공학박사학위논문

An Approach to Motion Planning and Behavior Coordination for Multi-Robot Systems

다중 로봇 시스템의 이동 계획 및

행동 조정을 위한 접근법에 관한 연구

2014년 8월

서울대학교 대학원

기계항공공학부

곽 동 준

An Approach to Motion Planning and

Behavior Coordination for Multi-Robot Systems

다중 로봇 시스템의 이동 계획 및

행동 조정을 위한 접근법에 관한 연구

지도교수 김 현 진

이 논문을 공학박사 학위논문으로 제출함

2014년 6월

서울대학교 대학원

기계항공공학부

곽 동 준

곽동준의 공학박사 학위논문을 인준함

2014년 6월

위 원 장 : 권 유 근

부위원장 : 김 현 진

위 원 : 박 종 우

위 원 : 이 범 희

위 원 : 이 민 철

An Approach to Motion Planning and

Behavior Coordination for Multi-Robot Systems

DONG JUN KWAK

Department of Mechanical and Aerospace Engineering

Seoul National University

APPROVED:

Youdan Kim, Chair, Ph.D.

H. Jin Kim, Ph.D.

Frank Chongwoo Park, Ph.D.

Beom-Hee Lee, Ph.D.

Min Cheol Lee, Ph.D.

An Approach to Motion Planning and Behavior Coordination for Multi-Robot Systems

A Dissertation

by

DONG JUN KWAK

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 Mechanical and Aerospace Engineering

Seoul National University

Supervisor : Professor H. Jin Kim

AUGUST 2014

to my

MOTHER, FATHER, and BROTHER

with love

Abstract

An Approach to Motion Planning and Behavior Coordination for Multi-Robot Systems

Dong Jun Kwak
Department of Mechanical and Aerospace Engineering
The Graduate School
Seoul National University

This thesis suggests a cooperative control architecture and algorithms for mission planning, path planning, and learning for behavior coordination in heterogeneous multi-robot systems. We apply our proposed algorithms to unmanned combat systems. To achieve integrated group objectives of unmanned combat systems consisting of heterogeneous multiple robots in ever-changing battlefield situations, each robot has to make a decision properly by considering characteristics of the predefined mission and real situations. To this end, we first design a cooperative control architecture of a command and control vehicle, unmanned ground vehicles, and unmanned aerial vehicles.

We assign the mission points (the threats' location) to the ground robots by employing a decentralized task assignment called consensus-based bundle algorithm (CBBA). Here, we use a scoring matrix reflecting heterogeneity when the robots plan the mission because the different types of ground robots and threats have different capabilities for performing the mission according to their types. In addition, we suggest an episodic parameter optimization method using reinforcement learning (RL) and particle swarm optimization (PSO). This method is applied to optimize the scoring matrix, and we finally establish the optimal engagement strategy to maximize the team survivability of the ground robots.

To solve a path planning problem between the robot and the target, we propose a decentralized trajectory optimization method using virtual motion camouflage (VMC) and PSO. VMC is inspired from biology, in which an insect actively camouflages its motion

while tracking a prey. By using VMC a typical nonlinear constrained trajectory optimization problem can be transformed to an optimization problem of path control parameters (PCPs), so it reduces the dimension of the original problem. We employ PSO to optimize PCPs. The proposed algorithm called decentralized VMCPSO is applied to solve rendezvous problems considering terminal time and angle constraints in an unban-like environment, and it is validated with simulations and an experiment. By utilizing the algorithms proposed above, we can easily implement behaviors of the robots in complex combat situations.

Lastly, we propose a distributed multi-agent reinforcement learning algorithm in a semi-Markov Decision Process framework to solve a behavior coordination problem for making each robot select the most proper behavior in given situations. In order to address complexity issues, linear function approximation and diffusion adaptation have been employed. As a result, we can achieve the group objectives of the heterogeneous multi-robot systems in combat situations involving many uncertainties by using the proposed approaches.

keywords: multi-robot systems, cooperative control architecture, mission planning, path planning, behavior learning

Student Number: 2011-30196

Table of Contents

Page

Abstract vi

Table of Contents viii

List of Tables xi

List of Figures xii

Chapter

1 Introduction 1

1.1 Literature Survey 3

1.1.1 Multi-robot mission planning 3

1.1.2 Multi-robot path planning 4

1.1.3 Multi-robot learning 5

1.2 Research Objectives and Contributions 6

1.3 Thesis Organization 8

2 Cooperative Mission of Multi-Robot Systems 9

2.1 Probabilistic Engagement Scenario 10

2.2 Multi-Robot Systems Architecture 11

2.2.1 Command and control vehicle 12

2.2.2 Unmanned aerial vehicle 12

2.2.3 Unmanned ground vehicle 13

2.3 Threat Map 19

2.4 Visibility Map 20

3 Multi-Robot Mission Planning 24

3.1 Mission Assignment 25

3.2 Optimization for Mission Assignment 28

3.2.1 Reinforcement learning 28
3.2.2 Particle swarm optimization 30
3.3 Optimization Results 33
3.4 Analysis and Discussion 35
4 Multi-Robot Path Planning 37
4.1 Virtual Motion Camouflage 38
4.1.1 Nonlinear constrained trajectory optimization problem 38
4.1.2 VMC formulation 39
4.1.3 The proposed approach 42
4.2 Extension to Multi-Robot Path Planning 46
4.3 Simulation Results 50
4.3.1 Stationary target 50
4.3.2 Moving target 59
4.4 Experimental Results 63
4.5 Analysis and Discussion 66
5 Behavior Coordination 67
5.1 Design of Behaviors 68
5.2 Learning Framework 69
5.2.1 MDP vs. SMDP 69
5.2.2 Linear approximation of value functions 71
5.2.3 Learning value function approximations 73
5.3 Distributed Multi-Agent Reinforcement Learning 76
5.3.1 Diffusion adaptation method for distributed optimization 76
5.3.2 Cooperative GQ-learning 77
5.4 Distributed MARL Applied to Multi-Robot Systems 81
5.4.1 State space $\mathcal{S}$ 81
5.4.2 Option space $\mathcal{O}$ 82
5.4.3 Reward R 83

5.4.4 Event conditions $\mathcal{E}$ 84
5.5 Empirical Results 85
5.6 Analysis and Discussion 90
6 Conclusions 96
Abstract (*in Korean*) 106

List of Tables

2.1 Choices of μ_{k_i,k_j} (UGV side) 16

2.2 Choices of μ_{k_i,k_j} (Threat side) 16

2.3 Choices of t_{ac} depending on the type 17

3.1 Inputs of the EPO algorithm 29

4.1 Simulation conditions 51

4.2 Terminal time and angle errors and costs with respect to t_d and γ_d 56

4.3 Computation time with respect to S 56

4.4 Computation time of decentralized VMCPSO for the moving target 59

5.1 Features considered for function approximation 82

List of Figures

2.1 Engagement scenario 10
2.2 Multi-robot systems architecture 11
2.3 UAV model 13
2.4 State-tracking error transformation 14
2.5 Attack capabilities of the ground robots 18
2.6 Attack capabilities of the threats 19
2.7 Results for the visibility map (black: invisible, white: visible) 23

3.1 Flow chart of the EPO algorithm 32
3.2 Optimization result 34
3.3 Task allocation results with respect to the different value of $S_i^{\boldsymbol{p}_i}$ (red: $k_i, k_j = 1$, green: $k_i, k_j = 2$, blue: $k_i, k_j = 3$) 36

4.1 Relationship between the reference point, the prey motion, and the aggressor motion. 39
4.2 Example of local and global path planning 47
4.3 Multi-robot operation in an urban-like environment (a) A snapshot of a rendezvous experiment in which multiple UGVs try to arrive at the target point simultaneously with the specific heading (b) Ground display 50
4.4 VMCPSO results of the minimum-time problem 52
4.5 (a,b) The case in which path information of robot j is not considered in the planning of robot i, (c,d) The case in which path information of robot j is considered in the planning of robot i 54
4.6 VMCPSO results of the terminal angle and time constrained problem (a,b) $t_d = 20\,\text{sec}$, (c,d) $t_d = 25\,\text{sec}$, (e,f) $t_d = 30\,\text{sec}$ 58

4.7 Tangential and angular velocities 60
4.8 Snapshots of global and local path planning 62
4.9 Experimental environment 63
4.10 Experimental results of decentralized VMCPSO for the stationary target . 64
4.11 Snapshots of the rendezvous experiment in which three robots try to arrive at the stationary target point simultaneously with the specific heading . . 65

5.1 Change in values of performance measures as the number of episodes increases 87
5.2 Change in values of weights for each robot as the SMDP time step k increases 89
5.3 Integrated simulation results at the SMDP time step $k = 0$ 91
5.4 Integrated simulation results at the SMDP time step $k = 100$ 92
5.5 Integrated simulation results at the SMDP time step $k = 205$ 92
5.6 Integrated simulation results at the SMDP time step $k = 300$ 93
5.7 Integrated simulation results at the SMDP time step $k = 400$ 93
5.8 Integrated simulation results at the SMDP time step $k = 500$ 94
5.9 Integrated simulation results at the SMDP time step $k = 560$ 94
5.10 Change in options for each robot as the SMDP time step k increases 95

1 Introduction

Recent advances in intelligent robotic systems have allowed exploitation of autonomous vehicles in various fields. Currently, centered around the United States, utilization of unmanned robots in the actual battle is on the rise. For example, in Afghanistan, unmanned aerial vehicles (UAVs) called MQ-9 Reaper have performed reconnaissance and attack missions to suppress Taliban and Al-Qaeda. In addition to the unmanned aircraft, in 2001 the United States Congress said that one third of all ground combat vehicles should be unmanned until 2015. Subsequently, Defense Advanced Research Projects Agency (DARPA) defined the unmanned vehicle's autonomy as the core technology, and the DARPA Grand Challenge was held to secure key technologies that can be used in the city as well as the desert. As a result, autonomy's core technologies such as recognition of external environments, path planning, and vehicle control were acquired. Moreover, Google has nearly completed road tests of the autonomous car in urban areas, aiming to commercialize in 2017.

Thus, many key technologies have been secured for a single unmanned robot, but when we consider multiple robots, the other issues are occurred. To operate multiple robots,

a team of robots should collaborate to achieve group objectives. So, there are additional challenging problems. Typically, there are two required technologies: one is multi-robot mission planning for assigning the multiple tasks to the multiple robots, and the other is multi-robot path planning for generating paths without interference among the robots. These technologies have been applied to many multi-robot problems such as coordination, mission assignment, search optimization, rendezvous, etc [4, 14, 25, 67]. From another point of view, we can consider semi-autonomous systems including multiple robots and operators as a key technology to deal with more complex missions.

In this regard, Grand Challenge[1] sponsored by Ministry of Defense in the United Kingdom was held in 2008. Grand Challenge focused on producing an autonomous or semi-autonomous system designed to detect, identify, monitor, and report a comprehensive range of physical threats in a complex urban environment. The competitors were already aware of the danger points where the enemy may lie in ambush. Thus, they should gather the detailed information of the enemies from satellite and manned/unmanned ground/aerial vehicles. Many participating teams suggested innovative ideas and technology to solve some or all the challenges faced in a hostile urban environment.

The United States and Australian militaries held Multi-Autonomous Ground-robotic International Challenge (MAGIC)[2] in 2010 [43]. When multiple robots execute reconnaissance and surveillance missions in a hostile urban environment, each participating team tries to ensure maximum autonomy for multiple robots while minimizing operator's intervention. In this process, there basically needed the operational interface to observe the state of the robots and to steer the robot. In addition, there were interesting challenges in global state estimation, multi-robot cooperation, and robot perception.

The main goal of this thesis is to develop, implement, and test methodologies that can be used in real-world applications, particularly heterogeneous multi-robot combat systems in the ever-changing battlefield. To enhance the autonomy of the robots, we focus on solving

[1] http://www.challenge.mod.uk
[2] http://www.dsto.defence.gov.au/MAGIC2010

decision-making problems: multi-robot mission planning, multi-robot path planning, and multi-robot learning.

1.1 Literature Survey

This section provides a literature survey for key issues in multi-robot systems, for example, multi-robot mission planning, multi-robot path planning, and multi-robot learning. The overview of significant literature published on these topics is as follows.

1.1.1 Multi-robot mission planning

Centralized methods were traditionally used to solve a task assignment problem [5]. In [49], a task assignment problem was formulated as mixed integer linear programming (MILP). MILP provided the optimal task allocation results involving timing and task order constraints between the robots and the tasks. Game-theoretic approaches were also used to solve a vehicle-target assignment problem [3, 38]. In this centralized approach, a specific robot performs the role as a leader and plans its missions as well as the others. Then, the leader robot sends the information of the planned tasks to others through communication channels. This approach results in a simpler communication structure, in which the robots do not have to communicate among themselves, but a heavy computational load is placed on the leader robot. Moreover, in this structure the loss or malfunction of the leader robot can result in a total breakdown of the system.

The other approach is decentralized task assignment, in which each robot makes a decision for itself [1]. This approach is robust to malfunctions of the robots. Decentralized task assignment assumes exact situational awareness to make consensus among the robots. If the communication is not smooth, conflicts can occur among the mission plans of the robots. To overcome this weakness, consensus-based bundle algorithm (CBBA) was suggested in [12]. In CBBA, the robots make consensus within certain predefined rules to avoid task conflicts among the robots, and it is advantageous in situations involving various constraints. In [6],

for example, CBBA was extended to handle complications in realistic multi-UAV operations by considering obstacle regions in order to avoid collisions and reducing sensitivity due to sensor measurement noise. CBBA was also used to allocate heterogeneous tasks to robots that have different capabilities in a cooperative track and strike mission [13].

1.1.2 Multi-robot path planning

There have been variety of studies to build a real-time decentralized path planning algorithm in the multi-robot coordination problems. In [46, 28, 9, 2], decentralized model predictive control (MPC) or receding horizon control (RHC) was investigated by reformulating the original MPC problem as subproblems of each robot in a decentralized control problem. In [48, 33], a receding horizon strategy based on MILP was used to obtain the optimal trajectories of UAVs. In [17], authors proposed decentralized multi-agent rapidly-exploring random tree (DMA-RRT) and cooperative DMA-RRT which extend closed-loop RRT [34] by using merit-based token passing. RRT tree is used to quickly identify and compare the cost of paths for each robot, and a cooperation strategy improves team performance and prevents deadlock situations.

Traditionally, many researchers have studied nonlinear constrained trajectory optimization problems. There are many numerical techniques for solving this problem, and it is helpful to categorize them into two parts: indirect or direct [53, 7]. Indirect methods are based on the calculus of variation or the maximum principle and analytically seek a solution of the necessary conditions for optimality. There is no requirement for discretization. Direct methods do not require analytical expression unlike the indirect methods. Instead, direct methods discretize the original problem, then apply nonlinear programming techniques to the resulting finite-dimensional optimization problem [8, 16]. In [26], authors compared both analytical and numerical results of trajectory optimization problems for satisfying the global strike mission of a common aero vehicle in 2-D and 3-D environments . Recently, a virtual motion camouflage (VMC) subspace method was suggested in [63, 64, 65, 66]. VMC method is used to reformulate the typical nonlinear constrained trajectory optimiza-

tion problem by using path control parameters (PCPs). Then, the original problem with infinite dimension changes into the finite dimensional problem with PCPs. We can obtain arbitrary paths by changing PCPs, and the optimization problem can be solved by optimizing PCPs.

1.1.3 Multi-robot learning

In [45], an intelligent cooperative control architecture (iCCA) was introduced for learning and adapting cooperative control strategies in real-time stochastic environments. iCCA has two modules: one is a CBBA planner, and the other is an actor-critic learner with a risk analyzer. When CBBA generates a feasible plan for the mission, the actor-critic reinforcement learner incrementally adapts its policy to maximize the cumulative rewards. Afterward, iCCA was extended to AM-iCCA by adding the model estimation routine [23, 22]. In the mission, in which fuel-limited UAVs sequentially visit targets, empirical results showed that their proposed methods have advantage compared to pure learning and planning methods. In addition, an integrated learning-planning algorithm was suggested in [60], and the proposed algorithm was validated with experiments in a persistent search and track scenario.

To address prohibitively large learning spaces, reinforcement learning (RL) based on behaviors has been widely used in many multi-robot applications such as object transportation [32, 61, 24], multi-target observation [57, 35], and robot soccer [52, 62, 19, 47], etc. The use of behaviors provides a encoding, and it is useful to lend robustness to control and allow abstraction for handling large-scale learning [40]. In [39], a group of mobile robots learn a policy by finding a mapping from certain conditions to behaviors in a foraging task. Especially, robot soccer is the most popular test-bed for multi-robot learning. In [52], authors suggested episodic SARSA(λ) with linear tile-coding function approximation in a semi-Markov Decision Process (SMDP) framework to learn a *keepaway* task of RoboCup soccer, in which one team tries to keep possession of the ball while another team tries to steal the ball. In [62], empirical studies performed for the keepaway task by combining temporal difference methods with evolutionary techniques such as NEAT. RL based on

fuzzy neural network and a batch RL method called neural fitted Q-iteration were applied to learn crucial skills for soccer-playing robots [19, 47].

1.2 Research Objectives and Contributions

In this study, we focus on solving decision-making problems of multi-robot systems in order to achieve group objectives such as maximizing the team survivability of the ground combat robots in probabilistic engagement scenarios. The main contributions can be summarized as below.

- *Design of a cooperative control architecture for multi-robot systems*
  We design autonomous combat systems that consist of a command and control vehicle, multiple ground combat robots, and multiple surveillance flying robots. Concrete control modules and communication structure are proposed for each component of autonomous combat systems.

- *Optimization for mission planning*
  We suggest a combat strategy by employing CBBA and a scoring matrix reflecting heterogeneity between robots and threats in probabilistic engagement scenarios. For better performance such as the team survivability, an episodic parameter optimization (EPO) algorithm is proposed to find the optimal combat strategy by optimizing the scoring matrix. The EPO algorithm is performed during the numerous repeated simulation runs of the engagement and the reward of each episode is updated using RL. The optimal scoring matrix is selected by particle swarm optimization (PSO). The optimization results show that the team survivability of the ground robots is increased after performing the EPO algorithm and the values of the scoring matrix are also optimally selected.

- *Development of real-time distributed path planning algorithms*
  A decentralized trajectory optimization method to solve a nonlinear constrained tra-

jectory optimization problem is proposed by employing virtual motion camouflage (VMC) and PSO, and it is called decentralized VMCPSO (Dec-VMCPSO). We apply Dec-VMCPSO to a path planning problem constrained on the terminal time and angle in a multi-robot application. VMC changes a typical full space optimal problem to a subspace optimal problem, so it can reduce the dimension of the original problem by using path control parameters (PCPs). If PCPs are optimized, then the optimal path can be obtained. In this work, PSO is used to optimize these PCPs. In the multi-robot path planning, each robot sequentially generates its own optimal path by using VMC and PSO, and sends the optimized PCPs to the other robots. Then the other robots use these optimized PCPs when planning their paths. The numerical simulation and experimental results show that the optimal paths considering the terminal time and angle are effectively generated.

- *Learning for behavior coordination*

  Behaviors of the ground robots are implemented by using the mission and path planning modules of the ground robots. To solve a behavior coordination problem, we propose a distributed multi-agent reinforcement learning (MARL) algorithm using linear function approximation and diffusion adaptation in a semi-Markov Decision Process (SMDP) framework. The function approximation technique combined with the SMDP framework is used to solve high-dimensionality problems, and the diffusion adaptation method is used for making the robots learn their option-value functions by sharing their experiences. In a distributed MARL process, empirical results show that the most probable behavior of the robots is selected in a direction to maximize the performance even the completely different combat scenarios as the number of episodes increases.

1.3 Thesis Organization

This thesis consists of four main parts: systems architecture, mission planning, path planning, and learning. A brief overview of each technical component for multi-robot combat systems is presented as below.

Chapter 2 describes group objectives of multi-robot systems in engagement scenarios. Then, we design a multi-robot systems architecture and explain the role of each part for multi-robot systems.

In chapter 3, details of a mission planning algorithm called consensus-based bundle algorithm (CBBA) are presented, and a scoring matrix for reflecting heterogeneity among ground robots is introduced. For better mission performance such as the team survivability of the robots, an episodic parameter optimization (EPO) method is suggested to find the optimal combat strategy depending on CBBA and the scoring matrix. After applying the EPO algorithm to probabilistic combat scenarios, optimization results will be investigated.

Chapter 4 describes a basic nonlinear constrained trajectory optimization problem, and a basic concept of virtual motion camouflage (VMC) is presented. Then, we propose a numerical method called VMCPSO to find the optimal point-to-point path. Afterward, we show how VMCPSO is applied to trajectory optimization problems constrained on the terminal time and angle. Simulation and experimental results also will be discussed.

In chapter 5, we first design appropriate behaviors for dealing with our engagement scenarios by employing the proposed methods discussed in Chapters 2–4. To solve a behavior coordination problem, a distributed multi-agent reinforcement learning algorithm is proposed in a semi-Markov Decision Process framework. The proposed algorithm is validated with many repeated numerical simulations.

Chapter 6 summarizes the issues and its solutions considered in this thesis.

2

Cooperative Mission of Multi-Robot Systems

In multi-robot systems, many researchers mainly focus on solving how to make cooperative behaviors of the robots while satisfying group objectives. We can categorize required technical details according to the organization of multi-robot systems and the group objectives such as formation keeping, rendezvous, surveillance and reconnaissance. Typically, the main issues of multi-robot systems can be a cooperative control architecture, communication, task assignment, path planning, swarm robots, learning, and so on. In this thesis, our multi-robot combat systems consist of a command and control vehicle, unmanned ground vehicles (UGVs), and unmanned aerial vehicles (UAVs), and we consider probabilistic engagement scenarios. In a combat environment, there are many uncertainties such as the malfunction or destruction of robots, pop-up obstacles, and threats, so decision-making problems should be reasonably solved given situations. To this end, we focus on the design of multi-robot systems architecture and multi-robot problems for task assignment, path planning, and behavioral learning. All the considering algorithms will be designed in a distributed manner to ensure robustness, sustainability, and reliability. In this chapter, we first introduce group objectives of multi-robot systems in our considering engagement scenarios.

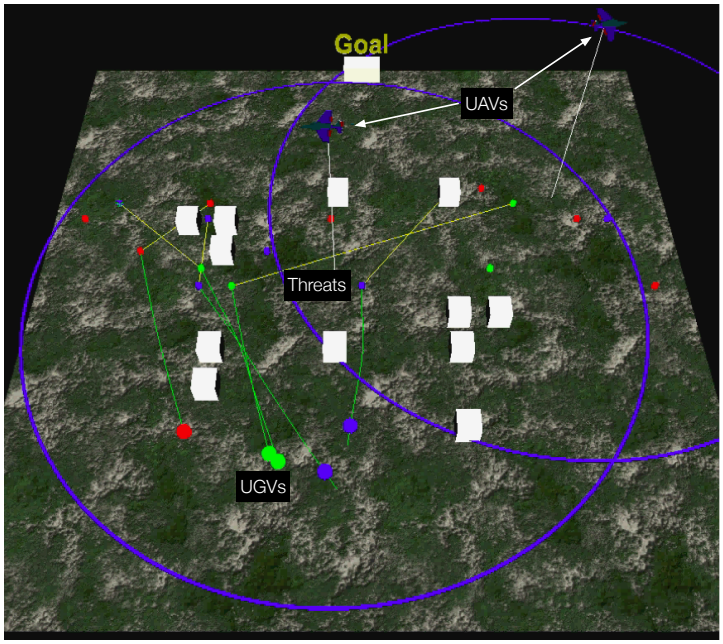


Figure 2.1: Engagement scenario

Then, details of multi-robot systems architecture will be presented. Furthermore, we will explain how to build a threat map and visibility maps.

2.1 Probabilistic Engagement Scenario

We consider engagement scenarios in a three-dimensional environment. There are a team of N_u heterogeneous UGVs and two UAVs for surveillance and reconnaissance, and N_t threats as shown in Fig. 2.1. The goal of ground robots is to overwhelm stationary threats while they move to the target location considering the team safety. The three types of ground robots $i \in \{1, \ldots, N_u\}$ compose one ground team. We denote k_i as the type of the robot i. The threats $j \in \{1, \ldots, N_t\}$ also consist of three types. k_j denotes the type of the threat j. Both the robots i and the threats j have different attack capabilities which are described by the attack probability, the attack range, and the attack cycle depending on the type k_i and k_j. We assume that each robot receives the information that contains the locations and

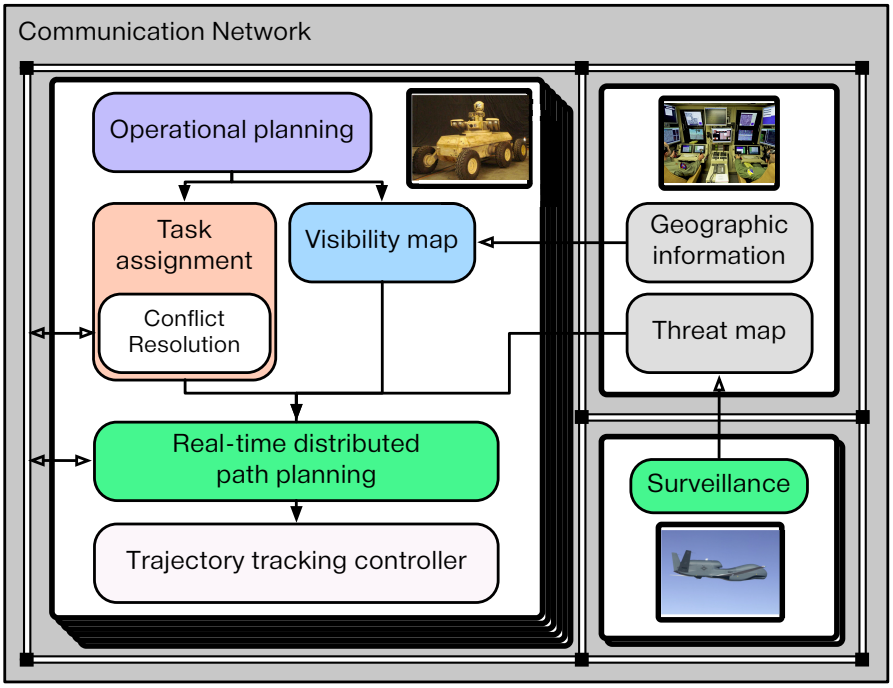


Figure 2.2: Multi-robot systems architecture

the types of the threats from a command and control vehicle. To maximize attainability of the mission objectives, the robots have to select the proper targets while they pass over or overwhelm the threats.

2.2 Multi-Robot Systems Architecture

Multi-robot systems consist of a command and control vehicle, ground combat robots, and flying surveillance robots as shown in Fig. 2.2. Each part of multi-robot systems can communicate with each other. In this thesis, we assume that they are fully connected in a communication network.

2.2.1 Command and control vehicle

In the command and control vehicle, an operator tries to reflect a mission came from a high-level commander according to the actual situation. So, the operator is allowed to access and control the ground robots from a high-level mission planning to a low-level vehicle control. Here, the role of the operator is restricted because we consider autonomous combat systems. Furthermore, the support map related to visibility and threats is created, and the ground robots utilize it to plan their trajectories. The threat map and the visibility map for the threats are updated at every second by using the information of threats given by UAVs. Details for generating these maps will be described in the end of this chapter.

2.2.2 Unmanned aerial vehicle

UAVs observe the position and type of threats scattered in the battlefield, flying along the predefined waypoints. Then, the collected information of the threats is transmitted to the command and control vehicle, and it is used to update the threat map and the visibility map.

We use the kinematic model as shown in Fig. 2.3. The UAV dynamics and control logic are applied by using the study of [30], and the trajectory generator is considered by the following:

$$\begin{aligned} \dot{x}^d &= v^d \cos\psi^d \\ \dot{y}^d &= v^d \sin\psi^d \\ \dot{\psi}^d &= u \\ \dot{v}^d &= 0 \\ \dot{h}^d &= 0 \end{aligned} \tag{2.1}$$

where (x^d, y^d, h^d) is the desired inertial position of UAV, ψ^d is the desired heading, v^d is the desired velocity, h^d is the desired altitude. The desired altitude is always positive, and

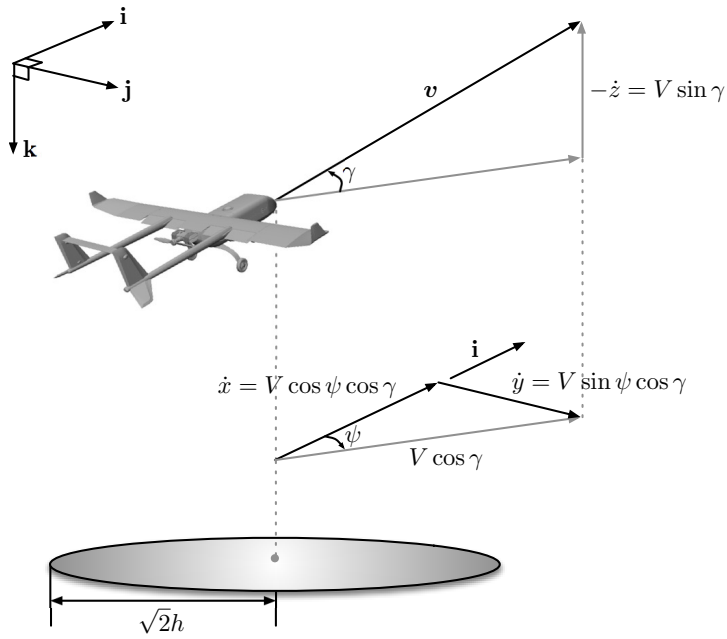


Figure 2.3: UAV model

the desired heading rate input u is designed by a limit cycle navigation [29]. The details related to the low-level control are omitted because this thesis is mainly focused on control of UGVs.

2.2.3 Unmanned ground vehicle

The internal control structure of the ground combat robot is divided into three kinds of modules. The mission planning module assigns mission points to the ground robots depending on mission objectives, and it is designed in a decentralized manner. As shown in Fig. 2.2, the mission planning module allows to communicate with the other robots for resolving conflicts between tasks assigned to each robot. In addition, anytime the operator can modify the already planned tasks when necessary although not considered in this thesis. Next, the support map applied to global/local path planning is built by using the information obtained from the command and control vehicle, and this map is regarded

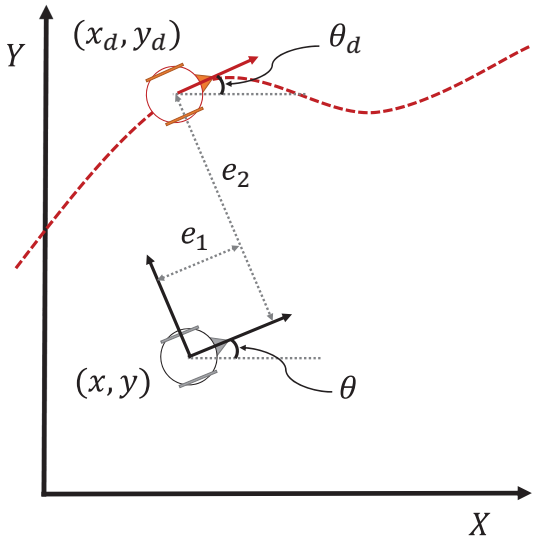


Figure 2.4: State-tracking error transformation

as the cost or performance index when the robot plans its trajectory. This path planning module should be designed in a distributed/real-time manner, and the trajectory tracking controller is also needed to perfectly follow the preplanned path. Furthermore, if we design behaviors well suited for the mission, it is advantageous to reduce the operator's fatigue in the situation required the manual control of the operator. To improve the robots' autonomy, a learning process for selecting the most proper behavior in arbitrary combat situations is required. This will be discussed in Chapter 5.

The kinematic model of the ground robot is considered as the following unicycle robot:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \tag{2.2}$$

where (x, y, θ) denote the position and heading angle of the robot, and (v, w) are the tangential and angular velocities. In this paper $\boldsymbol{u} = [v, w]^\top$ is treated as the velocity input vector to the robot dynamics. Let a feasible and smooth desired trajectory (x_d, y_d) be given

in a time interval $t \in [t_0, t_f]$. The generation of the desired trajectory (x_d, y_d) for each robot will be described in Chapter 4. Then, the desired tangential velocity v_d is defined as

$$v_d = \pm\sqrt{\dot{x}_d^2 + \dot{y}_d^2} \tag{2.3}$$

and the desired heading angle θ_d is calculated as

$$\theta_d = \text{atan}\left(\frac{\dot{y}_d}{\dot{x}_d}\right). \tag{2.4}$$

Differentiating (2.4), the desired angular velocity w_d is obtained as follows:

$$w_d = \frac{\dot{x}_d\ddot{y}_d - \dot{y}_d\ddot{x}_d}{\dot{x}_d^2 + \dot{y}_d^2}. \tag{2.5}$$

To design the trajectory tracking controller for the robot, the state-tracking error $\boldsymbol{e}$ is defined as shown in Fig. 2.4. Then, the trajectory tracking controller can be obtained from [36].

As mentioned in Sect. 2.1, the ground robots and the threats have different attack capabilities according to their types. The attack status $\boldsymbol{\Xi}_i$ of the robot i consists of the set 4-tuples as follows:

$$\boldsymbol{\Xi}_i = \langle \mathbf{W}_i, \mathbf{A}_i, \mathbf{S}_i, \mathbf{V}_i \rangle \tag{2.6}$$

where $(\mathbf{W}_i, \mathbf{A}_i, \mathbf{S}_i, \mathbf{V}_i)$ indicate weapon cooldown, possibility of attack, viability, and visibility, respectively. Each component of the attack status has the value 0 or 1. When the weapon cooldown state $\mathbf{W}_i$ has 0, this means that the weapon is ready to attack. For the other case the weapon has to be cooldown ($\mathbf{W}_i = 1$). The possibility of attack state $\mathbf{A}_i$ is closely related to the attack range as shown in Fig. 2.5. When one of the threats is located inside of the attack range, $\mathbf{A}_i$ becomes 1. Otherwise, $\mathbf{A}_i$ has 0. When the robot is alive, the viability state $\mathbf{S}_i$ has 1. The visibility state $\mathbf{V}_i$ becomes 1 when the robot i is located

Table 2.1: Choices of μ_{k_i,k_j} (UGV side)

	T1 ($k_j = 1$)	T2 ($k_j = 2$)	T3 ($k_j = 3$)
U1 ($k_i = 1$)	0.8	0	0
U2 ($k_i = 2$)	0.8	0.8	0
U3 ($k_i = 3$)	0.8	0.8 – 0	0.8 – 0

Table 2.2: Choices of μ_{k_i,k_j} (Threat side)

	U1 ($k_i = 1$)	U2 ($k_i = 2$)	U3 ($k_i = 3$)
T1 ($k_j = 1$)	0.5	0	0
T2 ($k_j = 2$)	0.5	0.5	0
T3 ($k_j = 3$)	0	0.5	0.5

in the threats' field of vision. Similarly, the threat j also has the information of its own status $\Xi_j = \langle \mathbf{W}_j, \mathbf{A}_j, \mathbf{S}_j, \mathbf{V}_j \rangle$. For the visibility state $\mathbf{V}_j$ of the threat j, when the threat j is located in the robots' field of vision, $\mathbf{V}_j$ becomes 1.

The probability for the robot i to kill the threat j is defined by the following:

$$P(\mathbf{S}_j = 0 | \mathbf{S}_j = 1, \mathbf{V}_j = 1, \mathbf{W}_i = 0, \mathbf{A}_i = 1, \mathbf{S}_i = 1) = \text{Bern}\left(\mathbf{S}_j = 0 | \mu_{k_i,k_j}\right) \tag{2.7}$$

where $\text{Bern}(x|\mu) = (1-\mu)^x \mu^{(1-x)}$ is known as the *Bernoulli* distribution, and μ_{k_i,k_j} is defined as shown in Table 2.1. Eq. (2.7) shows that when both the robot i and the threat j are alive ($\mathbf{S}_i = 1$, $\mathbf{S}_j = 1$), when the robot i is ready to attack ($\mathbf{W}_i = 0$), when the threat j is visible ($\mathbf{V}_j = 1$) and located inside of the attack range ($\mathbf{A}_i = 1$), then the probability of kill is given by the Bernoulli distribution. As shown in Table 2.1, U1 ($k_i = 1$) can only contribute to eliminate T1 ($k_j = 1$), so μ_{k_i,k_j} is set to 0.8 for T1 and 0 for T2/T3. In the case of U2 ($k_i = 2$), it can exert influence on both T1 ($k_j = 1$) and T2 ($k_j = 2$), so μ_{k_i,k_j} is set to 0.8 for T1/T2. The U3 can deal with all the threats.

Table 2.3: Choices of t_{ac} depending on the type

	Type 1	Type 2	Type 3
UGV	5 s	3 s	8 s
Threat	5 s	3 s	8 s

Similar to the ground robot, the threat j can attack the robot i with the probability of kill which is defined by the following:

$$P(\mathbf{S}_i = 0|\mathbf{S}_i = 1, \mathbf{V}_i = 1, \mathbf{W}_j = 0, \mathbf{A}_j = 1, \mathbf{S}_j = 1) = \text{Bern}\left(\mathbf{S}_i = 0|\mu_{k_i,k_j}\right). \tag{2.8}$$

As shown in Table 2.2, T1 ($k_j = 1$) can only attack U1 ($k_i = 1$), and T2 can strike both U1 and U2. T3 can deal with both U2 and U3. If the robot is destroyed by the attack of threat ($\mathbf{S}_i = 0$), the robot cannot regenerate and move anywhere. In addition, the weapon cooldown state $\mathbf{W}$ is determined by the following:

$$\mathbf{W} = \begin{cases} 0 & \text{if } |t - t_a| > t_{ac} \\ 1 & \text{otherwise} \end{cases} \tag{2.9}$$

where t_a denotes a time to attack and is updated when the attack is begun at time t ($t_a \leftarrow t$). Here, the attack cycle t_{ac} is defined by Table 2.3 depending on the type.

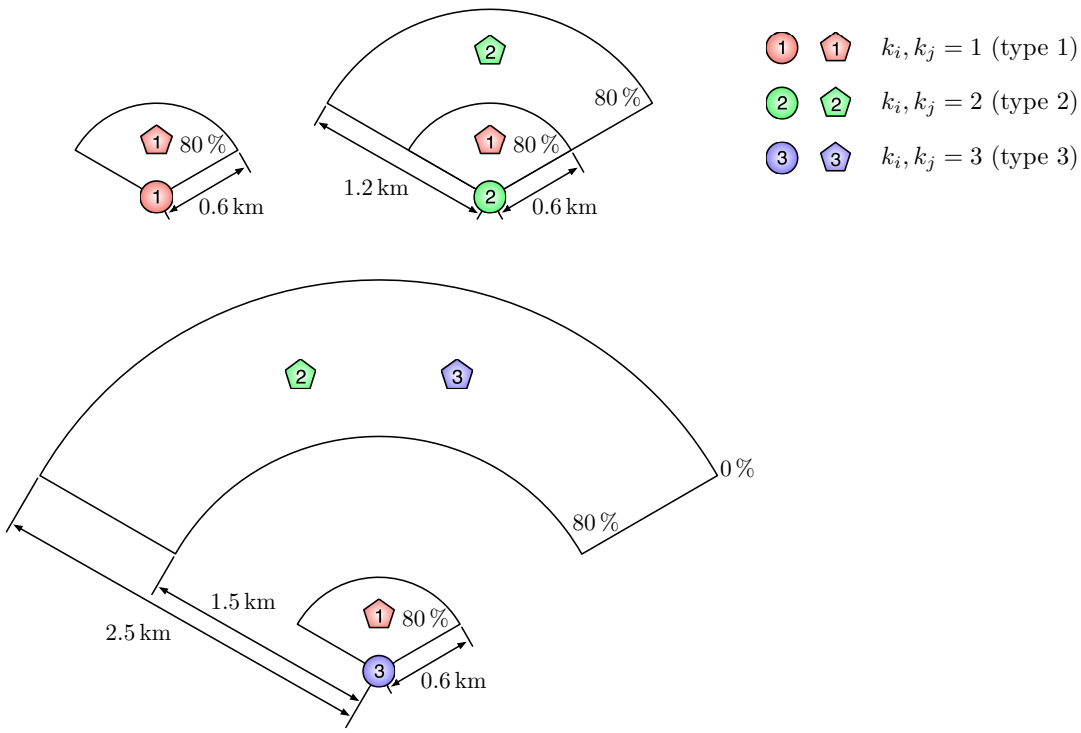


Figure 2.5: Attack capabilities of the ground robots

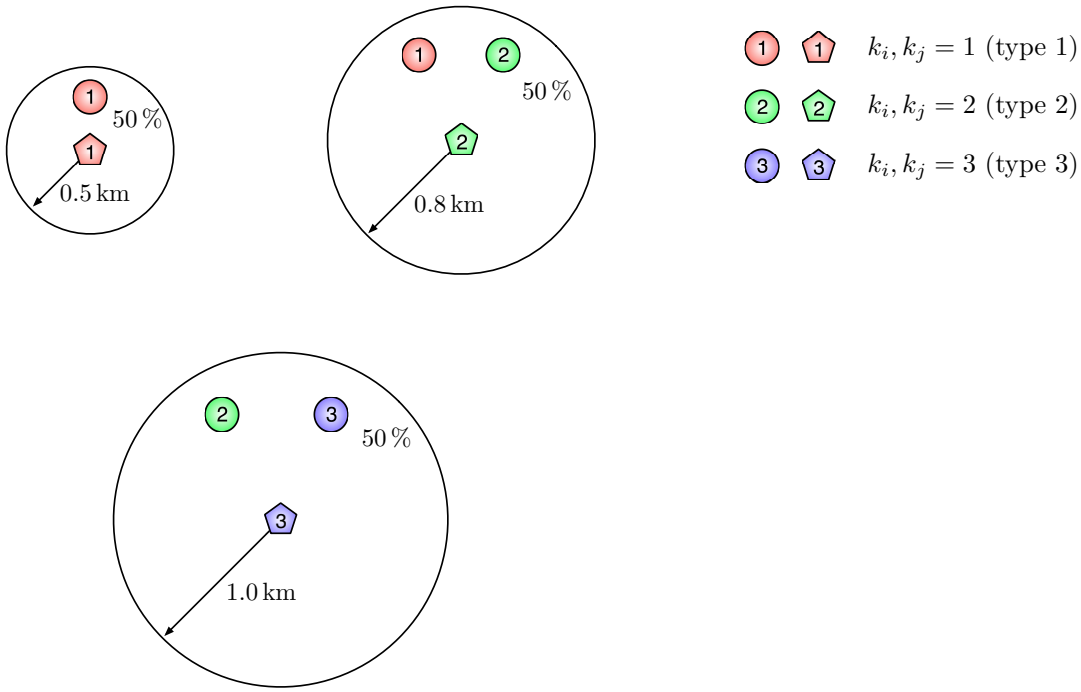


Figure 2.6: Attack capabilities of the threats

2.3 Threat Map

Each UAV uses a gimbaled camera with top-down perspective to estimate the threat level while it makes a flight to predetermined paths and shares the vision sensing information to build a grid-based threat map over $N_{\mathrm{ROW}} \times N_{\mathrm{COL}}$ cells $\boldsymbol{x}_c \in \mathcal{T}$. Let Y_t be the history of the UAVs' sensing data collected up to time t. Y_t consists of the coordinates of observed cells and the threat level information. Then the conditional probability for the threat level in cell $\boldsymbol{x}_c$ at time τ given the measurements Y_t is denoted by $p_e(\boldsymbol{x}_c, \tau | Y_t)$.

Here, we describe the threat map building steps:

Step 1 Initially, we assume that the cells $\boldsymbol{x}_c \in \mathcal{T}$ have a uniform probability for the threat level. Afterwards, the probability distribution is updated according to the vision sensing information.

Step 2 The threat level at the previous time $t - \Delta t$, i.e. $p_e(\boldsymbol{x}_c, t - \Delta t | Y_{t-\Delta t})$, is modified by the measurement Y_t at current time t as follows:

$$p_e(\boldsymbol{x}_c, t - \Delta t | Y_t) = \begin{cases} \sum_{j \in \mathcal{Y}_j} \exp\left(-\left(\frac{(x_c - x_j)^2}{2\sigma_j^2} + \frac{(y_c - y_j)^2}{2\sigma_j^2}\right)\right) & \text{if } \boldsymbol{x}_c \text{ is observed} \\ p_e(\boldsymbol{x}_c, t - \Delta t | Y_{t-\Delta t}) & \text{otherwise} \end{cases} \tag{2.10}$$

where $\mathcal{Y}_j$ means a set of all the detected threats, and σ_j is related with the attack range of the threat j.

Step 3 The probability of the threat level in cell $\boldsymbol{x}_c$ at time t given the measurement Y_t is defined as follows:

$$p_e(\boldsymbol{x}_c, t | Y_t) = \frac{p_e(\boldsymbol{x}_c, t - \Delta t | Y_t)}{\sum_{\boldsymbol{x}_c \in \mathcal{T}} p_e(\boldsymbol{x}_c, t - \Delta t | Y_t)}. \tag{2.11}$$

Step 4 Go to **Step 2** and repeat

2.4 Visibility Map

As shown in Fig. 2.2, UGV uses two visibility maps, one about UGVs, the other about threats, when it makes decision in any given situation. The visibility map about UGVs is used for discriminating whether the threat is occluded by the obstacle or not, i.e., possibility of attack. In the case of the visibility map about threats, UGV uses it for escaping from range of vision for threats. To build these visibility maps, we use an implicit ray casting algorithm introduced in [58].

In [58], the visibility problem is formulated as a boundary value problem of a first-order partial differential equation (PDE). The visibility information $\rho(\boldsymbol{x}; \boldsymbol{x}_o)$, i.e, the minimum value of ϱ along the line segment between any given point $\boldsymbol{x}$ and the vantage point $\boldsymbol{x}_o$, can

Algorithm 1 Basic visibility sweeping algorithm [58]

```
procedure BASIC VISIBILITY SWEEPING
    Set ρ(x_o) = ϱ(x_o)
    Do a star-shaped updating sequence on the grid
    For each grid point x_v, choose x'_v depending on the grid geometry
    Compute the value of ρ^h(x_v) via (2.14)
end procedure
```

be defined as follows:

$$\rho(\boldsymbol{x};\boldsymbol{x}_o) := \min_{t\in[0,1]} \varrho(\boldsymbol{x}_o + t(\boldsymbol{x}-\boldsymbol{x}_o)). \tag{2.12}$$

When we think of obstacles as a level set function $\varrho(\boldsymbol{x})$, this function is negative if a point $\boldsymbol{x}$ inside the obstacles is occluded, and vice versa. Then, $\rho(\boldsymbol{x})$ can be computed by solving the first-order PDE:

$$\begin{cases} \nabla\rho \cdot \frac{\boldsymbol{x}-\boldsymbol{x}_o}{|\boldsymbol{x}-\boldsymbol{x}_o|} = \min\{H(\rho-\varrho)\nabla\varrho \cdot \frac{\boldsymbol{x}-\boldsymbol{x}_o}{|\boldsymbol{x}-\boldsymbol{x}_o|}, 0\} \\ \rho(\boldsymbol{x}_o) = \varrho(\boldsymbol{x}_o) \end{cases} \tag{2.13}$$

where $H(x)$ denotes the Heaviside function:

$$H(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}.$$

To build a grid-based visibility map over $N_{\mathrm{VROW}} \times N_{\mathrm{VCOL}}$ cells $\boldsymbol{x}_v \in \mathcal{V}$, the value of $\rho(\boldsymbol{x})$ is approximated as $\rho^h(\boldsymbol{x}_v)$ on 2-D space:

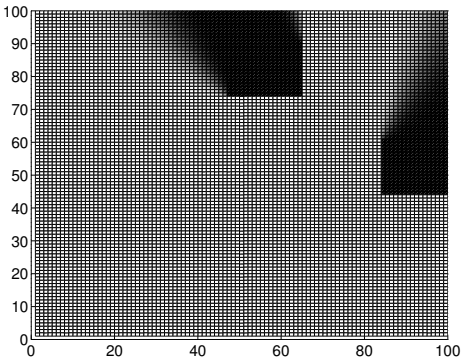
$$\rho^h(\boldsymbol{x}_v) = \min(\rho^h(\boldsymbol{x}'_v), \varrho(\boldsymbol{x}_v)) \tag{2.14}$$

where $\boldsymbol{x}'_v$ is a point immediately before $\boldsymbol{x}_v$ in the ray direction. From this, we can apply a visibility sweeping method given in Algorithm 1. For a complete presentation of the visibility sweeping algorithm see [58].

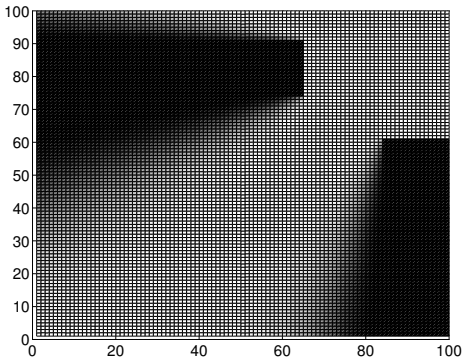
By using the sweeping method, we can obtain visible regions for each vantage point as shown in Figs. 2.7(a)-(c). When the robot i considers possibility of attack, it is sufficient to use the visibility map ρ_i^h based on its own view. However, when UGV wants to find the safe locations that are invisible areas from the viewpoint of the threats, a common visibility map incorporating all the threats' view should be considered. To this end, we define the common visibility map about the threats by the following:

$$\rho_e^h(\boldsymbol{x}_v) = \begin{cases} 0 & \rho_j^h(\boldsymbol{x}_v) < 0 \wedge \mathbf{S}_j = 1, \forall j \in \mathcal{J} \\ 1 & \text{otherwise} \end{cases} . \tag{2.15}$$

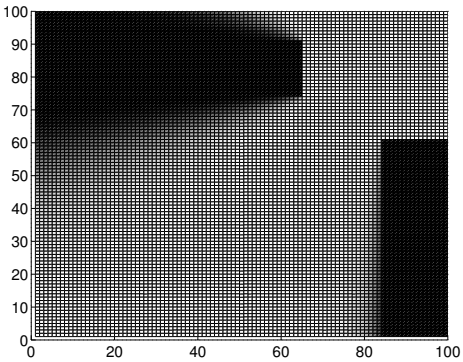
Then, the resulting visibility map is obtained as shown in Fig. 2.7(d).



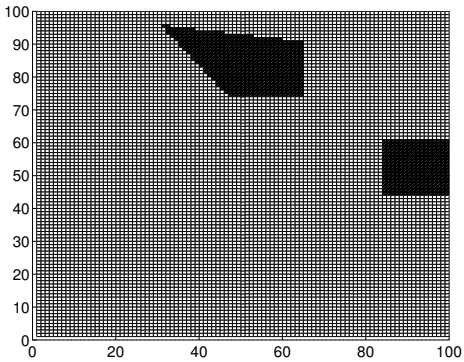
(a) First view



(b) Second view



(c) Third view



(d) Total view

Figure 2.7: Results for the visibility map (black: invisible, white: visible)

3

Multi-Robot Mission Planning

From our discussion in Chapter 2, mission planning plays an important role in multi-robot systems, and we especially consider the heterogeneous robots and threats in the probabilistic combat situations. Therefore, the robots should be allocated to the most proper target depending on their mission capabilities. Here, we consider the multiple robots, so if the mission planning module is designed in a decentralized manner, it can be robust to malfunctions of the robots unlike the centralized approaches resulting in a total breakdown of the system when the central system is out of control. Decentralized task assignment assumes exact situational awareness to make consensus among the robots. If the communication is not smooth, conflicts can occur among the mission plans of the robots. To overcome this weakness, we employ consensus-based bundle algorithm (CBBA). In CBBA, the robots make consensus within certain predefined rules to avoid task conflicts among the robots, and it is advantageous in a situation involving various constraints. We also use a scoring matrix to deal with heterogeneity of the robots and threats. Consequently, we can establish a combat strategy as a combination of CBBA and the scoring matrix. If we select the scoring matrix optimally, the reliability of the combat strategy increases. To optimize

the scoring matrix, we suggest an episodic parameter optimization (EPO) method using reinforcement learning (RL) and particle swarm optimization (PSO). We will show how the proposed method works in engagement scenarios to maximize the team survivability of the ground robots.

3.1 Mission Assignment

This section briefly describes CBBA which is used for our mission [12]. Here, we name the ground robot and the threat as an agent and a task, respectively.

CBBA is one of the decentralized task assignment methods based on the market auction algorithm. Especially, CBBA considers agents' paths that are decided by their sequential task lists to solve a multi-assignment problem. The multi-agent and multi-task assignment problem can be formulated as follows:

$$\max \sum_{i=1}^{N_u} \left(\sum_{j=1}^{N_t} c_{ij} \left(\boldsymbol{x}_i, \boldsymbol{p}_i \right) x_{ij} \right) \tag{3.1}$$

$$\text{subject to } \sum_{j=1}^{N_t} x_{ij} \leq L_t, \ \forall i \in \mathcal{I} \ \ (\mathcal{I} = \{1, \ldots, N_u\}) \tag{3.2}$$

$$\sum_{i=1}^{N_u} x_{ij} \leq 1, \ \forall j \in \mathcal{J} \ \ (\mathcal{J} = \{1, \ldots, N_t\}) \tag{3.3}$$

$$\sum_{i=1}^{N_u} \sum_{j=1}^{N_t} x_{ij} = \min \{N_u L_t, N_t\} \tag{3.4}$$

$$x_{ij} \in \{0, 1\}, \ \forall (i, j) \in \mathcal{I} \times \mathcal{J}$$

where x_{ij} is set to 1 when an agent i is assigned to a task j. Otherwise, $x_{ij} = 0$. The performance index (3.1) is defined by a sum of local reward values, and it should be maximized under the constraints. The first inequality constraint (3.2) limits the number of tasks which the agent i can have. Here, L_t denotes the maximum number of the tasks for each agent. The second inequality constraint (3.3) indicates that each task j should be assigned by at

most one agent. If the equality constraint (3.4) is satisfied, the assignment is said to be completed. CBBA consists of two phases in each iteration. The first is bundle construction and the second is conflict resolution by local communication. Details of the algorithm are given in Algorithms 2 and 3.

In the first phase, each agent continuously adds tasks to its bundle in the order of decreasing reward of the task until it is incapable of adding any others. Each agent carries four vectors: a bundle $\boldsymbol{b}_i \in \mathbb{N}^{L_t}$, the corresponding path $\boldsymbol{p}_i \in \mathbb{N}^{L_t}$, a winning bid list $\boldsymbol{y}_i \in \mathbb{R}^{N_t}$, and a winning agent list $\boldsymbol{z}_i \in \mathbb{N}^{N_t}$. Tasks in the bundle $\boldsymbol{b}_i$ are ordered based on which ones were added first, while in the path $\boldsymbol{p}_i$ they are ordered based on their locations in the assignment. $\boldsymbol{y}_i$ and $\boldsymbol{z}_i$ are the agent i's knowledge vectors which contain the knowledge about which agent takes each task (winning agent) and the successful bidding values (winning bid), respectively. In line 5 of Algorithm 2, $a \oplus_{end} b$ denotes the operation that inserts the list b after the last element of the list a. Every time the agent includes a new task to its bundle and path, it saves the knowledge about $\boldsymbol{y}_i$ and $\boldsymbol{z}_i$ (line 7 of Algorithm 2). In line 4 of Algorithm 2, when agent i takes the task j, the marginal score improvement $c_{i,j}$ according to the current path $\boldsymbol{p}_i$ is given as follows.

$$c_{ij} = \begin{cases} \max_{n \leq |\boldsymbol{p}_i|+1} S_i^{\boldsymbol{p}_i \oplus_n \{J_i\}} - S_i^{\boldsymbol{p}_i} & \text{if } j \notin \boldsymbol{p}_i \\ 0 & \text{if } j \in \boldsymbol{p}_i \end{cases} \tag{3.5}$$

where $|\cdot|$ denotes the cardinality of the list, and $a \oplus_n b$ denotes the operation that inserts the list b right after the n-th element of the list a. Let $S_i^{\boldsymbol{p}_i}$ be defined as the total reward value for agent i performing the task along the path $\boldsymbol{p}_i$. In this paper the value of $S_i^{\boldsymbol{p}_i}$ is inversely proportional to the distance of the path as in example below.

$$S_i^{[1,2,3]} = \frac{m_{k_i,k_1}}{d_{i,1}} + \frac{m_{k_i,k_2}}{d_{i,1} + d_{1,2}} + \frac{m_{k_i,k_3}}{d_{i,1} + d_{1,2} + d_{2,3}} \tag{3.6}$$

where m_{k_i,k_j} denotes a element of the scoring matrix $\boldsymbol{M} \in \mathbb{R}^{3\times 3}$ varying with the type k_i and k_j of the robot i and the threat j. The scoring matrix $\boldsymbol{M}$ is used to reflect the

Algorithm 2 CBBA Phase 1: for agent i at iteration t

procedure BUILD BUNDLE $(\boldsymbol{z}_i(t-1), \boldsymbol{y}_i(t-1), \boldsymbol{b}_i(t-1))$
  $\boldsymbol{z}_i(t) \leftarrow \boldsymbol{z}_i(t-1)$, $\boldsymbol{y}_i(t) \leftarrow \boldsymbol{y}_i(t-1)$, $\boldsymbol{b}_i(t) \leftarrow \boldsymbol{b}_i(t-1)$
  while $|\boldsymbol{b}_i(t)| < L_t$ **do**
    Find task J_i which gives the most marginal score improvement c_{i,J_i} for given $\boldsymbol{b}_i(t)$ and $\boldsymbol{p}_i(t)$
    $\boldsymbol{b}_i(t) \leftarrow \boldsymbol{b}_i(t) \oplus_{end} \{J_i\}$
    $\boldsymbol{p}_i(t) \leftarrow \boldsymbol{p}_i(t) \oplus_{n_{i,J_i}} \{J_i\}$
    $y_{i,J_i}(t) \leftarrow c_{i,J_i}$, $z_{i,J_i}(t) \leftarrow i$
  end while
end procedure

Algorithm 3 CBBA Phase 2: agent i's action for task j

1: Local communication exchanging $(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{s})$ with agent k
2: Decide action by update rules shown in [12]
3: 1) update : $y_{ij} \leftarrow y_{kj}$, $z_{ij} \leftarrow z_{kj}$
4: 2) reset : $y_{ij} \leftarrow 0$, $z_{ij} \leftarrow 0$
5: 3) leave : $y_{ij} \leftarrow y_{ij}$, $z_{ij} \leftarrow z_{ij}$

hostile relationship between the robots and the threats with m_{k_i,k_j} representing the relative strength of the agent i when confronting target j. $d_{a,b}$ denotes the distance from robot a to threat b or threat a to threat b. For better performance related to the increase in the team survivability, the scoring matrix $\boldsymbol{M}$ has to be selected optimally.

In the conflict resolution phase, three vectors are communicated for consensus. Two are the winning bid list $\boldsymbol{y}_i$ and the winning agent list $\boldsymbol{z}_i$, and the third vector $\boldsymbol{s}_i \in \mathbb{R}^{N_u}$ represents the time stamp of the last information updated from each of the other agents. When agent i receives a message from agent k, $\boldsymbol{z}_i$ and $\boldsymbol{s}_i$ are used to determine which agent's information contains the most recent data for each task. There are three possible actions agent i can do on task j as shown in lines 3–5 of Algorithm 3. Detailed action rules during communication are given in [12].

CBBA process with a synchronized conflict resolution phase over a static communication network with diameter D guarantees at least 50% optimality in performance with in $N_{\min}D$ convergence time, where $N_{\min} = \min\{L_t N_u, N_t\}$. As mentioned in Sect. 2.2, all the robots

are fully connected in communication. However, if some robots are destroyed, then they lose connections and the remaining robots acquire new mission points after performing CBBA except the robots that lost mission capability.

3.2 Optimization for Mission Assignment

The scoring matrix presented in Sect. 3.1 requires the proper selection of each element. This section describes reinforcement learning and particle swarm optimization to learn the best values for the elements of the scoring matrix.

3.2.1 Reinforcement learning

Reinforcement learning can provide solutions in the situation involing the probabilistic model for the attack [54]. For a specific set of the scoring matrices, the performance of each scoring matrix is evaluated by the expected return. Then their suboptimal values of elements of the scoring matrix will be determined using episodic optimization.

The target assignment depends on the scoring matrix $\boldsymbol{M}$ can be considered a function of $\boldsymbol{M}$, so the assignment policy is denoted by $\pi(\boldsymbol{M})$. In the considering scenario, the main purpose of the robots is to maximize the team survivability during the engagement by using the policy $\pi(\boldsymbol{M})$. Given initial conditions of an engagement, the expected return $V^{\pi(\boldsymbol{M})}$ is assigned to $\pi(\boldsymbol{M})$:

$$V^{\pi(\boldsymbol{M})} = E[R|\pi(\boldsymbol{M})] \tag{3.7}$$

where the random variable R denotes the return and is defined by the ratio of the survived robots:

$$R = \frac{N_s}{N_u} \tag{3.8}$$

where N_s is the number of the survived robots.

Because the exact evaluation of (3.7) is intractable [54], instead the average total reward

Table 3.1: Inputs of the EPO algorithm

Parameter	Explanation	Value
S	total number of particles ($1 < s < S$)	10
lb	lower bound of m_{k_i,k_j}	0
ub	upper bound of m_{k_i,k_j}	10
N_{ep}	total number of episodes ($1 < n_{ep} < N_{ep}$)	100
$iter_{\max}$	maximum iteration ($1 \leq iter \leq iter_{\max}$)	50

over the N_{ep} repeated runs is used by the following:

$$\bar{R} = \frac{1}{N_{ep}} \sum_{n_{ep}=1}^{N_{ep}} R^{(n_{ep})}. \tag{3.9}$$

Here, N_{ep} is the total number of monte-carlo type simulations called episodes. At the end of each n_{ep}-th episode, the return $R^{(n_{ep})}$ is obtained. This process is repeated $n_{ep} = 1, \ldots, N_{ep}$ times, then the average $\bar{R}$ is computed according to (3.9) as shown in lines 7–13 of Algorithm 4. This $\bar{R}$ will give an estimated performance for the particular values of elements of scoring matrix $\boldsymbol{M}$. The performance estimates $\bar{R}'$ for a different scoring matrix $\boldsymbol{M}'$ can be computed using the same process, and this process is repeated until convergence to the best values of $\boldsymbol{M}$. To guarantee the sub-optimal property of the resulting solution, the total number of episodes N_{ep} should be large enough (roughly speaking, some polynomial of the *complexity* of the problem described in [42]), and the *same* set of random elements should be used to evaluate the estimated performance of a scoring matrix. In the considered problem, the average total reward (3.9), i.e., the team survivability, should be maximized for a candidate scoring matrix.

3.2.2 Particle swarm optimization

To solve the parameter optimization problem, particle swarm optimization (PSO) is employed. PSO is a population-based stochastic optimization technique [27]. Every particle in the population travels in the search space looking for the global minimum (or maximum) similar to the behavior of bird flocking. Each particle adjusts its velocity according to its own experience and its swarm's experience while particles search for the global solution.

To determine the optimal scoring matrix $\boldsymbol{M}^*$, the particles are regarded as the candidate scoring matrices. All the particles in the population which begin with a random position $\boldsymbol{M}_s$ and random velocity $\boldsymbol{\nu}_s \in \mathbb{R}^{3\times 3}$, $s = 1, \ldots, S$ where S is the swarm size, are candidate solutions and iteratively move in the problem space. The best previous position of the particle s is remembered and represented as $\boldsymbol{pBest}_s \in \mathbb{R}^{3\times 3}$. The position of the best particle among all the particles is represented as $\boldsymbol{gBest} \in \mathbb{R}^{3\times 3}$. At each iteration, the velocity $\boldsymbol{\nu}_s$ and position $\boldsymbol{M}_s$ of each particle s can be updated by the following:

$$\begin{aligned} \boldsymbol{\nu}_s &= K[\boldsymbol{\nu}_s + c_1 r_1(\boldsymbol{pBest}_s - \boldsymbol{M}_s) + c_2 r_2(\boldsymbol{gBest} - \boldsymbol{M}_s)] \\ \boldsymbol{M}_s &= \boldsymbol{M}_s + \boldsymbol{\nu}_s \end{aligned} \tag{3.10}$$

where c_1 and c_2 are the acceleration constants, and r_1 and r_2 are chosen as uniform random values in the range $[0, 1]$, K is the constriction factor to ensure the convergence of PSO [15], and it is determined by

$$K = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|} \tag{3.11}$$

where $\varphi = c_1 + c_2, \varphi > 4$. Typically, φ is set to 4.1. Then, the value function $V^{\pi(\boldsymbol{M}_s)}$ determined by the scoring matrix $\boldsymbol{M}_s$ for each particle s can be evaluated in each PSO iteration *iter*. The details of this optimization process are shown in Algorithm 4 and Fig. 3.1, and the inputs of the EPO algorithm are set as shown in Table 3.1.

Algorithm 4 Episodic Parameter Optimization (EPO) algorithm

```
procedure EPISODIC PARAMETER OPTIMIZATION
    Setup N_ep initial conditions of the engagement
    Initialize particles (M_s, s = 1, ..., S) with random position and velocity matrices
    Initialize the pBest_s and the value function for each particle s, i.e., pBest_s ← M_s
    and R̄_s^(0) ← −∞
    for iter = 1 to iter_max do
        for each s = 1, ..., S do
            for n_ep ≤ N_ep do
                Start from the n_ep-th initial conditions
                Run episode n_ep of the engagement
                Update the rewards R_s^(n_ep)
                n_ep ← n_ep + 1
            end for
            Compute R̄_s^(iter) using (3.9)
            if R̄_s^(iter) > R̄_s^(iter−1) then
                pBest_s ← M_s
            else
                R̄_s^(iter) ← R̄_s^(iter−1)
            end if
            V^π(M_s) ← R̄_s^(iter)
        end for
        Set the best of pBests as gBest, i.e., gBest ← arg max_{M_s} V^π(M_s), s = 1, ..., S
        Update each particle's velocity and position by (3.10)
    end for
    return M* ← gBest
end procedure
```

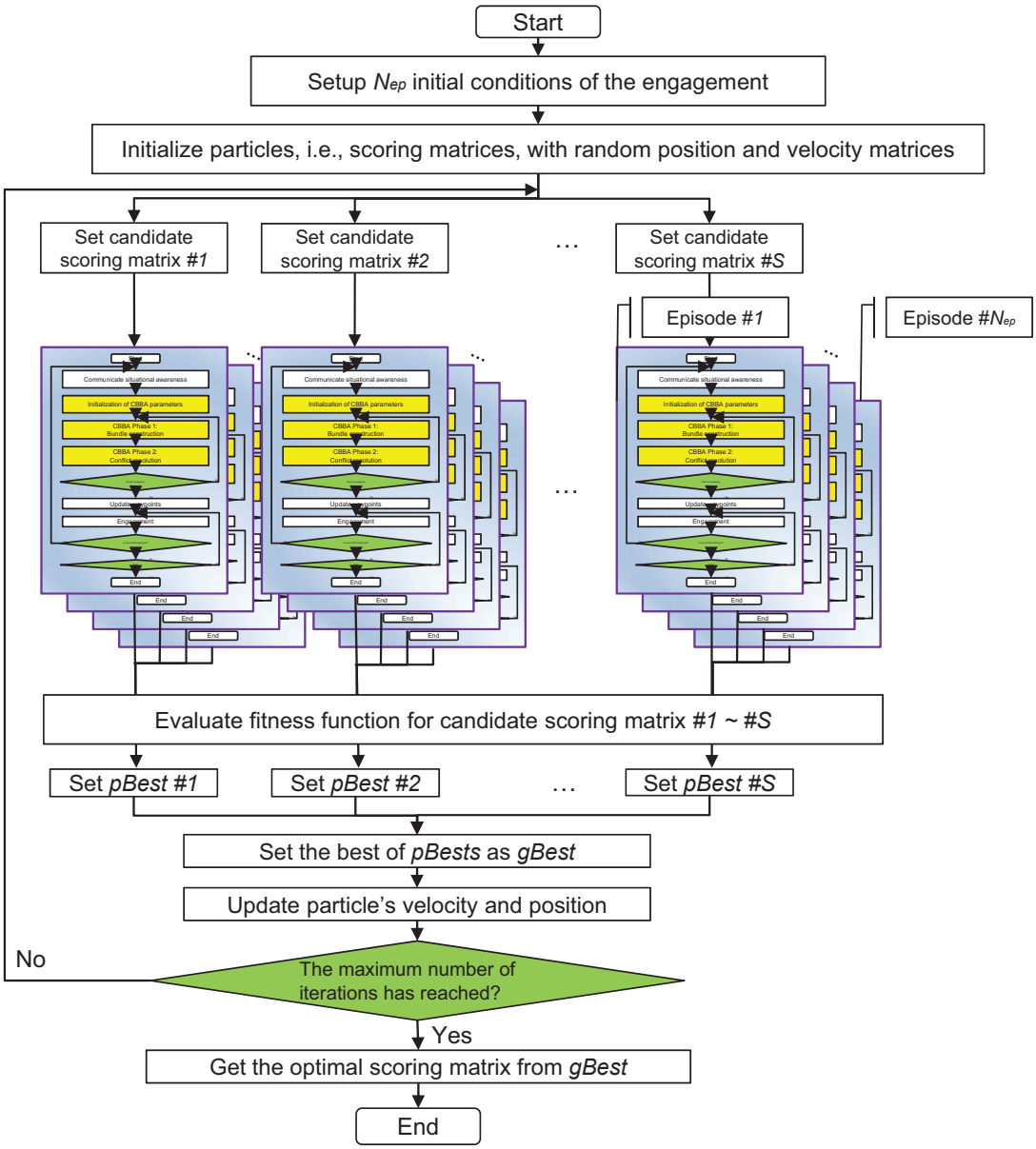


Figure 3.1: Flow chart of the EPO algorithm

3.3 Optimization Results

In this section, the EPO algorithm is applied to optimize the scoring matrix $\boldsymbol{M}$ which is involved in CBBA, and the performance of the proposed algorithm is analyzed.

For a 30 km × 30 km environment, total number of ground robots N_u is fixed at 5, and the combination of the robots is set as one robot of type 1, two robots of type 2, and two robots of type 3. The number of threats N_t is set to 20. Before the start of the EPO algorithm, N_{ep} samples of initial conditions about the locations and types of threats are uniformly randomly selected, and these initial conditions are applied for each episode.

Figure 3.2 shows change in the team survivability as the number of iterations increases. At each EPO iteration, ten candidate scoring matrices are selected and the team survivability for each candidate is evaluated as shown in Fig. 3.2. The result shows tendency in which the survivability improves after a few iterations. After ten iterations of the EPO algorithm, the average of the team survivability for the entire candidate scoring matrices increases from 20% to 70% compared to the first iteration. The optimized scoring matrix is determined after all the number of iterations as follows:

$$\boldsymbol{M}^* = \begin{bmatrix} 3.48517 & 0 & 0 \\ 0.464493 & 4.23875 & 0 \\ 3.15284 & 4.93702 & 9.29547 \end{bmatrix}.$$

As a result, the EPO algorithm presents the positive effect to enhance the team survivability by optimizing the scoring matrix. The optimal scoring matrix $\boldsymbol{M}^*$ tells the validity of this conclusion. As mentioned in Sect. 3.1, the rows of the scoring matrix $\boldsymbol{M}$ are related to the type of robots and the columns are for the type of threats. From the optimized scoring matrix $\boldsymbol{M}^*$, $m_{1,2}$ and $m_{1,3}$ are close to 0. This result means that the robot of type 1 will not choose the threat of type 2 or type 3 and satisfies our assumption in which the robot of type 1 can only handle the threat of type 1. The other example is that the value of $m_{3,3}$ is greater than the other elements because the robot of type 3 can only deal with

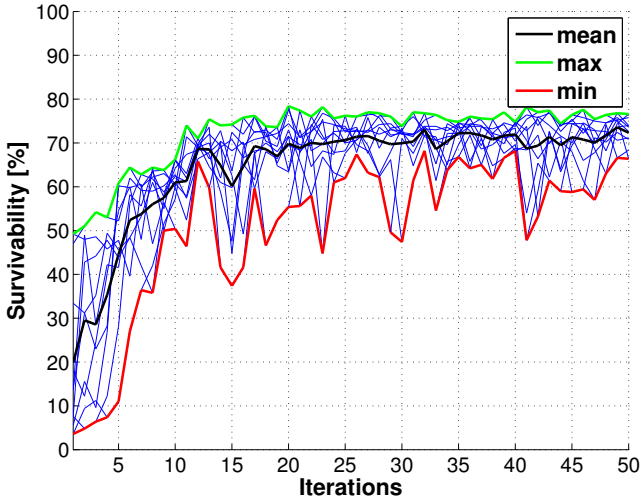


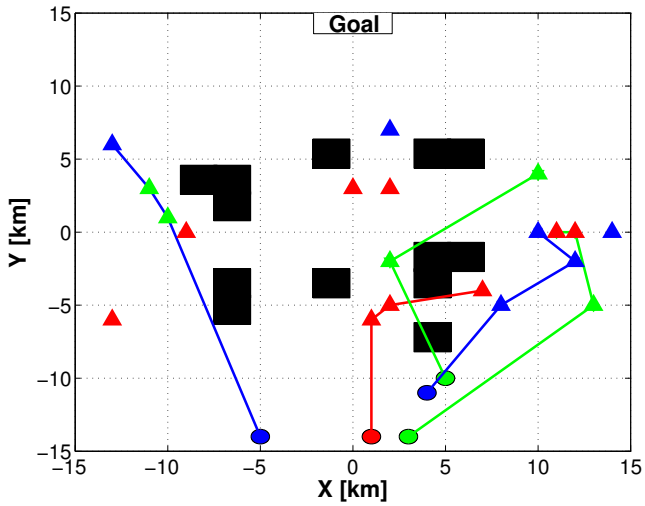
Figure 3.2: Optimization result

the threat of type 3. $m_{3,1}$ and $m_{3,2}$ also have the big values due to the excellent attack capabilities for the robot of type 3. Therefore, we can conclude that the EPO algorithm properly optimizes the scoring matrix.

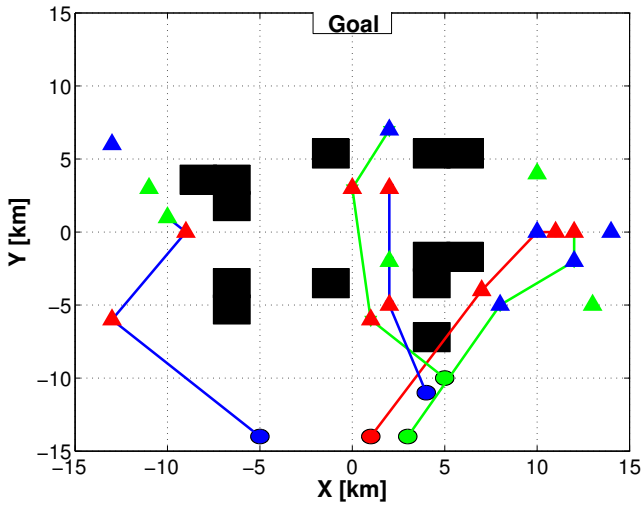
3.4 Analysis and Discussion

In the previous section, we obatined the optimal scoring matrix by using the EPO algorithm. Here, we will observe the influence of the optimal scoring matrix in mission planning.

Figure 3.3 shows task allocation results with respect to the different value of $S_i^{\boldsymbol{p}_i}$ in (3.5). When $S_i^{\boldsymbol{p}_i}$ is determined by considering both distance and the optimal scoring matrix, each robot selects the threats which are near and favorable to attack depending on its type without task conflict as shown in Fig. 3.3(a). Especially, the robot of type 3 prefers to select the threat of type 3 because the element $m_{3,3}$ of the optimal scoring matrix has the high value. On the other hand, when the reward for distance is only considered in CBBA, we can observe that the assigned threats are close to each relevant robot but three robots select improper target as shown in Fig. 3.3(b). So, we cannot ensure its survivability. Thus, we can grasp the fact that CBBA with the optimized scoring matrix can allocate the heterogeneous targets to the robots by considering the robots' attack capabilities. We expect that the proposed EPO algorithm can be utilized to handle other probabilistic situations where it is difficult to quantify performance metrics.



(a) Distance and scoring matrix $\boldsymbol{M}^*$



(b) Only distance

Figure 3.3: Task allocation results with respect to the different value of $S_i^{\boldsymbol{p}_i}$ (red: $k_i, k_j = 1$, green: $k_i, k_j = 2$, blue: $k_i, k_j = 3$)

4

Multi-Robot Path Planning

In the previous chapter, we obtained the optimal engagement strategy in the mission planning domain, but there was no consideration of path planning. To handle this problem, we propose a decentralized trajectory optimization algorithm called decentralized VMCPSO. We especially focus on solving the nonlinear constrained trajectory optimization problem with the terminal time and angle constraints in the multi-robot domain. The idea of VMCPSO comes from virtual motion camouflage (VMC) and particle swarm optimization (PSO). VMC transforms a typical full space optimal problem to a subspace optimal problem, and this reduces the dimension of the problem by using path control parameters (PCPs). If PCPs are optimized, then the optimal path is generated. In this work, we employ PSO to optimize these PCPs. In PSO, the candidate PCPs find the optimal solution during local and global interactions with the other candidates. In multi-robot path planning, each robot generates its own optimal path by using VMCPSO and sends the path information to the other robots. Then, the other robots use this path information when planning their own paths. The proposed algorithms will be validated with simulations and an experiment.

4.1 Virtual Motion Camouflage

In this section, a standard optimal trajectory generation problem with nonlinear constraints is considered, and a bio-inspired method is described to solve this problem. In addition, the proposed approach will be discussed.

4.1.1 Nonlinear constrained trajectory optimization problem

The general nonlinear constrained trajectory optimization problem is defined by the following:

$$J = \Phi[\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), t_0, t_f] + \int_{t_0}^{t_f} \mathcal{L}[\boldsymbol{x}(t), \boldsymbol{u}(t), t]dt \tag{4.1}$$

subject to the inequality constraints

$$\boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \leq 0, \boldsymbol{g} \in \mathbb{R}^{p\times 1} \tag{4.2}$$

and the equality constraints

$$\boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) = 0, \boldsymbol{h} \in \mathbb{R}^{q\times 1}. \tag{4.3}$$

Here, $(\boldsymbol{x}(t), \boldsymbol{u}(t), t, t_0, t_f)$ are the state, control, current time, initial time, and final time, respectively. The equality constraints (4.3) include the terminal constraints

$$\boldsymbol{\Psi}[\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), t_0, t_f] = 0, \boldsymbol{\Psi} \in \mathbb{R}^{l\times 1} \tag{4.4}$$

and the first-order dynamic constraints

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t), \boldsymbol{x} \in \mathbb{R}^{n\times 1}, \boldsymbol{u} \in \mathbb{R}^{m\times 1}. \tag{4.5}$$

The optimal trajectory will be found by maximizing (or minimizing) the performance index J. The solution of this optimal problem might be locally or globally optimal depending on

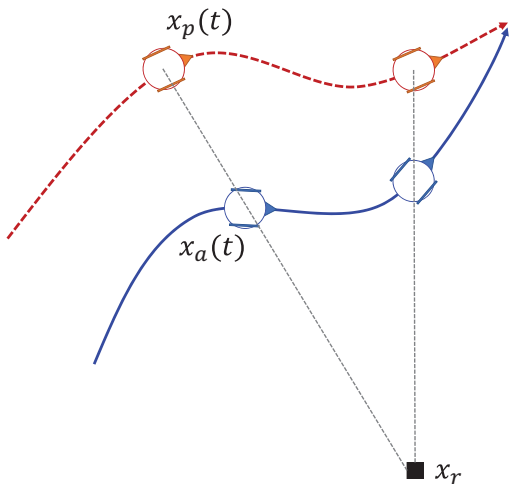


Figure 4.1: Relationship between the reference point, the prey motion, and the aggressor motion.

optimization methods.

4.1.2 VMC formulation

Srinivasan and Davey introduced *Motion Camouflage* (MC) for describing whether an agent can actively camouflage its motion while tracking a target, and developed algorithms to determine the agent's trajectories for a stationary or moving target [51]. In [63, 64, 65], two moving agents, an aggressor and a prey, are considered, and the idea of MC is applied to the nonlinear constrained optimal control problem. As shown in Fig. 4.1, the path of the aggressor $\boldsymbol{x}_a(t)$ related to the motion of the prey $\boldsymbol{x}_p(t)$ and the reference point $\boldsymbol{x}_r$ can be controlled by the path control parameter (PCP) $\lambda(t)$ as follows:

$$\boldsymbol{x}_a(t) = \boldsymbol{x}_r + \lambda(t)(\boldsymbol{x}_p(t) - \boldsymbol{x}_r) \tag{4.6}$$

and the derivatives of $\boldsymbol{x}_a(t)$ can be simply defined by the following:

$$\dot{\boldsymbol{x}}_a(t) = \dot{\boldsymbol{x}}_r + \dot{\lambda}(t)(\boldsymbol{x}_p(t) - \boldsymbol{x}_r) + \lambda(t)(\dot{\boldsymbol{x}}_p(t) - \dot{\boldsymbol{x}}_r) \tag{4.7}$$

$$\ddot{\boldsymbol{x}}_a(t) = \ddot{\boldsymbol{x}}_r + \ddot{\lambda}(t)(\boldsymbol{x}_p(t) - \boldsymbol{x}_r) + \lambda(t)(\ddot{\boldsymbol{x}}_p(t) - \ddot{\boldsymbol{x}}_r) + 2\dot{\lambda}(t)(\dot{\boldsymbol{x}}_p(t) - \dot{\boldsymbol{x}}_r). \tag{4.8}$$

Before the VMC formulation, the following two assumptions are made as in [66].

Assumption 1 *([66]). The state vector $\boldsymbol{x}(t) \in \mathbb{R}^{n\times 1}$ can be rearranged into two parts: the position state $\boldsymbol{x}_a(t) \in \mathbb{R}^{n_a\times 1}$ and the state rate $\boldsymbol{x}_{sr}(t) \in \mathbb{R}^{(n-n_a)\times 1}$. Correspondingly, the equations of motion $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t)$ can be rewritten into two parts: $\dot{\boldsymbol{x}}_a(t) = \boldsymbol{f}_a(\boldsymbol{x}(t), t)$ and $\dot{\boldsymbol{x}}_{sr}(t) = \boldsymbol{f}_{sr}(\boldsymbol{x}(t), \boldsymbol{u}(t), t)$.*

Assumption 2 *([66]). The mappings from $(\boldsymbol{x}_a(t), \dot{\boldsymbol{x}}_a(t), t)$ to $\boldsymbol{x}_{sr}(t)$ and from $(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), t)$ to $\boldsymbol{u}(t)$ are assumed to be injective, which means the control variables $\boldsymbol{u}(t)$ and the state rate $\boldsymbol{x}_{sr}(t)$ can be solved by $\boldsymbol{x}_{sr}(t) = \boldsymbol{f}_a^{-1}(\boldsymbol{x}_a(t), \dot{\boldsymbol{x}}_a(t), t)$ and $\boldsymbol{u}(t) = \boldsymbol{f}_{sr}^{-1}(\boldsymbol{x}(t), \dot{\boldsymbol{x}}_{sr}(t), t)$ either explicitly or implicitly using an iterative fashion.*

The nonlinear constrained trajectory optimization problem stated in Sect. 4.1.1 can be reformulated by considering Assumptions 1 and 2, and (4.6)-(4.8). Given $\boldsymbol{x}_r$ and $\boldsymbol{x}_p(t)$, the variables $\lambda(t), \dot{\lambda}(t), \ddot{\lambda}(t), \ldots$, and t_f will be designed to minimize the performance index

$$J = \Phi[\lambda(t), \dot{\lambda}(t), \ddot{\lambda}(t), \ldots, t_f] + \int_{t_0}^{t_f} \mathcal{L}[\lambda(t), \dot{\lambda}(t), \ddot{\lambda}(t), \ldots, t]dt \tag{4.9}$$

subject to the state and control inequality constraints

$$\boldsymbol{g}(\lambda(t), \dot{\lambda}(t), \ddot{\lambda}(t), \ldots, t) \leq 0, \boldsymbol{g} \in \mathbb{R}^{p\times 1} \tag{4.10}$$

and the equality constraints

$$\boldsymbol{h}(\lambda(t), \dot{\lambda}(t), \ddot{\lambda}(t), \ldots, t) = 0, \boldsymbol{h} \in \mathbb{R}^{l\times 1} \tag{4.11}$$

In the VMC formula, the first-order dynamics constraints (4.5) are already taken into account when calculating $\boldsymbol{x}_{sr}(t)$ and $\boldsymbol{u}(t)$ based on Assumption 2, so the boundary conditions are only considered as the equality constraints.

To obtain the numerical solution, the VMC-based nonlinear constrained trajectory optimization problem is directly formulated as a nonlinear programming via a collocation method based on the pseudo-spectral scheme. The Legendre-Gauss-Lobatto (LGL) pseudo-spectral scheme is used to discretize the PCP $\lambda(t)$ into the PCP nodes λ_k, $k = 0, \ldots, N$, and $\boldsymbol{\lambda} = [\lambda_0, \ldots, \lambda_N]^\top$ is the vector of the PCP nodes. Therefore, the discretized performance index can be described as follows:

$$J = \Phi[\lambda_N] + \left(\frac{t_f - t_0}{2}\right) \sum_{k=0}^{N} \mathcal{L}[\boldsymbol{\lambda}, \dot{\boldsymbol{\lambda}}, \ddot{\boldsymbol{\lambda}}, \ldots]\omega_k \tag{4.12}$$

where ω_k is the weight for the k-th LGL node. The state/control inequality and equality constraints can be written as

$$\boldsymbol{g}_k(\boldsymbol{\lambda}, \dot{\boldsymbol{\lambda}}, \ddot{\boldsymbol{\lambda}}, \ldots) \leq 0 \tag{4.13}$$

$$\boldsymbol{h}_k(\boldsymbol{\lambda}, \dot{\boldsymbol{\lambda}}, \ddot{\boldsymbol{\lambda}}, \ldots) = 0 \tag{4.14}$$

where the derivatives of the PCP vector are

$$\frac{d^k \boldsymbol{\lambda}}{dt^k} = \left(\frac{2}{t_f - t_0}\right)^k \boldsymbol{D}^k \boldsymbol{\lambda} \tag{4.15}$$

where the differentiation matrix $\boldsymbol{D}$ can be found in [20]. Then, (4.6)–(4.8) are rewritten by the following:

$$\boldsymbol{x}_a(k) = \boldsymbol{x}_r + \lambda_k(\boldsymbol{x}_p(k) - \boldsymbol{x}_r) \tag{4.16}$$

$$\dot{\boldsymbol{x}}_a(k) = \dot{\boldsymbol{x}}_r + \dot{\lambda}_k(\boldsymbol{x}_p(k) - \boldsymbol{x}_r) + \lambda_k(\dot{\boldsymbol{x}}_p(k) - \dot{\boldsymbol{x}}_r) \tag{4.17}$$

$$\ddot{\boldsymbol{x}}_a(k) = \ddot{\boldsymbol{x}}_r + \ddot{\lambda}_k(\boldsymbol{x}_p(k) - \boldsymbol{x}_r) + \lambda_k(\ddot{\boldsymbol{x}}_p(k) - \ddot{\boldsymbol{x}}_r) + 2\dot{\lambda}_k(\dot{\boldsymbol{x}}_p(k) - \dot{\boldsymbol{x}}_r). \tag{4.18}$$

The reference point $\boldsymbol{x}_r$ will remain fixed, so (4.17) and (4.18) can be simply rewritten as follows:

$$\dot{\boldsymbol{x}}_a(k) = \dot{\lambda}_k(\boldsymbol{x}_p(k) - \boldsymbol{x}_r) + \lambda_k \dot{\boldsymbol{x}}_p(k) \tag{4.19}$$

$$\ddot{\boldsymbol{x}}_a(k) = \ddot{\lambda}_k(\boldsymbol{x}_p(k) - \boldsymbol{x}_r) + \lambda_k \ddot{\boldsymbol{x}}_p(k) + 2\dot{\lambda}_k \dot{\boldsymbol{x}}_p(k). \tag{4.20}$$

Here, the selection of the reference point $\boldsymbol{x}_r$ is very important because it is closely related to the problem space. For example, if we consider the minimum-time problem as described in Fig. 4.4(a), a prey motion can be determined by a straight line. Then, if we select a reference point on the same line as the prey motion, the aggressor motion $\boldsymbol{x}_a$ is constrained on the line of the prey motion no matter how we change the PCP vector $\boldsymbol{\lambda}$. In the planar motion, the most appropriate reference point of this problem can be selected on the line which is perpendicular to the center point of the prey motion.

In [66], the sequential VMC method involving two steps in an iteration process is proposed to solve this problem. In the first step, an optimal solution can be found within subspace constructed by the PCP nodes, and then a linear programming and a line search algorithm are used in the second step to improve the PCP nodes. The detailed algorithms and results can be found in [66].

4.1.3 The proposed approach

In our approach, the constrained optimization problem should be changed into an unconstrained optimization problem as follows:

$$\text{minimize}_{(\boldsymbol{\lambda})} \;\; J = \left(\frac{t_f - t_0}{2}\right) \sum_{k=0}^{N} \mathcal{L}[\boldsymbol{\lambda}, \dot{\boldsymbol{\lambda}}, \ddot{\boldsymbol{\lambda}}, \ldots]\omega_k + C \sum_{k=0}^{N} \max_k \Big(0, \boldsymbol{g}_k(\boldsymbol{\lambda}, \dot{\boldsymbol{\lambda}}, \ddot{\boldsymbol{\lambda}}, \ldots)\Big) \tag{4.21}$$

where the second term of (4.21) is the penalty function to satisfy the inequality constraints, and C is the weighting parameter. Here, the boundary conditions such as the initial and final position can be included in (4.21) by setting $\lambda_0 = 1$ and $\lambda_N = 1$. As stated in Sect.

4.1.2, the most important task to find the optimal trajectory is how to determine the PCP nodes optimally. Thus, to deal with this parameter optimization problem, particle swarm optimization (PSO) mentioned in Sect. 3.2.2 is employed again.

Let the particles be the candidate PCP vectors, then each particle begins with randomly selected position $\boldsymbol{\lambda}_s \in \mathbb{R}^{(N+1)\times 1}$ and zero velocity $\boldsymbol{\nu}_s \in \mathbb{R}^{(N+1)\times 1}(s = 1, 2 \ldots, S)$, where S is the swarm size, and iteratively moves in the problem space. To generate reliable candidate particles, i.e., the PCP vector $\boldsymbol{\lambda}_s$, each candidate PCP node $\lambda_{k,s}$ is computed with the combination of M harmonic functions as follows:

$$\lambda_{k,s} = 1 + r_0 \sum_{m=0}^{M} A \sin\left(m\pi\left(\frac{\omega_{\langle 0:k\rangle}}{\omega_{\langle 0:N\rangle}}\right)\right) \tag{4.22}$$

where $\omega_{\langle a:b\rangle}$ denotes $\sum_{i=a}^{b} \omega_i$, and A is constant to handle the range of the candidate paths. r_0 is chosen as uniform random variables in the range $[-1, 1]$. If we can properly generate a set of candidate paths according to the problem domain, the reliability of optimization results can be increased. The best previous position of the particle s is remembered and represented as $\boldsymbol{pBest}_s \in \mathbb{R}^{(N+1)\times 1}$. The position of the best particle among all the particles is represented as $\boldsymbol{gBest} \in \mathbb{R}^{(N+1)\times 1}$. Similar to (3.10), at each iteration the velocity $\boldsymbol{\nu}_s$ and position $\boldsymbol{\lambda}_s$ of each particle s can be updated by the following:

$$\begin{aligned} \boldsymbol{\nu}_s &= K[\boldsymbol{\nu}_s + c_1 r_1(\boldsymbol{pBest}_s - \boldsymbol{\lambda}_s) + c_2 r_2(\boldsymbol{gBest} - \boldsymbol{\lambda}_s)] \\ \boldsymbol{\lambda}_s &= \boldsymbol{\lambda}_s + \boldsymbol{\nu}_s. \end{aligned} \tag{4.23}$$

The proposed algorithm using VMC and PSO (VMCPSO) is detailed in Algorithm 5.

In order to make the robot follow the optimal trajectory, we can obtain the path conditions $(v_d, w_d, x_d, y_d, \theta_d)$ as described in lines 11–33 of Algorithm 6. Here, $t_{\text{est}}(k)$ is the estimated time of arrival at k-th PCP node and Δt denotes the time step. The PCP node number k and path index p are initially set to 0 (line 4 of Algorithm 6). Then, we can compute the velocity input vector $\boldsymbol{u}(t) = [v(t), w(t)]^\top$.

Algorithm 5 Virtual Motion Camouflage and Particle Swarm Optimization (VMCPSO) algorithm

procedure VMCPSO($N, iter_{\max}, S$)
  Create the $(N+1)$ LGL nodes with the weight vector $\boldsymbol{\omega}$
  Create the differentiation matrix $\boldsymbol{D}$
  Set the virtual prey motion $\boldsymbol{x}_p$ and the reference point $\boldsymbol{x}_r$
  Initialize particles $(\boldsymbol{\lambda}_s, s = 1, \ldots, S)$ with random position (4.22) and zero velocity
  Initialize the $\boldsymbol{pBest}_s$ and the cost J_s for each particle s, i.e., $\boldsymbol{pBest}_s \leftarrow \boldsymbol{\lambda}_s$ and $J_s^{(0)} \leftarrow \infty$
  for $iter = 1$ to $iter_{\max}$ **do**
    for each $s = 1, \ldots, S$ **do**
      Compute $(\dot{\boldsymbol{\lambda}}_s, \ddot{\boldsymbol{\lambda}}_s)$ and $(\boldsymbol{x}_a, \dot{\boldsymbol{x}}_a, \ddot{\boldsymbol{x}}_a)$ using (4.15)-(4.20)
      Compute $J_s^{(iter)}$ using (4.21)
      if $J_s^{(iter)} < J_s^{(iter-1)}$ **then**
        $\boldsymbol{pBest}_s \leftarrow \boldsymbol{\lambda}_s$
      else
        $J_s^{(iter)} \leftarrow J_s^{(iter-1)}$
      end if
      $J(\boldsymbol{\lambda}_s) \leftarrow J_s^{(iter)}$
    end for
    Set the best of $\boldsymbol{pBest}s$ as $\boldsymbol{gBest}$, i.e., $\boldsymbol{gBest} \leftarrow \arg\min_{\boldsymbol{\lambda}_s} J(\boldsymbol{\lambda}_s), s = 1, \ldots, S$
    Update each particle's velocity and position by (4.23)
  end for
  return $\boldsymbol{\lambda}^* \leftarrow \boldsymbol{gBest}$
end procedure

Algorithm 6 Overall path planning and tracking algorithm

procedure VMCPSO & PATH TRACKING
  $\boldsymbol{\lambda}^* \leftarrow \text{VMCPSO}(N, iter_{\max}, S)$
  Compute $(\boldsymbol{x}_a^*(k), v^*(k), w^*(k))$ by using $\left(\boldsymbol{\lambda}^*, \dot{\boldsymbol{\lambda}}^*, \ddot{\boldsymbol{\lambda}}^*, \boldsymbol{x}_p(k), \boldsymbol{x}_r, t_f\right), \forall k = 0, \ldots, N$
  $k \leftarrow 0, p \leftarrow 0$
  for every time step $t = 0, \Delta t, \ldots$ **do**
    $n \leftarrow \lfloor (t_{\text{est}}(k+1) - t_{\text{est}}(k)) / \Delta t \rfloor$
    $(v_d(t), w_d(t), x_d(t), y_d(t), \theta_d(t), k, p) \leftarrow$ SET DESIRED PATH CONDITIONS (n, k, p)
    Compute $(v(t), w(t))$ and update robot's position/orientation
  end for
end procedure
procedure SET DESIRED PATH CONDITIONS(n, k, p)
  if $n \neq 0$ **then**
    if $p \geq n$ **then**
      $k \leftarrow k + 1, p \leftarrow 0$
      $(v_d, w_d) \leftarrow (v^*(k), w^*(k))$
    else
      $p \leftarrow p + 1$
      $(v_d, w_d) \leftarrow (v^*(k), 0)$
    end if
    $x_d \leftarrow x_a^*(k) + (x_a^*(k+1) - x_a^*(k))p/n$
    $y_d \leftarrow y_a^*(k) + (y_a^*(k+1) - y_a^*(k))p/n$
    $\theta_d \leftarrow \text{atan2}(y_a^*(k+1) - y_a^*(k), x_a^*(k+1) - x_a^*(k))$
  else
    $k' \leftarrow 0, p \leftarrow 0$
    while $n = 0$ **do**
      $k \leftarrow k + 1, k' \leftarrow k' + 1$
      $n \leftarrow \lfloor (t_{\text{est}}(k+1) - t_{\text{est}}(k)) / \Delta t \rfloor$
    end while
    $(v_d, w_d, x_d, y_d) \leftarrow (v^*(k - k'), w^*(k - k'), x_a^*(k), y_a^*(k))$
    $\theta_d \leftarrow \text{atan2}(y_a^*(k) - y_a^*(k - k'), x_a^*(k) - x_a^*(k - k'))$
  end if
  return $(v_d, w_d, x_d, y_d, \theta_d, k, p)$
end procedure

4.2 Extension to Multi-Robot Path Planning

Here, we suggest two types of algorithms for applying the VMCPSO algorithm to multi-robot path planning. The first type of the multi-robot path planning algorithms is developed for stationary targets, and the other type is for moving targets. Basically, we assume that the robots use a round-robin approach, cycling through a fixed planning order when there is no dynamically changing environment. For a dynamically changing environment, we can also consider the different planning order for the robots. Assume that there are several robots and they are fully connected in a communition network. If the path information $\mathcal{P}_j$ for the neighbors $j \in \mathcal{N}_i$ of the robot i is updated in the planning module of the robot i in real time, the robot i only needs to use it while planning its path. In other words, if there is no problem in collecting the path information of neighbors and each robot can make a decision fast enough, the different planning order does not matter too much because the VMCPSO method is good at finding the optimal path fast enough for given situations.

To handle the multi-robot path optimization problem for stationary targets, we extend the VMCPSO algorithm as decentralized VMCPSO (Dec-VMCPSO). Dec-VMCPSO includes the broadcast and receiving steps as shown in lines 3 and 5 of Algorithm 8. The robot i plans the optimal trajectory with the fixed planning order the same as the robot's index $i \in \{1, \ldots, N_u\}$. When the performance index is computed, the optimal path information of the other robots $j \in \mathcal{N}_i$ where $\mathcal{N}_i = \{j \in \{1, \ldots, N_u\} : j < i\}$ is used to generate the path for cooperation or avoiding collisions between the robots. After all the robots plan their own paths (lines 2–8 of Algorithm 8), each robot moves along the preplanned optimal path (lines 9–13 of Algorithm 8). From the merits of VMCPSO, the robot i can easily reconstruct its path information $\mathcal{P}_i$ or its neighbors' path information $\mathcal{P}_j$ by manipulating $\left(\boldsymbol{\lambda}^*, \dot{\boldsymbol{\lambda}}^*, \ddot{\boldsymbol{\lambda}}^*, \boldsymbol{x}_p(k), \boldsymbol{x}_r, t_f\right), k = 0, \ldots, N$, and it only requires a small amount of communication because the path information to be sent consists of $(\boldsymbol{\lambda}^*, \boldsymbol{x}_p(k), \boldsymbol{x}_r, t_f), k = 0, \ldots, N$. In this paper, we consider the reconstructed path information as the preplanned path points, i.e, aggressor motions $\boldsymbol{x}_a(k)$, and the estimated time of arrival $t_{\text{est}}(k), k = 0, \ldots, N$.

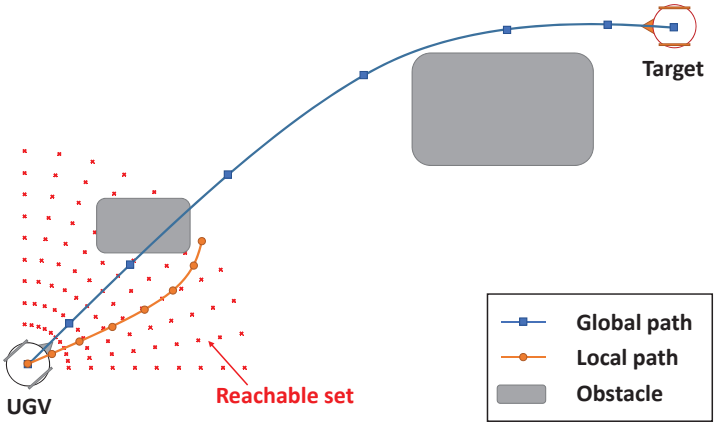


Figure 4.2: Example of local and global path planning

Therefore, we can obtain the optimal trajectories for all the robots according to given performance index without communication and computational load by using Dec-VMCPSO. Results to support this statement will be discussed in the next section.

To achieve the robots reach the moving target for given the terminal time and angles, each robot has to make a decision as soon as possible. The computation time of the VM-CPSO algorithm is important and closely related with the number of the PCP nodes and the candidate PCPs, and there is tradeoff between the computation time and reliability of the optimal path. If the number of the PCP nodes is reduced, the computation time can be decreased, but the gap between the PCP nodes will be widened. So, the PCP nodes will be sparsely located for the faraway target as shown in Fig. 4.2. Therefore, we adopt the concepts of global and local path planning. Global path planning (GPP) is mainly used to achieve the main objectives such as making cooperative behavior among robots. The terminal location of GPP is given by the target's position, and as the distance to the target is increased GPP cannot consider the small obstacle as shown in Fig. 4.2. So, local path planning (LPP) is employed to solve this problem. LPP plans the optimal path in the neighborhood of the robot's location to follow the global plan and to avoid collisions. In LPP, the terminal state is not specified. So, we define a reachable set $\mathcal{R}$, which is related to the

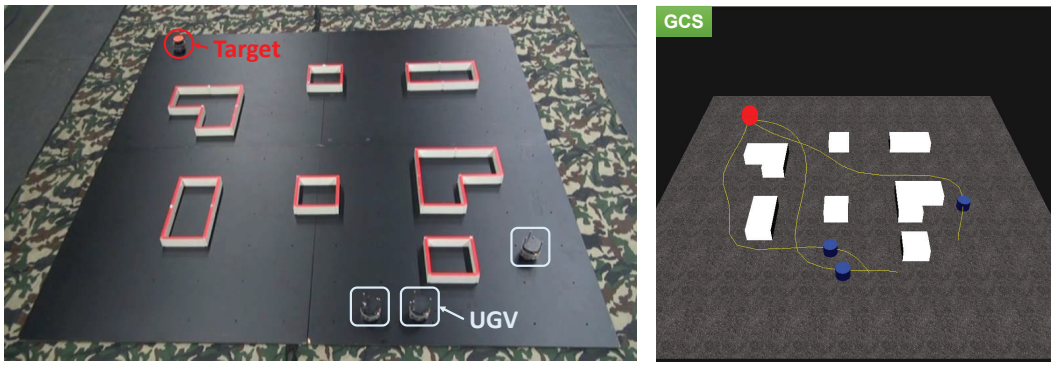
robot's position and orientation. Then, the terminal state $(x_f, y_f) \in \mathcal{R}$ can be obtained as described in Algorithm 7. At line 8 of Algorithm 7, the cost J_h for each candidate terminal state is computed by considering the planned global path and collision avoidance. When the most proper terminal state is decided, the optimal path for LPP is obtained by using the basic VMCPSO algorithm. In addition, if the target is located inside the reachable set, the terminal state of LPP is set the same as the location of the target, and the objective of LPP is changed to the same as the objective of GPP. As a result, GPP performs to find the optimal long-term path, which mainly satisfies the terminal time and angle constraints. LPP is used to find the optimal short-term path to follow the global path as soon as possible and to avoid collisions. Each robot generates the global/local path at every time step, and it only moves to the desired position on the local path at the next time step $t + \Delta t$ by using Dec-VMCPSO and the path tracking algorithm as described in lines 15—30 of Algorithm 8. Whenever its teammates change their positions or plans, if this information becomes available, GPP/LPP can reflect the new information immediately. Therefore, we can use the proposed algorithm as a reactive path planner.

Algorithm 7 Algorithm for setting a boundary point

```
1: procedure SET BOUNDARY POINT(x, y, θ)
2:     J_min ← ∞
3:     for m = 1 to (m_h + 1) do
4:         α ← α_min + (α_max − α_min) · (m − 1)/m_h
5:         for n = 1 to (n_h + 1) do
6:             x_h ← x + w_h · n · cos(α + θ)
7:             y_h ← y + w_h · n · sin(α + θ)
8:             Compute cost J_h of point (x_h, y_h)
9:             if J_h < J_min then
10:                J_min ← J_h
11:                (x_f, y_f) ← (x_h, y_h)
12:            end if
13:        end for
14:    end for
15:    return (x_f, y_f)
16: end procedure
```

Algorithm 8 Overall multi-robot path planning and tracking algorithm for the stationary and moving target

procedure DEC-VMCPSO & PATH TRACKING FOR STATIONARY TARGET
  for each robot i **do**
    Receive the path information $\mathcal{P}_j$ of prior robot $j \in \mathcal{N}_i$
    $\boldsymbol{\lambda}_i^* \leftarrow \text{VMCPSO}(N, iter_{\max}, S)$ by considering $\mathcal{P}_j$
    Broadcast the path information $\mathcal{P}_i$ of robot i
    Compute $\left(\boldsymbol{x}_{a,i}^*(k), v_i^*(k), w_i^*(k)\right)$ by using $\left(\boldsymbol{\lambda}_i^*, \dot{\boldsymbol{\lambda}}_i^*, \ddot{\boldsymbol{\lambda}}_i^*, \boldsymbol{x}_{p,i}(k), \boldsymbol{x}_{r,i}, t_{f,i}\right), \forall k = 0, \ldots, N$
    $k_i \leftarrow 0, p_i \leftarrow 0$
  end for
  for each robot i, every time step $t = 0, \Delta t, \ldots$ **do**
    $n_i \leftarrow \lfloor (t_{\text{est},i}(k_i + 1) - t_{\text{est},i}(k_i)) / \Delta t \rfloor$
    $(v_{d,i}(t), w_{d,i}(t), x_{d,i}(t), y_{d,i}(t), \theta_{d,i}(t), k_i, p_i) \leftarrow$ SET DESIRED PATH CONDITIONS (n_i, k_i, p_i)
    Compute $(v_i(t), w_i(t))$ and update robot i's position/orientation
  end for
end procedure
procedure DEC-VMCPSO & PATH TRACKING FOR MOVING TARGET
  for each robot i, every time step $t = 0, \Delta t, \ldots$ **do**
    Receive the path information $\left(\mathcal{P}_j^g, \mathcal{P}_j^l\right)$ of robot $j \in \mathcal{N}_i$
    // Global Path Planning (GPP)
    $\boldsymbol{\lambda}_i^g \leftarrow \text{VMCPSO}(N, iter_{\max}, S)$ by considering $\mathcal{P}_j^g$
    // Local Path Planning (LPP)
    $(x_{f,i}(t), y_{f,i}(t)) \leftarrow$ SET BOUNDARY POINT $(x_i(t), y_i(t), \theta_i(t))$
    $\boldsymbol{\lambda}_i^l \leftarrow \text{VMCPSO}(N, iter_{\max}, S)$ by considering $\left(\mathcal{P}_i^g, \mathcal{P}_j^l\right)$
    Broadcast the path information $\left(\mathcal{P}_i^g, \mathcal{P}_i^l\right)$ of robot i
    Compute $\left(\boldsymbol{x}_{a,i}^*(k), v_i^*(k), w_i^*(k)\right)$ by using $\left(\boldsymbol{\lambda}_i^l, \dot{\boldsymbol{\lambda}}_i^l, \ddot{\boldsymbol{\lambda}}_i^l, \boldsymbol{x}_{p,i}(k), \boldsymbol{x}_{r,i}, t_{f,i}\right), \forall k = 0, \ldots, N$
    $k_i \leftarrow 0, p_i \leftarrow 0$
    $n_i \leftarrow \lfloor (t_{\text{est},i}(k_i + 1) - t_{\text{est},i}(k_i)) / \Delta t \rfloor$
    $(v_{d,i}(t), w_{d,i}(t), x_{d,i}(t), y_{d,i}(t), \theta_{d,i}(t), \cdot, \cdot) \leftarrow$ SET DESIRED PATH CONDITIONS (n_i, k_i, p_i)
    Compute $(v_i(t), w_i(t))$ and update robot i's position/orientation
  end for
end procedure



(a) (b)

Figure 4.3: Multi-robot operation in an urban-like environment (a) A snapshot of a rendezvous experiment in which multiple UGVs try to arrive at the target point simultaneously with the specific heading (b) Ground display

4.3 Simulation Results

This section describes simulation results when VMCPSO and Dec-VMCPSO are applied to the nonlinear constrained trajectory optimization problem with the terminal time and angle constraints for the stationary and moving target. Simulations are carried out under the same environment as described in Fig. 4.3(a).

4.3.1 Stationary target

As shown in Fig. 4.4(a), we first consider collision-free trajectory planning using the minimum-time criterion in an urban-like environment. In this problem, t_f is free and re-

Table 4.1: Simulation conditions

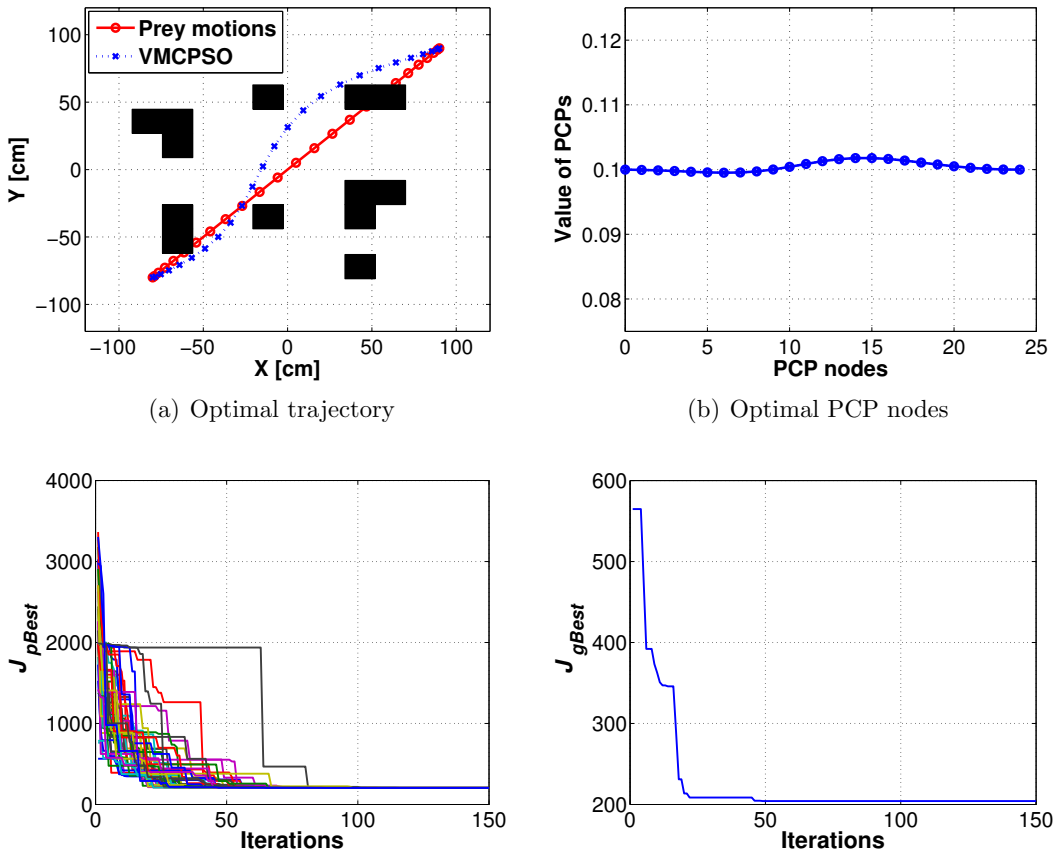
Parameter	Value	Parameter	Value
A	1	R_{obs}	16 cm
M	4	$C_1 \sim C_3$	100
N_{obs}	13	t_0	0
N	24	Δt	0.1 sec
S	200	w_h	10
$iter_{\max}$	150	m_h	10
$v_{\max}$	13 cm/s	n_h	20
$w_{\max}$	135 deg/s	$(\alpha_{\min}, \alpha_{\max})$	$(-\frac{\pi}{4}, \frac{\pi}{4})$

garded as cost to be optimized in VMCPSO.

$$
\begin{aligned}
\text{minimize} \quad & J = \left(\frac{t_f - t_0}{2}\right) \sum_{k=0}^{N} \omega_k \\
\text{subject to} \quad & |v(k)| \leq v_{\max} \\
& |w(k)| \leq w_{\max} \\
& (x(k) - x_{\mathrm{obs},i})^2 + (y(k) - y_{\mathrm{obs},i})^2 \geq R_{\mathrm{obs}}^2, i = 1, \ldots, N_{\mathrm{obs}} \\
& (x(0), y(0)) = (-80, -80) \\
& (x(N), y(N)) = (90, 90)
\end{aligned}
\tag{4.24}
$$

where R_{obs} is the safe radius of the square obstacle and N_{obs} is the number of obstacles.

Given (2.2), the position state $\boldsymbol{x}_a$ and state rate $\boldsymbol{x}_{sr}$ are set as $\boldsymbol{x}_a = [x, y]^\top$ and $\boldsymbol{x}_{sr} = \theta$, respectively. Basically, the reference point is set as $\boldsymbol{x}_r = [1000, -950]^\top$ and the prey motion $\boldsymbol{x}_p$ is given by the straight line shown in Fig. 4.4(a). From $\boldsymbol{x}_r$, $\boldsymbol{x}_p$ and $\boldsymbol{\lambda}$, $\dot{\boldsymbol{x}}_a = [\dot{x}, \dot{y}]^\top$ and $\ddot{\boldsymbol{x}}_a = [\ddot{x}, \ddot{y}]^\top$ are directly computed by (4.15)–(4.20) for each k-th LGL node, so the following results can be obtained.



(a) Optimal trajectory

(b) Optimal PCP nodes

(c) Cost of $\boldsymbol{pBest}_s$

(d) Cost of $\boldsymbol{gBest}$

Figure 4.4: VMCPSO results of the minimum-time problem

$$v(k) = \sqrt{\dot{x}^2(k) + \dot{y}^2(k)} \tag{4.25}$$

$$\theta(k) = \text{atan}\left(\frac{\dot{y}(k)}{\dot{x}(k)}\right) \tag{4.26}$$

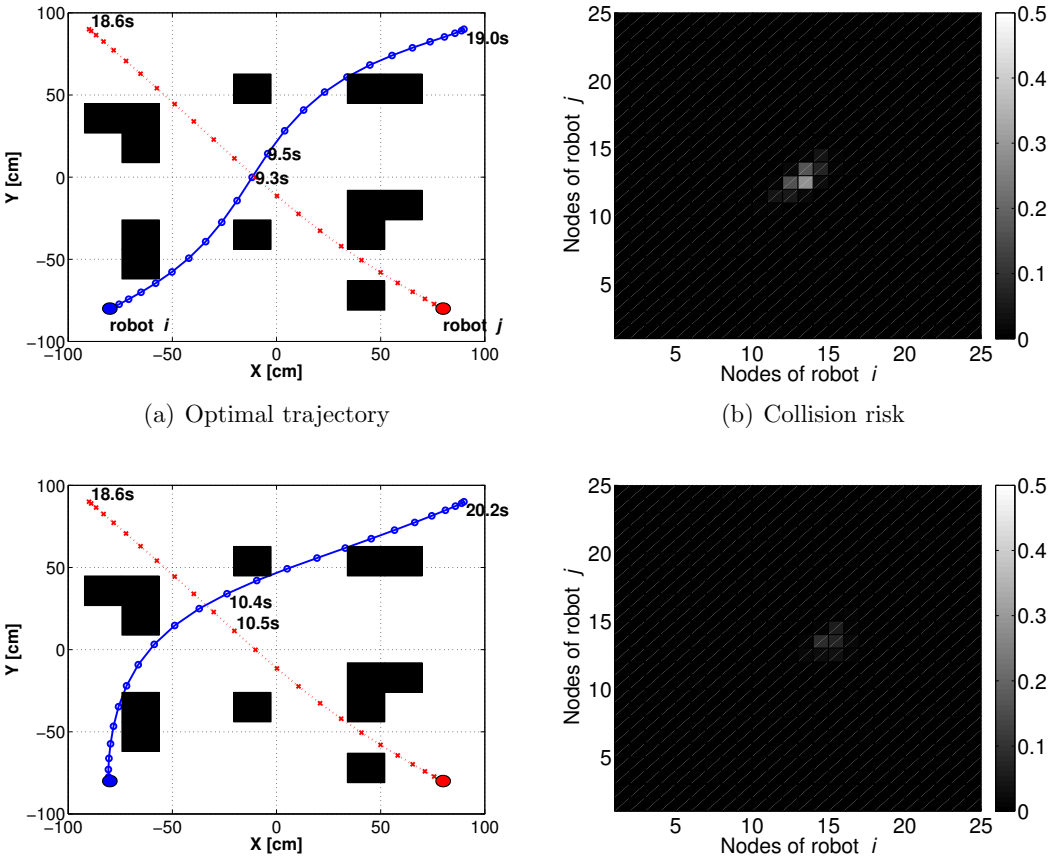
$$w(k) = \frac{\dot{x}(k)\ddot{y}(k) - \dot{y}(k)\ddot{x}(k)}{\dot{x}^2(k) + \dot{y}^2(k)}, (\dot{x}^2(k) + \dot{y}^2(k) \neq 0). \tag{4.27}$$

Thus, we can evaluate whether the constraints such as $|v| \leq v_{\max}$ and $|w| \leq w_{\max}$ are violated for each k-th LGL node by only using $\boldsymbol{x}_a$, $\dot{\boldsymbol{x}}_a$ and $\ddot{\boldsymbol{x}}_a$. To solve (4.24), we change

the formula (4.24) into (4.21) as follows:

$$\begin{aligned}\text{minimize } \; J = & \left(\frac{t_f - t_0}{2}\right) \sum_{k=0}^{N} \omega_k \\ & + C_1 \sum_{k=0}^{N} \max_k \left(0, |v(k)| - v_{\max}\right) \\ & + C_2 \sum_{k=0}^{N} \max_k \left(0, |w(k)| - w_{\max}\right) \\ & + C_3 \sum_{i=1}^{N_{\text{obs}}} \sum_{k=0}^{N} \max_k \left(0, 1/d_i(k) - 1/R_{\text{obs}}\right)\end{aligned} \tag{4.28}$$

where $d_i(k)$ denotes the distance between the position of i-th square obstacle and $\boldsymbol{x}_a(k)$. Then, we apply VMCPSO to solve (4.28) for the parameters given in Table 4.1. As shown in Fig. 4.4(a) the minimum-time optimal trajectory is well generated, and the resulting optimal PCP nodes are shown in Fig. 4.4(b). Figures 4.4(c) and 4.4(d) show a good convergence rate to an optimal solution. Figure 4.4(c) shows $J_{\boldsymbol{pBest}_s}$, the cost defined in (4.28) computed for each particle, i.e., the change in costs for the candidate paths as the number of VMCPSO iterations increases. It can be seen that the values of $J_{\boldsymbol{pBest}_s}$ do not get trapped in the local minima until they converge to the minimum values about 10 to 80 iterations, respectively. In addition, the change in the minimum cost among the candidate paths described in Fig. 4.4(d) converges fast within 50 iterations. So, we can conclude it effectively prevents from falling into local minima by maintaining randomness.



(a) Optimal trajectory (b) Collision risk

(c) (d)

Figure 4.5: (a,b) The case in which path information of robot j is not considered in the planning of robot i, (c,d) The case in which path information of robot j is considered in the planning of robot i

We perform additional simulations to address how the path information of the robots is taken into account. When the robot i uses the path information $\mathcal{P}_j$ collected by its neighbors $j \in \mathcal{N}_i$, the aggressor motions $\boldsymbol{x}_{a,j}(k)$ and the estimated time of arrival $t_{\text{est},j}(k), k = 0, \ldots, N$, are reconstructed from the path information $\mathcal{P}_j$. To prevent collision between the robot i and the robot j, we can compute the cost of collision risk between the k-th path point of the robot i and the k'-th path point of the robot j. Figures 4.5(b) and 4.5(d) show this relation between the k-th path point of the robot i and the k'-th path point of the

robot j. Therein, the horizontal axis corresponds to the PCP node k and the vertical axis corresponds to the PCP node k' in a two-robot situation. Then, the total cost of collision risk for the robot j can be defined by the following:

$$c_j^{\text{risk}} = \sum_{k=0}^{N} \sum_{k'=0}^{N} \exp(-(||\boldsymbol{x}_a(k) - \boldsymbol{x}_{a,j}(k')||_2 + ||t_{\text{est}}(k) - t_{\text{est},j}(k')||_2)). \tag{4.29}$$

Then, for the following minimum-time problem, the optimal trajectory of robot i can be obtained by considering the path information of the robot j.

$$\begin{aligned} \text{minimize} \;\; J = & \left(\frac{t_f - t_0}{2}\right) \sum_{k=0}^{N} \omega_k \\ & + C_0 \sum_{j \in \mathcal{N}_i} c_j^{\text{risk}}(\mathcal{P}_i, \mathcal{P}_j) \\ & + C_1 \sum_{k=0}^{N} \max_k (0, |v(k)| - v_{\max}) \\ & + C_2 \sum_{k=0}^{N} \max_k (0, |w(k)| - w_{\max}) \\ & + C_3 \sum_{i=1}^{N_{\text{obs}}} \sum_{k=0}^{N} \max_k (0, 1/d_i(k) - 1/R_{\text{obs}}) \end{aligned} \tag{4.30}$$

When the robot i does not consider the path information of the robot j ($C_0 = 0$), the optimal path is generated as shown in Fig. 4.5(a), and Figure 4.5(b) shows collision risks for the path points of the robot i and the robot j. When the robot i uses the path information of the robot j, the resulting optimal path shown in Fig. 4.5(c) takes longer time and also has low collision risks compared with the case not using the path information of the robot j as described in Fig. 4.5(d).

Now, our main problem which is the trajectory optimization with the terminal time and angle constraints is going to be discussed. Here, we consider the same environment, so the difference of cost functions between the minimum-time problem and the terminal time and angle constrained problem is just the first term of (4.30). If we denote $\phi(\boldsymbol{g})$ as the penalty

Table 4.2: Terminal time and angle errors and costs with respect to t_d and γ_d

	$\lvert t_{err}\rvert$			$\sum_{k=N-5}^{N-1} \lvert\gamma_{err}\rvert$			J		
$t_d =$	20	25	30	20	25	30	20	25	30
$\gamma_d = 0$	3.6780e-05	0.0214	4.8220e-04	3.2250	1.8958	1.3912	3.2251	3.8580	4.6457
45	8.2665e-04	2.1566e-05	1.4634e-04	7.1095	5.5934	5.4237	9.5107	7.5441	10.5731
90	1.2641e-04	8.1267e-04	0.0015	1.7506	1.6789	3.4234	1.7508	2.1583	3.4264

function including all the inequality constraints, our cost function can be defined by the following:

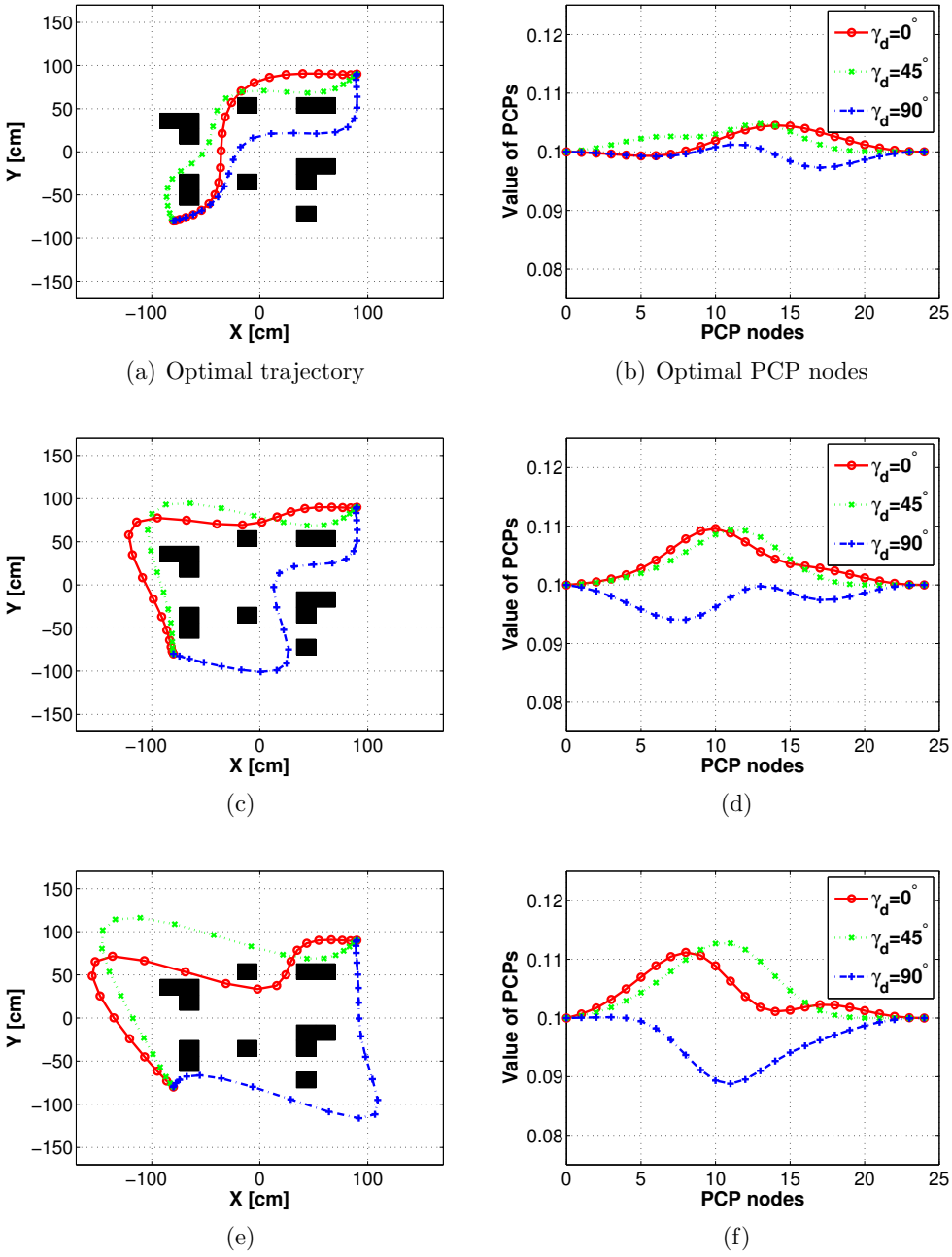
$$\text{minimize} \;\; J = I_T \left|t_{err}\right| + I_A \sum_{k=N-5}^{N-1} \left|\gamma_{err}\right| + \phi(\boldsymbol{g}) \tag{4.31}$$

where $t_{err} = t_d - t_{\text{est}}(N)$ and $\gamma_{err} = \gamma_d - \text{atan}\left(\frac{y(N)-y(k)}{x(N)-x(k)}\right)$. The weighting parameters I_T and I_A are set to 1 and 10, respectively. The optimization is performed for $t_d = (20, 25, 30)[\,\text{sec}]$ and $\gamma_d = (0, 45, 90)[\,\text{deg}]$ by using VMCPSO with Table 4.1. Figure 4.6 and Table 4.2 show that satisfactory results of the optimal trajectory under the cost function (4.31) and the PCP nodes for all the cases of t_d and γ_d. All the resulting trajectories do not collide with obstacles. The VMCPSO algorithm is implemented in C/C++, and all optimizations are performed on the same computer (Core i7-3290M CPU @ 2.90 GHz with 8 GB random-access memory) to observe the computation time for each optimal trajectory. The resulting average computation times for the three different numbers of particles are shown in Table 4.3, which confirms that the computation is fast enough for real-time applications. The

Table 4.3: Computation time with respect to S

	S		
	100	200	300
Computation time [sec]	0.076	0.107	0.167

VMCPSO algorithm is easy to implement, and if we use the parallel algorithm structure [50, 31], the computation time can be reduced more. As a result, our approach effectively solve the trajectory optimization problem with the terminal time and angle constraints in real time.



(a) Optimal trajectory (b) Optimal PCP nodes

(c) (d)

(e) (f)

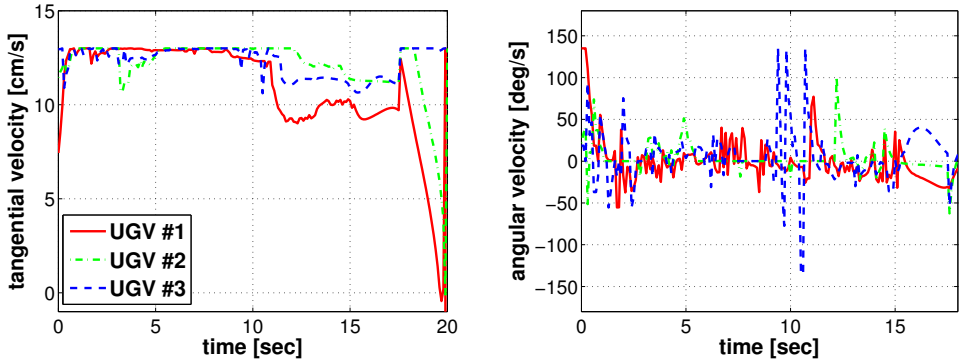
Figure 4.6: VMCPSO results of the terminal angle and time constrained problem (a,b) $t_d = 20\,\text{sec}$, (c,d) $t_d = 25\,\text{sec}$, (e,f) $t_d = 30\,\text{sec}$

4.3.2 Moving target

In Sect. 4.3.1, we obtained the satisfactory results to generate the optimal trajectory for the stationary target. However, when the target is moving, each robot needs to make a decision quickly. To make this possible, we utilized the concept of GPP and LPP as stated in Sect. 4.2. When GPP and LPP are performed, the parameter values for VMCPSO are set as given in Table 4.1 except N, M, S, and $iter_{\max}$ related to the computation time. Here, we set $N = 9$, $M = 2$, $S = 50$, and $iter_{\max} = 100$, especially 50 particles provide good enough performance. If these four parameters are properly selected according to environments, we can reduce the computation time to generate the optimal path for one robot, and the results are shown in Table 4.4. Again, fast enough for robot path planning. Three ground robots and one target are initially located at $(-100, -70)$, $(-110, -90)$, $(-80, -80)$, and $(100, 100)$, respectively. The target moves to the west with constant velocity at 4.3 cm/s. In our terminal angle and time constrained trajectory optimal problem, the desired terminal angle and time for each robot are fixed as $\gamma_d = (0, 45, 90)[\,\mathrm{deg}]$ and $t_d = (20, 20, 20)[\,\mathrm{sec}]$, and the terminal time t_d needs to be changed as time goes on such as $t_d \leftarrow t_d - \Delta t$. The cost function for GPP is considered as (4.31), and (4.30) is used for LPP.

Table 4.4: Computation time of decentralized VMCPSO for the moving target

	GPP	LPP	Total
Computation time [sec]	0.025	0.025	0.05



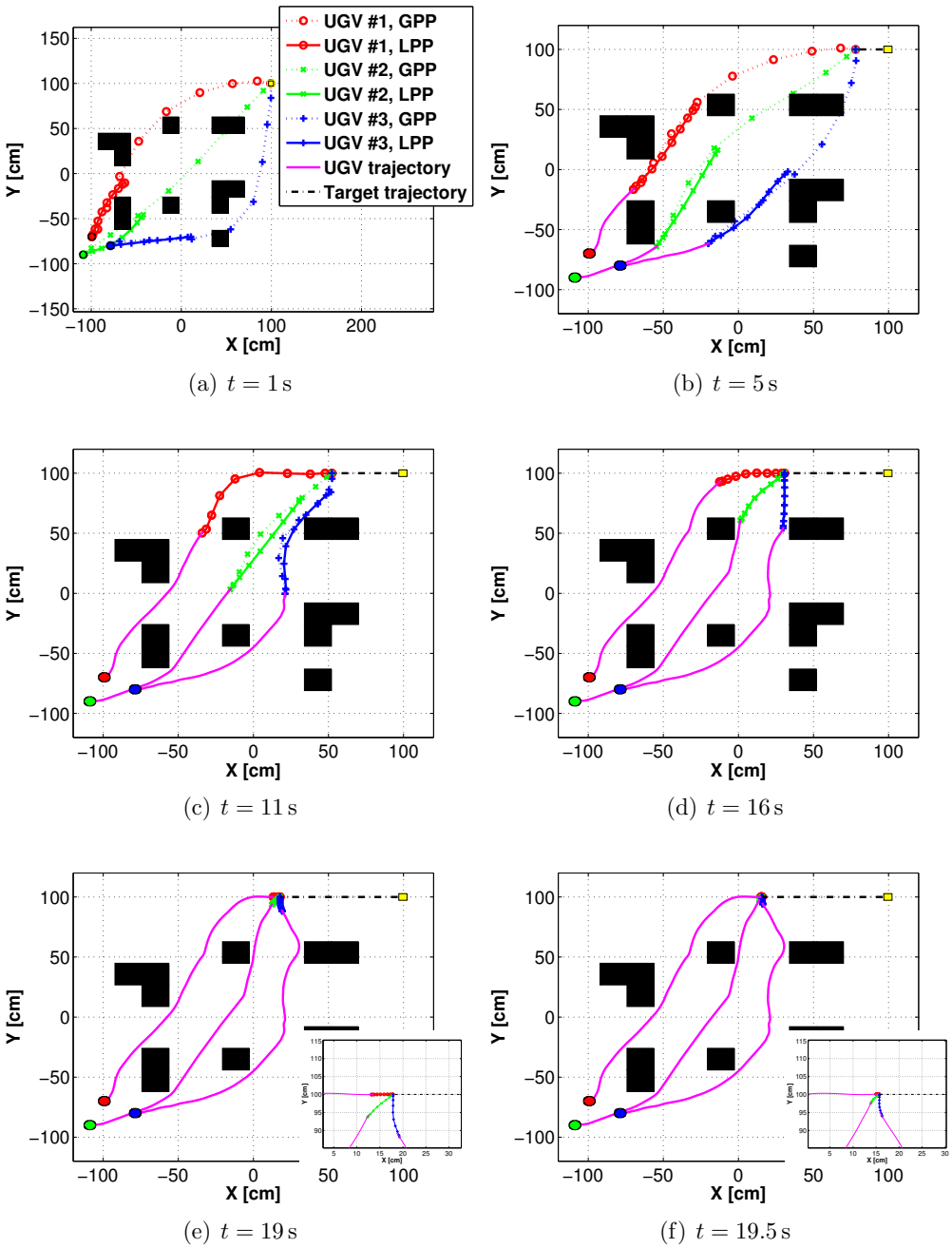
(a) v (b) w

Figure 4.7: Tangential and angular velocities

Figure 4.8 shows the trajectories of the robots and target over time t, and also the planned long-term and short-term optimal paths. As shown in Figs. 4.8(a) and 4.8(b), the robots planned the long-term paths satisfying the terminal constraints by using GPP, and LPP found the short-term path to follow the long-term path while preventing collision with obstacles. Although the global paths overlap with the obstacles, LPP can handle this problem with densely distributed nodes. The resulting trajectories of the robots support this as shown in Fig. 4.8(c). After $t = 11\,\mathrm{s}$, when the target is located inside the reachable set of the first robot, both GPP and LPP plan the optimal trajectories using the same criteria on the terminal time and angle constraints. So, the global and local paths of the first robot are the same as shown in Fig. 4.8(c).

After $t = 16\,\mathrm{s}$, all the robots concentrate on approaching the target at the same time with the predefined heading. Finally, the robots achieved the goal of reaching the target with the given terminal time and angle constraints as described in Figs. 4.8(e) and 4.8(f). Figure 4.7 shows the tangential and angular velocities of the robots. About 11 seconds before, all the robots move fast to follow the short-term paths obtained by solving the minimum-time problem. After that, the robots move at the proper velocity obtained from the trajectory optimization to satisfy the terminal constraints. In Fig. 4.7(a), we can observe that the velocities of the first and second robots are significantly decreased at around 18

seconds, because they face the target. As a result, we confirmed our algorithm successfully solves the trajectory optimization problem constrained on the terminal time and angle for the moving target.



(a) $t = 1\,\mathrm{s}$ (b) $t = 5\,\mathrm{s}$

(c) $t = 11\,\mathrm{s}$ (d) $t = 16\,\mathrm{s}$

(e) $t = 19\,\mathrm{s}$ (f) $t = 19.5\,\mathrm{s}$

Figure 4.8: Snapshots of global and local path planning

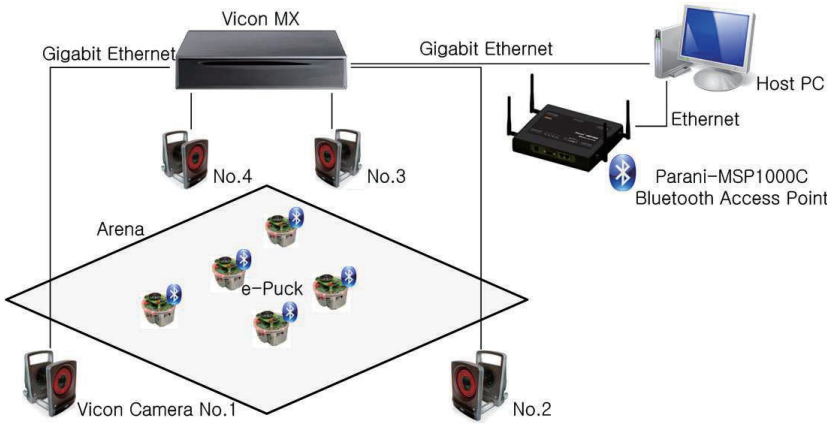
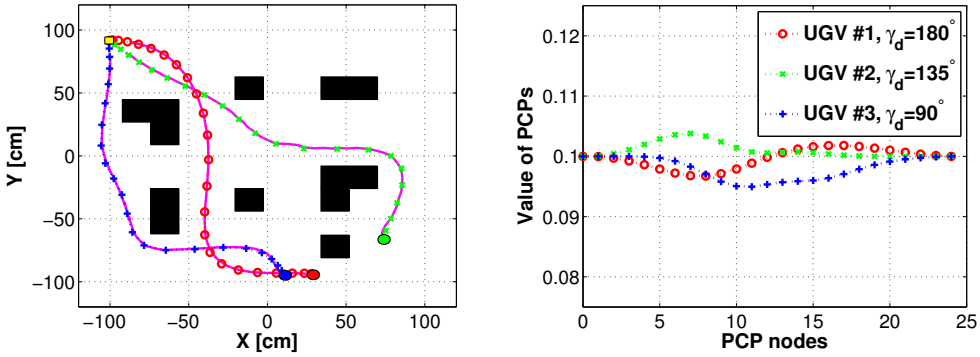


Figure 4.9: Experimental environment

4.4 Experimental Results

In this section, experimental results of our algorithm are presented for the multi-robot rendezvous problem described previously. Our experimental environment using e-puck ground robots [41] is set up as shown in Fig. 4.9. Vicon MX camera system measures the position and orientation of the robots in real time. Parani-MSP1000 is used for communication between the robots and the host PC. As shown in Fig. 4.11(a), there are three robots and one stationary target. They are initially located at $(29.2, -94.4)$, $(74, -66.4)$, $(11.3, -94.9)$, and $(-100.5, 91.7)$, respectively. We set the terminal constraints as $t_d = (19.5, 19.5, 19.5)[\,\text{sec}]$ and $\gamma_d = (180, 135, 90)$ $[\,\text{deg}]$ for each robot. Dec-VMCPSO was performed in the same manner as the simulation study described in Sect. 4.3.1, and the cost function (4.31) and Table 4.1 are used for this experiment.

Figure 4.10(a) shows the optimal trajectories satisfying the terminal constraints in the complex environment, and PCPs of each robot were optimized as displayed in Fig. 4.10(b). Each robot planned its optimal path by considering the path information of other robots to prevent collision with other robots as well as the stationary obstacles. Then, they perfectly moved along the planned optimal paths by using the tracking controller. Figure 4.11 shows

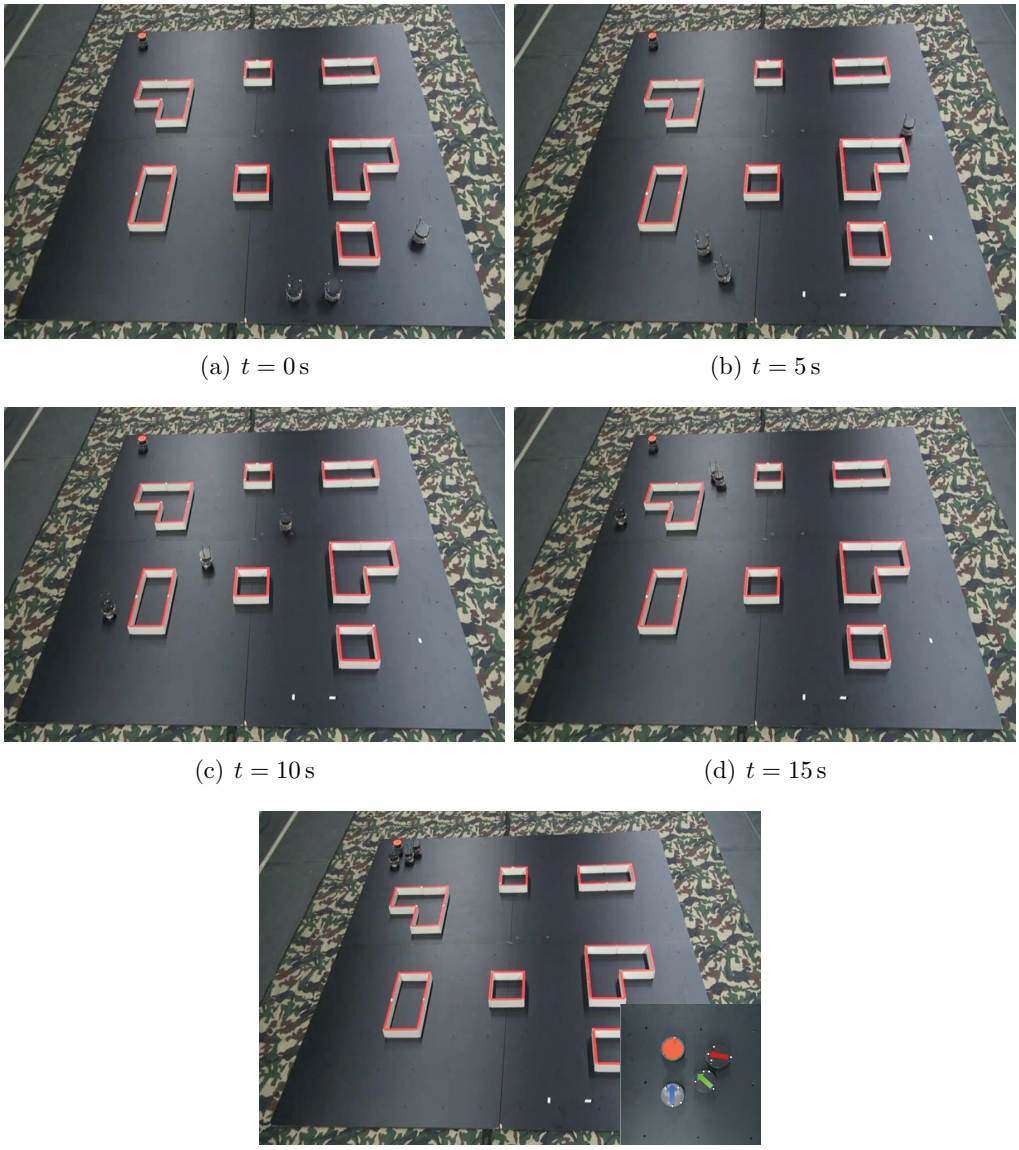


(a) Optimal trajectory

(b) Optimal PCP nodes

Figure 4.10: Experimental results of decentralized VMCPSO for the stationary target

snapshots of the rendezvous experiment. After 19 seconds, all the robots approached the target by maintaining the desired terminal angles as shown in Fig. 4.11(e). To summarize, the proposed Dec-VMCPSO was validated by the experimental results, and it efficiently solved the nonlinear constrained trajectory optimization problem in multi-robot systems.



(a) $t = 0\,\mathrm{s}$ (b) $t = 5\,\mathrm{s}$

(c) $t = 10\,\mathrm{s}$ (d) $t = 15\,\mathrm{s}$

(e) $t = 19\,\mathrm{s}$

Figure 4.11: Snapshots of the rendezvous experiment in which three robots try to arrive at the stationary target point simultaneously with the specific heading

4.5 Analysis and Discussion

In this chapter, we proposed VMCPSO and Dec-VMCPSO to solve a nonlinear constrained trajectory optimization problem, especially the rendezvous problem considering terminal time and angle constraints in an unban-like environment. From the simulation and experimental results, we can realize that the optimal trajectories were properly generated while satisfying the given performance objectives. We can also find that the proposed algorithms have a small amount of computation time and a good convergence rate to an optimal solution while not getting trapped in the local minima by inheriting merits of both VMC and PSO. These advantages suggest a good potential for VMCPSO as a reactive path planner. For multi-robot path planning, Dec-VMCPSO could handle both static and dynamic situations because it is irrespective of the planning order. However, in this thesis we only performed the simulation study for the whole optimization and communication process. So, onboard implementation of the whole optimization and communication process remains an ongoing work.

5

Behavior Coordination

In this chapter, we create behaviors of the ground robots, and a distributed multi-agent reinforcement learning algorithm is suggested to solve a behavior coordination problem in engagement scenarios. We first consider five types of behaviors, and each behavior is implemented by the proposed algorithms for mission/path planning. The threat map and the visibility map are also considered to build the behaviors. CBBA and VMCPSO have advantages to produce the behaviors because they are designed in a distributed manner and applicable to real-time applications and dynamic situations as mentioned in Chapters 3 and 4. Although we can design the proper behaviors in the engagement scenario, there is a remaining problem. We should solve the problem of how to choose the most appropriate behavior for given situations. In order to solve this problem we can consider dynamic programming (DP) and reinforcement learning (RL). They are widely used to solve decision-making problems in fields of artificial intelligence. The main concept of DP and RL is that an agent interacts with environment by means of states and actions, and receives rewards according to a reward function. DP as a model-based RL requires a model of Markov Decision Process (MDP) such as the transition dynamics and the reward function.

On the contrary, model-free RL techniques are useful when a model is difficult or costly to construct. In our considering scenarios, there are many probabilistic situations and robots, so it is hard to define environmental models. Consequentially, we use model-free RL algorithms to obtain the optimal policy from repeated simulations.

5.1 Design of Behaviors

By using an integrated mission and path planning framework, we can implement five types of behaviors as follows.

- *Engagement*

  The target location $\boldsymbol{x}_{\text{target}}$ is determined as the first allocated location of path $\boldsymbol{p}_i$ given by CBBA. The robot i uses Dec-VMCPSO to generate the minimum-time optimal trajectory in order to move to the given target location $\boldsymbol{x}_{\text{target}}$. We assume that the engagement is not possible during the fast movement, so the original speed limit constraint is changed into one-tenth of the maximum speed $v_{\max}$. Then, we can obtain the optimal trajectory between $\boldsymbol{x}_i$ and $\boldsymbol{x}_{\text{target}}$ by solving (4.30). In addition, if the robot i is ready to attack ($\mathbf{W}_i = 0$ and $\mathbf{A}_i = 1$), it combats the threat with the probability of attack depending on its type k_i.

- *Concealment*

  The target location $\boldsymbol{x}_{\text{target}}$ is obtained by using the visibility map about the threats at time t, i.e., $\rho_e^h(\boldsymbol{x}_v, t)$. The robot i first finds a location having the minimum distance between its location and an invisible region in $\rho_e^h(\boldsymbol{x}_v, t)$. Then, the robot follows the optimal path obtained by solving (4.30) to reach that position as soon as possible with the maximum speed.

- *Move to the allocated target*

  Similar to *Engagement* the target location $\boldsymbol{x}_{\text{target}}$ is obtained from CBBA, but the robot cannot deal with the threat and shows fast movements.

- *Move to the goal*

  The target location is determined as the goal position $\boldsymbol{x}_{\text{goal}}$. Then, the trajectory between $\boldsymbol{x}_i$ and $\boldsymbol{x}_{\text{goal}}$ is generated by solving (4.30).

- *Standby*

  The robot just stays at its current location and waits for next order.

5.2 Learning Framework

In this section, we first explain differences between Markov Decision Process (MDP) and semi-Markov Decision Process (SMDP) in reinforcement learning (RL). Then, we introduce a RL algorithm in a SMDP framework with linear function approximation.

5.2.1 MDP vs. SMDP

We first consider a finite MDP which consists of state and action sets, $\mathcal{S}, \mathcal{A}_s$, for $s \in \mathcal{S}$, and its dynamics given by one-step state-transition probabilities, $p^a_{ss'} = \Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$, and one-step expected rewards, $r^a_s = E\{r_{t+1} | s_t = s, a_t = a\}$, for all $s \in \mathcal{S}, a \in \mathcal{A}_s$, and $s' \in \mathcal{S}^+$. $\mathcal{S}^+$ is the union of regular states and the terminal state in an episodic task. The agent learns a policy, $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, that maximizes the expected discounted future reward from each state s:

$$
\begin{aligned}
V^\pi(s) &= E\left\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s_t = s, \pi\right\} \\
&= E\left\{r_{t+1} + \gamma V^\pi(s_{t+1}) | s_t = s, \pi\right\} \\
&= \sum_{a \in \mathcal{A}(s)} \pi(s, a) \left[r^a_s + \gamma \sum_{s'} p^a_{ss'} V^\pi(s') \right]
\end{aligned} \tag{5.1}
$$

where $\gamma \in [0, 1)$ is the discount factor, and we call the function V^π the state-value function for π. Then, the optimal state-value function can be obtained under an optimal policy:

$$V^*(s) = \max_\pi V^\pi(s). \tag{5.2}$$

We can also define an optimal state-action value function $Q^*(s, a)$ in terms of $V^*(s)$ as:

$$Q^*(s, a) = E\left\{r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\right\} \tag{5.3}$$

For more background on RL see [54].

In [56], a set of *options*, macro-actions across longer lengths of time, is included in the formal framework of MDP. Options consist of a policy $\mu : \mathcal{S} \times \mathcal{A}(s) \to [0, 1]$, a termination condition $\xi : \mathcal{S}^+ \to [0, 1]$, and initiation set $\mathcal{I} \subseteq \mathcal{S}$. So, SMDP can be defined as an extension of the MDP model, and it consists of a set of states $\mathcal{S}$, a set of options $\mathcal{O}_s, s \in \mathcal{S}$, and expected rewards r_s^o. In SMDP, rewards and state-transition probabilities should be redefined depending on the option o and its termination time. The reward r_s^o initiated in state s at time t is defined as follows:

$$r_s^o = E\left\{r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{k-1} r_{t+k} \middle| \mathcal{E}(o, s, t)\right\} \tag{5.4}$$

where $\mathcal{E}(o, s, t)$ is the event of an option o being initiated in state s at time t, and $t + k$ is the termination time of o. The state-transition probabilities $p_{ss'}^o$ is defined as follows:

$$p_{ss'}^o = \sum_{k=0}^{\infty} p(s', k)\gamma^k \tag{5.5}$$

where $p(s', k)$ is the probability that the option terminates in s' after k steps. From (5.4)

and (5.5), we can define the state-value function for μ as follows:

$$\begin{aligned} V^{\mu}(s) =& E\{r_{t+1} + \cdots + \gamma^{k-1} r_{t+k} + \gamma^{k} V^{\mu}(s_{t+k}) | \mathcal{E}(\mu, s, t)\} \\ =& \sum_{o \in O_s} \mu(s, o) \left[r_s^o + \sum_{s'} p_{ss'}^o V^{\mu}(s') \right]. \end{aligned} \tag{5.6}$$

Then, the optimal state-value function is given by the following:

$$V_{\mathcal{O}}^{*}(s) = \max_{\mu} V^{\mu}(s). \tag{5.7}$$

Similarly, the option-value function and the optimal option-value function can be defined as follows:

$$Q^{\mu}(s, o) = E\{r_{t+1} + \cdots + \gamma^{k-1} r_{t+k} + \gamma^{k} V^{\mu}(s_{t+k}) | \mathcal{E}(o, s, t)\} \tag{5.8}$$

$$\begin{aligned} Q_{\mathcal{O}}^{*}(s, o) =& \max_{\mu} Q^{\mu}(s, o) \\ =& E\{r_{t+1} + \cdots + \gamma^{k-1} r_{t+k} + \gamma^{k} V_{\mathcal{O}}^{*}(s_{t+k}) | \mathcal{E}(o, s, t)\}. \end{aligned} \tag{5.9}$$

5.2.2 Linear approximation of value functions

In large or continuous-space problems, it is intractable to obtain the optimal action-value function exactly due to "*curse of dimensionality*". If we consider n-dimensional state space, the number of states grows exponentially in n when assuming some fixed number of discretization levels per coordinate. So, such as generalization often called function approximation is promisingly demanded. The function approximation technique combined with the SMDP framework is very advantageous to solve high-dimensionality problems because of the benefits of the hierarchical structure [18].

The option-value function $Q(s, o)$ can be approximated by the following:

$$\hat{Q}(s, o) = \sum_{i=1}^{n} \phi_i(s, o)\varphi_i = \boldsymbol{\phi}(s, o)^\top \boldsymbol{\varphi} \tag{5.10}$$

where $\boldsymbol{\phi}(s, o) = [\phi_1(s, o), \ldots, \phi_n(s, o)]^\top$ is the vector of basis functions and $\boldsymbol{\varphi}$ is an n-dimensional parameter vector. In the discretized option space, the state-option basis function vector $\boldsymbol{\phi}(\boldsymbol{s}, o_j)$ can be defined as follows:

$$\boldsymbol{\phi}(\boldsymbol{s}, o_j) = [\underbrace{0, \ldots, 0}_{o_1}, \ldots, 0, \underbrace{\bar{\phi}_1(\boldsymbol{s}), \ldots, \bar{\phi}_{N_\phi}(\boldsymbol{s})}_{o_j}, 0, \ldots, \underbrace{0, \ldots, 0}_{o_{N_o}}]^\top \in \mathbb{R}^{N_{\boldsymbol{\phi}} N_o}. \tag{5.11}$$

Here, the feature state vector $\boldsymbol{s}$ should be properly selected by a designer in order to reflect the nature of the problem. Given the feature state vector $\boldsymbol{s} = [s_1, \ldots, s_N]^\top$, the state basis functions are obtained by employing polynomial basis functions:

$$\phi_i(\boldsymbol{s}) = \prod_{j=1}^{N} s_j^{n_{i,j}} \tag{5.12}$$

where each $n_{i,j}$ is integer to determine the order of a polynomial. For example, a second order polynomial basis defined over two state variables x and y would have feature vector: $\boldsymbol{\phi} = [1, x, y, xy, x^2y, xy^2, x^2y^2]^\top$. If we obtain this kind of a basis function vector $\boldsymbol{\phi}(\boldsymbol{s}) = [\phi_1(\boldsymbol{s}), \ldots, \phi_{N_{\boldsymbol{\phi}}}(\boldsymbol{s})]^\top$, it shoud be normalized as follows:

$$\bar{\phi}_i(\boldsymbol{s}) = \frac{\phi_i(\boldsymbol{s})}{\sum_{i=1}^{N_{\boldsymbol{\phi}}} \phi_i(\boldsymbol{s})}. \tag{5.13}$$

Finally, we have the state-option basis function vector (5.11). For more details see [54, 10].

5.2.3 Learning value function approximations

In Q-learning using a lookup table representation, the Q-learning algorithm can find the optimal Q-function iteratively by the following simple update rule:

$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha_k[r_{k+1} + \gamma \max_{\bar{a}} Q_k(s',\bar{a}) - Q_k(s,a)] \tag{5.14}$$

where (s', r_{k+1}, α_k) are the next state, reward, and learning rate, respectively. When we learn a linear parameter vector $\boldsymbol{\varphi}$, the learning algorithm aims to minimize the mean-squared Bellman error:

$$\text{MSBE}(\boldsymbol{\varphi}) = \frac{1}{2}||V_{\boldsymbol{\varphi}} - \mathcal{T}V_{\boldsymbol{\varphi}}||_D^2 \tag{5.15}$$

where $\mathcal{T}$ is the Bellman operator and the matrix D has the d_s on its diagonal, $d_s = \lim_{t\to\infty} p(s_t = s)$, and the norm $||v||_D^2 = v^\top Dv$. Depending on the policy π, $\boldsymbol{\varphi}$ is optimized by using a stochastic gradient algorithm:

$$\begin{aligned} \boldsymbol{\varphi}_{k+1} &= \boldsymbol{\varphi}_k - \alpha_k \nabla_{\boldsymbol{\varphi}} \text{MSBE}(\boldsymbol{\varphi}) \\ &= \boldsymbol{\varphi}_k + \alpha_k \delta^\pi (\boldsymbol{\phi}_k - \gamma \boldsymbol{\phi}'_k) \end{aligned} \tag{5.16}$$

where the temporal difference δ^π is given by

$$\delta^\pi = r_{k+1} + \gamma \boldsymbol{\varphi}_k^\top \boldsymbol{\phi}'_k - \boldsymbol{\varphi}_k^\top \boldsymbol{\phi}_k. \tag{5.17}$$

However, there is no convergence guarantees when we use (5.16). It also showed significant instabilities in counterexamples [54].

To overcome this problem, the mean-squared projected Bellman error is considered as

follows [55]:

$$\mathrm{MSPBE}(\boldsymbol{\varphi}) = \frac{1}{2}||V_{\boldsymbol{\varphi}} - \Pi \mathcal{T} V_{\boldsymbol{\varphi}}||_D^2 \tag{5.18}$$

where Π is the projection operator. Then, the gradient of the MSPBE is derived as follows:

$$\nabla_{\boldsymbol{\varphi}}\mathrm{MSPBE}(\boldsymbol{\varphi}) = -E\left[(\boldsymbol{\phi} - \gamma\boldsymbol{\phi}')\boldsymbol{\phi}^\top\right] E\left[\boldsymbol{\phi}\boldsymbol{\phi}^\top\right]^{-1} E\left[\delta^\pi \boldsymbol{\phi}\right], \tag{5.19}$$

and an update algorithm is given by

$$\begin{aligned} \boldsymbol{\varphi}_{k+1} &= \boldsymbol{\varphi}_k - \alpha_k \nabla_{\boldsymbol{\varphi}}\mathrm{MSPBE}(\boldsymbol{\varphi}) \\ &= \boldsymbol{\varphi}_k + \alpha_k E\left[(\boldsymbol{\phi} - \gamma\boldsymbol{\phi}')\boldsymbol{\phi}^\top\right] E\left[\boldsymbol{\phi}\boldsymbol{\phi}^\top\right]^{-1} E\left[\delta^\pi \boldsymbol{\phi}\right]. \end{aligned} \tag{5.20}$$

Here, we employ a linear predictor vector $\boldsymbol{\chi} \approx E\left[\boldsymbol{\phi}\boldsymbol{\phi}^\top\right]^{-1} E\left[\delta^\pi \boldsymbol{\phi}\right]$, and $\boldsymbol{\chi}$ can be updated by the following:

$$\boldsymbol{\chi}_{k+1} = \boldsymbol{\chi}_k + \beta_k \left(\delta^\pi - \boldsymbol{\phi}_k^\top \boldsymbol{\chi}_k\right) \boldsymbol{\phi}_k. \tag{5.21}$$

Then, we have the gradient

$$\begin{aligned} \nabla_{\boldsymbol{\varphi}}\mathrm{MSPBE}(\boldsymbol{\varphi}) &= -E\left[(\boldsymbol{\phi} - \gamma\boldsymbol{\phi}')\boldsymbol{\phi}^\top\right] E\left[\boldsymbol{\phi}\boldsymbol{\phi}^\top\right]^{-1} E\left[\delta^\pi \boldsymbol{\phi}\right] \\ &\approx -E\left[(\boldsymbol{\phi} - \gamma\boldsymbol{\phi}')\boldsymbol{\phi}^\top\right] \boldsymbol{\chi} \qquad (5.22) \\ &\text{or} \\ &\approx -\left(E\left[\delta^\pi \boldsymbol{\phi}\right] - \gamma E[\,\boldsymbol{\phi}'\boldsymbol{\phi}^\top\,]\boldsymbol{\chi}\right). \qquad (5.23) \end{aligned}$$

Therefore, we have two algorithms. The first algorithm called generalized temporal difference 2 (GTD2) is obtained by (5.22):

$$\boldsymbol{\varphi}_{k+1} = \boldsymbol{\varphi}_k + \alpha_k \left(\boldsymbol{\phi}_k - \gamma\boldsymbol{\phi}_k'\right) \left(\boldsymbol{\phi}_k^\top \boldsymbol{\chi}_k\right). \tag{5.24}$$

Algorithm 9 GQ-learning for agent i

1: **procedure** GQ-LEARNING$(\boldsymbol{\varphi}_i, \boldsymbol{\chi}_i, \bar{R}_i, \boldsymbol{s}_i, o_i, \boldsymbol{s}'_i, o'_i, K_i, \alpha_{n_{ep}}, \beta_{n_{ep}})$
2: $\quad \delta'_i \leftarrow \bar{R}_i + \gamma^{K_i} \boldsymbol{\varphi}_i^\top \boldsymbol{\phi}(\boldsymbol{s}'_i, o'_i) - \boldsymbol{\varphi}_i^\top \boldsymbol{\phi}(\boldsymbol{s}_i, o_i)$
3: $\quad \boldsymbol{\varphi}'_i \leftarrow \boldsymbol{\varphi}_i + \alpha_{n_{ep}} \left(\delta'_i \boldsymbol{\phi}(\boldsymbol{s}_i, o_i) - \gamma^{K_i} \boldsymbol{\phi}(\boldsymbol{s}'_i, o'_i)(\boldsymbol{\phi}(\boldsymbol{s}_i, o_i)^\top \boldsymbol{\chi}_i)\right)$
4: $\quad \boldsymbol{\chi}'_i \leftarrow \boldsymbol{\chi}_i + \beta_{n_{ep}} \left(\delta'_i - \boldsymbol{\phi}(\boldsymbol{s}_i, o_i)^\top \boldsymbol{\chi}_i\right) \boldsymbol{\phi}(\boldsymbol{s}_i, o_i)$
5: $\quad$ **return** $(\boldsymbol{\varphi}'_i, \boldsymbol{\chi}'_i)$
6: **end procedure**

The second algorithm called temporal difference with gradient corrector (TDC) is given by (5.23):

$$\boldsymbol{\varphi}_{k+1} = \boldsymbol{\varphi}_k + \alpha_k \left(\delta^\pi \boldsymbol{\phi}_k - \gamma \boldsymbol{\phi}'_k (\boldsymbol{\phi}_k^\top \boldsymbol{\chi}_k)\right). \tag{5.25}$$

According to our learning setup, we now propose the SMDP version of GQ-learning as described in Algorithm 9. Initially, $(\boldsymbol{\varphi}_i, \boldsymbol{\chi}_i)$ are set to $\mathbf{0}^\top$. Afterward, each agent i individually learns its own option-value function $\hat{Q}(\boldsymbol{s}_i, o_i)$ based on TDC as shown in lines 3–4 of Algorithm 9. Here, $(\boldsymbol{s}_i, o_i)$ are the state and option at the beginning of the SMDP time step, and $(\boldsymbol{s}'_i, o'_i)$ are the state and option at the end of the SMDP time step, where o'_i is some maximizing option of $\hat{Q}(\boldsymbol{s}'_i, o'_i)$. We use the average reward $\bar{R}_i$ at the terminal SMDP time step K_i as follows:

$$\bar{R}_i = \frac{1}{K_i} \sum_{k=0}^{K_i} R_k. \tag{5.26}$$

The learning rate $\alpha_{n_{ep}}$ for n_{ep}-th episode can be selected as follows:

$$\alpha_{n_{ep}} = \frac{1}{(n_{ep})^\kappa} \tag{5.27}$$

where $\kappa \in (\frac{1}{2}, 1]$. Here, we employ $\eta = \frac{\beta_{n_{ep}}}{\alpha_{n_{ep}}} > 0$, the ratio between the learning rates. For more details see [44].

5.3 Distributed Multi-Agent Reinforcement Learning

In this section, we introduce a distributed multi-agent reinforcement learning algorithm in the SMDP framework by employing a diffusion adaptation method.

5.3.1 Diffusion adaptation method for distributed optimization

A global cost function in the distributed optimization problem is defined by the following [11]:

$$J^{\text{glob}}(\boldsymbol{\varphi}) = \sum_{i=1}^{N_a} J_i(\boldsymbol{\varphi}) \tag{5.28}$$

where $J_i(\boldsymbol{\varphi}), i = 1, 2, \ldots, N_a$, are the local cost function for the agent i (or node) over a communication network. The main objective of the distributed optimization is to find an optimal vector $\boldsymbol{\varphi}^*$ that minimizes $J^{\text{glob}}(\boldsymbol{\varphi})$. We assume that $J_i(\boldsymbol{\varphi})$ is differentiable and convex, and $J^{\text{glob}}(\boldsymbol{\varphi})$ is strictly convex, so $\boldsymbol{\varphi}^*$ is unique. Then, we can express $J^{\text{glob}}(\boldsymbol{\varphi})$ as follows:

$$J^{\text{glob}}(\boldsymbol{\varphi}) = J_i^{\text{loc}}(\boldsymbol{\varphi}) + \sum_{l \neq i}^{N_a} J_l^{\text{loc}}(\boldsymbol{\varphi}) \tag{5.29}$$

where

$$J_i^{\text{loc}}(\boldsymbol{\varphi}) = \sum_{l \in \mathcal{N}_i} c_{l,i} J_l(\boldsymbol{\varphi}) \tag{5.30}$$

$$\sum_{i=1}^{N_a} c_{l,i} = 1, \quad c_{l,i} = 0 \text{ if } l \notin \mathcal{N}_i, \quad l = 1, 2, \ldots, N_a.$$

Here, $c_{l,i}$ is a nonnegative coefficient, and $\mathcal{N}_i$ is the neighborhood of node i. On the basis of [11], (5.29) is redefined by the following:

$$J_i^{\mathrm{glob}'}(\boldsymbol{\varphi}) = \sum_{l \in \mathcal{N}_i} c_{l,i} J_l(\boldsymbol{\varphi}) + \sum_{l \in \mathcal{N}_i \setminus \{i\}} b_{l,i} ||\boldsymbol{\varphi} - \boldsymbol{\varphi}_l^{\mathrm{loc}}||^2 \tag{5.31}$$

where $c_{l,i}$ and $b_{l,i}$ are nonnegative weight coefficients used by agent i that control the importance of information diffused by its neighbors. Each agent i can minimize (5.31) by using the steepest-descent method in a distributed manner. There are two types of diffusion strategies: Adapt-then-Combine (ATC) and Combine-then-Adapt (CTA).

The ATC diffusion strategy consists of the following 2-steps iterative algorithm.

Step 1 (Adapt) Each agent i updates an intermediate estimate $\boldsymbol{\varpi}_i'$ by sharing gradient information based on its own local estimate $\boldsymbol{\varphi}_i$ with its neighbors.

$$\boldsymbol{\varpi}_i' = \boldsymbol{\varphi}_i + \alpha_i \sum_{l \in \mathcal{N}_i} c_{l,i} \nabla_{\boldsymbol{\varphi}} J_l(\boldsymbol{\varphi}_i) \tag{5.32}$$

Here, α_i is a small step size.

Step 2 (Combine) Each agent i updates its own local estimate $\boldsymbol{\varphi}_i'$ by combining the intermediate estimates of its neighbors, $\boldsymbol{\varpi}_l', \forall l \in \mathcal{N}_i$.

$$\boldsymbol{\varphi}_i' = \sum_{l \in \mathcal{N}_i} a_{l,i} \boldsymbol{\varpi}_l' \tag{5.33}$$

Here, $a_{l,i}$ is a nonnegative weight coefficient similar to $b_{l,i}$.

The CTA diffusion strategy is defined by reversing the order of steps (5.32) and (5.33).

5.3.2 Cooperative GQ-learning

In [37], a distributed multi-agent reinforcement learning algorithm called cooperative GTD2 (C-GTD2) was proposed, and C-GTD2 showed trustworthy results for the Baird's coun-

terexamples [54], in which TD(0) with linear approximation can diverge. C-GTD2 is based on GTD2 and diffusion adaptation, and update rules can be obtained by considering (5.21), (5.24), (5.32), and (5.33). The complete C-GTD2 algorithm is defined as follows:

$$\boldsymbol{\sigma}_{i,k+1} = \boldsymbol{\varphi}_{i,k} + \alpha_{i,k} \sum_{l \in \mathcal{N}_i} c_{l,i} \left(\boldsymbol{\phi}_{l,k} - \gamma \boldsymbol{\phi}'_{l,k} \right) \left(\boldsymbol{\phi}_{l,k}^\top \boldsymbol{\chi}_{i,k} \right) \tag{5.34}$$

$$\boldsymbol{\varphi}_{i,k+1} = \sum_{l \in \mathcal{N}_i} a_{l,i} \boldsymbol{\sigma}_{l,k+1} \tag{5.35}$$

$$\boldsymbol{\varsigma}_{i,k+1} = \boldsymbol{\chi}_{i,k} + \beta_{i,k} \sum_{l \in \mathcal{N}_i} c_{l,i} \left(\delta_{l,k} - \boldsymbol{\phi}_{l,k}^\top \boldsymbol{\chi}_{i,k} \right) \boldsymbol{\phi}_{l,k} \tag{5.36}$$

$$\boldsymbol{\chi}_{i,k+1} = \sum_{l \in \mathcal{N}_i} a_{l,i} \boldsymbol{\varsigma}_{l,k+1} \tag{5.37}$$

where $\delta_{l,k} = r_{l,k} + \gamma \boldsymbol{\varphi}_{i,k}^\top \boldsymbol{\phi}'_{l,k} - \boldsymbol{\varphi}_{i,k}^\top \boldsymbol{\phi}_{l,k}$.

From the above, we propose the SMDP version of cooperative GQ-learning as shown in Algorithm 10. As described in lines 4 and 7 of Algorithm 10, each agent i iteratively updates two intermediate estimates $(\boldsymbol{\sigma}_i, \boldsymbol{\varsigma}_i)$ by reflecting its current local estimates $(\boldsymbol{\varphi}_i, \boldsymbol{\chi}_i)$ and all the information of its neighbors $l \in \mathcal{N}_i$, i.e., $(\bar{R}_l, \boldsymbol{s}_l, o_l, \boldsymbol{s}'_l, o'_l, K_l)$. Here, the state and option $(\boldsymbol{s}_l, o_l)$ are determined at the begining of the SMDP time step depending on the agent l, and the state $\boldsymbol{s}'_l$ as the terminal state is obtained when the agent i interrupts the agent l to request the information. Then, the option o'_l can be selected based on $\boldsymbol{\varphi}_i$ as follows:

$$o'_l = \arg\max_{\bar{o}} \boldsymbol{\phi}(\boldsymbol{s}'_l, \bar{o})^\top \boldsymbol{\varphi}_i. \tag{5.38}$$

For the terminal SMDP time step K_l, we have the average reward $\bar{R}_l$ of the agent l as follows:

$$\bar{R}_l = \frac{1}{K_l} \sum_{k=0}^{K_l} R_k. \tag{5.39}$$

Algorithm 10 Cooperative GQ-learning for agent i and its neighbors $l \in \mathcal{N}_i$

1: **procedure** C-GQ-LEARNING($\boldsymbol{\varphi}_i, \boldsymbol{\chi}_i, \boldsymbol{\sigma}'_l, \boldsymbol{\varsigma}'_l, \bar{R}_l, \boldsymbol{s}_l, o_l, \boldsymbol{s}'_l, o'_l, K_l, \alpha_{n_{ep}}, \beta_{n_{ep}}$)
2: $\quad \delta'_l \leftarrow \bar{R}_l + \gamma^{K_l} \boldsymbol{\varphi}_i^\top \boldsymbol{\phi}(\boldsymbol{s}'_l, o'_l) - \boldsymbol{\varphi}_i^\top \boldsymbol{\phi}(\boldsymbol{s}_l, o_l)$
3: $\quad$ *// Locally sampled expected value of the gradient of the MSPBE*
4: $\quad \boldsymbol{\sigma}'_i \leftarrow \boldsymbol{\varphi}_i + \alpha_{n_{ep}} \sum_{l \in \mathcal{N}_i} c_{l,i} \left(\delta'_l \boldsymbol{\phi}(\boldsymbol{s}_l, o_l) - \gamma^{K_l} \boldsymbol{\phi}(\boldsymbol{s}'_l, o'_l) (\boldsymbol{\phi}(\boldsymbol{s}_l, o_l)^\top \boldsymbol{\chi}_i) \right)$
5: $\quad \boldsymbol{\varphi}'_i \leftarrow \sum_{l \in \mathcal{N}_i} a_{l,i} \boldsymbol{\sigma}'_l$
6: $\quad$ *// Local long-term estimate of the global LMS solution*
7: $\quad \boldsymbol{\varsigma}'_i \leftarrow \boldsymbol{\chi}_i + \beta_{n_{ep}} \sum_{l \in \mathcal{N}_i} c_{l,i} \left(\delta'_l - \boldsymbol{\phi}(\boldsymbol{s}_l, o_l)^\top \boldsymbol{\chi}_i \right) \boldsymbol{\phi}(\boldsymbol{s}_l, o_l)$
8: $\quad \boldsymbol{\chi}'_i \leftarrow \sum_{l \in \mathcal{N}_i} a_{l,i} \boldsymbol{\varsigma}'_l$
9: $\quad$ **return** $(\boldsymbol{\sigma}'_i, \boldsymbol{\varphi}'_i, \boldsymbol{\varsigma}'_i, \boldsymbol{\chi}'_i)$
10: **end procedure**

The major parameter vectors $(\boldsymbol{\varphi}_i, \boldsymbol{\chi}_i)$ can be learned by combining the intermediate estimates $(\boldsymbol{\sigma}'_l, \boldsymbol{\varsigma}'_l)$ obtained from the neighbors $l \in N_i$ as shown in lines 5 and 8 of Algorithm 10. As a result, we design the total learning framework as shown in Algorithm 11.

Here, we briefly explain our distributed multi-agent reinforcement learning (MARL) algorithm in the SMDP framework as shown in Algorithm 11. Firstly, initial conditions for all the episodes are randomly selected depending on a scenario, and all the learning parameter vectors are initialized (line 2). Every agent makes a decision based on epsilon-greedy exploration at the beginning of the SMDP time step across all episodes (lines 6 and 13). The reward R_k for each SMDP time step k is stored until the termination condition is activated (line 23). When the termination condition is activated, the average reward $\bar{R}_i$ is calculated (line 11), and the terminal feature state $\boldsymbol{s}'_i$ and next candidate option o'_i are determined (lines 12 and 13). If there is no neighborhood, the agent i individually learns its learning parameter vectors $(\boldsymbol{\varphi}_i, \boldsymbol{\chi}_i)$ by using GQ-learning (line 19). If there exist neighbors, the agent i collects the information $(\boldsymbol{\sigma}_l, \boldsymbol{\varsigma}_l, \bar{R}_l, \boldsymbol{s}_l, o_l, \boldsymbol{s}'_l, K_l)$ via local communication, $\forall l \in \mathcal{N}_i$ (line 15). Then, the option o'_l for the agent l is selected (line 16). Afterward, the learning parameter vectors $(\boldsymbol{\sigma}_i, \boldsymbol{\varphi}_i, \boldsymbol{\varsigma}_i, \boldsymbol{\chi}_i)$ are updated by using cooperative GQ-learning (line 17).

Algorithm 11 Distributed multi-agent reinforcement learning in the SMDP framework

procedure DISTRIBUTED MARL IN SMDP FRAMEWORK
  Setup N_{ep} initial conditions, and $(\boldsymbol{\sigma}_i, \boldsymbol{\varphi}_i, \boldsymbol{\varsigma}_i, \boldsymbol{\chi}_i) \leftarrow \mathbf{0}^\top, \forall i \in \mathcal{I}$
  for every episode $n_{ep} = 1, 2, \ldots, N_{ep}$ **do**
    Start from the n_{ep}-th initial conditions, and $k \leftarrow 0, R_k \leftarrow 0$
    Measure initial feature state vector $\boldsymbol{s}_i$
    $o_i \leftarrow \begin{cases} o \in \arg\max_{\bar{o}} \boldsymbol{\phi}(\boldsymbol{s}_i, \bar{o})^\top \boldsymbol{\varphi}_i & \text{with probability } 1 - \epsilon_{n_{ep}} \text{ (exploit)} \\ \text{a uniform random option} \in \mathcal{O} & \text{with probability } \epsilon_{n_{ep}} \text{ (explore)} \end{cases}$
    for each agent i, every time step $t = 0, \Delta t, \ldots, T$ **do**
      Apply o_i and update robot's position/orientation
      if termination condition is activated **then**
        Set SMDP terminal time step $K_i \leftarrow k$
        $\bar{R}_i \leftarrow \frac{1}{K_i} \sum_{k=0}^{K_i} R_k$
        Measure terminal feature state vector $\boldsymbol{s}'_i$
        $o'_i \leftarrow \begin{cases} o \in \arg\max_{\bar{o}} \boldsymbol{\phi}(\boldsymbol{s}'_i, \bar{o})^\top \boldsymbol{\varphi}_i & \text{with probability } 1 - \epsilon_{n_{ep}} \text{ (exploit)} \\ \text{a uniform random option} \in \mathcal{O} & \text{with probability } \epsilon_{n_{ep}} \text{ (explore)} \end{cases}$
        if $\mathcal{N}_i \neq \varnothing$ **then**
          Collect information $(\boldsymbol{\sigma}_l, \boldsymbol{\varsigma}_l, \bar{R}_l, \boldsymbol{s}_l, o_l, \boldsymbol{s}'_l, K_l)$ via local communication, $\forall l \in \mathcal{N}_i$
          $o'_l \leftarrow \arg\max_{\bar{o}} \boldsymbol{\phi}(\boldsymbol{s}'_l, \bar{o})^\top \boldsymbol{\varphi}_i$
          $(\boldsymbol{\sigma}_i, \boldsymbol{\varphi}_i, \boldsymbol{\varsigma}_i, \boldsymbol{\chi}_i) \leftarrow$C-GQ-LEARNING$(\boldsymbol{\varphi}_i, \boldsymbol{\chi}_i, \boldsymbol{\sigma}_l, \boldsymbol{\varsigma}_l, \bar{R}_l, \boldsymbol{s}_l, o_l, \boldsymbol{s}'_l, o'_l, K_l, \alpha_{n_{ep}}, \beta_{n_{ep}})$
        else
          $(\boldsymbol{\varphi}_i, \boldsymbol{\chi}_i) \leftarrow$GQ-LEARNING$(\boldsymbol{\varphi}_i, \boldsymbol{\chi}_i, \bar{R}_i, \boldsymbol{s}_i, o_i, \boldsymbol{s}'_i, o'_i, K_i, \alpha_{n_{ep}}, \beta_{n_{ep}})$
        end if
        Set $\boldsymbol{s}_i \leftarrow \boldsymbol{s}'_i, o_i \leftarrow o'_i$, and $k \leftarrow 0, R_k \leftarrow 0$
      else
        Update reward $R_{k+1} \leftarrow r_t$, and $k \leftarrow k + 1, t \leftarrow t + \Delta t$
      end if
    end for
    $n_{ep} \leftarrow n_{ep} + 1$
  end for
end procedure

5.4 Distributed MARL Applied to Multi-Robot Systems

In this section, we design a state space, an option space, rewards, and event conditions before applying our proposed learning algorithm to the probabilistic engagement scenario.

5.4.1 State space $\mathcal{S}$

In the approximation architecture, feature states, which fully represent the problem, should be properly selected. Consequently, we adopt the 13 feature states as follows:

$$
\begin{aligned}
s \in \mathcal{S} = \big\{ & \text{“Weapon cooldown”}, \\
& \text{“Elapsed mission time”}, \\
& \text{“Inside of attack range”}, \\
& \text{“Type of the first allocated target”}, \\
& \text{“Type of the closest threat”}, \\
& \text{“Type of the robot”}, \\
& \text{“Distance to the goal”}, \\
& \text{“Distance to the first allocated target”}, \\
& \text{“Distance to the closest threat”}, \\
& \text{“Survival state of the first allocated target”}, \\
& \text{“Survival state of the robot”}, \\
& \text{“Visibility of threats”}, \\
& \text{“The number of enemy units in range”} \big\}.
\end{aligned}
$$

The specifics of the feature states are described in Table 5.1. Here, we fix the number and kind of the feature states, but these feature states can be adaptively changed by employing the model learning scheme called incremental feature dependency discovery (iFDD) [21, 59].

Table 5.1: Features considered for function approximation

Feature	Description	Value
$\mathbf{W}_i$	Weapon cooldown	$\{0, 1\}$
$\mathbf{T}_m$	Elapsed mission time	$\mathbb{R}$
$\mathbf{A}_i$	Inside of attack range	$\{0, 1\}$
k_{j_1}	Type of the first allocated target	$\{0 \text{ or } 1, 2, 3\}$
k_{j^-}	Type of the closest threat	$\{0 \text{ or } 1, 2, 3\}$
k_i	Type of the robot	$\{1, 2, 3\}$
$\mathbf{D}_g$	Distance to the goal	$\mathbb{R}$
$\mathbf{D}_{j_1}$	Distance to the first allocated target	$\mathbb{R}$
$\mathbf{D}_{j^-}$	Distance to the closest threat	$\mathbb{R}$
$\mathbf{S}_{j_1}$	Survival state of the first allocated target	$\{0, 1\}$
$\mathbf{S}_i$	Survival state of the robot	$\{0, 1\}$
$\mathbf{V}_j$	Visibility of threats	$\{0, 1\}$
N_e	The number of enemy units in range	$\{0, 1, \ldots, N_t\}$

From the selected feature state vector $\boldsymbol{s}$, we obtain a second order polynomial basis function vector $\boldsymbol{\phi}(\boldsymbol{s})$ by (5.12).

5.4.2 Option space $\mathcal{O}$

The five types of behaviors introduced in Sect. 5.1 are directly defined as options:

$$
\begin{aligned}
o \in \mathcal{O} = \big\{ & \text{“Engagment”}, \\
& \text{“Concealment”}, \\
& \text{“Move to the allocated target”}, \\
& \text{“Move to the goal”}, \\
& \text{“Standby”} \big\}.
\end{aligned}
$$

5.4.3 Reward R

The reward R is designed by considering the feature states. We focus on five factors: the threat level, the possibility of attack, the destruction of the threat, survivability, and the mission time.

- *The threat level*

  The robot i has the penalty depending on the type of the robot i and the type of the target j^-. Here, j^- denotes the target j which has the minimum distance from the robot i. Then, we have the reward r_1 as follows:

$$r_1 = p_{k_i,k_{j^-}} \exp\left(-\frac{||\boldsymbol{x}_i - \boldsymbol{x}_{j^-}||}{v_{k_{j^-}}}\right) \tag{5.40}$$

  where $\boldsymbol{P} = \begin{bmatrix} -2 & -2 & -1 \\ -1 & -2 & -2 \\ -1 & -1 & -2 \end{bmatrix}$ and $\boldsymbol{v} = [0.5\ 0.8\ 1.0]^\top$. So, the robot i has a high penalty when the robot is close to the difficult threat j^-.

- *The possibility of attack*

  The robot i has a high reward when the robot i can attack the first allocated threat j_1 ($\mathbf{W}_i = 0$ and $\mathbf{A}_i = 1$), and the distance between the target point $\boldsymbol{x}_{\text{target}}$ given by the behavior module and the threat j_1 is increased. If the robot i cannot attack the threat j_1 due to the weapon cooldown time ($\mathbf{W}_i = 1$), the robot i has a high reward when the distance between the robot i and the threat j_1 is increased.

$$r_2 = \begin{cases} 2\exp\left(\frac{-||\boldsymbol{x}_{\text{target}} - \boldsymbol{x}_{j_1}||}{v_{j_1}}\right) & \text{if } \mathbf{W}_i = 0 \wedge \mathbf{A}_i = 1 \\ \exp\left(\frac{-||\boldsymbol{x}_{\text{target}} - \boldsymbol{x}_{j_1}||}{v_{j_1}}\right) & \text{if } \mathbf{W}_i = 0 \wedge \mathbf{A}_i = 0 \\ 1 - \exp\left(\frac{-||\boldsymbol{x}_i - \boldsymbol{x}_{j_1}||}{v_{j_1}}\right) & \text{if } \mathbf{W}_i = 1 \end{cases} \tag{5.41}$$

- *The destruction of the threat*

  When the first allocated target j_1 is killed by the robot i ($\mathbf{S}_{j_1} = 0$), a reward is given

by the following:

$$r_3 = \begin{cases} 10 & \text{if } \mathbf{S}_{j_1} = 1 \rightarrow \mathbf{S}_{j_1} = 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.42}$$

- *Survivability*

  When the survival state of the robot i is changed ($\mathbf{S}_i = 1 \rightarrow \mathbf{S}_i = 0$), a penalty is given. Here, $\mathbf{S}_i = 1$ means that the robot i is alive, and when the robot i is dead, then $\mathbf{S}_i = 0$.

$$r_4 = \begin{cases} -10 & \text{if } \mathbf{S}_i = 1 \rightarrow \mathbf{S}_i = 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.43}$$

- *The mission time*

  The reward is designed for preventing the robot i moves to the goal point within the mission time.

$$r_5 = \begin{cases} -5\exp\left(-||\boldsymbol{x}_{\text{target}} - \boldsymbol{x}_{\text{goal}}||\right) & \text{if } T_m > t \\ 5\exp\left(-||\boldsymbol{x}_{\text{target}} - \boldsymbol{x}_{\text{goal}}||\right) & \text{otherwise} \end{cases} \tag{5.44}$$

  where T_m denotes the mission time.

Finally, we have the total reward $R = r_1 + r_2 + r_3 + r_4 + r_5$.

5.4.4 Event conditions $\mathcal{E}$

As mentioned in Sect. 5.2, we should consider the event when an option is initiated or terminated. To this end, we select the termination conditions related with the SMDP time step k and the feature states ($\mathbf{W}_i, \mathbf{A}_i, \mathbf{D}_{j^-}, \mathbf{S}_{j_1}, \mathbf{S}_i$). So, we define the termination conditions

by the following:

$$\xi_1 = \begin{cases} 1 & \text{if } k > 20 \\ 0 & \text{otherwise} \end{cases} \tag{5.45}$$

$$\xi_2 = \begin{cases} 1 & \text{if } (\mathbf{W}_i = 0 \rightarrow \mathbf{W}_i = 1) \vee (\mathbf{W}_i = 1 \rightarrow \mathbf{W}_i = 0) \\ 0 & \text{otherwise} \end{cases} \tag{5.46}$$

$$\xi_3 = \begin{cases} 1 & \text{if } (\mathbf{A}_i = 0 \rightarrow \mathbf{A}_i = 1) \vee (\mathbf{A}_i = 1 \rightarrow \mathbf{A}_i = 0) \\ 0 & \text{otherwise} \end{cases} \tag{5.47}$$

$$\xi_4 = \begin{cases} 1 & \text{if } \mathbf{D}_{j^-} > 2v_{k_{j^-}} \rightarrow \mathbf{D}_{j^-} \leq 2v_{k_{j^-}} \\ 0 & \text{otherwise} \end{cases} \tag{5.48}$$

$$\xi_5 = \begin{cases} 1 & \text{if } \mathbf{S}_{j_1} = 1 \rightarrow \mathbf{S}_{j_1} = 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.49}$$

$$\xi_6 = \begin{cases} 1 & \text{if } \mathbf{S}_i = 1 \rightarrow \mathbf{S}_i = 0 \\ 0 & \text{otherwise} \end{cases} . \tag{5.50}$$

If at least one of the termination conditions, $\xi_i, i = 1, \ldots, 6$, become 1, then the SMDP learning routine is performed (lines 10–21 of Algorithm 11).

5.5 Empirical Results

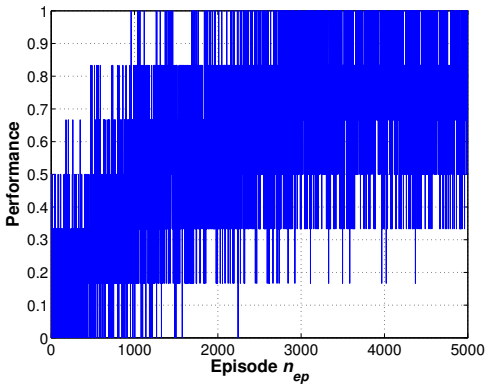
In this section, we apply the proposed MARL algorithm to the probabilistic engagement scenario and discuss the main learning results.

Before starting distributed MARL, we set up the parameters for each component of the heterogeneous multi-robot systems. For CBBA, the maximum number of tasks for each robot is set to 3 ($L_t = 3$), and we use the optimal scoring matrix obtained by from Sect. 3.3. CBBA is periodically performed every 0.5 seconds ($\Delta t = 0.5$). Dec-VMCPSO also runs every 0.5 seconds for path planning. To speed up the integrated simulation, the number of LGL nodes and candidate PCP vectors is set to 10, and we set the maximum number

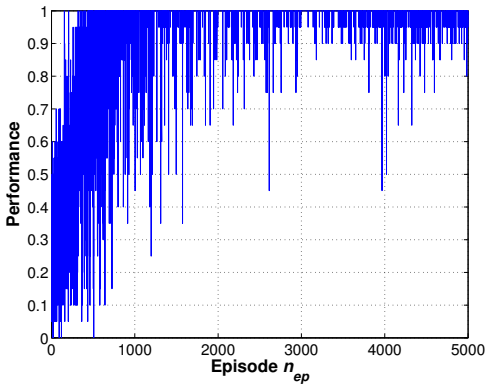
of VMCPSO iterations as 150. In each episode, two flying robots perform surveillance and reconnaissance missions and send the threat information to the command and control vehicle building the threat map and the visibility map. A total of six ground robots mainly perform infiltration operations while overwhelming the threats and maximizing the team survivability during the mission time T_m. The type of the first and second robots is set to 1 ($k_1, k_2 = 1$), and we determine the type of the third and forth robots as 2 ($k_3, k_4 = 2$). For the rest of the robots, we set the type as 3 ($k_5, k_6 = 3$) . As mentioned in Sect. 2.2, the third type of robot has powerful capabilities such as the longest striking distance, and the first type of robot is weak compared to the other types of robot. To select the most appropriate behavior of each the robots for the given situation, we apply distributed MARL with total 5000 episodes and consider completely different situations in the battlefield for each episode. To this end, we fix the number of the ground robots and the threats ($N_u = 6, N_t = 20$), but the position of the ground robots and the position and type of the threats are randomly selected in each episode. The battlefield contains total 20 obstacles ($N_{\text{obs}} = 20$), and the obstacles' positions are also randomly selected for each episode. In the learning process, we assume that each robot utilizes experience of the other robots which have the same type, and the reliability of information among the robots is guaranteed. Therefore, we have the following two matrices:

$$\boldsymbol{A} = \boldsymbol{C} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.51}$$

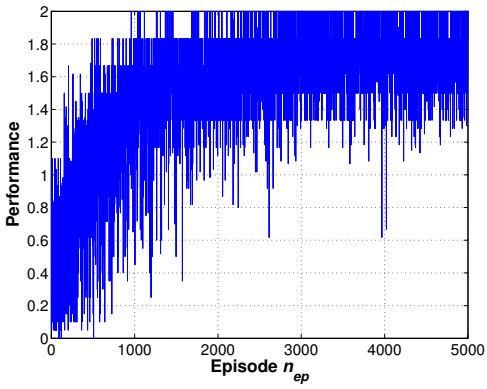
where $(a_{i,j}, c_{i,j})$ represent the reliability of information between the robot i and the robot j. When a robot is not available due to malfunction or destruction, the corresponding element of $\boldsymbol{A}$ and $\boldsymbol{C}$ is set to 0.



(a) Survival rate



(b) Destruction rate



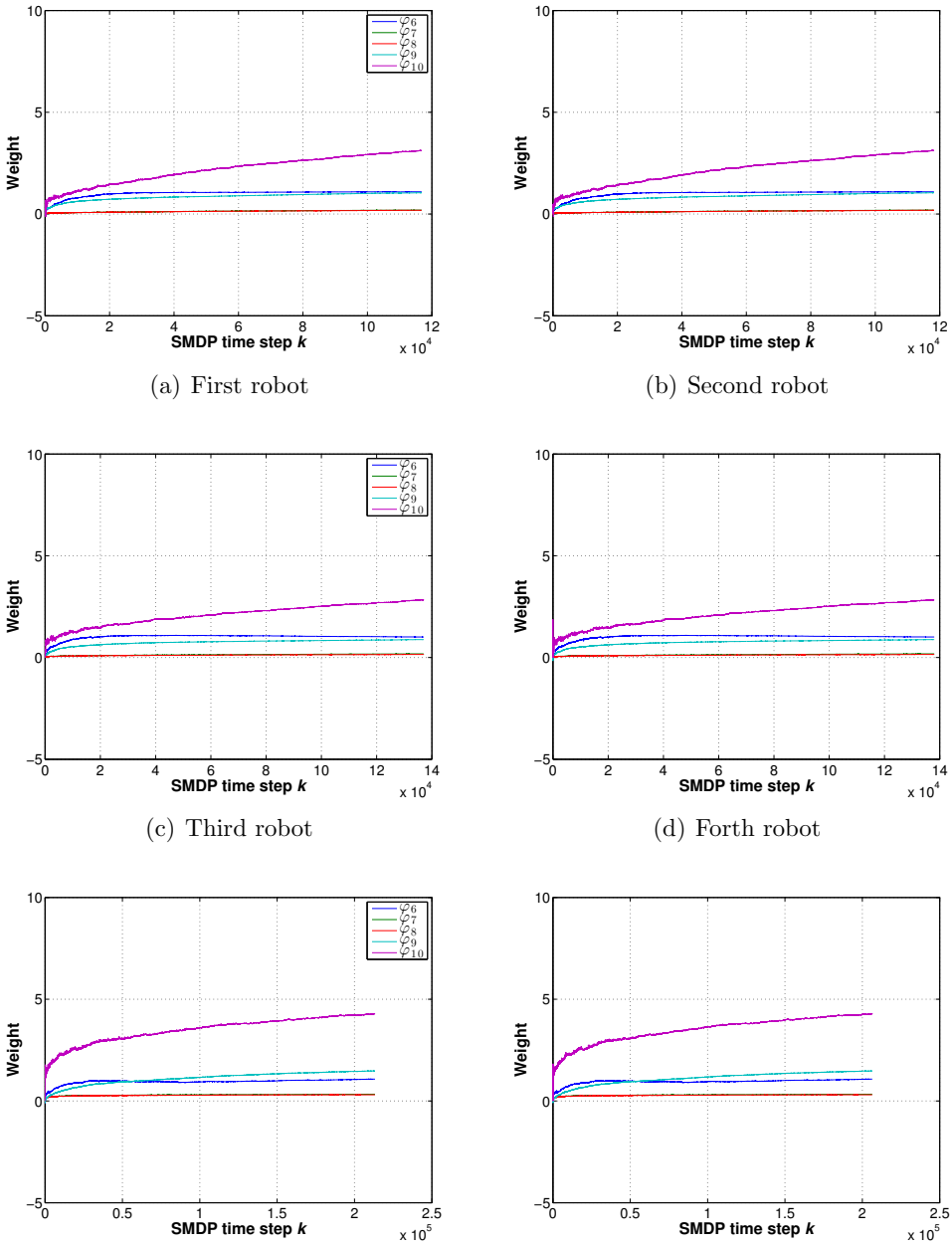
(c) Survival rate + destruction rate

Figure 5.1: Change in values of performance measures as the number of episodes increases

To observe the effect of learning, we select three performance measures, such as the survival rate ($^{N_s}/_{N_u}$), the destruction rate ($^{N_d}/_{N_t}$), and the sum of the survival and destruction rate. Figure 5.1 shows the change in values of the performance measures as the number of episodes increases. Figure 5.1(a) shows that the survival rate of the ground robots is increased as the number of episodes increases. The trend of survival rate is generally lower than the trend of destruction rate because two robots of type 1 are not good enough to confront new scenarios due to weak attack capabilities and viability compared to the other types. The destruction rate of the threats is also increased as shown in Fig. 5.1(b). Although

sometimes the destruction rate is low because we consider random scenarios throughout all episodes, the robots suppress almost all threats after about 2000 episodes. As a result, overall performance graphs show noisy inclinations due to the randomness of the scenarios, but the proposed learning algorithm makes the performance index increase and converge to the threshold value as the number of episodes increases.

Figure 5.2 shows that the change in values of weights for each robot as the SMDP time step increases. Here, we select five weights (φ_6–φ_{10}) among weights of the learning parameter vector $\boldsymbol{\varphi}_i$ for the robot i to observe variations of weights for the robot i which communicates with another robot. The robots in the same communication network have almost the same weight values throughout every SMDP time step. In other words, the distributed MARL algorithm based on diffusion adaptation makes the robots have almost the same option-value function by sharing their experiences. In conclusion, the proposed algorithm worked well to achieve behavior coordination of the robots by sharing their experiences for the combat situations in the SMDP framework.



(a) First robot

(b) Second robot

(c) Third robot

(d) Forth robot

(e) Fifth robot

(f) Sixth robot

Figure 5.2: Change in values of weights for each robot as the SMDP time step k increases

5.6 Analysis and Discussion

In the previous section, we found the optimal policy to select a behavior of each robot for the given environmental feature information. In this section, we will analyze the behavior trend of the robots with the optimal policy given by distributed MARL.

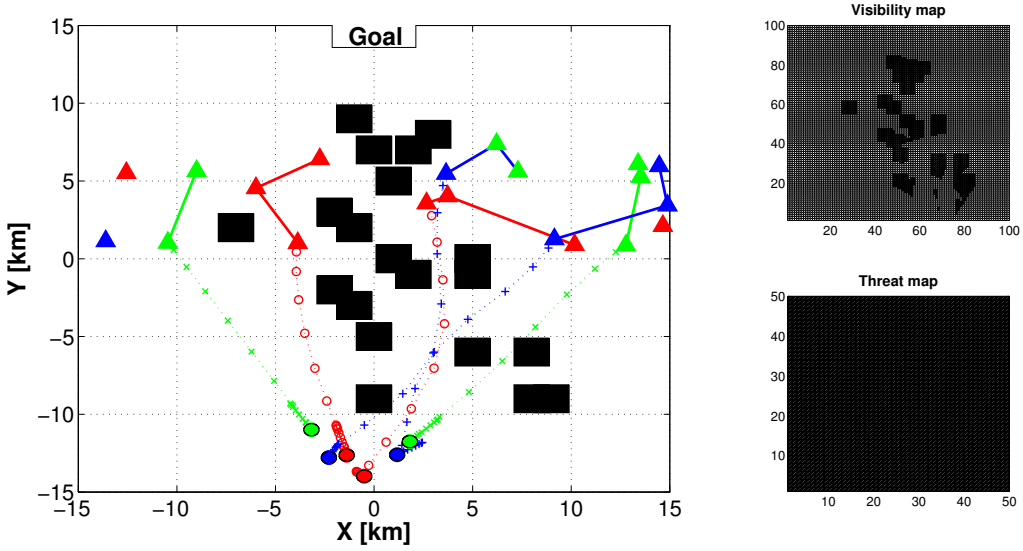
For the given feature vector $\boldsymbol{s}_i$, the optimal policy of the robot i is obtained by the following:

$$o_i^* = \arg\max_{\bar{o}} \boldsymbol{\phi}(\boldsymbol{s}_i, \bar{o})^\top \boldsymbol{\varphi}_i^*. \tag{5.52}$$

In order to observe whether the optimal policy works well or not in an arbitrary scenario, we consider one scenario as shown in Fig. 5.3. The integrated simulation is performed similar to the previous learning setup in a randomly generated scenario, but one different thing is that the robots make a decision at each time step by using (5.52).

Figures 5.3–5.9 show the integrated simulation results obtained from CBBA, GPP/LPP of Dec-VMCPSO, and the visibility/threat map. Initially, each ground combat robot selects the maximum three threats to be easily destroyed depending on its type by using the optimal engagement strategy, i.e, the combination of CBBA and the optimal scoring matrix. At the same time, the visibility map and the threat map are built by utilizing the information of the threats and environment gathered from two flying robots and a command and control vehicle. At time step k, when the robot i selects an option $o_i^*(k)$ by using (5.52), the target point is determined with respect to the selected option. Then, the global and local paths to reach the target point are generated by Dec-VMCPSO. As shown in Figs. 5.3–5.9, all the robots sequentially destroy the target while dynamically assigning tasks and avoiding the obstacles. Consequently, the threat level and the visible regions of the threat dwindle away over time.

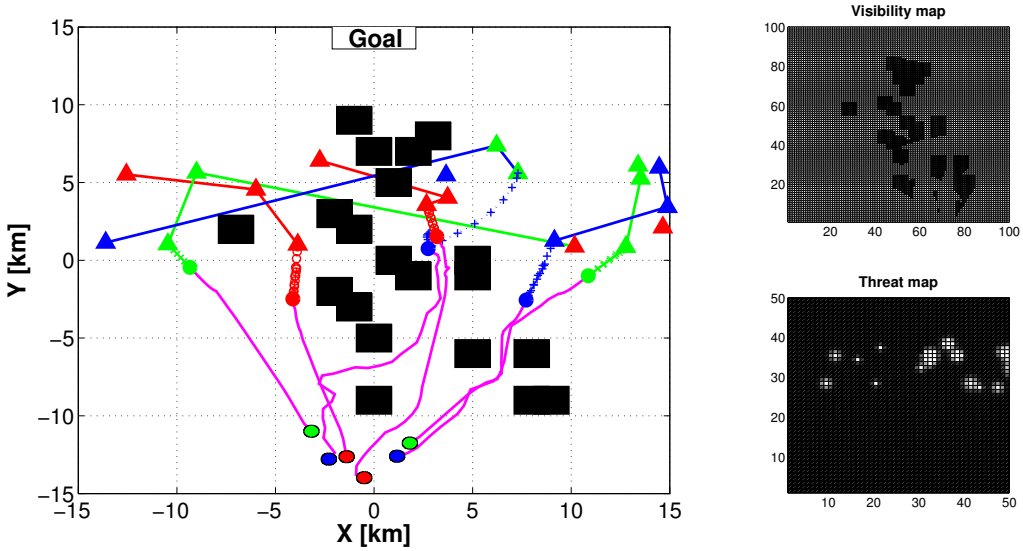
Figure 5.10 shows change in options for each robot as the SMDP time step k increases. All the robots adaptively switch their options depending on situations they are facing. When



(a) (b)

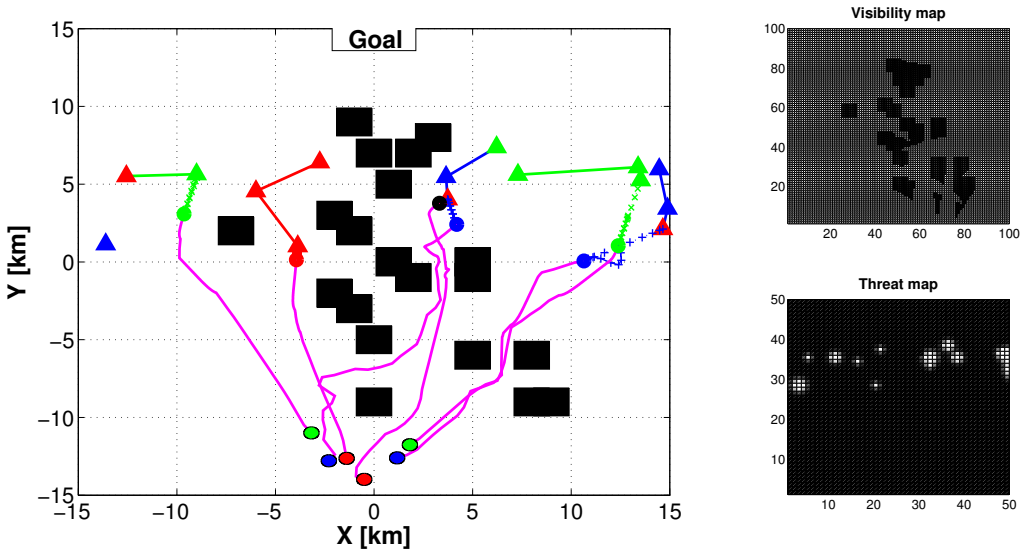
Figure 5.3: Integrated simulation results at the SMDP time step $k = 0$

the robot is far away from the target, it quickly moves to the target by selecting the third option. After that, when the robot is close enough to approach the target, the *Engagement* option is activated. In addition, if the robot does not have any tasks to be handled, its option is set to *Move to the goal*. The other options such as *Concealment* and *Standby* do not appear in this scenario. We expect that the different results can be obtained when both the robots and the threats learn their combat policies competitively. For example, when the robot or threat is unfavorable to attack, they can try to move behind an obstacle for achieving concealment. As a result, we obtained the proper behavior patterns by using the proposed learning algorithm, and it showed the promising performance in our combat scenario.



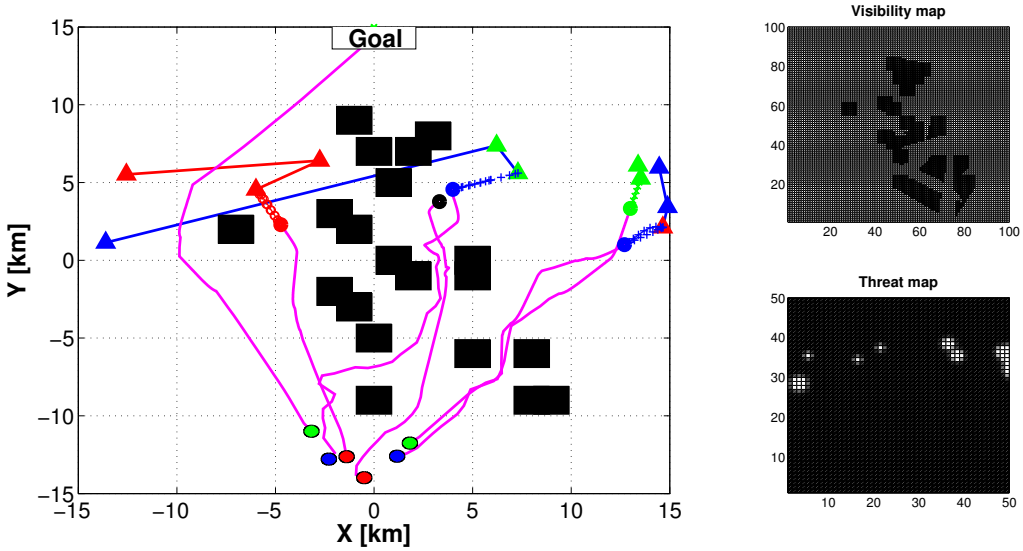
(a) (b)

Figure 5.4: Integrated simulation results at the SMDP time step $k = 100$



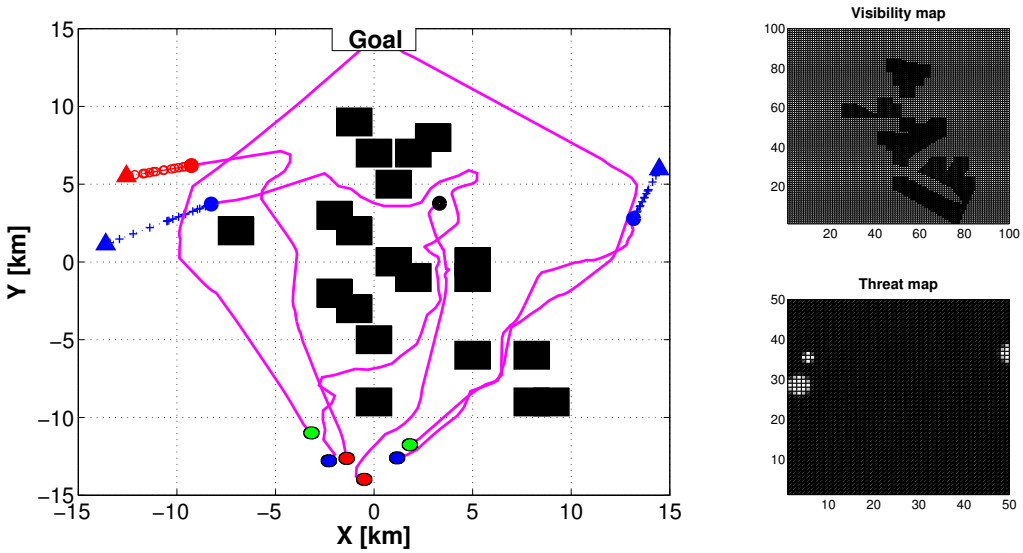
(a) (b)

Figure 5.5: Integrated simulation results at the SMDP time step $k = 205$



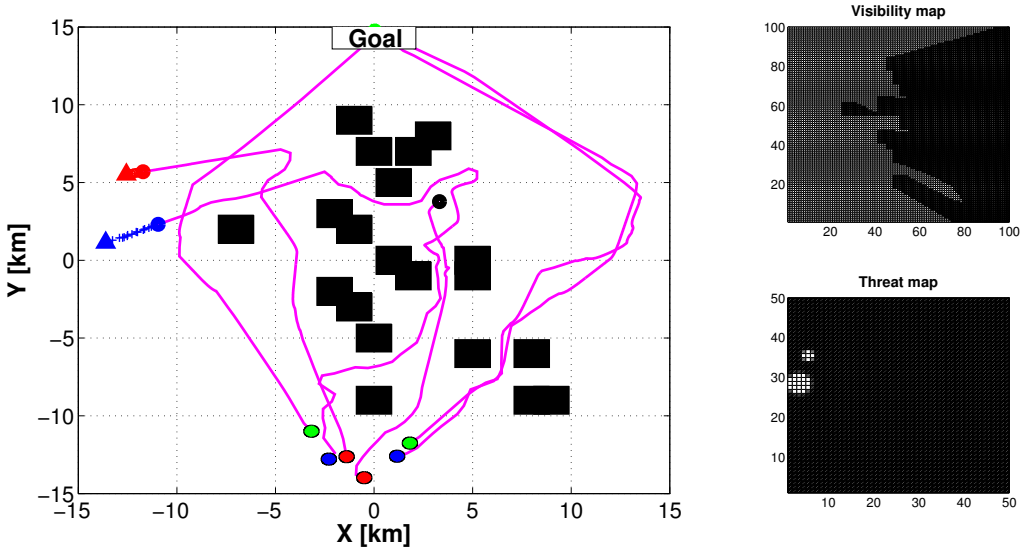
(a) (b)

Figure 5.6: Integrated simulation results at the SMDP time step $k = 300$



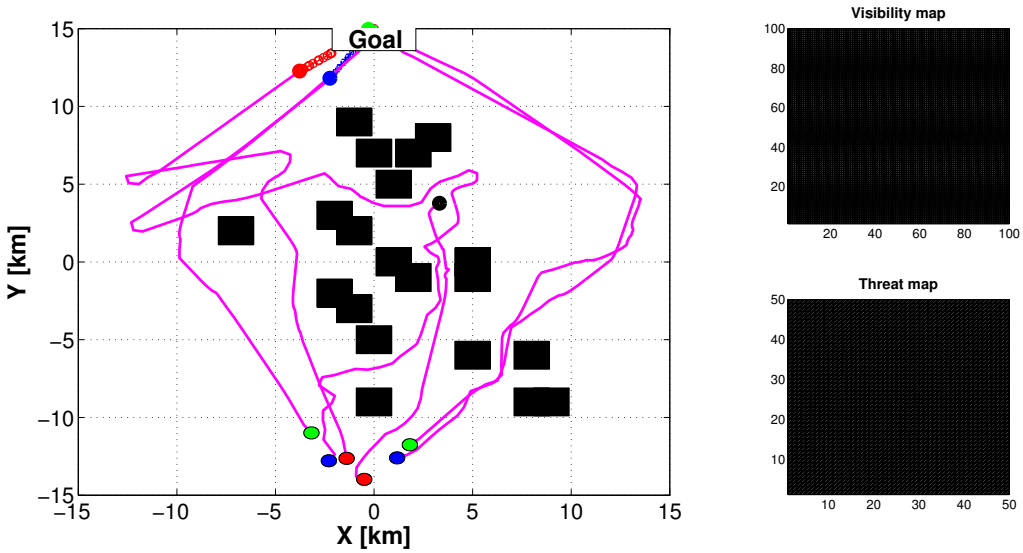
(a) (b)

Figure 5.7: Integrated simulation results at the SMDP time step $k = 400$



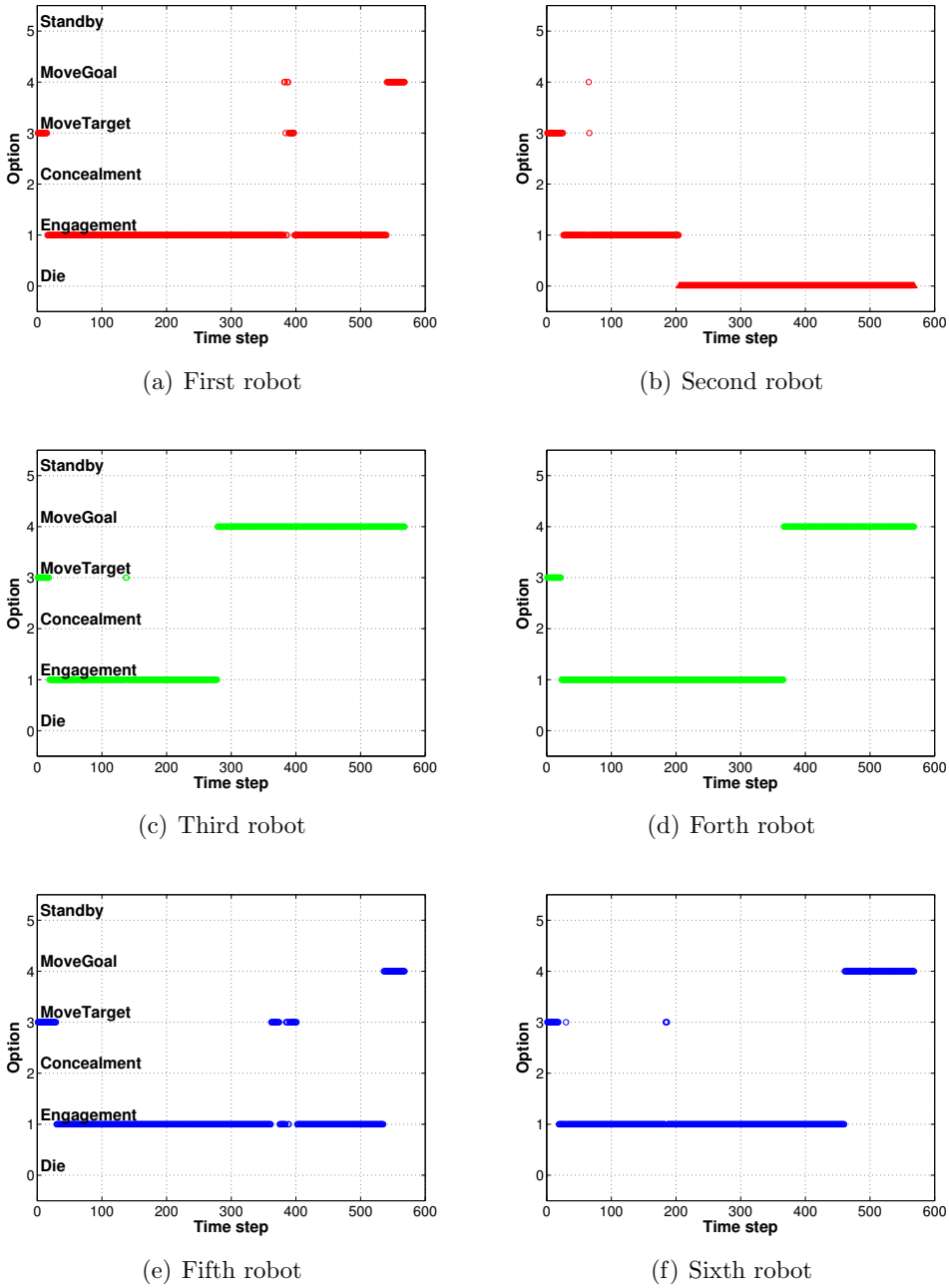
(a) (b)

Figure 5.8: Integrated simulation results at the SMDP time step $k = 500$



(a) (b)

Figure 5.9: Integrated simulation results at the SMDP time step $k = 560$



(a) First robot

(b) Second robot

(c) Third robot

(d) Forth robot

(e) Fifth robot

(f) Sixth robot

Figure 5.10: Change in options for each robot as the SMDP time step k increases

6

Conclusions

This thesis presents the development and implementation of decision-making methods consisting of mission planning, path planning, and behavior learning. The proposed methods were used to control multi-robot systems in probabilistic and ever-changing battlefield situations.

We first designed the cooperative control architecture of autonomous combat systems including the command and control vehicle, the multiple ground combat robots, and the multiple surveillance aerial robots. We specified the relationship between components of combat systems and defined the important factor technologies such as mission planning, path planning, and learning for behavior coordination.

In multi-robot mission planning, we proposed the episodic parameter optimization (EPO) method that utilizes reinforcement learning (RL) and particle swarm optimization (PSO) to improve the performance of existing consensus-based bundle algorithm (CBBA) in probabilistic engagement scenarios by optimizing the scoring matrix reflecting the heterogeneity between the ground robots and targets. Because both the robots and targets have different attack capabilities depending on the types, each of them has to choose the

most suitable targets. Therefore, the scoring matrix linked to the performance of CBBA should be optimized. The performance measure considered here was the team survivability of the ground robots. After several iterations of EPO elapsed, the optimal scoring matrix was obtained for enhancing the performance of CBBA.

To deal with multi-robot path planning, we proposed the numerical trajectory optimization method called VMCPSO: VMC transforms a typical full space optimal problem to a subspace optimal problem and the original problem is changed into finding the optimal PCPs, and PSO is used to optimize PCPs. Before applying the VMCPSO algorithm, the constrained trajectory optimization problem is converted into the unconstrained trajectory optimization problem by using penalty functions. Then, the candidate PCPs converge to the optimal solution through local and global interactions with the other candidates as the number of iterations increases. We also extended the VMCPSO algorithm by decentralization in order to achieve cooperative missions of multiple robots efficiently. The numerical simulation and experimental results showed that the optimal paths considering the terminal time and angle are effectively generated by decentralized VMCPSO.

The distributed multi-agent reinforcement learning (MARL) algorithm in the semi-Markov Decision Process framework was proposed to solve the behavior coordination problem for the ground combat robots in engagement scenarios. We designed five types of behaviors, and each behavior was implemented by using the mission and path planning algorithms developed in Chapters 3 and 4. The threat map and the visibility map were also considered to build the behaviors. The mission performances were maximized while increasing the number of episodes for distributed MARL in completely different engagement scenarios, and we found that the most probable behavior of the robots was properly selected in combat situations after learning the optimal policy.

In conclusion, we could reach the goal of the multi-robot group such as maximizing the team survivability in probabilistic combat situations by synthesizing each ingredient of the proposed techniques. The proposed decision-making algorithms designed in a distributed manner and can be applied to the robots having different characteristics.

References

[1] M. Alighanbari and J. P. How. Decentralized task assignment for unmanned aerial vehicles. In *Proceedings of the 44th IEEE Conference on Decision and Control-European Control Conference*, pages 5668–5673, Dec. 2005.

[2] David Anisi, John Robinson, and Petter Ögren. On-line trajectory planning for aerial vehicles: A safe approach with guaranteed task completion. In *Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit*, 2006.

[3] G. Arslan, J. R. Marden, and J. S. Shamma. Autonomous vehicle-target assignment: A game-theoretical formulation. *Journal of Dynamic Systems, Measurement, and Control*, 129(5):584–596, 2007.

[4] R. W. Beard, T. W. McLain, M. A. Goodrich, and E. P. Anderson. Coordinated target assignment and intercept for unmanned air vehicles. *IEEE Transactions on Robotics and Automation*, 18(6):911–922, Dec. 2002.

[5] John Bellingham, Michael Tillerson, Arthur Richards, and Jonathan P. How. Multi-task allocation and path planning for cooperating UAVs. In Sergiy Butenko, Robert Murphey, and Panos M. Pardalos, editors, *Cooperative Control: Models, Applications and Algorithms*, volume 1 of *Cooperative Systems*, pages 23–41. Springer US, 2003.

[6] Luca F. Bertuccelli, Han-Lim Choi, Peter Cho, and Jonathan P. How. Real-time multi-uav task assignment in dynamic and uncertain environments. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2009.

[7] John T. Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, 1998.

[8] Kevin Bollino and L. Ryan Lewis. Collision-free multi-uav optimal path planning and cooperative control for tactical applications. In *Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008.

[9] F. Borrelli, T. Keviczky, and G. J. Balas. Collision-free UAV formation flight using decentralized optimization and invariant sets. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, volume 1, pages 1099–1104, 2004.

[10] Lucian Busoniu, Robert Babuska, Bart De Schutter, and Damien Ernst. *Reinforcement learning and dynamic programming using function approximators*. CRC Press, 2010.

[11] Jianshu Chen, Sheng-Yuan Tu, and Ali H. Sayed. Distributed optimization via diffusion adaptation. In *Proceedings of the 4th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, pages 281–284, 2011.

[12] H. Choi, L. Brunet, and J. P. How. Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics*, 25(4):912–926, Aug. 2009.

[13] H. Choi, A. K. Whitten, and J. P. How. Decentralized task allocation for heterogeneous teams with cooperation constraints. In *Proceedings of the American Control Conference*, pages 3057–3062, July 2010.

[14] T. H. Chung and J. W. Burdick. Analysis of search decision making using probabilistic search strategies. *IEEE Transactions on Robotics*, 28(1):132–144, Feb. 2012.

[15] M. Clerc and J. Kennedy. The particle swarm explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002.

[16] Christopher L. Darby, William W. Hager, and Anil V. Rao. Direct trajectory optimization using a variable low-order adaptive pseudospectral method. *Journal of Spacecraft and Rockets*, 48(3):433–445, 2011.

[17] Vishnu R. Desaraju and Jonathan P. How. Decentralized path planning for multi-agent teams with complex constraints. *Autonomous Robots*, 32(4):385–403, 2012.

[18] Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13(1):227–303, 2000.

[19] Yong Duan, Qiang Liu, and XinHe Xu. Application of reinforcement learning in robot soccer. *Engineering Applications of Artificial Intelligence*, 20(7):936–950, 2007.

[20] Fariba Fahroo and I. Michael Ross. Costate estimation by a legendre pseudospectral method. *Journal of Guidance, Control, and Dynamics*, 24(2):270–277, 2001.

[21] Alborz Geramifard, Finale Doshi, Joshua Redding, Nicholas Roy, and Jonathan P. How. Online discovery of feature dependencies. In *Proceedings of the 28th International Conference on Machine Learning*, pages 881–888, 2011.

[22] Alborz Geramifard, Joshua Redding, and Jonathan P. How. Intelligent cooperative control architecture: A framework for performance improvement using safe learning. *Journal of Intelligent & Robotic Systems*, 72(1):83–103, 2013.

[23] Alborz Geramifard, Joshua D. Redding, Joshua Joseph, Nicholas Roy, and Jonathan P. How. Model estimation within planning and learning. In *Proceedings of the American Control Conference*, pages 793–799, June 2012.

[24] Hongliang Guo and Yan Meng. Distributed reinforcement learning for coordinate multi-robot foraging. *Journal of Intelligent & Robotic Systems*, 60(3–4):531–551, 2010.

[25] Yan Jin, A. A. Minai, and M. M. Polycarpou. Cooperative real-time search and task allocation in uav teams. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, volume 1, pages 7–12, Dec. 2003.

[26] Timothy R. Jorris and Richard G. Cobb. Three-dimensional trajectory optimization satisfying waypoint and no-fly zone constraints. *Journal of Guidance, Control, and Dynamics*, 32(2):551–572, 2009.

[27] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

[28] Tamás Keviczky, Francesco Borrelli, and Gary J Balas. A study on decentralized receding horizon control for decoupled systems. In *Proceedings of the American Control Conference*, volume 6, pages 4921–4926, 2004.

[29] D. Kim and J. Kim. A real-time limit-cycle navigation method for fast mobile robots and its application to robot soccer. *Robotics and Autonomous Systems*, 42(1):17–30, 2003.

[30] S. Kim and Y. Kim. Optimum design of three-dimensional behavioural decentralized controller for UAV formation flight. *Engineering Optimization*, 41(3):199–224, 2009.

[31] Byung-Il Koh, Alan D. George, Raphael T. Haftka, and Benjamin J. Fregly. Parallel asynchronous particle swarm optimization. *International Journal for Numerical Methods in Engineering*, 67(4):578–595, 2006.

[32] K. Kovac, I. Zivkovic, and B. D. Basic. Simulation of multi-robot reinforcement learning for box-pushing problem. In *Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference*, volume 2, pages 603–606, May 2004.

[33] Y. Kuwata and J. P. How. Cooperative distributed robust trajectory optimization using receding horizon MILP. *IEEE Transactions on Control Systems Technology*, 19(2):423–431, 2011.

[34] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How. Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology*, 17(5):1105–1118, 2009.

[35] Zheng Liu, V. M. H. Ang, and Winston Khoon-Guan Seah. Reinforcement learning of cooperative behaviors for multi-robot tracking of multiple moving targets. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1289–1294, Aug 2005.

[36] Alessandro Luca, Giuseppe Oriolo, and Marilena Vendittelli. Control of wheeled mobile robots: An experimental overview. In *Ramsete*, volume 270 of *Lecture Notes in Control and Information Sciences*, pages 181–226. Springer Berlin Heidelberg, 2001.

[37] S. V. Macua, P. Belanovic, and S. Zazo. Diffusion gradient temporal difference for cooperative reinforcement learning with linear function approximation. In *Proceedings of the 3rd International Workshop on Cognitive Information Processing*, pages 1–6, May 2012.

[38] J. R. Marden, G. Arslan, and J. S. Shamma. Joint strategy fictitious play with inertia for potential games. *IEEE Transactions on Automatic Control*, 54(2):208–220, Feb 2009.

[39] Maja J. Matarić. Reinforcement learning in the multi-robot domain. In Ronald C. Arkin and Georgev A. Bekey, editors, *Robot Colonies*, pages 73–83. Springer US, 1997.

[40] Maja J. Matarić. Learning in behavior-based multi-robot systems: policies, models, and other agents. *Cognitive Systems Research*, 2(1):81–93, 2001.

[41] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J. Zufferey, D. Floreano, and A. Martinoli. The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, pages 59–65, 2009.

[42] A. Y. Ng and M. I. Jordan. Pegasus: a policy search method for large MDPs and POMDPs. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 406–415, 2000.

[43] Edwin Olson, Johannes Strom, Ryan Morton, Andrew Richardson, Pradeep Ranganathan, Robert Goeddel, Mihai Bulic, Jacob Crossman, and Bob Marinier. Progress toward multi-robot reconnaissance and the MAGIC 2010 competition. *Journal of Field Robotics*, 29(5):762–792, 2012.

[44] Warren B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality, (Wiley Series in Probability and Statistics)*. Wiley, 2011.

[45] Josh Redding, Alborz Geramifard, and Jonathan P. How. Actor-critic policy learning in cooperative planning. In *Proceedings of the AAAI Spring Symposium: Embedded Reasoning*, 2010.

[46] Arthur Richards and Jonathan P. How. A decentralized algorithm for robust constrained model predictive control. In *Proceedings of the American Control Conference*, volume 5, pages 4261–4266, 2004.

[47] Martin Riedmiller, Thomas Gabel, Roland Hafner, and Sascha Lange. Reinforcement learning for robot soccer. *Autonomous Robots*, 27(1):55–73, 2009.

[48] Tom Schouwenaars, Jonathan How, and Eric Feron. Decentralized cooperative trajectory planning of multiple aircraft with hard safety guarantees. In *Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit*, 2004.

[49] Corey Schumacher, Phillip Chandler, Meir Pachter, and Lior Pachter. Constrained optimization for UAV task assignment. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, pages 1–14, 2004.

[50] J. F. Schutte, J. A. Reinbolt, B. J. Fregly, R. T. Haftka, and A. D. George. Parallel global optimization with the particle swarm algorithm. *International Journal for Numerical Methods in Engineering*, 61(13):2296–2315, 2004.

[51] Mandyam V. Srinivasan and Matthew Davey. Strategies for active camouflage of motion. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 259(1354):19–25, 1995.

[52] Peter Stone, Richard S. Sutton, and Gregory Kuhlmann. Reinforcement learning for robocup soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005.

[53] O. Stryk and R. Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of Operations Research*, 37(1):357–373, 1992.

[54] R. S. Sutton and A. G. Barto. *Reinforcement learning: an introduction.* MIT Press, Cambridge, Mass., 1998.

[55] Richard S. Sutton, Hamid Reza Maei, Doina Precup, Shalabh Bhatnagar, David Silver, Csaba Szepesvári, and Eric Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 993–1000, 2009.

[56] Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1—2):181–211, 1999.

[57] Claude F. Touzet. Robot awareness in cooperative mobile robot learning. *Autonomous Robots*, 8(1):87–97, 2000.

[58] Y.-H. R. Tsai, L.-T. Cheng, S. Osher, P. Burchard, and G. Sapiro. Visibility and its dynamics in a PDE based implicit framework. *Journal of Computational Physics*, 199(1):260–290, 2004.

[59] N. Kemal Ure, Girish Chowdhary, Yu Fan Chen, Jonathan P. How, and John Vian. Distributed learning for planning under uncertainty problems with heterogeneous teams. *Journal of Intelligent & Robotic Systems*, 74(1-2):529–544, 2014.

[60] N. Kemal Ure, Tuna Toksoz, Girish Chowdhary, Joshua Redding, Jonathan P. How, Matthew Vavrina, and John Vian. Experimental demonstration of multi-agent learning and planning under uncertainty for persistent missions with automated battery management. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2012.

[61] Ying Wang and C. W. de Silva. Multi-robot box-pushing: Single-agent Q-Learning vs. Team Q-Learning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3694–3699, Oct 2006.

[62] Shimon Whiteson, Matthew E. Taylor, and Peter Stone. Empirical studies in action selection with reinforcement learning. *Adaptive Behavior*, 15(1):33–50, 2007.

[63] Yunjun Xu. Virtual motion camouflage and suboptimal trajectory design. In *Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit*, 2007.

[64] Yunjun Xu. Subspace optimal control and motion camouflage. In *Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008.

[65] Yunjun Xu and Gareth Basset. Pre and post optimality checking of the virtual motion camouflage based nonlinear constrained subspace optimal control. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2009.

[66] Yunjun Xu and Gareth Basset. Sequential virtual motion camouflage method for nonlinear constrained optimal trajectory control. *Automatica*, 48(7):1273–1285, 2012.

[67] Chen Yao, Xu Chu Ding, and C. G. Cassandras. Cooperative receding horizon control for multi-agent rendezvous problems in uncertain environments. In *Proceedings of the 49th IEEE Conference on Decision and Control*, pages 4511–4516, Dec. 2010.

국문초록

본 박사학위 논문에서는 다중 로봇 시스템의 행동 조정을 위한 협업 아키텍처, 임무할당, 경로계획, 행동학습 기법을 제안하고 무인 전투 시스템에 적용하였다. 다중 로봇으로 구성된 무인 전투 시스템이 시시각각 변화하는 전시상황에서 유연하게 대처하여 통합적인 목표를 달성하기 위해서는 미리 지정된 임무의 특성 및 실제의 상황별로 적절한 의사결정을 해야 한다. 이를 위해 먼저 지휘통제 차량, 지상 전투 로봇, 감시정찰 공중 로봇 간 협업 아키텍처를 설계하였다.

다음으로 분산형 임무계획 기법인 합의 기반 번들 알고리즘을 사용하여 지상 로봇들을 적절한 임무지점(표적)으로 할당하도록 하였다. 여기서 지상 로봇 및 표적들은 종류에 따라 서로 다른 임무수행 능력을 지니고 있기 때문에 이들 간의 이질성을 반영하는 점수행렬을 사용하여 임무계획 시 반영하도록 하였다. 또한, 강화학습과 입자 군집 최적화 기법을 이용한 에피소드 매개변수 최적화 기법을 제안하였고, 이 기법을 점수행렬을 최적화하는 데 사용하여 지상 로봇팀의 생존 가능성을 높일 수 있는 최적의 교전 전략을 수립하였다.

임무지점 간 다중 로봇의 경로계획 문제를 해결하기 위해 가상의 속임수 기동 기법을 이용한 실시간 분산형 경로계획 기법을 제안하였다. 가상의 속임수 기동 기법은 곤충이 먹잇감을 쫓아갈 때 속임수 기동을 하는 것으로부터 영감을 얻어 제안된 기법으로 일반적인 비선형 구속조건을 고려한 궤적 최적화 문제를 경로 제어 매개변수만을 최적화하는 문제로 변환시켜 문제의 차원을 줄이는 역할을 한다. 여기서 경로 제어 매개변수를 최적화하기 위해 입자 군집 최적화 기법을 이용하였다. 제안된 경로계획 알고리즘은 도심환경에서의 종말 시간 및 진입 각 구속조건을 고려한 랑데부 문제를 푸는데 적용되었으며 시뮬레이션과 실험을 통해 검증하였다. 앞서 제안된 다중 로봇 임무계획 및 경로계획 알고리즘을 이용하면 복잡한 전장상황에서의 로봇의 행동양식을 쉽게 구현할 수 있다.

마지막으로 각 로봇이 상황에 맞게 적절한 행동양식을 선택할 수 있도록 하는 행동 조정 문제를 풀기 위해 분산형 다중 에이전트 강화학습 기법을 제안하였고 문제의 복잡성을 해결하기 위해 함수 근사화와 확산 적응 기법을 사용하였다. 결과적으로 본 논문에서 제안한 부분별 요소 기술을 이용하여 불확실성이 존재하는 전장상황에서 다중 로봇 그룹의 목표를 달성할 수 있었다.

주요어: 다중 로봇 시스템, 협업 아키텍처, 임무할당, 경로계획, 행동학습
학 번: 2011-30196