decentralized deadlock-free trajectory planning for quadrotor swarm in obstacle-rich environments - extended version," 2022. [Online]. Available: https://github.com/qwerty35/lsc_dr_planner

[53] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[54] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.

[55] D. Goldfarb and S. Liu, "An o(n^3l) primal interior point algorithm for convex quadratic programming," *Mathematical programming*, vol. 49, no. 1, pp. 325–340, 1990.

[56] K. Okumura, Y. Tamura, and X. Défago, "Iterative refinement for real-time multi-robot path planning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 9690–9697.

[57] 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.

[58] M. Montanari and N. Petrinic, "Opengjk for c, c# and matlab: Reliable solutions to distance queries between convex bodies in three-dimensional space," *SoftwareX*, vol. 7, pp. 352–355, 2018.

[59] M. Foltin, "Automated maze generation and human interaction," *Brno: Masaryk University Faculty Of Informatics*, 2011.

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

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

국 문 초 록

본 논문에서는 동적 장애물이 있는 미로와 같은 환경에서 쿼드로터 군집을 위한 온라인 분산 다중 로봇 경로 계획 알고리즘을 제안한다. 제안하는 방법은 교착 상태 해소를 위해 에이전트가 목표지점에 수렴할 수 있도록 중간 목표 지점을 최적화하는 방법을 사용하였으며 그리드 기반 다중 에이전트 경로 계획 알고리즘에서 계산되는 경로점들을 사용하여 중간 목표점을 최종 목표점으로 유도하는 방법을 사용한다. 제안하는 알고리즘은 안전 비행 복도 (safe flight corridor)을 사용하여 정적 장애물과의 충돌을 방지하며 선형 안전 복도 (linear safe corridor)를 사용하여 에이전트 간의 충돌을 방지한다. 그 결과, 제안하는 방법은 제한된 통신 범위 아래에서도 정적 장애물 환경에서 충돌 회피, 제한 조건들의 실현 가능성, 목적지 수렴을 보장한다. 또한 본 논문에서는 상대적 안전 비행 복도 (relative safe flight corridor)을 사용하여 장애물의 도달 가능 영역을 우회하는 방법으로 동적 장애물을 회피하는 방법을 제안한다. 그리고 좁은 환경에서는 에이전트가 서로 뭉치는 현상이 자주 발생하게 되므로 우선순위 기반의 다중 에이전트 경로 계획 알고리즘을 사용하여 동적 장애물을 회피할 때 다른 에이전트들이 간섭하는 현상을 방지한다.

제안하는 방법을 검증하기 위해 장애물이 없는 공간, 랜덤 포레스트, 미로 환경에서 시뮬레이션을 진행하였다. 제안하는 방법은 장애물이 없는 공간에서 Inter i7 노트북으로 에이전트당 평균 9.69 ms만에 70개의 에이전트에 대한 경로를 생성할 수 있으며 100%의 성공률을 보여주었다. 또한, 제안하는 방법은 보로노이 기반 방법 (buffered Voronoi cell) 보다 비행시간이 44.7% 더 짧았으며 기존 선행 연구보다 비행시간이 23% 더 짧은 것으로 나타났다. 그리고 제안하는 방법은 동적 장애물이 있는 환경에서는 최신 알고리즘과 비교했을 때 가장 높은 성공률과 낮은 비행시간을 보여주는 것을 확인하였다. 특히 제안하는 방법은 동적 장애물의 속도가 에이전트의 최대속도 이하일때 90 % 이상의 성공률을 보여주었다. 제안하는 알고리즘의 안전성과 강건성을 검증하기 위해 보행자가 있는 미로 같은 환경에서 실험을 10대의 쿼드로터를 진행하는 실험을 진행하였으며 충돌 또는 교착상태가 발생하지 않음을 확인하였다.

주요어: 다중 로봇 경로 계획, 분산 로봇 시스템, 충돌 회피, 교착 상태 해소
학 번: 2020-39650